Приложение 6.1

Сборник практических и лабораторных работ по специальному курсу «Большие данные»

Руководитель мероприятия _____ (подпись)

Севостьянов Петр Алексеевич
профессор кафедры Автоматизированных систем обработки информации и управления,
д.т.н., профессор
тел. (495) 811-01-01 доб.1057
e-mail: sevostyanov-pa@rguk.ru

Сборник практических и лабораторных работ по специальному курсу «Большие данные».

Для учащихся 10-11 классов

Авторы:

Севостьянов П.А., доктор технических наук, профессор Российского государственного университета им. А.Н.Косыгина

Монахов В.И., кандидат технических наук, доцент Российского государственного университета им.А.Н.Косыгина

СОДЕРЖАНИЕ

ч.1 Введение в вероятностное моделирование

В данной части сборника рассмотрены базовые понятия, методы и средства теории вероятности и статистического анализа данных. Рассмотрены типовые прикладные задачи статической обработки данных и приведены методы их решения

10 класс, 1-е полугодие

Оборудование для проведения занятий

1. Рабочая станция ученика (Intel i5, 8Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)
2. Рабочая станция учителя (Intel i7, 16Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)

Программное обеспечение для проведения занятий (в том числе системное ПО)

1. ОС Windows 10
2. MS Office 2016

Рекомендации учителю по проведению занятий.

Все практические занятия включают краткие теоретические сведения по теме занятия, типовые задачи с разбором их решения, вопросы для проверки усвоения материала.

На занятиях данного цикла решение задач выполняется в форме ручных расчетов или с использованием приложения MS Excel.

В начале занятия необходимо проверить выполнение заданий для самостоятельного выполнения, выданных на предыдущем практическом занятии.

В ходе проведения занятия рекомендуется изложить учащимся краткие теоретические сведения, разобрать решение типовых задач, описанных в практических занятиях. Для усвоения материала необходимо выдать задания для самостоятельной проработки материала, разобранного на практическом занятии

Практическое занятие № 1
Вводное занятие. Что такое математическая модель?

Цель занятия. Усвоить понятие модели, научиться различать основные типы моделей, уметь привести примеры моделей

Краткие теоретические сведения

1. Понятие модели

Вначале всегда возникает задача. Суть задачи формулируется в её описании. Описание задачи состоит из двух частей. В первой части - условии задачи - описывается система или процесс. Во второй части формулируется вопрос, на который решение задачи должно дать ответ.

Для получения ответа строится модель системы или процесса. Модель - это другая система или процесс, всегда искусственная, созданная «модельером» - человеком, решающим задачу. Итак, модель - это искусственная система, которая в рамках решаемой задачи близка, подобна, практически не отличается по своему проведению от системы, описанной в задаче.

Человеческий мозг постоянно строит модели, которые он использует для выработки решений, управляющих всей жизнедеятельностью человеческого организма и поведением и поступками человека. Эти модели в большинстве случаев создаются подсознанием и являются неосознанными. Однако человек и вполне сознательно строит модели и пользуется ими на протяжении всей жизни. Модель всегда создается для решения какой-либо задачи. Например, литературное произведение есть модель событий и ситуаций, о которых автор произведения хотел сообщить читателям.

Для многих задач наиболее удобными и содержательными являются математические модели. Математическая модель – это совокупность математических соотношений: равенств, неравенств, логических условий, - связывающих переменные величины и параметры, которые описывают условие задачи.

Контрольные вопросы

Можно ли назвать моделями:

1. Роман Л.Н. Толстого "Война и мир"?
2. Картину И.Е. Репина "Царь Иван Васильевич IV у тела сына Ивана"?
3. Чебурашку из мультфильма "Чебурашка и Крокодил Гена"?
4. Игрушку "Оловянный солдатик"?
5. Смартфон?
6. Квадрокоптер?
7. Компьютер?
8. Квадратное уравнение?
9. Геометрическое тело, например, пирамида?
10. Математическое уравнение?

Дайте обоснование вашему ответу.

2. Некоторые типы моделей

Далее рассматриваем только математические модели. Приведем некоторые категории математических моделей, дуальные (противоположные) по своему типу:

А. Статическая - динамическая. Модель считается статической, если в ней отсутствует связь между значениями переменных величин в разные моменты времени. Иначе модель называют динамической. Динамическая модель – это всегда с «памятью», поскольку для связи значений переменных в разные моменты времени модель должна хранить эти значения, т.е. иметь «память». Статические модели могут и не иметь «памяти».

Б. Линейная - нелинейная. Модель линейная по некоторому параметру или переменной, если ответ задачи, решенной с помощью модели, пропорционален величине этого параметра или переменной. Иначе модель - нелинейная по этому параметру или переменной. Моделируемые системы всегда нелинейные. Однако, во многих случаях их можно приближенно заменить линейными, что упрощает получение , возможно, приближенного, решения задачи.

В. Вероятностная - детерминированная. Модель называют вероятностной, если для описания задачи приходится использовать теорию вероятностей. Иначе модель называют детерминированной. Вероятностные модели используют при недостатке информации или из-за объективно существующей вероятностной природы той системы, для которой строится модель.

Контрольные вопросы

Определите тип математической модели для описания следующих задач:

1. Определение объёма и площади поверхности пирамиды по заданным линейным размерам.

2. Определение среднего расстояния, пройденного броуновской частицей за определённый отрезок времени.

3. Задача о бассейне и двух трубах: по одной трубе бассейн заполняется, по другой трубе опорожняется.

4. Задача о передаче наследственных признаков следующим поколениям через гены.

5. Задача о желаемом билете на экзамене по математике.

Рассмотрите уточнения и детализации объекта моделирования, при которых модель изменит тип: превратится из статической в динамическую, из линейной в нелинейную, из детерминированной в вероятностную.

Порядок выполнения работы

1. Изучить теоретический материал
2 . Дать ответы на контрольные вопросы

Содержание отчета

1. Краткая теория
2. Ответы на контрольные вопросы с обоснованием

Практическое занятие № 2
Интуитивные понятия теории вероятностей

Цель занятия. Усвоить понятие невозможного, достоверного и случайного события, вероятности случайного события. Уметь приводить примеры событий из окружающего мира и определять тип этих событий

Краткие теоретические сведения

В одной и той же ситуации происходят события, которые возникают всегда (например, по утрам восходит Солнце). Другие события в этой ситуации не происходят никогда. Третьи события могут происходить, а могут и не происходить заранее не предсказуемым образом, при одних и тех же условиях.

Первый тип событий называют достоверными (для рассматриваемой ситуации). Второй тип событий называют невозможными. Третий тип событий называют случайными. Теория вероятностей и математическая статистика - это разделы математики, изучающие закономерности в мире случайных событий и методы получения информации о случайных событиях.

С помощью логических операций объединения (дизъюнкции) и пересечения (конъюнкции) можно определить события, производные от рассматриваемых событий. Объединение всех возможных в рассматриваемой ситуации исходов (событий) есть достоверное событие, которое обязательно произойдёт.

Случайные события могут появляться, а могут и не появляться при одинаковых условиях наблюдения, их появление с достоверностью не предсказуемо. Однако про случайные события можно сказать, что одни из них более ожидаемы, чем другие случайные события. Если ситуация, при которой события наблюдаются, воспроизводится многократно, то, хотя при каждом таком воспроизведении предсказать появление или непоявление события невозможно, при большом количестве воспроизведений обнаруживается, что одно случайное событие появляется чаще другого.

Следовательно, со случайным событием связана числовая мера, указывающая возможность появления этого события. Эту числовую меру называют вероятностью случайного события.

В теории вероятностей и математической статистике события принято обозначать заглавными буквами латинского алфавита A, B, C,..., а вероятность случайного события - латинской буквой p (от латинского слова probabilitas - вероятность) и другими близкими буквами: q, r, ..., - заглавными или строчными.

Для вероятности принят числовой диапазон от нуля до единицы: чем выше возможность появления случайного события, тем ближе вероятность этого события к единице, и наоборот, чем ниже возможность появления случайного события, тем ближе его вероятность к нулю.

Вероятность достоверного события принята равной единице. Вероятность невозможного события принята равной нулю. Если появление событий равновозможно, то их вероятности одинаковы.

Примечание. Если вероятность случайного события равна нулю, это не означает, что оно невозможно: просто вариантов исходов наблюдения за ситуацией (случайных событий) бесконечно много. Тогда, чтобы суммарная вероятность объединения всех событий была равна единице, приходится считать, что вероятность каждого исхода равна нулю. Этот результат получил название «парадокса нулевой вероятности».

Пример: выбираем случайным образом точку на отрезке [0;1]. Общая вероятность выбрать хоть какую-нибудь точку на отрезке равна единице. Вероятность выбрать любую конкретную точку равна нулю.

Задание

1. Приведите примеры детерминированных и случайных событий. Обоснуйте ответ.
2. Приведите примеры достоверных, невозможных и случайных событий. Обоснуйте ответ.
3. Из журнала записей погоды по дням в течение месяца определите, какое событие более вероятно: дни с осадками (дождь или снег) или дни без осадков?
4. Из журнала записей погоды по дням в течение месяца определите, какое событие более вероятно: день без осадков после дня с осадками, день с осадками после дня с осадками или день без осадков после дня с осадками?
5. Последовательность значений стоимости одного доллара в рублях (курс доллара) по дням – это разные значения одной и той же случайной величины или значения последовательности случайных величин (по случайной величине на каждый день) ?

Порядок выполнения работы

1. Изучить теоретический материал
2. Дать ответ на вопросы, сформулированные в задании и обосновать свой ответ

Содержание отчета

1. Краткая теория
2. Ответы на задание

Практическое занятие № 3
Исчисление вероятностей и элементы комбинаторики

Цель занятия. Научиться вычислять вероятность случайного события. Усвоить основные понятия комбинаторики. Научиться решать задачи комбинаторики, связанные с выбором и расположением элементов множества

Краткие теоретические сведения

В задачах с конечным числом равновозможных исходов вероятность случайного события определяется отношением числа исходов, при которых это случайное событие происходит, к общему числу равновозможных исходов.

Рассмотрим типовые задачи на вычисление вероятности случайного события в двух модельных ситуациях.

I. Модель ситуации с бесконечно большим числом равновозможных исходов.

На волчке (юле) поставлена метка. По окончании вращения волчка метка может расположиться равновозможно под любым углом по отношению к начальному положению.

1. Какова вероятность того, что этот угол будет: а) от нуля до 90 градусов? б) от 145 до 235 градусов? в) от 30 до -60 градусов?
2. Какова вероятность того, что этот угол будет лежать в первой или в третьей четверти круга?
3. Ось вращения волчка находится в центре круга. Круг разделён на 90 одинаковых красных и чёрных сегментов. а) какова вероятность того, что метка попадет в один из красных сегментов? б) метка попала во вторую четверть круга. Чему равна вероятность попадания её в чёрный сегмент?
4. Чему равна вероятность того, что метка будет под углом: а) 37 градусов? б) 90 градусов? в) 360 градусов? г) минус 32 градуса?
5. Чему равна вероятность того, что метка на волчке расположится под каким-нибудь любым углом относительно исходного положения?

Примечание к задачам. Мерой всех возможных исходов следует принять меру множества значений угла, определяющего положение волчка. Этим множеством является полуинтервал [0 ; 360 градусов). Вероятность случайного события, которую нужно найти в задаче, равна отношению меры множества положений волчка, при которых это событие происходит, к общей мере всех возможных исходов. Например, для задачи 1а ответ вероятность будет равна $(90 – 0) / (360 – 0) = 0{,}25$.

II. **Модель ситуации с конечным числом равновозможных исходов**.

Игральным кубиком называют тело идеальной кубической формы из однородного материала, т.е. куб со всеми прямыми углами, равными

квадратными гранями, равными ребрами и одинаковой плотностью массы материала по всему объему куба. На гранях игрального кубика нанесены номера граней от единицы до шести.

Бросают на стол два игральных кубика. Опишите и перечислите равновозможные исходы ситуации после бросания кубиков. Вероятность любого события в этой ситуации равна

p(событие) = {число исходов, при которых событие появилось} / {общее число равновозможных исходов}

1. Чему равно число равновозможных исходов?
2. Чему равна вероятность выпадения четной суммы очков?
3. Чему равна вероятность суммы очков, больше шести?
4. Чему равна вероятность суммы очков, равной шести?
5. Чему равна вероятность четного числа очков на одном кубике и нечетного числа очков на втором кубике?

Рассмотрим теперь типовые задачи комбинаторики, связанные с выбором и расположением элементов множества

Комбинаторика. Число перестановок

Задача: имеется n предметов и n ячеек, в каждой из которых разместить любой один из предметов. Сколькими разными способами можно разместить n предметов по n ячейкам?

Решение: один предмет можно разместить в любой из n ячеек. При этом остаются свободными остальные n-1 ячейки. Следовательно, при фиксированном положении первого предмета второй предмет можно разместить n-1 разными способами. Всего для двух предметов получаем n*(n-1) разных способов размещения. Продолжим эти рассуждения для третьего и последующих предметов.

В итоге получим общее число разных способов заполнения n ячеек n предметами:

$N = n*(n-1)*(n-2)*\ldots 3*2*1 = n!$ (читается н-факториал)

перестановок n предметов по n ячейкам.

Комбинаторика. Число сочетаний

Задача: из n разных предметов отбирают k предметов, $k \leq n$. Сколько таких сочетаний предметов можно отобрать, отличающихся хотя бы одним предметом?

Решение: один предмет из множества n предметов можно отобрать n способами. Второй предмет можно отобрать из оставшихся (n-1)-го предмета n-1 способами. Всего для двух предметов получим n*(n-1) способ отбора. В этих отборах повторяются отборы с одинаковыми предметами: первым и вторым, но в разном порядке.

По условию задачи, порядок отбора не имеет значения. Каждая отобранная группа должна отличаться от другой хотя бы одним элементом.

Поэтому число таких групп по два предмета будет меньше в число перестановок из двух предметов, т.е. n*(n-1)/2!.

Продолжая подсчет, для k предметов найдем число способов отбора отличающихся по составу групп из общего числа n предметов равным

C(n,k) = n*(n-1)*(n-2)*…*(n-k+2)*(n-k+1) / k! = n! / (k! * (n-k)!)

Это число называют числом сочетаний из n по k предметов.

Комбинаторика. Число размещений

Задача: из n разных предметов отбирают k предметов, $k \leq n$. Сколько таких сочетаний предметов можно отобрать, отличающихся хотя бы одним предметом порядком их отбора?

Решение: с учетом решения для числа сочетаний учтем, что важны не только различия в предметах среди отобранных k предметов из их общего числа n, но и порядок их отбора. Тогда число таких способов отбора равно

A(n,k) = n*(n-1)*(n-2)*…*(n-k+2)*(n-k+1) = n! / k!

Задача.

Имеется множество из пяти разных предметов. Обозначим предметы буквами A, B, C, D, E. Для этого множества число перестановок равно 5! = 1*2*3*4*5 = 120. Приведем некоторые из возможных перестановок

ABCDE BACDE BCADE BCDAE BCDEA ACBDE ACDBE ACDEB … EDCBA

Выполним отбор 3 предметов из пяти. Число отборов, различающихся предметами, равно C(n=5,k=3) = 5! / (3!*2!) = 4*5 / (1*2) = 10/ Вот они:

ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE, CDE

Теперь отберем группы, содержащие два предмета из пяти, отличающихся не только предметами, но и порядком их следования в отборе. Всего таких разных вариантов отбора будет A(n=5,k=2) = 5! / 3! = 20

AB AC AD AE BC BD BE CD CE DE
BA CA DA EA CB DB EB DC EC ED

Задание.

Решить задачи:

1. В ящике 10 одинаковых деталей, пронумерованных номерами от 1 до 10. Из ящика наугад вынимают шесть деталей. Найти вероятность того, что среди отобранных деталей будет деталь номер 3?

Ответ: P = C(9,5) / C(10,6) = 0,6.

2. Для условия задачи 1 найти вероятность того, что среди отобранных будут детали 3 и 7?

Ответ: P = C(8,4) / C(10,6) = 1/3.

3. Игральный кубик бросают два раза. Найти вероятность того, что сумма очков будет больше 10?

Ответ: P = 3 / 36 = 1/12.

4. Круг вращения волчка разделен на 24 одинаковых сектора, которые пронумерованы от 1 до 24. Найти вероятность того, что метка волчка укажет на сектор с номером, кратным трем?

Ответ: $P = 8 / 24 = 1/3$.

5. Пароль состоит из двух строчных букв русского алфавита и трех десятичных цифр. Найти вероятность того, что случайно набранная комбинация будет правильной?

Ответ: $P = 1 / (33*33*10*10*10)$.

6. Шарик диаметра d бросают наугад между параллельными прутьями толщиной b и промежутком между прутьями a > d. Найти вероятность того, что шарик пролетит между прутьями, не коснувшись их.

Ответ: $P = (a - d) / (a + b)$

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить все задачи задания

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 4
Условная и полная вероятность

Цель занятия. Усвоить понятие условной и полной вероятностей случайного события. Научиться вычислять условную и полную вероятности

Краткие теоретические сведения

1. Условная вероятность случайного события

Условная вероятность случайного события – это вероятность, вычисленная с учетом некоторого дополнительного условия.

Если случайное событие не зависит от этого дополнительного условия, то его вероятность останется неизменной. В противоположном случае вероятность события будет отличаться от вероятности события без учета условия.

В качестве дополнительного условия может быть как детерминированное ограничение, так и другое случайное событие, которое может произойти, а может и не произойти.

Соответственно, условная вероятность случайного события – вообще говоря, разная величина, в зависимости от того, произошло или нет событие – условие. Условная вероятность события A при условии, что произошло событие B, записывается $P(A \mid B)$.

Замечание: если случайное событие A зависит от случайного события B, то и **обязательно** случайное событие B зависит от случайного события A. Эта зависимость проявляется в том, что вероятность появления события A зависит от того, произошло или не событие B.

Формула умножения вероятностей двух случайных событий A и B

$$P(A \text{ и } B) = P(A \mid B) \cdot P(B)$$

В силу симметрии

$$P(A \text{ и } B) = P(B \mid A) \cdot P(A)$$

Если события A и B независимы, то

$$P(A \mid B) = P(A) \text{ и } P(B \mid A) = P(B),$$

т.е. в этом случае $P(A \text{ и } B) = P(B) \cdot P(A)$

Задание. Решить задачи

1. На отрезке числовой оси $[0 ; 1]$ выбирают наугад точку. Найти вероятности того, что:

а) точка находится на интервале $(0{,}1 ; 0{,}3)$,

б) точка находится на интервале $(0{,}1 ; 0{,}3)$, если известно, что точка взята из половины отрезка $[0 ; 0{,}5]$,

в) точка находится на интервале (0,1 ; 0,3), если известно, что точка взята из половины отрезка [0,5 ; 1].

2. Мишень для дротиков (darts) – десять концентрических кругов. Радиус внутреннего круга – 10 см. Каждый следующий круг имеет радиус на 5 см больше предыдущего. Попадание в центральный круг приносит 10 очков, в следующий круг – 9 очков, и т.д. до одного очка. Брошенный дротик гарантированно попадает в мишень. Найти вероятность того, что:

а) брошенный дротик даст не менее 5 очков;

б) брошенный дротик даст не менее 5 очков, если известно, что он попал в правую верхнюю четверть мишени;

в) брошенный дротик даст не менее 9 очков, если известно, что он попал в пределы пятого круга. (Замечание: вероятность пропорциональна площади мишени).

3. Пароль состоит из двух строчных букв русского алфавита и трех десятичных цифр. Найти вероятность того, что случайно набранная комбинация будет правильной, если буквы известны и известно, что они занимают первые две позиции пароля?

2. Полная вероятность случайного события

В ряде задач можно выделить группу событий { H1, H2, …, Hm }, которые, во-первых, не могут произойти вместе, т.е. это множество несовместных событий, а во-вторых, какое-то из них обязательно происходит, т.е. объединение этих событий является достоверным событием.

При этих условиях группу событий { H1, H2, …, Hm } называют **полной группой событий**. Формально свойства полной группы записываются формулами

P (Нк и Hn) =0, k, n = 1,…,m

Сумма (P (H к), к = 1,…,m) = 1

Рассмотрим случайное событие A, которое связано с событиями { H1, H2, …, Hm }, причем известны вероятности этих событий и условные вероятности для события A в зависимости от того, какое из событий { H1, H2, …, Hm } произошло.

В этом случае вероятность появления события A можно выразить через вероятности появления событий полной группы событий { H1, H2, …, Hm } и условные вероятности события A при условии, что произошло одно из событий полной группы

$$P(A) = P(A \mid H1) \cdot P(H1) + ... + P(A \mid Hm) \cdot P(Hm) =$$

$$= \sum_{k=1}^{m} P(A \mid Hk) \cdot P(Hk)$$

Задание. Решить задачи

1. Изделие собирают из комплектующих, которые поступают с трех заводов. Доли комплектующих в изделии от каждого завода, соответственно, равны 0,5; 0,3; 0,2. Доля бракованных комплектующих среди поступивших, равна (по заводам) 0,01; 0,005; 0,015. Найти вероятность того, что собранное изделие будет бракованным, если в годном изделии бракованные изделия не допускаются.

Ответ: P (A) = 0,5*0,01 + 0,3*0,005 + 0,2*0,015 = 0,095.

2. Лес можно разделить на три участка с вероятностью грибов на каждом участке 0,4; 0,5; 0,1. Для грибника этот лес – незнакомый. В зависимости от участка вероятность найти гриб у грибника 0,8; 0,5; 0,4. Найти вероятность того, что грибник найдет грибы.

Ответ: P (A) = 0,4*0,8 + 0,5*0,5 + 0,1*0,4 = 0,61.

3. Программист при написании программы делает ошибки. Из них синтаксические ошибки составляют долю 0,3, семантические ошибки составляют долю 0,55 и семиотические ошибки – 0,15. Компилятор не обнаруживает синтаксические ошибки с вероятностью 0,05, семантические ошибки с вероятностью 0,8 и семиотические ошибки с вероятностью 0,9. Найти вероятность того, что написанная программистом программа после компиляции будет содержать ошибки

Ответ: P (A) = 0,3*0,05 + 0,55*0,8 + 0,15*0,9 = 0,59.

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить все задачи по заданиям.

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 5
Понятие случайной величины

Цель занятия. Усвоить понятие случайной величины, функции распределения, функции плотности распределения случайной величины. Научиться вычислять функцию распределения для дискретной и непрерывной случайных величин, вычислять основные характеристики случайных величин

Краткие теоретические сведения

При решении вероятностных задач для вычисления вероятностей случайных событий удобно пользоваться действительными переменными величинами, которые называют случайными величинами.

Переменная величина X, принимающая значения из некоторого подмножества действительных чисел $\{ x \}$, называется **случайной величиной**, если значения, которые она принимает, заранее не предсказуемы с достоверностью. Следовательно, например, исходы $X = x$, или $X < x$, или $x1 < X < x2$ являются случайными событиями.

Любому случайному событию A можно поставить в соответствие случайную величину IndA – **индикатор события**: IndA = 1, если A произошло, и IndA = 0, если A не произошло.

Отдельные возможные значения x случайной величины X называют ее **реализациями**. Множество всех возможных реализаций $\{ x \}$ называют **генеральной совокупностью**. Если генеральная совокупность – конечное или счетное множество, то случайную величину называют дискретной.

Функцией распределения (интегральной функцией распределения) $F_X(x)$ случайной величины X называют функцию

$$F_X(x) = P(X \le x), \quad -\infty < x < +\infty$$

Функция распределения определена на всей числовой оси, т.е. для всех действительных значений своего аргумента x.

Свойства функции распределения следуют из ее определения и свойств вероятностей невозможных, достоверных и случайных событий

$$F_X(-\infty) = 0, \qquad F_X(+\infty) = 1, \qquad P(x_1 < X \le x_2) = F_X(x_2) - F_X(x_1)$$

Таким образом, функция распределения – всегда монотонно неубывающая функция, определенная на всей числовой оси, - и с областью изменения на отрезке от нуля до единицы.

Задачи

1. Построить функцию распределения для числа очков, выпадающих при одном бросании игрального кубика. (Ввести случайную величину X – число очков при одном бросании кубика).

Ответ: значения случайной величины и их вероятности

x	1	2	3	4	5	6
$P(\mathrm{X} = x)$	1/6	1/6	1/6	1/6	1/6	1/6

Функция распределения случайной величины X

x	1	2	3	4	5	6
$F_{\mathrm{X}}(x)$	1/6	2/6	3/6	4/6	5/6	6/6 = 1

На рис.1 показан график функции распределения $F_{\mathrm{X}}(x)$.

2. Построить функцию распределения для координаты точки, бросаемой наугад на отрезок [0 ; 1].

Ответ: при бросании точки наугад на отрезок вероятность случайного события, что точка попадет левее некоторой точки x, пропорциональна длине части отрезка левее брошенной точки. Введем случайную величину X – координату брошенной на отрезок точки. Тогда вероятность события $X \le x$ равна отношению длин $(x - 0) / (1 - 0) = x$. Следовательно

$$F_X(x) = \begin{cases} 0, & \text{при } x < 0 \\ x, & \text{при } 0 < x < 1 \\ 1, & \text{при } x > 1 \end{cases}$$

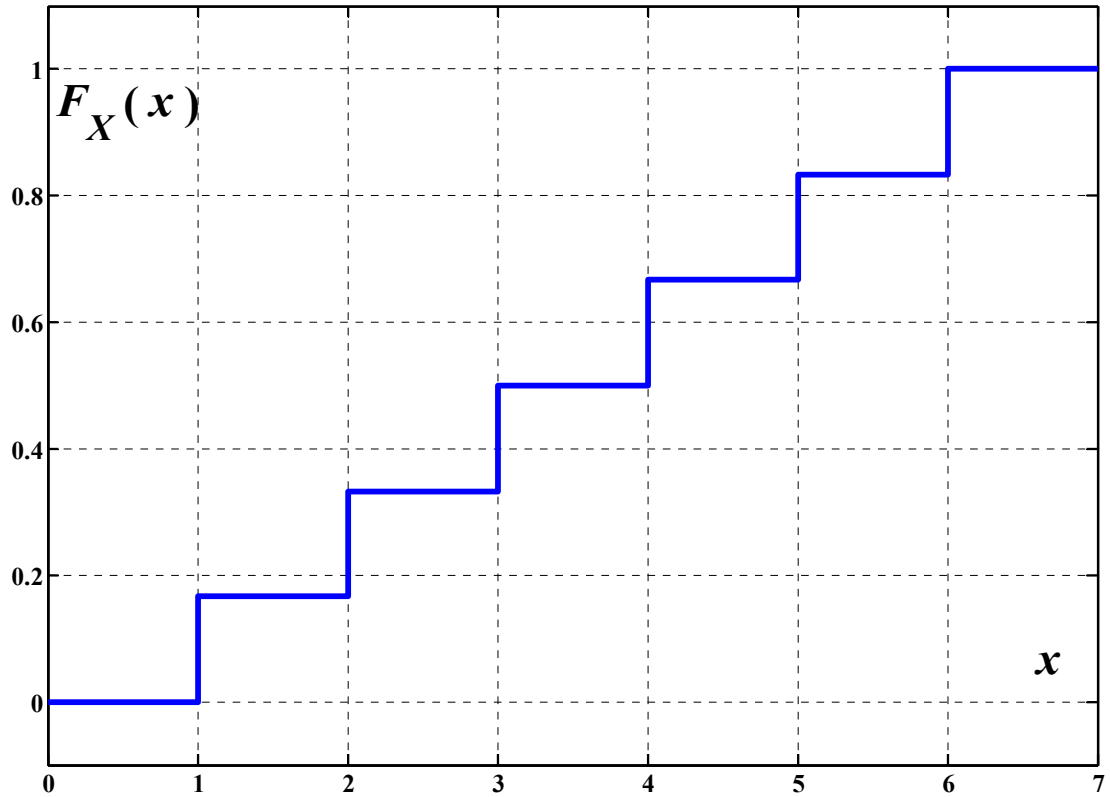


Рис.1. Функция распределения числа очков при одном бросании игрального кубика

Если функция распределения случайной величины – дифференцируемая функция, то ее производную называют **функцией плотности** распределения вероятности (дифференциальной функцией распределения)

$$f_X(x) = \frac{dF_X(x)}{dx}$$

Свойства функции плотности распределения вероятностей

$$f_X(x) \geq 0, \ -\infty < x < +\infty, \qquad F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt$$

$$\int_{-\infty}^{+\infty} f_X(t)\,dt = 1, \qquad P\left(x_1 < X \leq x_2\right) = \int_{x_1}^{x_2} f_X(t)\,dt$$

Задачи

1. Найти функцию плотности распределения вероятности для функции распределения, найденной в задаче 2 (см. выше).

Ответ: в соответствии с определением

$$f_X(x) = \begin{cases} 0, & \text{при} \quad x < 0 \\ 1, & \text{при} \quad 0 < x < 1 \\ 0, & \text{при} \quad x > 1 \end{cases}$$

Построить графики функции распределения и функции плотности распределения вероятности.

2. Может ли функция $f_X(x) = C\exp(-ax)$, $x > 0$, и $f_X(x) = 0$ при $x < 0$, быть функцией плотности распределения?

Ответ: да, поскольку она при всех возможных значениях x неотрицательна, и при $C = a$ интеграл от этой функции по всей числовой оси равен единице

$$\int_{-\infty}^{+\infty} f_X(x)\,dx = C \cdot \int_{0}^{+\infty} \exp(-ax)\,dx = C / a = 1 \rightarrow C = a$$

$$f_X(x) = a \cdot \exp(-ax), \quad 0 \leq x < +\infty$$

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt = 1 - \exp(-ax), \quad 0 \leq x < +\infty$$

Такое распределение вероятностей случайной величины называется экспоненциальным, или показательным.

Середину распределения значений случайной величины характеризуют числом, которое называют **математическим ожиданием** случайной

величины MX и которое равно для дискретной и непрерывной случайной величины соответственно

$$MX = \sum x \cdot P(x)$$

$$MX = \int_{-\infty}^{+\infty} x \cdot f(x)\ dx$$

Числовыми **мерами рассеяния** значений случайной величины относительно ее математического ожидания являются дисперсия случайной величины *DX*, среднее квадратическое отклонение случайной величины *SigmaX* и и коэффициент вариации случайной величины *CVX*, которые для дискретной и непрерывной случайной величины определяются формулами

$$DX = \sum_{x} \left(x - MX\right)^2 \cdot p(x)$$

$$DX = \int_{-\infty}^{+\infty} \left(x - MX\right)^2 \cdot f(x)\,dx$$

$$SigmaX = \sqrt{DX} \qquad CVX = \frac{SigmaX}{MX} \cdot 100\%$$

Задание. Решить задачи

1. Найти математическое ожидание числа очков, выпадающих при бросании игрального кубика.

Ответ:

MX = 1·1/6 + 2·1/6 + 3·1/6 + 4·1/6 + 5·1/6 + 6·1/6 = 21/6 = 3,5

2. Найти математическое ожидание координаты точки, брошенной наудачу на отрезок от 4 до 12.

Ответ:

$$MX = \int_{-\infty}^{+\infty} x \cdot f(x)\,dx = \frac{1}{8} \cdot \int_{4}^{12} x\,dx = \frac{144 - 16}{16} = 8$$

3. Найти математическое ожидание случайной величины, распределенной по экспоненциальному закону

$$MX = a \cdot \int_{0}^{+\infty} x \cdot \exp(-a \cdot x)\,dx = \frac{1}{a}$$

4. Найти дисперсию, среднее квадратическое отклонение и коэффициент вариации числа очков, выпадающих при бросании игрального кубика.

Ответ:

DX = (1 – 3,5)^2·1/6 + (2 – 3,5)^2·1/6 + (3 – 3,5)^2·1/6 +

$+ (4 - 3{,}5)^2 \cdot 1/6 + (5 - 3{,}5)^2 \cdot 1/6 + (6 - 3{,}5)^2 \cdot 1/6 = 17{,}5/6 = 2{,}9167$

$SigmaX = 1{,}7078$

$CVX = 1{,}7078 / 3{,}5 \cdot 100\% = 48{,}795\%$

5. Найти дисперсию, среднее квадратическое отклонение и коэффициент вариации координаты точки, брошенной наудачу на отрезок от 4 до 12.
   Ответ:

$$DX = \int_{-\infty}^{+\infty} (x - MX)^2 \cdot f(x)\, dx = \frac{1}{8} \int_{4}^{12} (x - MX)^2\, dx = \frac{(12-4)^2}{12} = 5{,}3333$$

$$SigmaX = 2{,}3094 \qquad CVX = 28{,}8675\%$$

6. Найти дисперсию, среднее квадратическое отклонение и коэффициент вариации случайной величины, распределенной по экспоненциальному закону с параметром a.
   Ответ:

$$DX = \int_{0}^{+\infty} (x - MX)^2 \cdot a \cdot \exp(-a \cdot x)\, dx = \frac{1}{a^2}$$

$$SigmaX = \frac{1}{a} \qquad CVX = 100\%$$

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить все задачи задания.

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 6
Обработка результатов наблюдений. Понятие статистической оценки

Цель занятия. Усвоить понятие выборки случайной величины, способы оценивания свойств случайной величины по выборке, требования к оценкам, различие между точными и интервальными оценками. Научиться вычислять статистические оценки характеристик случайной величины и оценки функции распределения

Краткие теоретические сведения

Наблюдения за некоторой системой или процессом регистрируют в виде последовательности числовых значений некоторой переменной X. Если условия наблюдений неизменны, то значения x переменной X будут либо одними и теми же, либо отличаться случайным образом.

В первом случае переменную X можно считать детерминированной. Разным условиям наблюдения отвечают разные значения детерминированной переменной, одним и тем же условиям – одни и те же значения детерминированной переменной. В этом случае достаточно получить одно значение переменной для каждого условия наблюдения.

Во втором случае переменная X является случайной. Требуются повторные наблюдения этой переменной при одних и тех же условиях, чтобы получить набор ее разных значений. Встает задача об извлечении информации из этого набора значений о свойствах случайной величины X. Этот набор n значений $\mathbf{X} = \{ x(1), x(2),\ldots, x(n) \}$ называют выборкой объема n. Если все элементы выборки получены при одинаковых условиях наблюдений и независимо друг от друга, то выборку называют простой случайной выборкой.

Полная информация обо всех свойствах случайной величины X содержится в ее функции распределения. Выборка всегда содержит меньше информации, чем функция распределения. Поэтому сведения, извлекаемые из выборки любого объема, всегда являются приближенными и неполными. Это принципиальная особенность любой обработки любых выборочных данных.

С помощью обработки выборочных результатов наблюдений можно получить лишь оценку некоторой характеристики случайной величины. Эти оценки называют статистическими, а науку об извлечении информации о случайных величинах из выборочных данных называют математической статистикой.

Любая оценка – это некоторое функциональное преобразование выборочных данных $a = g(\mathbf{X})$. Например, оценка математического ожидания случайной величины – это среднее арифметическое элементов выборки. Записывается это следующим образом

$$MX^* = g(\mathbf{X}) = \frac{x(1) + x(2) + \ldots + x(n)}{n} = \bar{x}$$

Оценкой дисперсии случайной величины также является функция выборки, которая имеет вид

$$DX^* = \frac{(x(1) - \bar{x})^2 + (x(2) - \bar{x})^2 + \ldots + (x(n) - \bar{x})^2}{n-1}$$

Оценки среднеквадратического отклонения и коэффициента вариации случайной величины равны

$$SX^* = \sqrt{DX^*} \qquad CVX^* = \frac{SX^*}{\bar{x}} \cdot 100\%$$

Задача

Выполнено 20 наблюдений за положением указателя юлы после завершения ее вращения на плоскости. В первой четверти плоскости указатель оказался шесть раз, во второй четверти – три раза, в третьей четверти пять раз, в четвертой четверти – шесть раз. Найти оценки математического ожидания, дисперсии, среднеквадратического отклонения и коэффициента вариации случайной величины – номера четверти, в которой остановился указатель юлы.

Решение. Случайная величина X принимает значения 1, 2, 3, 4. При наблюдении эти значения появились соответственно 6, 3, 5 и 6 раз. Вычислим оценки, используя приведенные выше формулы:

$$MX^* = \bar{x} = \frac{6 \cdot 1 + 3 \cdot 2 + 5 \cdot 3 + 6 \cdot 4}{20} = 2{,}55$$

$$DX^* = \frac{6 \cdot (1 - 2{,}55)^2 + 3 \cdot (2 - 2{,}55)^2 + 5 \cdot (3 - 2{,}55)^2 + 6 \cdot (4 - 2{,}55)^2}{20} = 0{,}2637$$

$$SX^* = \sqrt{DX^*} = 0{,}5135 \qquad CVX^* = \frac{SX^*}{\bar{x}} \cdot 100\% = 20{,}1373\%$$

При повторении серии из n наблюдений за случайной величиной при тех же самых условиях получают другую выборку и, соответственно, другие числовые значения по тем же самым формулам оценивания. Это значит, что оценки, в отличие от оцениваемых характеристик случайной величины, сами являются случайными величинами, вернее, значениями некоторых других случайных величин, полученных в результате функционального преобразования наблюдаемой случайной величины.

Формулы для оценивания характеристики случайной величины можно получать различными способами. Предпочтение отдают оценкам, отвечающим трем основным требованиям:

1. **Состоятельность** оценки означает, что с увеличением объема выборки точность оценивания увеличивается (т.е. дисперсия оценки уменьшается).

2. **Несмещенность** оценки означает, что при увеличении количества серий из n наблюдений средняя величина оценок некоторой числовой характеристики случайной величины сколь угодно мало отличается от точного значения этой характеристики.

3. **Эффективность** оценки означает, что из всех возможных состоятельных и несмещенных оценок числовой характеристики случайной величины выбрана наиболее точная оценка, т.е. имеющая наименьшую дисперсию.

В математической статистике доказано, что приведенные выше формулы для оценивания математического ожидания и дисперсии дают состоятельные, несмещенные и эффективные оценки этих характеристик случайных величин.

Статистическую оценку функции распределения случайной величины по результатам наблюдений (по выборке объема n) получают следующим образом:

1. Элементы выборки упорядочивают по возрастанию и, соответственно, перенумеровывают: $x_1 < x_2 < \ldots < x_n$. Полученная последовательность называется вариационным рядом.

2. Строят таблицу из трех строк. В первой строке записывают значения вариационного ряда. Во второй строке записывают значения *1* / *n*. Если какие-либо элементы ряда равны, например, три элемента $x_k = x_{k+1} = x_{k+2}$, то в *k*-м столбце второй строки помещают 3 / *n*. Проверяют, что сумма всех чисел во второй строке равна единице.

3. Вычисляют и записывают в третьей строке накопленную сумму всех чисел второй строки с первого по *k*-й столбец. Значения в третьей строке и есть оценка $F(x_k)$ значений функции распределения для значений ее аргумента x_k.

4. Если наблюдаемая случайная величина – непрерывная, то по парам значений (x_k , $F(x_k)$) строят график в виде непрерывной линии - ломаной или гладкой, которая и является оценкой функции распределения этой случайной величины. Если случайная величина – дискретная, то по этим парам значений строят ломаную кривую, наподобие кривой на рис.1, которая и является оценкой функции распределения.

Наряду с приведенными выше «точечными» оценками характеристик случайных величин, которые дают всего одно значение оценки («точку») применяют интервальные оценки, которые объединяют в себе саму оценку в виде интервала значений, показатели точности и надежности оценивания.

Эти оценки записывают в виде доверительного утверждения

$$p\{C_1 < A < C_2)\} = P_{\text{доверит.}}$$

В этой записи A – истинное неизвестное значение оцениваемой числовой характеристики; C_1 и C_2 – нижняя и верхняя границы доверительного интервала; $P_{\text{доверит.}}$ – доверительная вероятность оценки.

Границы C_1 и C_2 вычисляются по значениям, содержащимся в выборке и с учетом принятой доверительной вероятности. Граничные значения зависят от конкретной выборки и являются случайными. Доверительная вероятность есть показатель надежности оценивания: чем ближе она к единице, тем выше надежность полученной интервальной оценки. Обычно ее выбирают равной: в технике 0,95 или 0,99; в медицине и химии 0,99 и 0,995; в экономике и социальных науках 0,6 и 0,8.

Читается доверительное утверждение следующим образом: случайный доверительный интервал (C_1 ; C_2) содержит в себе («накрывает») истинное значение оцениваемой характеристики A с надежностью $P_{\text{доверит.}}$ и точностью, равной $d = C_2 - C_1$.

Вопросы для текущего контроля

1. Чем оценка характеристики случайной величины отличается от самой характеристики случайной величины?
2. Можно ли, обрабатывая выборочные данные, получить точное значение характеристики случайной величины?
3. Что такое объем выборки?
4. Какую выборку называют простой случайной выборкой?
5. Каким требованиям должны отвечать оценки характеристик случайных величин?
6. Что такое вариационный ряд и как его можно получить из выборочных данных?
7. В чем отличие графиков для оценок функции распределения дискретной и непрерывной случайных величин?
8. В чем отличие точечных и интервальных оценок числовых характеристик случайных величин?
9. Какая величина характеризует точность интервальной оценки?
10. Какая величина задает надежность интервальной оценки?

Порядок выполнения работы

1. Изучить теоретический материал
2. Дать ответы на вопросы

Содержание отчета

1. Краткая теория
2. Ответы на вопросы

Практическое занятие № 7
Числовые оценки выборочных характеристик

Цель занятия. Научиться вычислять оценки положения и рассеяния выборочных характеристик случайной величины

Краткие теоретические сведения

Приведем формулы для получения точечных оценок наиболее востребованных числовых характеристик случайных величин. Оценки вычисляются как функции простой случайной выборки **X** = { $x(1), x(2),\ldots, x(n)$ } объема n.

Числовые характеристики середины расположения выборочных данных:

Оценки математического ожидания и медианы

$$MX^* = \frac{1}{n}\sum_{k=1}^{n} x(k) = \bar{x}$$

$$MeX^* = \begin{cases} x_{k=(n+1)/2}, & \text{для нечетного } n \\ 0.5\cdot\left(x_{k=n/2} + x_{k=n/2+1}\right), & \text{для четного } n \end{cases}$$

В формуле для оценки медианы x_k – элементы вариационного ряда, построенного по выборке.

Медиана является характеристикой середины распределения значений случайной величины – альтернативой математическому ожиданию: она является робастной характеристикой, т.е. меньше зависит от особенностей конкретной выборки и грубых ошибок в наблюдениях по сравнению с оценкой математического ожидания.

Числовые характеристики рассеяния значений случайной величины:

Оценки дисперсии DX, среднего абсолютного отклонения SAX, среднеквадратического отклонения DX, коэффициента вариации CVX, размаха WX

$$DX^* = \frac{1}{n-1}\sum_{k=1}^{n}\left(x(k)-\bar{x}\right)^2 \qquad SAX^* = \frac{1}{n}\sum_{k=1}^{n}\left|x(k) - MeX^*\right|$$

$$SX^* = \sqrt{DX^*} \qquad CVX^* = \frac{SX^*}{\bar{x}}\cdot 100\%$$

$$WX^* = \max\{x(k), k=1,\ldots,n\} - \min\{x(k), k=1,\ldots,n\}$$

Многообразие числовых характеристик рассеяния связано с их особенностями, достоинствами и недостатками. Например, дисперсия имеет размерность, равную квадрату размерности случайной величины. Поэтому среднеквадратическое отклонение является более предпочтительным, поскольку его размерность та же, что и у случайной величины. Коэффициент

вариации оценивает относительную вариацию значений случайной величины и вообще не зависит ни от единицы измерения, ни от масштаба измеряемой случайной величины. Размах легко оценивается, но весьма чувствителен к погрешностям измерений.

Пример

Выполнено 100 наблюдений за бросанием игрального кубика. количество появлений каждой грани приведено в таблице. Найти оценки математического ожидания, медианы, дисперсии, среднего абсолютного отклонения, среднеквадратического отклонения, коэффициента вариации и размаха случайной величины – числа очков, выпадающих при бросании кубика

x	**1**	**2**	**3**	**4**	**5**	**6**
nk	**14**	**17**	**18**	**15**	**19**	**17**

Решение:

$$MX^* = \frac{14\cdot 1 + 17\cdot 2 + 18\cdot 3 + 15\cdot 4 + 19\cdot 5 + 17\cdot 6}{100} = 3{,}5900$$

$$MeX = 0.5\cdot(x_{50} + x_{51}) = 0.5\cdot(3+3) = 3$$

$$DX^* = \frac{1}{99}\left(14\cdot(1-3{,}59)^2 + 17\cdot(2-3{,}59)^2 + \ldots + 17\cdot(6-3{,}59)^2\right) = 2{,}8504$$

$$SAX^* = \frac{1}{100}\cdot\left(14\cdot|1-3| + 17\cdot|2-3| + 18\cdot|3-3| + 15\cdot|4-3| + 19\cdot|5-3| + 17\cdot|6-3|\right) =$$

$$= 1{,}4782$$

$$SX^* = \sqrt{2{,}8504} = 1{,}6883 \qquad CVX^* = \frac{1{,}6883}{3{,}5900}\cdot 100\% = 47{,}0282\%$$

$$WX^* = 6 - 1 = 5$$

Задание. Решить задачи

1. Регистрировали число документов *X*, поступающих на компьютер для обработки в течение часа. Это число колебалось в диапазоне от 5 до 10 документов, причем количество случаев для каждого из значений приведено в таблице

X	**5**	**8**	**7**	**8**	**9**	**10**
nk	**28**	**21**	**14**	**10**	**6**	**3**

Найти оценки математического ожидания, медианы, дисперсии, среднего абсолютного отклонения, среднеквадратического отклонения, коэффициента вариации и размаха случайной величины – числа документов, поступающих на обработку в компьютер в течение часа.

2. Двадцать измерений диаметра вала дали выборку значений

2,067	**1,879**	**2,072**	**2,163**	**2,049**	**2,104**	**2,073**	**1,970**	**2,029**	**1,921**
2,088	**1,885**	**1,893**	**1,919**	**1,705**	**2,144**	**2,033**	**1,924**	**2,137**	**1,829**

Найти оценки математического ожидания, медианы, дисперсии, среднего абсолютного отклонения, среднеквадратического отклонения, коэффициента вариации и размаха случайной величины – диаметра вала.

3. Измерялись интервалы времени между звонками на телефон. Получено 8 значений, в минутах: 36, 84, 17, 25, 73, 55, 8, 18. Найти оценки математического ожидания, медианы, дисперсии, среднего абсолютного отклонения, среднеквадратического отклонения, коэффициента вариации и размаха случайной величины – интервала времени между звонками на телефон.

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить все задачи задания.

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 8
Вероятностные модели случайной величины

Цель занятия. Усвоить понятие модели случайной величины, основные типы распределений случайной величины. Научиться вычислять характеристики основных типов распределений. Научиться решать практические задачи для основных вероятностных схем

Краткие теоретические сведения

Во многих вероятностных задачах распределения случайных величин близки к некоторым, наиболее употребительным и распространенным распределениям, которые и используются в качестве моделей для этих случайных величин. По этой причине целесообразно познакомиться с этими «модельными» распределениями.

Равномерное распределение непрерывной случайной величины. Диапазон значений случайной величины – интервал от A до B: $A < X < B$. Функция плотности вероятности – постоянная величина C в этом диапазоне. Поскольку площадь под графиком этой функции должна быть равна единице (условие нормировки функции плотности вероятности), то константа $C = 1/(B - A)$. Математическое ожидание и медиана равны $MX = MeX = (A + B)/2$, дисперсия равна $DX = (B - A)^2 / 12$. Размах равен $WX = B - A$.

Экспоненциальное распределение непрерывной случайной величины. Диапазон значений случайной величины – полуинтервал $[\, 0 < X < +\infty \,)$. Функция плотности вероятности – экспоненциально убывающая в этом диапазоне функция $f(x) = C \exp(-a x)$. Единственный параметр распределения - коэффициент a – любое положительное число $a > 0$. Из условия нормировки найдем, что константа $C = a$. Функция распределения (интегральная) на этом интервале равна $F(x) = 1 - \exp(-a x)$. Все числовые характеристики случайной величины выражаются через параметр a. Математическое ожидание равно $MX = 1 / a$. Медиана равна $MeX = \mathrm{Ln}2 / a$. Дисперсия равна $DX = 1 / a^2$. Среднеквадратическое отклонение равно $SX = 1 / a$, коэффициент вариации равен $CVX = 100\%$.

Нормальное распределение (Гаусса) непрерывной случайной величины. Диапазон значений случайной величины – вся числовая ось $-\infty < X < +\infty$. Функция плотности вероятности – $f(x) = C \exp(-(x - m)^2 / 2s^2)$. Для задания конкретного распределения из класса нормальных распределений нужно задать числовые значения двум его параметрам: m – любое действительное число, s – любое положительное число $s > 0$. Из условия нормировки найдем, что константа $C = (2 \pi s^2)^{-1/2}$. Интегральная функция распределения Гаусса через элементарные функции не выражается. Ее значения протабулированы и могут быть получены из таблицы или вычислены с помощью встроенных во многие математические пакеты

функции, например, в Excel. Все числовые характеристики случайной величины выражаются через параметры m и s. Математическое ожидание и медиана равны $MX = MeX = m$. Дисперсия равна $DX = s^2$. Среднеквадратическое отклонение равно $SX = s$, коэффициент вариации равен $CVX = s / m$ 100%. Размах равен бесконечности. График кривой распределения (функции плотности вероятности) имеет характерную колоколообразную форму симметричной кривой с единственным максимумом в точке $x = m$. В интервале от $m - 3s$ до $m + 3s$ сосредоточено площадь под кривой, равна 0,999, т.е. значения случайной величины, которые появятся с этой обще вероятностью.

Биномиальное распределение дискретной случайной величины. Как известно, формула бинома Ньютона порядка n имеет вид

$$(a+b)^n = \sum_{k=0}^{n} C(n,k) \cdot a^k \cdot b^{n-k}, \qquad C(n,k) = \frac{n!}{k!\cdot(n-k)!}$$

Примем $a = p$, $0 < p < 1$ и $b = q = 1 - p$. Обозначим также $p_n(k) = C(n,k)\ p^k\ q^{n-k}$. *N.* Тогда значения $p_n(k)$ задают некоторое дискретное распределение вероятностей. Диапазон значений биномиальной случайной величины – целые числа от 0 до n: 0, 1, 2, …, n. Вероятности их появления равны $p_n(k)$. Распределение такой дискретной случайной величины называют биномиальным. У него два параметра. Параметр n – любое целое положительное число n = 1, 2, … . Параметр p принимает дробные значения от нуля до единицы. Математическое ожидание равно $MX = p\ n$. Дисперсия равна $DX = n\ p\ q$.

Распределение Пуассона дискретной случайной величины. Как известно, степенной ряд для экспоненциальной функции равен

$$e^a = \sum_{k=0}^{\infty} \frac{a^k}{k!}$$

Поэтому, если принять в качестве дискретного распределения вероятностей ряд значений вероятностей $p(k) = a^k / k!\ \exp(-a)$, то их сумма будет равна единице. Случайная величина X, принимающая значения k = 0, 1, 2, 3, … с вероятностями $p(k)$, называется Пуассоновской случайной величиной, а само распределение – распределением Пуассона. Его математическое ожидание равно $MX = a$. Дисперсия равна $DX = a$. Заметим, что параметр $a > 0$ и не имеет размерности, а распределение уникально тем, что у него математическое ожидание и дисперсия равны друг другу.

Связи между распределениями

При определенных условиях, которым должны отвечать параметры распределений, распределения случайных величин переходят друг в друга. Приведем эти условия.

При $n \to \infty$ и $n \cdot p = a = \text{const}$, т.е. $a \to 0$, биномиальное распределение превращается в распределение Пуассона.

При $n \to \infty$ и $n \cdot p = m = \text{const}$ и $n \cdot p \cdot (1 - p) = n \cdot p \cdot q = s^2 = \text{const}$, биномиальное распределение превращается в нормальное.

Другие связи между распределениями возникают при рассмотрении так называемых вероятностных схем, т.е. моделей некоторых вероятностных моделей типичных задач, к которым сводятся многие задачи встречающиеся на практике. Приведем эти связи в виде задач.

Задание. Решить задачи

1. Интервалы времени между звонками являются независимыми непрерывными случайными величинами, распределенными по одному и тому же экспоненциальному распределению с параметром a = const. Как распределено число звонков, поступающих на этот телефон за фиксированный интервал времени T ?

Решение. Число звонков будет дискретной случайной величиной, распределенной по закону Пуассона с параметром $b = a \cdot T$.

2. Количество документов, поступающих на компьютер для обработки в течение 8-часового рабочего дня, распределено по закону Пуассона с математическим ожиданием b = 16 документов. Как распределены интервалы времени между моментами поступлениями документов на компьютер?

Решение. Интервалы времени имеют экспоненциальное распределение с параметром $a = b / T = 2$ документа в час.

3. Центральная предельная теорема теории вероятностей: сумма независимых случайных величин с конечными математическими ожиданиями и конечными дисперсиями одного порядка по величине при числе слагаемых, стремящемся к бесконечности, распределена по нормальному закону. Фактически теорема выполняется уже при числе слагаемых свыше десятка. Как распределена сумма 12 независимых случайных величин, имеющих одинаковые равномерные распределения в диапазоне от $A = 0$ до $B = 1$?

Решение. Математические ожидания всех слагаемых равны $(0 + 1) / 2 = -0{,}5$. Их дисперсии равны $(1 - 0)^2 / 12 = 1/12$. В соответствии со свойствами математических ожиданий и дисперсий суммы независимых случайных величин математическое ожидание и дисперсия суммы двенадцати одинаково равномерно распределенных случайных величин равны, соответственно $12 \cdot 0{,}5 = 6$ и $12 \cdot 1/12 = 1$. Согласно центральной предельной теореме сумма распределена по нормальному закону распределения с $m = 6$ и $s = 1$.

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить все задачи задания.

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 9
Оценка параметров распределения случайной величины

Цель занятия. Научиться вычислять оценки параметров основных законов распределения случайной величины

Краткие теоретические сведения

Оценки числовых характеристик случайной величины могут быть получены обработкой выборочных данных. Эти же числовые характеристики являются функциями параметров распределений этих случайных величин. Поэтому, имея оценки числовых характеристик и зная тип распределения случайной величины, можно найти оценки параметров распределения.

Задачи

1. В течение 8-часового рабочего дня каждый час на компьютер для обработки поступило 6, 8, 3, 12, 5, 9, 7, 1 документов. Предполагая, что число документов, поступающих на компьютер в течение часа, распределено по закону Пуассона, оценить параметр этого распределения.

Решение. Математическое ожидание Пуассоновской случайной величины X равно значению параметра этого распределения $MX = a$. Оценка математического ожидания, а следовательно, и параметра a равна

$a^* = MX^* = (6 + 8 + 3 + 12 + 5 + 9 + 7 + 1) / 8 = 6{,}375$ документа.

2. Проведено $n = 10$ измерений веса студентов. Получена выборка со значениями: 67, 73, 75, 81, 79, 69, 65, 80, 77, 74 кг. Предполагая, что вес распределен по нормальному закону, оценить его параметры.

Решение. Математическое ожидание нормальной случайной величины равно его параметру m. Дисперсия случайной величины равна квадрату второго параметра s^2. Оценки математического ожидания и дисперсии равны $MX^* = 74$, $DX^* = 30{,}6667$. Следовательно, в качестве оценок параметров можно принять $m = 74$ и $s = 5{,}5377$.

3. В нескольких студенческих группах из 20 студентов в каждой проведена контрольная работа с решением задачи. По полученным значениям числа студентов, решивших задачу в каждой из групп, вычислено среднее число студентов, решивших задачу. Это значение оказалось равным $Mx^* = 16{,}5$. Используя схему Бернулли в качестве модели, найти оценки параметров биномиального распределения числа студентов, решивших задачу.

Справка. Схемой Бернулли в теории вероятностей называют следующую схему повторения наблюдений: а) наблюдения проводятся n раз при неизменных условиях; б) результаты наблюдений независимы друг от друга; в) каждое наблюдение завершается одним из двух исходов – «Успех» или «Неудача»; г) вероятность успеха в каждом наблюдении одна и та же и равна p. В условиях схемы Бернулли число успехов в серии из n

наблюдений – дискретная случайная величина, распределенная по биномиальному закону с параметрами n и p.

Решение. Решение или не решение задачи каждым из студентов группы можно рассматривать как схему Бернулли. Число повторных испытаний в схеме равно $n = 20$ - числу студентов в группе. В соответствии с биномиальным распределением математическое ожидание и дисперсия числа «Успехов» равны, соответственно, $MX = n \cdot p$. Оценки этих величин, полученные на основе обработки выборочных данных по нескольким группам из 20 студентов, известны. Это позволяет найти оценку параметра $p^* = MX^* / n = 16{,}5 / 20 = 0{,}825$.

4. По выборке для дискретной случайной величины получены оценки математического ожидания $MX^* = 16{,}5$ и дисперсии $DX^* = 9{,}6$. Выбрав в качестве модели для этой случайной величины биномиальное распределение, оценить его параметры.

Решение. Для биномиального распределения $MX = n \cdot p$ и $DX = n \cdot p \cdot q$. Приравнивая эти выражения найденным оценкам, решим уравнения и получим оценки параметров биномиального распределения

$n \cdot p = 16{,}5 \qquad n \cdot p \cdot (1 - p) = 9{,}6$

$p^* = 1 - 9{,}6 / 16{,}5 = 0{,}4182 \quad n^* = 16{,}5 / 0{,}4182 = 39{,}4565$

Поскольку параметр n целое число, то округляем результат до ближайшего целого и пересчитываем p^*. Окончательно получим оценки

$n^* = 39 \qquad p^* = 16{,}5 / 39 = 0{,}4231$.

Вопросы для текущего контроля

1. В чем разница между параметрами распределения случайной величины и ее числовыми характеристиками?
2. В чем недостаток дисперсии как меры рассеяния значений случайной величины?
3. В чем недостаток размаха ка меры рассеяния значений случайной величины?
4. В чем недостаток математического ожидания как меры середины распределения случайной величины?
5. В чем недостаток медианы ожидания как меры середины распределения случайной величины?
6. В чем преимущество медианы перед математическим ожиданием как мерой середины распределения значений случайной величины?
7. Каковы ограничения и условия осуществления вероятностной модели по схеме Бернулли?
8. В чем суть центральной предельной теоремы?
9. При каких условиях выполняется центральная предельная теорема?
10. Какова связь между биномиальным, нормальным распределениями и распределением Пуассона?

11. Какая связь существует между распределением Пуассона и экспоненциальным распределением? Приведите пример.

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить выборочно задачи на занятии, а остальные дать учащимся для самостоятельного решения.
3. Дать ответы на контрольные вопросы

Содержание отчета

1. Краткая теория
2. Решение задач
3. Ответы на контрольные вопросы

Практическое занятие № 10
Интервальные оценки.
Проверка статистических гипотез

Цель занятия. Научиться вычислять интервальные оценки числовых характеристик нормально распределенной случайной величины. Освоить методику проверки статистических гипотез, связанных с параметрами распределения случайной величины

Краткие теоретические сведения

В теме 6 дано определение и перечислены элементы интервальной оценки числового параметра или характеристики случайной величины и их вероятностное содержание. Приведем два примера получения интервальной оценки для числовой характеристики случайной величины, распределенной по нормальному закону распределения.

Примеры

1. Пять наблюдений за случайной величиной, распределенной по нормальному закону с параметром $s = 2$ и неизвестным параметром m, привели к получению выборки (-1 4 0 1 2). Необходимо получить интервальную оценку параметра m.

Решение. Поскольку параметр m совпадает с математическим ожиданием случайной величины, то его точечной оценкой является оценка математического ожидания, т.е. среднее арифметическое значение выборки *xsa*. Можно доказать, что сумма нормально распределенных случайных величин распределена нормально. Поэтому среднее арифметическое *xsa* распределено по нормальному закону. Его математическое ожидание и дисперсия равны

$$Mxsa = \frac{1}{n} \cdot \left(Mx(1) + ... + Mx(n) \right) = \frac{1}{n} \cdot \left(m + ... + m \right) = m$$

$$Dxsa = D\left(\frac{1}{n} \cdot \left(x(1) + ... + x(n) \right) \right) = \frac{1}{n^2} \left(s^2 + ... + s^2 \right) = \frac{s^2}{n}$$

Следовательно, среднее арифметическое выборки есть случайная величина, распределенная по нормальному закону с параметрами $ma = m$ и $sa = s / \sqrt{n}$. Если из арифметического среднего вычесть m и затем разделить на sa, то случайная величина $Z = (xsa - m) / (s / \sqrt{n})$ распределена по нормальному закону с параметрами $m = 0$ и $s = 1$. Для нормального распределения с такими параметрами функция распределения протабулирована. Выберем доверительную вероятность $P_{\text{доверит}} = 0{,}95$ и найдем в таблице значение Zc, отвечающее условию $p \{ |Z| < Zc \} = P_{\text{доверит}}$. Это значение равно $Zc = 1{,}96$. При отсутствии таблицы значение может быть найдено, например, в программе Excel с помощью функции

=НОРМ.СТ.ОБР(0,975) = 1,959964, в которой 0,975 = 1 – (1 - $P_{доверит}$)/2

Теперь, используя тождественные алгебраические преобразования для неравенства, преобразуем выражение $p\ \{\ |Z| < Zc\ \} = P_{доверит}$.

$$p\left\{\left|\frac{xsa - m}{s/\sqrt{n}}\right| < Zc\right\} = P_{доверит}$$

$$p\left\{xsa - \frac{1{,}96 \cdot s}{\sqrt{n}} < m < xsa + \frac{1{,}96 \cdot s}{\sqrt{n}}\right\} = 0{,}95$$

Вычислив значение *xsa* для выборки и границы двойного неравенства для *m*, получим интервальную оценку для параметра *m*

$$p\left\{1{,}2 - \frac{1{,}96 \cdot 2}{\sqrt{5}} < m < 1{,}2 + \frac{1{,}96 \cdot 2}{\sqrt{5}}\right\} = 0{,}95, \text{ или}$$

$$p\{-0{,}5531 < m < 2{,}9531\} = 0{,}95$$

Итак, из данных выборки и априорной информации о законе распределения случайной величины следует, что неизвестное истинное значение параметра *m* с надежностью 0,95 содержится в случайном интервале (-0,55 ; +2,95).

2. Точечная оценка дисперсии нормальной случайной величины по выборке объема $n = 5$ равна $s^2* = 9$. Построить интервальную оценку дисперсии для доверительной вероятности 0,90.

Решение. В теории вероятностей доказано, что случайная величина, равная

$$z = \frac{DX* \cdot (n-1)}{s^2}, \quad где\ DX* = \frac{1}{n-1}\sum_{j=1}^{n}\left(x(j) - \overline{x}\right)^2,$$

в которой $x(j)$ – независимые нормальное распределенные случайные величины с одинаковыми дисперсиями S^2, распределена по закону распределения хи-квадрат с параметром $k = n$ -1 (этот параметр называется числом степеней свободы).

Найдем значения этой случайной величины *z*1 и *z*2 для симметричных пределов

$P\ \{\ z < z1\ \} = (1 - P_{доверит})/2 \quad P\ \{\ z < z2\ \} = (1 + P_{доверит})/2$

Для этого используем, например, функцию Excel ХИ2.ОБР (p ; *k*)

*z*1 = =ХИ2.ОБР(0,05;4) = 0,711

*z*2 = =ХИ2.ОБР(0,95;4) = 9,488

Записываем вероятностное неравенство

$$P\left(z1 < \frac{DX* \cdot (n-1)}{s^2} < z2\right) = p_{доверит}$$

Выполним тождественные алгебраические преобразования и получим

$$P\left(\frac{DX^{*}\cdot(n-1)}{z2} < s^2 < \frac{DX^{*}\cdot(n-1)}{z1}\right) = p_{доверит}$$

$$P\left(\frac{9\cdot 4}{9,488} < s^2 < \frac{9\cdot 4}{0,711}\right) = p_{доверит}$$

$$P\left(3,79 < s^2 < 50,63\right) = p_{доверит}$$

Результат показывает низкую точность оценивания дисперсии при невысокой надежности. Это связано, прежде всего, с малым объемом выборки для оценивания дисперсии нормальной случайной величины.

Проверка статистических гипотез

Статистической гипотезой называют некоторое предположение, связанное со случайной величиной. Примеры статистических гипотез: 1) H_0 - математическое ожидание случайной величины равно 5; 2) H_0 – дисперсия случайной величины больше 9; 3) H_0 – случайная величина распределена по экспоненциальному закону. Пример гипотезы, не относящейся к категории статистических гипотез: на Марсе есть жизнь.

Источником информации для проверки гипотезы является выборка. Однако, никакая выборка не позволяет получить достоверный ответ: гипотеза верна или гипотеза ложна. Это объясняется неполной информацией в любой выборке. Поэтому проверка истинности гипотезы заменяется проверкой адекватности: противоречат гипотеза и выборка друг другу или не противоречат. Останавливаясь на одном из решений, всегда есть опасность совершить ошибку: отвергнуть правильную гипотезу или принять ложную гипотезу.

В некоторых случаях проверка гипотезы, связанной с параметром распределения случайной величины, проводится с использованием интервальной оценки параметра. Приведем задачи проверки гипотез, решаемые с помощью интервальных оценок.

Задание. Решить задачи

Задачи

1. Случайная величина распределена по нормальному закону. Параметр *m* неизвестен, параметр *s* = 3. Оценка по выборке объема *n* дала значение среднего арифметического *xSr* = 7. Найти интервальные оценки для *m* для доверительной вероятности 0,95 в предположении, что *n* = 5; 25; 100; 500. Построить графики зависимости границ доверительного интервала от объема выборки.

2. Случайная величина распределена по нормальному закону. Параметр *m* неизвестен, параметр *s* = 3. Оценка по выборке объема *n* =10 дала значение среднего арифметического *xSr* = 7. Найти интервальные оценки для *m* для доверительных вероятностей 0,6; 0,8; 0,9; 0,95; 0,99; 0,999.

Построить графики зависимости границ доверительного интервала от доверительной вероятности.

3. Используя функцию ХИ2.ОБР программы Excel, построить графики функций распределения хи-квадрат для числа степеней свободы: $k = 1$; $k = 5$; $k = 10$; $k = 30$.

4. Случайная величина распределена по нормальному закону. Параметр s неизвестен. Оценка по выборке объема n дала значение оценки дисперсии $DX^* = 7$. Найти интервальные оценки для s^2 для доверительной вероятности 0,95 в предположении, что n = 5; 10; 30; 100. Построить графики зависимости границ доверительного интервала от объема выборки.

5. Случайная величина распределена по нормальному закону. Параметр s неизвестен. Оценка по выборке объема n =10 дала значение оценки дисперсии $DX^* = 7$. Найти интервальные оценки для s^2 для доверительных вероятностей 0,6; 0,8; 0,9; 0,95; 0,99; 0,999. Построить графики зависимости границ доверительного интервала от доверительной вероятности.

6. Проверяется гипотеза H_0 – случайная величина, распределенная по нормальному закону, имеет параметр $m = 3$. Проверить эту гипотезу, если известно, что параметр $s = 2$, по выборке (-1 4 0 1 2).

Решение. Интервальная оценка параметра m при доверительной вероятности 0,95 равна

$$p\{-0{,}5531 < m < 2{,}9531\} = 0{,}95$$

Поскольку гипотетическое значение $m = 3$ лежит вне пределов интервальной оценки, гипотезу отвергаем с вероятностью ошибочности этого решения, равной $\alpha = 1 - P_{\text{доверит}} = 0{,}05$.

7. Проверяется гипотеза H_0 – случайная величина, распределенная по нормальному закону, имеет параметр $m = 1$. Проверить эту гипотезу, если известно, что параметр $s = 2$, по выборке (-1 4 0 1 2).

Решение. Интервальная оценка параметра m при доверительной вероятности 0,95 равна

$$p\{-0{,}5531 < m < 2{,}9531\} = 0{,}95$$

Поскольку гипотетическое значение $m = 1$ лежит в пределах интервальной оценки, гипотеза не противоречит выборочным данным и может быть принята.

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить выборочно задачи на занятии, а остальные дать учащимся для самостоятельного решения.

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 11
Базовые понятия линейной алгебры

Цель занятия. Усвоить базовые понятие линейной алгебры. Научиться решать системы линейных уравнений геометрическим способом и алгебраическим способом через определители

Краткие теоретические сведения

Линейная функция – зависимость между аргументом X и зависимой переменой Y вила $Y = a\ X + b$. Числа a и b – коэффициенты (параметры) линейной функции. Графиком линейной функции в декартовых координатах на плоскости является прямая линия. При $b = 0$ линия проходит через начало координат. При $a = 0$ линия параллельна оси абсцисс. Линия пересекает ось ординат в точке $Y = b$. Коэффициент a равен тангенсу угла наклона линии к оси абсцисс. Если a не равно нулю, то линия пересекает ось абсцисс в точке, ордината которой равна нулю $Y = 0$, а абсцисса является решением линейного уравнения $a\ X_0 + b = 0$: $X_0 = - b / a$. Если обозначить $(X_0\ ;\ 0)$ и $(0\ ;\ Y_0)$ – координаты точек на осях координат, через которые проходит прямая, то функцию, описывающую эту прямую, можно записать в виде $X / X_0 + Y / Y_0 = 1$. Выражение $A\ X + B\ Y = C$ описывает линейную функцию. Ее график – прямая линия, пересекающая оси координат в точках $(C / A\ ;\ 0)$ и $(0\ ;\ C / B)$.

Два выражения $A_1\ X + B_1\ Y = C_1$ и $A_2\ X + B_2\ Y = C_2$ описывают две прямых линии. Координаты точки их пересечения $(X_0\ ;\ Y_0)$, если такая точка существует, т.е. линии не параллельны, должны удовлетворять обоим выражениям. Это означает, что при подстановке координат этой точке в обе функции правая и левая части равенств должны быть одинаковы

$$\begin{cases} A_1 X_0 + B_1 Y_0 = C_1 \\ A_2 X_0 + B_2 Y_0 = C_2 \end{cases} \rightarrow X_0 = \frac{C_1 B_2 - C_2 B_1}{A_1 B_2 - A_2 B_1} \qquad Y_0 = \frac{A_1 C_2 - A_2 C_1}{A_1 B_2 - A_2 B_1}$$

Выражения $\Delta = A_1\ B_2 - A_2\ B_1$, $\Delta X = C_1\ B_2 - C_2\ B_1$, $\Delta Y = A_1\ C_2 - A_2\ C_1$ называют, соответственно, определителем (детерминантом) системы, переменной X и переменной Y. Точка пересечения существует, если существует решение системы линейных уравнений, т.е. при $\Delta \neq 0$.

Аналогично, для трех переменных X, Y, Z в декартовых координатах трехмерного пространства линейная функция $Z = a\ X + b\ Y$ описывает плоскость в этом пространстве. Более симметрично линейную функцию, описывающую плоскость, можно записать в виде $A\ X + B\ Y + C\ Z = D$. Эта плоскость пересекает оси координат в точках $(D / A, 0 , 0)$; $(0, D / B, 0)$; $(0, 0, D / C)$.

Две линейные функции $A_1\ X + B_1\ Y + C_1\ Z = D_1$ и $A_2\ X + B_2\ Y + C_2\ Z = D_2$ задают множество точек, общих для обеих плоскостей, т.е. прямую линию – пересечение этих плоскостей. Три линейные функции задают множество

точек, образованное пересечением трех плоскостей, т.е. точку в трехмерном пространстве

$$\begin{cases} A_1X_0 + B_1Y_0 + C_1Z_0 = D_1 \\ A_2X_0 + B_2Y_0 + C_2Z_0 = D_2 \\ A_3X_0 + B_3Y_0 + C_3Z_0 = D_3 \end{cases} \rightarrow X_0 = \frac{\Delta X}{\Delta} \qquad Y_0 = \frac{\Delta Y}{\Delta} \qquad Z_0 = \frac{\Delta Z}{\Delta}$$

Правило вычисления определителя системы: коэффициенты из левой части системы линейных уравнений записываются в виде таблицы (матрицы) и дополняются справа еще двумя своими столбцами, затем вычисляются произведения элементов всех диагоналей, содержащих по три элемента, и алгебраически складываются. Произведения диагоналей, идущих слева сверху вниз направо, входят в сумму со знаком +, произведения диагоналей, идущих слева снизу вверх направо, входят в сумму со знаком -. Для вычисления определителей соответствующей неизвестной его коэффициенты в матрице расчетов заменяют на коэффициенты D_1, D_2, D_3.

Пример.

$$\left\{\begin{array}{l} 2X_0 - 4Y_0 + 6Z_0 = 9 \\ 3X_0 + 0Y_0 + 5Z_0 = -30 \\ 1X_0 + 4Y_0 + 3Z_0 = 20 \end{array}\right. \quad \left|\begin{array}{ccc|cc} 2 & -4 & 6 & 2 & -4 \\ 3 & 0 & 5 & 3 & 0 \\ 1 & 4 & 3 & 1 & 4 \end{array}\right.$$

$$\Delta = 2\cdot 0\cdot 3 + (-4)\cdot 5\cdot 1 + 6\cdot 3\cdot 4 - 6\cdot 0\cdot 1 - 2\cdot 5\cdot 4 - (-4)\cdot 3\cdot 3 =$$

$$= 0 - 20 + 72 - 0 - 40 + 36 = 48$$

Определители ΔX, ΔY, ΔZ b значения неизвестных, соответственно, равны

$$\left\{\begin{array}{l} 2X_0 - 4Y_0 + 6Z_0 = 9 \\ 3X_0 + 0Y_0 + 5Z_0 = -30 \\ 1X_0 + 4Y_0 + 3Z_0 = 20 \end{array}\right. \quad \left|\begin{array}{ccc|cc} 9 & -4 & 6 & 9 & -4 \\ -30 & 0 & 5 & -30 & 0 \\ 20 & 4 & 3 & 20 & 4 \end{array}\right.$$

$$\Delta X = 9\cdot 0\cdot 3 + (-4)\cdot 5\cdot 20 + 6\cdot(-30)\cdot 4 - 6\cdot 0\cdot 20 - 9\cdot 5\cdot 4 - (-4)\cdot(-30)\cdot 3 =$$

$$= 0 - 400 - 720 - 0 - 180 - 360 = -1660 \qquad X_0 = -\frac{1660}{48} \approx -34{,}583(3)$$

$$\left\{\begin{array}{l} 2X_0 - 4Y_0 + 6Z_0 = 9 \\ 3X_0 + 0Y_0 + 5Z_0 = -30 \\ 1X_0 + 4Y_0 + 3Z_0 = 20 \end{array}\right. \quad \left|\begin{array}{ccc|cc} 2 & 9 & 6 & 2 & 9 \\ 3 & -30 & 5 & 3 & -30 \\ 1 & 20 & 3 & 1 & 20 \end{array}\right.$$

$$\Delta Y = 2\cdot(-30)\cdot 3 + 9\cdot 5\cdot 1 + 6\cdot 3\cdot 20 - 6\cdot(-30)\cdot 1 - 2\cdot 5\cdot 20 - 9\cdot 3\cdot 3 =$$

$$= -180 + 45 + 360 + 180 - 200 - 81 = 124 \qquad Y_0 = \frac{124}{48} \approx 2{,}583(3)$$

$$\begin{cases} 2X_0 - 4Y_0 + 6Z_0 = 9 \\ 3X_0 + 0Y_0 + 5Z_0 = -30 \\ 1X_0 + 4Y_0 + 3Z_0 = 20 \end{cases} \quad \begin{array}{ccc|cc} 2 & -4 & 9 & 2 & -4 \\ 3 & 0 & -30 & 3 & 0 \\ 1 & 4 & 20 & 1 & 4 \end{array}$$

$$\Delta Y = 2 \cdot 0 \cdot 20 + (-4) \cdot (-30) \cdot 1 + 9 \cdot 3 \cdot 4 - 9 \cdot 0 \cdot 1 - 2 \cdot (-30) \cdot 4 - (-4) \cdot 3 \cdot 20 =$$

$$= 0 + 120 + 108 - 0 + 240 + 240 = 708 \qquad Z_0 = \frac{708}{48} = 14{,}75$$

Из формул видно, что решение, т.е. точка пересечения трех плоскостей, существует, если определитель $\Delta \neq 0$.

Задание. Решить задачи

1. Построить графики функций :

а) $Y = -5X + 6$ б) $Y = 5X - 6$ в) $Y = -7$ г) $X = 8$

2. Записать через отсекаемые на осях отрезки функции:

а) $Y = -5X + 6$ б) $Y = 5X - 6$

3. Решить системы уравнений геометрически (построив графики функций) и алгебраически, используя определители

$$\begin{cases} 4X - 5Y = 20 \\ 8X - 10Y = 40 \end{cases}$$

$$\begin{cases} 4X - 5Y = 20 \\ 8X + 10Y = 40 \end{cases}$$

4. Решить системы уравнений алгебраически, используя определители

$$\begin{cases} -5X + 20Y - 3Z = 120 \\ 12Y - 15Z = 100 \\ 10X + Y + 6Z = 60 \end{cases}$$

Проверить на тождество найденные корни подстановкой в уравнения.

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить выборочно задачи на занятии, а остальные дать учащимся для самостоятельного решения.

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 12
Элементы многомерного статистического анализа и моделирования. Базовые элементы корреляционного анализа и регрессионного анализа

Цель занятия. Усвоить понятие многомерного статистического анализа модели, корреляционного и регрессионного анализа. Освоить методику получения оценок коэффициентов линейной регрессионной модели методом наименьших квадратов в программах MS Excel

Краткие теоретические сведения

В таблице приведены значения различных показателей за пять рабочих дней недели: *T* – средняя температура дня, градусов Цельсия; *P* – давление воздуха, миллиметров ртутного столба, *fi* – влажность воздуха, %; Инт – средняя интенсивность автомобильного движения, автомобилей в час; V – средняя скорость автомобилей, километров в час; W – средняя выручка сети магазинов сезонной одежды, сотен тысяч рублей за день.

Таблица показателей *x (j, j)*

Дни	*i*	T, гр.	P, р.ст.	fi,%	Инт. м/ч	V, км/ч	W, р.
Показатель	j / i	1	2	3	4	5	6
Пн	1	4	736	78	1000	18	12
Вт	2	6	750	65	1200	27	14
Ср	3	8	768	52	1450	30	11
Чт	4	3	740	68	1100	22	15
Пт	5	-1	735	70	860	20	14
xsr (*j*)		4	745,8	66,6	1122	23,4	13,2
S (*j*)		3,391	13,755	9,476	222,3	4,980	1,643

Пример основ множественного корреляционного анализа

Оценить взаимосвязь между данными. Использовать коэффициенты парной корреляции. Коэффициенты парной корреляции образуют корреляционную матрицу

$$\mathbf{R} = \left\| r(i,j) \right\|_{j=1,\ldots,6}^{i=1,\ldots,6}, \qquad r(i,j) = \frac{K(i,j)}{S(i) \cdot S(j)}$$

$$K(i,j) = \frac{1}{6-1} \sum_{k=1}^{6} \left(x(k,i) - xsr(i) \right) \cdot \left(x(k,j) - xsr(j) \right)$$

$$S(i) = \sqrt{\frac{1}{6-1} \sum_{k=1}^{6} \left(x(k,i) - xsr(i) \right)^2}, \qquad xsr(i) = \frac{1}{6} \sum_{k=1}^{6} x(k,i)$$

Результаты вычислений средних *xsr* и среднеквадратических отклонений *Sx* приведены в последних строках таблицы. Матрица коэффициентов парной корреляции равна

$$R = \begin{pmatrix} 1 & 0{,}84 & -0{,}62 & 0{,}93 & 0{,}77 & -0{,}58 \\ 0{,}84 & 1 & -0{,}93 & 0{,}96 & 0{,}95 & -0{,}58 \\ -0{,}62 & -0{,}93 & 1 & -0{,}86 & -0{,}94 & 0{,}36 \\ 0{,}93 & 0{,}96 & -0{,}86 & 1 & 0{,}91 & -0{,}52 \\ 0{,}77 & 0{,}95 & -0{,}94 & 0{,}91 & 1 & -0{,}32 \\ -0{,}58 & -0{,}58 & 0{,}36 & -0{,}52 & -0{,}32 & 1 \end{pmatrix}$$

В матрице по диагонали стоят единицы, вне диагонали – коэффициенты парной корреляции, которые могут быть как положительными, так и отрицательными, а по абсолютной величине меньше или равны единице. Матрица всегда симметрична, т.е. $r(i, j) = r(j, i)$.

Для интерпретации элементов матрицы руководствуются следующим правилом: если $|r(i, j)| < 0{,}2$, то корреляция между показателями $x(i)$ и $x(j)$ отсутствует. Если $0{,}2 < |r(i, j)| < 0{,}4$, то корреляция есть, но она слабая. Если $0{,}4 < |r(i, j)| < 0{,}7$, то корреляция есть, но она средняя по «силе взаимодействия». Если $0{,}7 < |r(i, j)| < 0{,}95$, то корреляция есть, и она сильная. Если $0{,}95 < |r(i, j)| < 1$, то между показателями $x(i)$ и $x(j)$ существует практически функциональная линейная зависимость.

Если у значимого коэффициента парной корреляции знак положительный, это означает, что большим значениям одного показателя отвечают большие значения второго показателя, а его меньшим значениям - меньшие значения второго показателя.

Если у значимого коэффициента парной корреляции знак отрицательный, это означает, что большим значениям одного показателя отвечают меньшие значения второго показателя, а его меньшим значениям - большие значения второго показателя.

Для приведенных многомерных данных – показателей *x* – можно предположить, что показатель W является переменной, зависимой от остальных переменных: температуры на улице, давления и влажности воздуха, интенсивности автомобильного движения и его средней скорости. В

линейном приближении эту зависимость можно записать как линейную функцию зависимости W от остальных переменных

$$W^*(k) = C_0 + C_1 x(k,1) + C_2 x(k,2) + C_3 x(k,3) + C_4 x(k,4) + C_5 x(k,5)$$

Это представление зависимости называют регрессией (регрессионной моделью). Параметры C_j называют коэффициентами регрессии. Они показывают величину и знак влияния соответствующего показателя x(i , j) на W.

Для получения оценок коэффициентов регрессии используют метод наименьших квадратов: наилучшими признаются числовые значения коэффициентов, для которых критерий

$$Q(C_0,...,C_5) = \sum_{k=1}^{5} \left(W(k) - W^*(k, C_0,...,C_5)\right)^2 \to \min_{C_0,...,C_5}$$

принимает наименьшее значение. Метод наименьших квадратов приводит к формуле для оценки коэффициентов регрессии, которую компактно записывают в матричной форме

$$\mathbf{C} = \left(\mathbf{F}^{\mathbf{T}} \cdot \mathbf{F}\right)^{-1} \cdot \mathbf{F}^{\mathbf{T}} \cdot \mathbf{W}$$

В этой формуле индекс T означает операцию транспонирования матрицы, а верхний индекс -1 означает операцию обращения матрицы. Вектор **C** – это вектор оценок коэффициентов регрессии, вектор **W** – это вектор значений переменной величины *W*. Матрица **F** строится из функций показателей, являющихся множителями при коэффициентах регрессии в модели для всех значений этих показателей в матрице данных X по правилу:

$\mathbf{F} =$

k	$x(k,0)$	$x(k,1)$	$x(k,2)$	$x(k,3)$	$x(k,4)$	$x(k,5)$
1	1	4	736	78	1000	18
2	1	6	750	65	1200	27
3	1	8	768	52	1450	30
4	1	3	740	68	1100	22
5	1	−1	735	70	860	20

В первом столбце матрицы помещены значения функции – множителя при коэффициенте регрессии C_0.

Дальнейшие операции целесообразно выполнять в Excel, Matlab или SciLab. Произведение матрицы **F** на транспонированную матрицу приводит к симметричной квадратной матрице

$$\mathbf{F}^{\mathbf{T}} \cdot \mathbf{F} = \begin{pmatrix} 5 & 20 & 3729 & 333 & 5610 & 117 \\ 20 & 126 & 15073 & 1252 & 25240 & 520 \\ 3729 & 15073 & 2781845 & 247864 & 4195700 & 87518 \\ 333 & 1252 & 247864 & 22537 & 366400 & 7615 \\ 5610 & 25240 & 4195700 & 366400 & 6492100 & 135300 \\ 117 & 520 & 87518 & 7615 & 135300 & 2837 \end{pmatrix}$$

После обращения этой матрицы и вычисления произведений в матричной формуле получаем вектор коэффициентов регрессии

$$\mathbf{C} = (134{,}7813 \quad -1{,}8583 \quad -0{,}2435 \quad 0{,}5648 \quad 0{,}0367 \quad -0{,}9585)$$

Вычислим значения W, которые прогнозирует модель для значений показателей из таблицы данных. Подставим значения показателей из каждой строки таблицы в регрессионную модель и вычислим прогнозируемые значения W. Получим значения: (11,55 11,93 11,09 12,24 12,00). Сравнение этих результатов с значениями W из таблицы данных (12 14 11 15 14) приводит к выводу о близости полученной регрессии. Заметим, что изменения данных и прогнозируемых значений имеют одинаковую закономерность.

Задание.

Дать ответы на контрольные вопросы

1. Объясните, почему диагональные элементы корреляционной матрицы всегда равны единице?
2. Объясните, почему коэффициенты парной корреляции по абсолютной величине никогда не могут быть больше единицы?
3. Объясните, почему корреляционная матрица всегда симметричная?
4. По числовым значениям корреляционной матрицы выделите пары показателей с положительной корреляцией и пары показателей с отрицательной корреляцией.
5. По числовым значениям корреляционной матрицы выделите пары показателей с незначимой корреляцией, со слабой корреляцией, со средней корреляцией, с сильной корреляцией, с практически функциональной пропорциональной зависимостью.

Выполнить расчеты

1. Повторить расчеты для получения линейной регрессионной модели 1-го порядка, описывающей зависимость W от одного из показателей. Сделать выводы о наилучшей из построенных моделей.
2. Повторить расчеты для получения линейной регрессионной модели 1-го порядка, описывающей зависимость W от одной из пар показателей. Сделать выводы о наилучшей из построенных моделей.

3. Повторить расчеты для получения линейной регрессионной модели 1-го порядка, описывающей зависимость W от одной из троек показателей. Сделать выводы о наилучшей из построенных моделей.

4. Повторить расчеты для получения линейной регрессионной модели 1-го порядка, описывающей зависимость W от одной из четверок показателей. Сделать выводы о наилучшей из построенных моделей.

5. Построить линейную квадратичную регрессионную модель для W

$$W^*(k) = C_0 + C_1 x^2(k,1) + C_2 x^2(k,2) + C_3 x^2(k,3) + C_4 x^2(k,4) + C_5 x^2(k,5)$$

Сравнить качество прогнозирования с помощью моделей 1-го и 2-го порядков.

Порядок выполнения работы

1. Изучить теоретический материал
2. Дать ответы на контрольные вопросы
3. Выполнить расчеты: выборочно на занятии, а остальные дать учащимся для самостоятельного выполнения.

Содержание отчета

1. Краткая теория
2. Ответы на контрольные вопросы
3. Выполненные расчеты

Практическое занятие № 13
Понятие классификации и кластеризации

Цель занятия. Усвоить понятия классификации и кластеризации множества объектов, их сходства и различия, понятия меры близости объектов. Понять и объяснить идею алгоритмов классификации и кластеризации. Научиться вычислять расстояние между объектами

Краткие теоретические сведения

Классификации и кластеризации подвергают множество объектов, каждый из которых описывается набором выраженных в числовой форме показателей - признаков. Обозначим число объектов N, число признаков n. Пронумеруем объекты индексом k = 1, …, N, а признаки индексом j = 1, …, n. Сами признаки обозначим x (k, j). Матрица || x (k, j) || образует многомерный набор данных, несущих информацию об объектах через их признаки.

Целью классификации является отнесение каждого из объектов к заранее определенному классу, который описывается типичным набором признаком, отличным от значений набора этих же признаков у другого класса.

Целью кластеризации является группирование (объединение в кластер) объектов, близких друг к другу. Для оценки близости объектов используют понятие расстояния между объектами. Расстояние r (i, k), i = 1, …, N; k = 1, …, N – это неотрицательное число, вычисляемое для каждой пары объектов. Расстояние можно задавать разными способами, но в любом случае расстояния должны отвечать условиям:

1) Для любых двух объектов расстояние между ними – неотрицательная величина r (i, k) >= 0.

2) Для каждого объекта его расстояние от него самого равно нулю: r (k, k) = 0.

3) Должно выполняться «условие треугольника»: r (i, k) + r (k, t) >= r (i, t)

Поскольку признаки могут выражаться в разных единицах измерения и в разных масштабах, то перед проведением многомерного анализа данных их преобразуют к безразмерному виду и масштабируют, т.е. кодируют исходные данные. С этой целью для каждого признака вычисляют среднеквадратическое отклонение sx(j) и делят значение признака у каждого из объектов на это среднеквадратическое значение

$$xSr(j)=\frac{1}{N}\sum_{k=1}^{N}x(j,k);\qquad sx(j)=\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(x(j,k)-xSr(j)\right)^2}$$

$$xc(i,k)=\frac{x(j,k)}{sx(j)},\quad j=1,...,n;\quad k=1,...,N$$

Дальнейшая обработка многомерных данных выполняется для кодированных данных.

Одной из популярных мер расстояния между двумя объектами k и t является обобщенное Евклидово расстояние, происходящее от выражения гипотенузы прямоугольного треугольника через его катеты по теореме Пифагора в Евклидовой геометрии

$$r(k,t) = \sqrt{\sum_{j=1}^{n} w^2(j) \cdot \left(xc(j,k) - xc(j,t)\right)^2}, \qquad k = 1,...,N; \quad t = 1,...,N$$

В этом выражении w (j) – весовые множители, отображающие субъективные представления исследователя многомерных данных об относительной значимости j-го признака по сравнению с другими признаками объектов. Если априори все признаки считают равнозначимыми, то w (j) = 1.

В задаче классификации назначаются реперные точки в n-мерном пространстве признаков xc. Затем строят поверхности в n-мерном пространстве, например, гиперплоскости, проходящие через реперные точки и являющиеся границами классов. Классификация объектов заключается в вычислении расстояния от точки, отображающей объект в n-мерном пространстве, до границ. По тому, с какой стороны границы находится объект, и по расстоянию от объекта до границы определяют принадлежность объекта к тому или иному классу.

В простейшем случае двух классов с плоской границей между ними строят линейную регрессионную модель 1-го порядка, которую записывают в виде равенства

$$C_0 + C_1X_1 + C_1X_2 + ... + C_1X_n = 0$$

В это выражение в качестве X_1, X_2, …, X_n подставляют кодированные значения признаков классифицируемого объекта. Если левая часть выражения оказывается больше нуля, объект относят к одному классу, если меньше нуля – к другому классу.

В кластерном анализе классы объектов и число классов (кластеров) заранее не известно. Поэтому кластеризация проводится последовательно по следующему алгоритму:

1. Каждый объект объявляют кластером. Таким образом, в начале анализа есть M = N кластеров, совпадающих с объектами.
2. По матрице расстояний r_M (k, t) находят два ближайших кластера и объединяют их в один кластер. Перенумеруют кластеры. Их количество становится равным M = M – 1.
3. Пересчитывают расстояния между кластерами и получают новую матрицу расстояний r_M (k, t).
4. Если число кластеров M > 1, возвращаются к п.2 алгоритма.

В пункте 3 алгоритма в качестве расстояния между кластерами принимают: а) расстояние между ближайшими объектами этих кластеров

(метод «ближайшего соседа»); б) расстояние между наиболее удаленными объектами этих кластеров (метод «дальнего соседа»); в) расстояние между геометрическими центрами объектов каждого кластера (метод «центров»).

Результаты кластерного анализа отображают геометрически в виде дендрограммы в декартовых координатах. По одной оси координат размещают кластеризуемые объекты, по другой оси – расстояние между кластерами, объединяя в группу (кластер) все объекты, удаленные на это или меньшее расстояние.

Задание

1. На знакомом языке программирования написать программу преобразования матрицы признаков объектов произвольного размера Nxn в матрицу кодированных значений признаков.
2. Разработать последовательность действий в Excel для преобразования матрицы признаков объектов в матрицу кодированных значений признаков. Записать эти действия в виде макроса.
3. Написать программу вычисления обобщенного Евклидова расстояния между любой парой объектов.
4. Вычислить матрицу расстояний между всеми парами объектов для заданной матрицы признаков объектов.
5. Построить регрессионную модель по заданной матрице значений признаков. Выполнить классификацию объектов на два класса с помощью полученной регрессионной модели.
6. Написать программу поиска ближайших кластеров по матрице расстояний между кластерами.
7. Написать программу вычисления расстояния между двумя кластерами по объектам, входящим в кластеры и матрице расстояний между кластерами и объектами, методом «ближайшего соседа».
8. Написать программу вычисления расстояния между двумя кластерами по объектам, входящим в кластеры и матрице расстояний между кластерами и объектами, методом «дальнего соседа».
9. Написать программу расчета координат геометрического центра объектов одного кластера в пространстве n признаков.
10. Написать программу вычисления расстояния между двумя кластерами по объектам, входящим в кластеры и матрице расстояний между кластерами и объектами, методом «центра».

Порядок выполнения работы

1. Изучить теоретический материал
2. Составить программы в соответствии с заданиями. Выдать учащимся по 2 задания для самостоятельного выполнения.

Содержание отчета

1. Краткая теория
2. Программы решения задач и полученные результаты

Практическое занятие № 14
Понятие градиента

Цель занятия. Усвоить понятие градиента функции, свойства градиента. Научиться вычислять градиент функции

Краткие теоретические сведения

Понятие градиента применимо к функциям, у которых несколько скалярных аргументов, а зависимая переменная – одна и также является скалярной величиной. Говорят, что такая функция описывает скалярное поле. Примеры: а) функция $T = T(x, y, z)$ описывает температуру в каждой точке пространства в декартовых координатах (X, Y, Z); б) $h = h(x, y, t)$ описывает высоту волны на поверхности водоема в момент времени t в точке с координатами x, y в декартовых координатах на плоскости поверхности воды (X, Y).

Градиент – это векторная функция, т.е. функция нескольких переменных, у которой столько же зависимых переменных, сколько и аргументов (обычно это только аргументы, имеющие смысл пространственных переменных x, y, z), который сопоставляется скалярной функции нескольких переменных в каждой точке определения этой функции.

Компоненты вектора градиента равны значениям частных производных функции по ее аргументам, вычисленных во всех точках области определения скалярной функции $u = f(x, y, z)$. Градиент функции $u = f(x, y, z)$ обозначают: $\mathbf{G} = \operatorname{grad} f(x, y, z)$, или $\mathbf{G} = \nabla f(x, y, z)$, или $G = \mathbf{d}u / \mathbf{dR}$. Векторы принято обозначать прямыми заглавными буквами латинского алфавита жирным шрифтом. Знак ∇ читается «набла», вектор $\mathbf{R} = (x, y, z)$.

Важнейшее свойство градиента: вектор градиента в некоторой точке **R** показывает направление, в котором скалярная функция нарастает быстрее всего. Длина градиента пропорциональна скорости нарастания функции в этой точке.

Задачи

1. Скалярная функция определена на плоскости в декартовых координатах (X, Y) выражением $u = 4x^2 + y^2$. Найти градиент этой функции и вычислить его в точке $\mathbf{R} = (4; 8)$.

Решение: найдем частные производные функции $u(x, y)$ и градиент

$$\frac{\partial u(x,y)}{\partial x} = 8x \qquad \frac{\partial u(x,y)}{\partial y} = 2y \qquad \nabla u(x,y) = \begin{pmatrix} 8x & 2y \end{pmatrix}$$

Вычислим значение градиента в точке **R**

$$\nabla u(x,y)\big|_{\mathbf{R}} = \nabla u(x=4, y=8) = \begin{pmatrix} 32 & 16 \end{pmatrix}$$

2. Скалярная функция определена на плоскости в декартовых координатах (X, Y) выражением $u = 4x^2 + y^2 - 8x \cdot y$. Найти градиент этой функции и вычислить его в точках: $\mathbf{R_1} = (4; 8)$; $\mathbf{R_2} = (0; 0)$.

3. Потенциал u поля точечного электрического заряда зависит расстояния точки пространства (x, y) от заряда. Зависимость описывается функцией $u = q / r$, где $r = \sqrt{(x^2 + y^2)}$, а q – величина заряда (с учетом его знака). Напряженность электрического поля есть градиент функции потенциала. Найти вектор напряженности в точках (0, 5 мм) и (-3 мм, 4 мм) для заряда q = 1000 кулон.

4. Найти градиент поля температуры в комнате, которая описывается функцией $T = T(x, y, z) = 20 + 2x + 0{,}5y + 1{,}5z$.

Вопросы для текущего контроля

1. Во что превращается градиент функции, если у нее всего один аргумент – одна независимая переменная, принимающая все возможные значения на всей числовой оси.
2. Запишите формулу, описывающую прямую линию, на которой лежит вектор градиента в данной точке.
3. Найдите длину вектора градиента функции в некоторой точке.
4. Найдите углы между осями координат и направлением градиента в некоторой точке.
5. Запишите условие равенства нулю градиента некоторой функции.
6. Для вышеприведенных задач 1 и 2 найдите точку, в которой градиент равен нулю.
7. Чему равен градиент суммы/разности двух скалярных функций нескольких переменных?
8. Чему равен градиент произведения скалярной функции нескольких переменных на число?
9. Постройте градиент суммы двух функций в некоторой точке, если известны градиенты каждой из функций в этой точке
10. Постройте градиент разности двух функций в некоторой точке, если известны градиенты каждой из функций в этой точке
11. Постройте градиент произведения функции на число в некоторой точке, если известен градиент функции в этой точке.

Порядок выполнения работы

1. Изучить теоретический материал
2. Дать ответы на вопросы текущего контроля

Содержание отчета

1. Краткая теория
2. Ответы на вопросы текущего контроля

ч.2 Анализ и визуализация данных на Python

В данной части сборника рассмотрены программные средства обработки, анализа данных и их использование для решения прикладных задач. Рассмотрены вопросы установки и настройки программных средств, типовые алгоритмы обработки и анализа данных

10 класс 2-е полугодие

Оборудование для проведения занятий

1. Рабочая станция ученика (Intel i5, 8Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)
2. Рабочая станция учителя (Intel i7, 16Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)

Программное обеспечение для проведения занятий (в том числе системное ПО)

1. ОС Windows 10
2. MS Office 2016
3. PyCharm CE (Anaconda)
4. Python 3.7

Рекомендации учителю по проведению занятий

Для проведения практических занятий данного цикла необходимо установить язык программирования Python и среду разработки программ для решения задач анализа данных.

Для разработки программ на языке Python можно использовать различные программные средства и платформы. Рекомендуется установить платформу Anaconda, которая является одной из самых популярных в мире платформой для решения задач анализа данных и является простой в использовании. Данная платформа включает Python, необходимые пакеты *для решения задач обработки и анализа данных* (с использованием Python) и объединяет все эти пакеты в единую систему.

Процесс установки программного обеспечения подробно разобран на практическом занятии 16.

Кроме того в ряде тем потребуется установка дополнительных библиотек и пакетов, не входящих в стандартную установку. Процесс установки дополнительных пакетов подробно разобран на практическом занятии 19.

Все практические занятия включают краткие теоретические сведения по теме занятия, типовые задачи с разбором их решения, вопросы для проверки усвоения материала.

В начале занятия необходимо проверить выполнение заданий для самостоятельного выполнения, выданных на предыдущем практическом занятии.

В ходе проведения занятия рекомендуется изложить учащимся краткие теоретические сведения, разобрать решение типовых задач, описанных в практических занятиях. Для усвоения материала необходимо выдать задания для самостоятельной проработки материала, разобранного на практическом занятии

Практическое занятие № 15
Анализ данных. Примеры и задачи

Цель занятия. Получить представление о задачах анализа данных. Научиться находить простые задачи анализа данных в повседневной жизни

Краткие теоретические сведения

Анализ данных – это извлечение нужной информации из имеющихся наборов данных и/или решение задач на основе извлеченной информации.

Примеры

1. Первоклассник ежедневно в одно и то же время записывает в журнал наблюдений показания термометра на подоконнике. На основе этого потока ежедневных данных дать прогноз температуры на подоконнике на следующий день.
2. Десятиклассник ежедневно записывает официальные сообщения о стоимости 1 доллара в рублях, о стоимости 1 евро в рублях, о стоимости одной бочки (барреля) или тонны нефти марки Брент в рублях. На основе этих потоков данных получить ответ на вопрос о существовании или отсутствии взаимосвязи между этими тремя показателями.
3. В школьном журнале по каждому ученику класса есть записи оценок по предметам и пропускам занятий по этим же предметам. Есть ли взаимосвязь между этими данными? Одинакова ли эта взаимосвязь в разных параллельных классах и в классах разных лет обучения?
4. Мама ученика еженедельно записывает семейные денежные расходы по категориям трат: на разные виды продовольствия, на разные виды одежды, на транспорт, на разные статьи коммунальных расходов, на разные виды отдыха. Постройте модель такого набора данных за квартал (3 месяца или 12 недель) и проранжируйте эти данные по размерам расходов и степени их важности.

Задание

Решить задачи:

1. В таблице приведены данные об утренней температуре по дням недели. Предложите линейную функцию зависимости температуры от номера дня недели. Дайте числовую оценку близости табличных данных к результатам, предсказываемым предложенной линейной функцией.

Дни	**Пн**	**Вт**	**Ср**	**Чт**	**Пт**	**Сб**	**Вс**
№ дня	**1**	**2**	**3**	**4**	**5**	**6**	**7**
T, град.	**15**	**17**	**16**	**20**	**19**	**21**	**18**

2. В таблице приведен фрагмент классного журнала с итоговыми оценками по стобалльной шкале за четверть 10 школьников по математике и

литературе. Предложите способ проверки наличия или отсутствия взаимосвязи между оценками по этим двум предметам.

№ ученика	1	2	3	4	5	6	7	8	9	10
Математика	45	87	65	73	77	82	90	71	84	79
Литература	65	32	72	84	75	88	70	67	75	68

3. Для условий предыдущей задачи предложить графический способ кластеризации (разделения на однородные группы) учеников по двум предметам на три группы.

4. Из общего количества 90 школьников в трех параллельных классах оценки по математике распределились следующим образом: 25 учеников получили «отлично», 36 учеников получили оценку «хорошо», 18 учеников получили оценки «удовлетворительно», 3 ученика получили оценку «неудовлетворительно» и 8 учеников не аттестованы по причине болезни. Предложить метод оценки среднего балла учеников по математике и разброса оценок вокруг этого среднего.

Порядок выполнения работы

1. Изучить теоретический материал
2. Решить две задачи по выбору

Содержание отчета

1. Краткая теория
2. Решение задач

Практическое занятие № 15
Установка программного обеспечения для решения задач анализа данных

Цель занятия. Получить навыки установки программного обеспечения экосистемы Anaconda. Научиться пользоваться интерактивной графической веб-оболочки Jupyter notebook и интегрированной среды Spyder для программирования на языке Python

Краткие теоретические сведения

1. Установка платформы Anaconda

Язык Python широко используется в задачах науки о данных. Существует множество пакетов с открытым исходным кодом и библиотек, в которых можно использовать Python для разных целей.

Для разработки программ на языке Python можно использовать различные программные средства и платформы. Anaconda является одной из самых популярных в мире платформой. Данная платформа включает Python, необходимые пакеты *для решения задач обработки и анализа данных* (с использованием к Python) и объединяет все эти пакеты в единую систему.

Для установки Anaconda в Windows необходимо выполнить следующие шаги:

1. Загрузите установочный файл с сайта https://www.anaconda.com/products/individual#windows.
2. Запустите скачанный файл.
3. Следуйте рекомендациям установщика и в конце каждого шага нажимайте *Next*

   а) примите лицензионное соглашение, нажав *I Agree*

   б) выберите режим установки *Just Me* (для текущего пользователя). При выборе режима for all users (для всех пользователей) потребуются права администратора.

   в) выберите папку установки Anaconda (рис.1). Путь не должен содержать пробелов или символов в кодировке `unicode`.

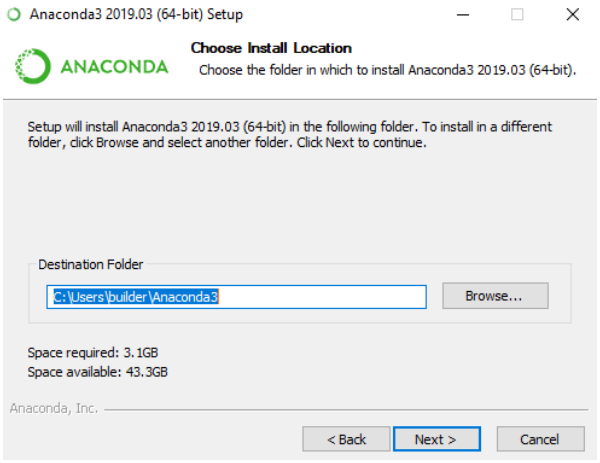


Рис.1. Выбор места установки платформы

г) В следующем окне расширенных опций рекомендуется НЕ включать флаг *add Anaconda to the PATH environment variable*, чтобы не мешать другому ПО (рис.2).

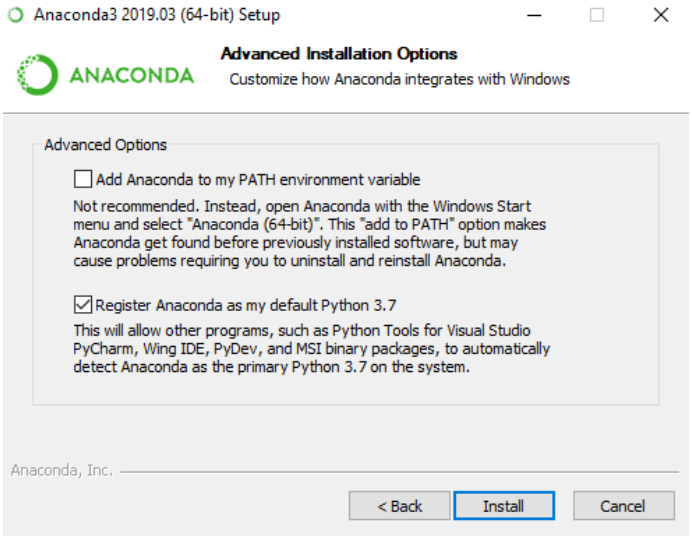


Рис.2 Задание дополнительных параметров установки

д) Выберите вариант регистрации Anaconda в качестве ПО Python по умолчанию.

е) Нажмите кнопку *Install*.

ж) На следующих шагах нажимайте просто *Next.*

После успешной установки будет выведено окно (рис.3). Отключите все флажки и нажмите *Finish.*

Рис.3 Окно завершения установки Anaconda

После завершения установки для запуска экосистемы Anaconda, необходимо в меню *Пуск* открыть *Anaconda Navigator*. Если программа запуститься, то установка завершена успешно

В состав Anaconda входят пакеты, которые будут использоваться в следующих работах:

- удобная среда разработки Spyder для ввода и запуска программ на Python
- удобная среда выполнения кода в интерактивном режиме Jupyter Notebook

2. Запуск программ из навигационной платформы

Для запуска пакета экосистемы Anaconda необходимо в меню Пуск открыть *Anaconda Navigator*.

В открывшемся окне навигационной панели отобразятся все компоненты (пакеты) экосистемы (рис.4).

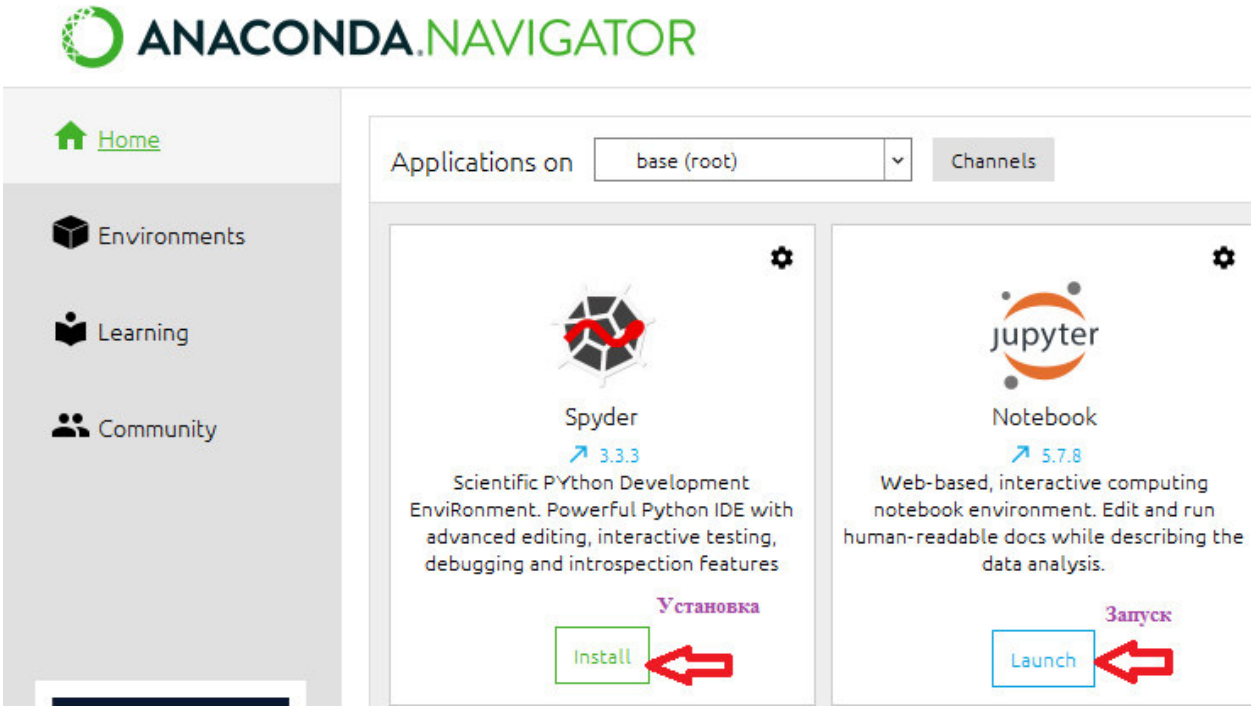


Рис. 4 Навигационная панель платформы Anaconda

Компоненты (пакеты), у которых в нижней части отображается кнопка *Install*, требуют установки. Установка будет выполнена при нажатии на данную кнопку. Компоненты (пакеты), у которых в нижней части отображается кнопка *Launch*, готовы к использованию, и после нажатия данной кнопки будут запущены.

3. Использование Jupyter Notebook

Jupyter Notebook является графической веб-оболочкой для интерактивной оболочки IPython языка программирования Python.

В Jupyter Notebook можно разрабатывать, документировать и выполнять приложения на языке Python, отображать результаты вычислений с медиа представлением (схемы, графики). Введенный код, комментарии и медиаданные сохраняются в файлах (ноутбуках).

Запустить Jupyter Notebook можно из экосистемы Anaconda, нажав кнопку Launch.

Подробное руководство по использованию Jupyter Notebook можно найти на официальном сайте [1]. Ниже даются основные правила работы.
После запуска программы она откроется в браузере (рис. 5).

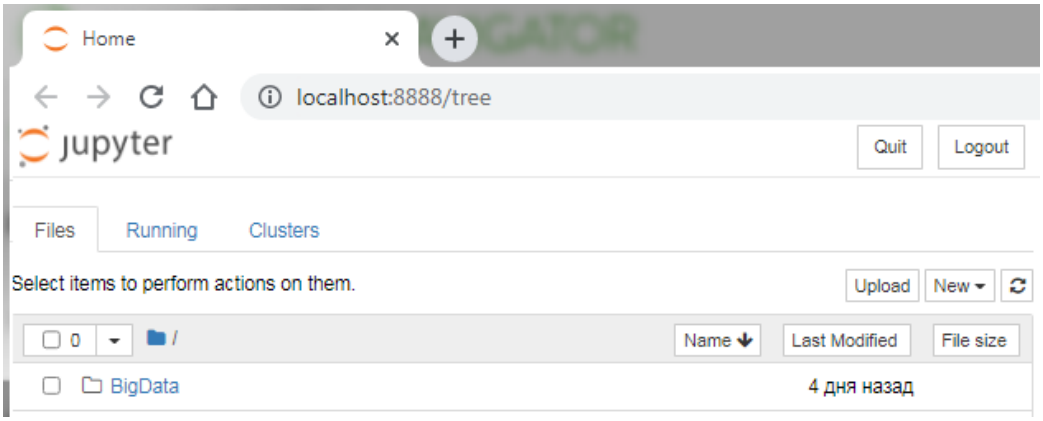


Рис.5 Стартовое окно программы Jupyter Notebook

В начале работы необходимо создать папку для хранения файлов практических и лабораторных работ. Для этого нажмите на кнопку `New` в правой части экрана и выберите в выпадающем списке `Folder` (рис.6).

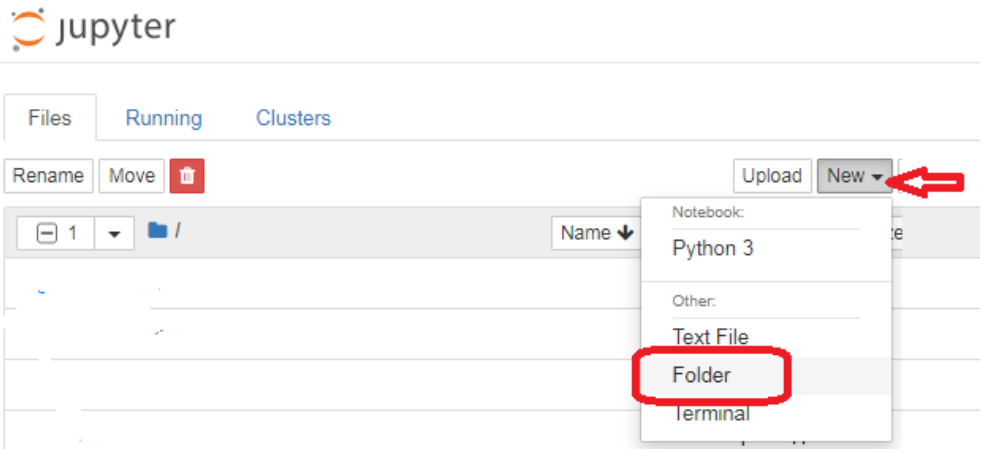
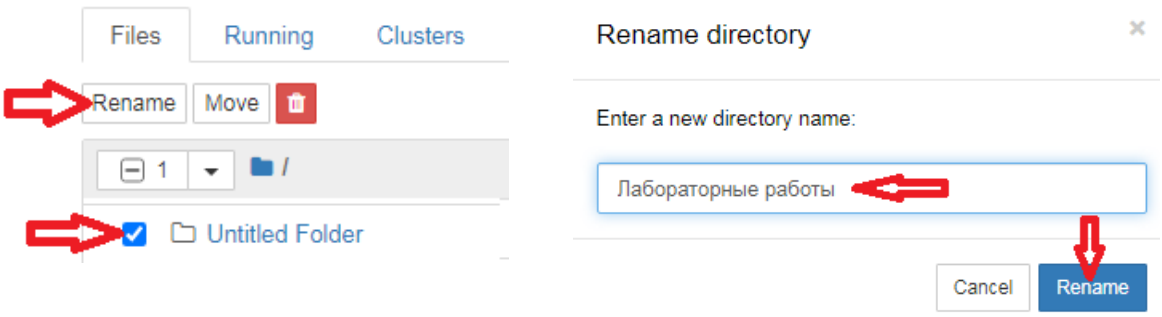


Рис. 6 Создание папки для сохранения файлов-блокнотов

По умолчанию папке присваивается имя *Untitled folder*. Переименуйте ее, например, в *Лабораторные работы*. Для этого поставьте галочку напротив имени папки и нажмите на кнопку *Rename* (рис.7а). В появившемся окне измените прежнее имя папки на новое и снова нажмите на кнопку *Rename* (рис.7б).



а) **б)**

Рис.7 Переименование папки

Зайдите в эту папку и создайте в ней ноутбук (файл с будущим кодом), воспользовавшись той же кнопкой *New*, только на этот раз нужно выбрать *Python 3* (рис.8).

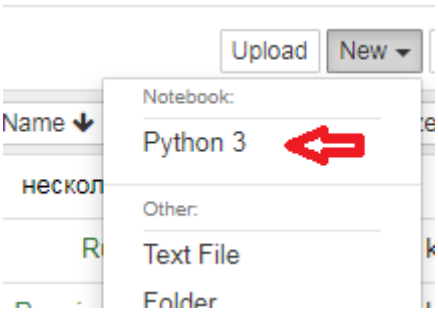
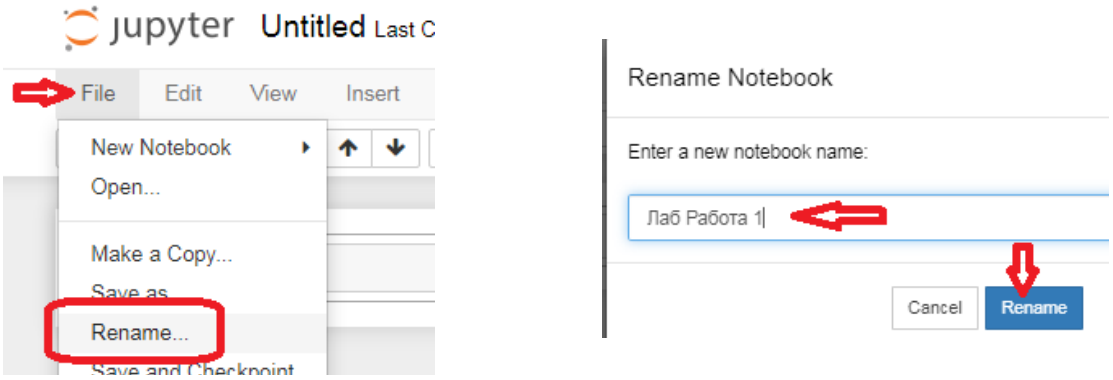


Рис. 8. Создание нового файла-блокнота

Новый файл откроется на новой вкладке браузера и ему будет присвоено имя *Untitled*. Файл также следует переименовать, выбрав пункт *File – Rename* или щелкнув по имени файла (рис.9а). В появившемся окне измените прежнее имя папки на новое и снова нажмите на кнопку *Rename* (рис.9б).



а) **б)**

Рис.9 Переименование файла-блокнота

Код на языке Python или текст в нотации Markdown (произвольный текст для комментариев) нужно вводить в ячейки блокнота (рис.10).

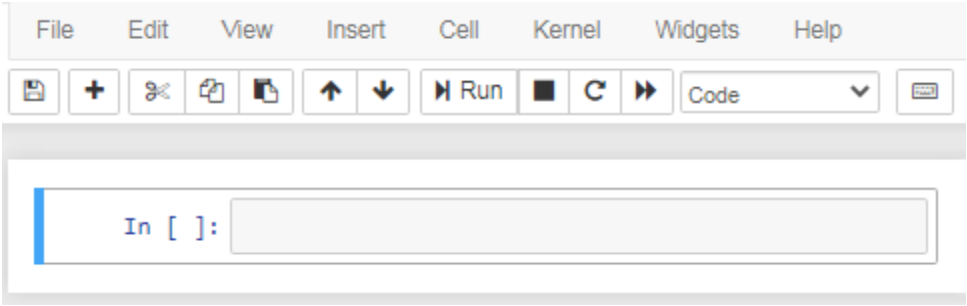
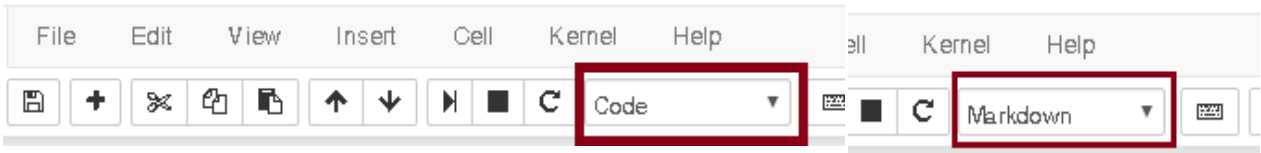


Рис. 10. Ячейка для ввода программного кода или комментариев

Если это код Python, то на панели инструментов в выпадающем списке нужно выставить свойство `Code` (рис.11а), а для комментариев `Markdown` (рис.11б).



а) **б)**

Рис. 11. Выбор типа вводимых данных

Для начал решим простую арифметическую задачу: вычислим произведение двух чисел. Вначале введем комментарий. Зададим свойство *Markdown*, в ячейке введем текст комментария *Первый пример*, а затем нажмите Ctrl+Enter или Shift+Enter или кнопку Run на панели инструментов (рис. 12). В первом случае будет просто добавлен комментарий в блокнот, а в других случаях после ввода комментария будет создана новая ячейка для ввода

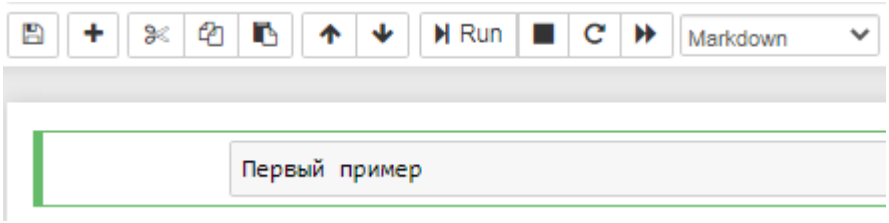


Рис. 12. Ввод комментария

Теперь введем программный код для вычисления произведения. Для этого зададим свойство вида вводимых данных `Code`, введем в ячейке 3 * 5 нажмем кнопку Run на панели инструментов (рис. 13).

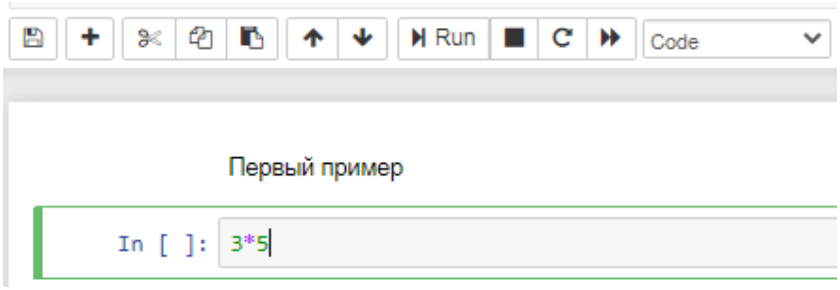


Рис. 13. Ввод программного кода

Интерпретатор Python выполнит введенный код, будет выведен результат и добавлена новая ячейка (рис.14).

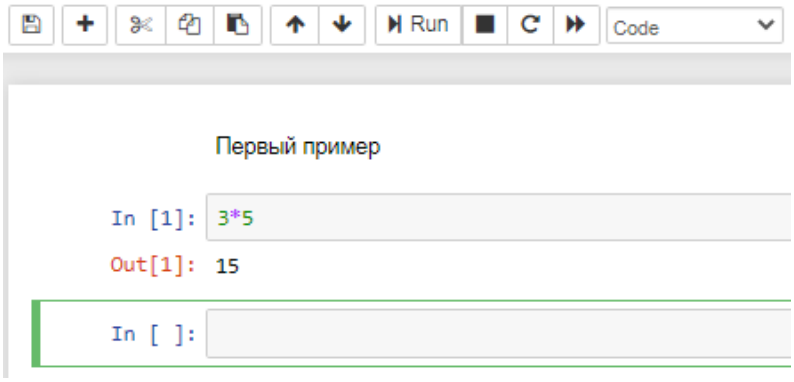


Рис. 14. Вывод результата

Из других элементов интерфейса можно выделить, панель меню (рис. 15а) и расположенную под меню панель инструментов (рис.15б).

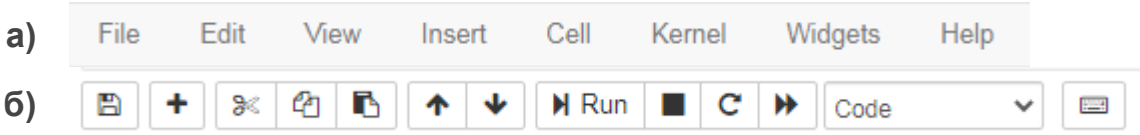


Рис. 15. Элементы интерфейса окна Jupyter Notebook

Программа может находиться в одном из двух режимов: режиме правки (Edit mode) и командном режиме (Command mode). Текущий режим отображается на панели меню в правой части, в режиме правки появляется изображение карандаша (рис.16), отсутствие этой иконки значит, что ноутбук находится в командном режиме.

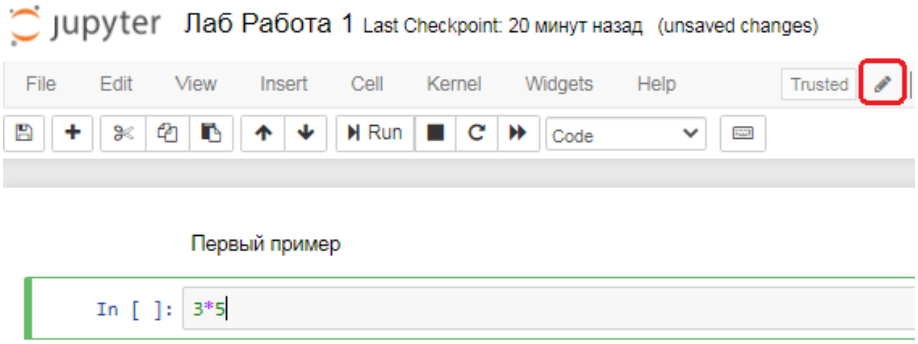
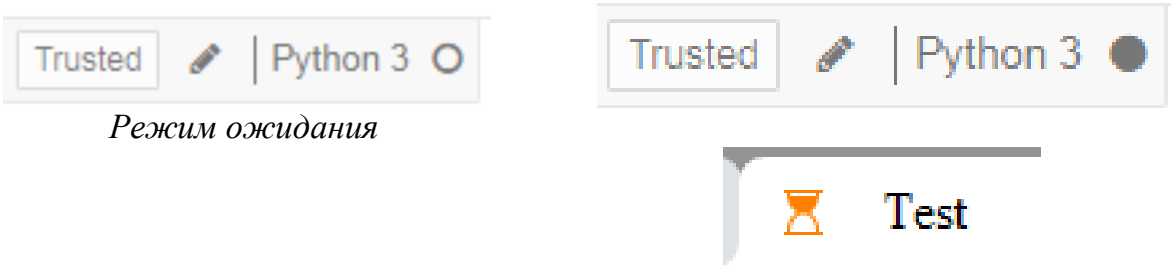


Рис.16 Программа в режиме правки

В самой правой части панели меню находится индикатор загруженности ядра Python. Если ядро находится в режиме ожидания, то индикатор представляет собой окружность (рис.17а). Если оно выполняет какую-то задачу, то изображение измениться на закрашенный круг, а на закладке страницы браузера отображается иконка песочных часов (рис. 17б).



Режим ожидания

Режим работы

а) **б)**

Рис 17. Индикаторы загруженности ядра Python

Если программа зависла, то можно прервать ее выполнение, выбрав на панели меню пункт *Kernel -> Interrupt*.

Для добавления новой ячейки в середине блокнота используйте меню:

`Insert->Insert Cell Above` (выше текущей ячейки)

или

`Insert->Insert Cell Below` (ниже текущей ячейки).

Во втором случае можно также использовать кнопку панели  .

4. Использование Spyder

Spyder - это мощная научная среда, написанная на Python для Python, включающая много функций редактирования, анализа, отладки программ для исследования данных.

Подробное руководство по использованию Spyder можно найти на официальном сайте [2]. Ниже даются основные правила работы.

Вначале необходимо создать проект. Проект может содержать программы всех лабораторных работ. Для создания проекта необходимо выполнить пункт меню

Project – New Project

В открывшемся окне необходимо задать имя проекта *Project Name*, выбрать место размещения файлов проекта *Location* и выбрать тип проекта *Project type* (рис. 18). Будем использовать тип проекта по умолчанию Empty Project.

Для создания новой программы необходимо выполнить пункт меню (рис. 19):

Project – New Project

или нажать комбинацию клавиш *Ctrl-N*.

Новому файлу будет присвоено имя Untitled1.py. Его можно сохранить под другим именем выбрав пункт меню:

File - Save As

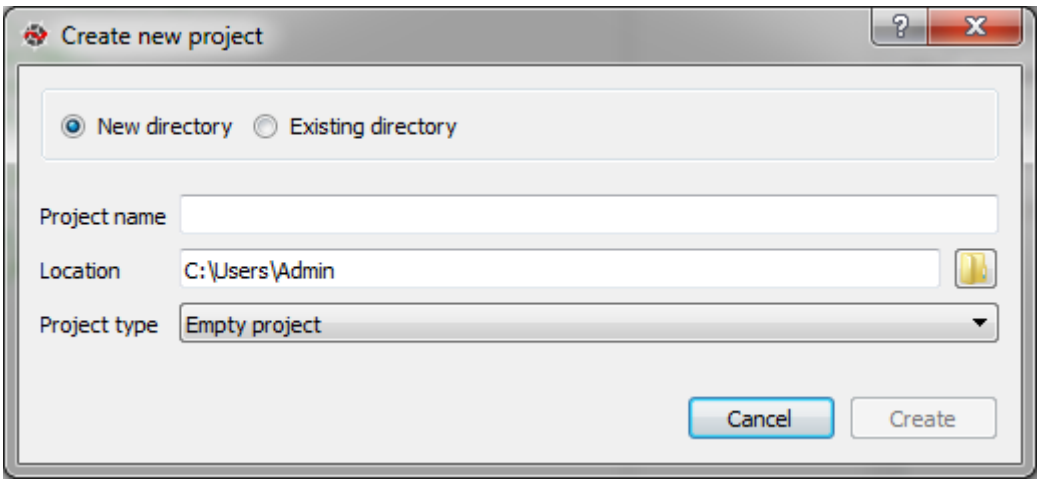


Рис. 18 Создание нового проекта

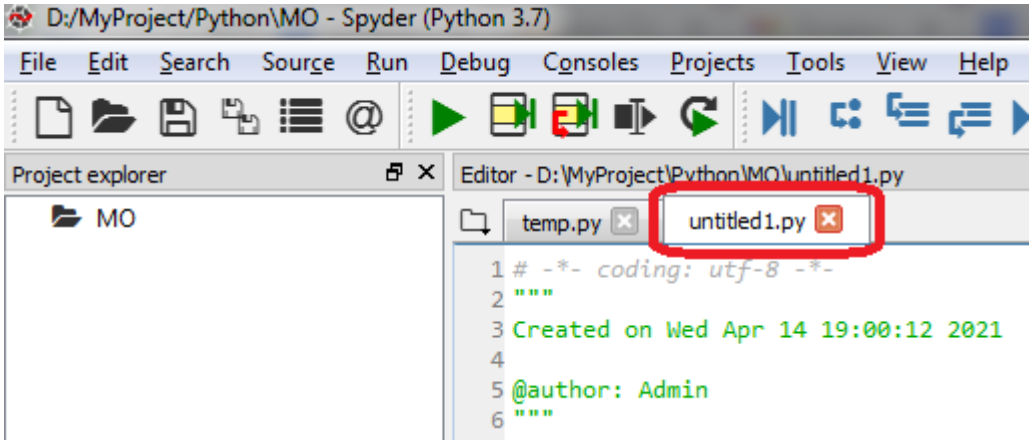


Рис. 19 Создание нового файла (программы)

После ввода программного кода программу запускают на выполнение командой меню

Run – Run

Также можно воспользоваться функциональной клавишей F5 или кнопкой на панели инструментов  .

Результат отобразиться в консоли, находящейся в правой части окна программы (рис. 20).

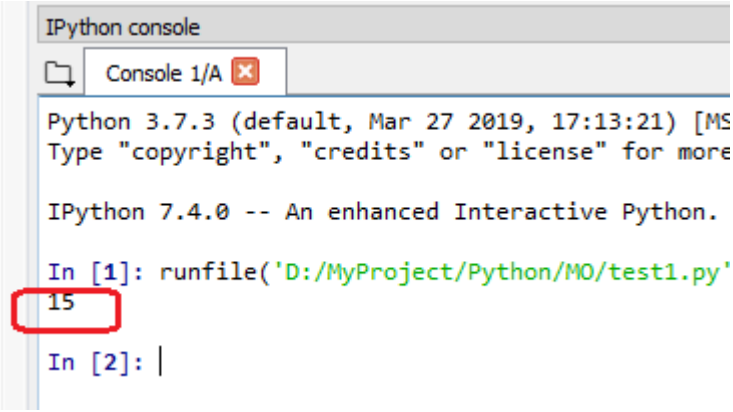


Рис. 20. Окно консоли

Окно консоли можно также использовать для ввода программного кода в интерактивном режиме. Причем после выполнения программы можно использовать переменные этой программы.

Список источников

1. Руководство по использованию Jupyter Notebook.- https://jupyter-notebook.readthedocs.io/en/stable/
2. Руководство по использованию Spyder.- https://docs.spyder-ide.org/current/editor.html

Задание

1. Выбрать и установить рекомендованное программное обеспечение
2. Протестировать установленные программные средства

Порядок выполнения работы

1. Ознакомиться с представленными рекомендациями
2. Выполнить установку необходимого программного обеспечения
3. Протестировать установленное программное обеспечение на небольших тестовых заданиях

Содержание отчета

1. Скриншоты процесса установки
2. Выполнение простых тестовых задач

Практическое занятие № 17
Введение в язык Python

Цель занятия. Изучить основы языка Python. Научиться составлять программы для решения типовых задач.

Краткие теоретические сведения

1. Основы синтаксиса Python

Вначале рассмотрим несколько простых правил записи программ:

– Все действия описываются командами или операторами

– Оператор завершается в конце строки либо точкой с запятой (последний вариант используется, если в строке несколько операторов)

– Большой оператор может занимать несколько строк, но для этого его заключают в круглые, квадратные или фигурные скобки.

– Вложенные блоки операторов управления записываются после двоеточия на новой строке с отступом под строкой основного оператора (стандартно 4 пробела):

```
основной оператор
    Вложенный блок операторов
```

Задания практического задания будем выполнять в интерактивной среде Jupyter Notebook.

2. Данные и выражения. Оператор присваивания

Данные, которые используются в программе, могут быть представлены постоянными значениями (константами) и изменяемыми, которые хранятся в переменных. Переменные обозначаются именами, которые могут содержать латинские буквы, цифры, знака подчёркивания и должны начинаться с буквы.

Данные могут иметь один из следующих типов (перечислим основные):

целое число (int);

вещественное число (float);

логическое (boolean), принимающее 2 противоположных значения: «истина» или «ложь»;

символьное значение (str).

Python использует неявную динамическую типизацию, когда тип переменной определяется присвоенным ей значением.

Например,

```
a = 1     # переменная a будет иметь целый тип,
a = 3.5   # переменная a будет иметь вещественный тип.
```

В примерах был использован оператор присваивания. Оператор задается символом «равно», слева от этого символа записывается имя переменной, а справа - значение, которое мы хотим ей присвоить (задать). Причем значение может определяться выражением. О выражениях будет сказано ниже.

Над данными в программе можно выполнять вычисления с использованием арифметических операций. Арифметические операции записываются также как во многих других языках программирования

```
In [11]: a=7
         b=5

In [12]: a+b # сложение
Out[12]: 12

In [13]: a-b # вычитание
Out[13]: 2

In [14]: a*b # умножение
Out[14]: 35

In [15]: a/b  # деление
Out[15]: 1.4

In [16]: a//b # целая часть от деления
Out[16]: 1

In [17]: a % b # остаток от деления
Out[17]: 2

In [18]: a**2  # возведение в степень
Out[18]: 49
```

Математические функции входят в библиотеку math и перед использованием функции ее необходимо импортировать.

```
In [21]: import math

In [22]: math.sin(a)  # вычисление функции синус
Out[22]: 0.6569865987187891
```

Сложные вычисления записываются в виде выражения, которые включают выполнение арифметических операций над переменными и константами, вычисление функций. Операции выполняются последовательно в порядке их следования в выражении с учетом приоритета. Порядок выполнения операций можно изменить, заключив операцию в скобки.

Например, математическое выражение $\frac{a+b}{b}$ следует в программе записать в виде `(a+b)/b`. Запись вражения `a+b/b` даст совершенно другой результат, так как операция деления имеет более высокий приоритет и будет

выполнена раньше.

```
In [23]: (a+b)/b
Out[23]: 2.4

In [24]: a+b/b
Out[24]: 8.0
```

Пример 1. Вычислить выражение

$$\sqrt{a \cdot \sin\frac{b+c}{16}} + \frac{\sqrt[3]{a+b-3}}{a-b} - 2.1$$

```
In [25]: c=3
         math.sqrt(a*math.sin((b+c)/16))+math.pow(a+b-3,1/3)/(a-b)-2.1
Out[25]: 0.7719749807807816
```

В программном выражении были использованы стандартные функции:

Математическая функция	Функция Python
$\sqrt{x}$	math.sqrt (x)
$\sqrt[3]{x} = x^{1/3}$	math.pow (x, 1/3)
sin (x)	math.sin(x)

3. Операторы ввода и вывода

В предыдущих примерах все значения для переменных задавались явно через операцию присваивания. Часто данные должны быть заданы при выполнении программы. Это позволяет сделать функция клавиатурного ввода `input`. Функции задаются ключевым словом, за которым следуют скобки. Внутри скобок может быть задан аргумент. Когда программа встречает функцию `input`, она приостанавливает свою работу и ожидает, когда пользователь введет данные и нажмет клавишу Enter. Как правило, функция input является правой частью оператора присваивания:

```
In [19]: a=input()
         5

In [20]: a
Out[20]: '5'
```

Обратите внимание, что введенное значение всегда представляется символьным типом. Вводимое значение можно привести к другому типу (если введенное значение допускает это). Например, функция `int`() преобразует введенное значение к целому типу:

```
In [21]: a=int(input())
         5

In [22]: a
Out[22]: 5
```

Для удобства пользователя целесообразно перед вводом вывести строку-приглашение (подсказку, что вводится). Для вывода строки-приглашения используется аргумент функции *input():*

```
In [*]: a=int(input("a="))
        a=
```

Вывод данных выполняется функцией print(). Параметры функции разделяются запятыми и из их значений формируется выводимая строка, в которой отдельные значения разделены пробелами.

Выводимые данные можно отформатировать в соответствии с некоторыми шаблонами. Общий вид форматированного вывода

```
print ("Текст с шаблонами вывода" %(список переменных))
```

Шаблон начинается со знака %. Основные шаблоны:

m**d** - для вывода целого числа (m - число знаков)

m.n**f** - для вывода вещественных чисел (n-число дробных разрядов).

```
In [25]: x=1; y=6.789
         print("x=%4d y=%5.2f"%(x,y))
         x=   1 y= 6.79
```

4. Операторы управления

Все операторы программы выполняются последовательно, составляя так называемый линейный алгоритм выполнения. Естественный порядок выполнения операторов можно изменить с использованием операторов управления. К числу таких операторов относятся условный оператор и оператор цикла.

4.1 Условный оператор

Условный оператор *if* позволяет выполнить блоки операторов в зависимости от некоторого условия. Условие представляет некоторое высказывание, которое может быть истинным (true) или ложным (false). Часто условие задается операцией сравнения, определяющей отношение между двумя величинами:

```
a оператор b
```

где оператором может быть:

== равно

!= не равно

> больше

>= больше или равно

< меньше

<= меньше или равно

Самый простой вариант - оператор с одним условием. После ключевого слова *if* записывается условное выражение. После выражения нужно поставить двоеточие. Если это выражение истинно (true), то выполняются операторы, следующие за данным оператором. Выражение является истинным, если его результатом является число не равное нулю, непустой объект, либо «истинное» логическое выражение.

```
In [26]: if (a>0):
             print ("a - положительное число")

         a - положительное число
```

Часто необходимо предусмотреть альтернативный вариант действий. Т.е. при истинном условии нужно выполнить один блок операторов, а при ложном – другой. Для этого используется вариант с альтернативой *if – else*.

```
In [27]: a=7; b=9;
         if (a>b):
             print ("a больше b")
         else:
             print ("b больше a")

         b больше a
```

Для реализации выбора из нескольких альтернатив можно использовать вариант оператора if - elif - else.

```
k=3
if k==3:
   y=a**k
elif (k<3):
   y=a*k
else:
   y=(b-a)*k
print (y)

343
```

```
k=5
if k==3:
   y=a**k
elif (k<3):
   y=a*k
else:
   y=(b-a)*k
print (y)

10
```

```
k=1
if k==3:
   y=a**k
elif (k<3):
   y=a*k
else:
   y=(b-a)*k
print (y)

7
```

Ранее были рассмотрены простые условия, включающие только операцию сравнения. Условия могут быть составными (сложными), которые кроме операций сравнения включают логические функции:

and - логическое И, дающее «истину» в случае истинности обоих простых условий, а в других случаях - «ложь»;

or - логическое ИЛИ, дающее «ложь» в случае ложности обоих простых условий, а в других случаях - «истину»;

not - логическое НЕ, меняющее значение логического выражения на противоположное.

Например, условие принадлежности переменной X отрезку [a,b] или [c,d] записывается следующим образом:

`x>=a  and x<=b or x>=a  and x<=b.`

Скобки для записи не использованы, так как приоритет **and** выше, чем у **or**.

Пример 2. Вычислить сложную функцию

$$y=\begin{cases} 2x+\sin x, & \text{при } x=0,\ x\le -1 \\ x^3+4, & \text{при } 5\le x\le 12,\ x=20,\ x>80 \\ 2.5x^2+5x-3, & \text{при } x=15,\ 25, 32\le x\le 35 \\ \sqrt{x}, & \text{в остальных случаях} \end{cases}$$

```
In [9]: x=-10
        if x==0 or x<=-5:
            y=2*x+math.sin(x)
        elif 5<=x and x<=12 or x==20 or x>80:
            y=x**3+4
        elif x==15 or x==25 or 32<=x and x<=35:
            y=2.5*x**2+5*x-3
        else:
            y=math.sqrt(x)
        print (y)

        -19.45597888911063
```

Легко проверить, что истинным будет первое логическое выражение и поэтому y=2x+sin x = 2·(-10) + sin(-10) = -19,45

4.2 Оператор цикла

Операторы цикла позволяют записать повторяющиеся действия, где каждое повторение выполняет схожие действия, отличающиеся только значениями данных.

Циклы вводятся двумя операторами:

`for` - цикл по элементам объекта;

`while` - цикл по условию.

При работе с циклами используются операторы **`break`** и **`continue`**. Оператор **`break`** предназначен для досрочного прерывания работы цикла. Оператор **`continue`** переходит к следующей итерации цикла, при этом код, расположенный после данного оператора, не выполняется.

Оператор цикла `for`

Этот цикл проходится по некоторому объекту, состоящему из элементов (диапазону чисел, строке или списку), и во время каждого прохода выполняет тело цикла.

Пример 3. Вывести целые числа от 1 до 5.

Будем использовать функцию диапазон **`range(n,m)`**, где n определяет начальное значение, а конечное равно m-1.

```
In [28]: for i in range(1,6):
             print(i)

         1
         2
         3
         4
         5
```

Пример 4. Вычислить сумму целых чисел от 1 до 100. Числа, которые кратны 5 в сумму не включать

```
In [29]: s=0
         for i in range (1,101):
            if i%5==0: continue
            s+=i
         print(s)

         4000
```

Обратите внимание, что кратность 5 проверяется нулевым остатком от деления i на 5. Если число из диапазона кратно 5, то происходит переход к следующей итерации цикла.

Для накопления суммы используется переменная s, которой вначале присваивается значение 0. В цикле на каждой итерации к сумме добавляется очередное слагаемое, каковым является число `i` из диапазона. Формула накопления суммы (добавления нового слагаемого к уже накопленной сумме):

```
s = s +i
```

В программе использована сокращенная запись оператора присваивания. Если слагаемое добавляется к переменной и присваивается ей же, то эту переменную справа от знака равно не указывают, а операцию суммирования ставят перед знаком «равно».

```
s += i
```

Пример 5. Вычислить сумму всех целых чисел от 1 до 100. Вычисления производить пока сумма не превысит 120

```
In [30]: s=0
         for i in range (1,101):
            s+=i
            if s>120: break
         print(s)

         136
```

В данной программе для прекращения вычислений использован оператор `break`.

Оператор цикла `while`

Цикл выполняется до тех пор, пока условие цикла истинно

```
while (условие):
    блок цикла.
```

Составим программу примера 5 с использованием `while`.

```
In [35]: s=0; i=1
         while s<=120:
            s+=i
            i+=1
         print(s)

         136
```

Рассмотрим более сложный пример, когда нужно вычислить выражения включающие суммы и произведения

Пример 6. Вычислить выражение.

$$\sum_{i=1}^{N}\frac{2i+p}{i+4}\Bigg/\left(\sum_{i=1}^{N}\frac{i^2+p^2}{2i+1}+\prod_{k=1}^{M}\frac{k}{4+k}\right)$$

Знаком $\sum$ обозначается сумма, а знаком $\prod$ - произведение. Члены суммы и произведения (слагаемые и сомножители) зависят от целочисленных переменных-параметров. Под знаком суммы и произведения задано начальное значение параметра, а над знаком - конечное.

```
In [41]: n=int(input("n="));
         m=int(input("m="));
         p=int(input("p="));
         s1=0;s2=0;p1=1;
         for i in range (1,n+1):
             s1+=(2*i+p)/(i+p)
             s2+=(i**2+p**2)/(2*i+1)
         for k in range (1,m+1):
             p1*=k/(4+k)
         y=s1/(s2+p1)
         print("y=%6.2f"%y)

         n=3
         m=2
         p=4
         y=  0.30
```

Проверим результат, выполнив ручной расчет:

$$S_1=\frac{2\cdot1+4}{1+4}+\frac{2\cdot2+4}{2+4}+\frac{2\cdot3+4}{3+4}=1.2+1.33+1.43=3.96$$

$$S_2=\frac{1^2+4^2}{2\cdot1+1}+\frac{2^2+4^2}{2\cdot2+1}+\frac{3^2+4^2}{2\cdot3+1}=5.67+4+3.57=13.24$$

$$P_1=\frac{1}{4+1}\cdot\frac{2}{4+2}=0.2\cdot0.33=0.07$$

$$y=\frac{3.96}{13.24+0.07}=0.30$$

5. Функции

Фрагменты программного кода, которые могут быть использованы в различных местах программы, принято представлять в виде функций. Функцией называют именованный фрагмент программного кода, к которому можно обратиться из другого места. Как правило, функции создаются для работы с данными, которые передаются ей в качестве аргументов. Функция может формировать результирующее значение, которое возвращается в место вызова этой функции.

Для создания функции используется ключевое слово **`def`**, после которого указывается имя и список аргументов в круглых скобках. В конце заголовка функции ставится двоеточие и на следующей строке с отступом записывается тело функции.

```
def имя(аргументы):
    тело функции
```

Если функция не имеет аргумента, то записываются пустые скобки.

Возврат значения функцией осуществляется с помощью ключевого слова **`return`**, после которого указывается возвращаемое значение.

Для вызова функции необходимо указать имя функции и в скобках поставить значения ее аргументов.

Оформим программу примера 6 в виде функции.

```
In [43]: def expr(n,m,p): # Описание функции
             s1=0;s2=0;p1=1;
             for i in range (1,n+1):
                 s1+=(2*i+p)/(i+p)
                 s2+=(i**2+p**2)/(2*i+1)
             for k in range (1,m+1):
                 p1*=k/(4+k)
             return s1/(s2+p1) # конец описания
         n=int(input("n="));
         m=int(input("m="));
         p=int(input("p="));
         print("y=%6.2f"%expr(n,m,p))

         n=3
         m=2
         p=4
         y=  0.30
```

Задание

1. Вычислить сложную функцию по одному из заданных индивидуальных вариантов. Оформить вычисление в виде функции. Вызвать функцию для четырех значений аргумента. Значения аргумента подобрать так, чтобы были использованы все варианты сложной функции. Привести ручной расчет для всех значений аргумента.

Варианты задания

№ п/п	Сложная функция
1	$y=\begin{cases}3-\sin x, & \text{при } x=0,\ 2\le x\le 4\\ \frac{x}{2}+1, & \text{при } 10\le x\le 27, x>115\\ x-2, & \text{при } x<-7, x=-1,\ 5.5,\ 40\le x\le 53\\ x^2, & \text{в остальных случаях}\end{cases}$
2	$y=\begin{cases}3\cos x, & \text{при } x=0,\ 2\le x\le 4,\ x=6\\ \frac{x}{4}-5, & \text{при } x=12,\ 30\le x\le 37.5\\ 2x+1, & \text{при } x<0, x=90,\ 105.5,\ 150\le x\le 159\\ x, & \text{в остальных случаях}\end{cases}$
3	$y=\begin{cases}\log_2 x, & \text{при } x=4,8,\ 20\le x\le 65\\ x^2+\frac{1}{x}, & \text{при } x=12,\ 10\le x\le 17\\ 7x-1, & \text{при } x<3,\ x=5,\ x\ge 100\\ \sqrt{x}, & \text{в остальных случаях}\end{cases}$
4	$y=\begin{cases}3x+2\cos x, & \text{при } x=0,\ 3\le x\le 5\\ \frac{x}{2}+7, & \text{при } 9\le x\le 12, x=18,\ x>70\\ 30-x, & \text{при } x<0,\ x=2,\ 25\\ 1, & \text{в остальных случаях}\end{cases}$
5	$y=\begin{cases}x+3, & \text{при } 15\le x\le 25\\ 2x-5, & \text{при } 1\le x\le 10,\ x=12, x>90\\ 10x^2+5x-3, & \text{при } x=-4,-1,\ 47\le x\le 50,\ x=60\\ 20-x, & \text{в остальных случаях}\end{cases}$

2. Вычислить выражение, включающее суммы и произведения, по одному варианту из представленных в таблице. Вычисления оформить в виде функции. Привести ручной расчет контрольного примера.

Варианты задания

№ пп	Выражение	Данные контрольного примера
1	$\left(\sum_{i=1}^{N}\frac{i^2+1}{i+2}+\prod_{j=1}^{M}\frac{2j+1}{j^2+1}\right)\Big/\sum_{k=1}^{L}\frac{2k^2+1}{k+3}$	N=2; M=2; L=3
2	$\left(\sum_{i=1}^{N}\frac{i^2+p}{i+2}\prod_{j=1}^{M}\frac{2p+i}{j^2+i}\right)$	N=2; M=2; p=3

3	$\left(\prod_{i=1}^{N}\frac{i^2+2}{i^2+1}+\sum_{j=1}^{M}\frac{i^2+1}{i+2}\right)\Big/\sum_{k=1}^{N}\frac{k\cdot p}{k+p}$	N=2; M=3;p=2
4	$\left(\sum_{i=1}^{N}\frac{i+p}{2i+1}+\prod_{k=1}^{M}\frac{2k}{2+k}\right)\Big/\sum_{i=1}^{N}\frac{i+p}{i\cdot p}$	N=2; M=3;p=2
5	$\left(\prod_{k=1}^{M}\frac{k+M}{2k+1}+\sum_{j=1}^{N}\frac{i^2}{i+2}\right)\Big/\sum_{i=1}^{N}\frac{i\cdot p}{p+2i}$	N=2; M=3;p=2

Порядок выполнения работы

1. Изучить теоретический материал
2. Составить программу по одному варианту из каждого задания
3. Выполнить ручной расчет
4. Выполнить расчет по программе и сравнить результаты

Содержание отчета

1. Программы по выбранным вариантам
2. Ручной расчет на контрольных данных
3. Результаты расчетов по программам

Практическое занятие № 18
Вектора и матрицы

Цель занятия. Изучить базовые понятия вектора и матрицы. Научиться выполнять основные операции с векторами и матрицами. Изучить представление векторов и матриц и выполнение основных операций над ними в программах на языке Python

Краткие теоретические сведения

1. Понятие матрицы и вектора

Числовой матрицей называют прямоугольную таблицу *чисел*. Строки и столбцы нумеруют целыми числами от единицы. Обозначим число строк *M* и число столбцов *N*. Матрицы принято обозначать заглавными латинскими прямыми жирными буквами, например, **A**, **M**, **Q**. Элемент матрицы **A** на пересечении *k*-й строки и *j*-м столбца обозначим $a_{k,j}$ или `a(k,j)`.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Всегда на первом месте стоит индекс, нумерующий строки матрицы, на втором месте – индекс, нумерующий столбцы матрицы. Запись $\mathbf{A}_{MN}$ используется, если необходимо в явном виде указать число строк и столбцов в матрице.

Матрицы бывают прямоугольными и квадратными. У прямоугольной матрицы число строк и столбцов различаются, а у квадратной - совпадают.

Матрица называется **нулевой**, когда все ее элементы равны нулю.

Главной диагональю квадратной матрицы называется диагональ, проведённая из левого верхнего угла матрицы в правый нижний.

Диагональная матрица — это квадратная матрица, все элементы которой, стоящие вне главной диагонали, равны нулю.

$$D = \begin{bmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & d_{nn} \end{bmatrix}$$

Частным случаем диагональной матрицы является единичная.

$$E = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Вектором называют частный случай матрицы, у которой одна строка или один столбец. Соответствующий вектор называют вектором – строкой и вектором – столбцом.

На языке Python вектора и матрицы представляются списками. Вектор можно определить простым списком чисел:

```
a = [1, 3, 7, 10]
```

Нумерация элементов начинается с нуля. Для обращения к элементу списка указывается его имя и в квадратных скобках его номер.

```
print (a[1])=> 3
```

Матрицу на Python можно определить вложенным списком:

```
a = [[1, 3, 7, 10],[2,5,11,9]]
```

Данный оператор определяет матрицу, состоящую из 2 строк и 4 столбцов.

Для обращения к элементу вложенного списка (матрице) вначале указывается номер строки, а затем номер столбца

```
print (a[1][2])=> 11
```

Указание только первого индекса определяет строку

```
print (a[1])=> [2,5,11,9]
```

Более удобно с векторами и матрицами работать с использованием библиотеки `numpy`, которая будет рассмотрена на следующих занятиях.

https://pythontutor.ru/lessons/2d_arrays/

https://silvertests.ru/GuideView.aspx?id=32174

Для обработки матриц используются вложенные циклы.

```
for i in range(N):
    for j in range(M):
       операция, например, print(A[i][j])
```

2. Алгебраические операции с матрицами

Перечислим основные операции с матрицами, векторами и числами и покажем их на примерах.

2.1 Бинарные операции

В бинарных операциях участвуют 2 операнда: матрицы, скаляры.

2.1.1 Умножение / деление матрицы на число

Для выполнения данной операции все элементы матрицы умножают / делят на число.

Пример 1.

$$\begin{pmatrix} -2 & 3{,}5 & 0 \\ 1 & -0{,}8 & 7 \end{pmatrix} \cdot 3 = \begin{pmatrix} -6 & 10{,}5 & 0 \\ 3 & -2{,}4 & 21 \end{pmatrix}$$

Порядок сомножителей не имеет значения.

Составим программу на Python

```
In [20]: a=[[-2,3.5,0],[1,-0.8,7]]
z=3
n=2;m=3;
for i in range(n):
    for j in range (m):
        a[i][j]=a[i][j]*z
print (a)
```

```
[[-6, 10.5, 0], [3, -2.4, 21]]
```

Создадим функцию вычисление произведения матрицы на скаляр:

```
def mscal(n,m,a,z):
    for i in range(n):
        for j in range(m):
            a[i][j]=a[i][j]*z
    return a
```

При вызове функции необходимо задать 4 параметра: размеры матрицы (n, m), матрицу и скаляр. Зададим исходные данные как в предыдущем примере и вызовем функцию:

```
a=[[-2,3.5,0],[1,-0.8,7]]
n=2;m=3;z=3;
```

```
print(mscal(n,m,a,z))
```

```
[[-6, 10.5, 0], [3, -2.4, 21]]
```

2.1.2 Сложение / вычитание матриц

Операции сложения и вычитания определены только для матриц одинакового размера. Операции сложения/вычитания выполняются поэлементно.

Пример 2.

$$\begin{pmatrix} -2 & 3{,}5 & 0 \\ 1 & -0{,}8 & 7 \end{pmatrix} + \begin{pmatrix} -6 & 10{,}5 & 0 \\ 3 & -2{,}4 & 21 \end{pmatrix} = \begin{pmatrix} -8 & 14 & 0 \\ 4 & -3{,}2 & 28 \end{pmatrix}$$

Составим программу на Python:

```
n=2;m=3;
# Исходные матрицы
a=[[-2,3.5,0],[1,-0.8,7]]
b=[[-6,10.5,0],[3,-2.4,21]]
# Подготовка матрицы для результата
c = [[0]*m for i in range(n)]
# Вычисление суммы матриц
for i in range(n):
    for j in range (m):
        c[i][j]=a[i][j]+b[i][j]
print (c)
```

```
[[-8, 14.0, 0], [4, -3.2, 28]]
```

Результат сложения матриц будем записывать в новую матрицу. Предварительно создадим матрицу для результата такого же размера, что и исходные матрицы, и заполним ее нулями.

2.1.3 Правила раскрытия скобок

В данной операции участвуют сумма (вычитание) матриц и числовой множитель.

$a\ (\mathbf{A}_{MN} + \mathbf{B}_{MN}) = a\ \mathbf{A}_{MN} + a\ \mathbf{B}_{MN}$, a – числовой множитель.

2.1.4 Скалярное произведение вектора-строки на вектор-столбец

Выполняется поэлементное перемножение элементов векторов, после чего произведения складываются. Произведение называют скалярным, поскольку его результат – число, т.е. скаляр.

Пример 3.

$$\begin{pmatrix}2{,}5 & -1 & 3\end{pmatrix}\cdot\begin{pmatrix}-2\\0{,}5\\2\end{pmatrix}=2{,}5\cdot(-2)+(-1)\cdot 0{,}5+3\cdot 2=0{,}5$$

Составим программу на Python:

```
n=3
v1=[2.5,-1,3]
v2=[-2,0.5,2]
sp=0
for i in range(n):
    sp+=v1[i]*v2[i]
print(sp)
```

```
0.5
```

2.1.5 Произведение матриц:

Операция определена для матриц с определенным соотношением между их размерами:

$\mathbf{A}_{MK} \cdot \mathbf{B}_{KN} = \mathbf{C}_{MN}$,

Т.е. число столбцов матрицы – 1-го сомножителя должно быть равно числу строк матрицы – 2-го сомножителя. Строки 1-й матрицы умножаются на столбцы 2-й матрицы по правилам скалярного произведения. Результаты умножения (числа) формируют результирующую матрицу.

Пример 4.

$$\begin{pmatrix}3 & 2 & 1 & 4\\7 & 5 & 6 & 0\end{pmatrix}\cdot\begin{pmatrix}2 & 0 & 3\\4 & 5 & 1\\3 & 2 & 4\\1 & 6 & 2\end{pmatrix}=$$

$$=\begin{pmatrix}3\cdot 2+2\cdot 4+1\cdot 3+4\cdot 1 & 3\cdot 0+2\cdot 5+1\cdot 2+4\cdot 6 & 3\cdot 3+2\cdot 1+1\cdot 4+4\cdot 2\\7\cdot 2+5\cdot 4+6\cdot 3+0\cdot 1 & 7\cdot 0+5\cdot 5+6\cdot 2+0\cdot 6 & 7\cdot 3+5\cdot 1+6\cdot 4+0\cdot 2\end{pmatrix}=$$

$$=\begin{pmatrix}21 & 36 & 23\\52 & 37 & 50\end{pmatrix}$$

Составим программу на Python:

```
n=2;m=4;p=3;
# Исходные матрицы
a=[[3,2,1,4],[7,5,6,0]]
b=[[2,0,3],[4,5,1],[3,2,4],[1,6,2]]
# Подготовка матрицы для результата
c = [[0]*k for i in range(n)]
# Вычисление произведения матриц
for i in range(n):
    for j in range(k):
        c[i][j]=0
        for k in range(m):
            c[i][j]+=a[i][k]*b[k][j]
print (c)
```

```
[[21, 36, 23], [52, 37, 50]]
```

2.2 Унарные операции

Унарная операция всегда применяется только к одной матрице

2.2.1 Транспонирование матрицы

Транспонирование - это такое преобразование матрицы, при котором строки матрицы переписываются столбцами преобразованной матрицы.

Пример 5.

$$\begin{pmatrix} 2 & -3 & 0 & 1 \\ 4 & 5 & -2 & 3 \\ -4 & 0 & 1 & 6 \end{pmatrix}^T = \begin{pmatrix} 2 & 4 & -4 \\ -3 & 5 & 0 \\ 0 & -2 & 1 \\ 1 & 3 & 6 \end{pmatrix}$$

Применение операции транспонирования к вектору – строке превращает его в вектор-столбец. Повторное применение операции транспонирования восстанавливает матрицу и приводит к исходному виду.

Составим программу на Python:

```
n=3;m=4;
# Исходная матрица
a=[[2,-3,0,1],[4,5,-2,3],[-4,0,1,6]]
# Подготовка матрицы для результата
at = [[0]*n for i in range(m)]
# Транспонирование
for i in range(n):
    for j in range(m):
        at[j][i]=a[i][j]
print (at)
```

```
[[2, 4, -4], [-3, 5, 0], [0, -2, 1], [1, 3, 6]]
```

2.2.2 Обращение матрицы A

Операция обращения матрицы заключается в подборе такой матрицы **Ainv**, умножение которой на исходную матрицу дает единичную матрицу **E**

A · Ainv = E

Реализация операции взятия обратной матрицы программным способом на языке Python будет рассмотрена в работе по использованию библиотеки `numpy`.

2.2.3 Определитель (детерминант)

Определителем квадратной матрицы является число, получаемое в результате определенных преобразований квадратной матрицы, которое используется, например, при решении системы линейных уравнений.

Пример 6

На рисунке показаны результаты вычисления обратной матрицы Ainv для матрицы A, перемножения матриц A и Ainv, и определителя DA матрицы A в программе Excel. Вычисления выполнены автоматически с помощью функций для матричных операций МОБР, МУМНОЖ, МОПРЕД. Правила пользования этими функциями с примерами приведены в Справке к Excel.

	A	B	C	D	E	F	G	H
1								
2								
3		2	-5	6		-0,067	-0,225	-0,048
4	A =	3	0	-2	Ainv =	0,0526	0,1053	-0,105
5		4	7	1		-0,1	0,1627	-0,072
8		-1	0	-1,11E-16				
9	A*Ainv =	-5,55E-17	-1	-2,78E-17	DA =	209		
10		-2,78E-17	2,78E-17	-1				

Рисунок 1. Обращение, умножение матриц, вычисление определителя

Заметим, что при обращении матрицы и перемножении матриц накапливается погрешность вычислений. Поэтому в матрице A · Ainv некоторые значения близки к нулю, например, -1,11E-16, но не равны точно нулю, как это должно быть для единичной матрицы.

Программная реализация вычисления определителя также будет рассмотрена позже.

3. Длина (модуль) вектора

Скалярное произведение вектора-строки на его транспонированный вектор принимается как квадрат длины вектора $\mathbf{A} = (a_1, \ldots, a_n)$ $\mathbf{A}^T = (a_1, \ldots, a_n)^T$: $|\mathbf{A}|^2 = a_1^2 + a_2^2 + \ldots + a_n^2$. Если рассматривать элементы вектора как его проекции на оси декартовой системы координат n-мерного Евклидова пространства, то по теореме Пифагора полученная сумма квадратов координат и есть квадрат длины вектора.

Пример 7.

$$|A|^2 = (3,5,2,7) \cdot \begin{pmatrix} 3 \\ 5 \\ 2 \\ 7 \end{pmatrix} = 3^2 + 5^2 + 2^2 + 7^2 = 9 + 25 + 4 + 49 = 87$$

$$|A| = \sqrt{87} = 9{,}33$$

Составим программу на Python:

```
import math
n=4
a=[3,5,2,7]
da=0
for i in range(n):
    da+=a[i]*a[i]
da=math.sqrt(da)
print(da)
```

```
9.327379053088816
```

4. Геометрическое представление объектов и операций линейной алгебры.

Числа, векторы и матрицы имеют наглядное соответствие с геометрическими объектами и преобразованиями. Ограничимся рассмотрением этого соответствия в двумерном пространстве, т.е. на плоскости. Полученные представления в большинстве случаев без особых трудностей распространяются и на пространства большей размерности.

Построим на плоскости систему декартовых координат в виде двух перпендикулярных числовых осей с началом отсчета в точке их пересечения. Выберем направления осей так, чтобы положительное направление оси X переходило в положительное направление оси Y поворотом на 90 градусов против часовой стрелки. В этой системе координат любая точка плоскости однозначно задается своими координатами (x; y). Обратим внимание, что это пара – упорядоченная, т.е., например, пара чисел (4; 7) и (7; 4) соответствуют разные точки плоскости. Упорядоченные пары чисел – это векторы из двух элементов.

Вектор изображается на плоскости направленным отрезком (стрелкой), выходящей из начала координат и заканчивающейся в точке, задаваемой элементами вектора. Каждый элемент вектора – это его проекция на соответствующую числовую ось.

В рассматриваемом представлении вектор-строка и вектор-столбец имеют одинаковое геометрическое представление. Длина вектора, по теореме Пифагора, равна квадратному корню из скалярного произведения вектора самого на себя.

Косинусы углов между положительными направлениями осей координат и направлением вектора называют направляющими косинусами.

Они равны:

$$\mathrm{Cos}\,\alpha_1 = \frac{a_1}{\sqrt{\mathbf{A}\cdot\mathbf{A}^T}} \qquad \mathrm{Cos}\,\alpha_2 = \frac{a_2}{\sqrt{\mathbf{A}\cdot\mathbf{A}^T}}, \quad \alpha_1 + \alpha_2 = \frac{\pi}{2}$$

Умножение вектора на число *C* ведет к изменению длины вектора в *C* раз при сохранении направления. Если числовой множитель – отрицательное число, то вектор меняет направление на противоположное.

Умножение вектора на матрицу (или матрицы на вектор) дает новый вектор. Если матрица квадратная, то размерность вектора – результата будет та же, что и у вектора – сомножителя. Поэтому результат тоже можно изобразить на плоскости. При этом результирующий вектор в общем случае будет отличаться и длиной, и направлением от вектора – сомножителя.

Задание

Все матричные операции в указанных ниже заданиях представьте в виде функций. Задайте исходные данные и выполните программы. Оцените правильность полученных результатов.

1. Создайте на Python матрицы `A` и `B` одинакового размера **`n`** x **`m`** и выполните операции их сложения и вычитания
2. Выполните умножение матрицы `A` на скаляр `z`.
3. Создайте на Python матрицу `A` размера **`n`** x **`m`** и матрицу B размера **`m`** x **`k`**. Выполните умножение матриц

Порядок выполнения работы

1. Изучить теоретический материал
2. Составить программы по всем пунктам задания
3. Придумать матрицы и выполнить ручной расчет матричных операций
4. Выполнить расчет по программе и сравнить результаты

Содержание отчета

1. Программы по всем пунктам задания
2. Ручной расчет на контрольных данных
3. Результаты расчетов по программе

Практическое занятие № 19
Установка дополнительных библиотек в среде Python

Цель занятия. Научиться устанавливать дополнительные библиотеки в среде Python через платформу Anaconda.

Краткие теоретические сведения

При разработке программ важно настроить среду разработки для выполнения какого-либо проекта. При работе с языком программирования Python необходимо настроить виртуальную среду окружения (environment), которая является средой хранения и использования дополнительных пакетов и библиотек, необходимых для работы над программным проектом.

Для решения задач анализа и машинного обучения потребуются дополнительные библиотеки Python, которых нет в установленной среде Python. Такие библиотеки могут быть добавлены через платформу Anaconda.

При установке Anaconda автоматически создается стандартное окружение `base (root)`. Можно использовать это окружение и устанавливать библиотеки в него либо создать новое окружение.

Для создания нового окружения необходимо перейти панель Environment и нажать кнопку Create (создать), как показано на рисунке 1.

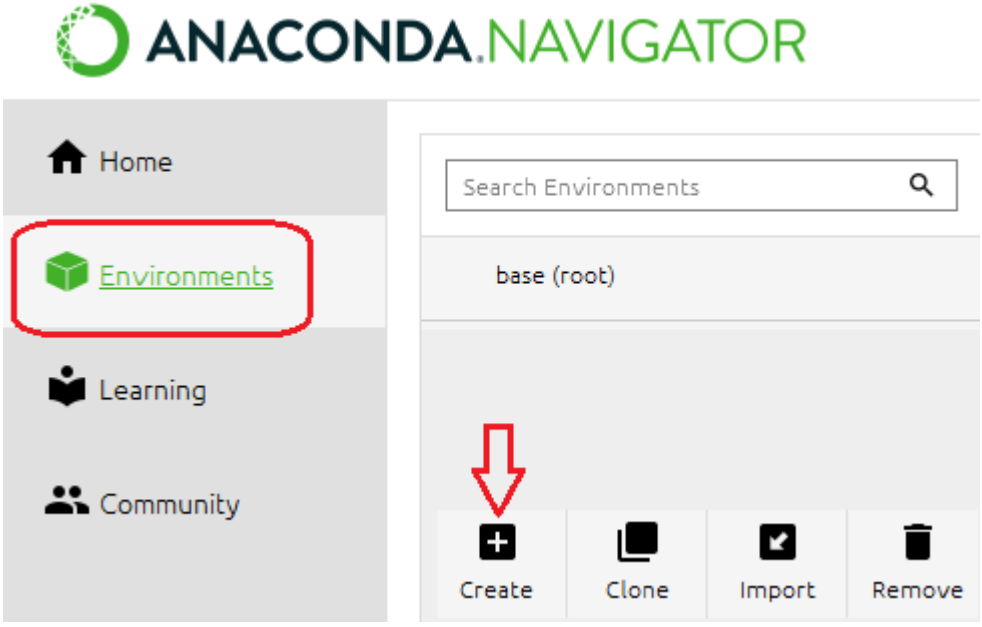


Рис. 1 Создание нового окружения

В открывшемся окне необходимо задать (рис. 2):

1- имя окружения Name (например, bigdata_env),

2 - установить флажок напротив языка, который будет использоваться (выберем Python),

3 - выбрать версию языка (выберем последнюю 3.8)

4 - нажать кнопку Create для запуска процесса создания окружения

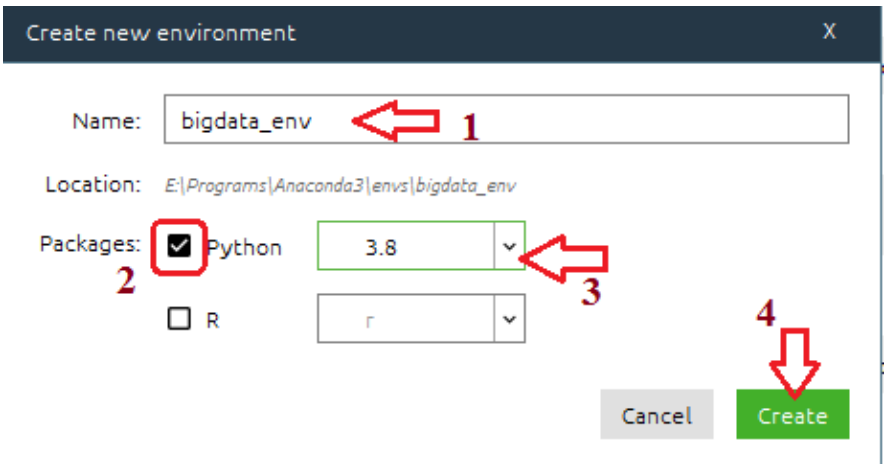


Рис. 2 Задание параметров окружения

После завершения процесса создания новое окружение будет добавлено в список доступных в среде Anaconda (рис. 3).

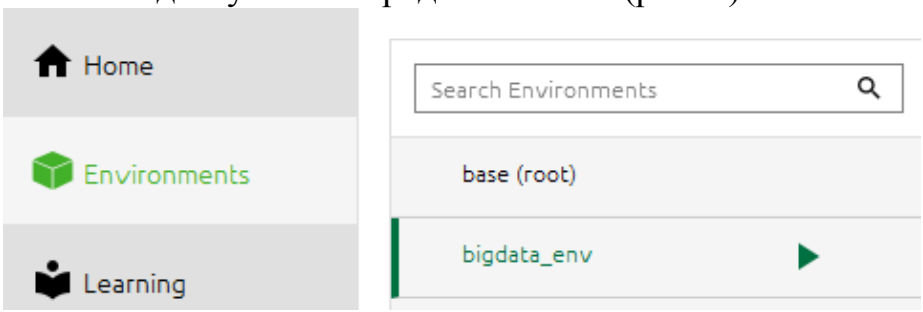


Рис.3 Список доступных окружений среды Python

После создания окружения потребуется повторно установить необходимые приложения (в частности Jupyter Notebook), перейдя на вкладку Home и нажав кнопку Install (рис.4).

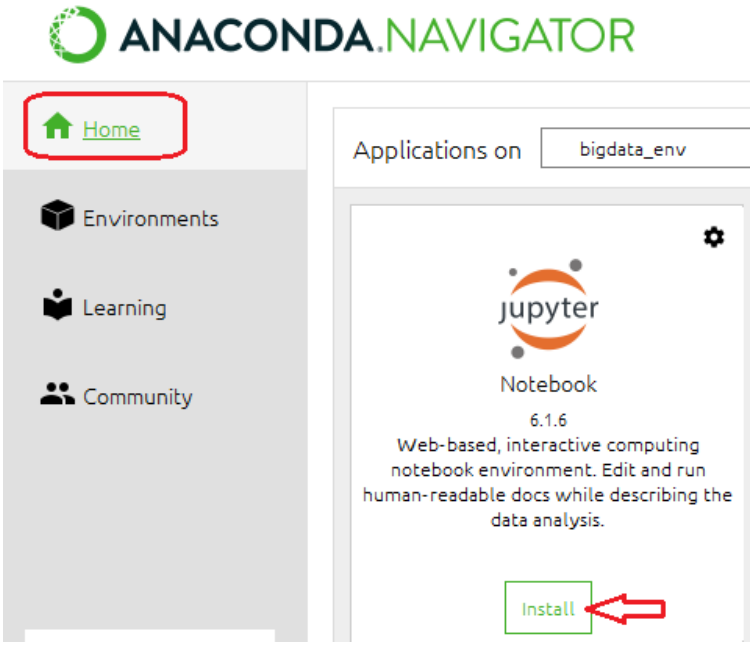


Рис.4 Установка приложения Jupyter Notebook в новое окружение

Дополнительные библиотеки будем устанавливать в новое окружение. И в дальнейшей работе перед запуском Jupyter Notebook или другого приложения предварительно необходимо переключаться на данное окружение (рис.5).

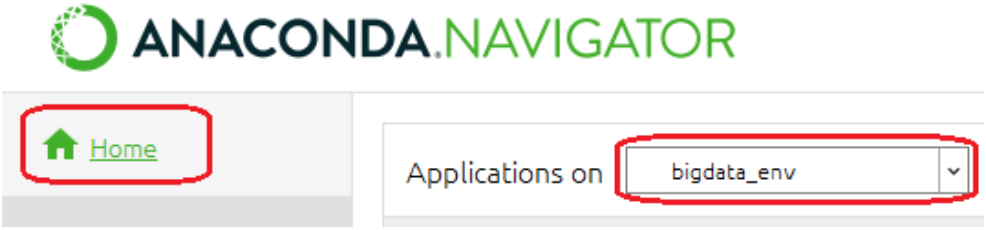


Рис.5 Выбор окружения перед запуском Jupyter Notebook

На данном занятии рассмотрим методику установки дополнительных библиотек в среде окружения Python и используем ее на примере установки следующих библиотек:

`numpy` - библиотека высокоуровневых математических функций для работы с многомерными массивами;

`pandas` - библиотека высокоуровневых функций для работы со специальными структурами данных и для манипулирования числовыми таблицами;

`matplotlib` - библиотека визуализации данных средствами двумерной графики.

Установка библиотек через платформу Anaconda выполняется с использованием Anaconda.Navigator. Необходимо запустить данный инструмент через меню Пуск и выполнить следующую последовательность шагов:

1) выбрать закладку Environments;
2) выбрать окружение (Environment) в котором необходимо установить библиотеку;
3) выбрать элемент списка Not Installed (не установленные библиотеки);
4) в окне поиска ввести название библиотеки;
5) в найденном списке отметить нужную библиотеку
6) запустить процесс установки, нажав кнопку Apply

Выполним указанные действия вначале для установки библиотеки `numpy` (рис. 6).

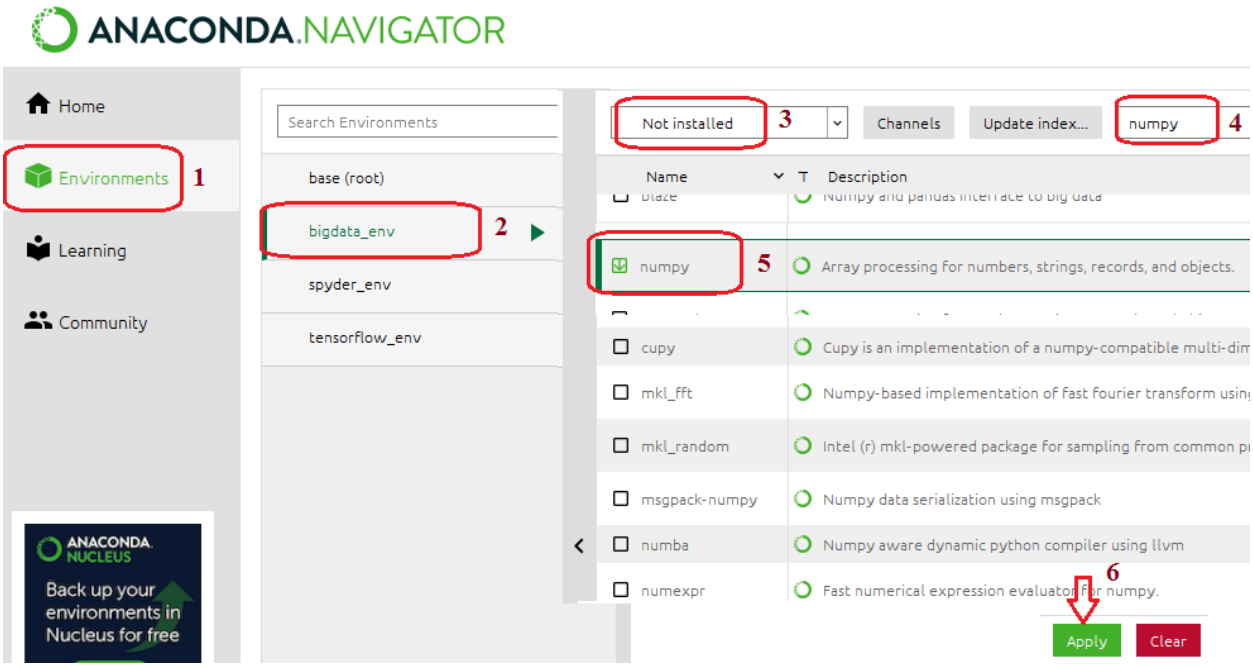


Рис. 6 Установка библиотеки numpy в Anaconda.Navigator

Будет выполнен поиск библиотеки и связанных с ним пакетов и предложено их установить (рис.7). Согласимся и нажмем кнопку `Apply` .

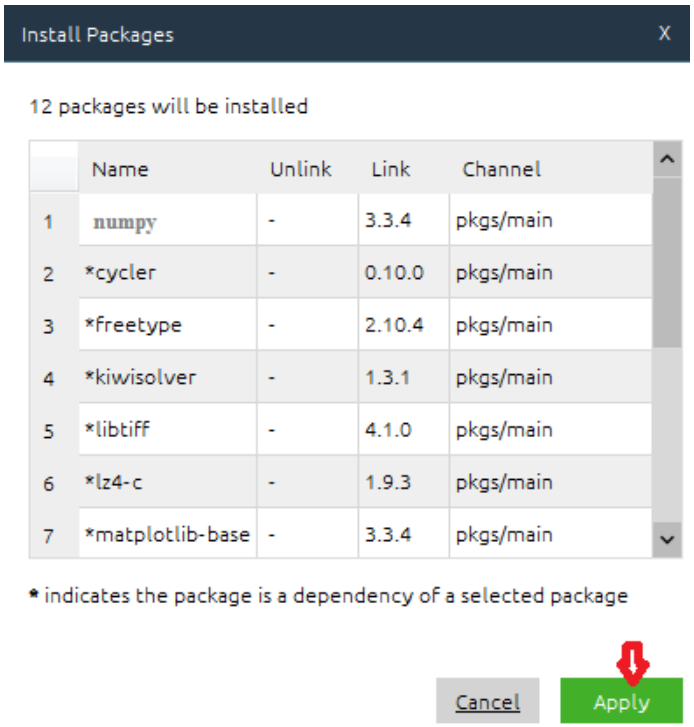


Рис. 7 Список найденных библиотек для установки

Повторим те же действия для библиотек pandas и matplotlib.

Задание

Установить рекомендованные пакеты и библиотеки

Порядок выполнения работы

1. Ознакомиться с представленными рекомендациями
2. Выполнить установку необходимых пакетов и библиотек
3. Протестировать установленные пакеты и библиотеки на примере вывода версий библиотек в режиме блокнота (`имя_библиотеки.__version__`)

Содержание отчета

1. Скриншоты процесса установки
2. Результаты выполнения тестовых задач

Практическое занятие № 20
Библиотека numpy. Операции над массивами

Цель занятия. Изучить основные функции библиотеки numpy и научиться их использовать при выполнении операций с многомерными массивами

Краткие теоретические сведения

Библиотека numpy очень эффективна для численных вычислений. Перед использованием библиотеки ее необходимо импортировать в программу:

```
import numpy as np
```

Имя после «as» используется в качестве краткого псевдонима библиотеки при вызове ее функций. То есть одинаковыми будут вызовы функций:

`numpy.функция` и `np.функции`

но вторая запись более краткая. Именно этот псевдоним будем использовать для вызова функций библиотеки на практических занятиях.

Можно проверить версию библиотеки:

```
print("Версия numpy:",np.version.version)
```

```
Версия numpy: 1.16.2
```

Массивы

Многомерные массивы numpy или тензоры являются основной структурой данных, используемой в машинном обучении.

Создание массива

Массив создается методом (функцией) `array`:

`np.array(список)`

Тензоры отличаются по количеству измерений (*рангу*). Ранг тензора можно проверить через свойство `ndim`. Основные виды тензоров:

- скаляр (ранг 0), например `a=np.array(15);`
- вектор (ранг 1), например `b= np.array([11, 5, 7, 15, 8]) ;`
- матрица (ранг 2), например `c=np.array([[3, 5, 1], [2, 7, 4]]);`

```
a=np.array(15);                 # скаляр
print("ранг a=",a.ndim);

b=np.array([11, 5, 7, 15, 8]);  # вектор
print("ранг b=",b.ndim);

c=np.array([[3, 5, 1],          # матрица
            [2, 7, 4]]);
print("ранг c=",c.ndim);
```

```
ранг a= 0
ранг b= 1
ранг c= 2
```

Все элементы массивов имеют значения одного типа данных. Основными типами являются числовые:

- целые `int32` и `int64`
- вещественные `float32` и `float64`

Тип назначается по умолчанию по типу записываемых элементов. Тип элементов массива можно определить через свойство `dtype`. Тип можно задать явно при создании массива, используя дополнительный параметр конструктора `dtype`.

```
a=np.array([1,2,4.])
b=np.array([[1,3,6],[1,8,0]],dtype='int64')
print ("тип a:",a.dtype)
print ("тип b:",b.dtype)
```

```
тип a: float64
тип b: int64
```

Массивы кроме ранга и типа имеют еще несколько свойств:

- `itemsize` - объем памяти, занимаемой одним элементом массива;
- `nbytes` - объем памяти, занимаемой массивом;
- `shape` - форма - длина массива (количество элементов) по каждому измерению.

```
a=np.array([1,2,4.])
b=np.array([[1,3,6],[1,8,0]],dtype='int64')
print ("a - размер 1 эл.:",a.itemsize,"байт  размер масс.:",a.nbytes, "байт  форма:", a.shape)
print ("b - размер 1 эл.:",b.itemsize,"байт  размер масс.:",b.nbytes, "байт  форма:", b.shape)
```

```
a - размер 1 эл.: 8 байт  размер масс.: 24 байт  форма: (3,)
b - размер 1 эл.: 8 байт  размер масс.: 48 байт  форма: (2, 3)
```

У матрицы первое число формы – это количество строк, а второе – количество столбцов.

Можно изменить форму массива. Например, одномерный массив-последовательность (вектор) можно преобразовать в матрицу используя функцию изменения формы `reshape`. При этом число элементов новой формы должно остаться таким же, как в исходном массиве. Например, последовательность, содержащую 10 чисел можно преобразовать в матрицу 2x5:

```
a=np.array([1,2,5,6,3,7,9,0,10,8])
b=a.reshape(2,5)
print ("a:",a,"\n")
print ("b:",b)
```

```
a: [ 1  2  5  6  3  7  9  0 10  8]

b: [[ 1  2  5  6  3]
 [ 7  9  0 10  8]]
```

Нумерация элементов начинается с нуля. Можно задать отрицательный индекс. В этом случае отсчет идет с конца (от последнего элемента по измерению).

```
a=np.array([[11,12,13,14,15],
            [21,22,23,24,25],
            [31,32,33,34,35]])
print("a(1,1)=",a[1][1])
print("a(2,-1)=",a[2][-1])
print("a(-2,-2)=",a[-2][-2])
```

```
a(1,1)= 22
a(2,-1)= 35
a(-2,-2)= 24
```

Из массива можно выделить его часть - срезы. Например, в матрице можно выделить одну строку, указав в квадратных скобках ее номер. Также в измерении можно указывать часть этого измерения:

«**:**» - берется все измерение;

«**:m**» – часть измерения с его начала до элемента с номером **m** (не включая его);

«**n:**» - часть измерения, начиная с элемента с номером **n** до конца измерения;

«**n:m**» - часть измерения, начиная с элемента с номером **n** до элемента с номером **m** (не включая его);

```
a=np.array([[11,12,13,14,15],
            [21,22,23,24,25],
            [31,32,33,34,35]])

print ("2-я строка: ",a[1])
print ("1-й столбец: ",a[:,0])
print("первые 2 строки:\n", a[:2,:])
print("правый нижний угол:\n", a[1:,3:])
```

```
2-я строка:  [21 22 23 24 25]
1-й столбец:  [11 21 31]
первые 2 строки:
 [[11 12 13 14 15]
 [21 22 23 24 25]]
правый нижний угол:
 [[24 25]
 [34 35]]
```

Автоматическое заполнение массива

Элементы массива можно задать при его создании непосредственно в конструкторе, ввести с клавиатуры, а также получить значения с использованием генератора. Для генерирования значения можно использовать: функцию `range` и генератор случайных чисел

а) генерация с использованием функции `range`

```
n=10;
a=np.fromiter((i for i in range (1, n+1)),dtype='int32')
a
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

Выражение `(i for i in range (1, n+1))` создает генератор последовательности чисел от 1 до `n`, а функция `fromiter`, используя это генератор, создает массив чисел заданного типа.

б) Использование генератора случайных чисел

Для заполнения массива случайными числами можно использовать объект генератора случайных чисел `np.random`. Перед использованием генератора его надо инициализировать методом `seed`, задав в качестве параметра любое число. Затем для генерации чисел можно использовать:

а) функцию `rand` для получения равномерно распределенных чисел в диапазоне от 0 до 1. При задании одного аргумента получается последовательность заданного размера, а двух - матрица. Полученные числа можно округлить до заданного числа разрядов после запятой функцией `round(число)`:

```
gen=np.random  # Создание генератора СЧ
gen.seed(869)  # Подготовка генератора
a=gen.rand(5)  # Генерация последовательности СЧ
print("a:",a,"\n");
b=gen.rand(2,4).round(2) # Генерация матрицы СЧ с округлением
print("b:\n",b)
```

```
a: [0.13482046 0.35314807 0.38974998 0.98020001 0.15893401]

b:
 [[0.43 0.29 0.67 0.3 ]
 [0.4  0.67 0.79 0.21]]
```

б) функцию `randn` для получения случайных чисел с нормальным распределением со средним значением 0 и стандартным отклонением 1:

```
c=gen.randn(5).round(3) # Генерация СЧ с нормальным распределением
print("c:",c)
```

```
c: [ 0.087 -0.253 -0.505 -2.027  0.152]
```

в) функцию `randint` для получения равномерно распределенных целых чисел. Первые 2 аргумента задают диапазон изменения целых чисел, а третий - количество чисел. Для преобразования в матрицу можно воспользоваться функцией reshape (n,m):

```
d=gen.randint(1,100,10) # Генерация 10 СЧ в диапазоне от 1 до 100
print("d:",d,"\n")
e=d.reshape(2,5)   # Преобразование в матрицу 2 x 5
print("e:\n",e)
```

```
d: [51 91 43 26 73 96 80 87 64 95]

e:
 [[51 91 43 26 73]
 [96 80 87 64 95]]
```

Создание стандартных математических матриц и векторов

Для создания стандартных математических матриц и векторов используются методы:

а) `np.ones(форма)` - единичный вектор или матрица;

б) `np.zeros(форма)` - нулевой вектор или матрица;

в) `np.eye(размер)` - диагональная единичная матрица.

Примеры:

а)

```
a1=np.ones((2,3))
print (a1)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

б)

```
z=np.zeros((4))
print (z)
```

```
[0. 0. 0. 0.]
```

в)

```
d=np.eye(3)
print (d)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Выполнение операций линейной алгебры

На одном из предыдущих занятий рассматривались линейной алгебры операции над матрицами и векторами. Библиотека `numpy` снабжена более удобными функциями для выполнения операций линейной алгебры над массивами

а) Операции над матрицей и скаляром

Операции матрицы со скаляром записываются естественным образом:

`Матрица` **`операция`** `Скаляр`

где операцией может быть сложение, вычитание, умножение и деление

Пример:

```
a=np.array([[1,5,0.5],
   [3,5,8]])
s=2
print(a)
```

```
[[1.  5.  0.5]
 [3.  5.  8. ]]
```

Сложение	Вычитание	Умножение

`b=a+s`
`print(b)`
`[[ 3.  7.  2.5]`
`[ 5.  7.  10. ]]`
	`b=a-s`
	`print (b)`
	`[[-1.  3.  -1.5]`
	`[ 1.  3.  6. ]]`
		`b=a*s`
		`print(b)`
		`[[ 2. 10.  1.]`
		`[ 6. 10. 16.]]`

б) Операции над векторами и матрицами

Сложение и вычитание матриц также записывается естественным образом:

```
Матрица1 операция Матрица 2
```

Матрицы при этом должны быть одного размера.

Пример

```
A
 [[1.  5.  0.5]
 [3.  5.  8. ]]
```

```
B
 [[4 3 9]
 [5 1 0]]
```

Сложение

```
c=a+b
print(c)
```

```
[[5.  8.  9.5]
 [8.  6.  8. ]]
```

Вычитание

```
c=a-b
print(c)
```

```
[[-3.   2.  -8.5]
 [-2.   4.   8. ]]
```

Умножение матриц выполняется функцией `matmul`. При этом число столбцов первой матрицы должно совпадать с числом строк у второй.

Пример

```
A
 [[1.  5.  0.5]
 [3.  5.  8. ]]
```

```
B
 [[1 4]
 [3 8]
 [2 1]]
```

Произведение

```
c=np.matmul(a,b)
print(c)
```

```
[[17.  44.5]
 [34.  60. ]]
```

Скалярное произведение векторов выполняется методом `dot`

Пример

```
a=np.array([1,5,2,9])
b=np.array([4,8,6,3])
print("a:",a)
print("b:",b)
c=np.dot(a,b)
print("скалярное произведение:",c)
```

```
a: [1 5 2 9]
b: [4 8 6 3]
скалярное произведение: 83
```

г) Транспонирование матрицы

Транспонирование матрицы выполняется методом `матрица.T`

Пример

Исходная матрица

```
A
 [[1.  5.  0.5]
 [3.  5.  8. ]]
```

Транспонированная матрица

```
at=a.T
print(at)
```

```
[[1.  3. ]
 [5.  5. ]
 [0.5 8. ]]
```

д) Вычисление обратной матрицы

Для вычисления обратной матрицы используется функция `inv` объекта `numpy.linalg`. Напомню, что исходная матрица должна быть квадратной.

Пример

Исходная матрица

```
a=np.array([[3,5,1],
            [2,1,8],
            [1,8,7]])
print (a)
```

```
[[3 5 1]
 [2 1 8]
 [1 8 7]]
```

Обратная матрица

```
ainv = np.linalg.inv(a)
print (ainv)
```

```
[[ 0.30645161  0.14516129 -0.20967742]
 [ 0.03225806 -0.10752688  0.11827957]
 [-0.08064516  0.10215054  0.03763441]]
```

Для проверки умножим исходную матрицу на обратную. Должна получиться единичная матрица. Результаты проверки подтверждают это:

```
b=np.matmul(a,ainv).round(1)
print(b)
```

```
[[ 1. -0.  0.]
 [ 0.  1. -0.]
 [ 0.  0.  1.]]
```

Задание

Взять задания из практического занятия №18 и выполнить их с использованием библиотеки numpy. Сравнить результаты.

Список источников

1. Справочник по библиотеке numpy .- https://numpy.org/doc/stable/reference/

Порядок выполнения работы

1. Изучить теоретический материал
2. Составить программы для выполнения задания
3. Выполнить расчет по программе и сравнить полученные результаты с результатами выполнения практического занятия 18

Содержание отчета

1. Программы выполнения операций над матрицами
2. Результаты расчетов по программам

Практическое занятие № 21
Библиотека numpy. Агрегирование данных

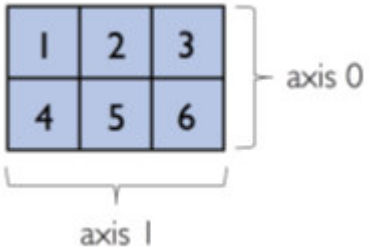
Цель занятия. Изучить функции библиотеки numpy по агрегированию данных. Научиться выполнять агрегирование данных, содержащихся в многомерных массивах

Краткие теоретические сведения

1. Агрегатные функции

В библиотеке `numpy` есть методы, позволяющие заменять данные интегральными характеристиками вдоль некоторых осей (измерений). Например, можно посчитать среднее значение, максимальное, минимальное, вариацию и другие характеристики для одного или нескольких измерений или по всему массиву.

Операцию можно выполнять по разным измерениям (осям). Для матрицы (двумерный массив) такими осями или измерениями являются строки (axis=0) и столбцы (axis=1).



Например, можно выполнять суммирование по строкам, двигаясь по измерению столбцов (axis=1), суммирование по столбцам, двигаясь по измерению строк (axis=0) и суммирование по всему массиву. Операцию (функцию агрегирования) можно записывать в двух формах, дающих одинаковый результат:

`функция``(массив, ось)` - массив передается функции в качестве аргумента

`массив.`**`функция`**`(ось)` - функция вызывается для массива.

Все функции будем рассматривать на примере матрицы размера 3 x 4.

```
a=np.array([[10,4,8,3],
            [9,5,7,12],
            [0,6,11,15]])
print(a)
```

```
[[10  4  8  3]
 [ 9  5  7 12]
 [ 0  6 11 15]]
```

1) Операция суммирования `sum`

а) суммирование по строкам (axis=1). Используем оба формата

```
s1=np.sum(a,axis=1)
print("сумма по строкам\n",s1)
```

```
сумма по строкам
 [25 33 32]
```

```
s1=a.sum(axis=1)
print("сумма по строкам\n",s1)
```

```
сумма по строкам
 [25 33 32]
```

Видно, что оба формата суммирования дают одинаковый результат. Поэтому далее для других агрегатных функций будем использовать второй вариант, как более краткий.

б) суммирование по столбцам (axis=0)

```
s0=a.sum(axis=0)
print("сумма по столбцам\n",s0)
```

```
сумма по столбцам
 [19 15 26 30]
```

в) суммирование по всей матрице

```
s=a.sum()
print("общая сумма\n",s)
```

```
общая сумма
 90
```

2) Операция произведения `product`

Есть только первый формат (массив передается как аргумент)

```
p1=np.product(a,axis=1)
print("произведение по строкам\n",p1)
```

```
произведение по строкам
 [ 960 3780    0]
```

```
p0=np.product(a,axis=0)
print("произведение по столбцам\n",p0)
```

```
произведение по столбцам
 [  0 120 616 540]
```

```
p=np.product(a)
print("общее произведение\n",p)
```

```
общее произведение
 0
```

3) Операция поиска максимального значения max

```
m1=a.max(axis=1)
print("max по строкам\n",m1)
```

```
max по строкам
 [10 12 15]
```

```
m0=a.max(axis=0)
print("max по столбцам\n",m0)
```

```
max по столбцам
 [10  6 11 15]
```

```
m=a.max()
print("общий max\n",m)
```

```
общий max
 15
```

4) Операция поиска минимального значения min

```
mi1=a.min(axis=1)
print("min по строкам\n",mi1)
```

```
min по строкам
 [3 5 0]
```

```
mi0=a.min(axis=0)
print("min по столбцам\n",mi0)
```

```
min по столбцам
 [0 4 7 3]
```

```
mi=a.min()
print("общий min\n",mi)
```

```
общий min
 0
```

5) Операция вычисления среднего арифметического значения mean

```
me1=a.mean(axis=1)
print("среднее по строкам\n",me1)
```

```
среднее по строкам
 [6.25 8.25 8.  ]
```

```
me0=a.mean(axis=0)
print("среднее по столбцам\n",me0)
```

```
среднее по столбцам
 [ 6.33  5.  8.66     7 10. ]
```

```
me=a.mean()
print("общее среднее\n",me)
```

```
общее среднее
 7.5
```

Имеются еще другие функции агрегирования:

`std` - стандартное отклонение

`var` - коэффициент вариации

`argmin` - индекс минимального элемента

`argmax` - индекс максимального элемента

`median` - медиана (средний элемент упорядоченной последовательности)

`percentile` - процентиль (Мера, в которой процентное значение общих значений равно этой мере или меньше ее. Например, 75-й процентиль - это значение, ниже которого находятся 75 % всех значений данных)

Также в библиотеке numpy есть ряд полезных функций:

`sort` - сортировка (упорядочение)

`array_equal` - проверка совпадений двух массивов по форме и элементам

В реальных данных могут быть пропуски значений (отсутствие данные). Такие пропуски имеют специальное обозначение NaN. Если массив имеет пропуски, то агрегатные функции могут работать неправильно. Для таких случаев есть специальные агрегатные функции, которые исключают из вычислений пропущенные значения:

`nansum, nanprod, nanmean` и другие.

Задание

1. Для массива данного занятия вычислить агрегатные функции, по которым не приводились примеры вычислений

2. Задать свой массив и вычислить все агрегатные функции.

Список источников

1. Справочник по библиотеке numpy .- https://numpy.org/doc/stable/reference/

Порядок выполнения работы

1. Изучить теоретический материал
2. Составить программы выполнения пунктов задания
3. Выполнить расчет по программе

Содержание отчета

1. Программный код по всем пунктам задания
2. Результаты расчетов по программе

Практическое занятие № 22
Библиотека `pandas`. Операции с наборами данных

Цель занятия. Изучить основные функции библиотеки pandas и научиться их использовать при выполнении операций с наборами данных

Краткие теоретические сведения

Pandas - очень популярная и простая в освоении библиотека Python для обработки табличных наборов данных. Библиотека может принимать данные из различных источников, таких как файлы CSV, файлы Excel, таблицы реляционных баз данных и текстовые файлы.

Она позволяет применять методы очистки и анализа данных с использованием оптимизированных встроенных функций, которые очень хорошо масштабируются на больших наборах данных.

Перед использованием библиотеки ее необходимо импортировать в программу. Также необходимо импортировать `numpy`, т.к. мы будем использовать функции этой библиотеки:

```
import numpy as np
import pandas as pd
```

Имя после «as» используется в качестве краткого псевдонима библиотеки при вызове ее функций.

Можно проверить версию библиотеки:

```
print(pd.__version__)
```

```
1.3.4
```

Чтобы научиться использовать `pandas`, необходимо знать, как импортировать и экспортировать данные различных типов, манипулировать данными и изменять их форму, сводить и агрегировать данные, получать простую информацию из наборов данных.

1. Создание объектов данных

Основными объектами данных являются серии и наборы данных (фреймы).

Серии (`Series`) - это одномерная структура данных. Создание серий выполняется по заданному списку значений. Для серии по умолчанию создается целочисленный индекс

```
s = pd.Series([11, 8, np.nan, 6, 2, 15, 21])
print(s)
```

```
0    11.0
1     8.0
2     NaN
3     6.0
4     2.0
5    15.0
6    21.0
dtype: float64
```

В выведенной таблице первая колонка - индекс элемента, а вторая - элемент серии. В серию было специально добавлено неопределенное значение `nan`, которое соответствует пропущенному значению.

Фреймы DataFrame

`DataFrame` – это двумерная структура данных со столбцами потенциально разных типов. И этим фрейм отличается от массивов-матриц, в которых все колонки должны содержать значения одного типа. Каждым столбцом фрейма является объект `Series`. Фрейм можно представить как электронную таблицу или таблицу БД SQL.

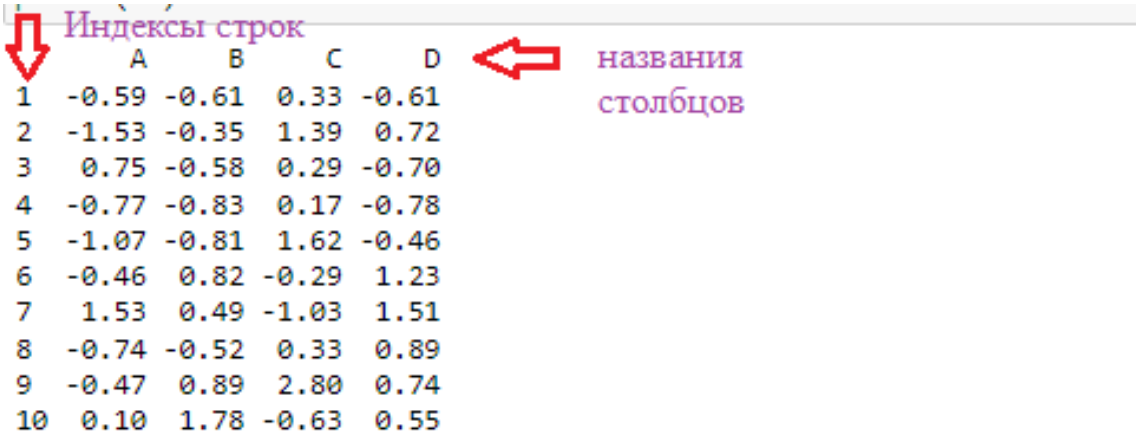
Создание `DataFrame` выполняется с использованием массива `numpy` или серий.

а) При создании фрейма на базе массива `numpy` конструктору набора данных передается:

- массив `numpy`;
- серия для индекса строк;
- список имен столбцов.

Пример. Создадим фрейм. В качестве базового массива будем использовать массив случайных чисел, индексами будет серия из целых чисел, а заголовками столбцов будут латинские буквы:

```
n=10
# Создание серии целых чисел для индекса
si = pd.Series((i for i in range(1,n+1)))
# Создание фрейма
df = pd.DataFrame(np.random.randn(10, 4), # массив случайных чисел
                  index=si,               # серия с индексами
                  columns=list('ABCD'))   # название колонок
df=np.round(df,2) # округление до 2 знаков в дробной части
print (df)
```



	A	B	C	D
1	-0.59	-0.61	0.33	-0.61
2	-1.53	-0.35	1.39	0.72
3	0.75	-0.58	0.29	-0.70
4	-0.77	-0.83	0.17	-0.78
5	-1.07	-0.81	1.62	-0.46
6	-0.46	0.82	-0.29	1.23
7	1.53	0.49	-1.03	1.51
8	-0.74	-0.52	0.33	0.89
9	-0.47	0.89	2.80	0.74
10	0.10	1.78	-0.63	0.55

б) При создании фрейма на базе набора серий конструктору набора данных передается словарь из набора серий. Каждый объект словаря определяет столбец фрейма и задается именем и значением-серией (массивом значений). Индексами по умолчанию являются целые числа.

Пример. Создадим фрейм из следующего набора серий:

A - серия целых чисел

B - серия дат

C - серия категорий (набор ключевых слов)

В качестве индексов строк в параметре `index` зададим символьные значения цифр от '1' до '7'

```
df2 =pd.DataFrame({'A': [10,11,12,13,14,15,16],
                   'B': pd.date_range('1/1/2021', periods=7),
                   'C':['r','w','b','w','b','r','r']}, index=list('1234567'))
df2
```

	A	B	C
1	10	2021-01-01	r
2	11	2021-01-02	w
3	12	2021-01-03	b
4	13	2021-01-04	w
5	14	2021-01-05	b
6	15	2021-01-06	r
7	16	2021-01-07	r

2. Выборочный просмотр данных

Методы `head` и `tail` используют для вывода ***первых и последних строк*** фрейма. Выводится по умолчанию 5 первых строк в методе `head()` или последних - `tail()`. Можно указать количество строк (но не более 5)

```
df2.head(2)
```

	A	B	C
1	10	2021-01-01	r
2	11	2021-01-02	w

```
df2.tail(3)
```

	A	B	C
5	14	2021-01-05	b
6	15	2021-01-06	r
7	16	2021-01-07	r

Можно вывести все индексы (свойство **index)** и информацию о заголовках (свойство **columns**):

```
df2.index
```

```
Index(['1', '2', '3', '4', '5', '6', '7'], dtype='object')
```

```
df2.columns
```

```
Index(['A', 'B', 'C'], dtype='object')
```

3. Выборка данных из набора

Выборка столбцов

Часто из набора данных надо выбрать какую-то его часть (отдельные столбцы, строки и т.д.).

Выбор одного столбца, который возвращает объект Series. Выбрать один столбец можно указав его имя в квадратных скобках или через точку

```
df2.B # тот же результат даст df2['B']
```

```
1   2021-01-01
2   2021-01-02
3   2021-01-03
4   2021-01-04
5   2021-01-05
6   2021-01-06
7   2021-01-07
Name: B, dtype: datetime64[ns]
```

Для выбора нескольких столбцов необходимо указать список их имен через запятую, дополнительно заключив в квадратные скобки.

Например,

```
df2[['B','C']]
```

вернет данные из столбцов с именами B и C.

Выборка строк

Для выборки строк необходимо в квадратных скобках указать их диапазон их номеров или индексов

Например

`df2[3:5]` - вернет строки с номерами с 3 по 4 (номера строк начинаются с 0)

`df2['3':'5']` - вернет строки с индексами от `'3'` до `'5'`

```
df2[3:5]
```

	A	B	C
4	13	2021-01-04	w
5	14	2021-01-05	b

```
df2['3':'5']
```

	A	B	C
3	12	2021-01-03	b
4	13	2021-01-04	w
5	14	2021-01-05	b

Выборка строк и столбцов

Можно также воспользоваться функциями:

а) **`loc (список индексов,список колонок)`**

```
df2.loc[:, ['A', 'B']] - выбирает все строки для столбцов A,B
df.loc['1':'3', ['A', 'B']] выбирает строки с индексами от 1
   до 3 для столбцов A и B
df1.loc['2', 'A'] - выбор одной ячейки
```

```
df2.loc[:, ['A', 'B']]
```

	A	B
1	10	2021-01-01
2	11	2021-01-02
3	12	2021-01-03
4	13	2021-01-04
5	14	2021-01-05
6	15	2021-01-06
7	16	2021-01-07

```
df2.loc['1':'3', ['A', 'B']]
```

	A	B
1	10	2021-01-01
2	11	2021-01-02
3	12	2021-01-03

```
df2.loc['2', 'A']
```

```
11
```

б) **`iloc (список номеров строк, список колонок)`**

```
df2.iloc[3]  - выбор 4-й строки
df2.iloc[[1,2,4],[0,2]] - выбор строк 2,3,5 для столбцов 1 и 3
```

```
df2.iloc[3]
```

```
A                          13
B     2021-01-04 00:00:00
C                           w
Name: 4, dtype: object
```

```
df2.iloc[[1,2,4],[0,2]]
```

	A	C
2	11	w
3	12	b
5	14	b

Выборка по условию

Можно отбирать строки набора, удовлетворяющие заданным условиям. Например, отберем строки для дат, больше заданной

```
df2[df2.B >'2021-01-03']
```

	A	B	C
4	13	2021-01-04	w
5	14	2021-01-05	b
6	15	2021-01-06	r
7	16	2021-01-07	r

Нарезка строк и соединение наборов

Из исходного набора можно выделить несколько фрагментов строк набор, а затем объединить их в новый набор.

Выберем строки 1, 3, 4, 6, 7, а затем соединим из в новом наборе функцией `concat`:

```
parts=[df2[:1],df2[2:4],df2[5:]]
df5=pd.concat(parts)
```

	A	B	C
1	10	2021-01-01	r
3	12	2021-01-03	b
4	13	2021-01-04	w
6	15	2021-01-06	r
7	16	2021-01-07	r

4. Анализ данных. Описательная статистика

Можно вычислить функции описательной статистики по столбцам (по оси 0). Будем использовать набор данных `df`**:**

```
n=10
# Создание серии целых чисел для индекса
si = pd.Series((i for i in range(1,n+1)))
# Создание фрейма
df = pd.DataFrame(np.random.randn(10, 4), # массив случайных чисел
                  index=si,               # серия с индексами
                  columns=list('ABCD'))   # название колонок
df=np.round(df,2) # округление до 2 знаков в дробной части
print (df)
```

```
       A     B     C     D
1  -0.75  0.52  0.51  0.12
2   0.18 -0.04 -0.38 -1.17
3  -0.09  2.42  1.25 -0.11
4   0.09 -0.36 -0.84 -0.94
5  -0.81 -2.00  1.18 -0.30
6  -0.26  0.33  0.83  1.22
7  -2.35  0.80 -0.32  1.27
8   0.38 -0.75 -2.39 -1.32
9   1.08  0.72 -0.42  2.04
10  1.00 -1.34  2.47 -1.08
```

Например, для вычисления среднего значения по столбцам используется функция `mean`:

```
df.mean()
```

```
A   -0.153
B    0.030
C    0.189
D   -0.027
dtype: float64
```

Полный набор описательной статистики можно получить, вызвав функцию `describe` (из расчета исключаются неопределенные значения NaN):

```
df.describe()
```

	A	B	C	D
count	10.000000	10.000000	10.000000	10.000000
mean	-0.153000	0.030000	0.189000	-0.027000
std	0.996327	1.241612	1.353534	1.180085
min	-2.350000	-2.000000	-2.390000	-1.320000
25%	-0.627500	-0.652500	-0.410000	-1.045000
50%	0.000000	0.145000	0.095000	-0.205000
75%	0.330000	0.670000	1.092500	0.945000
max	1.080000	2.420000	2.470000	2.040000

Кол-во значений
Среднее
Станд.отклонение
Минимальное
Процентили
Максимальное

5. Отсутствующие данные. Очистка данных

В реальных наборах данных могут быть пропуски данных. В pandas отсутствие данных обозначается `np.nan`. Пропущенные значения не участвуют в вычислениях и поэтому могут исказить общую картину.

С пропущенными значениями поступают по-разному:

- удаление строк с пропущенными значениями функцией `dropna`
- замена пропущенных значений (нулевыми, средними) функции `fillna`

Для эксперимента заменим значения набора `df` пропусками. Заменим значение в ячейке 2-й строки и 2-го столбца пропуском. Для этого воспользуемся функцией **`iat(индекс строки,индекс столбца)`**. Вспомним, что индексация начинается с 0:

```
df.iat[1,1]=np.NaN
```

	A	B	C	D
1	-0.75	0.52	0.51	0.12
2	0.18	NaN	-0.38	-1.17
3	-0.09	2.42	1.25	-0.11
4	0.09	-0.36	-0.84	-0.94
5	-0.81	-2.00	1.18	-0.30
6	-0.26	0.33	0.83	1.22
7	-2.35	0.80	-0.32	1.27
8	0.38	-0.75	-2.39	-1.32
9	1.08	0.72	-0.42	2.04
10	1.00	-1.34	2.47	-1.08

```
df.describe()
```

	A	B
count	10.000000	9.000000
mean	-0.153000	0.037778
std	0.996327	1.316670

Можно увидеть, что пропущенные значения изменили описательную статистику, т.к. пропущенные значения исключаются из расчетов.

Используем разные походы очистки данных:

а) удаление строк с пропущенными значениями

```
df1=df.dropna()
df1.describe()
```

	A	B	C	D
count	9.000000	9.000000	9.000000	9.000000
mean	-0.190000	0.037778	0.252222	0.100000
std	1.049452	1.316670	1.419892	1.176956

Надо иметь в виду, что функция `dropna` (как и другие функции) не изменяет исходный набор данных, поэтому результат выполнения функции мы присваиваем новому набору.

б) замена пропущенных значений нулями

```
df1=df.fillna(value=0)
df1.describe()
```

	A	B	C	D
count	10.000000	10.000000	10.000000	10.000000
mean	-0.153000	0.034000	0.189000	-0.027000
std	0.996327	1.241426	1.353534	1.180085

в) замена пропущенных значений средним по оставшимся строкам

```
df1=df.fillna(value=df['B'].mean())
df1.describe()
```

	A	B	C	D
count	10.000000	10.000000	10.000000	10.000000
mean	-0.153000	0.037778	0.189000	-0.027000
std	0.996327	1.241368	1.353534	1.180085

6. Группировка с агрегированием

Выше мы рассмотрели простой анализ данных с вычислением описательной статистики. Давайте проведем более сложный анализ данных. Для этого сформируем новый набор, который будет включать:

- столбец с категориями: `'r','g','b'`
- столбец с категориями: `'hi','lo'`
- столбец с целыми числами
- столбец с равномерно распределенными случайными целыми числами от 0 до 100
- столбец с нормально распределенными случайным числами

```
df2 =pd.DataFrame({'K1':['r','g','b','g','b','r','r','g','b','g'],          # Категория 1
                   'K2':['hi','lo','lo','hi','lo','lo','hi','hi','hi','lo'],# Категория 2
                   'A': [8,11,15,9,14,15,4,17,10,7], # Целые числа
                   'B1': np.random.randint(1,100,10),# Целые СЧ с равномерным распределением
                   'B2': np.random.randn(10)         # СЧ с нормальным распределением
                  })
```

	K1	K2	A	B1	B2
0	r	hi	8	55	1.881846
1	g	lo	11	40	1.162910
2	b	lo	15	58	0.291189
3	g	hi	9	14	-0.231729
4	b	lo	14	57	-1.659384
5	r	lo	15	22	0.445083
6	r	hi	4	49	2.746187
7	g	hi	17	68	-0.463908
8	b	hi	10	59	-0.880554
9	g	lo	7	4	-0.110083

Агрегирование данных по колонкам с числовыми данными заключается в нахождении обобщенных значений по этим колонкам (суммы, среднего значения и др.). Например, для вычисления суммы по столбцам A и B1 необходимо применить функцию `sum` к списку колонок:

```
df2[['A','B1']].sum()
```

```
A     110
B1    426
```

Если необходимо вычислить среднее для тех же колонок, то можно повторить вычисления, заменив одну функцию на другую. Но в pandas есть удобная агрегатная функция `agg` (появилась только в версии 0.25.0 от 18 июля 2019), которая упрощает задачу. В ней необходимо указать список агрегатных функций для заданного списка колонок:

```
df2[['A','B1']].agg(['sum','mean'])
```

	A	B1
sum	110.0	426.0
mean	11.0	42.6

Если же надо для разных колонок вычислять разные агрегатные функции, то в функцию `agg` надо передать словарь, в котором ключом будет колонка, а значением список агрегатных функций:

```
df2.agg({'A':['sum'],'B1':['max'], 'B2':['sum','mean']})
```

	A	B1	B2
sum	110.0	NaN	3.181556
max	NaN	68.0	NaN
mean	NaN	NaN	0.318156

Часто обобщенные значения необходимо вычислить для отдельных категорий. В этом случае сначала группируют данные, собирая в отдельные группы данные по каждой категории, а затем по каждой группе вычисляют обобщенные значения. Для группировки используют функцию `groupby(список_колонок)`. Разобьем исходный набор на группы по первой категории, а затем выполним обобщение данных:

```
df2.groupby('K1').agg({'A':['sum'],'B1':['max'], 'B2':['sum','mean']})
```

	A	B1	B2	
	sum	max	sum	mean
K1				
b	39	59	-2.248749	-0.749583
g	44	68	0.357189	0.089297
r	27	55	5.073116	1.691039

7. Загрузка данных из внешних файлов

Данные, по которым проводится анализ, обычно берутся из внешних источником (текстовых файлов, файлов в формате CSV, из баз данных). Рассмотрим загрузку данных для анализа из файлов в формате CSV.

CSV — это очень распространенный формат (расшифровывается как «comma separated values», т.е. «значения разделенные запятыми»). Обычно в таких файлах в первой строке находится заголовок с названиями столбцов, все остальные строки — это данные. Разделитель между элементами это символ запятая. Для того чтобы загрузить данные из этого файла в DataFrame нужно использовать функцию:

```
pd.read_csv(путь к файлу)
```

Файл может находиться на вашем компьютере, в какой-то папке, а может находиться на интернет ресурсе. Если файл находится на внешнем ресурсе, то его нужно скачать на свой компьютер.

В качестве данных для примера будем использовать популярный набор в машинном обучении Ирисы Фишера, который можно скачать по адресу (можно не скачивать, а указать этот путь в функции read_csv):

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Положим скачанный файл в каталог Data на диске D. Если вы будете использовать другой диск и каталог, то путь в команде надо изменить. В этом файле нет заголовочной строки, поэтому надо использовать дополнительный параметр header=None. Выполним команду загрузки и запишем файл в набор данных DataFrame:

```
ir=pd.read_csv('d:/data/iris.data', header=None)
```

```
ir
```

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

В наборе данных содержаться данные о 150 цветках Ириса, по каждому из которых представлена информация в 5 колонках:

- длина чашелистика
- ширина чашелистика
- длины лепестка
- ширина лепестка
- вид Ириса

Для примера сгруппируем данные по колонке с индексом 4 и вычислим по первой колонке среднее, а по второй - стандартное отклонение:

```
ir.groupby(4).agg({0:['mean'],1:['std']})
```

	0	1
	mean	std
4		
Iris-setosa	5.006	0.381024
Iris-versicolor	5.936	0.313798
Iris-virginica	6.588	0.322497

Задание.

1. Скачайте набор данных Airbnb - объявлений о возможности снять жилье (файл `listings.csv` для одного из городов США). Набор можно скачать по адресу [http://insideairbnb.com/get-the-data.html].

Набор данных объявлений включает в себя название объявления и объекта жилья, информацию о местоположении, включая координаты района, долготы и широты, тип объекта (весь дом, отдельная комната, гостиница и общая комната), информацию о прошлых обзорах, а также цену за ночь и минимальное количество ночей для бронирования.

Список колонок набора данных

Название	**Имя колонки**
Идентификатор записи	**Id**
Название объекта	**name**
Идентификатор объекта	**host_id**
имя объекта	**host_name**
группа соседства	**neighbourhood_group**
Район	**neighbourhood**
Широта	**latitude**
Долгота	**longitude**
Тип объекта	**room_type**
Цена за 1 ночь	**price**
минимум ночей	**minimum_nights**
количество отзывов	**number_of_reviews**
последний отзыв	**last_review**
отзывов в месяц	**reviews_per_month**
рассчитанное количество списков хостов	**calculated_host_listings_count**
Доступность (количество дней в году)	**availability_365**

2. Загрузите набор данных в программу и выполните следующие действия

а) выведите несколько начальных и конечных строк

б) выполните подготовку данных

- выведите типы столбцов
- определите количество незаполненных значений в каждом столбце
- определите количество строк после удаления строк с незаполненными значениями

в) выполните анализ данных

- выведите данные описательной статистики средней и медианной цены
- сгруппируйте данные по типам помещений и вычислите среднюю цену и медианную по группе
- выполните группировку по типам объектам и годам

- выполните отбор данных по условиям

Список источников

1. Руководство по начальному использованию библиотеки pandas . - https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html
2. Документация по библиотеке pandas . - https://pandas.pydata.org/docs/

Порядок выполнения работы

1. Изучить теоретический материал
2. Загрузить набор данных
3. Составить программы выполнения пунктов задания
4. Выполнить расчет по программе

Содержание отчета

1. Программный код (блокнот) по всем пунктам задания
2. Результаты расчетов по программе

Практическое занятие № 23
Библиотека matplotlib. Визуализация данных

Цель занятия. Изучить основные функции библиотеки matplotlib и научиться использовать средства библиотеки для визуализации данных

Краткие теоретические сведения

Библиотека matplotlib предназначена для построения различных графиков и диаграмм. Для использования библиотеки на Python необходимо ее импортировать и проверить ее версию

```
import matplotlib
matplotlib.__version__
```

```
'3.4.3'
```

Для построения графиков и диаграмм используется объект графика `pyplot`, который также надо импортировать

```
import matplotlib.pyplot as plt
```

Самой простой график - это график линии. Для построения графика надо задать 2 массива:

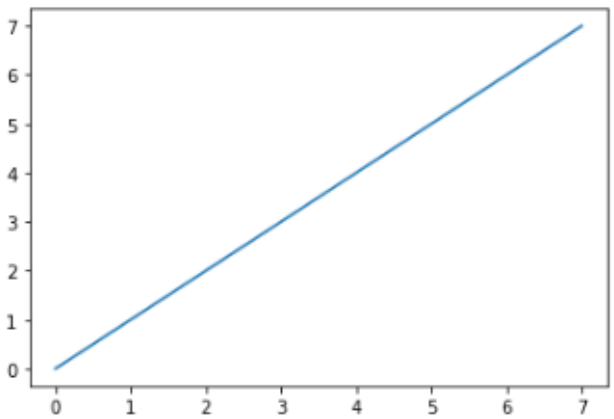
- массив X по оси абсцисс;
- массив Y по оси ординат;

Элементы массивов с одинаковым индексом представляют координаты точек на плоскости $y_i=f(x_i)$. Метод plot соединяет точки линиями, а метод show выводит график.

В качестве массивов можно использовать такие структуры данных Python как списки, кортежи, а также массивы `numpy`.

Пример. Построим график линейной функции `y=x`

```
import numpy as np
x=np.array([0,1,2,3,4,5,6,7])
y=x
plt.plot(x,y)
plt.show()
```



По документации метод plot имеет включает следующие параметры:

```
plot([x],y,[формат линии],[дополнительные свойства])
```

где *формат линии* - строка для задания формата линии графика, задается в виде строки из трех символов: [маркер][тип_линии][цвет]

дополнительные свойства задают параметры тонкой настройки: прозрачность и некоторые другие свойства, которые можно посмотреть в документации.

Некоторые параметры формата линии приведены в таблице 1 (порядок задания параметров произвольный).

Таблица 1 - Параметры формата линии

Маркер точек графика (символы или латинские буквы)	
«точка»	«точка» - маркер по умолчанию
o (строчная)	окружность
v (строчная)	треугольник, направленный вниз
s (строчная)	квадрат
«плюс»	плюс
x (строчная)	х-образный маркер
Тип линии	
«дефис»	сплошная линия (по умолчанию)
-- (2 дефиса)	штриховая линия
«-.» (дефис и точка)	штрих-пунктирная линия
«двоеточие»	линия из точек
Цвет линии (задается строчной латинской буквой)	
b	Синий
g	Зеленый
r	Красный
c	Бирюзовый
m	Фиолетовый (пурпурный)
y	Желтый
k	Черный
w	Белый

Пример. Зададим вывод графика пунктирной линией зеленого цвета (символ «g» от green) с маркерами в виде квадратов.

```
plt.plot(x,y,'g--s')
plt.show()
```

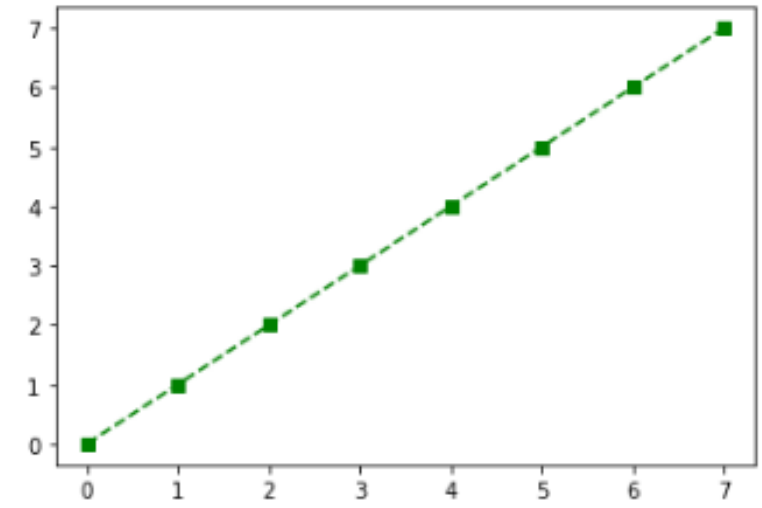
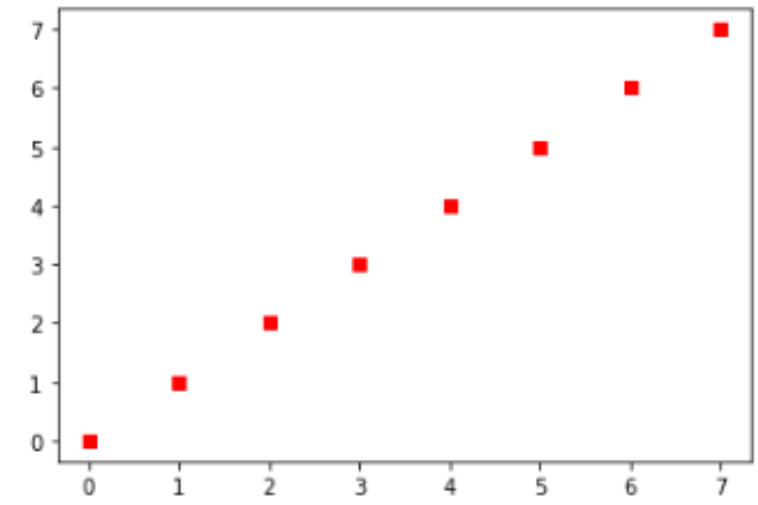


График можно включать только точки без соединения их линиями. Для этого используется метод `scatter`. Тип маркера задается параметром `marker`, а цвет - параметром `c` (латинская буква). Значения параметров такие же как для графиков-линий (таблица 1).

```
plt.scatter(x,y,marker='s',c='r')
plt.show()
```



Основные элементы графика

Основным компонентом при построении графиков в библиотеке `Matplotlib` является Фигура (*Figure*). Это контейнер самого верхнего уровня, та область, в которой выполняется построение графиков. В этой области можно создать отдельные подобласти. Все, что нарисовано в области фигуры, является элементами фигуры. Они повышают информативность графика. Основными элементами являются (рис.1):

1) График. Имеется 2 вида графиков: в виде линии (строится методом `plot`) и точечный (строится методом `scatter`). Параметры настройки для

построения более «красивого» графика: цвет, толщина и тип линии, стиль линии приведены в таблице 1.

2) Заголовок графика (title) задает название графика.

3) Оси. Оси представляют области изменения переменных, отображаемые на графике. Для двумерного графика - это горизонтальная ось абсцисс и вертикальная ось ординат. Для каждой оси можно задать метку (подпись), основные (*major*) и дополнительные (*minor*) тики, их подписи, размер и толщину, диапазоны изменения по осям.

4) Легенда. Легенда представляет обозначения графиков (названия или функции) Используется когда выводится несколько графиков в области построения..

5) Сетка. Сетка позволяет более точно определять значения графика. Сетка может быть основной (*major*) и дополнительной (*minor*). Для сеток можно задавать цвет, толщину линии и тип.

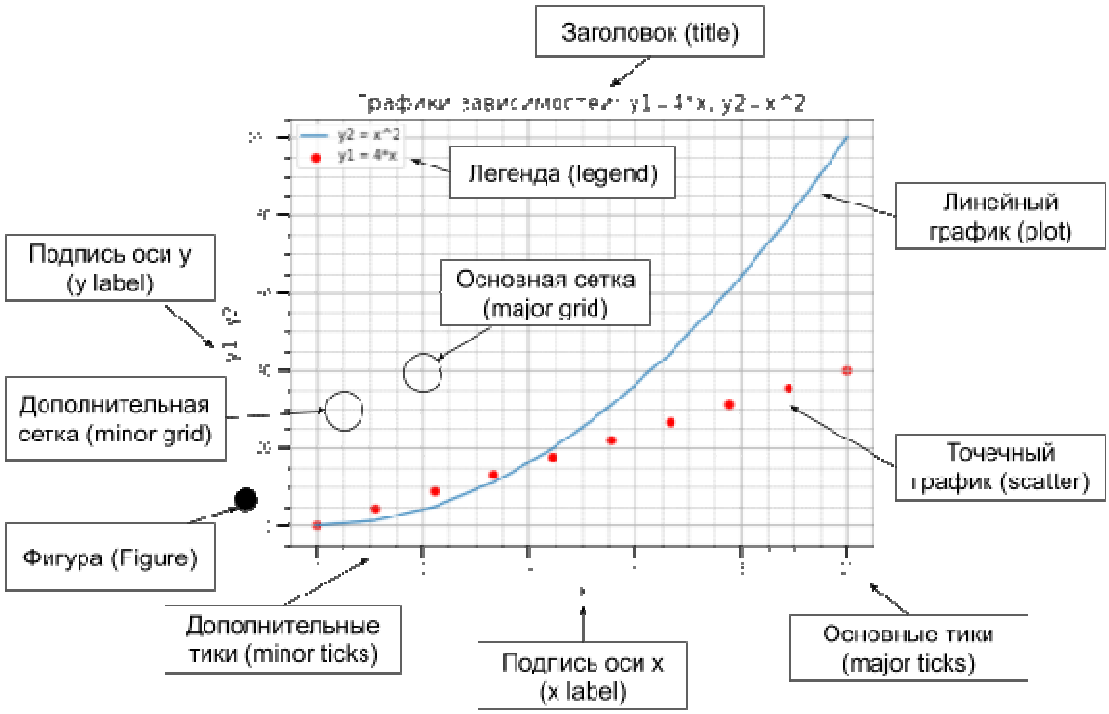
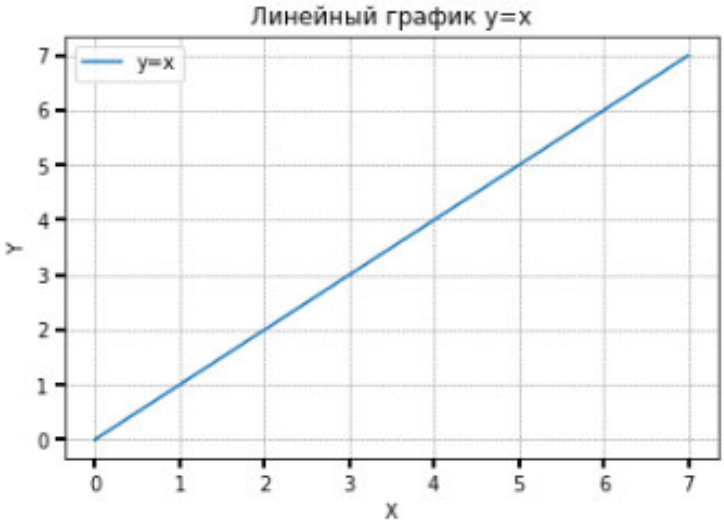


Рис.1 Основные элементы построения графика

Для вывода всех элементов используются соответствующие методы.

Пример 2. Выведем график функции и дополнительные элементы

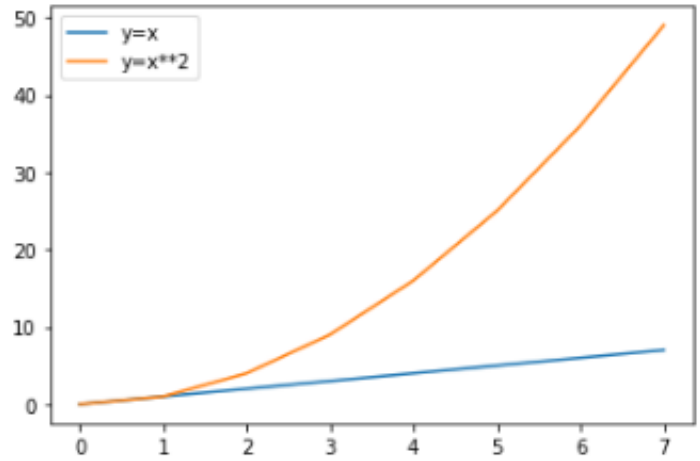
```
plt.plot(x,y)
plt.title('Линейный график y=x') # Заголовок
plt.xlabel('X') # Подписи на осях
plt.ylabel('Y')
plt.grid(linewidth=0.5, color='grey', linestyle='--')  # Вывод сетки
plt.tick_params(length=5, width=2) # Вывод тиков на осях
plt.legend(['y=x'])    # Добавление легенды
plt.show()
```



В одной области построения можно вывести несколько графиков. Необходимо для каждого графика вызвать метод plot.

Пример 3. Вывести линейный график и параболу.

```
plt.plot(x,y)
y2=x**2
plt.plot(x,y2)
plt.legend(['y=x','y=x**2'])
plt.show()
```

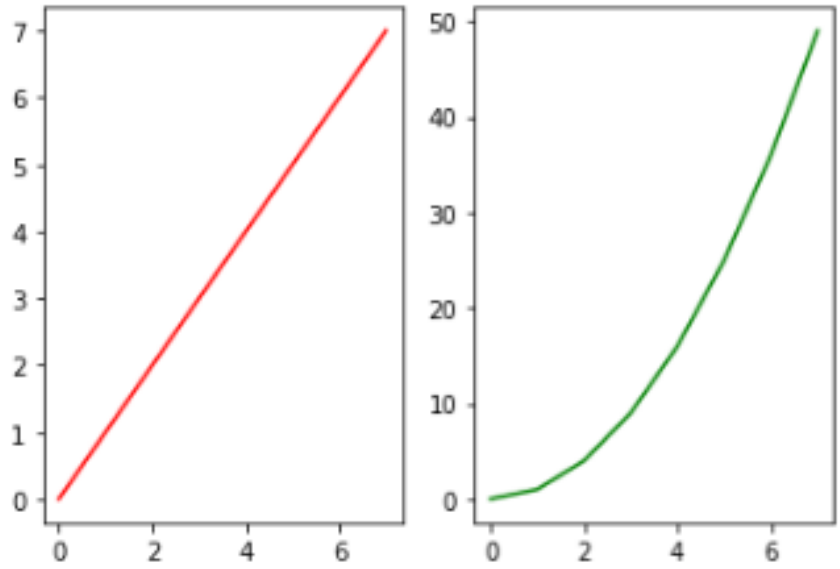


Также можно область построения разбить на несколько подобластей для вывода отдельных графиков. Существует несколько способов разбиения. Одним из способов является использование функции `subplot`, которая имеет 3 параметра:

- Количество строк, на которые делится основная область;
- Количество столбцов, на которые делится основная область;
- Местоположение элемента. Задается порядковым числом. Порядок отсчитывается слева направо и далее сверху вниз.

Пример 4. Разбить область на 2 части. В левой подобласти вывести линейный график, а в правой - параболу

```
plt.subplot(1,2,1)
plt.plot(x,y,'r')
plt.subplot(1,2,2)
plt.plot(x,y2,'g')
plt.show()
```



Задание

1. Создать массив X из n элементов: равноотстоящих чисел на отрезке $[x_1, x_2]$. x_1, x_2 - произвольные числа
2. Построить график функции y=ax+b. a и b - произвольные числа
3. Построить в той же области точечный график для функции

   y=ax+b+E,

   где E - нормально распределенное случайное число
4. Построить линейный и точечный графики в отдельных подобластях

Список источников

1. Документация по библиотеке matplotlib .- https://matplotlib.org/stable/

Порядок выполнения работы

1. Изучить теоретический материал
2. Составить программы построения графиков по пунктам задания
3. Выполнить программы

Содержание отчета

1. Программный код (блокнот) по всем пунктам задания
2. Графики по пунктам задания

Практическое занятие № 24
Одномерный анализ данных. График функции. Гистограммы. Распределения

Цель занятия. Научиться выполнять одномерный анализ данных, визуализировать данные анализа в виде графиков функций, гистограмм. Освоить средства вывода графиков функций и гистограмм на Python

Краткие теоретические сведения

Одномерный анализ данных – это анализ результатов наблюдений или измерений ($x(1), x(2), \ldots, x(n)$) одной переменной величины X. Для анализа данных важно понимать, имеет или нет значение порядок следования значений. Если порядок не важен, то последовательность значений можно, при необходимости, изменить: при этом информация, содержащаяся в данных, не теряется. Если порядок важен, то последовательность значений в наборе данных менять нельзя: может быть потеряна важная информация, содержащаяся в данных.

Анализ данных заключается в вычислении некоторых числовых оценок и графическом представлении данных. Числовые оценки вычисляют для характеристик:

А) центра, вокруг которого группируются данные - среднее арифметическое *Xs*, среднее геометрическое *Xg*, среднее гармоническое *Xh*, медиана *Xm*.

Б) величины разброса данных (их вариабельности) относительно центра – дисперсия *DX*, среднеквадратическое отклонение *SX*, коэффициент вариации *CVX*, размах *WX*, интердецильный размах *WdX*.

Графически упорядоченные данные изображают кривыми эволюции данных. Любые данные графически отображают гистограммами и графиками накопленных частот.

Пример

Подсчитывалось количество $n(k)$ документов, поступающих на компьютерную обработку по дням за четыре недели месяца $k = 1,2,\ldots,28$. Результаты наблюдений отражены в таблице 1.

Таблица 1 - Количество документов, поступающих на обработку

День	1	2	3	4	5	6	7
***n*(*k*)**	**20**	**13**	**16**	**12**	**18**	**12**	**15**
День	8	9	10	11	12	13	14
***n*(*k*)**	**17**	**19**	**20**	**16**	**11**	**11**	**12**
День	15	16	17	18	19	20	21
***n*(*k*)**	**19**	**12**	**18**	**12**	**20**	**13**	**12**
День	22	23	24	25	26	27	28
***n*(*k*)**	**12**	**16**	**15**	**13**	**19**	**16**	**16**

Вычислим оценки центра:

$$Xs = \frac{1}{28}\left(x(1) + ... + x(28)\right) = \frac{425}{28} = 15{,}1786$$

$$Xg = \sqrt{\frac{1}{28}\left(x(1)^2 + ... + x(28)^2\right)} = \sqrt{\frac{6707}{28}} = 15{,}4769$$

$$Xh = \sqrt[28]{x(1)\cdot x(2)\cdot ...\cdot x(28)} = \exp\left(\frac{1}{28}\left(\ln x(1) + ... + \ln x(28)\right)\right) =$$

$$= \exp\left(\frac{75{,}5985}{28}\right) = 14{,}8789$$

Для оценки медианы упорядочим значений по возрастанию $x_1 < \ldots < x_{28}$ и найдем элемент со средним номером (при нечетном числе значений) или среднее арифметическое двух средних (при четном числе значений). В примере $Xm = 0.5\ (\ x_{14} + x_{15}\ ) = 0.5\ (\ 15 + 16\ ) = 15{,}5$.

Вычислим оценки вариабельности данных

$$DX = \frac{1}{28-1}\left(\left(x(1) - Xsr\right)^2 + ... + \left(x(28) - Xsr\right)^2\right) = \frac{256{,}1071}{27} = 9{,}4854$$

$$SX = \sqrt{DX} = 3{,}0798 \qquad CVX = SX\ /\ Xs\,100\% = 20{,}2907\%$$

$$WX = \max(x(1),...,x(28)) - \min(x(1),...,x(28)) = 20 - 11 = 9$$

Для оценки интердецильного размаха следует упорядочить данные по возрастанию, отбросить 10% самых маленьких и 10% самых больших по величине данных, а заем найти размах оставшихся. В примере $WdX = x_{25} - x_4 = 19 - 12 = 7$.

Эволюцию данных по номерам дней их поступления отобразим кривой на графике (рис. 1).

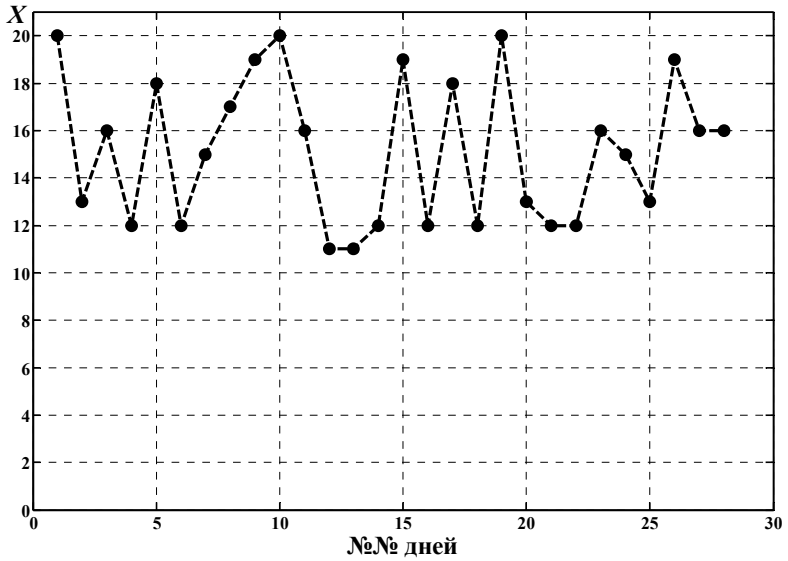


Рис. 1. Эволюция данных по номерам дней месяца.

Выполним одномерный анализ данных на Python.

Создадим массив X по данным таблицы 1.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
```

```
x=np.array([20,13,16,12,18,12,15,17,19,20,16,11,11,12,
            19,12,18,12,20,13,12,12,16,15,13,19,16,16])
```

Для выполнения одномерного анализа необходимо воспользоваться набором функций описательной статистики библиотеки `numpy`. Для некоторых оценок центра и вариабельности данных метода нет. Для них будем выполнять вычисления по формулам.

Начнем с оценок центра. Вычислим среднее и медиану.

```
sr=x.mean().round(4)                # среднее арифметическое
xg=math.sqrt(np.sum(x**2)/28)       # среднее геометрическое
xh=math.exp(np.sum(np.log(x))/28) # среднее гармоническое
median= np.median(x)                # медиана

print('ср.арифм=',sr)
print('ср.геометрич.=',np.round(xg,4))
print('ср.гармонич.=' ,np.round(xh,4))
print('медиана=', median)
```

```
ср.арифм= 15.1786
ср.геометрич.= 15.4769
ср.гармонич.= 14.8789
медиана= 15.5
```

Далее вычислим оценки вариабельности данных.

```
dx=1/(28-1)*np.sum((x-np.mean(x))**2) # дисперсия
std=math.sqrt(dx)                     # станд.отклонение
var=std/np.mean(x)*100                # коэф.вариации
razmax=np.ptp(x)                      # размах
print("дисперсия=", dx.round(4))
print("станд.отклонение=", np.round(std,4))
print("коэф.вариации=", np.round(var,4), '%')
print("размах=",razmax )
```

```
дисперсия= 9.4854
станд.отклонение= 3.0798
коэф.вариации= 20.2907 %
размах= 9
```

Все вычисленные значения совпадают с ручным расчетом, выполненным выше.

Гистограмма

Гистограмма – это столбиковая диаграмма, отображающая частоту появления значений. Поскольку значения X в примере – дискретные целочисленные, то на гистограмме отображают долю числа случаев появления значений от общего числа случаев. Результат анализа данных для построения гистограммы приведен в таблице 2

Таблица 2

X	11	12	13	14	15	16	17	18	19	20
$n(x)$	2	7	3	0	2	5	1	2	3	3
$h(x)$	0,071	0,25	0,107	0	0,071	0,180	0,036	0,071	0,107	0,107
$Sn(x)$	0,071	0,321	0,428	0,428	0,499	0,679	0,715	0,786	0,893	1,000

График гистограммы приведен на рисунке 2.

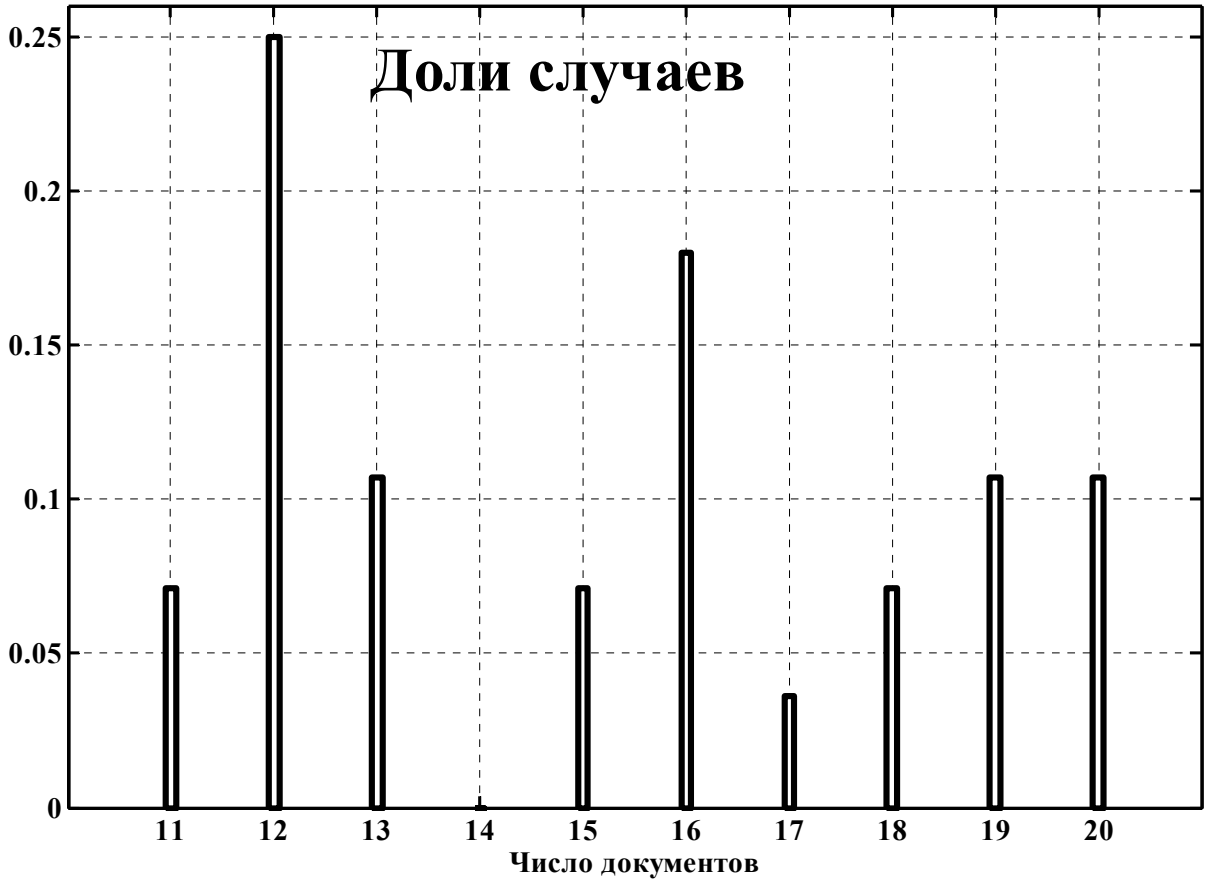


Рис. 2. Гистограмма числа документов

График накопленных частот (рисунок 3) строится в виде ступенчатой диаграммы на основе данных последней строки в таблице 2. Он показывает долю случаев для числа документов, равного или меньшего *x,* в анализируемом наборе данных.

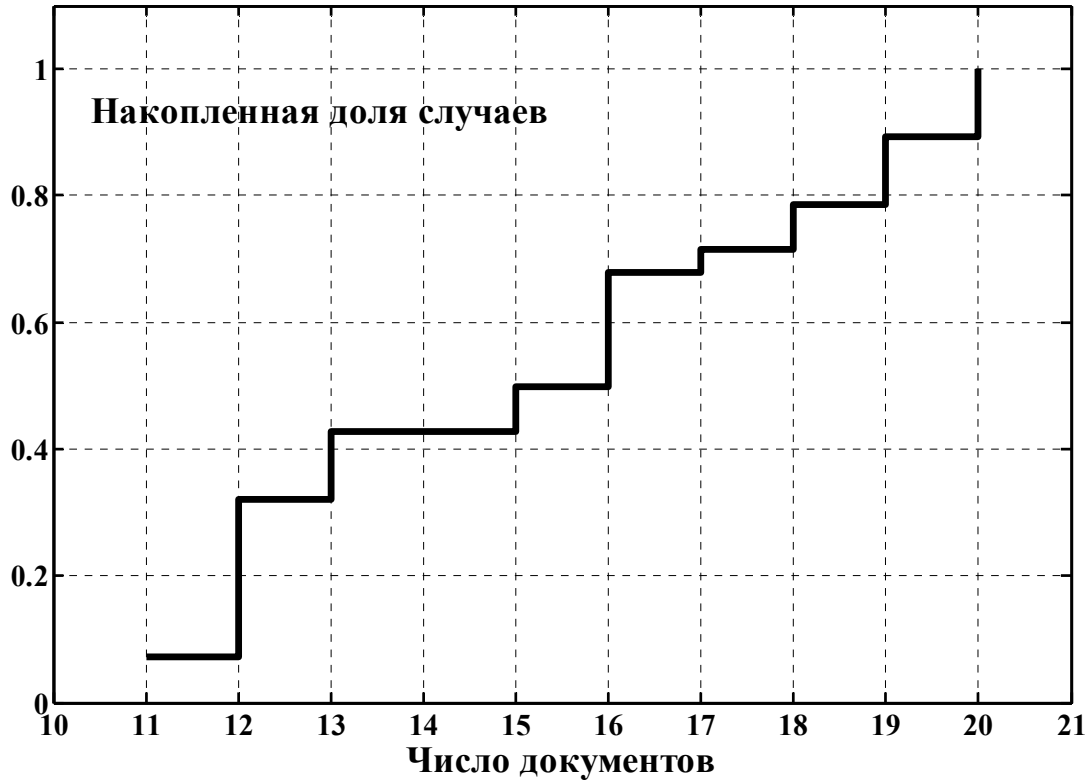


Рис. 3. Накопленные частоты случаев для числа документов на обработку

Построим гистограмму на Python. Для этого вначале вычислим частоты значений с помощью метода `histogram` библиотеки `numpy`. В метод надо передать 2 параметра:

- массив данных x
- диапазон изменений значений в массиве x.

Определим диапазон (метод `np.arange`) по минимальному и максимальному значениям. Учитывая особенность вычисления частот в методе `histogram`, верхняя граница диапазона значений должна превышать максимальное значение. Также надо учесть, что при задании диапазона методом `arrange` верхнюю границу надо указать больше последнего значения. Таким образом, границы диапазона передаваемые в метод `arange`: (`минимальное , максимальное+2`).

```
diap=np.arange(np.min(x),np.max(x)+2)
hist=np.histogram(x,diap)
print("Значения:", diap)
print("Частоты: ", hist[0])
```

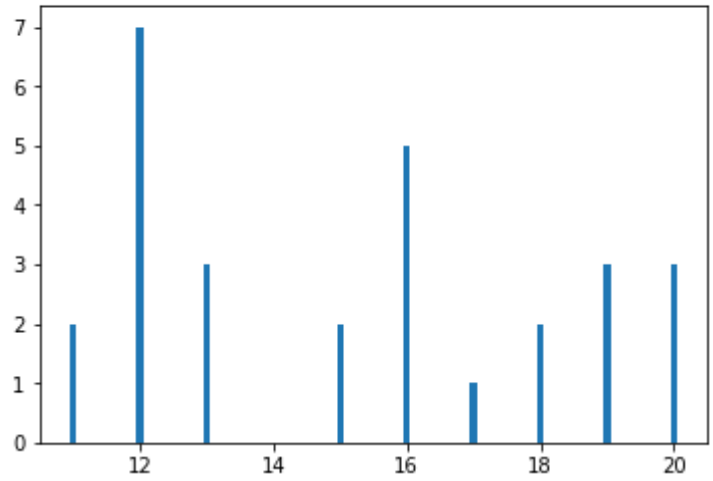
```
Значения: [11 12 13 14 15 16 17 18 19 20 21]
Частоты:  [2 7 3 0 2 5 1 2 3 3]
```

Для построения гистограммы воспользуемся методом `bar` библиотеки `matplotlib`. В метод надо передать 3 параметра:

- диапазон значений (получен ранее)
- массив частот (первая строка полученного объекта гистограммы)
- ширину столбика гистограммы относительно шага изменения значений (возьмем 0.1)

Так как размер массива диапазона на 1 больше размера массива часто, необходимо в первом параметре передавать массив, исключив последний элемент. Т.е. верхняя граница передаваемого массива «-1).

```
plt.bar(diap[:-1], hist[0], width = 0.1)
plt.show()
```



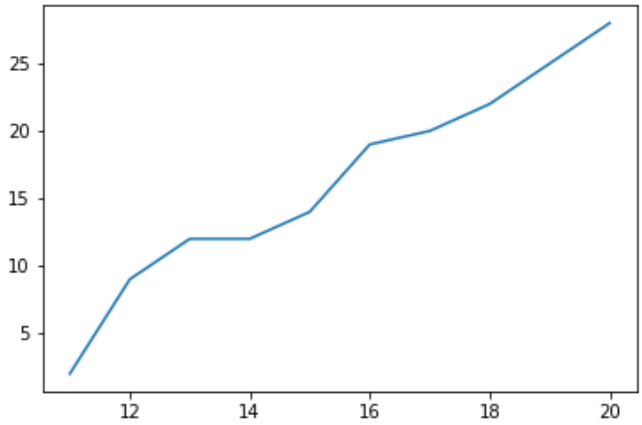
Построим график накопленных частот. Вначале воспользуемся функцией np.cumsum для вычисления накопленных частот:

```
hsum=np.cumsum(hist[0]) # Вычисление накопленных частот
print(hsum)
```

```
[ 2  9 12 12 14 19 20 22 25 28]
```

Попробуем построить график для этих частот

```
plt.plot(diap[:-1],hsum)
plt.show()
```



Чтобы получить более красивый ступенчатый график создадим вспомогательные массивы

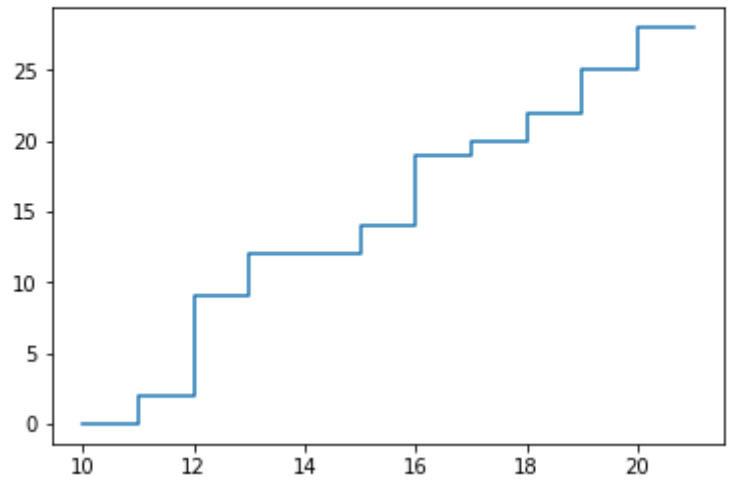
```
# формирование массивов для графиков накопленных частот
n= hist[0].size;
h2=np.array([0]*(hist[0].size*2+2)) # Размер массивов 2*n+2
d2=np.array([0]*h2.size)
# Цикл формирования массивов
i=0;
h2[0]=0;d2[0]=diap[0]-1
while (i<=n-1):
    h2[i*2+1]=h2[i*2]
    d2[i*2+1]=diap[i]
    h2[i*2+2]=hsum[i]
    d2[i*2+2]=diap[i]
    i=i+1
h2[h2.size-1]=h2[h2.size-2]
d2[d2.size-1]=diap[diap.size-2]+1
```

```
print (h2)
print(d2)
```

```
[ 0  0  2  2  9  9 12 12 12 12 14 14 19 19 20 20 22 22 25 25 28 28]
[10 11 11 12 12 13 13 14 14 15 15 16 16 17 17 18 18 19 19 20 20 21]
```

Построим график по вспомогательным массивам

```
plt.plot(d2,h2)
plt.show()
```



Задание

1. По диаграмме эволюции данных (рисунок 1) сделать вывод о наличии или отсутствии закономерностей в вариабельности данных за время наблюдений.
2. По гистограмме оценить наиболее вероятное количество документов, поступающих на обработку.
3. По кривой накопленных частот оценить вероятность того, что количество поступающих на обработку за день документов будет меньше 0,3.
4. По кривой накопленных частот оценить вероятность того, что количество поступающих на обработку за день документов будет больше 0,5.

5. По гистограмме (рисунок 2) оценить вероятность того, что число документов, поступающих на обработку в течение дня, будет от 14 до 17 включительно.

6. Какие выводы можно сделать из оценок центра положения данных, если разные оценки центра, как в приведенном примере, оказались примерно одинаковыми?

7. Почему для оценивания вариабельности используют дисперсию, среднеквадратическое отклонение и коэффициент вариации, если эти характеристики функционально связаны между собой?

Порядок выполнения работы

1. Изучить теоретический материал
2. Дать ответы на вопросы задания
3. Составить программы построения гистограмм и графиков накопленных частот
4. Выполнить программы

Содержание отчета

1. Ответы на вопросы задания
2. Программный код (блокнот) построения гистограмм и графиков накопленных частот
3. Скриншоты гистограмм и графиков

Практическое занятие № 25
Понятие корреляции. Примеры на pandas и numpy

Цель занятия. Получить представление об оценке связи между рядами данных, научиться вычислять количественные характеристики для оценки такой связи. Освоить средства Python для получения количественных характеристик связи и визуального их представления

Краткие теоретические сведения

Во многих задачах обработки и анализа данных необходимо оценить взаимосвязь между двумя рядами данных. Пример таких двух рядов данных наблюдений температуры и влажности воздуха по дням недели приведен в таблице 1.

Таблица 1 - Наблюдения погодных условий

День недели		**Пн**	**Вт**	**Ср**	**Чт**	**Пт**	**Сб**	**Вс**
Температура, град.	***T***	**15**	**18**	**21**	**22**	**16**	**17**	**14**
Влажность, %	***f***	**78**	**74**	**65**	**60**	**68**	**75**	**80**

Простейшей числовой характеристикой взаимосвязи между рядами данных является коэффициент корреляции (парный коэффициент корреляции). Процедура его вычисления включает следующие этапы:

а) вычисление средних для каждого ряда

$$Tsr = \frac{1}{7}(15+18+21+22+16+17+14) = 17{,}57$$

$$fsr = \frac{1}{7}(78+74+65+60+68+75+80) = 71{,}43$$

б) вычисление оценок дисперсии для каждого ряда

$$DT* = \frac{1}{n-1}\sum_{k=1}^{n}(T_k - Tsr)^2 = \frac{1}{n-1}\sum_{k=1}^{n}T_k^2 - \frac{n}{n-1}Tsr^2 =$$

$$= \frac{1}{7-1}\left(15^2 + ... + 14^2\right) - \frac{7}{7-1}17{,}57^2 = 8{,}95$$

$$Df* = \frac{1}{n-1}\sum_{k=1}^{n}(f_k - fsr)^2 = \frac{1}{n-1}\sum_{k=1}^{n}f_k^2 - \frac{n}{n-1}fsr^2 =$$

$$= \frac{1}{7-1}\left(78^2 + ... + 80^2\right) - \frac{7}{7-1}71{,}43^2 = 53{,}29$$

в) вычисление оценки так называемого корреляционного момента

$$K_{Tf} = \frac{1}{n-1}\sum_{j=1}^{n}\left(T_k - Tsr\right)\cdot\left(f_k - fsr\right) =$$

$$= \frac{1}{n-1}\sum_{j=1}^{n} T_k \cdot f_k - \frac{n}{n-1} Tsr \cdot fsr =$$

$$= \frac{1}{7-1}\left(15\cdot 78 + \ldots + 14\cdot 80\right) - \frac{7}{7-1} 17{,}57 \cdot 71{,}43 = -19{,}29$$

г) вычисление оценки коэффициента корреляции температуры и влажности воздуха

$$R_{Tf} = \frac{K_{Tf}}{\sqrt{DT^{*}\cdot Df^{*}}} = \frac{-19{,}29}{\sqrt{8{,}95\cdot 53{,}29}} = -0{,}88$$

В математической статистике доказано, что коэффициент корреляции любых данных лежит по абсолютной величине от 0 до 1, а по знаку может быть положительным и отрицательным. Если абсолютная величина коэффициента корреляции не превышает 0,2. То считается, что корреляция между переменными величинами отсутствует. Если это значение лежит в пределах от 0,2 до 0,3, считается, корреляция есть, но она мала и требует дополнительной проверки. Если значение коэффициента корреляции находится в пределах от 0,3 до 0,7, то принимается решение, что корреляция существует, но она не слишком сильна. Если $0{,}7 < |R| < 0{,}95$, то принимается, что корреляция сильная, и если $0{,}95 < |R| < 1$, то зависимость между переменными величинами практически совпадает с линейной функциональной зависимостью. Знак коэффициента корреляции указывает на направление корреляции: положительная корреляция говорит о том, что нарастающим значениям одной переменной отвечают нарастающие значения второй переменной. Отрицательная корреляция означает, что нарастающим значениям одной переменной отвечают убывающие значения другой переменной.

В рассмотренном примере большое абсолютное значение коэффициента корреляции $|R_{Tf}| = 0{,}88$ говорит о существовании сильной взаимосвязи между температурой и влажностью воздуха. Отрицательный знак коэффициента корреляции говорит о том, что с возрастанием температуры важность воздуха уменьшается и наоборот, с уменьшением температуры воздуха его влажность увеличивается. Следует подчеркнуть, что корреляция означает не жесткую функциональную зависимость между переменными величинами, а тенденцию взаимосвязи между ними.

Рассмотрим средства языка Python для оценки связи между рядами данных. Будем использовать библиотеки `numpy`, `pandas`, `matplotlib`.

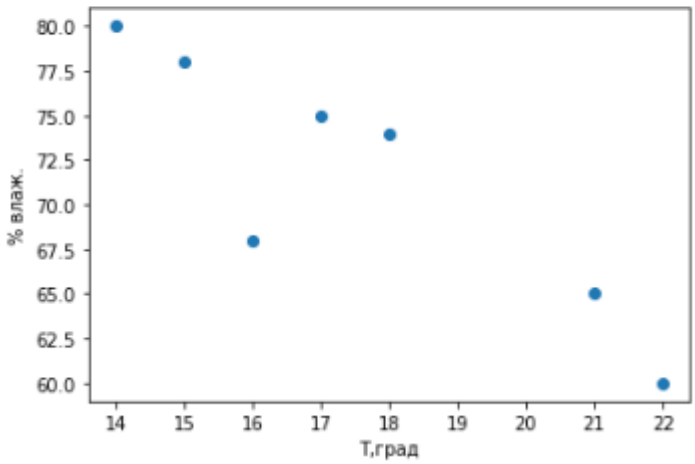
Создадим массив numpy по данным наблюдений (таблица 1):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# массив (тензор) наблюдений ранга 2
# 1-я строка - температура
# 2-я строка - влажность
x=np.array([[15,18,21,22,16,17,14],
            [78,74,65,60,68,75,80]])
print ("Температура:",x[0])
print ("Влажность:  ",x[1])
```

```
Температура: [15 18 21 22 16 17 14]
Влажность:   [78 74 65 60 68 75 80]
```

Один из самых быстрых и самых простых способов определить наличие взаимосвязи между двумя рядами данных состоит в выводе графика рассеяния. Это точечный график, который можно вывести средствами matplotlib.

```
plt.scatter(x[0],x[1])
plt.xlabel("T,град")
plt.ylabel("% влаж.")
plt.show()
```



Быстрый анализ показывает, что график вытянут из левого верхнего угла к нижнему правому. Т.е. при возрастании температуры, влажность падает. Такую связь принято называть обратной зависимостью. Никаких выводов о реальной взаимосвязи температуры и влажности по этим данным не будем, т.к. это гипотетический пример.

Для количественной оценки силы связи между двумя переменными является их ковариация (корреляционный момент) . Она измеряет тенденцию взаимосвязанного изменения двух переменных. Для вычисления ковариации можно использовать функцию `pandas cov`. Для того чтобы использовать данную функции необходимо массивы `numpy` преобразовать в набор `DataFrame`.

```
dx=pd.DataFrame({'Темп.':x[0],
                 'Влаж.':x[1]})
print(dx)
```

```
   Темп.  Влаж.
0     15     78
1     18     74
2     21     65
3     22     60
4     16     68
5     17     75
6     14     80
```

Вычислим ковариацию:

```
kov=dx['Темп.'].cov(dx['Влаж.'])
print("Ковариация=",np.round(kov,2))
```

```
Ковариация= -19.29
```

Ковариация температуры и влажности равна -19.29. Это число показывает направление связи, но степень (силу) связи сложно интерпретировать, т.к. величины температуры и влажности представлены в разных единицах измерения и имеют разные масштабы изменения.

Коэффициент корреляции (его еще называют корреляцией Пирсона) позволяет трансформировать единицы измерения в безразмерные величины, а количественную оценку степени связи ограничить числом в диапазоне между -1 и +1. Для вычисления коэффициента корреляции можно использовать функцию `pandas corr`:

```
r=dx['Темп.'].corr(dx['Влаж.'])
print("Корреляция=",np.round(r,2))
```

```
Корреляция= -0.88
```

Как видно из выполненных расчетов результаты, полученные на Python, совпадают с ручным расчетом.

Задание

Решить задачи

1. В таблице приведен фрагмент классного журнала с итоговыми оценками по стобалльной шкале за четверть 10 школьников по математике и литературе. Вычислите коэффициент корреляции между оценками по математике и литературе и дайте интерпретацию найденному значению.

№ ученика	1	2	3	4	5	6	7	8	9	10
Математика	45	87	65	73	77	82	90	71	84	79
Литература	65	32	72	84	75	88	70	67	75	68

2. Запишите ряд данных о курсе доллара и евро по дням в течение месяца (данные можно найти в Интернете). Вычислите коэффициент корреляции между курсом доллара и евро и дайте интерпретацию найденному значению.

3. Запишите ряд данных о курсе доллара и стоимости бочки (барреля) нефти по дням в течение месяца (данные можно найти в Интернете). Вычислите коэффициент корреляции между курсом доллара и ценой нефти и дайте интерпретацию найденному значению.

Порядок выполнения работы

1. Изучить теоретический материал
2. Выполнить ручной расчет по пунктам задания
3. Составить программный код выполнения расчетов
4. Выполнить программы

Содержание отчета

1. Результаты ручного расчета по пунктам задания
2. Программный код (блокнот) выполнения расчетов
3. Результаты программных расчетов

Практическое занятие № 26
Линейная регрессия на Python

Цель занятия. Получить представление о функциональном представлении связи между рядами данных, методике вычисления уравнения регрессии. Освоить средства Python для получения уравнений регрессии и оценки качества полученного уравнения

Краткие теоретические сведения

Регрессией называют функциональную зависимость математического ожидания случайной величины (или его оценки) от неслучайной переменной – аргумента функции. Эту функцию также называют регрессионной моделью данных. Регрессию получают обработкой данных, которые содержат два ряда связанных данных: ряд значений аргумента и ряд значений случайной величины, полученных для соответствующих значений аргумента. Важно, что эти значения случайны, т.е. могут отличаться от полученных значений при той же самой регрессии. Следовательно, значения, прогнозируемые функцией регрессии, не обязаны совпадать с имеющимися значениями, а должны быть лишь близки к ним в целом для всего ряда.

Регрессионным анализом называют процедуру выбора вида регрессионной функции, критерия оптимального подбора параметров этой функции - коэффициентов регрессии, получения оценок коэффициентов регрессии, оценки качества подобранной функции. Для осуществимости и осмысленности регрессионного анализа важно, чтобы выбранная в качестве аргумента переменная величина была упорядоченной, т.е. отображаемой на числовой оси.

Пример

В таблице 1 приведены данные наблюдений температуры воздуха в течение недели.

Таблица 1 - Наблюдения температуры по дням недели

День	**Пн**	**Вт**	**Ср**	**Чт**	**Пт**	**Сб**	**Вс**
t	**0**	**1**	**2**	**3**	**4**	**5**	**6**
***T*, град.**	**13**	**16**	**17**	**15**	**19**	**22**	**20**
***Tsr* (*t*)**	**13,68**	**14,93**	**16,18**	**17,43**	**18,68**	**19,93**	**21,18**

В качестве аргумента введена переменная *t*, для которой назначены целочисленные значения, начиная с нуля. Выполним процедуру регрессионного анализа.

А) Требуется построить функцию – регрессионную модель, описывающую зависимость среднего уровня температуры от дня недели, отображаемого переменной *t*. Анализируя изменение данных в таблице, приходим к выводу, что температура в среднем возрастает от понедельника к

воскресенью. Поэтому в качестве пробной модели выбираем линейную зависимость вида

$$Tsr(t) = a + b \cdot t$$

В этом выражении Tsr (t) - средняя температура в день t. Параметры a и b – неизвестные коэффициенты регрессии, которые предстоит оценить так, чтобы модель описывала имеющиеся данные.

Б) В качестве критерия оптимального подбора коэффициентов регрессии выберем критерий наименьших квадратов

$$Q(a,b) = \sum_{k=1}^{n} \left(T(k) - Tsr(t_k)\right)^2 \to \min_{a,b}$$

Применяя критерий наименьших квадратов к примеру, получим

$$Q(a,b) = \sum_{k=1}^{7} \left(T_k - a - b \cdot t_k\right)^2 = \left(13 - a - b \cdot 0\right)^2 + \left(16 - a - b \cdot 1\right)^2 + ... +$$

$$+ \left(20 - a - b \cdot 6\right)^3 \to \min_{a,b}$$

В) Методами математического анализа легко убедиться, что функция параметров Q (a, b) имеет только один экстремум, и этот экстремум – минимум, который достигается при значениях параметров, которые обращают в ноль частные производные критерия по этим параметрам

$$\begin{cases} \dfrac{\partial Q(a,b)}{\partial a} = -2\sum\limits_{k=1}^{n} \left(T_k - a - b \cdot t_k\right) = 0 \\ \dfrac{\partial Q(a,b)}{\partial b} = -2\sum\limits_{k=1}^{n} \left(T_k - a - b \cdot t_k\right) \cdot t_k = 0 \end{cases}$$

Тождественными алгебраическими преобразованиями приведем полученные уравнения к системе из двух линейных уравнений относительно параметров a и b

$$\begin{cases} a + b \cdot ts = Ts \\ a \cdot ts + b \cdot ts^2 = tTs \end{cases}$$

В этой системе

$$ts = \frac{1}{n}\sum_{k=1}^{n} t_k = \frac{1}{7}\left(0 + 1 + 2 + 3 + 4 + 5 + 6\right) = 3$$

$$Ts = \frac{1}{n}\sum_{k=1}^{n} T_k = \frac{1}{7}\left(13 + 16 + 17 + 15 + 19 + 22 + 20\right) = 17{,}43$$

$$ts^2 = \frac{1}{n}\sum_{k=1}^{n} t_k^2 = \frac{1}{7}\left(0^2 + 1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2\right) = 13$$

$$tTs = \frac{1}{n}\sum_{k=1}^{n} T_k \cdot t_k = \frac{1}{7}\left(13 \cdot 0 + 16 \cdot 1 + 17 \cdot 2 + 15 \cdot 3 + 19 \cdot 4 + 22 \cdot 5 + 20 \cdot 6\right) = 57{,}28$$

Решение системы дает наилучшие по методу наименьших квадратов оценки параметров – коэффициентов регрессии

$$a = \frac{Ts \cdot ts^2 - tTs \cdot ts}{ts^2 - (ts)^2} \qquad b = \frac{tTs - Ts \cdot ts}{ts^2 - (ts)^2}$$

В числовом выражении получим

$$a = \frac{Ts \cdot ts^2 - tTs \cdot ts}{ts^2 - (ts)^2} = \frac{17{,}43 \cdot 13 - 57{,}28 \cdot 3}{13 - 3^2} = 13{,}68$$

$$b = \frac{tTs - Ts \cdot ts}{ts^2 - (ts)^2} = \frac{57{,}28 - 17{,}43 \cdot 3}{13 - 3^2} = 1{,}25$$

Регрессионная модель имеет вид: $Tsr\ (\ t\ ) = 13{,}68 + 1{,}25\ t$

Г) Получим прогноз значений температуры $Tsr\ (\ t\ )$ по дням t= 0, 1,…, 6, который дает регрессионная модель. Значения приведены в последней строке таблицы. Для проверки достоверности полученной модели вычисляется коэффициент R^2

$$R^2 = 1 - \frac{Q_e}{Q_c}$$

$$Q_e = \frac{1}{n-r}\sum_{k=1}^{n}\left(T_k - Tsr(t_k)\right)^2$$

$$Q_c = \frac{1}{n-1}\sum_{k=1}^{n}\left(T_k - Ts\right)^2$$

В этих формулах дисперсия Q_e – это остаточная дисперсия ряда данных, которая описывает изменения этих данных относительно регрессионной моде Q_c – это полная дисперсия ряда данных; r – число оцениваемых коэффициентов регрессии. Чем ближе значение R^2 к единице, тем лучше построенная модель. В нашем примере r = 2; Q_e = 2,793; Q_c = 11,543; R^2 = 0,758. Это значение не слишком большое, чтобы считать регрессионную модель достоверной. Однако для такого небольшого ряда данных и большой изменчивости этих данных вряд ли удастся построить более достоверную модель.

На рисунке 1 приведены точки, отображающие анализируемый набор данных, линия регрессии, функция, описывающая регрессионную модель и значение R^2 , полученные с помощью инструментария ТОЧЕЧНЫЙ ГРАФИК и ДОБАВИТЬ ЛИНИЮ ТРЕНДА программы Excel. Инструкция и пример работы с инструментарием приведены в Справке к этой системе.

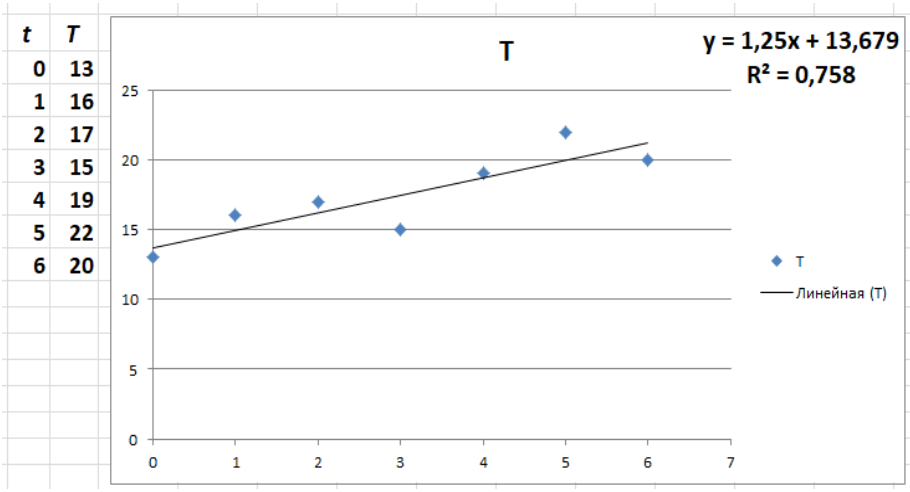


Рис. 1. График линии регрессии

Рассмотрим построение регрессионной модели на Python.

Вначале необходимо установить библиотеку `scikit-learn`. Это можно сделать через Anaconda (как описано в практическом занятии 19), а можно в Jupyter Notebook. Для этого необходимо создать новый файл и ввести команду: `!pip install scikit-learn`

```
!pip install scikit-learn
```

```
Collecting scikit-learn
  Downloading scikit_learn-1.0.1-cp38-cp38-win_amd64.whl (7.2 MB)
Collecting joblib>=0.11
  Downloading joblib-1.1.0-py2.py3-none-any.whl (306 kB)
Requirement already satisfied: numpy>=1.14.6 in e:\programs\anaconda3\envs\bigdat
(1.21.2)
Requirement already satisfied: scipy>=1.1.0 in e:\programs\anaconda3\envs\bigdata
7.1)
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.0.0-py3-none-any.whl (14 kB)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.1.0 scikit-learn-1.0.1 threadpoolctl-3.0.0
```

Далее можно приступать к вычислениям. Мы будем вычислять коэффициенты линейной модели регрессии:

$$Tsr(t) = a + b \cdot t$$

На первом шаге необходимо импортировать библиотеки и необходимые классы. Импортируем библиотеку NumPy и класс линейной регрессии `LinearRegression` из `sklearn.linear_model`:

```
import numpy as np
from sklearn.linear_model import LinearRegression
```

На следующем шаге создадим массивы по данным таблицы 1: массив входных данных `t` (первая строка таблицы) и массив выходных данных `tsr` (вторая строка таблицы 1).

```
t=np.array([0,1,2,3,4,5,6])
tsr=np.array([13,16,17,15,19,22,20])
```

Для использования модели регрессии преобразуем массив t в вектор столбец, использовав метод изменения формы массива reshape с аргументами -1, чтобы получить столько строк, сколько необходимо, и 1, чтобы получить один столбец: (-1,1).

```
t = t.reshape((-1, 1))
```

```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5],
       [6]])
```

На следующем шаге создадим модель линейной регрессии. Для этого создадим модель, являющуюся экземпляром класса LinearRegression:

```
model = LinearRegression()
```

Для вычисления коэффициентов модели используем метод fit, передав ему в качестве параметров входной и выходной массивы (t и tsr):

```
model.fit(t, tsr)
```

Получим из модели ее коэффициенты: атрибут intercept_ возвращает свободный коэффициент a, а атрибут coef_ - коэффициент при входной переменной:

```
print('a=', np.round(model.intercept_,3))
print('b=', np.round(model.coef_,3))
```

```
a= 13.679
b= [1.25]
```

Отметим, что свободный коэффициент является скаляром, а коэффициент при неизвестной - массивом. Это сделано так потому, что в общем случае модель линейной регрессии может включать несколько входных переменных

На следующем шаге выполним прогноз, вычислив по модели значения выходной переменной для известных значений входной. Для этого подставим в модель значения входного массива, вызвав метод predict:

```
t_pred = model.predict(t)
print('Предсказ.значения:', np.round(t_pred,2), sep='\n')
```

```
Предсказ.значения:
[13.68 14.93 16.18 17.43 18.68 19.93 21.18]
```

На следующем шаге проверим качество полученной модели, вычислив коэффициент детерминации R^2. Для этого воспользуемся методом score:

```
r_sq = model.score(t, tsr)
print('Коэф детерминации:', np.round(r_sq,3))
```

```
Коэф детерминации: 0.758
```

Отметим, что полученные результаты совпадают с данными ручных расчетов, представленных выше.

Задание

1. Получить формулы для оценки коэффициентов линейной регрессионной модели 2-го порядка $Ysr = a + b\,X + c\,X^2$, используя критерий наименьших квадратов и описанную выше процедуру получения оценок.
2. Получить формулы для оценки коэффициентов линейной регрессионной модели 1-го порядка для n аргументов - независимых переменных $Ysr = a0 + a1\,X1 + a2\,X2 + \ldots + an\,Xn$, используя критерий наименьших квадратов и описанную выше процедуру получения оценок.
3. Построить регрессионные модели для температуры воздуха и для влажности воздуха и выполнить регрессионный анализ, проверив качество полученных моделей.
4. Построить регрессионные модели для доходов Д фирмы по месяцам М и выполнить регрессионный анализ, проверив качество полученных моделей. Данные приведены в таблице.

М	**Янв**	**Февр**	**Март**	**Апр**	**Май**	**Июнь**
Д	**35**	**67**	**90**	**110**	**125**	**100**
М	**Июль**	**Август**	**Сент.**	**Окт.**	**Ноя.**	**Дек.**
Д	**85**	**70**	**65**	**40**	**30**	**15**

Используйте для этих данных линейную регрессионную модель 2-го порядка.

Порядок выполнения работы

1. Изучить теоретический материал
2. Выполнить ручной расчет по пунктам задания
3. Составить программный код выполнения расчетов
4. Выполнить программы

Содержание отчета

1. Результаты ручного расчета по пунктам задания
2. Программный код (блокнот) выполнения расчетов
3. Результаты программных расчетов

ч.3 Хранение, обработка и управление большими данными

В данной части сборника рассматриваются Базовые понятия и технологии хранения больших данных, базовые понятия реляционных баз данных, технологии сбора, подготовки, обработки и анализа больших данных. Рассматриваются вопросы установки и настройки программных средств моделирования, проектирования и управления базами данных, типовые алгоритмы обработки и анализа больших данных

11 класс 1-е полугодие

Оборудование для проведения занятий

1. Рабочая станция ученика (Intel i5, 8Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)
2. Рабочая станция учителя (Intel i7, 16Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)

Программное обеспечение для проведения занятий (в том числе системное ПО)

1. ОС Windows 10
2. MS Office 2016
3. SQL Power Architect
4. PostgreSQL
5. PyCharm CE (Anaconda)
6. Python 3.7

Рекомендации учителю по проведению занятий.

Для проведения практических занятий данного цикла необходимо установить программное средство проектирования баз данных SQL Power Architect, реляционную СУБД PostgreSQL и интерактивную оболочку PgAdmin для взаимодействия с базами данных.

Процесс установки SQL Power Architect и вопросы проектирования реляционных баз данных подробно рассмотрены в материале практических занятий 31 и 32.

Процесс установки СУБД PostgreSQL и правила работы с интерактивной оболочкой PgAdmin подробно разобраны на практическом занятии 33.

Занятия данного цикла посвящены проектированию реляционных баз данных, выполнения операций с базами данных на языке SQL. Знания архитектуры реляционных баз данных и языка SQL является важным, т.к. часто данные для последующего анализа извлекаются из баз данных.

В ходе данного цикла учащиеся познакомятся со средствами аналитической обработки данных для крупномасштабных мощных приложений распределенной обработки данных и машинного обучения Apache Spark (практическое занятие 41) . Данную платформу тяжело развернуть в рамках школьного компьютерного класса. Но освоить функционал Apache Spark можно с использованием PySpark - библиотеки Spark, написанной на Python. Для работы с PySpark в приложении Jupyter Notebook необходимо загрузить библиотеку pyspark в экосистему Anaconda.

Все практические занятия включают краткие теоретические сведения по теме занятия, типовые задачи с разбором их решения, вопросы для проверки усвоения материала.

В начале занятия необходимо проверить выполнение заданий для самостоятельного выполнения, выданных на предыдущем практическом занятии.

В ходе проведения занятия рекомендуется изложить учащимся краткие теоретические сведения, разобрать решение типовых задач, описанных в практических занятиях. Для усвоения материала необходимо выдать задания для самостоятельной проработки материала, разобранного на практическом занятии

Практическое занятие №27
Хранение и обработка данных. Основные виды баз данных

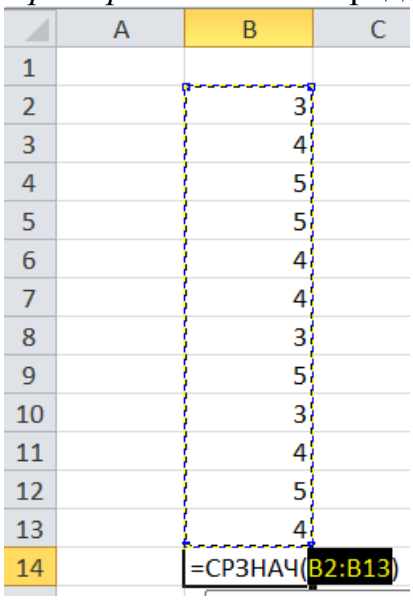
Цель занятия. Познакомиться со средствами хранения и обработки различных типов данных. Получить представление о реляционных и NoSQL базах данных, их отличиях, областях применения

Краткие теоретические сведения

Мы живем в век информации. Все что нас окружает можно представить в виде чисел, текста или в структурированной табличной форме. Необходимо уметь получать, хранить и обрабатывать эту информацию для решения различных прикладных задач. Например, получить и проанализировать успеваемость по оценкам, оценить характер отзывов на фильм (положительный или отрицательный) по текстам отзывов, оценить комфортность по различным критериям (количественным и категориальным) столиц различных государств.

Удобным средством хранения и обработки табличных данных является табличный процессор MS Excel.

Пример. Вычислить средний балл оценок по математике в классе.



Упражнение. Вычислите среднее арифметическое значение результатов контрольной работы по любому предмету.

В данном случае обработкой данных управляет человек. Для ускорения обработки используют ЭВМ и программные средства.

Чтобы перейти к вопросам программной обработки информации, рассмотрим кратко понятие информации. Определений данного понятия существует множество. Основоположник кибернетики Норберт Винер дал следующее определение информации: «Информация — это обозначение

содержания, полученное нами из внешнего мира в процессе приспосабливания к нему нас и наших чувств». С позиций использования информации можно дать такое определение. «Информация – это знания относительно фактов, событий, вещей, идей и понятий, которые в определённом контексте имеют конкретный смысл» (ISO/IEC 2382:2015 Информационные технологии - Словарь).

Замечание. Часто также используется термин «данные». В принципе термины «данные» и «информация» трактуются различно. *Данные* – это значения, реально используемые при вычислениях. *Информация* – это смысловое значение данных с точки зрения человека. Мы будем рассматривать эти термины как синонимы.

Элементы реального мира, информацию о которых мы получаем, называют объектами. Объекты могут быть материальными (учащийся, продукт, город) и нематериальными (заказ, налог). Объекты имеют различные свойства или атрибуты (фамилия, цвет, наименование, стоимость). При обработке данных в программных системах имеют дело с совокупностью однородных объектов (например, учащиеся одной школы), записывая информацию об одних и тех же свойствах каждого из них.

Для программной обработки информации важно иметь удобные структуры описания данных. Самый простой способ связать свойства конкретных объектов с элементами данных (атрибутами) – записать данные в виде фиксированной последовательности. Простое двумерное отображение данных называется плоским файлом.

Например, Информация об учащихся

Класс	ФИО	Дата рождения
10А	Акулов И.С.	10.03.2006
….		

Совокупность хранимых данных традиционно называют базой данных (БД). Часто к этому термину относят любые файлы данных. Исторически первыми системами хранения данных были именно файловые системы, в которых вся информация сохранялась в файлах.

Но реализация хранения данных в виде обычных файлов несет много неудобств. Почему так происходит? При использовании файлов управление данными ложится на прикладную программу, которая эти данные использует. Проблем не будет, если данные использует только одна программа. Но, если к этим файлам будет обращаться сразу много программ, каждая из которых захочет получить данные или внести изменения, то необходимо управлять этим процессом. Организовать такой доступ на уровне прикладных программ файловой системы невозможно. Поэтому для хранения данных и управления процессами хранения и извлечения данных необходимо использовать специальные программные средства. Такие средства необходимы и потому, чтобы не перегружать программы обработки данных функциями управления

доступом к данным, и не делать программы обработки зависимым от особенностей данных.

Такое специальное программное средство получило название Система Управления Базами Данных (СУБД) или сервер баз данных. Ее основными функциями является обеспечение централизованного хранения и доступа к данным.

СУБД управляет доступом к БД. *Основными функциями* СУБД являются следующие:

- определение данных – описание хранимых данных (структур данных);
- обработка данных – обработка запросов пользователя на выборку, добавление, изменение и удаление данных в БД;
- обеспечение безопасного и надежного хранения данных во внешней памяти, контроль пользовательских запросов и предотвращение попыток несанкционированного;
- обеспечение целостности (правильности) данных в любой момент времени;
- обеспечение параллельной работы многих пользователей и согласованности операций, выполняемых несколькими пользователями одновременно с одним и тем же фрагментом данных;
- восстановление данных после любого программного или аппаратного сбоя и ведение архива.

Концептуально данные можно представить в виде некоторой логической структуры. Такую логическую структуру хранимых данных называют ***моделью представления данных*** *(как представляются объекты и как они связаны между собой).*

Модель данных – это абстрактное, логическое определение *объектов, операторов для работы с ними*, в совокупности составляющих машину доступа к данным, с которой взаимодействует пользователь. *Объекты моделируют структуру данных, а операторы – поведение данных*

К основным моделям представления данных (или кратко *моделям данных*) относятся следующие: системы с инвертированными списками, иерархическая, сетевая, реляционная, постреляционная, в последнее время NoSQL (ключ-значение, документные, графовые, с расширяемыми колонками).

Наиболее распространенными являются базы данных, использующие реляционную модель данных, предложенную Э.Коддом. *Реляционная модель характеризуется* следующими особенностями:

1. *Данные представлены строками в таблицах* (кортежами в отношениях) и эти строки можно рассматривать как истинные высказывания. Например, строку об учащемся можно рассматривать как высказывание: «Учетный номер 1001 имеет учащийся Иванов Иван Иванович в классе 10А». Или «учащийся с учетным номером 1001 отсутствовал 10.02.2021 года на занятии математики».

2. *Для обработки данных (строк или кортежей) предоставляются операторы* которые поддерживают процесс логического вывода новых истинных высказываний из существующих. Например, 10.02.2021 отсутствовали: Иванов, Махов, Семенов.

Упражнение. Придумайте несколько новых высказываний из уже существующих, отражающих информацию о проведении урока математики.

Основной областью применения реляционных баз данных являются разнообразные экономические системы, которые можно отнести к так называемым транзакционным системам OLTP (On-Line Transaction Processing). Такие системы предполагают интенсивное изменение данных, причем операции изменения данных (транзакций) малы по объему обрабатываемой информации. Таким условиям удовлетворяют системы обработки информации на производственных предприятиях, в банковской сфере, торговле и т.п.

Пример. Реляционная модель данных, отражающая результаты сдачи экзаменов, может включать таблицы: Классы, Учащиеся, Предметы, Экзамены.

Но реляционные базы данных подходят не для всех предметных областей. К числу недостатков реляционных БД можно отнести:

- невозможность обработки разнообразных неструктурированных данных, количество которых постоянно возрастает, и которые трудно привести к табличным структурам;

- жесткая структура данных. Бывает проблематично изменить таблицы и связи между ними. Изменение отношений (связей) между таблицами или добавление новой таблицы может повлиять на существующие отношения (связи). Это требует изменение схемы. В реляционных БД каждая запись таблицы должна иметь одинаковое количество столбцов. И, например, при добавлении нового столбца нужно решать, как заполнить значения данного столбца в уже существующих строках.

В таких случаях необходимо использовать другие модели данных. В последнее время популярность приобрели базы данных **NoSQL (**«Not only SQL» переводится как «Не только реляционные» базы данных»).

Особенностями БД NoSQL являются:

- не жесткая схема данных;
- возможность управлять разнообразными большими объемами данных.

Базы данных NoSQL делятся на несколько основных видов:

- хранилища типа «ключ-значение»;
- хранилище с широкими столбцами или хранилища расширяемых записей;
- документные БД;
- графовые БД.

Модель хранилища **Ключ-значенние** (key-value) использует для доступа к данным хеш-таблицу, в которой значение уникального ключа

является указателем на конкретный элемент данных. Такая структура данных похожа на телефонный справочник, в котором имена людей и номера их телефонов сопоставляются друг с другом. Областью использования таких хранилищ являются системы, используемые в социальных сетях для хранения комментариев, списков пользователей, подписчиков. Данную модель поддерживают БД Riak, Redis.

Модель хранилища **расширяемых записей** (wide column) является хранилищем столбцов, т.е. столбцы каждой строки содержатся в этой строке. Отдельные строки могут иметь *различное количество* столбцов. Столбцы могут быть добавлены в любую строку в любое время без необходимости добавлять ее в другие строки. Примером использования таких хранилищ являются системы для хранения атрибутов профиля пользователя и метаданных (рис. 1). Данная модель реализована в БД Cassandra.

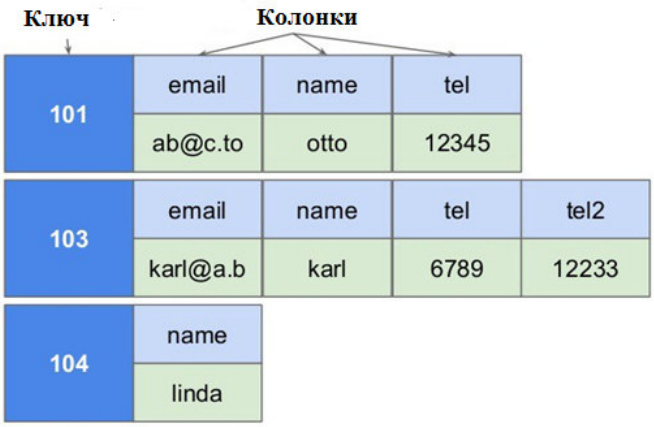


Рис. 1 Модель хранилища расширяемых записей

Модель хранилища **документов** (document) используют для хранения документов, представленных в форматах JSON, XML. Хранилище документов можно использовать для сохранения данных клиентов и их заказов (рис.2). Данная модель реализована в БД MongoDB. Например, компания SEGA использует БД MongoDB для управления 11 миллионами учетных записей игроков.

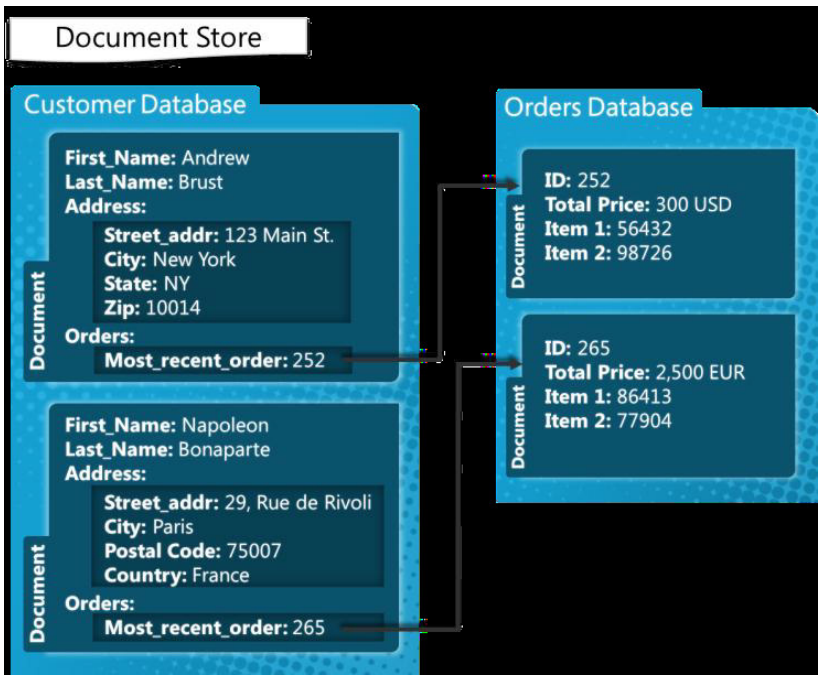


Рис. 2 Документная БД «Клиенты-Заказы»

В модели **графовых (**graph DB**)** баз данных основными элементами являются узлы и связи. Узел представляет собой объект. Связь представляет как два узла связаны (рис. 3). Данная модель реализована в БД Neo4J, HyperGraphDB.

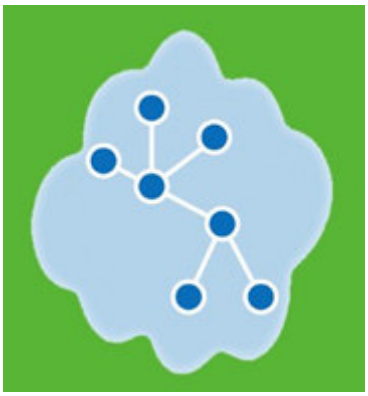


Рис. 3 Модель графовой БД

Подводя итог данного занятия можно отметить, что фактически именно тип данных, которые будут использоваться в той или иной задаче, определяет выбираемую модель данных.

На следующих занятиях мы будем рассматривать реляционную модель данных и реляционную СУБД PostgreSQL.

Контрольные вопросы

1. Дайте понятие информации
2. В чем различие терминов «информация» и «данные» ?
3. Приведите примеры материальных и нематериальных объектов

4. Что называется файлом ?

5. В чем недостаток файловых систем для хранения и управления данными ?

6. Каково основное назначение сервера баз данных (СУБД) ?

7. Назовите основные функции СУБД

8. Дайте определение модели данных

9. Какие основные модели данных существуют ?

10. Дайте характеристику реляционной модели данных

11. В чем особенность баз данных NoSQL ?

12. Назовите основные виды баз данных NoSQL

Порядок выполнения работы

1. Изучить теоретический материал

2. Ответить на контрольные вопросы

Содержание отчета

1.Ответы на контрольные вопросы

Практическое занятие № 28
Реляционная модель данных. Реляционные структуры данных

Цель занятия. Познакомиться с реляционной моделью данных. Получить представление о реляционных структурах данных

Краткие теоретические сведения

Первыми моделями, реализованными в БД, были системы инвертированных списков, иерархические и сетевые. Но все они имели недостатки: сложные в использовании, зависимость прикладных программ от организации данных.

В теории БД был найден подход, позволивший упростить описание данных. Таким подходом стал метод **нормализации**. Этот метод был разработан Э.Коддом в 1970 г., когда он сформулировал принципы реляционной модели данных [*«A Relational Model of Data for Large Shared Data Banks»,* CASM, 1970, № 6). Наиболее распространенная трактовка реляционной модели данных принадлежит К.Дж.Дейту. Согласно Дейту реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода:

- структура данных (объекты данных)
- целостность данных (правильность данных)
- обработка данных

В структурной части модели рассматриваются основные объекты (домены, отношения). Э.Ф.Кодд, математик по образованию, понял, что математические дисциплины можно использовать, чтобы привнести в область управления данными строгие принципы и точность. Кодд предложил использовать для описания и обработки данных аппарат теории множеств. Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как ***отношение*** (англ. ***relation)***.

Основными понятиями реляционных баз данных являются тип данных, атомарное значение, домен, атрибут, кортеж, первичный ключ и отношение. Рассмотрим смысл этих понятий на примере отношения Сотрудники, содержащего информацию о сотрудниках некоторой организации (рис.1).

Атомарное значение данных - это наименьшая единица данных реляционной модели – неразложимое на более мелкие элементы без потери смысла. Например, фамилия состоит из букв, но рассматривая фамилию по буквам мы потеряем значение (смысл).

Доменом(D) называется множество атомарных значений одного и того же типа и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена. Так, на рисунке домен «фамилия сотрудника» – множество правильных фамилий, а домен

«оклад» – множество действительных положительных чисел (денежного типа).

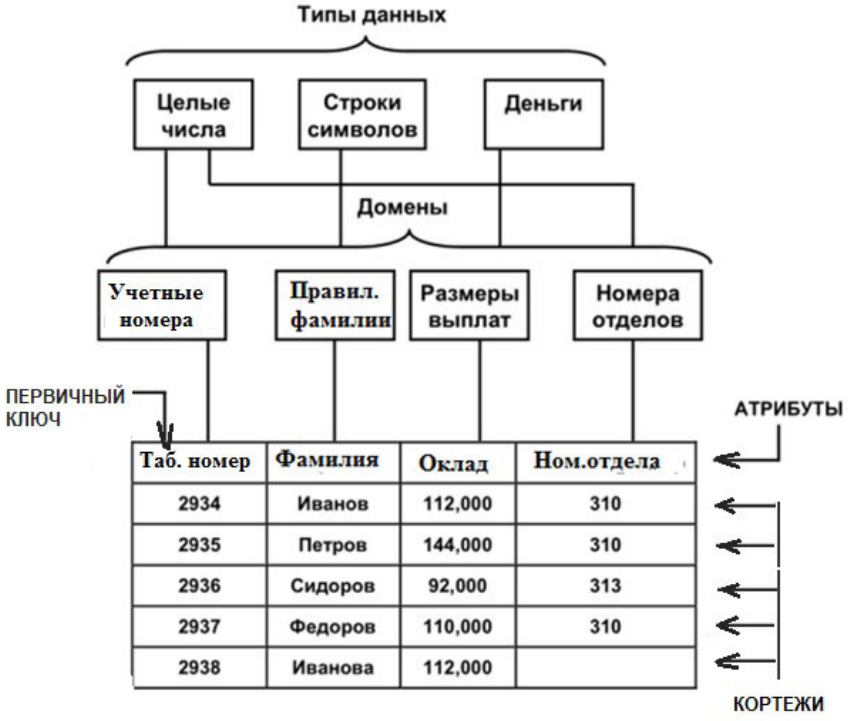


Рис. 1 Отношение Сотрудники

Отношение на доменах D1, D2, ..., Dn состоит из заголовка и тела.

Заголовок состоит из фиксированного множества атрибутов A1, A2, ..., An, и множества определяющих их доменов Di (i=1,2,...,n). Т.е заголовок - этомножество пар <A1:D1>, <A2:D2>, …

Все имена атрибутов уникальны в пределах одного отношения.

В рассмотренном примере заголовок

<Таб.номер: Таб.номера>, <Фамилия: Фамилии>,<Оклад:Денежные выплаты>,

<Номер отдела:Номера отделов>

Тело отношения состоит из меняющегося во времени множества ***кортежей***, где каждый кортеж состоит из множества пар «атрибут-значение» (Ai:Vi), (i=1,2,...,n),. Для любой заданной пары Vi является значением из домена Di.

В рассмотренном примере тело:

<Таб.номер: 2934>, <Фамилия: Иванов>,<Оклад:112000>,<НомОтдела:310>

<Таб.номер: 2935>, <Фамилия: Петров>,<Оклад:144000>,<НомОтдела:310>

…..

Степень отношения – это число его атрибутов. Отношение степени 1 - унарное,2 - бинарное, 3 – тренарное, ..., а степени n – n-арное.

Кардинальное число или ***мощность отношения*** – это число его кортежей, т.е. число заполненных строк.

При фиксированной структуре данных кардинальное число отношения изменяется во времени, а степень - нет.

Задание

1. Назовите степень отношения Сотрудники.

2. Назовите мощность отношения Сотрудники.

3. Составьте реляционные отношения для объектов:

Учащийся
Класс
Предмет

Порядок выполнения работы

1. Изучить теоретический материал
2. Выполнить задание

Содержание отчета

1. Основные определения и термины
2. Результаты выполнения задания

Практическое занятие № 29
Реляционная модель данных. Целостность данных

Цель занятия. Познакомиться с понятием целостности реляционной моделью данных. Получить представление о средствах поддержания целостности: первичных и внешних ключах.

Краткие теоретические сведения

В любой момент времени любая БД содержит некоторый набор значений данных и предполагается, что эти данные «отражают действительность», т.е. является моделью части реального мира. Произвольные значения могут не иметь смысл.

Примеры некорректных значений:

- нулевой возраст у человека
- отрицательный остаток наличных средств в кассе
- оценка на экзамене 7 при пятибалльной шкале оценивания
- наличие у двух разных людей паспорта с одним номером
- указание в адресе проживания несуществующего номера дома

Следовательно, БД надо дополнить правилами, назначение которых информировать СУБД о различных ограничениях реального мира, а значит предотвращать появление недопустимых конфигураций данных. Такие правила называются *правилами целостности*

Т.о. *целостность* можно упрощенно определить как правильность данных в любой момент времени.

Выделяют следующие правила целостности:

1. Целостность атрибута
2. Целостность отношений.
3. Целостность по ссылкам.
4. Целостность, определяемая пользователем.

Три первых правила являются базовыми в реляционной модели данных и должны поддерживаться в любой реляционной СУБД.

Целостность атрибута.

Первое правило требует, чтобы значения каждого атрибута брались из соответствующего домена (рис. 2).

Целостность отношений. Первичный ключ.

Объекту реального мира в реляционных БД соответствует кортеж отношения. Целостность отношения требует, чтобы кортеж любого отношения был отличим от любого другого кортежа этого отношения, т.е. другими словами, любое отношение должно обладать первичным ключом.

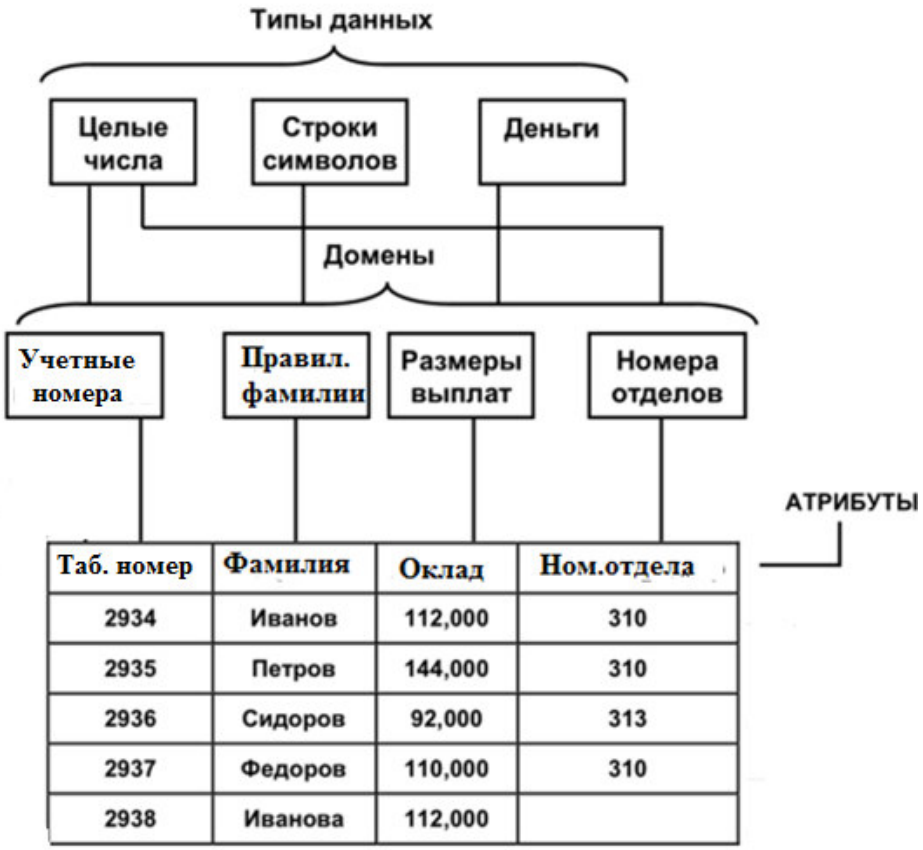


Рис.1 Целостность атрибута

Первичный ключ **(Primary Key - PK)** – это набор атрибутов, обладающий следующими свойствами:

- свойством *уникальности*: нет двух разных кортежей с одинаковыми значениями первичного ключа (рис.3).
- свойством *неизбыточности*: никакое подмножество атрибутов первичного ключа не обладает свойством уникальности (исключение из набора любого атрибута уже не позволяет идентифицировать кортеж по оставшимся атрибутам.). Пример избыточного первичного ключа для отношения Сотрудник будет ключ «Таб.номер+Номер отдела» (рис.4). После удаления атрибута «Номер отдела» первичный ключ будет по-прежнему однозначно идентифицировать кортеж.

Рис.2 Свойство уникальности первичного ключа

ПЕРВИЧНЫЙ КЛЮЧ

Таб.номер	Номер отдела	Фамилия	Оклад
2934	310	Иванов	112,000
2935	310	Петров	144,000
2934	313	Сидоров	92,000
2937	310	Федоров	110,000
2938		Иванова	112,000

Рис.3 Пример избыточного первичного ключа

При выборе первичного ключа следует отдавать предпочтение несоставным ключам или ключам, составленным из минимального числа атрибутов. Нецелесообразно также использовать ключи с длинными текстовыми значениями (предпочтительнее использовать цифровые символьные значения или целочисленные атрибуты). Учитывая выше сказанное, в качестве РК следует выбрать атрибут «Таб.номер».

Причина важности первичных ключей заключается в том, что они обеспечивают основной механизм адресации на уровне кортежей. Т.е. единственный способ точно указать на какой-нибудь кортеж – это указать значение первичного ключа.

Например, с помощью условия `Таб.номер ='2934'` мы получим строго один кортеж.

Целостность по ссылкам. Внешний ключ.

Третье правило является более сложным. При соблюдении нормализованности отношений сложные сущности реального мира представляются в реляционной БД в виде нескольких кортежей нескольких отношений. Рассмотрим представление информации о сотрудниках отдела. В реляционной БД потребуется два отношения: Отдел и Сотрудник (рис.5)

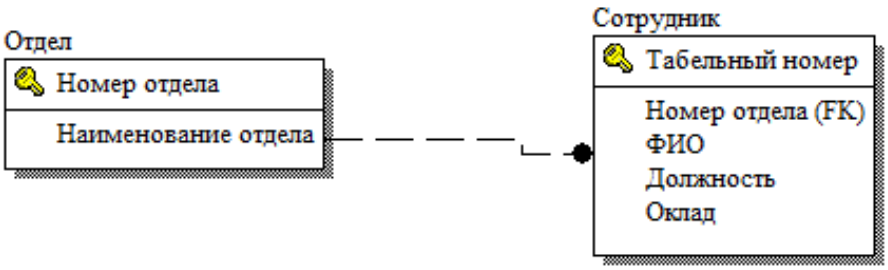


Рис.4 Отношения Отдел и Сотрудник

Атрибут «Номер отдела» появляется в отношении Сотрудник не потому, что номер отдела является собственным свойством сотрудника, а лишь для того, чтобы иметь возможность восстановить при необходимости полную сущность Отдел. Значение атрибута «Номер отдела» в любом кортеже отношения Сотрудник должно соответствовать значению атрибута «Номер отдела» в некотором кортеже отношения Отдел. Атрибут такого рода называется ***внешним ключом*** **(Foreign Key FK)**, поскольку его значения однозначно определяют кортежи некоторого другого отношения (т.е. задают значения их первичного ключа).

Отношение, которое содержит внешний ключ называется **ссылающимся отношением**, а отношение, которое содержит соответствующий первичный ключ – **целевым отношением** (рис. 6)

Рис.5 Связь отношений Отдел и Сотрудник

Требование целостности по ссылкам заключается в следующем: для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, должен найтись кортеж с таким же значением первичного ключа в целевом отношении, либо значение внешнего ключа должно быть неопределенным (т.е. ни на что не указывать).

Для нашего примера это означает, что если для сотрудника указан номер отдела, то этот отдел должен существовать (рис.7).

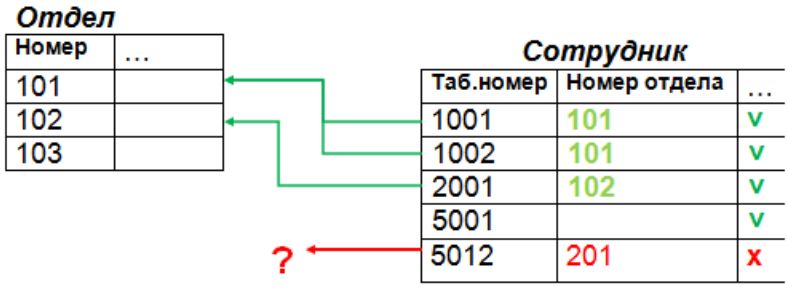


Рис.6 Правильные и неправильные ссылки

Любое состояние БД не удовлетворяющее правилу ссылочной целостности некорректно. Как избежать таких некорректных состояний?

Первый вариант – **запретить** любые операции, которые могут привести к некорректному состоянию. *Например*, запретить удалять отделы при наличии в них сотрудников.

Второй вариант – **допускать** такие операции, **но** предусмотреть выполнение **компенсирующих операций**, которые обеспечивают корректное состояние системы.

Например, при удалении конкретного отдела в отношении Отдел необходимо удалить всех сотрудников этого отдела в отношении Сотрудник (или вывести их из удаляемого отдела).

Прежде чем продолжить сделаем маленькое отступление и введем понятие **NULL-значений**

В БД существует проблема отсутствия данных. Было предложено обозначать факт отсутствия информации специальными маркерами **NULL-значениями**.Если атрибут имеет NULL-значение, то это означает, что в данной позиции значение отсутствует (например, для сотрудника неизвестна принадлежность к конкретному отделу).

В определении атрибута есть спецификации:

NULL – разрешено иметь NULL-значения

NOT NULL – не разрешено иметь NULL-значения

Есть ограничения на допустимость значений NULL.

В первичных ключах: с точки зрения целостности объектов ***ни один элемент первичного ключа не может быть NULL-значением.*** Причина такого требования – объекты должны быть различимыми.

Во внешних ключах реляционная модель разрешает появление NULL-значений.

Все операции над реляционными данными можно условно разделить на «опасные» и «неопасные». Опасными будем называть операции, которые могут привести к неправильным конфигурациям данных. В контексте ссылочной целостности «опасная» операция может привести к неправильным ссылкам. К опасным операциям можно отнести:

а) в целевом (родительском) отношении (в примере ОТДЕЛ):

- удаление записи

- изменение значения ключевого атрибута

б) в ссылающемся (дочернем) отношении (в примере Сотрудник):

- добавление (вставка) новой записи
- изменение значения внешнего ключа

«Опасные» операции можно дополнительно разделить на две группы: запрещенные и условно разрешенные. В ссылающемся отношении Сотрудник все «опасные» операции, приводящие к нарушению ссылочной целостности запрещены тем, что СУБД выполняет контроль – проверку значения внешнего ключа. Нельзя вставить кортеж или изменить кортеж в дочернем отношении, если значение внешнего ключа некорректно (отсутствует такое же значение первичного ключа в целевом отношении).

В целевом же отношении (в примере Отдел) «опасные» операции разрешены при условии выполнения компенсирующих действий.

Поэтому необходимо определить правила поведения при выполнении таких операций.

Рассмотрим операцию **УДАЛИТЬ** кортеж в целевом отношении, на который есть ссылка. Например, операция удаления кортежа отдела, в котором есть сотрудники (рис. 8).

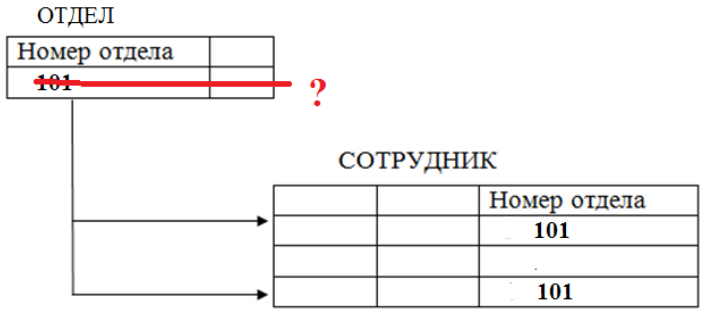


Рис.7 «Опасная» операция удаления кортежа в целевом отношении

Имеется 3 варианта действий при выполнении такой операции.

Первый вариант - ОГРАНИЧИВАТЬ (RESTRICT). Удалять разрешено лишь те отделы, которые еще не имеют сотрудников. Иначе операция удаления отвергается (рис. 9).

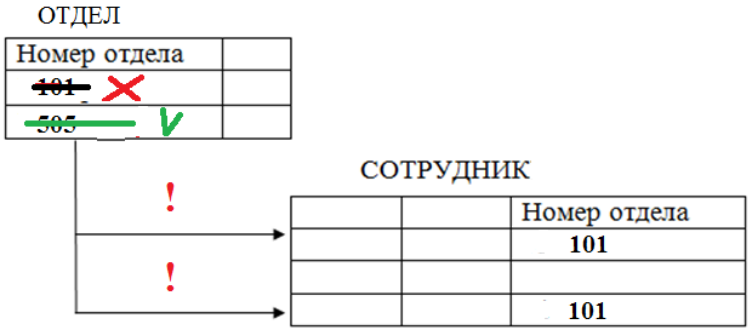


Рис.8 Реализация правила ОГРАНИЧИТЬ

Второй вариант - ***КАСКАДИРОВАТЬ (CASCADE).*** Операция удаления «каскадируется» (распространяется) на ссылающееся отношение (в примере

на отношение Сотрудник). Операция выполняется в 2 шага. На первом шаге удаляются сотрудники удаляемого отдела, а на втором - удаляется отдел (рис. 10).

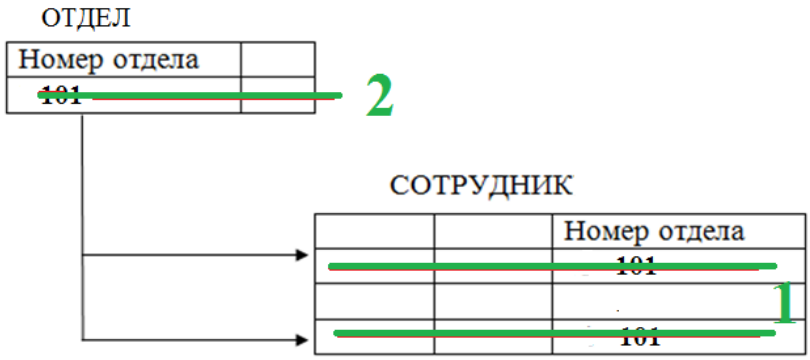


Рис.9 Реализация правила КАСКАДИРОВАТЬ

Третий вариант - **СБРАСЫВАТЬ ССЫЛКУ (SET NULL).** Для всех сотрудников удаляемого отдела на первом шаге внешний ключ устанавливается в неопределенное значение, а на втором – удаляется отдел (рис. 11). Исключением (нельзя применять данное правило) является ситуация, когда FK не должен содержать неопределенные значения null.

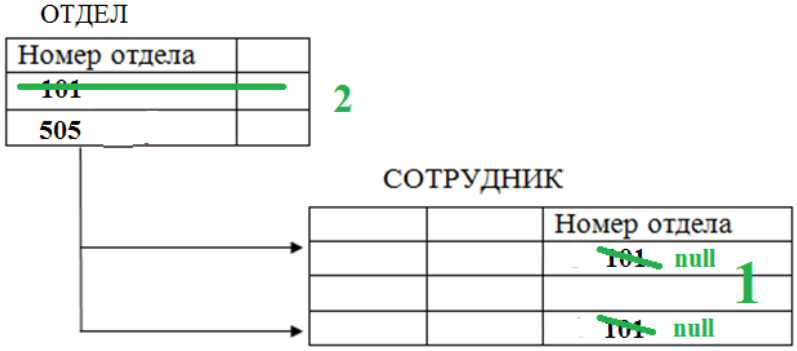


Рис.10 Реализация правила СБРАСЫВАТЬ ССЫЛКУ

Рассмотрим другую «опасную» операцию **ИЗМЕНИТЬ** кортеж в целевом отношении, на который есть ссылка. Например, операция удаления кортежа отдела, в котором есть сотрудники (рис. 12)

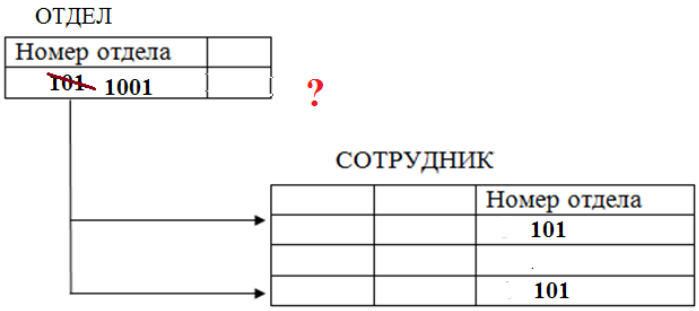


Рис.11 «Опасная» операция изменения кортежа в целевом отношении

Имеется 3 варианта действий при выполнении такой операции.

Первый вариант - ОГРАНИЧИВАТЬ (RESTRICT). Изменять первичный ключ разрешено лишь для тех отделов, которые еще не имеют сотрудников. Иначе операция отвергается (рис. 13).

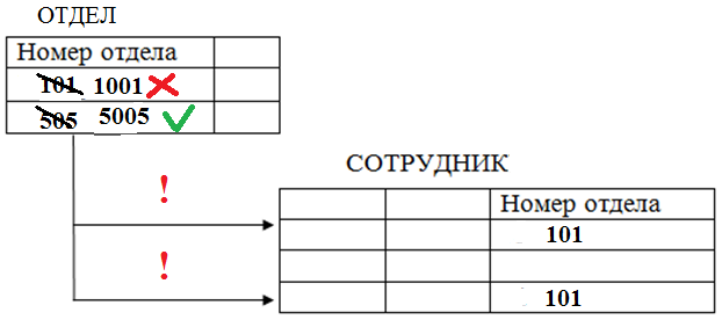


Рис.12 Реализация правила ОГРАНИЧИТЬ изменение

Второй вариант - ***КАСКАДИРОВАТЬ (CASCADE).*** Операция изменения первичного ключа «каскадируется» (распространяется) на ссылающееся отношение (в примере на отношение Сотрудник). Операция выполняется в 2 шага. На первом шаге изменяется значение внешнего ключа у сотрудников изменяемого отдела (контроль целостности временно отключается), а на втором – вносятся изменения в отдел (рис. 14).

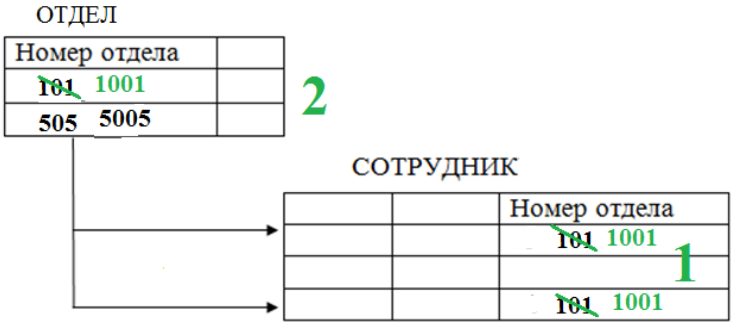


Рис.13 Реализация правила КАСКАДИРОВАТЬ изменения

Третий вариант - **СБРАСЫВАТЬ ССЫЛКУ (SET NULL).** Для всех сотрудников изменяемого отдела на первом шаге внешний ключ устанавливается в неопределенное значение, а на втором – вносятся изменения в кортеж отдела (рис. 15). Исключением (нельзя применять данное правило) является ситуация, когда FK не должен содержать неопределенные значения null.

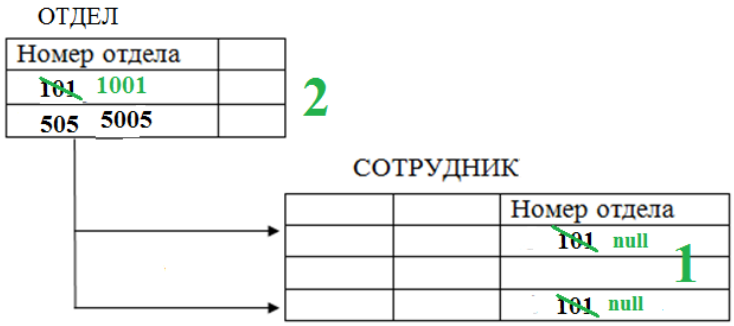


Рис.14 Реализация правила СБРАСЫВАТЬ ССЫЛКУ при изменениях

Сформулируем правило ссылочной целостности.

Значением внешнего ключа может быть или NULL-значение или значение, совпадающее со значением первичного ключа некоторого кортежа в целевом отношении

Для внешнего ключа также формулируется набор дополнительных требований (спецификаций):

1) допустимость NULL-значения: NOT NULL | NULL

2) действия при удалении кортежа из целевого отношения:

RESTRICT | CASCADE | SET NULL

3) действия при обновлении первичного ключа в целевом отношении:

RESTRICT | CASCADE | SET NULL

Целостность, определяемая пользователем.

Любая БД кроме базовых правил имеет набор специфических правил целостности, определяемых предметной областью.

Примеры таких правил в схеме отношений Отдел – Сотрудник:

а) Атрибут ***Табельный номер*** должен задаваться в виде 999,
где 9 – одна из цифр 0 - 9

б) Атрибут ***Номер отдела*** должен задаваться в виде 99,
где 9 – одна из цифр 0 – 9

в) Атрибут ***Оклад*** не может быть отрицательным

Задание

1. В модели данных Отдел-Сотрудник (рис.6) придумайте операции, которые могут привести к неправильным ссылкам.

2. Сформулируйте правила целостности (отношения, по ссылкам, пользовательские) для схем данных:

Учащийся - Класс

Учащийся - Предмет - Экзамен

Порядок выполнения работы

1. Изучить теоретический материал
2. Выполнить задание

Содержание отчета

1. Основные определения и термины
2. Результаты выполнения задания

Практическое занятие № 30
Операции над реляционными данными.
Манипулирование реляционными данными

Цель занятия. Познакомиться с операциями над реляционными данными, с особенностями выполнения операций реляционной алгебры. Научиться вычислять реляционные алгебраические выражения

Краткие теоретические сведения

Имеется два базовых механизма манипулирования реляционными данными:

- **реляционная алгебра (РА)**, основанная на теории множеств
- **реляционное исчисление (РИ)**, базирующееся на математической логике.

Остановимся подробно на аппарате реляционной алгебры.

Исходя из идеи автора реляционной теории Э.Кодда, что отношения являются множествами, логично использовать для операции над реляционными данными теоретико-множественные операции. Учитывая специфику реляционных отношений, Э.Кодд дополнил аппарат реляционной алгебры специальными операциями.

В варианте РА, который был предложен Э.Коддом, набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса: теоретико-множественные (взятые из теории множеств) и специальные реляционные операции.

В состав **теоретико-множественных операций** входят операции:

- объединение;
- пересечение;
- вычитание;
- прямое произведение.

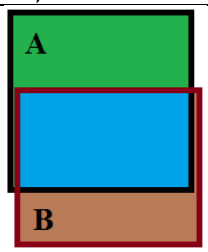
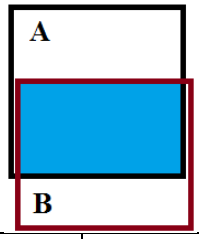
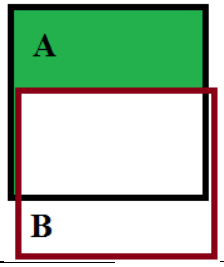
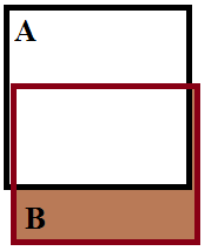
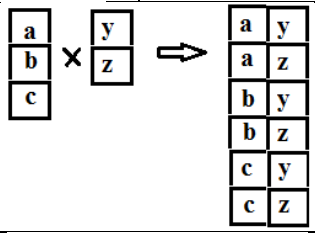
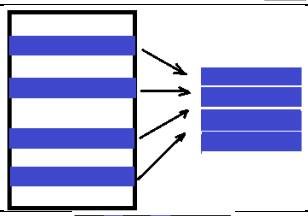
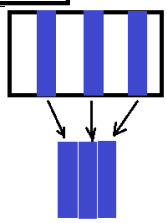
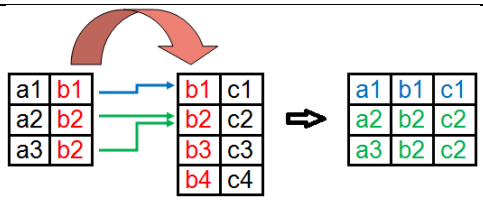
Специальные реляционные операции включают:

- выборка;
- проекция;
- соединение;
- деление.

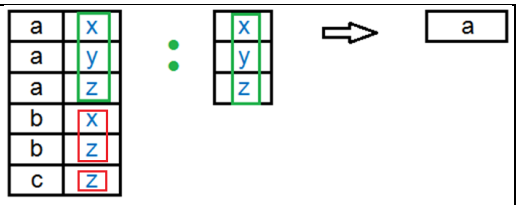
Кроме того, в состав алгебры включается *операция присваивания*, позволяющая сохранить в базе данных результаты вычисления алгебраических выражений, и *операция переименования атрибутов*, дающая возможность корректно сформировать заголовок результирующего отношения.

Все операции предложенного выше набора обладают очевидной и простой интерпретацией (таблица 1).

Таблица 1 – Общая интерпретация операций РА

Объединением двух отношений является отношение, включающее все кортежи, которые принадлежат или одному из двух исходных отношений (области зеленого и коричневого цвета) или обоим (область бирюзового цвета)	
Пересечением двух отношений является отношение, включающее все кортежи, которые принадлежат одновременно двум исходным отношениям (область бирюзового цвета)	
Вычитание (или **разность**) двух отношений возвращает отношение, включающее все кортежи, которые принадлежат первому отношению и не принадлежат второму	**A-B**  **B-A** 
Прямое произведение двух отношений есть отношение, содержащее кортежи, которые являются конкатенацией (сцеплением) кортежей исходных отношений	
Выборка возвращает отношение, содержащее кортежи исходного отношения, удовлетворяющее определенному условию (горизонтальный срез)	
Проекция возвращает отношение, содержащее все кортежи после исключения из него некоторых атрибутов (вертикальный срез)	
Соединением двух отношений являеется отношение, кортежи которого – это сочетание двух кортежей исходных отношений, имеющих общее значение по общим атрибутам	

Продолжение таблицы 1

Делением двух отношений - бинарного и унарного, является отношение, содержащее все значения одного атрибута бинарного отношения, которые соответствуют (по другому атрибуту) всем значениям в унарном отношении	
Операция **переименования** производит отношение, тело которого совпадает с телом исходного отношения, но имена атрибутов изменены	**A (X,rename Y as Y1, Z as Z1)** **A (X,Y1,Z1)** Например, Сотрудник(Таб_номер as Тн, ФамилияИмяОтчество as ФИО, …)
Операция **присваивания** позволяет сохранить результат вычисления реляционного выражения в существующем или новом отношении	

Операции реляционной алгебры обладают свойством замкнутости, которое формулируется следующим образом:

Результат любой операции над отношениями также является отношением.

Следствием из этого свойства является то, что результат одной операции может использоваться в качестве исходных данных для другой операции.

Хотя в основе теоретико-множественных операций лежит классическая теория множеств, соответствующие операции реляционной алгебры обладают некоторыми особенностями. В теории множеств операция объединения имеет смысл для любых двух множеств-операндов.

Например, объединение множеств

Учащиеся (ФИО, класс)

и

Учителя (ФИО, должность)

дает новое множество Школа – набор элементов с различными заголовками.

В реляционной алгебре результатом операции должно являться отношение. Операция объединения двух произвольных отношений (с разными заголовками, как в примере Учащиеся и Учителя), даст в качестве результата множество разнотипных кортежей, т.е. это не будет отношением. Если исходить из требования замкнутости реляционной алгебры, то такая операция объединения является недопустимой. То же относится к операциям пересечения и вычитания.

Поэтому для теоретико-множественных операций (кроме прямого произведения) должна соблюдаться ***совместимость отношений-операндов по типу*:**

Два отношения совместимы по типу, если у них одинаковые заголовки, т.е. в заголовках обоих отношений содержится один и тот же набор имен атрибутов и одноименные атрибуты определены на одном и том же домене.

Дадим теперь точное определение реляционных операций и рассмотрим их выполнение на примере двух отношений.

A - (Поставщики сырья)

Код	Наименование	Город
01	АО Свобода	Москва
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново

B - (Поставщики материалов)

Код	Наименование	Город
01	АО Свобода	Москва
05	ООО Эмульсия	СПб
06	АО Синтез	Владимир

Операция Пересечение

Пересечение двух отношений **A INTERSECT B** дает в качестве результата отношение с тем же заголовком, что A и B, и телом, включающим кортежи, которые принадлежат и A и B

Результат для примера

Код	Наименование	Город
01	АО Свобода	Москва

Операция Объединение

Объединение двух отношений **A UNION B** дает в качестве результата отношение с тем же заголовком, что A и B, и телом, включающим кортежи, которые принадлежат A или B или обоим. В результате кортежи-дубликаты удаляются.

Результат для примера

Код	Наименование	Город
01	АО Свобода	Москва
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново
01	АО Свобода	Москва
05	ООО Эмульсия	СПб
06	АО Синтез	Владимир

Операция Вычитание

Вычитание двух отношений **A MINUS B** дает в качестве результата отношение с тем же заголовком, что A и B, и включающим кортежи, которые принадлежат A и не принадлежат B.

Результат A Minus B для примера

Код	Наименование	Город
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново

Операция Прямое произведение

Прямое произведение двух отношений A **TIMES B** дает в качестве результата новое отношение, кортежи которого образуются конкатенацией (или слиянием) каждого кортежа первого отношения с каждым кортежем второго отношения.

Рассмотрим выполнение прямого произведения на примере двух отношений.

A - (Поставщики)

Код поставщика	Наименование поставщика	Город
01	АО Свобода	Москва
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново

B - (Поставки)

Номер документа	Дата поставки	Код поставщика
101	02.09.20	01
105	02.09.20	02
120	07.09.20	01

Вначале необходимо переименовать один из одноименных атрибутов в любом отношении (для получения правильного заголовка в результате):

B (НомерДокумента, ДатаПоставки, Rename КодПоставщика as КодПост)

Получим

A - (Поставщики)

Код поставщика	Наименование поставщика	Город
01	АО Свобода	Москва
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново

B - (Поставки)

Номер документа	Дата поставки	Код Пост
101	02.09.20	01
105	02.09.20	02
120	07.09.20	01

И теперь выполним операцию прямого произведения A Times B.

Результат для примера

A - (Поставщики) **B** - (Поставки)

Код поставщика	Наименование поставщика	Город	Номер документа	Дата поставки	Код Пост
01	АО Свобода	Москва	101	02.09.20	01
01	АО Свобода	Москва	105	02.09.20	02
01	АО Свобода	Москва	120	07.09.20	01
02	ПАО Синтетика	Москва	101	02.09.20	01
02	ПАО Синтетика	Москва	105	02.09.20	02
02	ПАО Синтетика	Москва	120	07.09.20	01
03	ООО Волокно	Иваново	101	02.09.20	01
03	ООО Волокно	Иваново	105	02.09.20	02
03	ООО Волокно	Иваново	120	07.09.20	01

Специальные реляционные операции будем рассматривать на примере следующего отношения

A - Поставщики

Код поставщика	Наименование поставщика	Город
01	АО Свобода	Москва
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново
04	ПАО Лен	Владимир
05	ООО Эмульсия	СПб
06	АО Синтез	Владимир

Операция Выборка (ограничение)

Операция выборки, требует наличия:

- исходного отношения A
- простого условия выборки, задаваемым оператором Θ (тета).

Здесь Θ – любой скалярный оператор сравнения:

= , > , >= , <, <=, <>

Условие выборки может иметь следующий вид:

1) Θ-выборка отношения A по атрибутам X и Y

A Where X Θ Y

Результатом будет отношение, имеющее тот же заголовок, что и отношение A, и тело, содержащее множество кортежей отношения A, для которых проверка условия X Θ Y дает значение «истина». X и Y должны быть определены на одном и том же домене

2) Если один из атрибутов заменить на литерал (константу), получим

A Where X Θ Const

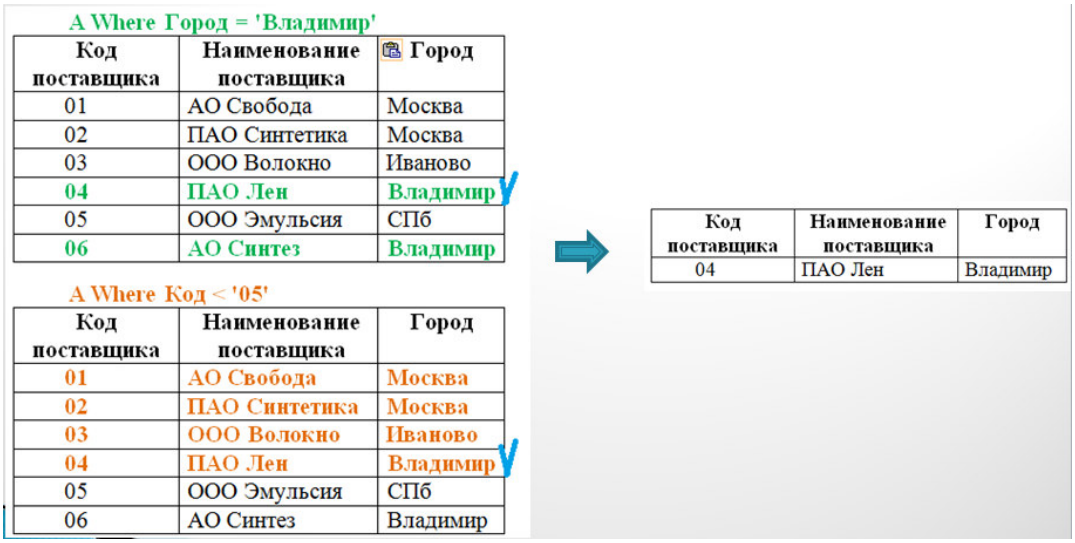
Пример выборки. Выбрать предприятия из города Владимир

A Where Город = 'Владимир'

В примере было использовано одно простое условие сравнения. Могут встретиться ситуации, когда имеется не одно, а несколько простых условий, которые должны выполняться все (этому соответствует логическая операция И) или хотя бы одно (этому соответствует логическая операция ИЛИ). Известны тождества, которые позволяют привести проверку составных условий к проверке простых условий и одной из теоретико-множественных операций.

Пример составного условия. Выбрать предприятия из города Владимир с кодом меньше '05'

A Where Город = 'Владимир' AND Код<'05'



Операция Проекция

Операция взятия проекции требует исходного отношения A и списка имен атрибутов, входящих в заголовок отношения A, которые необходимо выбрать.

$$A[x_1,x_2,\ldots,x_k]$$

Результатом проекции отношения A по атрибутам x_1, x_2, ..., x_k является отношение, включающее все кортежи исходного отношения, в которых оставлены только атрибуты x_1, x_2, ..., x_k.

Тем самым, при выполнении операции проекции выделяется "вертикальная" вырезка отношения-операнда с последующим удалением кортежей-дубликатов.

Пример Выбрать все города поставщиков

A [Город]

Операция соединение отношений

Операция соединения имеет несколько разновидностей. Наиболее важной является естественное соединение.

Пусть отношения A и B имеют заголовки соответственно (X,Y) (Y,Z). Здесь Y – общие атрибуты для A и B.

Естественным соединением A и B **A JOIN B** [A.y=B.y] называется отношение с заголовком (X,Y,Z) и телом, содержащим множество кортежей, для которых в отношении A значения атрибутов X=x, Y=y и в отношении B значения атрибутов Y=y, Z=z.

Как правило, в одном отношении общий атрибут играет роль *первичного ключа*, а в другом – роль *внешнего ключ*а. Основной смысл (и цель) операции естественного соединения - *восстановление сложного отношения*, разбитого при нормализации на более простые

Если A и B не имеют общих атрибутов, то естественное соединение превращается в прямое произведение.

Рассмотрим операцию естественного соединения на следующем примере.

A (Поставщики)

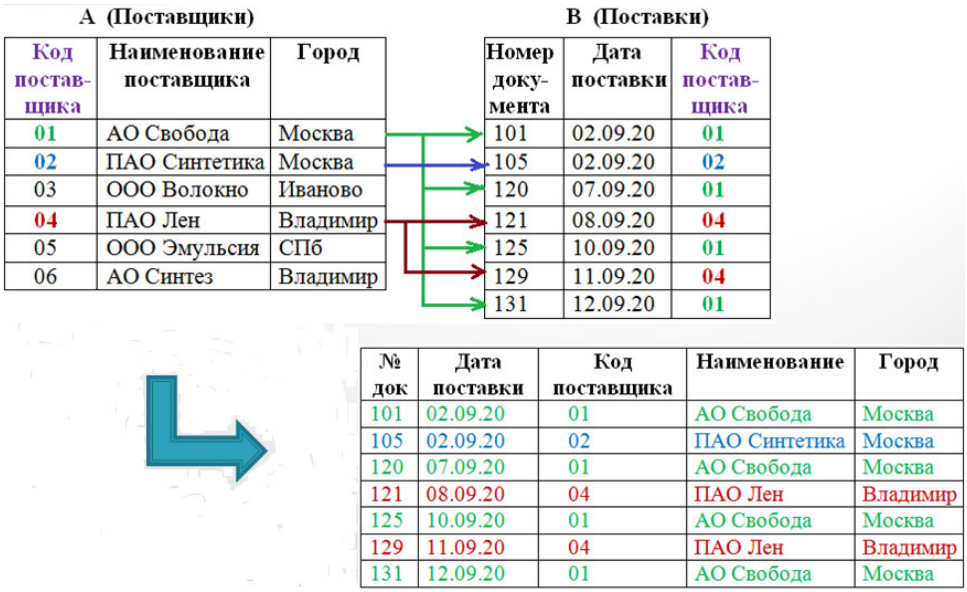
Код поставщика	Наименование поставщика	Город
01	АО Свобода	Москва
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново
04	ПАО Лен	Владимир
05	ООО Эмульсия	СПб
06	АО Синтез	Владимир

B (Поставки)

Номер документа	Дата поставки	Код поставщика
101	02.09.20	01
105	02.09.20	02
120	07.09.20	01
121	08.09.20	04
125	10.09.20	01
129	11.09.20	04
131	12.09.20	01

Отношения A и B имеют общий атрибут – Код поставщика. Причем в отношении Поставщики – это PK, а в отношении Поставки – это FK. Операция естественного соединения:

Поставщики JOIN Поставки [Поставщики.Код пост=Поставки.Код пост]



A (Поставщики)

Код поставщика	Наименование поставщика	Город
01	АО Свобода	Москва
02	ПАО Синтетика	Москва
03	ООО Волокно	Иваново
04	ПАО Лен	Владимир
05	ООО Эмульсия	СПб
06	АО Синтез	Владимир

B (Поставки)

Номер документа	Дата поставки	Код поставщика
101	02.09.20	01
105	02.09.20	02
120	07.09.20	01
121	08.09.20	04
125	10.09.20	01
129	11.09.20	04
131	12.09.20	01

№ док	Дата поставки	Код поставщика	Наименование	Город
101	02.09.20	01	АО Свобода	Москва
105	02.09.20	02	ПАО Синтетика	Москва
120	07.09.20	01	АО Свобода	Москва
121	08.09.20	04	ПАО Лен	Владимир
125	10.09.20	01	АО Свобода	Москва
129	11.09.20	04	ПАО Лен	Владимир
131	12.09.20	01	АО Свобода	Москва

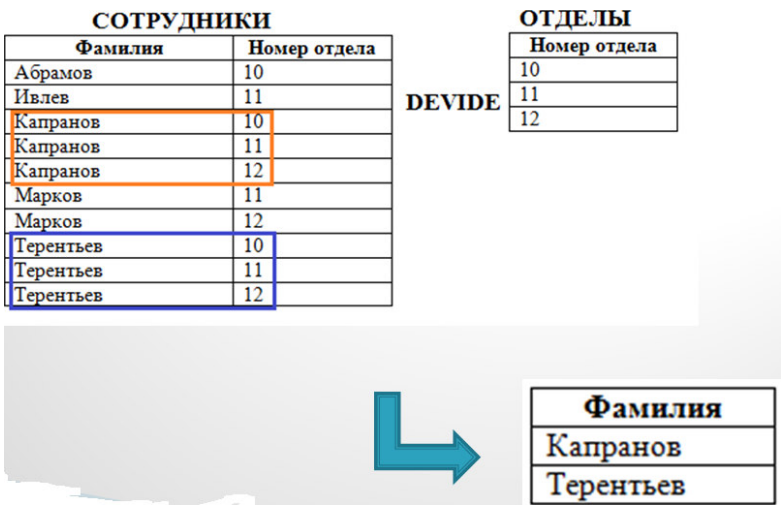
Операция Деление

Эта операция наименее очевидна из всех операций реляционной алгебры и поэтому нуждается в более подробном объяснении.

Деление отношения A(X,У) на отношение B (У) A **DEVIDE** B дает в результате отношение с заголовком X и телом, содержащим множество кортежей X=x таких, что существует кортеж X=x, принадлежащий отношению A для всех кортежей У=y, принадлежащих отношению B. Другими словами результат и телом, содержащим такие значения X из отношения A, для которых соответствующие значения У отношения A включают все значения У отношения B.

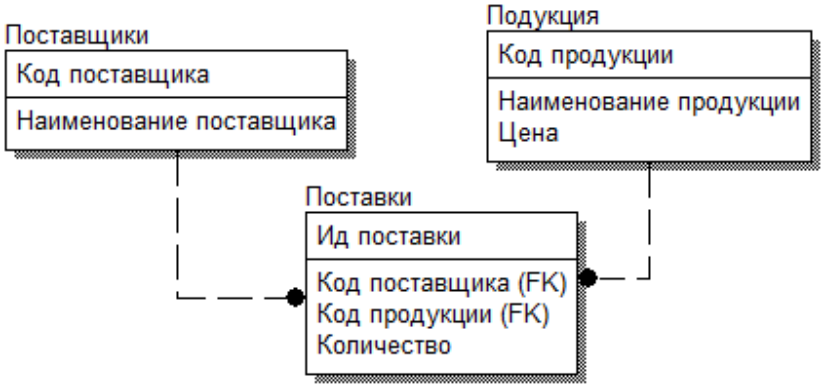
Пример операции деления.

Предположим, что в базе данных сотрудников имеются два отношения: СОТРУДНИКИ(ИМЯ, ОТД_НОМ) и ОТДЕЛЫ(ОТД_НОМ). операции реляционного деления отношения СОТРУДНИКИ на отношение ОТДЕЛЫ будет получено унарное отношение, содержащее сотрудников, которые работают по совместительству во всех отделах данной организации.



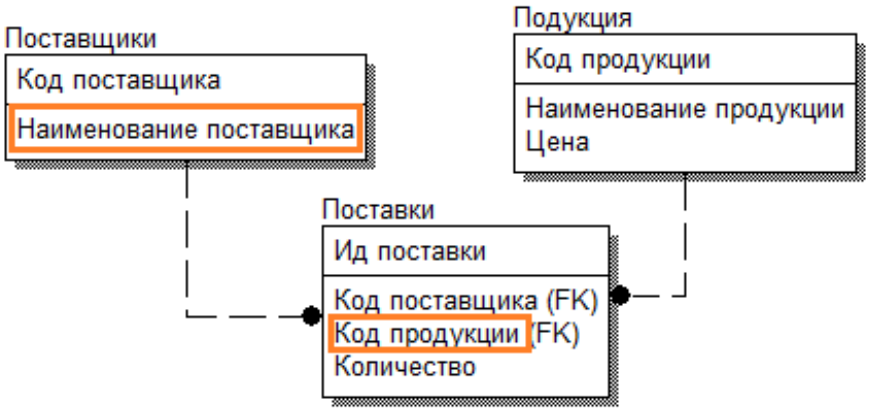
Рассмотрим примеры выполнения запросов.

Дана модель данных *Поставки продукции*, включающая 3 отношения: Поставщики, Продукция, Поставки.

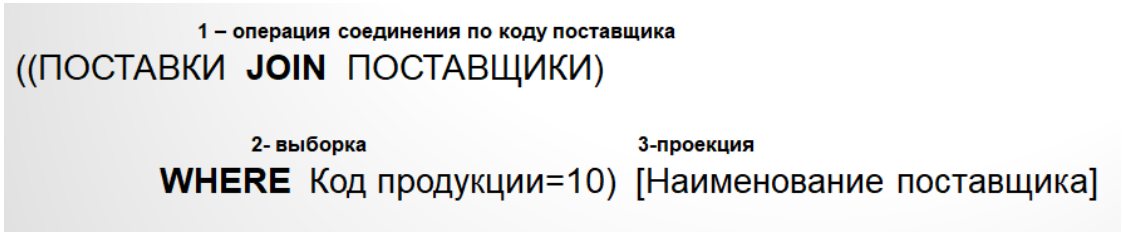


Задача 1. Вывести наименования поставщиков, которые поставили продукцию с кодом 10

В задаче требуется вывести наименования поставщиков. Поэтому нам потребуется отношение Поставщики. Условие задано на поставки продукции с кодом 10, поэтому потребуется также отношение Поставки. Чтобы выяснить, какую продукцию поставлял Поставщик, надо выполнить естественное соединение этих отношений.



Чтобы решить задачу необходимо вычислить следующее выражение



Оно включает 3 операции, которые выполняются в следующем порядке:

1 – операция соединения отношений Поставщик и Поставки по коду поставщика

2 - операция выборки кортежей соединенных отношений по коду продукции

3 – операция проекции, оставляющая только атрибут Наименование поставщика

Задание

1. Выполните операцию выборки для отношения Поставщик:

- Выбрать предприятия из города Владимир либо с кодом > '04'
- Выбрать предприятия НЕ из города Владимир

2. Составьте выражения для выполнения следующих запросов:

а) Вывести наименования поставщиков, которые не поставили продукцию с кодом «10»

б) Вывести наименования поставщиков и объемы поставок, в которых цена поставки больше 100

Порядок выполнения работы

1. Изучить теоретический материал
2. Выполнить задание

Содержание отчета

1. Определение основных операций реляционной алгебры
2. Результаты выполнения задания

Практическое занятие № 31
Проектирование реляционных баз данных.
Логическое проектирование

Цель занятия. Познакомиться с этапами проектирования баз данных, программными средствами проектирования. Научиться разрабатывать логическую модель базы данных.

Краткие теоретические сведения

1. Основные понятия модели данных

Проектирование базы данных начинают с анализа предметной области и выявления требований к ней будущих пользователей. Объединяя частные представления о содержимом базы данных отдельных пользователей, вначале создается обобщенное описание создаваемой базы данных, называемое концептуальной схемой. Данный этап еще называют моделированием или логическим проектированием базы данных.

Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД. Поэтому данное представление будет понятно даже для неспециалистов (заказчиков информационной системы).

Цель моделирования данных состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы "сущность-связь" (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Первый шаг моделирования - извлечение информации на основе анализа предметной области и выделение сущностей.

Рассмотрим в качестве примера проектирование БД, содержащей информацию о сотрудниках предприятия. Вначале введем основные понятия.

Вначале введем понятие сущности.

Сущность (Entity) - реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информацию о котором необходимо сохранить в БД.

Каждая сущность должна обладать следующими свойствами:

- иметь уникальное имя (задается именем существительным в именительном падеже в единственном числе), например «Сотрудник»;
- содержать один или несколько атрибутов, описывающими наиболее важные свойства объекта предметной области (например: «фамилия», «должность»);

- обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности (выделяют среди других экземпляров данной сущности) – первичным ключом, например: атрибут «табельный номер сотрудника»;
- иметь связи с другими сущностями модели.

Перейдем к рассматриваемой БД. В данной системе важны отделы и работающие в них сотрудники. Поэтому можно выделить 2 сущности: «Отдел», «Сотрудник».

Следующим важным понятием являются связи между сущностями.

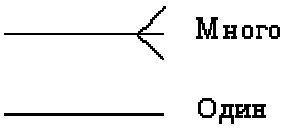
Связь (Relationship) - поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. В этой связи-ассоциации каждый экземпляр одной сущности, называемой родительской, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности родителя.

Связи можно дать имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Имя связи формируется как с позиции сущности-родителя, так и с позиции сущности-потомка.

Например, связь отела с сотрудником может быть выражена следующим образом:

- отдел *(имя родителя)* может содержать *(имя связи)* одного или более *(степень связи)* сотрудников *(имя потомка)*;
- сотрудник должен работать только в одном отделе.

Степень связи (количество экземпляров сущности, которым она соответствует) графически изображаются одной или многими линиями на концах связи.



Связь сущностей Отдел и Сотрудник графически будет выражены следующим образом (рис. 1).

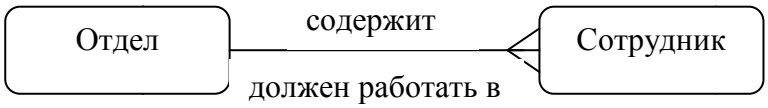


Рис. 1. Связь сущностей

Наконец еще одним важным понятием является понятие атрибута.

Атрибут - любая характеристика сущности, значимая для рассматриваемой предметной. Например, в сущности «Отдел» атрибутами могут быть «Код отдела», «Наименование отдела».

Атрибут может быть либо описательным (неключевым), либо входить в состав уникального идентификатора (первичного ключа). Атрибут может быть либо обязательным к заполнению, либо необязательным. Обязательность означает, что атрибут не может принимать неопределенных значений (null).

Уникальный идентификатор (ключ) - это атрибут или совокупность атрибутов, предназначенная для уникальной идентификации каждого экземпляра сущности. Например, в сущности «Отдел» таким атрибутом может быть «Код отдела».

Каждый атрибут обозначается уникальным именем, выражаемым грамматическим оборотом существительного, описывающим представляемую атрибутом характеристику (например, «Фамилия»). Атрибуты изображаются в виде списка имен внутри блока сущности, причем каждый атрибут занимает отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком "#".

Каждая сущность должна обладать хотя бы одним возможным ключом. Возможный ключ сущности - это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные - как альтернативные ключи.

Сущности с указанием атрибутов приведены на рис. 2.

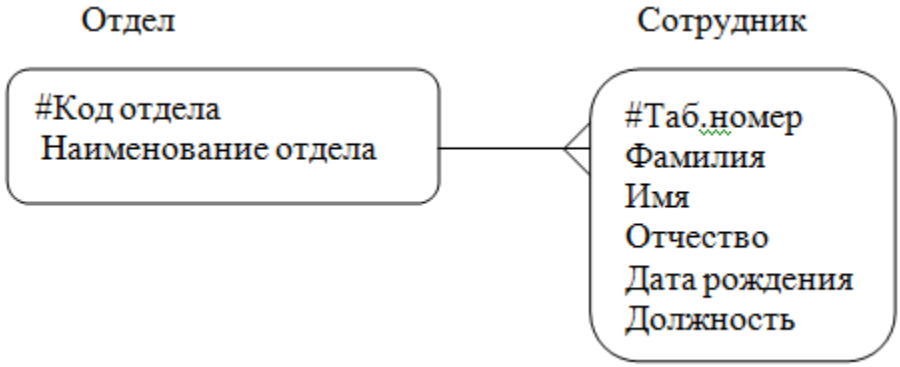


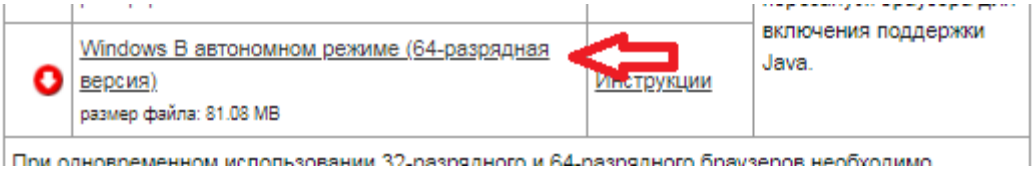
Рис.2 Атрибуты сущностей

2. Программные средства проектирования БД

Для автоматизации процесса проектирования используются CASE (Computer-Aided Software/System Engineering) технологии и инструментальные средства, позволяющие максимально упростить все этапы разработки базы данных. Удобно использовать свободно распространяемые программные средства. К таким средствам относится SQL Power Architect

Community Edition. Среда SQL Power Architect предназначена для визуального проектирования модели данных для распространенных СУБД и организации взаимодействия с серверами БД. Данная программная среда выбрана еще потому, что она имеет драйверы подключения к СУБД PostgreSQL. И непосредственно из среды можно по модели создать схему БД.

Программа для использования требует установки платформы Java, точнее JRE (Java Runtime Environment) версии 8 или выше. Скачать пакет Java можно по ссылке https://www.java.com/ru/download/manual.jsp, выбрав нужную операционную систему.



Затем необходимо запустить скачанный пакет на установку.

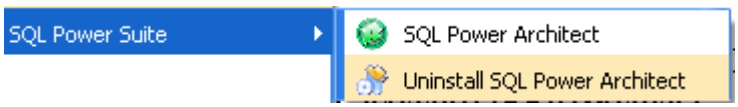
Для установки программы SQL Power Architect необходимо скачать установочный пакет (SQL-Power-Architect-Setup-Windows-1.0.9.exe) по адресу:

http://www.bestofbi.com/page/architect_download_os

Чтобы затем не скачивать драйверы для соединения с распространенными серверами БД, лучше скачать установочный пакет, содержащий эти драйверы, (SQL-Power-Architect-Setup-Windows-jdbc-1.0.8.exe) по адресу:

http://www.bestofbi.com/page/deprecated-downloads#architect

После скачивания необходимо запустить пакет на установку. После установки в меню *Пуск* операционной системы Windows появится новая папка.



Для установки русскоязычного интерфейса программы необходимо изменить параметр языка, выбрав пункт меню:

`File - User Preferences`

и изменить параметр `Default Language` на `Russian`.

3. Разработка модели данных в среде SQL Power Architect

В некоторых программных средствах четко разделены этапы логического и физического проектирования БД.

Логическое проектирование БД – это создание *непротиворечивого и эффективного* отображения реальных объектов предметной области *в абстрактные объекты* модели данных. Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Атрибуты имеют

универсальные обобщенные типы данных: символьный, числовой и временной. Логическая модель данных является универсальной и не связана с конкретной реализацией СУБД.

Физическое проектирование БД - это представление абстрактных объектов логической модели *в структуры данных конкретной СУБД* для обеспечения эффективности выполнения запросов. Сущности называются таблицами, а атрибуты - колонками. Колонкам назначается более конкретный тип данных, из возможных типов выбранной СУБД.

В программной среде SQL Power Architect такое разделение осуществляется только на уровне имен:

Logical Names (на этом уровне будем использовать имена в кириллице);
Physical Names (будем использовать имена на английском языке).

После открытия программы появится основное окно (рис.3). Для разработки модели необходимо создать новый проект.

Для сохранения проекта под выбранным именем используется пункт меню:

`Файл – Сохранить Проект как`

Можно настроить свойства проекта, вызвав пункт меню:

`Файл – Параметры проекта`

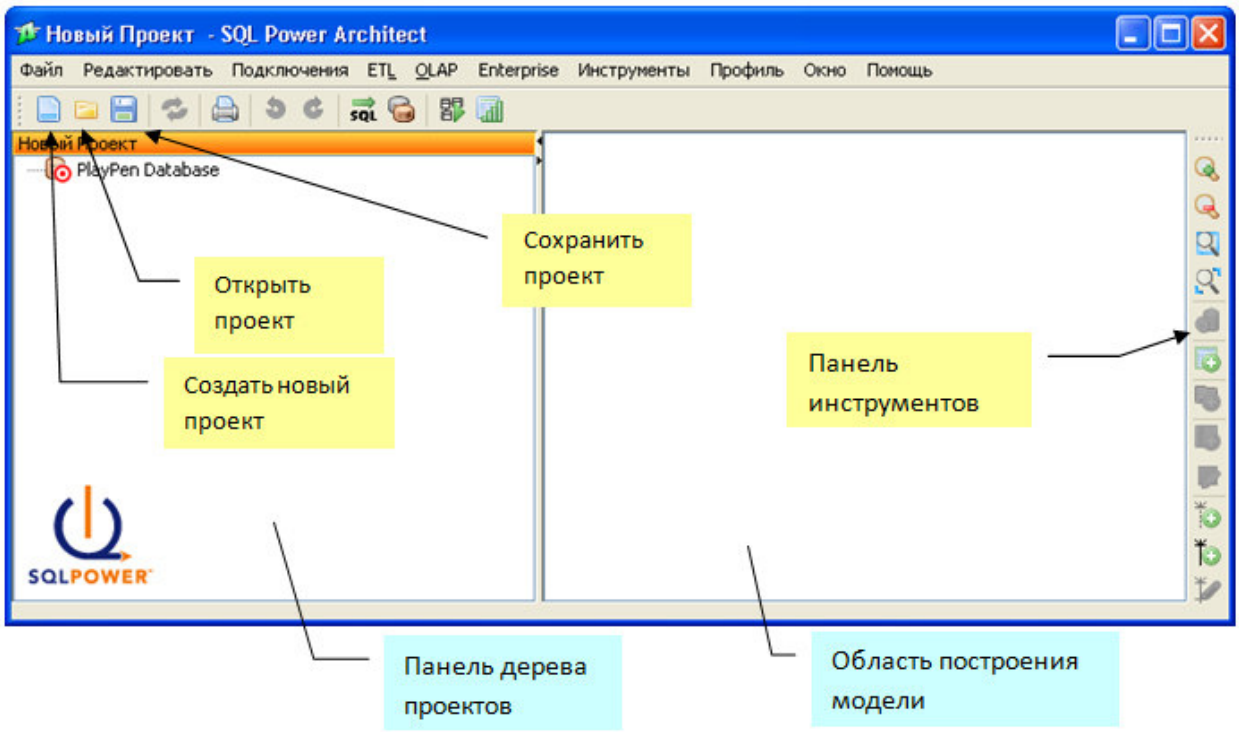


Рис.3 Основное окно программы

В окне настройки свойств проекта (рис.4) можно задать:

1) Вид отображаемой модели (переключатель `Display Tables and Columns`):

   `Logical Names` - логической

`Physical Names` - физической

2) Отображение колонок (переключатели `Show …`)

3) Отображение спецификаций ключей (флажки). Например флаг *Показать теги PK* позволяет отобразить/скрыть пометку первичного ключа.

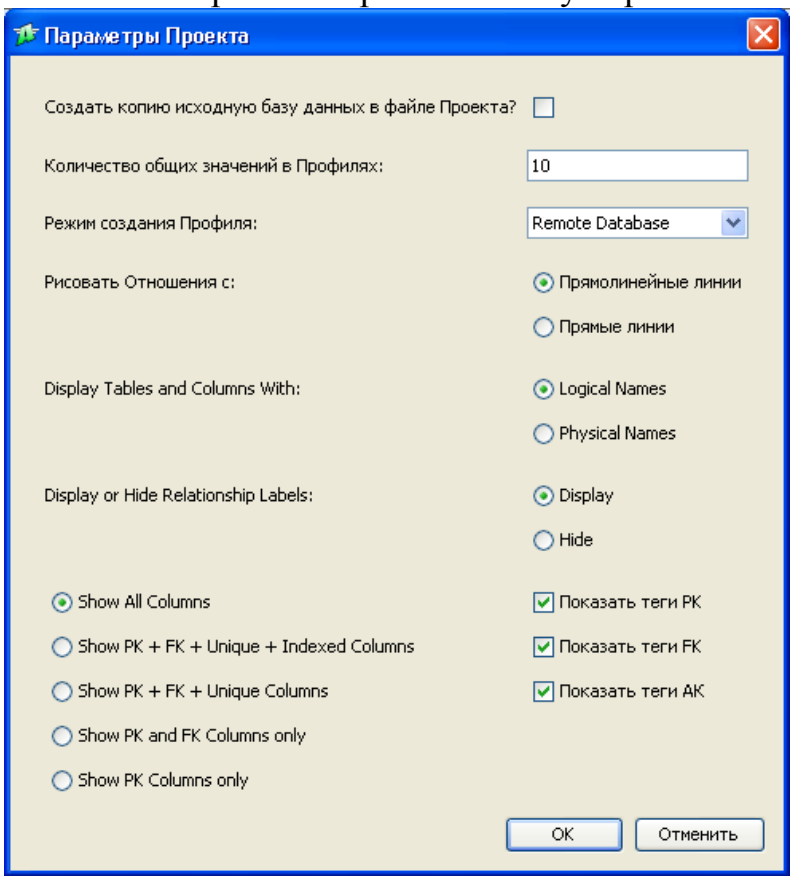


Рис. 4 Окно настройки свойств проекта

Для создания модели данных используются кнопки панели инструментов:

 добавить сущность (таблицу)

 изменить свойства таблицы

 добавить новый индекс

 добавить атрибут (колонку)

 добавить неидентифицирующую связь

 добавить идентифицирующую связь

 изменить свойства связи

В качестве примера рассмотрим предметную область, связанную с **продажами**. Описание предметной области можно сформулировать следующим образом. Предприятие производит готовую продукцию. Продукцию приобретают покупатели. На продаваемую продукцию оформляется накладная. В накладной может быть несколько видов продукции. Продажи фиксируются в книге продаж.

Разработка модели данных будет включать несколько шагов.

Шаг 1. Создание проекта

Создадим новый проект и сохраним его под именем Продажи.

Шаг 2. Создание сущностей (отношений)

В соответствии с описанием предметной области можно выделить следующие сущности: Продукция, Покупатель, Договор (описывает условия поставок продукции), Книга продаж (описывает партию продукции, отпущенную покупателю).

Для добавления новой сущности необходимо выбрать мышью иконку «Добавить новую таблицу» и следом щелкнуть в любом месте рабочей области построения модели. Откроется окно «Свойства таблицы» (рис. 5).

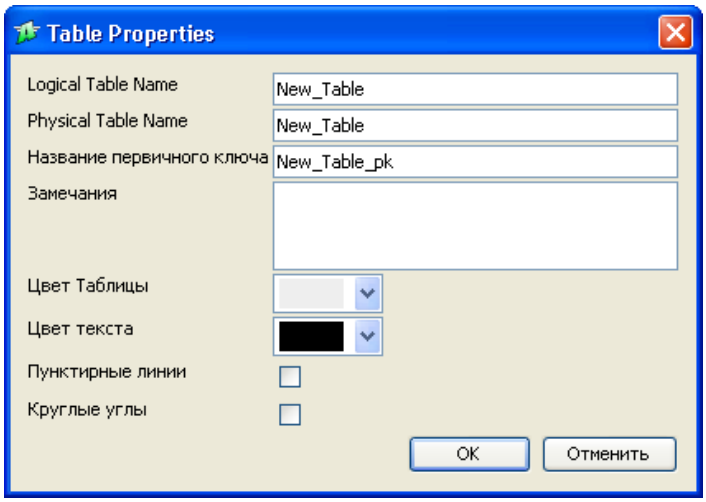


Рис.5 Окно создания новой таблицы

В окне необходимо задать:

- имя сущности логической модели (`Logical Table Name`)
- имя таблицы физической модели (`Physical Table Name`)
- имя ограничения первичного ключа
- комментарий (замечание) к названию таблицы (будет сохранен в виде комментария в БД)

Дополнительно можно задать элементы оформления таблицы.

На рис. 6 показано заполненное окно свойств для сущности *Продукция*. После нажатия кнопки «OK» новая таблица появится в дереве проекта (рис.7).

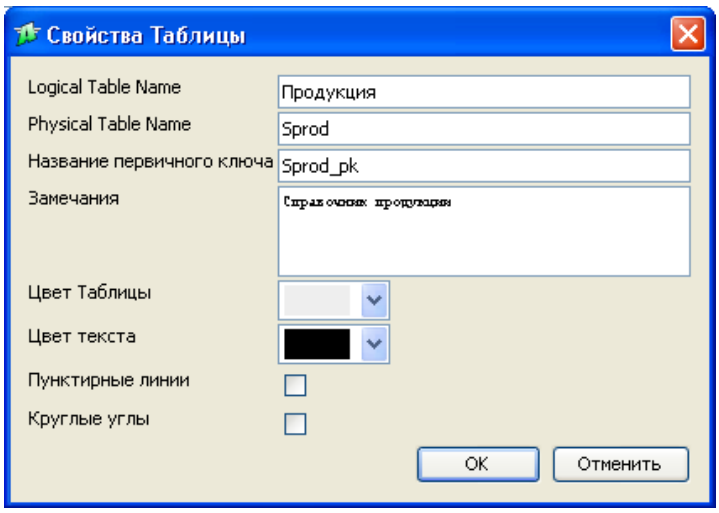


Рис.6 Окно создания новой сущности Продукция

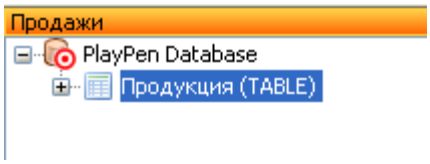


Рис. 7 Новая сущность в дереве проекта

Добавим в сущность Продукция атрибуты: код, наименование, цена, вид. Для добавления атрибутов сущности необходимо выделить требуемую сущность и на панели инструментов щелкнуть по кнопке «Вставить столбец». Откроется окно редактирования свойств столбца (рис. 8).

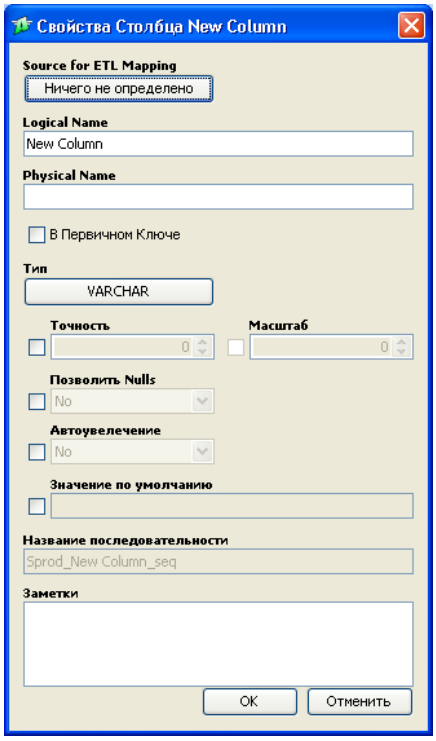


Рис.8 Окно редактирования свойств столбца

В окне необходимо задать (не все элементы обязательны):

- имя атрибута логической модели (Logical Name);
- имя колонки физической модели (Physical Name);
- для атрибута первичного ключа включить флажок «В первичном ключе»;
- тип данных;
- для некоторых типов задать длину поля (Точность) и количество дробных разрядов (Масштаб);
- обязательность заполнения поля (Позволить Null);
- признак автоматического увеличения для целочисленных полей первичного ключа; при таком выборе необходимо указать имя объекта автоувеличения в БД (последовательность – sequence);
- значение по умолчанию (присваивается колонке, если не задано явного значения);
- заметки (необходимы для формирования комментария к колонке в БД).

На рис. 9 показано окно для создания атрибута первичного ключа сущности *Продукция*.

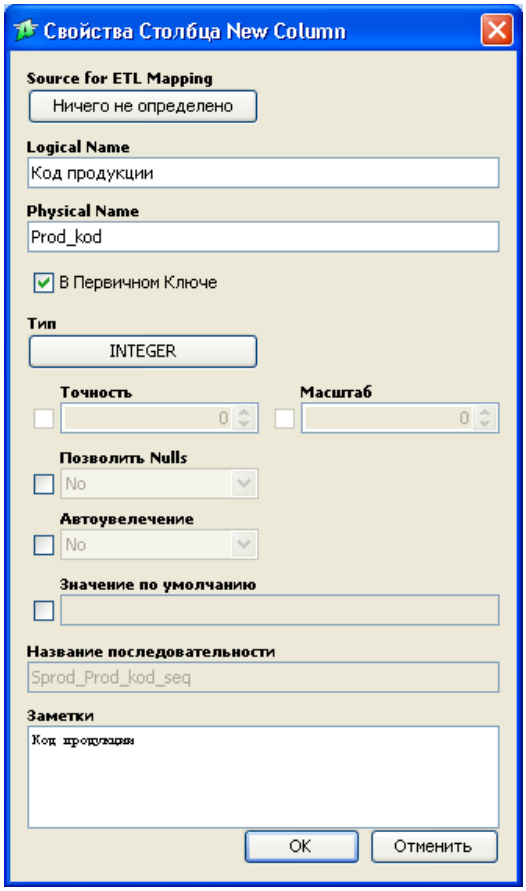


Рис.9 Создание атрибута первичного ключа сущности Продукция

На рис 10 показана созданная сущность *Продукция* в рабочей области модели и в панели с деревом проекта. Для атрибутов-столбцов выбраны следующие типы данных:

Integer - целочисленный тип;

Varchar - символьный тип, число в скобках указывает максимально возможное количество символов;

Numeric - числовой вещественный тип. Числа в скобках означают: 10 - общее количество позиций на все число, 2 - количество разрядов в дробной части числа.

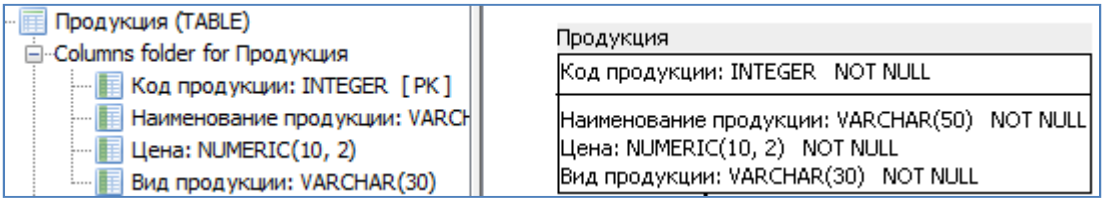


Рис.10 Сущность Продукция с атрибутами

По аналогии создадим сущность Покупатель с атрибутами: Код, Наименование, Город, Телефон (рис. 11).

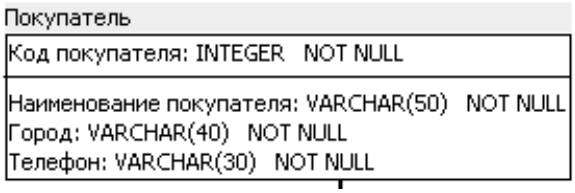


Рис.11 Сущность Покупатель с атрибутами

Шаг 3. Установка связей (отношения) между сущностями

В модели данных сущности связаны так же, как объекты в реальной действительности. Например, сущности *Покупатель* и *Продукция* связаны отношением: покупатель приобретает продукцию, а продукция отпускается покупателю. Данная связь-отношение отражается сущностью Продажа (рис.12).

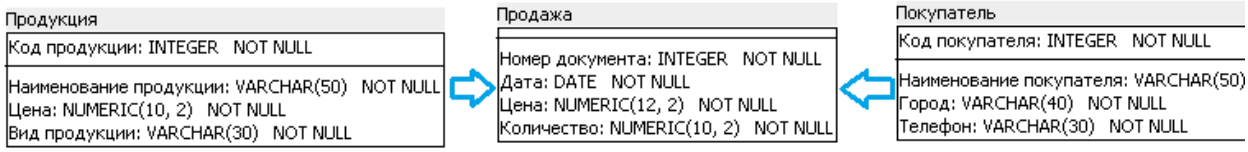


Рис.12 Сущность Продажа

Сущность *Продажа* должна включать атрибуты: *Номер документа, Дата продажи, Объем продажи (количество), Цену продажи*. Покупатель определяется связью с сущностью *Покупатель*, а отпускаемая продукция - связью с сущностью *Продукция*. Поясним почему атрибут *Цена* присутствует в сущности *Продукция* и в сущности *Продажа*. В сущности *Продукция* - это цена на текущий момент, а в сущности *Продажа* - цена на момент продажи.

В связи *Покупатель-Продажа* сущность Покупатель является родительской, а *Книга продаж* - подчиненной. Связь с покупателем в сущности Продажа устанавливается через атрибут, который будет указывать на экземпляр записи в сущности *Покупатель*. В качестве такого атрибута всегда выбирается первичный ключ родительской таблицы, т.к. только он однозначно идентифицирует конкретного покупателя. При установке связи ключевой атрибут будет мигрировать (копироваться) в дочернюю сущность.

Выберем неидентифицирующую связь. При выборе такого типа связи атрибут первичного ключа родительской сущности (в примере *Код покупателя* в сущности *Покупатель*) при миграции его в дочернюю сущность *Продажа* не будет являться частью первичного ключа этой сущности.

Для установления связи необходимо в панели инструментов выбрать кнопку инструмента *Новая неидентифицирующая связь*, а затем щелкнуть вначале по родительской сущности, а потом по подчиненной. На диаграмме между сущностями будет установлена и отображенная связи (рис.13).

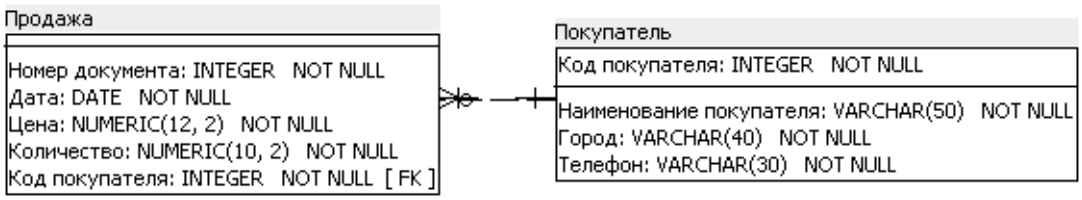


Рис. 13 Связь сущностей Продажа – Покупатель

Свойства связи можно отредактировать, дважды щелкнув по линии связи. Откроется окно (рис. 14). Группа переключателей *Type* задает тип связи (для выделенной связи тип неидентифицирующая). Группа переключателей *Cardinality* задает мощность связи. Мощность в данном случае «один-ко-многим» - одному покупателю может соответствовать много договоров (договора обычно каждый год заключаются заново). Группа переключателей *Update Rule* определяет действия при изменениях первичного ключа в родительской сущности (значение атрибута Код покупателя), когда в подчиненной сущности имеются связанные строки: *Restrict* (*Запретить*), *Cascade* (*Каскадировать*). Выберем *Restrict*. Аналогично для группы переключателей *Delete Rule,* которые определяют действия при удалении строки в родительской сущности при наличии в подчиненной сущности связанных строк. следует выбрать *Restrict*.

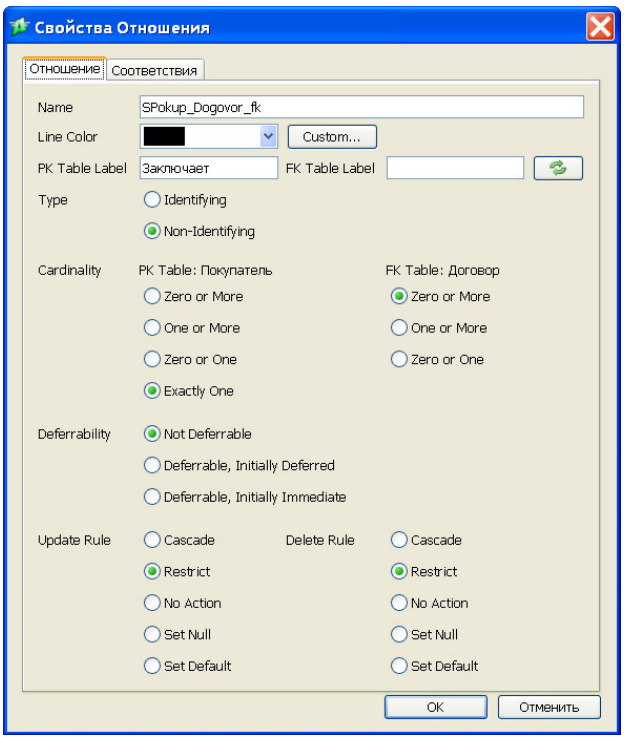


Рис. 14 Установка свойств связи

Также необходимо установить неидентифицирующую связь сущности *Продукция* (родительская) и сущности *Продажа* (подчиненная). Настроить связь надо также как связь *Покупатель-Продажа* (рис.15).

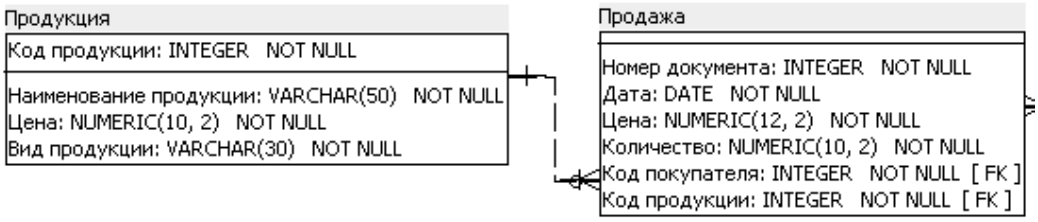


Рис. 15 Связь сущностей Продажа – Продукция

В сущности *Продажа* необходимо выбрать первичный ключ. Из существующих атрибутов в первичный ключ можно включить атрибуты *Номер документа* и *Дата*. Но это будет составной ключ. Желательно использовать простой первичный ключ. В этом случае добавляется суррогатный атрибут первичного ключа (не существующий в реальном объекте предметной области). Такой атрибут обычно называют идентификатором, т.к. его назначение идентифицировать экземпляры сущности. Имя таких атрибутов начинается со слова Идентификатор, к которому добавляется имя сущности. Так в сущности Продажа имя атрибута первичного ключа: *Идентификатор продажи*. Значения таким атрибутам присваиваются автоматически по принципу автоувеличения.

На рис. 16 показана настройка первичного суррогатного ключа в сущности *Продажа*. Обратите внимание на установку флага *Автоувеличение* в значения *Yes*

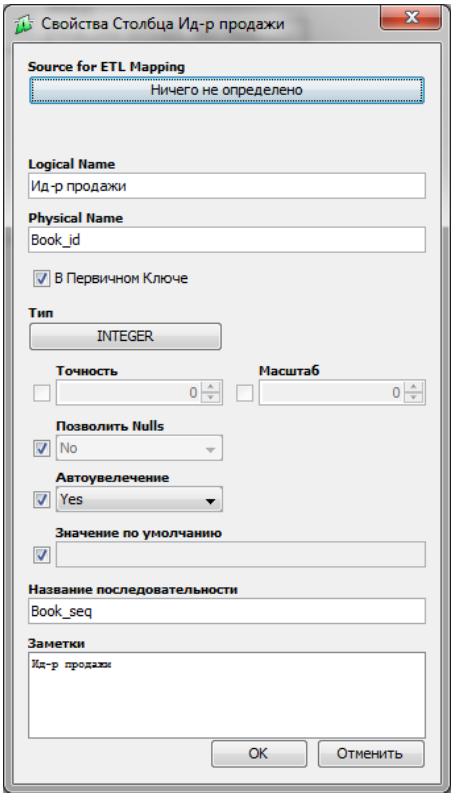


Рис.16 Создание атрибута суррогатного первичного ключа сущности Продажа

Полученная логическая модель представлена на рис. 17.

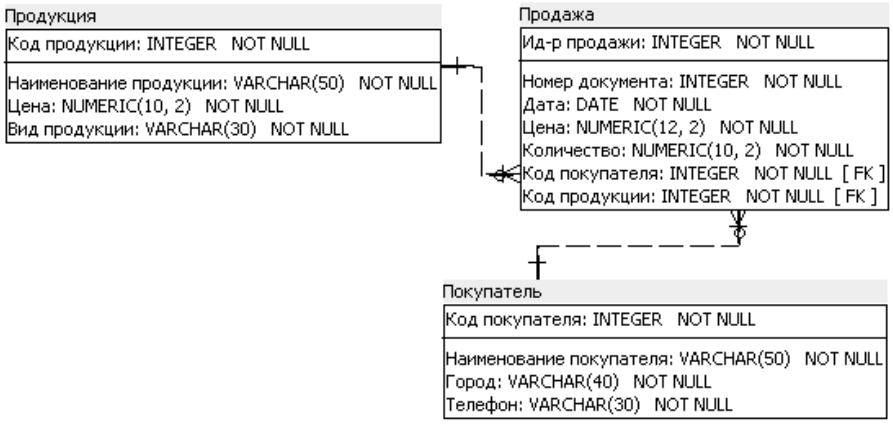


Рис.17 Логическая модель данных Продажи

Здесь следует сделать одно замечание. Партия продажи, как правило, включает не один, а несколько видов продукции. В этом случае в каждой записи одной партии будут повторяться атрибуты, общие для всей партии (*Номер документа, Дата, Код покупателя*). Чтобы исключить дублирование, необходимо создать новую сущность (дадим ей имя *Ассортимент продаж, имя таблицы - Kart*) и перенести в нее все атрибуты, относящиеся к

продукции (*Код продукции, Количество, Цена*), задать атрибут первичного ключа (*Идентификатор ассортимента*) и связать с сущностью *Продажа* (переименуем ее в *Книга продаж)*, как родительской таблицей. Также связь *Продукции* с *Книгой продаж* необходимо перенаправить на сущность *Ассортимент продаж*. Окончательная модель приведена на рис. 18.

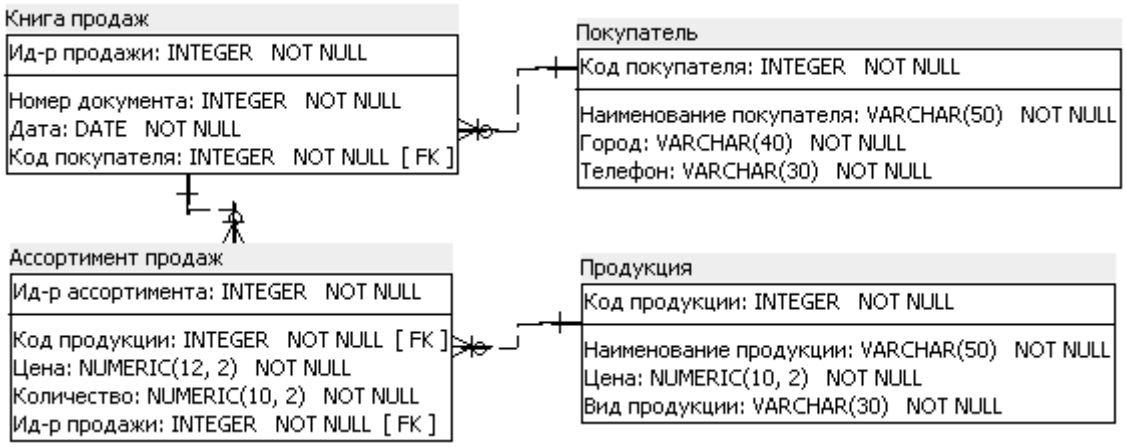


Рис.18 Окончательная логическая модель данных Продажи

Задание. Спроектировать модель БД для одного из следующих примеров:

1. Поставки сырья

Предприятие для производства готовой продукции использует сырье.

Сырье характеризуется: артикулом, наименованием, ед.измерения

Сырье приобретается у поставщиков. По каждому постащику известны: код, наименование, адрес, контактный телефон.

Поставки сырья фиксируются в книге поставок, где записываются: дата, номер накладной, поставщик, а также ассортимент поставки (артикул, цена, количество).

2. Производство готовой продукции

В цехах основного производства изготавливается готовая продукция, которая характеризуется артикулом, наименованием, стоимостью единицы каждого сорта (1, 2, 3) и принадлежностью к определенной группе. Каждая группа характеризуется кодом и наименованием.

Изготовление готовой продукции выполняют работники бригад, по каждой из которых известны: код, наименование, ФИО бригадира, цех (код и наименование).

Информация о выработке продукции сдается каждой бригадой в виде накладной, в которой указаны номер, дата, бригада, и ассортимент выпущенной продукции (артикул, сорт, цена и количество).

3. Ремонт автомобилей

Автотехцентр производит ремонт автомобилей.

По каждому клиенту известны: код, тип (физическое или юридическое лицо), наименование (ФИО), ИНН, адрес, контактный телефон.

Для ремонта оформляется заказ-наряд. По каждому заказ-наряду фиксируется: дата оформления, номер, срок исполнения, клиент, модель автомобиля, госномер автомобиля.

Ремонт заключается в выполнении определенного круга работ, которые характеризуются: кодом, наименованием, стоимостью 1 нормо-часа. На выполнение ремонтных работ расходуются запчасти, которые характеризуются: номенклатурным номером, наименованием.

В заказ-нарядах фиксируется перечень выполненных работ (код и трудоемкость в нормо-часах) и перечень израсходованных материалов (номенклатурный номер и количество).

4. Движения сырья на складе

Предприятие для производства продукции использует сырье.

Сырье характеризуется: артикулом(кодом), наименованием, ед.измерения, принадлежностью к определенной группе (код и наименование).

Сырье хранится на складе. Сырье поступает от поставщиков и отпускается в цеха (подразделения) основного производства. По каждому поставщику известны: код, наименование, адрес, контактный телефон. По каждому цеху известны: код, наименование, ФИО начальника. Другими операциями на складе могут быть: возврат сырья поставщику, возврат сырья из цехов основного производства, списание сырья, пришедшего в негодность. Для учета сырья на складе ведется книга движения сырья, в которой отражаются: дата, номер накладной, вид операции, контрагент (от кого получено, кому отпущено), а также ассортимент (код сырья, цена, количество).

Порядок выполнения работы

1. Изучить теоретический материал
2. Установить необходимые программные средства
3. Спроектировать модель БД для выбранного варианта

Содержание отчета

1. Результаты выполнения задания
2. Схема логической модели данных

Практическое занятие № 32
Физическое проектирование БД

Цель занятия. Познакомиться с этапом физического проектирования, типами данных PostgreSQL. Выполнить физическое проектирование модели данных.

Краткие теоретические сведения

Физическое проектирование БД - это представление абстрактных объектов логической модели *в структуры данных конкретной СУБД* для обеспечения эффективности выполнения запросов. Сущности в физической модели называются таблицами, а атрибуты – колонками (столбцами). Колонкам назначается более конкретный тип данных из возможных типов выбранной СУБД.

Задачами физического проектирования являются:

- выбор целевого сервера для создания БД;
- уточнение типов данных колонок (столбцов) таблиц.

Физическое проектирование будем выполнять для логической модели, разработанной для предметной области продаж (рис.1), созданной на предыдущем практическом занятии. Необходимо открыть проект *Продажи*, созданный при разработке логической модели. Открытие проекта выполняется командой меню *Файл – Открыть проект* или нажатием кнопки или нажатием комбинации клавиш *Ctrl - O*.

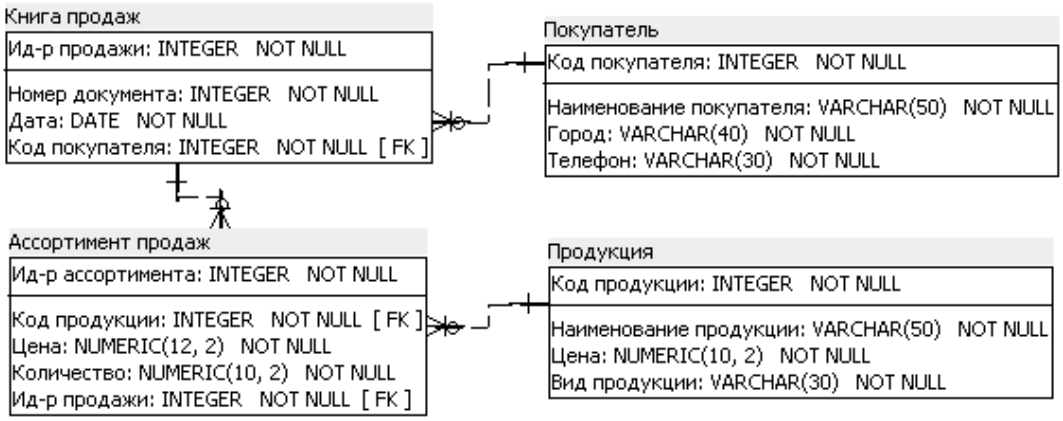


Рис.1 Логическая модель данных Продажи

Необходимо переключится на физический уровень представления модели. Для этого необходимо в окне настройки свойств проекта (рис.2) задать вид отображаемой модели (переключатель `Display Tables and Columns`) на `Physical Names`.

В качестве сервера БД выберем PostgreSQL.

Далее необходимо убедиться, что все имена на физическом уровне (таблиц, колонок, индексов) заданы в латинском алфавите.

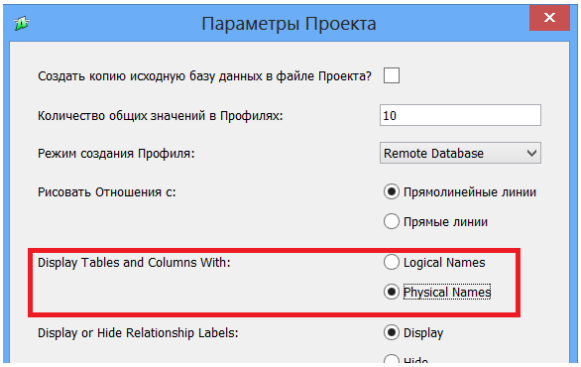


Рис.2 Установка физического представления

При выборе типов колонок целесообразно следовать следующим рекомендациям.

Для колонок таблицы следует выбирать типы данных:

для дат – тип DATE;

для целых числовых значений – INTEGER (для чисел длиной 4 байта, т.е. в диапазоне от -2147483648 до +2147483647) или BIGINT (8 байт);

для вещественных (дробная часть отделяется «точкой») – NUMERIC;

для символьных – VARCHAR(XXX), где XXX – максимальное количество символов в колонке.

Для количественных колонок вещественного типа (объем, количество и т.п.) зададим тип *Numeric(m,n)*. Здесь m – задает общее количество позиций, занимаемое числовым значением (включая дробную часть), а n – количество позиций в дробной части. Для колонок для стоимостных значений (цена, стоимость и т.п.) n=2.

Для колонок обязательных к заполнению необходимо задать опцию NOT NULL . Для этого необходимо выключить флаг *Позволить Nulls* (рис.3).

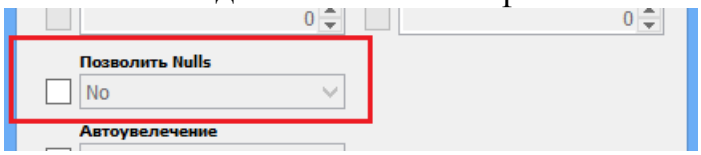


Рис. 3 Настройка обязательности заполнения колонки

Для колонок первичного ключа, являющихся суррогатными ключами, выбирается тип *Integer* с заданием параметра автоувеличения (рис.4). Дополнительно задается имя объекта последовательности, которая реализует механизм автоувеличения. Имя последовательности будем задавать по шаблону «имя_таблицы_seq». Для таблицы *Книга продаж* имя последовательности *book_seq*.

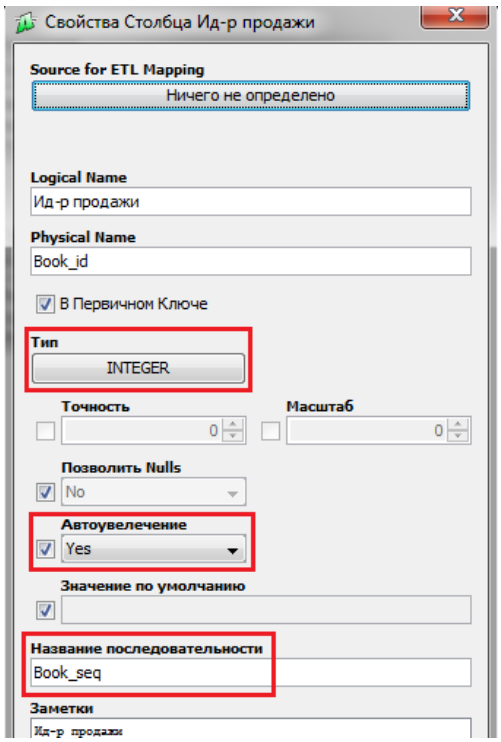


Рис. 4 Настройки столбца суррогатного первичного ключа

Для количественных колонок целесообразно задать значение по умолчанию, которое будет присвоено колонке, если значение не задано явно. Обычно, если значение не задано, целесообразно присвоить «нулевое» значение 0 (рис.5).

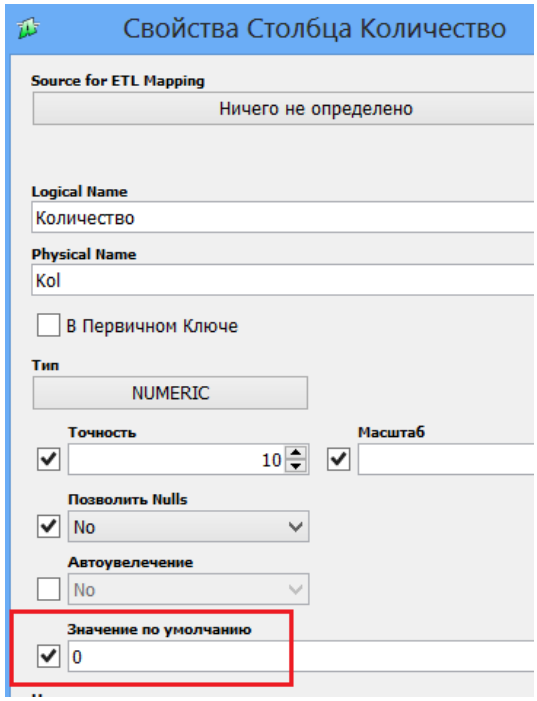


Рис. 5 Настройка значения по умолчанию

Далее необходимо проверить настройки колонок (столбцов) внешнего ключа. Необходимо установить возможность значению внешнего ключа

принимать значения Null, и снять отметку с флага *Автоувеличение* (рис. 6). В нашей физической модели во всех таблицах для внешних ключей необходимо установить *Позволить Null = No*.

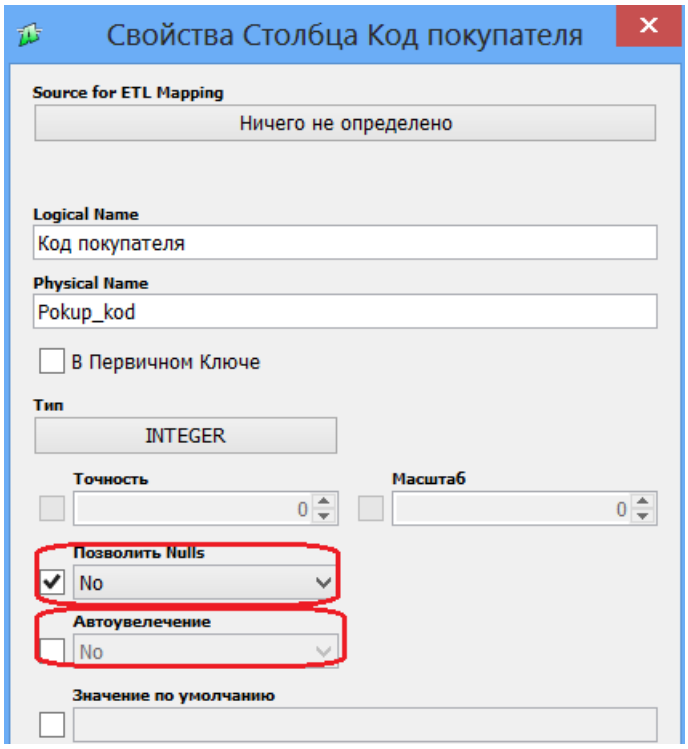


Рис.6 Задание свойств колонки внешнего ключа

Также целесообразно в колонках внешнего ключа очистить поле *Название последовательности*. Для этого надо временно задать *Автоувеличение=Yes* и очистить поле *Название последовательности* (рис. 7). Затем надо восстановить *Автоувеличение=No*.

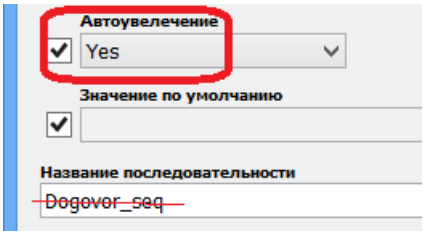


Рис. 7 Очистка Названия последовательности в колонке внешнего ключа

В заключение необходимо настроить свойства связей между отношениями. Для этого необходимо дважды щелкнуть мышкой по связи или выделив связь в контекстном меню выбрать пункт *Свойства отношения* (рис. 8). В открывшемся окне *Свойства отношения* (рис. 9) необходимо задать правила:

Update Rule =Restrict
Delete Rule =Restrict

Установленные правила запрещают изменять значения первичного ключа в целевом (родительском) отношении при наличии ссылающихся строк в ссылающемся (подчиненном) отношении.

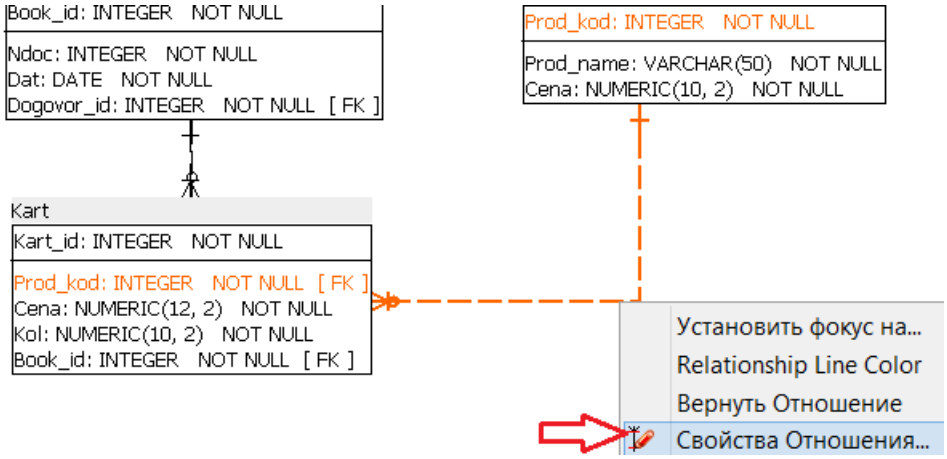


Рис. 8 Открытие окна Свойства отношения

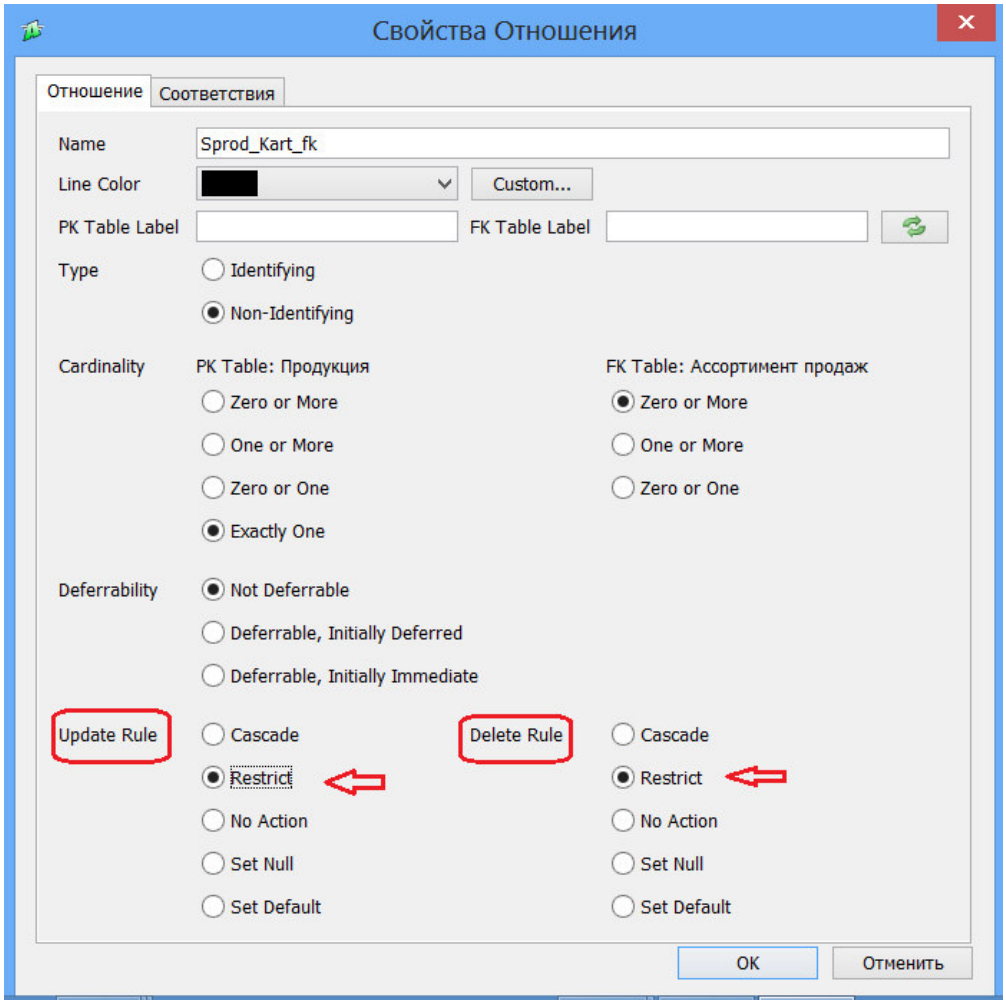


Рис.9 Настройка связи

Окончательная физическая модель данных представлена на рис.10.

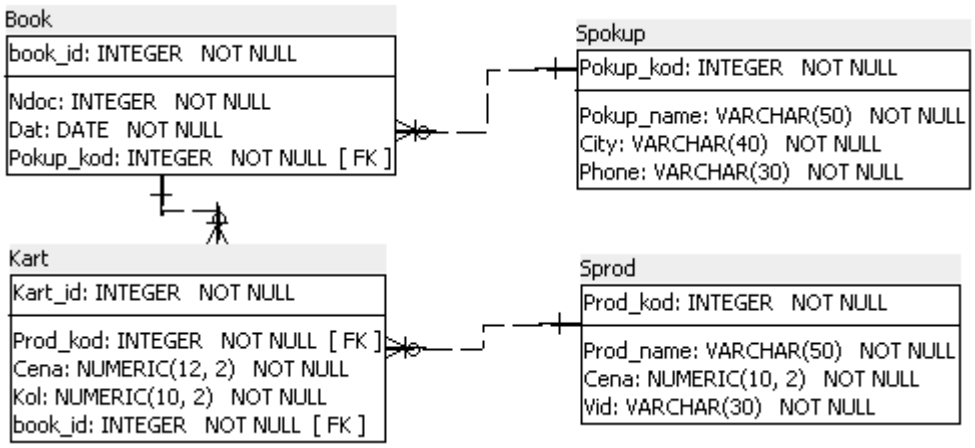


Рис. 10 Физическая модель данных Продажи

Задание.

Выполнить создание физической модели для логической модели, созданной на практическом занятии по логическому проектированию БД.

Порядок выполнения работы

1. Изучить теоретический материал
2. Спроектировать физическую модель БД для выбранного варианта

Содержание отчета

1. Результаты выполнения задания
2. Схема физической модели данных

Практическое занятие № 33
Программные средства работы с БД PostgreSQL. PgAdmin

Цель занятия. Познакомиться с программной средой администрирования и работы с базами данных pgAdmin. Научиться создавать базы данных и пользователей в PostgreSQL.

Краткие теоретические сведения

По данным сайта https://db-engines.com/en/ranking верхние строчки рейтинга по популярности (по данным на август 2021 года) занимают 4 СУБД:

1) Oracle
2) MySQL
3) Microsoft SQL Server
4) PostgreSQL

СУБД Oracle и Microsoft SQL Server являются проприетарными, т.е. платными в использовании (есть облегченные версии в открытом доступе), а MySQL и PostgreSQL являются свободными в использовании с полным функционалом.

Из двух открытых СУБД MySQL лучше подходит для веб-сайтов и простых онлайн-транзакций (высокая скорость на чтение), для простых веб-приложений. PostgreSQL лучше подходит для больших и сложных аналитических задач, операций с большими объемами данных, лучше справляется со сложными операциями чтения-записи с одновременной валидацией данных.

PostgreSQL – более функциональная и лучше подходит

- для управления большими базами данных,
- для выполнения сложных запросов,
- по поддержке NoSQL и разнообразию типов данных,
- по управление параллельным доступом.

PostgreSQL сопоставима с Oracle по следующим критериям:

- производительность,
- безопасность,
- масштабируемость,
- обновляемость,
- уровень техподдержки,
- работа с большими объемами данных

Многие разработчики рассматривают PostgreSQL как версию Oracle с открытым исходным кодом.

PostgreSQL вне конкуренции по цене владения (стоимость лицензии и поддержки). Для Oracle стоимость 1 лицензии на 1 многоядерный процессор

может составлять несколько миллионов рублей в год. Для PostgreSQL – 0 рублей.

В силу указанных выше причин для создания БД была выбрана СУБД PostgreSQL.

Установка PostgreSQL

Дистрибутив для установки СУБД PostgreSQL можно скачать по адресу:

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

При скачивании надо выбрать операционную систему компьютера, на котором будет устанавливаться СУБД и версию СУБД. В данном курсе рассматривается версия 9.6.

Предварительно до скачивания продукта необходимо зарегистрироваться.

Для установки PostgreSQL необходимо:

1) Загрузить пакет установки PostgreSQL с сайта и сохранить его в папке на диске
2) Запустить исполняемый файл. В процессе установки можно использовать значения по умолчанию, предлагаемые мастером установки. Только потребуется задать пароль для суперпользователя postgres. Для удобства можно задать пароль, совпадающий с именем суперпользователя - postgres. Запомните этот пароль, поскольку он является паролем администратора базы данных и потребуется позже.

Программа pgAdmin

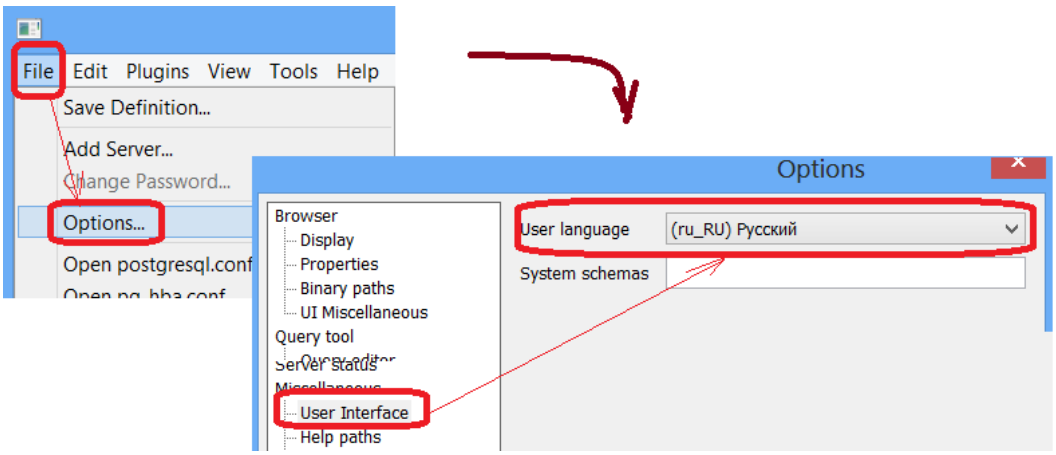
Все действия выполняются через административную утилиту для работы с БД pgAdmin. При установке сервера БД будет установлена одна из версий: pgAdmin III или pgAdmin 4. Первая работает как локальное приложение, а вторая – через браузер.

Желательно для работы установить русскоязычный пользовательский интерфейс (рис 1). После изменения языка пользовательского интерфейса необходимо перезагрузить программу.

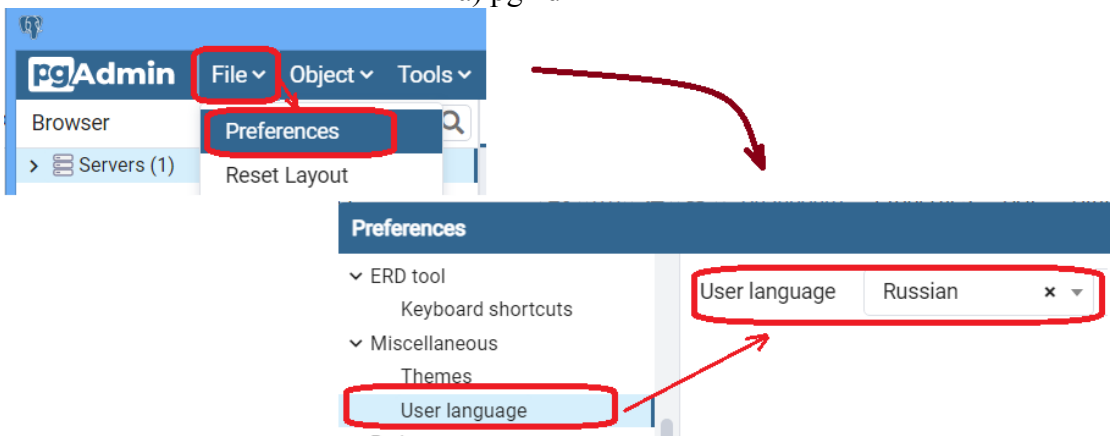
Общий вид запущенной программы приведен на рис. 2.

Далее рассмотрим процесс создания БД и пользователей. Рассмотрим следующие вопросы:

1) Создание подключения к серверу БД
2) Создание пользователей (ролей входа)
3) Создание БД
4) Создание схемы данных

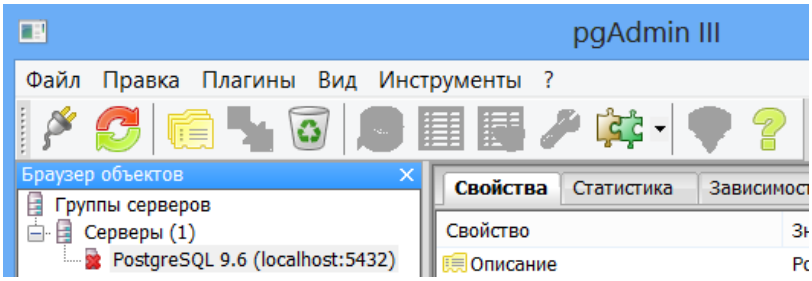


а) pgAdmin III

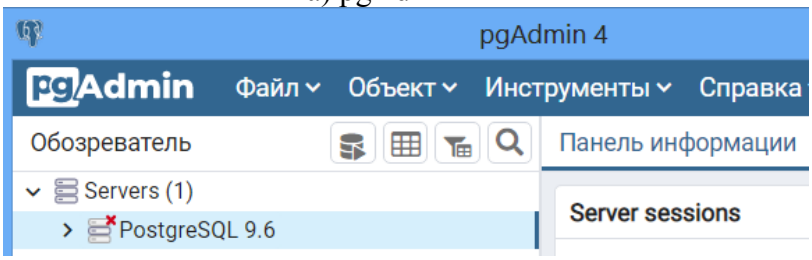


б) pgAdmin 4

Рис. 1. Установка русскоязычного интерфейса



а) pgAdmin III

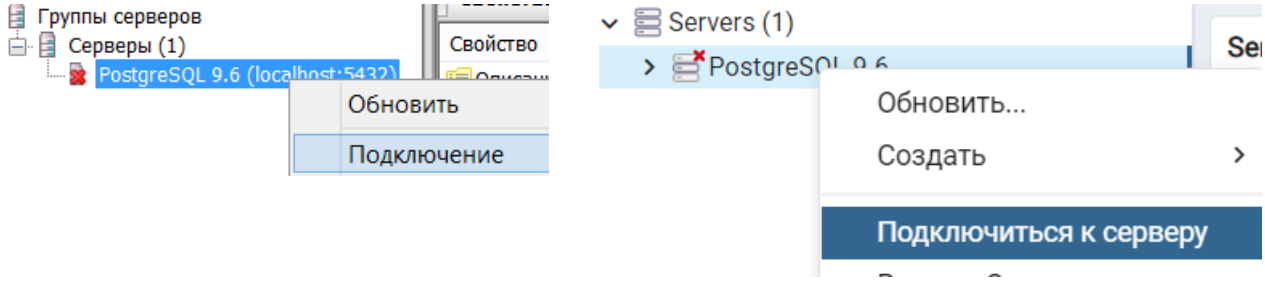


б) pgAdmin 4

Рис.2. Основное окно pgAdmin.

1) Создание подключения к серверу БД

Для выполнения любых действий требуется подключиться к серверу. Первое подключение выполняется под учетной записью суперпользователя postgres, созданной при установке СУБД (рис.3).



а) pgAdmin III б) pgAdmin 4

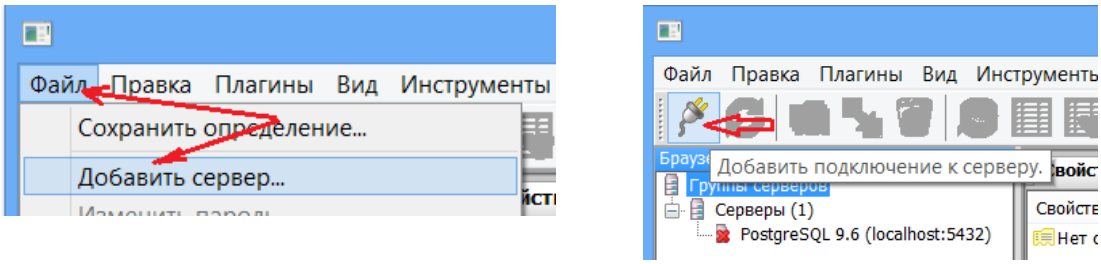
Рис. 3. Подключение к серверу БД

Можно создать свой сервер и выполнить к нему подключение. Для этого необходимо в pgAdmin III выполнить пункт меню

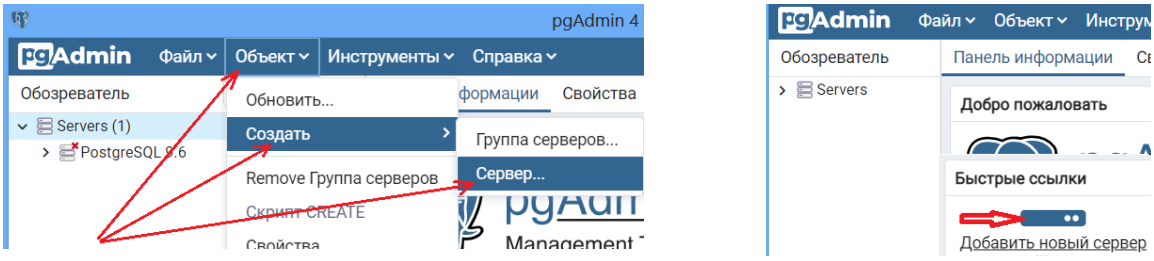
Файл-Добавить сервер

или щелкнуть по иконке **Добавить подключение к серверу** (рис.4а).
В pgAdmin 4 необходимо выполнить пункт меню (рис. 4б)

Объект – Создать - Сервер



а) pgAdmin III



б) pgAdmin 4

Рис. 4. Создание нового подключения

В окне создания нового подключения (сервера) в pgAdmin III необходимо ввести (рис. 5):

– *Имя* подключения: введем local_pg (можно ввести другое);

– *Хост*: введем localhost, т.к. сервер БД расположен на локальном компьютере. Если СУБД работает на другом компьютере, то вводится ip-адрес или DNS-имя этого компьютера;
– *Порт* 5432 оставляем без изменения;
– Имя БД (*Обслуживание DB*): введем postgres - имя стандартной БД;
– *Имя пользователя*: введем postgres - имя суперпользователя, созданного при установке;
– *Пароль: введем* пароль, заданный для суперпользователя при установке СУБД;
– Включаем флаг *Сохранить пароль,* чтобы не вводить его при следующих подключениях.

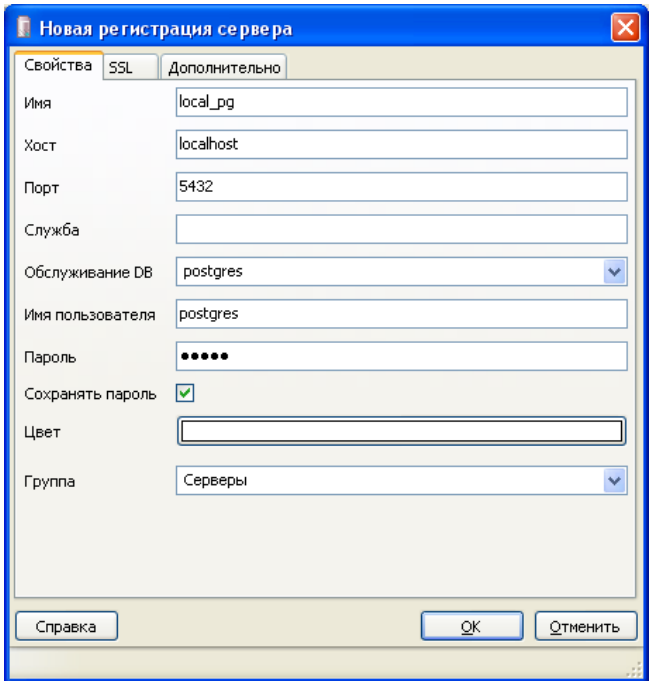


Рис. 5. Задание параметров нового подключения в pgAdmin III

В окне создания нового подключения (сервера) в pgAdmin 4 параметры подключения задаются на двух вкладках (рис. 6).

На вкладке Общие:

– *Имя* подключения: введем local_pg (можно ввести другое).

На вкладке *Соединение*:

– *Имя/адрес сервера*: введем localhost, т.к. сервер БД расположен на локальном компьютере. Если СУБД работает на другом компьютере, то вводится ip-адрес этого компьютера;
– *Порт* 5432 оставляем без изменения;
– Имя БД (*Служебная база данных*): введем postgres - имя стандартной БД;
– *Имя пользователя*: введем postgres - имя суперпользователя, созданного при установке;

– *Пароль: введем* пароль, заданный для суперпользователя при установке СУБД;
– Включаем флаг *Сохранить пароль,* чтобы не вводить его при следующих подключениях.

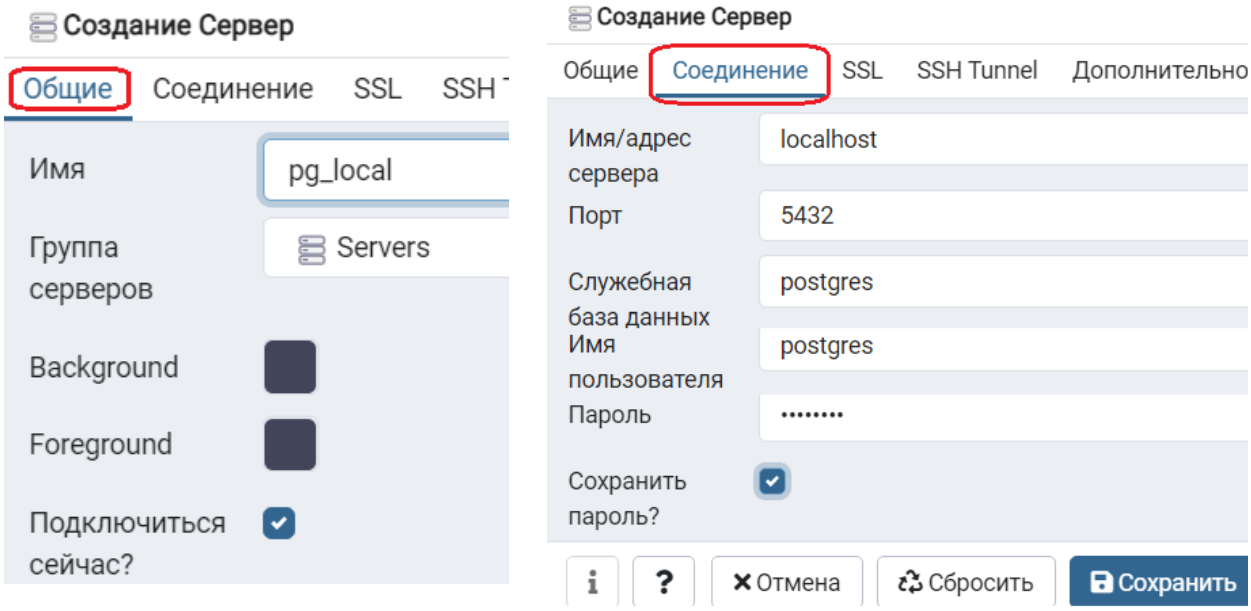
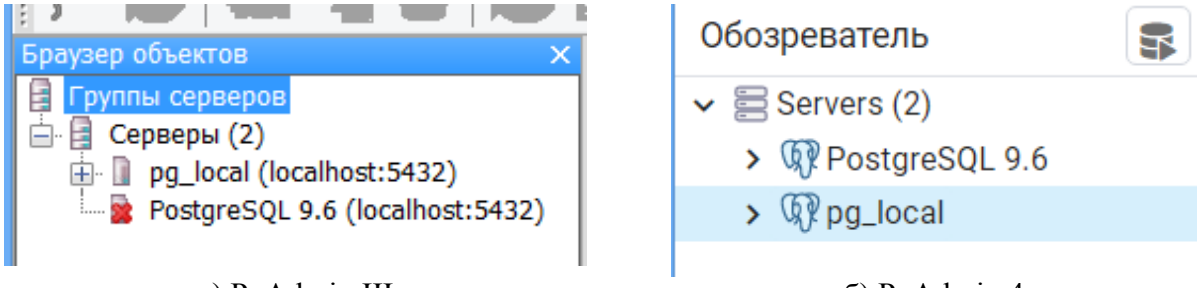


Рис. 6. Задание параметров нового подключения в pgAdmin 4

Если после нажатия OK (Сохранить) не выводится сообщение об ошибке, то соединение установлено правильно. Новое подключение отобразится в дереве браузера объектов (рис.7)



а) PgAdmin III б) PgAdmin 4

Рис.7. Новое подключение в дереве объектов

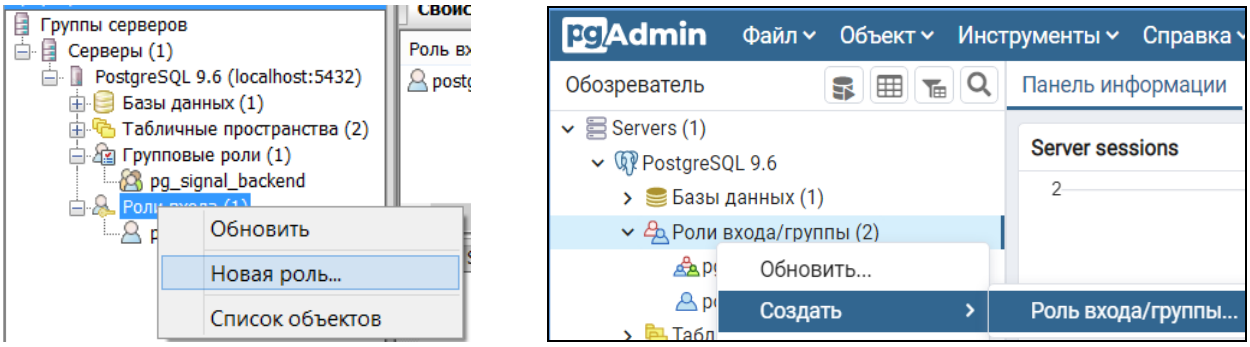
2) Создание пользователей (ролей входа)

Для работы в БД необходимо создать роль входа. Созданную при установке СУБД роль суперпользователя postgres использовать нецелесообразно. Ее используют только для создания своих ролей и в реальных БД после этого удаляют. Часто вместо термина «роль» используют термин «пользователь», так как именно пользователь выполняет работу в БД и его наделяют определенными правами. Фактически пользователь

выполняет определенную роль при взаимодействии с БД. Поэтому для пользователя и используется термин «роль».

Для создании роли в дереве объектов необходимо выделить узел **Роли входа** и в контекстном меню в pgAdmin III выбрать пункт **Новая роль**, а в pgAdmin 4 – пункт Создать и подпункт Роль входа/группы (рис. 8).

Замечание. Контекстное меню активизируется при нажатии на выделенном объекте правой кнопки мыши.



а) PgAdmin III б) PgAdmin 4

Рис.8. Создание новой роли входа

Будет открыто окно *Новая роль*, в котором надо задать параметры роли.

В pgAdmin III в окне на вкладке *Свойства* введите *Имя роли*, например admin (рис. 9).

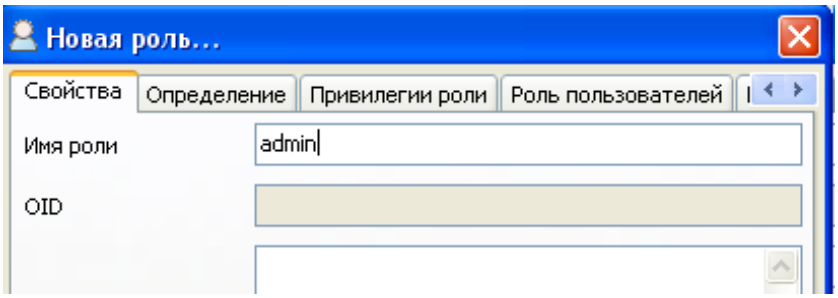


Рис.9. Задание имени роли

а вкладке *Определение* введите *Пароль* и его *Подтверждение* (рис.10). Можно также определить ограничение по времени, задав конечную дату активности пользователя. Но мы этого делать не будем.

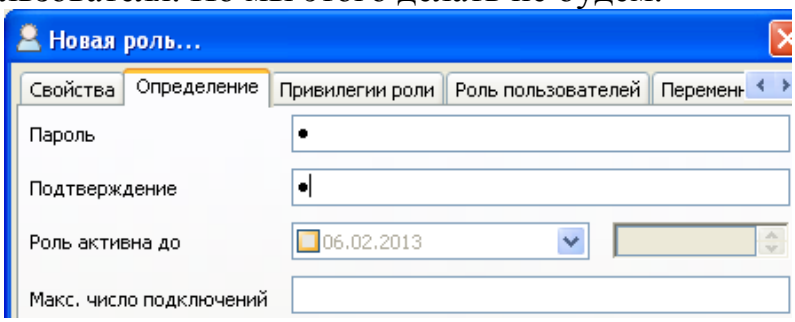


Рис.10. Задание пароля для подключения к серверу

На вкладке *Привилегии* необходимо назначить роли ее права , включив флаги нужные привилегий (рис.11). При создании новой роли по умолчанию будут включены флаги *Вход разрешен* и *Наследует права у родительских ролей*. Оставим эти привилегии, но дополнительно добавим привилегию *Суперпользователь*. Другие привилегии можно не добавлять, т.к. суперпользователь обладает всеми привилегиями. Отметим, что термины «права» и «привилегии» являются синонимами.

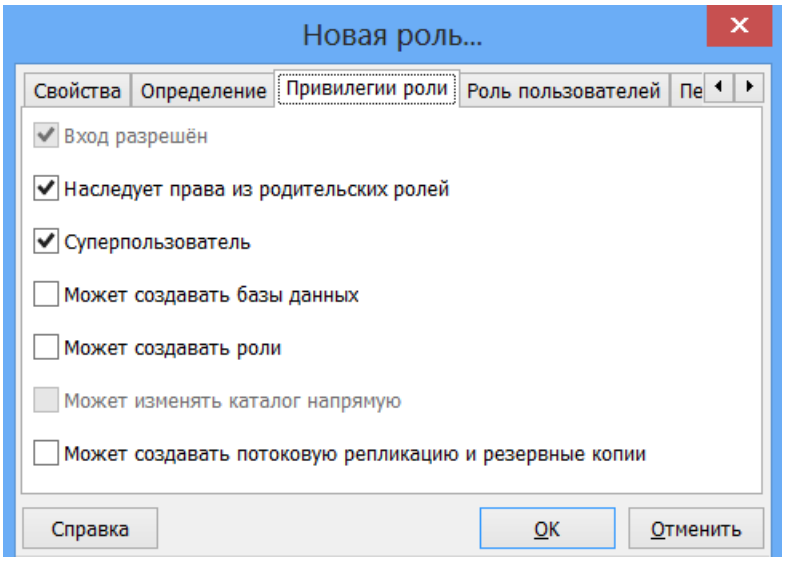


Рис.11. Задание привилегий для роли

Задав все параметры, нажмем кнопку *ОК* для сохранения роли.

В pgAdmin 4 вкладки задания параметров роли немного отличаются по названию (рис.12). На вкладке *Права* (вместо *Привилегии*) все права задаются явно. Вместо флажков здесь используются переключатели с положениями «Да»/«Нет». При щелчке мышью на переключателе он меняет свое значение на противоположное. При добавлении права *Superuser* (суперпользователь) остальные права добавляются автоматически. И в конце надо нажать кнопку *Сохранить*.

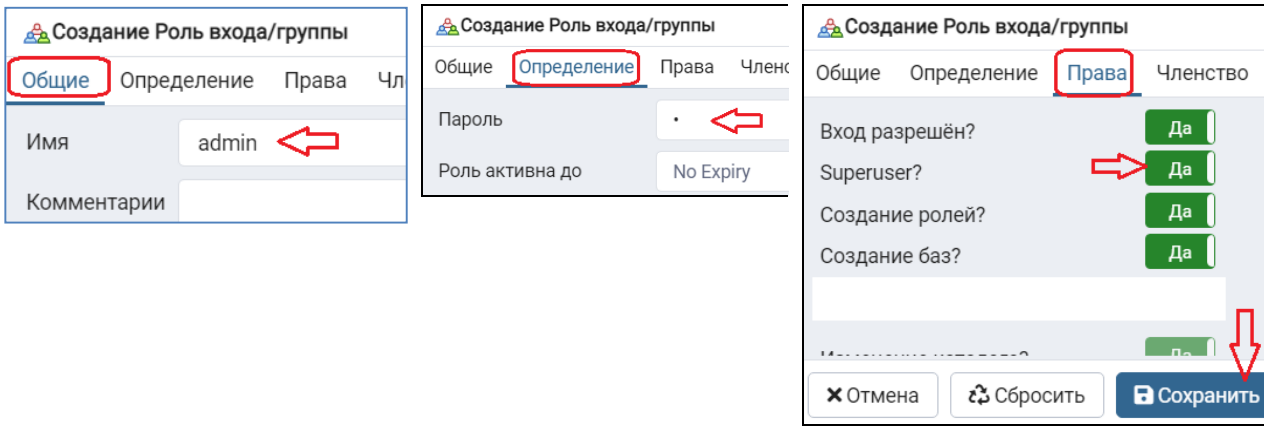
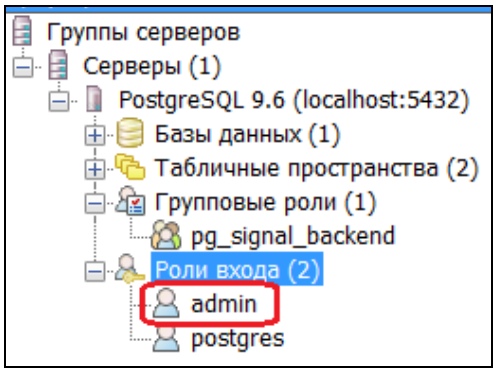


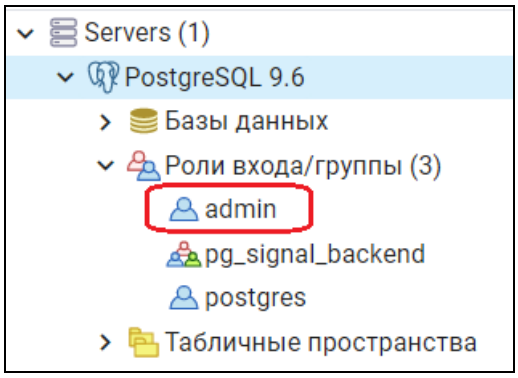
Рис.12. Задание параметров роли в pgAdmin 4

Созданная роль появится в списке ролей входа (рис. 13).

Позднее можно изменить параметры роли (пароль и привилегии). Для этого в дереве необходимо выделить нужную роль и в контекстном меню выбрать пункт *Свойства*.



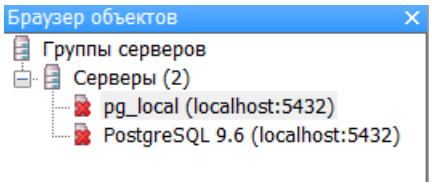
а) PgAdmin III



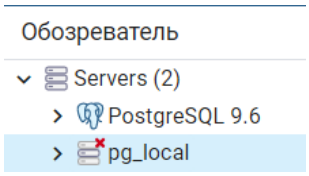
б) PgAdmin 4

Рис.13. Отображение созданной роли

Для дальнейшей работы будем использовать новую роль. Изменим ранее созданное подключение pg_local, использовав его для новой роли. Для изменения параметров подключения оно должно быть не активно. Неактивное подключение помечается красным крестиком слева от имени подключения (рис.14).



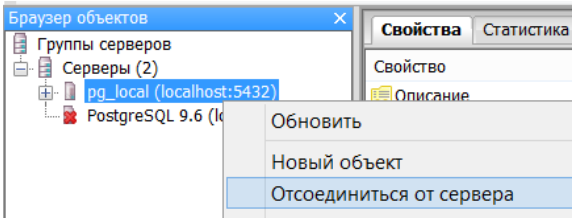
а) PgAdmin III



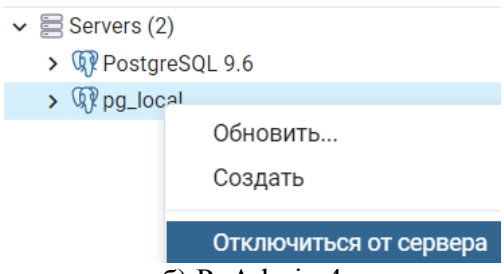
б) PgAdmin 4

Рис.14. Неактивное подключение

Если подключение активно, то необходимо отсоединится от сервера, закрыв подключение (рис. 15).



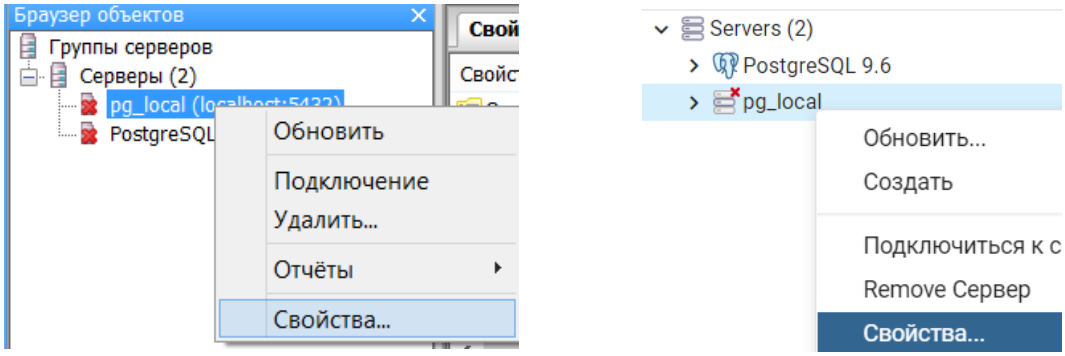
а) PgAdmin III



б) PgAdmin 4

Рис.15. Отключение от сервера

Для изменения подключения необходимо выделить его в дереве объектов и в контекстном меню выбрать пункт Свойства (рис. 16). Откроется окно подключения, в котором на вкладке подключения необходимо изменить имя пользователя и пароль (рис.5 и рис.6 выше по тексту) и сохранить подключение.



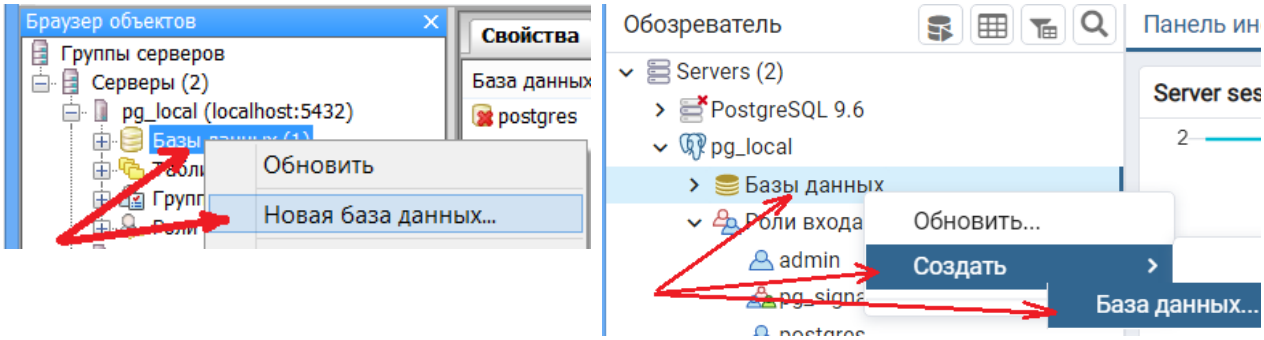
а) PgAdmin III б) PgAdmin 4

Рис.16. Изменение параметров подключения

3. Создание БД

Базу данных может создавать только пользователь (роль), которому было назначено право (привилегия) *Создание базы данных* или *Суперпользователь*. Такой пользователь с именем admin был создан на предыдущем шаге. Активизируем подключение к серверу pg_local (оно выполняется именно этой ролью), сделав двойной клик на имени подключения.

В pgAdmin III в браузере объектов на строке *Базы данных* необходимо в контекстном меню выбрать пункт *Новая база данных*, а в pgAdmin 4 – выбрать пункт *Создать* и подпункт *База данных* (рис. 17).



а) PgAdmin III б) PgAdmin 4

Рис. 17 Запуск процесса создания БД

Откроется окно создания БД *Новая база данных*. Окно создания БД в pgAdmin III и pgAdmin 4 не отличается.

На вкладке *Свойства* необходимо задать *Имя* БД и выбрать *Владельца* из числа созданных пользователей. Зададим имя db_test и выберем

владельцем пользователя admin (рис. 18). Вы можете выбрать другое имя для БД. Владелец БД обладает всеми правами на все ее объекты. Другие же пользователи (роли) могут работать с ее объектами, если им предоставит такое право владелец или суперпользователь.

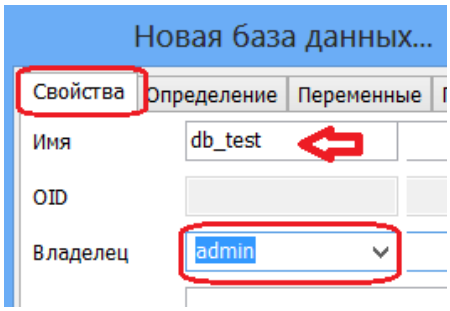


Рис.18 Задание имени БД и ее владельца

На вкладке *Определение* необходимо (рис.19):

- В поле Кодировка следует выбрать *UTF8* или *WIN_1251*. Кодировка используется для представления текстовых данных на локальных языках (мы будем использовать кириллицу). При этом надо иметь в виду, что при подготовке текстовых данных для загрузки в БД в текстовых редакторах необходимо использовать также выбранную кодировку. Для работы с такими файлами можно использовать редактор Notepad++ (notepad-plus-plus.org), в которых всегда показывается кодировка символом и просто перейти к нужной кодировке. В этой же кодировке будут выгружаться данные;
- В поле Шаблон выберем *template0*;
- В полях Сопоставление (в pgAdmin 4 *Правило сортировки*) и Тип символа выберем кириллическую *Russian_Russia.1251*;
- Остальные поля не будем изменять.
  После задания всех параметров БД необходимо нажать *OK*.

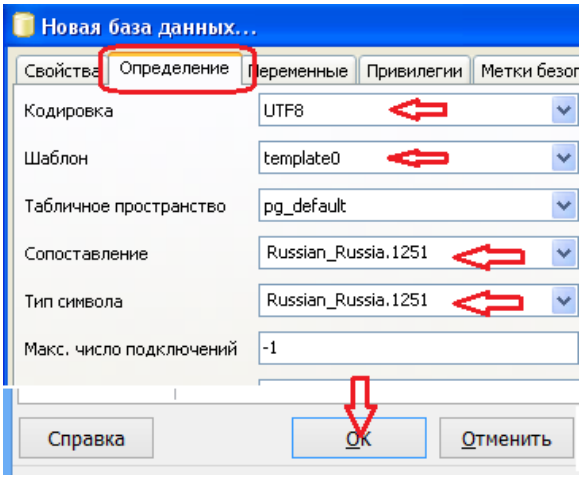


Рис. 19 Задание параметров БД

Надо иметь в виду, что созданная база данных пока представляет собой простой контейнер, содержащий только служебную информацию. База данных на этом этапе не содержит таблиц для хранения данных предметной области. Они будут добавлены в базу данных позже, и это будет рассмотрено на следующих практических занятиях.

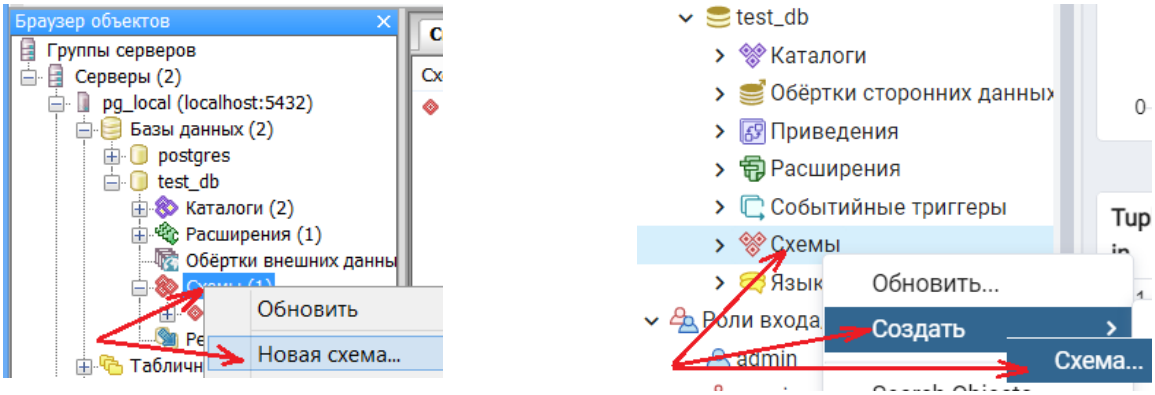
4. Создание схемы данных

Все объекты БД PostgreSQL хранятся в схемах. Схемы используются для логической группировки объектов БД (таблиц, представлений, процедур). Все объекты в БД имеют составное имя: *имя_схемы.имя_объекта*.

Пользователь может обращаться к объектам просто по имени объекта (без добавления имени схемы), если схему в сеансе сделать текущей.

Создадим в БД test_db схему prod для размещения объектов модели данных *Продажи*.

Для этого необходимо в браузере объектов на строке *Схемы* в контекстном меню в pgAdmin III выбрать пункт *Новая схема*, а в pgAdmin 4 – пункт *Создать* и подпункт *Схема* (рис. 20)



а) PgAdmin III б) PgAdmin 4

Рис.20 Создание новой схемы

В pgAdmin 4 будет выведено окно *Создание схема* (*Новая схема* в pgAdmin III).

На вкладке *Свойства* зададим *Имя* схемы `prod` и из списка выберите *Владельца* admin (рис. 21).

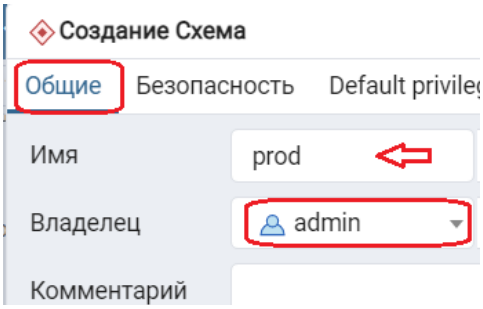


Рис. 21 Задание имени схемы и ее владельца

На вкладке Безопасность (Привилегии в pgAdmin III) можно задать права различным пользователям на работу с объектами схемы. Для владельца схемы будут автоматически предоставлены полные права, поэтому данную вкладку заполнять не будем. Нажмем кнопку ОК для сохранения схемы. Новая схема будет отображена в дереве объектов (рис. 22).

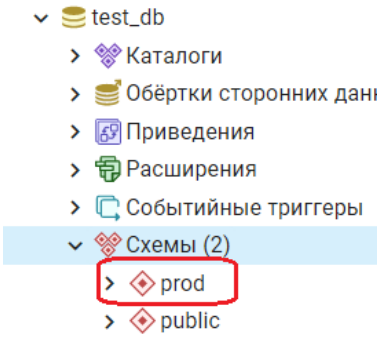


Рис. 22 Новая схема в дереве объектов

Задание.

1) Установить СУБД PostgreSQL
2) Создать роль входа с правами суперпользователя
3) Создать базу данных и схему для размещения данных

Порядок выполнения работы

1. Изучить теоретический материал
2. Установить необходимые программные средства
3. Создать роль входа с правами суперпользователя
4. Создать базу данных и схему данных

Содержание отчета

1. Скриншоты выполнения основных этапов работы
2. Схема физической модели данных

Практическое занятие № 34
Создание БД в PostgreSQL

Цель занятия. Научиться в среде SQL Power Architect для физической модели данных формировать сценарий SQL по созданию объектов БД, создавать подключение к серверу баз данных, выполнять сценарий SQL из программной среды SQL Power Architect и с использованием сценария в pgAdmin.

Краткие теоретические сведения

После создания физической модели данных в SQL Power Architect таблицы физической модели необходимо перенести в базу данных.

Для этого в SQL Power Architect формируется пакет команд SQL по созданию таблиц и других объектов базы данных, который называют сценарием SQL. Данный сценарий можно использовать двумя способами:

1) сохранить в файле для выполнения в pgAdmin;
2) выполнить команды сценария непосредственно из среды SQL Power Architect, используя подключение к серверу БД.

Первый способ используется при отсутствии соединения с БД на компьютере, на котором выполняется моделирование данных.

1. Создание сценария SQL схемы БД

Для формирования команд создания таблиц БД необходимо нажать кнопку *Сконструировать сценарий SQL*  или выполнить команду меню *Инструменты – Сконструировать* (рис.1).

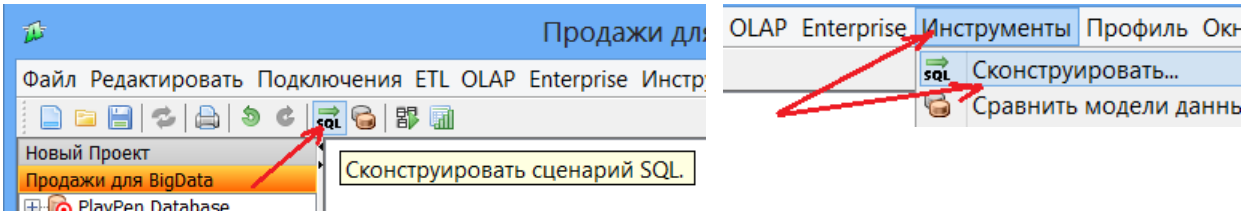


Рис. 1 Формирование сценария SQL

Откроется окно «Сконструировать сценарий SQL» (рис.2).

В окне необходимо:

- выбрать тип БД в списке *Генерировать DDL для БД типа* (в нашем случае PostgreSQL);
- Задать имя *Схемы* (в нашем случае prod). В данной схеме БД будут созданы таблицы;
- нажать кнопку ОК.

Если при формировании сценария будут ошибки в модели (дублирование имен объектов и т.п.) то будет выведено окно с перечнем таких ошибок (Рис. 3). Ошибки можно попытаться исправить. Варианты исправления показываются во всплывающем окне, если щелкнуть мышью в строке с предупреждением по колонке *Fix*. Для исправления ошибок надо нажать кнопку *Исправить все*. Можно также игнорировать ошибки, нажав кнопку *Игнорировать предупреждения*.

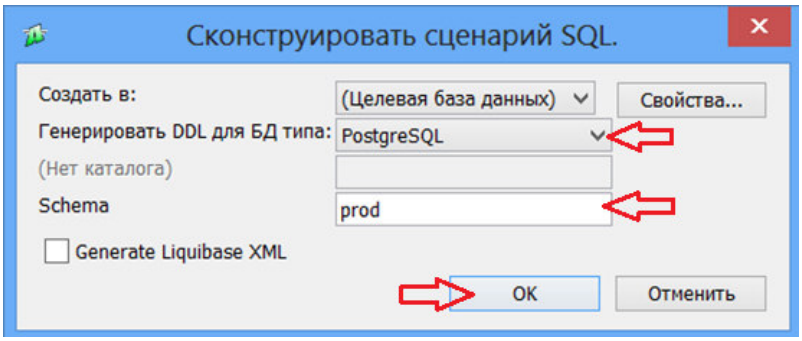


Рис.2 Задание параметров сценария

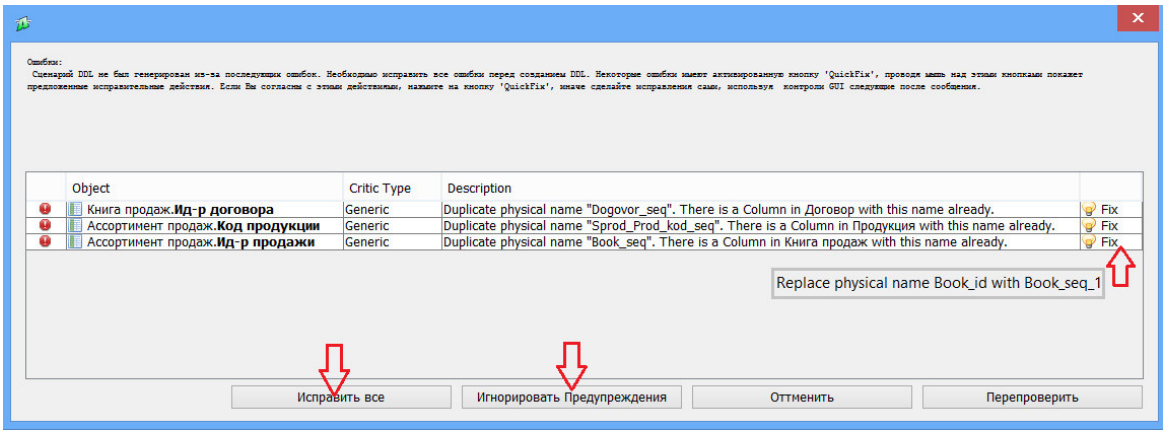


Рис.3 Окно с перечнем ошибок

Если ошибок при формирования сценария не обнаружено (также при игнорировании имеющихся ошибок), откроется окно с командами SQL (рис.4).

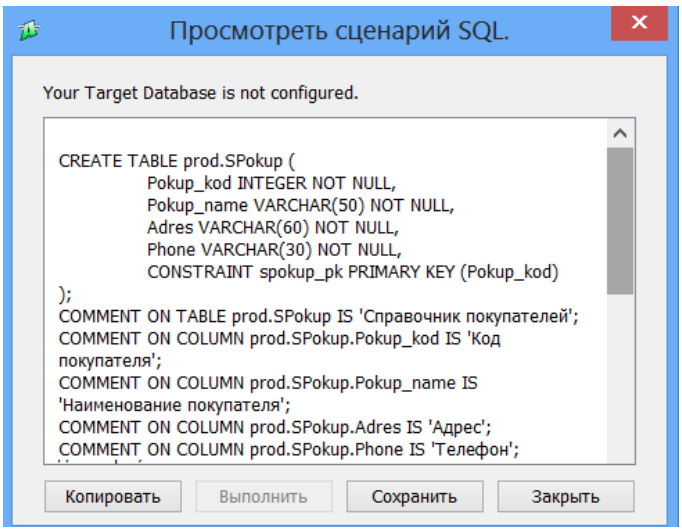
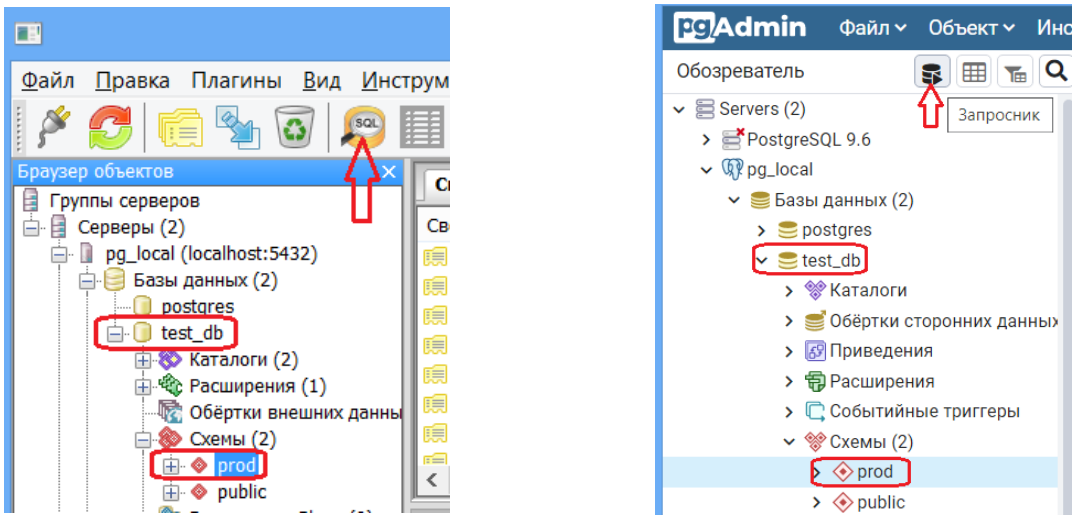


Рис. 4. Текст сгенерированного сценария SQL

Для сохранения команд сценария в файле необходимо нажать кнопку *Сохранить*. Далее надо выбрать папку и имя файла для команд сценария. Сгенерированный сценарий можно выполнить на целевом сервере БД. Необходимо гарантировать, что кодировка файла со сценарием UTF8. В этом можно убедиться, открыв сохраненный файл сценария в текстовом редакторе NotePad++. Если кодировка файла ANSI, то кодировку в редакторе надо преобразовать в UTF8.

Для выполнения сценария будем использовать программу pgAdmin. Запустим программу pgAdmin и выполним следующие действия (рис. 5):

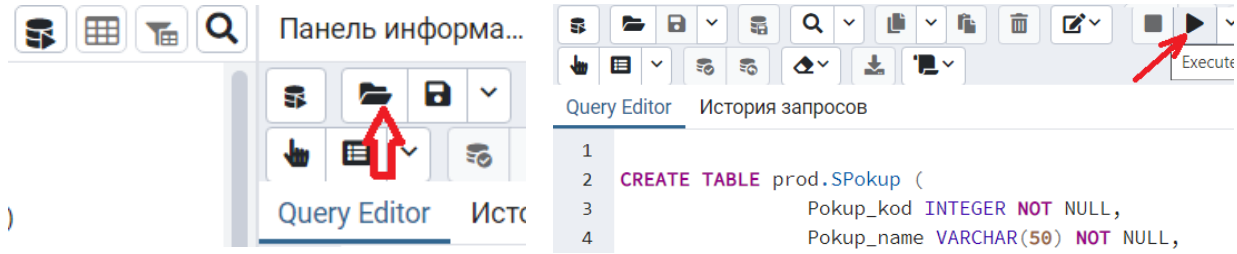
- активизируем подключение pg_local под учетной записью суперпользователя admin, созданное на предыдущем практическом занятии;
- выделим в браузере в базе данных test_db схему prod;
- откроем окно ввода SQL команд.



а) pgAdmin III б) pgAdmin 4

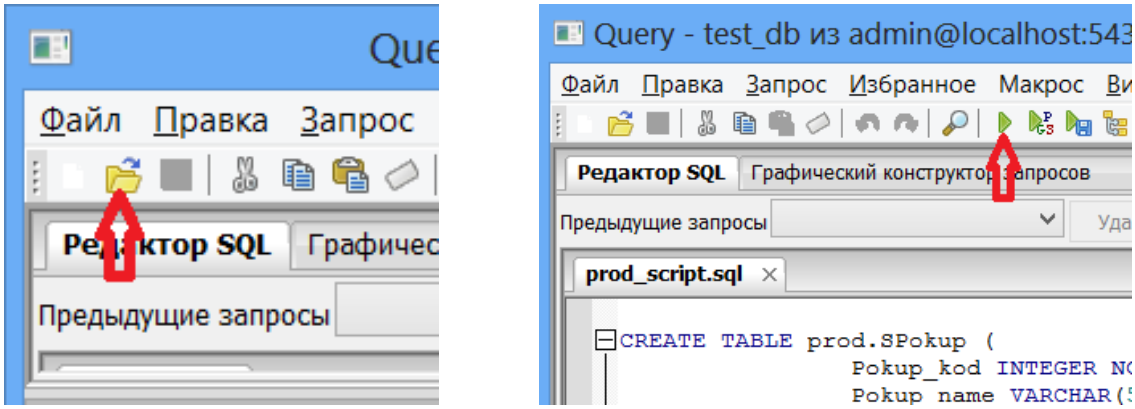
Рис.5 Вывод окна выполнения запросов

В окне ввода команд необходимо загрузить файл с командами сценария SQL, полученный при создании физической модели (рис. 6а, 7а). Для выполнения команд необходимо нажать кнопку *Выполнить запрос (Execute)* либо нажать кнопку F5 (рис. 6б, 7б).



а) б)

Рис. 6 Загрузка и выполнение сценария SQL в pgAdmin 4



а) б)

Рис. 7 Загрузка и выполнение сценария SQL в pgAdmin III

Если сценарий будет выполнен успешно, то в нижней части окна запросов на вкладке Сообщения будет выведен текст *Query returned successfully* (рис. 8), а в дереве объектов в схеме prod отобразятся таблицы и последовательности (рис. 9).

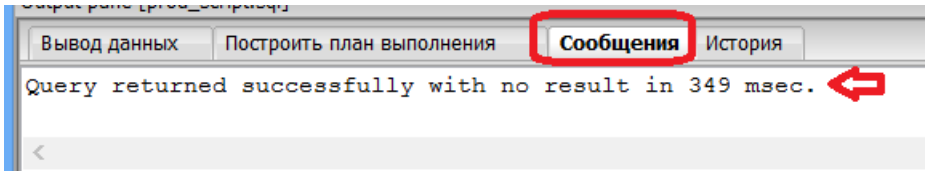


Рис.8 Успешное выполнение сценария

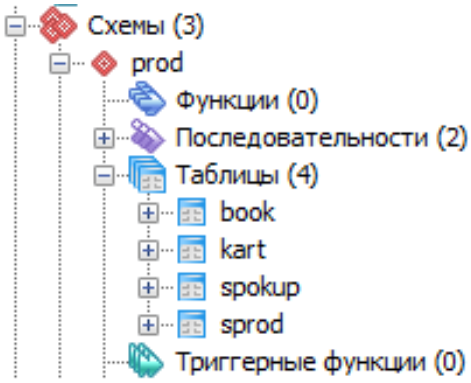


Рис. 9 Созданные таблицы в схеме prod

2. Выполнение сценария SQL из среды SQL Power Architect

Другим вариантом создания схемы базы данных является выполнение сценария в целевой БД непосредственно из среды SQL Power Architect. Для этого необходимо создать подключение к серверу БД PostgreSQL.

Для создания нового подключения выполним пункт меню

Подключения – Добавить подключение источника – Новое подключение

В окне создания нового подключения выполним следующие действия (рис.10):

1) Зададим имя подключения (например, pg_local);
2) Выберем тип сервера БД (PostgreSQL);
3) Зададим параметры подключения
   - имя хоста (*Hostname*), на котором запущен сервер БД. В нашем случае зададим *localhost*, т.к. это локальный компьютер, на котором выполняется работа;
   - номер порта (*Port*). *Порт* - это некоторое число, которое используется для идентификации процесса (программы), который должен обработать данные. Хотя работа выполняется на локальном компьютере, а не в сети, сервер БД подчиняется логике сетевой программы. Для всех сетевых программ существуют стандартные номера портов. Для сервера БД PostgreSQL таким стандартным номером порта является 5432. Данный номер будет подставлен автоматически при выборе сервера;
   - имя БД (Database). Введем имя созданной на предыдущем практическом занятии БД test_db;
4) Зададим имя пользователя (*Username*). Введем имя суперпользователя admin, созданного на прошлом практическом занятии;
5) Зададим пароль (*Password*). Также будем использовать пароль созданного суперпользователя – admin. Символы пароля при вводе будут замещаться символами «звездочка»;

6) Нажмем кнопку *Test Connection* для проверки работоспособности подключения. Если рядом с кнопкой появится текст

Connection test successful

то это означает, что подключение создано;

7) Нажмем кнопку ОК для сохранения подключения.

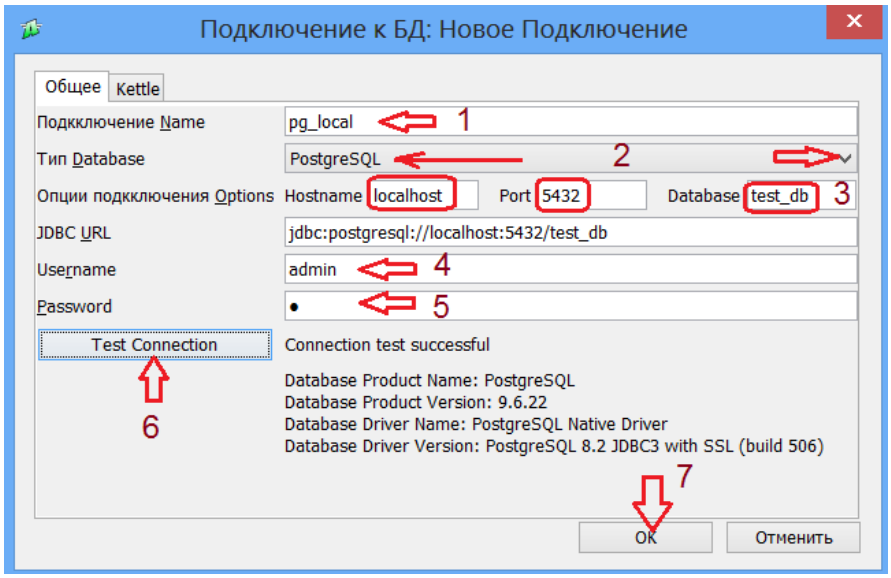


Рис.10 Создание подключения к серверу PostgreSQL

Замечание. При проверке подключения после нажатия кнопки *Test Connection* может быть выведено сообщение об ошибке. Оно может быть связано с неправильным заданием параметров подключения либо с отсутствием драйвера подключения к БД. Драйверы для различных БД устанавливаются автоматически, если при установке SQL Power Architect был выбран пакет установки, включающий эти драйверы (в имени пакета присутствует «jdbc»). Если использовался другой пакет, то драйвер надо скачать в интернете (найти самостоятельно) и подключить к программе в окне менеджера подключений, вызвав его в пункте меню

Подключения – Менеджер Подключений к Базам Данных

В окне менеджера необходимо нажать кнопку *Драйверы JDBC* (рис.11)

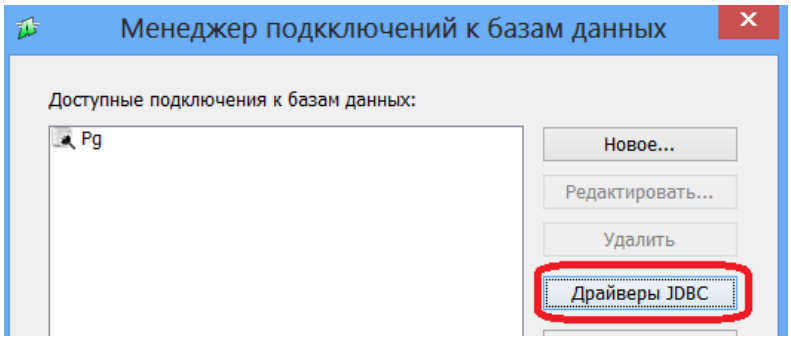


Рис. 11 Окно подключения драйверов JDBC

В открывшемся окне (рис.12) необходимо выбрать тип сервера БД (PostgreSQL) и, нажав кнопку Добавить JAR, найти скачанный драйвер. Подключенный драйвер отобразиться в окне.

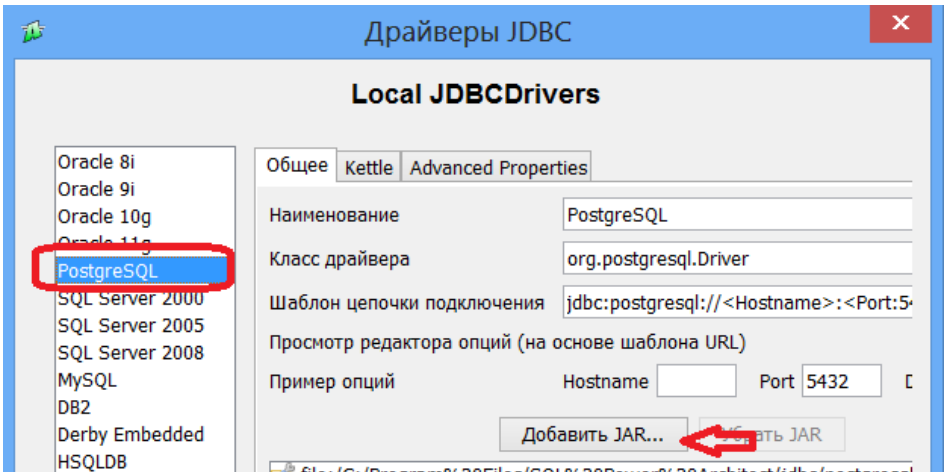


Рис. 12 Подключение драйвера JDBC для PostgreSQL

После создания подключения запустим формирование сценария, нажав кнопку *Сконструировать сценарий SQL*  или выполнив команду меню *Инструменты – Сконструировать* (рис.1). Но в отличие от первого варианта в поле «*Создать в»* выберем подключение к целевой БД, созданное выше, и нажмем кнопку ОК (рис.13).

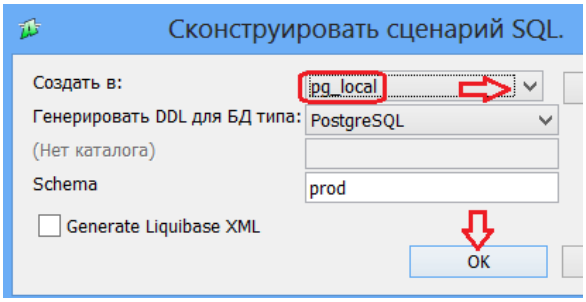


Рис. 13 Создание сценария с подключением к целевому серверу БД

В окне сценария SQL нажмем кнопку «Выполнить» (рис.14).

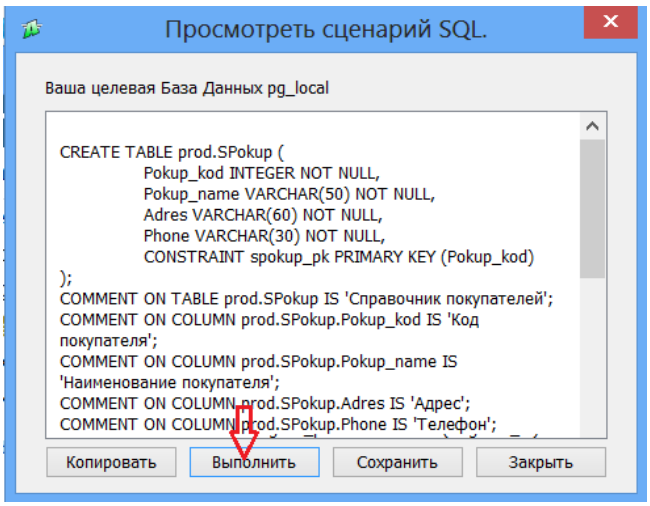


Рис.14 Выполнение сценария в целевой БД

Если все выполнено правильно, то получим сообщение об успешном выполнении операторов сценария (рис. 15).

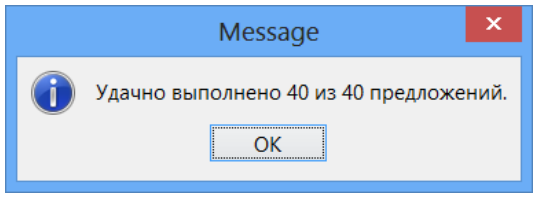


Рис. 15 Сообщение о выполнении команд файла

Задание.

Создать схему (таблицы и другие объекты) в целевой БД для модели данных, созданной на практическом занятии Физическое проектирование БД.

Порядок выполнения работы

1. Изучить теоретический материал
2. Создать таблицы БД в соответствии с индивидуальным заданием

Содержание отчета

1. Скриншоты выполнения задания
2. Схема физической модели данных

Практическое занятие № 35
Загрузка и выгрузка данных БД PostgreSQL

Цель занятия. Научиться подготавливать и загружать данные в базы данных PostgreSQL, выгружать данные из баз данных PostgreSQL.

Краткие теоретические сведения

Загрузка данных из текстовых файлов

Загрузку и выгрузку данных будем рассматривать на примере БД продаж, созданной на предыдущих занятиях (рис. 1).

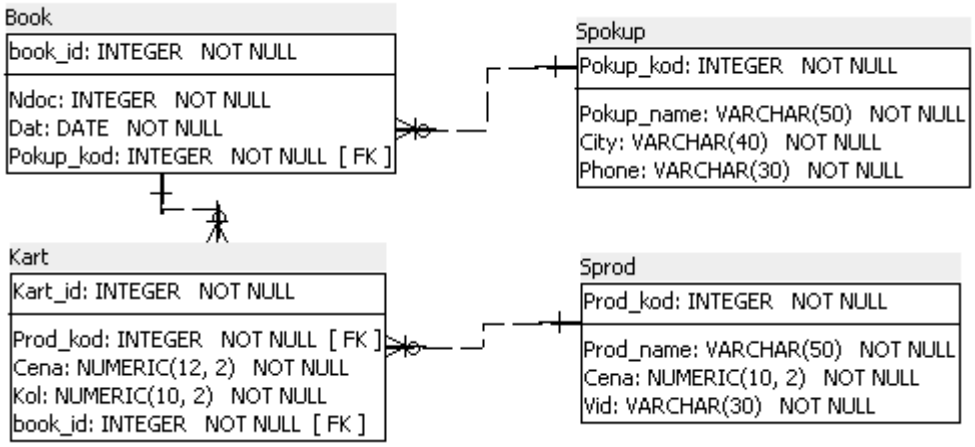


Рис. 1 База данных Продажи

Последовательность загрузки данных должна быть следующая. Вначале загружаются данные в таблицы, не имеющие внешних ключей, затем в таблицы, которые ссылаются на таблицы с уже загруженными данными и т.д.

Для рассматриваемой базы данных Продаж последовательность загрузки должна быть следующей:

- Продукция (таблица Sprod);
- Покупатель (таблица Spokup);
- Книга продаж (таблица Book);
- Ассортимент продаж (таблица Kart).

Данные могут быть подготовлены в текстовых файлах и файлах формата CSV.

Файлы CSV содержат данные в текстовом формате, разделенные запятыми. Одна строка — это одна запись об одном объекте. Однотипные свойства в каждой строке занимают фиксированные места. Некоторые значения могут быть пропущены. В месте пропуска будут подряд следовать символы разделители. В первой строке файла могут быть указаны имена свойств объекта, разделенные запятыми. Имена свойств обычно соответствуют именам колонок таблицы БД. Файлы формата удобно подготавливать и просматривать в Microsoft Excel.

Подготовка данных в Excel

Откройте программу Microsoft Excel и введите данные для таблиц (рис.2-5). Обратите внимание на правильность заполнения внешних ключей.

	A	B	C	D
1	**Код**	**Наименование**	**Цена**	**Вид продукции**
2	1	Крем детский	25.50	крем
3	2	Крем дневной Наташа	55.00	крем
4	3	Лифтинг-крем Диамант	110.50	крем
5	4	Крем ночной	95.00	крем
6	5	Мыло детское	26.00	мыло
7	6	Мыло Алиса	34.00	мыло

Рис. 2 Данные для таблицы sprod (Продукция)

	A	B	C	D
1	**Код**	**Наименование**	**Город**	**Телефон**
2	1	ООО Банг и Бонсомер	Москва	8 (495) 255-45-31
3	2	АО Аэлита	Москва	8 (499) 124-10-66
4	3	АО ТД Империал	Тверь	8 (418) 245-86-11
5	4	ООО Буревестник	Тула	8 (405) 165-41-62
6	5	ООО Ортекс	Тверь	8 (418) 134-54-12
7	6	АО Косметик рус	Москва	8 (499) 176-11-75

Рис. 3 Данные для таблицы spokup (Покупатели)

	A	B	C	D
1	**Ид-р продажи**	**Номер документа**	**Дата продажи**	**Код покупателя**
2	1	10	30-01-2021	1
3	2	15	31-01-2021	3
4	3	21	06-02-2021	2
5	4	30	15-02-2021	1
6	5	35	25-02-2021	4
7	6	40	01-03-2021	

Рис. 4 Данные для таблицы book (Книга продаж)

	A	B	C	D	E
1	**Ид-р ассортимета**	**Код продукции**	**Цена**	**Количество**	**Ид-р продажи**
2	1	1	25.50	10	1
3	2	2	55.00	16	1
4	3	1	25.50	12	2
5	4	3	110.50	5	2
6	5	2	55.00	11	3
7	6	3	110.50	15	3
8	7	4	95.00	8	3
9	8	2	55.00	14	4
10	9	1	25.50	6	5
11	10	5	26.00	16	5
12	11	3	34.00	25	6

Рис. 5 Данные для таблицы kart (Ассортимент продаж)

Введенные данные необходимо сохранить в формате CSV (текстовые данные с разделителями значений). Для этого необходимо выполнить команду меню

Сохранить как - Другие форматы

В открывшемся окне необходимо выбрать формат сохранения CSV, а имя файла на английском (рис.6). Для удобства выберем имя, совпадающее с именем таблицы, для которой данные предназначены.

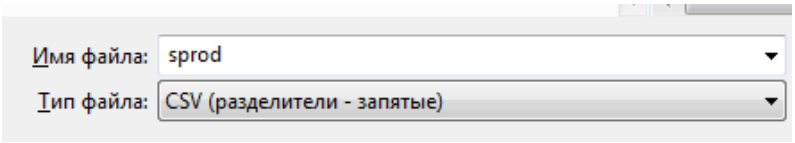


Рис. 6 Сохранение данных в формате CSV

Откроем один из сохраненных файлов в текстовом редакторе (рис.7). Первая строка содержит название атрибутов. Это надо учитывать при загрузке данных в таблицы. Разделителем является символ «точка с запятой».

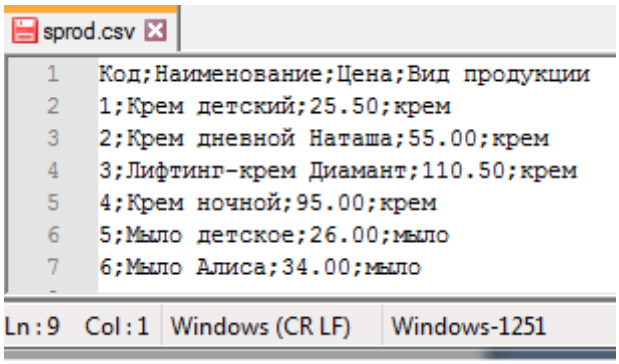


Рис. 7 Данные в формате CSV

Загрузка данных в таблицы БД

Для загрузки данных из подготовленных файлов в таблицы базы данных используется команда COPY, имеющая следующую структуру (в квадратные скобки заключается элемент, который может быть опущен):

```
COPY имя_таблицы [ ( имя_столбца , ... ) ]
    FROM 'имя_файла'
    [параметр , ...]
```

В команде могут быть использованы *параметры (ниже указаны основные)*:

имя_формата данных (по умолчанию txt). Часто используется формат CSV (нужно указывать явно);

DELIMITER '*символ_разделитель*' – символ-разделитель значений полей (для CSV по умолчанию запятая).

Если текстовые строки содержат запятые, то в качестве разделителя можно использовать символы ';' или '|';

NULL - '*маркер_NULL*' определяет строку, задающую значение NULL. По умолчанию в текстовом формате это \N (обратная косая черта и N), а в формате CSV — пустая строка без кавычек;

HEADER [*boolean*] - Указывает, что файл в 1-й строке содержит строку заголовка с именами столбцов. Этот параметр допускается только для формата CSV. При отсутствии значения *boolean* подразумевается *true;*

QUOTE '*символ_кавычек*' - Указывает символ кавычек, используемый для заключения значений в кавычки. Этот параметр поддерживается только для формата CSV;

ENCODING '*имя_кодировки*' - Указывает, что файл имеет заданную, *например* WIN1251 .

Для выполнения команды необходимо открыть окно ввода SQL-запросов Пример записи команды загрузки в таблицу `sprod` из файла `sprod.csv`, находящегося в каталоге `C:/data`:

```
SET SEARCH_PATH TO  prod; //  Установка текущей схемы
COPY sprod
   FROM 'C:/data/sprod.csv' CSV
DELIMITER ';' // разделитель «точка с запятой»
HEADER;  // первая строка содержит имена атрибутов
```

Если в файле использована кодировка 1251, а в БД используется кодировка UTF8, то при экспорте будет выведено сообщение об ошибке:

ОШИБКА: неверная последовательность байт для кодировки "UTF8"

В этом случае надо указать кодировку исходного файла в параметре:

```
ENCODING 'WIN1251'
```

И команда загрузки будет такой:

```
COPY sprod
   FROM 'c:/data/sprod.csv' CSV
DELIMITER ';'
HEADER
ENCODING 'WIN1251';
```

Если все данные соответствуют тем типам, которые определены в таблице БД, то не будет ошибок загрузки данных. Если же ошибки появляются, необходимо скорректировать подготовленные данные.

Полный пакет команд загрузки данных в базу данных Продажи (последовательность команд должна быть именно такой):

```
SET SEARCH_PATH TO  prod; --  Установка текущей схемы
COPY sprod FROM 'c:/data/sprod.csv' CSV
   DELIMITER ';' HEADER ENCODING 'WIN1251';
COPY spokup FROM 'c:/data/spokup.csv' CSV
   DELIMITER ';' HEADER ENCODING 'WIN1251';
COPY book FROM 'c:/data/book.csv' CSV
   DELIMITER ';' HEADER ENCODING 'WIN1251';
COPY kart FROM 'c:/data/kart.csv' CSV
```

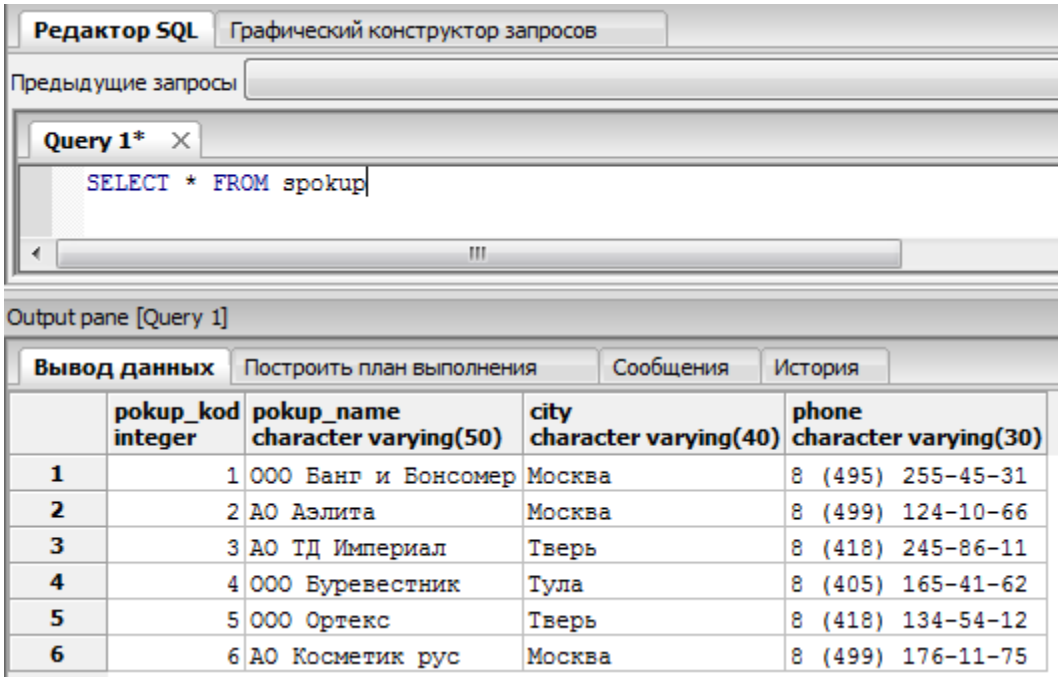
DELIMITER ';' HEADER ENCODING 'WIN1251';

Замечание. Путь к файлам рассматривается относительно сервера. Поэтому при подключении к удаленному серверу необходимо использовать сетевое имя. Для сервера БД на локальном компьютере задаем обычные пути к файлу.

Можно проверить загрузку данных, выполнив запрос к таблице spokup (Покупатели):

SELECT * FROM prod.spokup

Результат запроса приведен на рис.9



	pokup_kod integer	pokup_name character varying(50)	city character varying(40)	phone character varying(30)
1	1	ООО Банг и Бонсомер	Москва	8 (495) 255-45-31
2	2	АО Аэлита	Москва	8 (499) 124-10-66
3	3	АО ТД Империал	Тверь	8 (418) 245-86-11
4	4	ООО Буревестник	Тула	8 (405) 165-41-62
5	5	ООО Ортекс	Тверь	8 (418) 134-54-12
6	6	АО Косметик рус	Москва	8 (499) 176-11-75

Рис. 9 Результат запроса выборки данных

При загрузке данных мы явно задавали значения суррогатным первичным ключам (идентификаторам), которые при добавлении данных должны заполняться с использованием последовательностей. Чтобы механизм автоувеличения далее работал корректно необходимо изменить текущие значения последовательностей на последнее значение, заданное в файлах данных. Это выполняется оператором:

```
setval (имя_последовательности, последнее_значение, true)
```

Максимальные значения первичного ключа в таблицах (по файлам данных):

book: book_id=5

kart: kart_id=10

Поэтому для изменения текущего значения последовательностей ждя этих колонок необходимо выполнить операторы:

SELECT setval('prod.book_seq', 5, true);

SELECT setval('prod.kart_seq', 10, true);

Выгрузка данных в файл

Можно не только загружать данные в таблице, но также и выгружать во внешние файлы. Это также выполняется командой COPY:

```
COPY имя_таблицы TO 'путь и имя файла' [параметры];
```

Необязательные параметры такие же, как и при загрузке данных. Для примера выведем данные из таблицы Покупатель (spokup) в формате CSV с названием колонок в первой строке, с разделителем «запятая», в кодировки WIN-1251 в файл c:\data\spokup.csv

```
COPY prod.spokup TO 'c:/data/spokup.csv' DELIMITER ','
        CSV HEADER ENCODING 'WIN1251';
```

Содержимое файла в текстовом редакторе приведено на рис. 10.

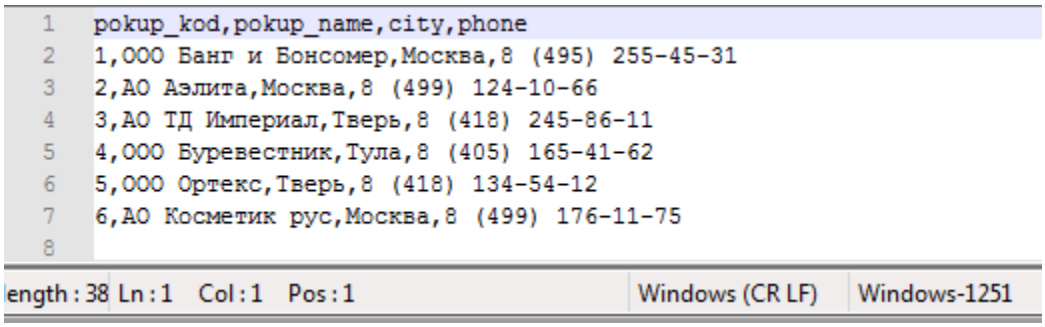


Рис. 10 Результат выгрузки данных

Задание. Для базы данных созданной на предыдущем практическом занятии:

1) Подготовить файлы данных
2) Выполнить загрузку данных в таблицы БД из внешних файлов
3) Выполнить выгрузку данных из таблиц БД во внешний файл

Порядок выполнения работы

1. Изучить теоретический материал
2.Подготовить файлы данных в формате CSV
3. Загрузить данные из внешних файлов в таблицы базы данных
4. Выгрузить данные из таблиц базы данных во внешние файлы

Содержание отчета

1. Подготовленные данные в текстовом виде
2. Команды загрузки данных
3. Запросы проверки загрузки данных
4. Команды выгрузки данных
5. Содержимое выгруженных файлов

Практическое занятие № 36
Язык SQL. Оператор запроса SELECT

Цель занятия. Научиться составлять оператор запроса к таблицам базы данных, использовать предикаты для отбора нужных данных, выполнять группировку данных в базе данных PostgreSQL.

Краткие теоретические сведения

Стандартным языком доступа к базам данных является язык SQL(Structured Query Language). Все реляционные СУБД, используют тот или иной диалект SQL. Теоретической основой языка SQL являются реляционная алгебра и реляционное исчисление.

Базовые операторы SQL по функциональному признаку можно разделить на группы:

- Язык определения данных DDL(Data Definition Language); DDL включает команды, которые создают объекты данных (таблицы, индексы, представления, и т.д.). Совокупность всех объектов БД – каталог БД или ее структура;
- Язык манипулирования данными DML (Data Manipulation Language); DML включает команды заполнения БД и формирования запросов к БД;
- Язык управления данными DCL (Data Control Language) – организация доступа к БД (средства определения прав пользователей на выполнение определенных действий над данными: операторы GRANT и REVOKE);
- Язык управления транзакциями TCL (Transaction Control Language) COMMIT|ROLLBACK

Основными операторами языка SQL для манипулирования данными (DML) являются:

- оператор запроса (SELECT)
- операторы обновления (INSERT, UPDATE, DELETE)

Оператор запроса SELECT

Оператор SELECT является фактически самым важным для пользователя и самым сложным оператором SQL. Оператор SELECT *предназначен для выборки данных из таблиц*, т.е. он реализует одно их *основных назначений* базы данных - *предоставлять информацию* пользователю.

Оператор SELECT всегда выполняется над некоторыми таблицами, входящими в базу данных (постоянно хранимыми в БД, временными таблицами и представлениями). *Результатом выполнения* оператора SELECT всегда является *таблица*.

Запрос может включать подзапросы, т.е. допускается вложение операторов запроса.

Общий вид оператора SELECT (в квадратных скобках указаны необязательные элементы):

SELECT [ALL|DISTINCT] <список выборки>
<табличное выражение>
[<ORDER BY раздел>]

где

< **табличное выражение** > **::= <FROM раздел>**
[<WHERE раздел>]
[<GROUP BY раздел>]
[<HAVING раздел>]

В определении запроса задается:

1) **список выборки** (список арифметических выражений, включающих столбцы табличного выражения и константы). Элементы списка разделяются запятыми. В частном случае можно задать символ * - в результат включаются все элементы табличного выражения.
   Пример. Выбрать все данные из справочника продукции
   SELECT * FROM Sprod
2) **Ключевые слова ALL или DISTINCT**. При наличии ключевого слова DISTINCT из таблицы-результата, удаляются строки-дубликаты; при указании ALL (или отсутствии слова) удаление строк-дубликатов не производится.
3) **Табличное выражение** формируется на основе последовательного применения разделов FROM, WHERE, GROUP BY и HAVING из таблиц, заданных в разделе FROM. FROM - единственный обязательный раздел. В результирующей таблице порядок следования строк не определен и среди строк могут находиться дубликаты.
4) **Раздел ORDER BY** используется для упорядочения строк результата.

Рассмотрим структуру и назначение разделов табличного выражения.

При рассмотрении оператора SQL в примерах будем использовать БД Продажи (рис.1), с заполненными таблицами (рис.2-5).

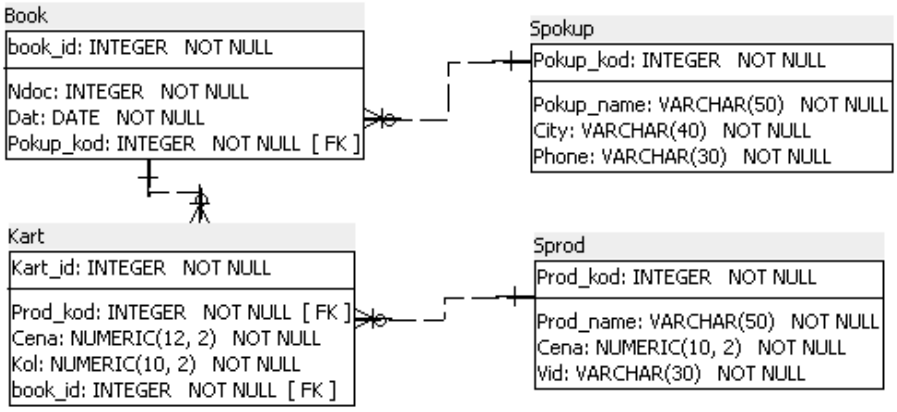


Рис. 1 База данных Продажи

Код	Наименование	Цена	Вид продукции
1	Крем детский	25.50	крем
2	Крем дневной Наташа	55.00	крем
3	Лифтинг-крем Диамант	110.50	крем
4	Крем ночной	95.00	крем
5	Мыло детское	26.00	мыло
6	Мыло Алиса	34.00	мыло
prod_kod	prod_name	cena	vid

Рис. 2 Данные для таблицы sprod (Продукция)

Код	Наименование	Город	Телефон
1	ООО Банг и Бонсомер	Москва	8 (495) 255-45-31
2	АО Аэлита	Москва	8 (499) 124-10-66
3	АО ТД Империал	Тверь	8 (418) 245-86-11
4	ООО Буревестник	Тула	8 (405) 165-41-62
5	ООО Ортекс	Тверь	8 (418) 134-54-12
6	АО Косметик рус	Москва	8 (499) 176-11-75
pokup_kod	pokup_name	city	phone

Рис. 3 Данные для таблицы spokup (Покупатели)

Ид-р продажи	Номер документа	Дата продажи	Код покупателя
1	10	30-01-2021	1
2	15	31-01-2021	3
3	21	06-02-2021	2
4	30	15-02-2021	1
5	35	25-02-2021	4
6	40	01-03-2021	
book_id	ndoc	dat	pokup_kod

Рис. 4 Данные для таблицы book (Книга продаж)

Ид-р ассортимета	Код продукции	Цена	Количество	Ид-р продажи
1	1	25.50	10	1
2	2	55.00	16	1
3	1	25.50	12	2
4	3	110.50	5	2
5	2	55.00	11	3
6	3	110.50	15	3
7	4	95.00	8	3
8	2	55.00	14	4
9	1	25.50	6	5
10	5	26.00	16	5
11	3	34.00	25	6
kart_id	prod_kod	cena	kol	book_id

Рис. 5 Данные для таблицы kart (Ассортимент продаж)

Раздел FROM

Раздел FROM задает таблицы-источники и имеет следующий синтаксис:

FROM <список таблиц>

где

<таблица>::= <имя таблицы> [<алиас>]

Рядом с именем таблицы можно указывать еще одно имя – **алиас (псевдоним)** таблицы. Алиас (псевдоним) вводится для использования в других разделах оператора SELECT в качестве префикса перед именем столбца и отделяются от имени столбца точкой. Алиасы используются:

а) Обязательно, если в разделе FROM используются копии одной и той же таблицы,

б) Необязательно - для замены длинных имен таблиц.

Результатом выполнения раздела FROM является РАСШИРЕННОЕ ДЕКАРТОВО ПРОИЗВЕДЕНИЕ таблиц-источников.

Если табличное выражение содержит только раздел FROM, то результат - это вся таблица или декартово произведение таблиц, т.е каждая строка одной таблицы сцепляется (приписывается справа) с каждой строкой второй таблицы.

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a_1 x_1 \\ a_1 x_2 \\ a_1 x_3 \\ a_2 x_1 \\ a_2 x_2 \\ a_2 x_3 \end{pmatrix}$$

Пример 1. SELECT prod_kod, prod_name FROM sprod

Из всех записей таблицы SPROD выбираются код продукции и наименование продукции.

Результат запроса*:*

	prod_kod integer	prod_name character varying(50)
1	1	Крем детский
2	2	Крем дневной Наташа
3	3	Лифтинг-крем Диамант
4	4	Крем ночной
5	5	Мыло детское
6	6	Мыло Алиса

Пример 2. SELECT s.pokup_kod kod1,d.pokup_kod kod2, ndoc
FROM spokup s, book b

Вычисляется декартово произведение (каждая строка таблицы SPROD сцепляется с каждой строкой таблицы BOOK)

Выбираются: код покупателя (kod1) из таблицы Покупатель (spokup), код покупателя (kod2) и номер документа из таблицы Книга продаж из всех полученных строк прямого произведения таблиц.

Первые 11 строк результата:

Вывод данных | Построить план

	kod1 integer	kod2 integer	ndoc integer
1	1	1	10
2	2	1	10
3	3	1	10
4	4	1	10
5	5	1	10
6	1	3	15
7	2	3	15
8	3	3	15
9	4	3	15
10	5	3	15
11	1	2	21

Упражнение 1. Вывести наименование и адрес всех покупателей.

Раздел WHERE

Из результатов запроса последнего примера видно, что выбрать данные из нескольких связанных таблиц, используя только раздел FROM невозможно, т.к. получается много бессмысленных соединенных строк. Здесь на помощь приходит раздел WHERE.

Раздел WHERE задает условия отбора данных. Синтаксис раздела:

```
WHERE <условие отбора>
```

В качестве условия в разделе WHERE можно использовать сложные логические выражения, включающие:

- поля таблиц, константы, выражения;
- операторы условия выборки (предикаты);
- скобки;
- логические операторы AND (и), OR (или), NOT (не);
- подзапросы (select …).

Условие отбора применяется ко всем строкам таблицы-результата, полученным после применения раздела FROM. В конечный результат оператора запроса попадают те строки, для которых условие дает «истину» (true).

Поскольку SQL допускает наличие в базе данных неопределенных значений, то вычисление условия производится не в булевой (двухзначной), а в трехзначной логике со значениями True, False и Unknown (null - неизвестно). Булевские операции AND, OR и NOT работают в трехзначной

логике по следующим правилам (true - обозначим 1, false - 0, null - un) (таблица 1).

Таблица 1 - Таблицы истинности в трехзначной логике

A	B	A and B	A or B	not A
0	0	0	0	1
0	1	0	1	
0	un	0	un	
1	0	0	1	0
1	1	1	1	
1	un	un	1	
un	0	0	un	un
un	1	un	1	
un	un	un	un	

В разделе WHERE используются следующие ***операторы условия отбора (предикаты)***:

- простые сравнения (>, <, = и т.д.),
- between,
- in,
- like,
- is null,
- одноместные операторы exists, all, any

Рассмотрим основные операторы условий отбора (предикаты)

1) Предикат сравнения реализует простое сравнение

<выражение1> {= | < | > | <= | >= | <>}
{<выражение2> | <подзапрос>}

Выражение1 и *выражение2* могут включать в общем случае имена столбцов таблиц из раздела FROM и константы.

При сравнении с подзапросом мощность результата подзапроса (количество возвращаемых строк) должна быть не более 1.

Значение оператора сравнения равно ***unknown*** если:

- выражение1 и/или выражение2 имеют неопределенное значение;
- подзапрос возвращает пустой результат.

Значение арифметического выражения не определено ***(unknown)***, если хотя бы один операнд имеет неопределенное значение.

Рассмотрим примеры. В предыдущих примерах вначале задавался оператор запроса, а затем давались словесные формулировки. Теперь сделаем наоборот: вначале дадим текстовую формулировку, а затем оператор запроса.

Пример 3. Выбрать наименования покупателей, получивших продукцию в феврале 2021 года.

Пояснение: Для выполнения запроса нам потребуются таблицы:

1) Покупатель (spokup), т.к. нам нужно вывести наименование покупателей;
2) Книга продаж (book), т.к. в результат должны попасть покупатели, которые получили продукцию в заданном периоде времени (феврале 2021 г.).

Оператор запроса:

```
SELECT  DISTINCT   sp.pokup_name
      FROM  book b, spokup sp
      WHERE b.pokup_kod=sp.pokup_kod  AND
            b.dat>='01-02-2021'AND  b.dat<'01-03-2021'
```

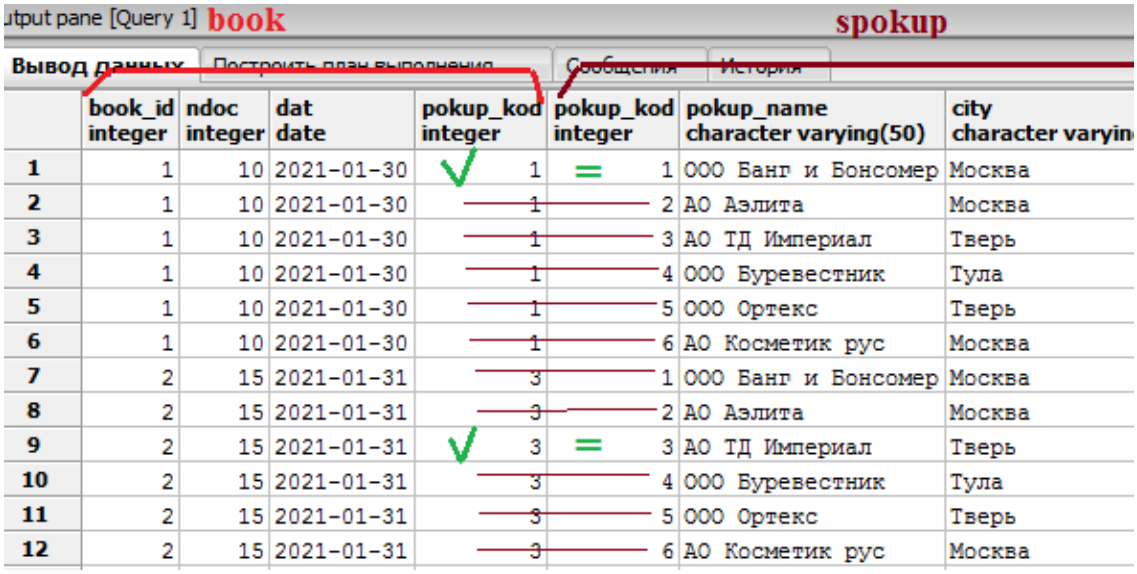
Алгоритм выполнения.

1) Вычисляется декартово произведение таблиц: каждая строка таблицы *Spokup* сцепляется с каждой строкой таблицы *Book*. Ниже показаны первые 12 строк.

	book_id integer	ndoc integer	dat date	pokup_kod integer	pokup_kod integer	pokup_name character varying(50)	city character varying(40)	phone character varying(30)
1	1	10	2021-01-30	1	1	ООО Банг и Бонсомер	Москва	8 (495) 255-45-31
2	1	10	2021-01-30	1	2	АО Аэлита	Москва	8 (499) 124-10-66
3	1	10	2021-01-30	1	3	АО ТД Империал	Тверь	8 (418) 245-86-11
4	1	10	2021-01-30	1	4	ООО Буревестник	Тула	8 (405) 165-41-62
5	1	10	2021-01-30	1	5	ООО Ортекс	Тверь	8 (418) 134-54-12
6	1	10	2021-01-30	1	6	АО Косметик рус	Москва	8 (499) 176-11-75
7	2	15	2021-01-31	3	1	ООО Банг и Бонсомер	Москва	8 (495) 255-45-31
8	2	15	2021-01-31	3	2	АО Аэлита	Москва	8 (499) 124-10-66
9	2	15	2021-01-31	3	3	АО ТД Империал	Тверь	8 (418) 245-86-11
10	2	15	2021-01-31	3	4	ООО Буревестник	Тула	8 (405) 165-41-62
11	2	15	2021-01-31	3	5	ООО Ортекс	Тверь	8 (418) 134-54-12
12	2	15	2021-01-31	3	6	АО Косметик рус	Москва	8 (499) 176-11-75

2) К каждой строке произведения таблиц применяется условие отбора раздела WHERE и в результат запроса отбираются строки, для которых условие дает «истину».

   Первый предикат сравнения (sp.pokup_kod=b.pokup_kod) не задан конкретным условием самого запроса, а выполняет ***естественное соединение*** строк по колонке первичного ключа *Код покупателя* в таблице Покупатель (spokup) и внешнего ключа *Код покупателя* в таблице Книга продаж (book). Строки с несовпадающими кодами будут отброшены.



	book_id integer	ndoc integer	dat date	pokup_kod integer	pokup_kod integer	pokup_name character varying(50)	city character varyin
1	1	10	2021-01-30	1	1	ООО Банг и Бонсомер	Москва
2	1	10	2021-01-30	1	2	АО Аэлита	Москва
3	1	10	2021-01-30	1	3	АО ТД Империал	Тверь
4	1	10	2021-01-30	1	4	ООО Буревестник	Тула
5	1	10	2021-01-30	1	5	ООО Ортекс	Тверь
6	1	10	2021-01-30	1	6	АО Косметик рус	Москва
7	2	15	2021-01-31	3	1	ООО Банг и Бонсомер	Москва
8	2	15	2021-01-31	3	2	АО Аэлита	Москва
9	2	15	2021-01-31	3	3	АО ТД Империал	Тверь
10	2	15	2021-01-31	3	4	ООО Буревестник	Тула
11	2	15	2021-01-31	3	5	ООО Ортекс	Тверь
12	2	15	2021-01-31	3	6	АО Косметик рус	Москва

Результат выполнения первого предиката сравнения

	book_id integer	ndoc integer	dat date	pokup_kod integer	pokup_kod integer	pokup_name character varying(50)	city character va
1	1	10	2021-01-30	1	1	ООО Банг и Бонсомер	Москва
2	2	15	2021-01-31	3	3	АО ТД Империал	Тверь
3	3	21	2021-02-06	2	2	АО Аэлита	Москва
4	4	30	2021-02-15	1	1	ООО Банг и Бонсомер	Москва
5	5	35	2021-02-25	4	4	ООО Буревестник	Тула

На следующих занятиях мы рассмотрим подробнее операцию соединения.

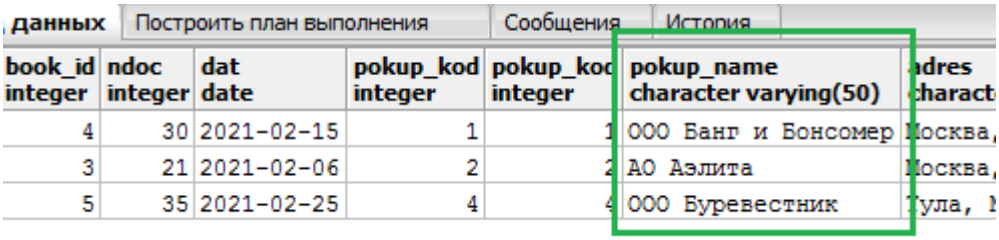
Два следующих предиката сравнения (b.dat>='01-02-2021' и b.dat<'01-03-2021') определяются условиями запроса и проверяют попадание продажи в заданный период времени.

book_id integer	ndoc integer	dat date	pokup_kod integer	pokup_kod integer	pokup charac
1	10	2021-01-30	1	1	ООО Б
2	15	2021-01-31	3	3	АО ТД
3	21	2021-02-06	2	2	АО Аэ
4	30	2021-02-15	1	1	ООО Б
5	35	2021-02-25	4	4	ООО Б

Результат выполнения предикатов сравнения даты с заданным периодом

	book_id integer	ndoc integer	dat date	pokup_kod integer	pokup_kod integer	pokup_name character varying(50)	city character varying(40)	phone characte
1	4	30	2021-02-15	1	1	ООО Банг и Бонсомер	Москва	8 (495)
2	3	21	2021-02-06	2	2	АО Аэлита	Москва	8 (499)
3	5	35	2021-02-25	4	4	ООО Буревестник	Тула	8 (405)

3) Из полученных строк отбирается колонка *Наименование покупателя*.



данных | Построить план выполнения | Сообщения | История

book_id integer	ndoc integer	dat date	pokup_kod integer	pokup_kod integer	pokup_name character varying(50)	adres charact
4	30	2021-02-15	1	1	ООО Банг и Бонсомер	Москва,
3	21	2021-02-06	2	2	АО Аэлита	Москва,
5	35	2021-02-25	4	4	ООО Буревестник	Тула,

4) Дублированные строки отбрасываются. В данном примере повторений нет.

Результат запроса:

	pokup_name character varying(50)
1	ООО Банг и Бонсомер
2	АО Аэлита
3	ООО Буревестник

Пример 4 (сравнение с результатом подзапроса). Выбрать наименование продукции, которая отпускалась покупателю «ООО Банг и Бонсомер».

Пояснение: Для выполнения запроса нам потребуются таблицы:

1) Продукция (sprod), т.к. нам нужно вывести наименование продукции;
2) Ассортимент продаж (kart), т.к. в результат должна попасть продукция, которая продавалась;
3) Книга продаж (book), т.к. необходимо иметь информацию о покупателе;
4) Покупатель (spokup), т.к. нас интересуют только продажи заданному покупателю «ООО Банг и Бонсомер»;

Первые 3 таблицы будем соединять в запросе, а четвертую будем включать в подзапрос.

Подзапрос можно использовать, если он возвращает не более 1 значения. Это гарантируется тем, что в таблице Spokup имеется ограничение уникальности по колонке *Наименование покупателя* (pokup_name).

Оператор запроса:

```
SELECT  DISTINCT  prod_name
      FROM  sprod s, kart k, book as b
      WHERE s.prod_kod =k.prod_kod -- Соединение таблиц Sprod и Kart
        AND k.book_id=b.book_id       -- Соединение таблиц Kart и Book
        AND pokup_kod = (SELECT pokup_kod FROM spokup
              WHERE pokup_name='ООО Банг и Бонсомер')
```

Алгоритм выполнения.

1) Вычисляется декартово произведение таблиц: каждая строка таблицы *Sprod* сцепляется с каждой строкой таблицы *Kart*, а затем с каждой строкой таблицы *Book*.
2) Выполняется подзапрос: определяется и запоминается код покупателя 'ООО Банг и Бонсомер'
3) К каждой строке произведения таблиц применяется условие отбора раздела WHERE и в результат запроса отбираются строки, для которых условие дает «истину».
   Первый предикат сравнения (s.prod_kod =k.prod_kod) выполняет ***естественное соединение*** строк таблиц *Sprod* и *Kart* по общей колонке *Код продукции*.
   Второй предикат сравнения (k.book_id=b.book_id) выполняет ***естественное соединение*** строк *Kart* и *Book* по общей колонке *Идентификатор продажи*.
   Строки с несовпадающими кодами будут отброшены.
   Последний предикат выполняет сравнение кода покупателя продажи с полученным ранее кодом покупателя 'ООО Банг и Бонсомер', полученного подзапросом.
4) Из полученных строк отбирается колонка *Наименование продукции*.
5) Дублированные строки отбрасываются.

Результат запроса:

Вывод данных | Построить пла

	prod_name character varying(50)
1	Крем детский
2	Крем дневной Наташа

Упражнение 2. Выбрать всю продукцию из накладной с заданным номером и датой. Вывести код и наименование продукции, цену, количество и стоимость каждой позиции накладной.
Пояснение: 1) *Стоимость* определяется выражением, равным произведению колонок *Цена* и *Количество*.
2) Для округления значения до заданного количества разрядов используют функцию CAST («выражение» as «тип»), где типом можно выбрать NUMERIC (10,2)

2) Предикат between проверяет принадлежность колонки (выражения) заданному диапазону.

<колонка> [NOT] BETWEEN <значение1> AND <значение2>

Предикат возвращает истину, если значения «колонки» находятся в диапазоне от «значения1» до «значения2». Колонка и значения должны быть одного типа

Пример 5. Вывести продажи, объем которых (колонка *Количество*) был в диапазоне от 10 до 20.

```
SELECT * FROM kart WHERE kol BETWEEN 10 AND 20
```

Результат запроса:

	kart_id integer	prod_kod integer	cena numeric(12,2)	kol numeric(10,2)	book_id integer
1	1	1	25.50	10.00	1
2	2	2	55.00	16.00	1
3	3	1	25.50	12.00	2
4	5	2	55.00	11.00	3
5	6	3	110.50	15.00	3
6	8	2	55.00	14.00	4
7	10	5	87.00	16.00	5

Упражнение 3. Выбрать продукцию, представленную в накладных за январь 2021 г. Вывести код и наименование продукции, цену, количество и стоимость каждой позиции накладной.

3) Предикат in проверяет принадлежность списку.

<выражение> [NOT] IN {<подзапрос> | (<список значений>)}

Сравниваемое выражение должно быть равно одному из элементов списка. Если используется **NOT IN**, то проверяется не принадлежность диапазону.

Типы левого операнда и значений из списка правого операнда должны быть сравнимыми. Подзапрос должен возвращать только один столбец.

Значение оператора равно:

- true в том случае, когда значение левого операнда совпадает хотя бы с одним значением списка правого операнда;
- false - если список правого операнда пуст (так может быть, если правый операнд задается подзапросом), или значение левого операнда не совпадает ни с одним из элементов списка правого операнда;
- unknown - в других случаях.
-

Пример 6. Отобрать все продажи с кодом продукции '1' или '2'

Вариант запроса *с предикатами сравнения*:

```
SELECT prod_kod, cena, kol
    FROM kart
    WHERE prod_kod =1 OR prod_kod =2
```

Вариант запроса *с предикатом in* будет более кратким:

```
SELECT prod_kod, cena, kol
    FROM kart
    WHERE prod_kod  in (1,2)
```

Результат:

Вывод данных | Построить план выполнения

	prod_kod integer	cena numeric(12,2)	kol numeric(10,2)
1	1	25.50	10.00
2	2	55.00	16.00
3	1	25.50	12.00
4	2	55.00	11.00
5	2	55.00	14.00
6	1	25.50	6.00

Пример 7. Вывести номера документов, в которых не было продукции с кодом '1'

```
SELECT   ndoc
      FROM  Book
      WHERE  book_id NOT IN (SELECT DISTINCT book_id
            FROM kart  WHERE  prod_kod = 1)
```

В данном запросе список для сравнения получается подзапросом.
Результат:

Вывод данных

	ndoc integer
1	21
2	30

Упражнение. Использовать предикат in с фиксированным списком и списком, полученным запросом.

1. Вывести наименования покупателей, которые получали продукцию с кодом 3 или 4.
2. Вывести наименования продукции, которая продавалась в январе 2021 г.

4) Предикат like используется для **поиска подстрок в столбцах символьного типа**. Он проверяет, совпадает ли часть текста с заданной строкой-шаблоном

<колонка> [NOT] LIKE <шаблон>

В шаблоне можно использовать 2 символа, обозначающие «любой текст»:

- символ подчеркивания ("_") обозначает любой одиночный символ;
- символ процента ("%") обозначает последовательность (в том числе пустую) произвольного количества любых.

*Пример 8.*Вывести покупателей, наименования которых начинаются на «АО»

```
SELECT pokup_name FROM Spokup
      WHERE Pokup_name LIKE 'АО%'
```

При этом шаблоны 'АО' или 'АО_' вернут неверный результат.
Результат запроса:

	pokup_name character varying(50)
1	АО Аэлита
2	АО ТД Империал
3	АО Косметик рус

Упражнение 5. Получить список продукции, в названии которых есть слово «крем».

Пояснение: Поиск является регистрозависимым, т.е. «Крем» и «крем» - это разные строки. Чтобы сделать поиск независимым от регистра, надо строки перевести в верхний регистр функцией UPPER(строка), а константу записать ПРОПИСНЫМИ символами.

5) Предикат IS NULL позволяет проверить незаполненность колонки

Null-значения нельзя сравнивать с другими значениями, даже если они Null. Поэтому, для того, чтобы отобрать строки, в которых некоторый столбец не определен, используется специальный оператор сравнения с Null:

<колонка> IS [NOT] NULL

Этот оператор всегда принимает значения true или false. При этом значение "x IS NULL" равно true тогда и только тогда, когда значение x не определено.

В данной ситуации нельзя использовать простое сравнение:

x=NULL или x<> NULL

Упражнение 6. Вывести наименование покупателей, у которых не заполнен телефон.

6) Операторы подзапросов Exists,All, Any относятся к специальным одноместные операторы, аргументом которых является подзапрос.

Оператор exists используется для определения наличия данных в результате подзапроса.

EXISTS <подзапрос>

Значение этого предиката равно true , когда результат вычисления подзапроса не пуст, иначе false (результата ***unknown*** – не может быть).

Пример 9. Получить перечень покупателей, которые купили продукцию с кодом 1

```
SELECT  pokup_name FROM spokup sp
      WHERE EXISTS (SELECT 1 FROM book b, kart k
            WHERE b.pokup_kod=sp.pokup_kod
                  and b.book_id=k.book_id
                  and k.prod_kod=1)
```

В операторе подзапроса на месте списки выборки можно поставить любое выражение: *, любую колонку или константу. Проверяется только наличие строк результата и не важно, что возвращает подзапрос.
Результат запроса:

	pokup_name character varying(50)
1	ООО Буревестник
2	ООО Банг и Бонсомер
3	АО ТД Империал

Упражнение 7. Вывести продукцию, которая не продавалась в январе 2021 г.

Операторы с квантором *существования или всеобщности* имеет следующий синтаксис:
<выражение> {= | < | > | <= | >= | <>} <квантор> <подзапрос>

где <квантор> ::= ALL | ANY (синоним SOME)
Пример KOL > ALL(select KOL FROM …)
Обозначим через
X - результат вычисления арифметического выражения левой части предиката
S - результат вычисления подзапроса.
Предикат всеобщности ***"X <оператор> ALL S"*** имеет значение:
- true, если S пусто или значение предиката "x <оператор> s" равно true для каждой строки, входящей в S.
- false, если значение предиката "x <оператор> s" равно false хотя бы для одной строки, входящей в S.
- unknown - в остальных случаях.

Предикат существования ***"X < оператор > ANY S"*** имеет значение
- false, если S пусто или значение предиката "x < оператор > s" равно false для каждой строки, входящей в S.
- true, если значение предиката "x < оператор > s" равно true хотя бы для одной строки, `входящей в S.
- unknown - в остальных случаях.

Пример 10. Вывести покупателей, которые купили продукцию, относящуюся к виду «Пена»

```
SELECT  DISTINCT pokup_name FROM spokup s, book b, kart k
    WHERE b.pokup_kod=s.pokup_kod
        and b.book_id=k.book_id
        and prod_kod =ANY(SELECT prod_kod FROM sprod
        WHERE vid='Пена')
```

Результат запроса:

	pokup_name character varying(50)
1	ООО Буревестник

Пример 11. Вывести покупателя, для которого объем покупки продукции с кодом 1 превосходит любую из поставок этой продукции других поставщиков.

```
SELECT  DISTINCT pokup_name FROM spokup s, book b, kart k
        WHERE b.pokup_kod=s.pokup_kod
            and b.book_id=k.book_id
            and prod_kod =1
            and  kol > ALL(SELECT kol FROM book b1, kart k1
                    WHERE b1.book_id=k1.book_id
                    and k1.prod_kod =k.prod_kod
                    AND b1.pokup_kod<>b.pokup_kod)
```

Результат запроса:

	pokup_name character varying(50)
1	АО ТД Империал

Упражнение 8

1) Вывести покупателей, которые купили продукцию в феврале 2021 г.
2) Вывести продукцию, объем продажи которой был наименьшим среди продаж любой другой продукции

Агрегатные функции

Агрегатные функции (в стандарте SQL они называются функциями над множествами) используются для обобщения отдельных значений данных. Агрегатные функции вычисляют некоторое интегральное значение для заданного множества строк. Таким множеством строк может быть или вся таблица или группа строк, если агрегатная функция применяется к сгруппированной таблице.

При вычислении агрегатной функции по некоторой колонке не учитываются неопределенные значения (NULL). Если в агрегатной функции используется ключевое слово DISTINCT, то из процесса вычислений исключаются значения-дубликаты. Если агрегатная функция применяется без ключевого слова DISTINCT, то вычисление производится над всеми значениями.

В стандарте SQL определены пять агрегатных функций:

1) COUNT - функция подсчета количества значений. Применяется в двух форматах:

COUNT(*) подсчитывает число строк в заданном множестве. Учитываются все строки - в том числе одинаковые и содержащие значение null.

COUNT([DISTINCT] колонка) подсчитывает число не Null-значений в заданном столбце (с DISTINCT – количество разных значений).

2) MAX(выражение) - определяет максимальное значение.
3) MIN(выражение) - определяет минимальное значение.
4) SUM(выражение) - вычисляет сумму значений.
5) AVG(выражение) - вычисляет среднее значение.

Последние 2 функции используются только для числовых значений.

Аргументом агрегатной функции может быть как отельная колонка, так и выражение

Например, SUM(KOL*CENA)

Пример 12. Вывести общий объем всех поставок. Функция вычисляется по всей таблице:

```
SELECT SUM(kol) from kart
```

Результат запроса:

	sum numeric
1	138.00

Пример 13. Определить количество строк, в которых заполнена колонка кода продукции

```
SELECT COUNT(prod_kod) from kart
```

Результат запроса:

	count bigint
1	11

Пример 14. Определить количество разных наименований продукции, проданной покупателю с кодом 1

```
SELECT COUNT (DISTINCT prod_kod)
    FROM book b, kart k
    WHERE b.book_id=k.book_id and pokup_kod=1
```

Результат запроса:

	count bigint
1	2

Упражнение 9.

1) Определить максимальный объем продажи.
2) Определить среднюю цену продажи.
3) Определить количество покупателей в январе 2021 г.

Раздел GROUP BY

Позволяет сгруппировать данные по одной или нескольким колонкам и вычислить для каждой группы агрегатную функцию

```
GROUP BY <список колонок>
```

Результатом будет "***сгруппированная таблица***".

Важно отметить, что в список выборки оператора SELECT, содержащего раздел GROUP BY, можно включать *только* агрегатные функции и колонки, *которые входят в раздел группировки*.

Пример 15. Получить суммарные продажи по продукции для покупателя с кодом 1

```
SELECT  prod_kod,sum(kol)
        FROM book b, kart k
        WHERE  b.book_id=k.book_id
            and pokup_kod=1
        GROUP BY prod_kod
```

Алгоритм выполнения запроса:

1) Будет выполнено прямое произведение таблиц book и kart.
2) К полученным строкам будет применен раздел WHERE, условия которого выполняют естественное соединение таблиц и отбирают строки (продажи) для покупателя с кодом 1.
3) Полученные строки будут сгруппированы так, чтобы в каждую группу попадут строки с одинаковыми значениями в колонке prod_kod.

	pokup_kod integer	kart_id integer	prod_kod integer	cena numeric(12,2)	kol numeric(10,2)	book_id integer
0	1	1	1	25.50	10.00	1
0	1	2	2	55.00	16.00	1
5	1	8	2	55.00	14.00	4

4) В каждой группе будет просуммированы значения в колонке kol.
5) От каждой группы в результирующую таблицу будет включена одна строка.

Результат запроса:

	prod_kod integer	sum numeric
1	1	10.00
2	2	30.00

Пример 16. Получить суммарные продажи в рублях по покупателям

```
SELECT  sp.pokup_name, sum(kol*cena) as stoim
        FROM spokup sp, book b, kart k
        WHERE sp.pokup_kod =b.pokup_kod
```

```
        and b.book_id=k.book_id
    GROUP BY sp.pokup_name
```

Результат запроса:

	pokup_name character varying(50)	stoim numeric
1	ООО Буревестник	569.0000
2	АО ТД Империал	858.5000
3	АО Аэлита	3022.5000
4	ООО Банг и Бонсомер	1905.0000

Упражнение 10.

1) Определить суммарные объемы продажи по каждой продукции
2) Определить количество разных наименований продукции по каждому покупателю.

Раздел HAVING

Условия отбора можно применять не только к строкам таблиц, но и к группам. Раздел HAVING содержит условное выражение, вычисляемое для каждой группы, определенной в разделе GROUP BY. Это условное выражение должно содержать агрегатные функции, вычисляемые для каждой группы.

Рассмотрим пример. Выбрать продукцию, для которой максимальный объем поставки больше 10.

Первый вариант запроса, который напрашивается будет таким:

```
SELECT prod_kod FROM kart WHERE MAX(kol)>1000
```

Но данный оператор запроса не верен, т.к. условие WHERE вычисляется для каждой строки BOOK, т.е. до того как будет вычислена агрегатная функция, которая применима только к группе строк.

Для отбора групп вместо раздела WHERE используется раздел HAVING. Этот раздел задает условие отбора для сгруппированной таблицы.

```
HAVING <условие отбора>
```

В условии отбора раздела HAVING используют те же операции (предикаты), что и в условии отбора раздела WHERE. Но в условии отбора раздела HAVING колонки можно использовать в качестве аргумента агрегатных функций, вычисляющих в данном случае некоторое агрегатное значение для всей группы строк.

Результатом выполнения раздела HAVING будут те строки сгруппированной таблицы, для которых условия отбора дает true.

Замечание. В одном запросе могут встретиться как условия отбора строк в разделе WHERE, так и условия отбора групп в разделе HAVING.

Разница в том, что условие раздела WHERE применяется к выражениям, которые можно вычислить по каждой строке, а в разделе HAVING - к агрегатным функция по группам строк.

Правильная запись оператора запроса для приведенного выше примера:

```
SELECT prod_kod FROM kart
    GROUP BY prod_kod
    HAVING MAX(KOL)>10
```

Пример 17. Получить суммарные поставки по видам продукции, для которых общий объем поставки превышает 20

```
SELECT  PROD_KOD,SUM(KOL) as SUM_KOL
    FROM kart
    GROUP BY prod_kod
    HAVING sum(kol)> 20
```

Результат запроса:

	prod_kod integer	sum_kol numeric
1	1	28.00
2	3	45.00
3	2	41.00

Пример 18.Получить наименования продукции, которая продавалась более одного раза

```
SELECT  prod_name, count(*)
    FROM sprod sp, kart k
    WHERE k.prod_kod =sp.prod_kod
    GROUP BY sp.prod_name
    HAVING count(k.prod_kod) > 1
```

Результат запроса:

	prod_name character varying(50)	count bigint
1	Крем дневной Наташа	3
2	Лифтинг-крем Диамант	3
3	Крем детский	3

Упражнение 11.

1) Вывести покупателей, у которых общий объем поставок в стоимостном выражении превышает 1000
2) Вывести продукцию, у которой минимальная поставка превышает 10

Раздел ORDER BY

Раздел ORDER BY задает желаемый порядок просмотра результата запроса.

Правила записи

```
ORDER BY <список колонок|
```

список порядковых номеров колонок>
[ASC | DESC]

Ключевые слова ASC (обеспечивает упорядочение по возрастанию), DESC (по убыванию) добавляются к каждому элементу списка. Если ключевое слово опущено, то применяется вариант ASC.

Колонки можно задавать:

- их именами
- порядковым номером колонки

Пример 19. Вывести продукцию, упорядочив по наименованию.

```
SELECT prod_kod,prod_name FROM sprod
                ORDER BY prod_name
```

Результат запроса:

	prod_kod integer	prod_name character varying(50)
1	1	Крем детский
2	2	Крем дневной Наташа
3	4	Крем ночной
4	3	Лифтинг-крем Диамант
5	6	Мыло Алиса
6	5	Мыло детское

Упражнение 12. Вывести продажи в обратном хронологическом порядке.

Дополнительные полезные запросы

1) Генерация фиксированной последовательности

Вызов функции

generate_series(нач_значение,конечное_значение [,шаг])

Пример. Сформироать нечетные числа от 1 до 10

```
SELECT * FROM generate_series(1, 10,2)
```

2) Генерация случайной последовательности

```
SELECT random() * макс_значение
    FROM generate_series(нач, конец)
```

Пример. Сформировать 1000 случайных чисел в диапазоне от 0 до 100000

```
SELECT random() * 100000 FROM generate_series(1, 1000)
```

3) Выборка части строк

```
SELECT список_выборки
  FROM табличное_выражение
  [ ORDER BY ... ]
  [ LIMIT  число ] [OFFSET число ]
```

где

LIMIT число - задает количество возвращаемых строк

OFFSET число - задает количество пропускаемых от начала выборки строк

Пример. Сформировать нечетные числа от 1 до 100. Вывести 10 чисел, пропустив от начал 5 чисел

```
SELECT * FROM generate_series(1, 100,2)
        LIMIT 10 OFFSET 5
```

Задание

1. Написать запросы для выполнения упражнений
2. Придумать запросы для своей базы данных

Порядок выполнения работы

1. Изучить теоретический материал
2. Написать запросы по упражнениям практического задания. Представить результаты выполнения запросов
3. Составить запросы для своей базы данных для операций выборки с условиями, группировки данных
4. Выполнить запросы и записать результаты их выполнения

Содержание отчета

1. Запросы по упражнениям практического занятия
2. Результаты выполнения запросов
3. Запросы операций выборки с условиями, группировки данных по своей базе данных
4. Результаты выполнения запросов

Ответы к упражнениям

Упражнение 1. SELECT pokup_name,adres FROM spokup

Упражнение 2. SELECT k.prod_kod,prod_name,k.cena,kol,
cast(kol*k.cena as numeric (10,2)) stoim
FROM sprod sp,kart k,book b
WHERE sp.prod_kod=k.prod_kod
and k.book_id=b.book_id
and ndoc=21 and dat='06-02-2021'

Упражнение 3. SELECT k.prod_kod,prod_name,k.cena,kol,
cast(kol*k.cena as numeric (10,2)) stoim
FROM sprod sp,kart k,book b
WHERE sp.prod_kod=k.prod_kod
and k.book_id=b.book_id
and dat BETWEEN '01-01-2021' and '31-01-2021'

Упражнение 4. 1) SELECT DISTINCT pokup_name
FROM spokup sp,book b, kart k
WHERE sp.pokup_kod=b.pokup_kod
and k.book_id=b.book_id
and prod_kod in (3,4)

2) SELECT DISTINCT prod_name FROM sprod sp, kart k
WHERE sp.prod_kod=k.prod_kod
and book_id in (SELECT DISTINCT book_id
FROM book
WHERE dat BETWEEN '01-01-2021'
and '31-01-2021')

Упражнение 5. SELECT * FROM sprod
WHERE UPPER(prod_name) LIKE '%КРЕМ%'

Упражнение 6. SELECT * FROM spokup WHERE phone is NULL

Упражнение 7. SELECT prod_name FROM sprod sp
WHERE NOT EXISTS (SELECT * FROM kart k, book b
WHERE k.book_id=b.book_id
and k.prod_kod=sp.prod_kod
and dat BETWEEN '01-01-2021'
and '31-01-2021')

Упражнение 8

1) SELECT pokup_name FROM spokup sp
WHERE pokup_kod=ANY (SELECT pokup_kod FROM book
WHERE dat BETWEEN '01-01-2021'
and '31-01-2021')

2) SELECT prod_name FROM sprod sp, kart k
WHERE sp.prod_kod=k.prod_kod

```
        and kol > ALL (SELECT kol FROM kart k1
                       WHERE k1.prod_kod<>k.prod_kod)
```

Упражнение 9

```
1) SELECT max(kol) FROM kart
2) SELECT avg(cena) FROM kart
3) SELECT count(DISTINCT pokup_kod) FROM book
       WHERE dat BETWEEN '01-01-2021' and '31-01-2021'
```

Упражнение 10

```
1) SELECT prod_kod, sum(kol) FROM kart GROUP BY prod_kod
2) SELECT pokup_kod, count(DISTINCT prod_kod)
    FROM book b, kart k
   WHERE b.book_id=k.book_id
   GROUP BY pokup_kod
```

Упражнение 11

```
1) SELECT pokup_name, sum(kol*cena)
       FROM spokup sp, book b, kart k
       WHERE sp.pokup_kod=b.pokup_kod
         and k.book_id=b.book_id
       GROUP BY pokup_name
       HAVING sum(kol*cena)>1000
2) SELECT prod_name, min (kol)
       FROM sprod sp, kart k
       WHERE sp.prod_kod=k.prod_kod
       GROUP BY prod_name
       HAVING min(kol)>10
```

Упражнение 12

```
1) SELECT * FROM book ORDER BY dat DESC
```

Практическое занятие № 37
Язык SQL. Соединение таблиц

Цель занятия. Научиться выполнять соединение таблиц, освоить различные варианты соединения: естественное (внутреннее), неестественное, внешнее.

Краткие теоретические сведения

В реляционной БД взаимосвязанная информация в ходе нормализации распределяется по отдельным таблицам. Соединение позволяет объединить данные связанных таблиц.

Различают следующие виды соединения:

- естественное;
- неестественное;
- внешнее соединение

Виды соединений рассмотрим на примере БД Продажи (рис.1), с заполненными таблицами (рис.2-5).

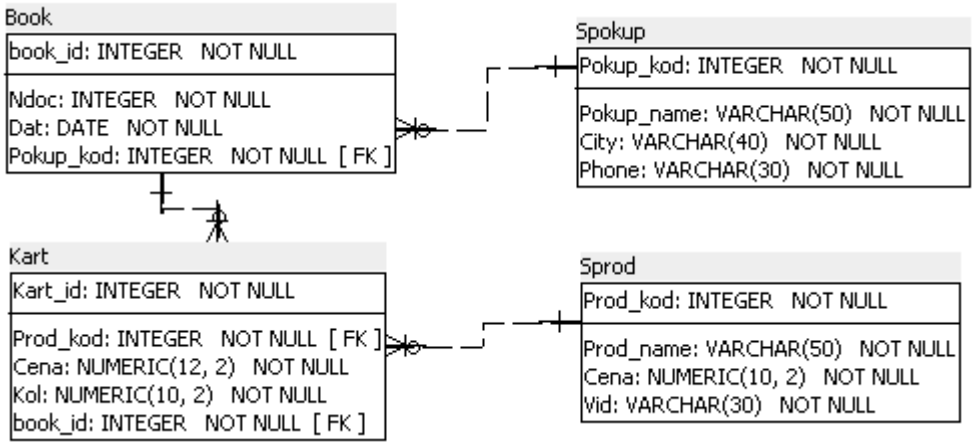


Рис. 1 База данных Продажи

Код	Наименование	Цена	Вид продукции
1	Крем детский	25.50	крем
2	Крем дневной Наташа	55.00	крем
3	Лифтинг-крем Диамант	110.50	крем
4	Крем ночной	95.00	крем
5	Мыло детское	26.00	мыло
6	Мыло Алиса	34.00	мыло
prod_kod	**prod_name**	**cena**	**vid**

Рис. 2 Данные для таблицы sprod (Продукция)

Код	Наименование	Город	Телефон
1	ООО Банг и Бонсомер	Москва	8 (495) 255-45-31
2	АО Аэлита	Москва	8 (499) 124-10-66
3	АО ТД Империал	Тверь	8 (418) 245-86-11
4	ООО Буревестник	Тула	8 (405) 165-41-62
5	ООО Ортекс	Тверь	8 (418) 134-54-12
6	АО Косметик рус	Москва	8 (499) 176-11-75
pokup_kod	pokup_name	city	phone

Рис. 3 Данные для таблицы spokup (Покупатели)

Ид-р продажи	Номер документа	Дата продажи	Код покупателя
1	10	30-01-2021	1
2	15	31-01-2021	3
3	21	06-02-2021	2
4	30	15-02-2021	1
5	35	25-02-2021	4
6	40	01-03-2021	
book_id	ndoc	dat	pokup_kod

Рис. 4 Данные для таблицы book (Книга продаж)

Ид-р ассортимета	Код продукции	Цена	Количество	Ид-р продажи
1	1	25.50	10	1
2	2	55.00	16	1
3	1	25.50	12	2
4	3	110.50	5	2
5	2	55.00	11	3
6	3	110.50	15	3
7	4	95.00	8	3
8	2	55.00	14	4
9	1	25.50	6	5
10	5	26.00	16	5
11	3	34.00	25	6
kart_id	prod_kod	cena	kol	book_id

Рис. 5 Данные для таблицы kart (Ассортимент продаж)

Для иллюстрации всех видов соединений в БД были внесены изменения. В таблице book (Книга продаж) изменена колонка Код покупателя (разрешено содержать значения null). Это изменение можно внести оператором

```
ALTER TABLE book
  alter pokup_kod DROP NOT NULL
```

1) Естественное соединение (внутреннее)

Естественное соединение использует существующие в БД связи таблиц Primary Key(PK) - Foreign Key (FK).

Пример 1. Выберем все продажи (номер документа, дату продажи) с указанием наименования покупателя. Для получения указанных данных необходимо выполнить соединение двух таблиц: *Книги продаж* (*book*) и *Покупателей* (spokup), которые имеют связь PK (spokup.pokup_kod) – FK(book.pokup_kod).

Составим оператор запроса и включим в раздел WHERE предикат сравнения:

```
SELECT ndoc, dat, b.pokup_kod, pokup_name
    FROM  spokup sp, book b
    WHERE sp.pokup_kod=b.pokup_kod
```

Алгоритм выполнения: После прямого произведения таблиц в разделе FROM из всех возможных комбинаций строк таблиц предикатом сравнения раздела WHERE отбираются те, которые совпадают по колонке *Код покупателя* (pokup_kod). Предикат сравнения задает условие естественного соединения таблиц.

Результат запроса

	ndoc integer	dat date	pokup_kod integer	pokup_name character varying(50)
1	10	2021-01-30	1	ООО Банг и Бонсомер
2	15	2021-01-31	3	АО ТД Империал
3	21	2021-02-06	2	АО Аэлита
4	30	2021-02-15	1	ООО Банг и Бонсомер
5	35	2021-02-25	4	ООО Буревестник

Предикат сравнения в разделе WHERE можно заменить оператором JOIN в разделе FROM:

Таблица1 JOIN Таблица2 ON Таблица1.FK=Таблица2.PK

Часто указывают полный вариант записи INNER JOIN, которое дословно переводится как внутреннее соединение, в отличие от внешних соединений, рассматриваемых далее.

Изменим оператор запроса примера 1:

```
SELECT ndoc, dat, b.pokup_kod, pokup_name
    FROM  spokup sp JOIN book b ON sp.pokup_kod=b.pokup_kod
```

Условие соединения таблиц может сочетаться с другими условиями.

Пример 2. Выбрать все продажи продукции с кодом 1: покупатель, дата продажи, количество, цена, стоимость.

В данном примере нам необходимо соединить уже 3 таблицы: Покупателя (spokup) с Книгой продаж (book) по колонке Код покупателя (pokup_kod) и полученные строки соединить с Ассортиментом продажи (kart) по колонке Ид-р продажи (book_id). И предикатом сравнения отобрать строки относящиеся к заданной продукции.

```
SELECT pokup_name, dat, kol, cena, cast (kol*cena as Numeric(10,2)) stoim
    FROM  spokup sp JOIN book b ON sp.pokup_kod=b.pokup_kod
```

```
    JOIN kart k ON b.book_id=k.book_id
  WHERE prod_kod=1
```

Результат запроса:

	pokup_name character varying(50)	dat date	kol numeric(10,2)	cena numeric(12,2)	stoim numeric(10,2)
1	ООО Банг и Бонсомер	2021-01-30	10.00	25.50	255.00
2	АО ТД Империал	2021-01-31	12.00	25.50	306.00
3	ООО Буревестник	2021-02-25	6.00	25.50	153.00

Упражнение 1. Выбрать продажи за январь 2021 г.: покупатель, дата продажи, код продукции, наименование продукции, цена, количество

2) Неестественное соединение

Этот вид соединения основывается на данных, а не на структуре БД (связях таблиц).

Пример 3. Отобрать все пары покупателей таким образом, чтобы пару составляли покупатели из одного города.

Так как надо образовать пары, то необходимо соединить таблицу *Покупатель* с ее копией. Далее из образованных пар покупателей выбрать те, где совпадает город.

```
SELECT p1.pokup_name, p2.pokup_name
    FROM spokup p1, spokup p2
    WHERE p1.city=p2.city
```

Результат запроса

	pokup_name character varying(50)	pokup_name character varying(50)
1	ООО Банг и Бонсомер	АО Косметик рус
2	ООО Банг и Бонсомер	АО Аэлита
3	ООО Банг и Бонсомер	ООО Банг и Бонсомер
4	АО Аэлита	АО Косметик рус
5	АО Аэлита	АО Аэлита
6	АО Аэлита	ООО Банг и Бонсомер
7	АО ТД Империал	ООО Ортекс
8	АО ТД Империал	АО ТД Империал
9	ООО Буревестник	ООО Буревестник
10	ООО Ортекс	ООО Ортекс
11	ООО Ортекс	АО ТД Империал
12	АО Косметик рус	АО Косметик рус
13	АО Косметик рус	АО Аэлита
14	АО Косметик рус	ООО Банг и Бонсомер

Виден недостаток данного запроса: в результат попали пары близнецы (A,A) и зеркальные пары (A,B) (B,A). Эти пары необходимо исключить. Для этого используем еще один предикат, чтобы код покупателя из первой копии был больше кода покупателя из второй (можно выбрать обратное условие).

```
SELECT p1.city, p1.pokup_name, p2.pokup_name
    FROM spokup p1, spokup p2
```

```
WHERE p1.city=p2.city
    and p1.pokup_kod>p2.pokup_kod
ORDER BY p1.city
```

Результат запроса:

	city character varying(40)	pokup_name character varying(50)	pokup_name character varying(50)
1	Москва	АО Аэлита	ООО Банг и Бонсомер
2	Москва	АО Косметик рус	АО Аэлита
3	Москва	АО Косметик рус	ООО Банг и Бонсомер
4	Тверь	ООО Ортекс	АО ТД Империал

При этом необходимо использовать ***алиасы*** (***псевдонимы***), которые позволяют различать соединяемые копии таблиц

3) Внешнее соединение.

Естественное или внутреннее соединение оставляло строки, совпадающие по соединяемым колонкам. В примере 1 покупатели, которые не получали продукцию не попали в результат запроса. Покупатель с кодом 5 не попали в результат выборки. Также в результат не попадут продажи, в которых по каким-то причинам не задан код покупателя (продажа с идентификатором 6).

Более полную картину дают внешние соединения. Внешнее соединение таблиц A и B – может быть:

- левое
- правое
- полное

Левое соединение

```
A LEFT OUTER JOIN B ON A.PK=B.FK
```

включает все строки левой таблицы A независимо от того, есть ли связанные с ней строки в таблице B.

Правое соединение

```
A RIGHT OUTER JOIN B ON A.PK=B.FK
```

включает все строки правой таблицы B независимо от того, есть ли связанные с ней строки в таблице A.

Полное соединение

```
A FULL OUTER JOIN B ON A.PK=B.FK
```

объединяет левое и правое соединения.

Все виды соединений можно представить в виде наглядных интерпретаций (рис. 6).

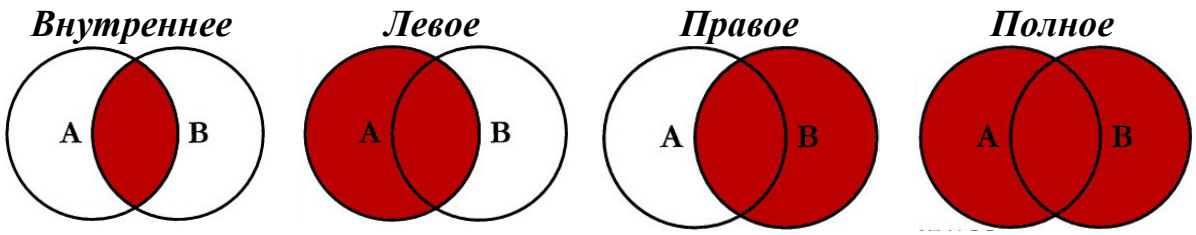


Рис.6 Наглядные интерпретации всех видов соединений

Пример 4. Вывести для всех покупателей вывести полную информацию об их покупках

```
SELECT sp.pokup_name, ndoc, dat
    FROM spokup sp LEFT OUTER JOIN book b
     ON b.pokup_kod=sp. pokup_kod
```

Будут отобраны все покупатели, независимо от того приобретали они продукцию или нет. Если покупатель не покупал продукцию, то колонки ndoc и dat будут содержать `null`; а если покупал, то число его вхождений в результат выборки равно числу его покупок.

Результат запроса:

	pokup_name character varying(50)	ndoc integer	dat date
1	ООО Банг и Бонсомер	10	2021-01-30
2	АО ТД Империал	15	2021-01-31
3	АО Аэлита	21	2021-02-06
4	ООО Банг и Бонсомер	30	2021-02-15
5	ООО Буревестник	35	2021-02-25
6	ООО Ортекс		
7	АО Косметик рус		

Пример 5. Вывести информацию о всех продажах с указанием наименования покупателя.

```
SELECT sp.pokup_name, ndoc, dat
    FROM spokup sp RIGHT OUTER JOIN book b
     ON b.pokup_kod=sp. pokup_kod
```

Будут отобраны все строки книги продаж, независимо от того указан ли в продаже покупатель. Если покупатель не указан, то колонка pokup_name будет содержать `null`.

Результат запроса:

	pokup_name character varying(50)	ndoc integer	dat date
1	ООО Банг и Бонсомер	10	2021-01-30
2	АО ТД Империал	15	2021-01-31
3	АО Аэлита	21	2021-02-06
4	ООО Банг и Бонсомер	30	2021-02-15
5	ООО Буревестник	35	2021-02-25
6		40	2021-03-01

Пример 6. Вывести информацию о всех продажах и покупателях

```
SELECT sp.pokup_name, ndoc, dat
    FROM spokup sp FULL OUTER JOIN book b
     ON b.pokup_kod=sp. pokup_kod
```

Результат объединяет данные предыдущих двух запросов:

	pokup_name character varying(50)	ndoc integer	dat date
1	ООО Банг и Бонсомер	10	2021-01-30
2	АО ТД Империал	15	2021-01-31
3	АО Аэлита	21	2021-02-06
4	ООО Банг и Бонсомер	30	2021-02-15
5	ООО Буревестник	35	2021-02-25
6		40	2021-03-01
7	ООО Ортекс		
8	АО Косметик рус		

Задание.

1. Составить запросы для выполнения упражнений практического занятия
2. В БД имеются две таблицы

Продукция (SPROD)

Код	**Наименов.**	**Группа**
010	Samsung 32″	Телевизоры
011	LG DF4	СВЧ печь
012	Bork A3	СВЧ печь
013	LG 41″	Телевизоры
014	Panasonic 26″	Телевизоры
Prod_kod	**Prod_name**	**Grup**

Книга продаж (BOOK)

Ид-р книги	**Покупатель**	**Код прод.(FK)**	**Кол-во**
1	АО Смена	010	10
2	ООО Темп	011	12
3	АО Смена	011	25
4	АО Смена	null	16
5	Null	012	8
6	ПАО Крок	013	18
Book_Id	**Pokup**	**Prod_kod**	**Kol**

Составить операторы запроса для выполнения различных видов соединений (JOIN):

- внутреннего;
- внешнего (левого);
- внешнего (правого);
- внешнего (полного);

и привести результаты их выполнения.

3. Составить и выполнить запросы на различные виды соединения таблиц для своей базы данных

Порядок выполнения работы

1. Изучить теоретический материал
2. Написать запросы по упражнения практического занятия. Представить результаты выполнения запросов
3. Написать запросы по заданию. Представить результаты выполнения запросов
4. Составить запросы различных видов соединения таблиц для своей базы данных. Выполнить запросы и записать результаты их выполнения

Содержание отчета

1. Запросы по упражнениям практического занятия. Результаты выполнения запросов
2. Запросы по пункту 2 задания и результаты их выполнения
3 Запросы соединения таблиц по своей базе данных и результаты их выполнения

Ответы

Упражнение 1.

```
SELECT pokup_name, dat, k.prod_kod, prod_name, kol, k.cena
      FROM  spokup sp JOIN book b ON sp.pokup_kod=b.pokup_kod
         JOIN kart k ON b.book_id=k.book_id
         JOIN sprod pr ON k.prod_kod=pr.prod_kod
      WHERE dat BETWEEN '01-01-2021' and '31-01-2021'
```

Ответы к упражнениям

```
1. SELECT prod_name
       FROM sprod
       WHERE vid=(SELECT vid FROM sprod where prod_kod=3)
            and prod_kod<>3
2. SELECT prod_name
      FROM sprod
      WHERE prod_kod IN (SELECT prod_kod FROM kart k
            JOIN book b ON k.book_id=b.book_id
            JOIN spokup sp ON b.pokup_kod=sp.pokup_kod
             WHERE pokup_name='ООО Банг и Бонсомер')
3. SELECT prod_name
      FROM sprod sp1
      WHERE EXISTS (SELECT 1 FROM kart k
            JOIN book b ON k.book_id=b.book_id
            JOIN spokup sp ON b.pokup_kod=sp.pokup_kod
             WHERE pokup_name='ООО Банг и Бонсомер'
               and k.prod_kod=sp1.prod_kod)
```

4. SELECT prod_name, (SELECT count (DISTINCT pokup_kod)
FROM kart k JOIN book b ON b.book_id=k.book_id
WHERE k.prod_kod=sp.prod_kod)
FROM sprod sp

Практическое занятие № 38
Язык SQL. Операторы изменения данных

Цель занятия. Научиться составлять операторы SQL для добавления, изменения и удаления данных в таблицах реляционных баз данных

Краткие теоретические сведения

Оператор добавления строк

Оператор INSERT применяется для добавления одной или нескольких строк в таблицы БД и имеет формат:

```
INSERT INTO <имя_табл> [( список колонок )]
    ... {VALUES ( <список выражений> )[(список)…]|
          <спецификация запроса>}
```

где

VALUES - вводит список выражений, которые подставляются в соответствующие колонки. Колонки, не указанные в списке колонок, получают значения по умолчанию или NULL

При использовании запроса он должен содержать такие же колонки и в том же порядке

Если список колонок не определен, то данные подставляются во все колонки в том порядке, в котором они были созданы.

Пример 1. Вставка одной строки в таблицу (добавление нового покупателя):

```
INSERT INTO spokup (pokup_kod, pokup_name, city)
          VALUES (7, 'ООО Темп', 'Москва');
```

Пример 2. Вставка в таблицу нескольких строк, выбранных из другой таблицы. В таблицу tmp_table вставляются данные о покупателях из таблицы spokup, имеющие коды, от 2 до 6:

```
INSERT INTO tmp_table (pokup_kod,pokup_name)
 SELECT pokup_kod,pokup_name
  FROM spokup
  WHERE pokup_kod BETWEEN 2 and 6
```

Оператор удаления строк

Оператор DELETE применяется для удаления строк из таблицы

```
DELETE FROM <имя таблицы>
   [WHERE <условие отбора>]
```

Раздел FROM указывает таблицу, в которой выполняется удаление. Если не указан раздел WHERE, то удаляются все строки таблицы.

Пример 3. Удалить все строки таблицы BOOK

```
DELETE book
```

Пример 4. Удаление строк, удовлетворяющих простому условию.

Удалить из списка продаж продукцию с кодом 1.

```
DELETE  kart  WHERE Prod_kod=1
```

Пример 5. Удаление строк, удовлетворяющих сложному условию (принадлежность списку).
Удалить продажу продукции, относящуюся к виду «мыло».

```
DELETE kart WHERE prod_kod IN (SELECT prod_kod FROM sprod
        WHERE vid='мыло')
```

Упражнение 1. Придумайте пример удаления строк по условию с использованием предиката EXISTS

Оператор обновления строк

Оператор UPDATE применяется для изменения одного или нескольких столбцов таблицы.

```
UPDATE <имя таблицы>
...SET <имя колонки> = {<выражение>| NULL | DEFAULT |
<подзапрос>}, ...
[ WHERE <условие отбора>]
```

Если в выражении справа используется имя колонки изменяемой таблицы, то используется ее прежнее значение. Например,

```
KOL=KOL+10
```

Раздел WHERE задает условие для строк, которые надо изменить.

Пример 6. Изменить продажи для продукции с кодом 2: объем в натуральном выражении увеличить на 10, а цену уменьшить в 10 раз

```
UPDATE book SET kol=kol+10, cena=cena/10
    WHERE prod_kod=2
```

Задание

1. Составить запросы для выполнения упражнений практического занятия
2. Задана таблица Продажи (BookSale) с колонками:

Имя колонки	**Описание**
Book_id	Идентификатор поставки
Dat	Дата поставки
Pokup_kod	Код покупателя
Kol	Количество
Brak	Количество брака (в том числе)

Написать операторы для выполнения следующих действий:

а) Добавить одну продажу
Ид-р=1000, Дата=29.03.2021, Код покупателя=23, количество

=55, Количество брака=3

б) По книге продаж сформировать справочник покупателей (таблица spokup) с колонками:
Pokup_kod - код покупателя
Pokup_name - наименование покупателя.
Код покупателя взять из книги продаж,
Наименование покупателя сформировать по правилу:
«Покупатель», пробел, Код покупателя.
Учесть, что покупатель мог совершить несколько покупок.
Для соединения (сцепления) строк используется оператор ||

в) Передать все покупки от покупателя с кодом 10 покупателю с кодом 15

г) В таблицу добавлена новая колонка Sort - количество небракованной продукции. Заполнить ее по имеющимся данным

д) Удалить все продажи за 1-й квартал 2021 года, для которых объем продажи меньше 100

е) Удалить продажи с браком для покупателя с кодом 8

3. Составить и выполнить операции добавления, удаления и изменения данных в своей базе данных

Порядок выполнения работы

1. Изучить теоретический материал

2. Написать запросы по упражнения практического занятия. Представить результаты выполнения запросов

3. Написать запросы по пункту 2 задания. Представить результаты выполнения запросов

4. Составить запросы различных видов изменения данных в таблицах для своей базы данных. Выполнить запросы и записать результаты их выполнения

Содержание отчета

1. Запросы по упражнениям практического занятия. Результаты выполнения запросов

2. Запросы по пункту 2 задания и результаты их выполнения

3 Запросы по изменению таблиц по своей базе данных и результаты их выполнения

Ответы на упражнения

1. DELETE kart k WHERE EXISTS (SELECT 1 FROM sprod sp
WHERE sp.prod_kod=k.prod_kod and vid='мыло')

Ответы на задание

1) INSERT INTO book_sale (book_id, dat, pokup_kod, kol, ,brak)
VALUES(1000,'29-03-2021', 23, 55, 3)
2) INSERT INTO spokup(pokup_kod,pokup_name)
SELECT DISTINCT pokup_kod, 'Покупатель'||' '||pokup_kod
FROM BookSale
ORDER BY pokup_kod
3) UPDATE BookSale set pokup_kod=15
WHERE pokup_kod=10
4) UPDATE BookSale set sort=kol-brak
5) DELETE FROM BookSale WHERE kol<100
AND dat BETWEEN '01-01-2021 ' and '31-03-2021 '
6) DELETE FROM BookSale WHERE pokup_kod=8
and brak>0

Практическое занятие № 39
Использование индексов

Цель занятия. Познакомиться с понятием индекса и типами индексов PostgreSQL. Изучить механизм поиска данных без использования и с использованием индексов. Научиться составлять планы выполнения запросов.

Краткие теоретические сведения

Понятие индекса

Индексы в PostgreSQL - это специальные объекты базы данных, предназначенные для:

- ускорения доступа к данным;
- поддержания ограничений целостности данных (например, индексы, поддерживающие уникальность по колонкам, не входящие в первичный ключ).

Индекс устанавливает соответствие между ключом (значением проиндексированного столбца) и строками таблицы, в которых этот ключ встречается. Строки идентифицируются с помощью TID (tuple id), который включает номер блока файла и номер позиции строки внутри блока. По индексу можно быстро перейти к нужной строке, не просматривая всю таблицу полностью.

Любой индекс можно удалить и восстановить заново по информации в таблице.

Надо учитывать, что использование индексов требует накладных расходов. При любой операции над колонками, входящими в индекс (добавление, удаление или обновление строки таблицы), необходимо перестраивать и индекс, на что требуются ресурсы (в первую очередь - время).

Любой выполняемый запрос проходит через этап оптимизации. Оптимизатор строит план запроса, оценивая различные пути его выполнения запроса: где надо использовать индекс, а где выполнять последовательный просмотр без использования индексов.

Основные способы сканирования (просмотра) таблиц

При выполнении запросов оптимизатор может использовать различные стратегии:

- использование индекса
- использование побитовой карты
- последовательный просмотр

Индексное сканирование

Рассмотрим пример. Создадим таблицу *test* с тремя колонками

```
create table test(id integer, txt text, logic boolean);
```

и заполним ее данными:

```
insert into test(id,txt,logic)
  select s.id, chr((32+random()*94)::integer),
               random() < 0.01
  from generate_series(1,100000) as s(id)
  order by random();
```

Первая колонка содержит случайные числа от 1 до 100000.

Вторая колонка содержит различные ASCII-символы, генерируемые случайным образом.

Третья колонка содержит логическое значение, истинное примерно для 1% строк, и ложное для остальных.

Строки вставлены в таблицу в случайном порядке.

Попробуем выбрать строку таблицы по условию «id = 100».

```
SELECT * FROM test WHERE id=100
```

В окне сообщений pgAdmin отобразится время выполнения запроса (33 мс)

Total query runtime: 33 msec

1 строка получена.

Построим план выполнения:

```
EXPLAIN (costs off) SELECT * FROM t WHERE a=2
```

и получим:

	QUERY PLAN text
1	Seq Scan on test
2	Filter: (id = 100)

Видно, что поиск выполняется методом последовательного (строка за строкой) просмотра.

Построим по 1-й колонке индекс.

```
CREATE INDEX ON test(id);
```

Повторим выполнение того же запроса и оценим время его выполнения:

Total query runtime: 13 msec

1 строка получена.

Построим снова план запроса:

	QUERY PLAN text
1	Index Scan using test id idx on test
2	Index Cond: (id = 100)

Видно, что при выполнении запроса используется индекс по условию предиката сравнения и время выполнения запроса сократилось в 2,5 раза.

В условии используется предикат сравнения с оператором «равно», и сравнение проводится с константой «100». Это именно то условие, когда оптимизатор для поиска подключает индекс.

При индексном просмотре сервер возвращает значения TID по одному, до тех пор, пока подходящие строки не закончатся. Для каждого полученного TID сервер обращается к тем страницам памяти, на которые указывают TID, и возвращает полученные данные.

Сканирование по битовой карте

Индексное сканирование хорошо работает, когда запрос должен вернуть несколько строк. Однако при увеличении выборки часто придется возвращаться к одной и той же странице памяти несколько раз. Поэтому в таком случае оптимизатор переключается на сканирование по битовой карте (bitmap scan).

Для примера рассмотрим план выполнения запроса, возвращающего порядка 100 строк:

```
explain (costs off) select * from test where id <= 100;
```

	QUERY PLAN text
1	Bitmap Heap Scan on test
2	Recheck Cond: (id <= 100)
3	-> Bitmap Index Scan on test id idx
4	Index Cond: (id <= 100)

Сначала возвращаются все TID, соответствующие условию (узел Bitmap Index Scan), и по ним строится битовая карта строк. Затем строки читаются из таблицы (Bitmap Heap Scan). При этом каждая страница будет прочитана только один раз.

Если условия применены к нескольким проиндексированным колонкам таблицы, то сканирование битовой карты позволяет использовать несколько индексов одновременно. Для каждого индекса строятся битовые карты строк, которые затем побитово логически умножаются (если выражения соединены условием AND), либо логически складываются (если выражения соединены условием OR).

Создадим индекс по второй колонке таблицы test:

```
create index on test(txt)
```

и проверим план выполнения запроса с условие по двум индексированным колонкам

```
explain (costs off) select * from test where id<=100 and txt='1';
```

	QUERY PLAN text
1	Bitmap Heap Scan on test
2	Recheck Cond: ((id <= 100) AND (txt = '1'::text))
3	-> BitmapAnd
4	-> Bitmap Index Scan on test id idx
5	Index Cond: (id <= 100)
6	-> Bitmap Index Scan on test txt idx1
7	Index Cond: (txt = '1'::text)

Здесь узел Bitmap объединяет две битовые карты с помощью битовой операции «и».Сканирование по битовой карте позволяет избежать повторных обращений к одной и той же

Последовательное сканирование

Но даже при наличии индекса оптимизатор может выбрать последовательное сканирование, если будет задано «неселективное» условие (т.е. возвращающее очень большое количество строк):

```
explain (costs off) select * from test where id <= 50000;
```

	QUERY PLAN text
1	Seq Scan on test
2	Filter: (id <= 50000)

Такой план выбирается потому, что при увеличении выборки возрастают и накладные расходы на чтение страниц индекса.

В PostgreSQL можно использовать несколько видов индексов. Рассмотрим наиболее популярные.

Хеш-индекс (Hash)

Как правило, типы данных имеют очень большие диапазоны допустимых значений: сколько различных строк можно теоретически представить в столбце типа text? В то же время, сколько разных значений реально хранится в текстовом столбце какой-нибудь таблицы? Обычно не так много.

Идея хеширования состоит в том, чтобы значению любого типа данных сопоставить некоторое небольшое число (от 0 до $N-1$). Такое сопоставление называют *хеш-функцией.* Полученное число можно использовать как индекс. Элементы такого индекса называют *корзинами хеш-таблицы* — в одной корзине могут лежать несколько TID, если одно и то же проиндексированное значение встречается в разных строках таблицы.

Хеш-функция тем лучше, чем равномернее она распределяет исходные значения по корзинам. Но даже хорошая функция будет иногда давать одинаковый результат для разных входных значений — это называется

коллизией. Так что в одной корзине могут оказаться TID, соответствующие разным ключам, и поэтому полученные из индекса TID необходимо перепроверять.

При вставке в индекс вычислим хеш-функцию для ключа. В эту корзину и кладется TID. Хеш-функции в PostgreSQL всегда возвращают тип integer, что соответствует диапазону 2^{32} значений (приблизительно 4 миллиарда). Число корзин изначально равно двум и увеличивается динамически, подстраиваясь под объем данных; номер корзины можно вычислить по хеш-коду с помощью битовой арифметики.

При поиске в индексе вычисляется хеш-функция для ключа и получается номер корзины. Далее перебирается все содержимое корзины и возвращаются только подходящие TID. Это делается эффективно, поскольку пары «хеш-код — TID» хранятся упорядоченно.

Хеш-индекс (рис.1), использует страницы четырех видов:

- Метастраница (meta page) — нулевая страница, содержит информацию об индексе;
- Страницы корзин (bucket page) — основные страницы индекса, хранят данные в виде пар «хеш-код — TID»;
- Страницы переполнения (overflow page) — аналогичны страницам корзин и используются в случае, когда одной страницы для корзины не хватает;
- Страницы битовой карты (bitmap page)— в них отмечаются освободившиеся страницы переполнения, которые можно использовать для других корзин.

Стрелочки вниз, идущие от элементов индексных страниц, символизируют TID — ссылки на табличные строки.

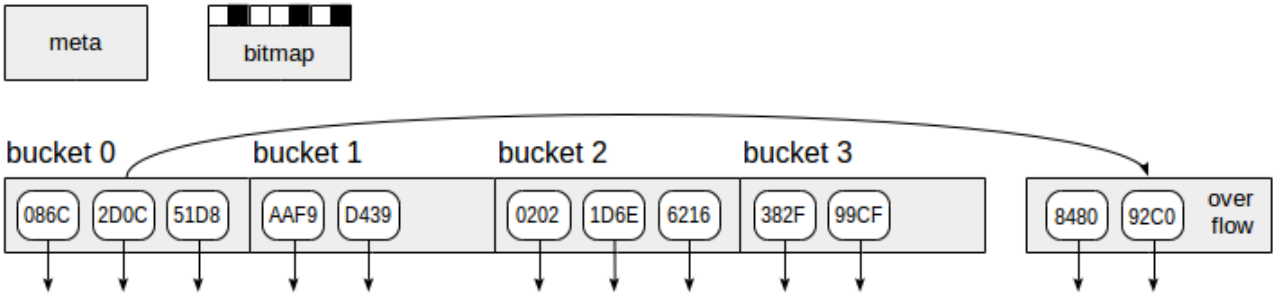


Рис. 1 Организация хэш-индекса

Хеш-функция не сохраняет отношение порядка: из того, что значение хеш-функции одного ключа меньше значения функции другого ключа, нельзя сделать никаких выводов о том, как упорядочены сами ключи. Поэтому хеш-индекс в принципе может поддерживать единственную операцию «равно». Также хеш-индекс не работает с неопределенными значениями: операция «равно» не имеет смысла для null.

Особенностью PostgreSQL до 10 версии является то, что действия с хеш-индексом не попадают в журнал транзакций (о чем предупреждается при создании индекса). Как следствие, хеш-индексы не могут быть восстановлены после сбоя и не участвуют в репликации. Кроме того, хеш-индекс значительно уступает B-дереву по универсальности применения, и его эффективность тоже вызывает вопросы. То есть до версии 10 применять такие индексы не имеет практического смысла.

Рассмотрим пример создания хеш-индекса на тестовой таблице. Создадим индекс по второй колонке:

```
create index on test using hash(txt);
```

и определим план выполнения запроса

```
explain (costs off) select * from test where txt = 'a';
```

	QUERY PLAN text
1	Bitmap Heap Scan on test
2	Recheck Cond: (txt = 'a'::text)
3	-> Bitmap Index Scan on test txt idx
4	Index Cond: (txt = 'a'::text)

Двоичное дерево (Btree)

Индекс Btree, т.е. B-дерево (двоичное дерево), пригоден для данных, которые можно отсортировать. Для таких данных должны быть определены операторы «больше», «больше или равно», «меньше», «меньше или равно» и «равно». Индексные записи B-дерева упакованы в страницы. В листовых страницах эти записи содержат индексируемые данные (ключи) и ссылки на строки таблицы (TID-ы); во внутренних страницах каждая запись ссылается на дочернюю страницу индекса и содержит минимальное значение ключа в этой странице. По умолчанию для таблицы БД создается именно индекс Btree.

B-деревья обладают несколькими важными свойствами:

- Они сбалансированы, то есть любую листовую страницу отделяет от корня одно и то же число внутренних страниц. Поэтому поиск любого значения занимает одинаковое время.
- Они сильно ветвисты, то есть каждая страница (как правило, 8 КБ) содержит сразу много (сотни) TID. За счет этого глубина B-деревьев получается небольшой; на практике до 4–5 для очень больших таблиц.
- Данные в индексе упорядочены по неубыванию (как между страницами, так и внутри каждой страницы), а страницы одного уровня связаны между собой двунаправленным списком. Поэтому получить упорядоченный набор данных можно, просто проходя по списку в одну или в другую сторону, не возвращаясь каждый раз к корню.

На рис. 2 приведен пример индекса по одному полю с целочисленными ключами. В самом начале файла находится метастраница, которая ссылается

на корень индекса. Ниже корня расположены внутренние узлы; самый нижний ряд — листовые страницы. Стрелочки вниз символизируют ссылки из листовых узлов на строки таблицы (TID).

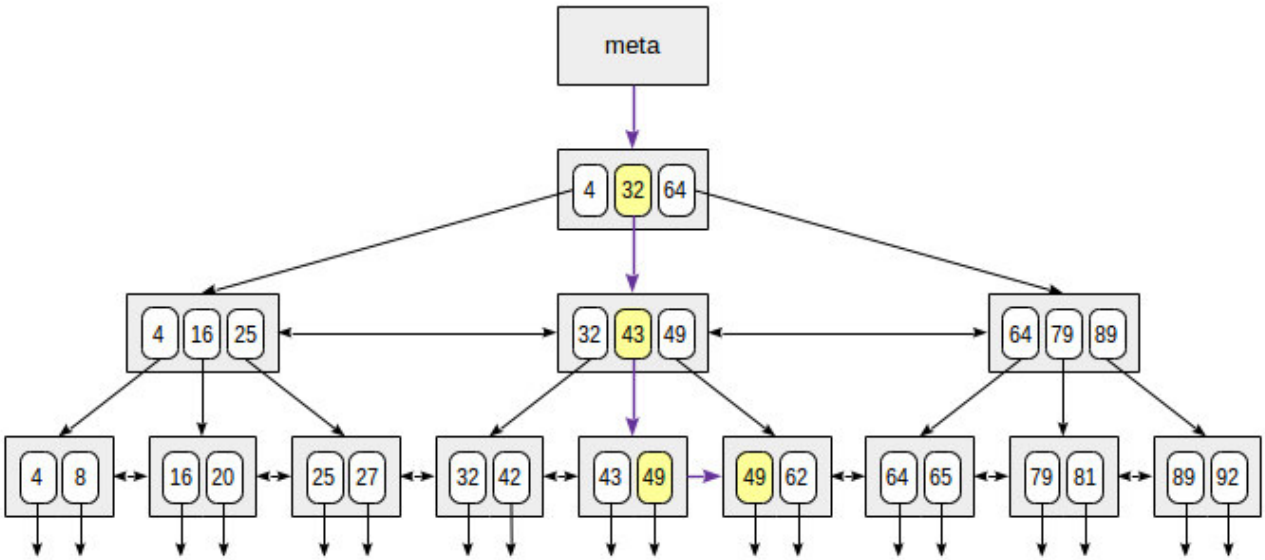


Рис. 2 Организация индекса Btree

Рассмотрим поиск значения в дереве по условию «*индексированное-поле = выражение*». Допустим, нас интересует ключ 49.

Поиск начинается с корневого узла, и нам надо определить, в какой из дочерних узлов спускаться. Зная находящиеся в корневом узле ключи (4, 32, 64) мы тем самым понимаем диапазоны значений в дочерних узлах. Поскольку $32 \leq 49 < 64$, надо спускаться во второй дочерний узел. Дальше та же процедура повторяется рекурсивно до тех пор, пока не будет достигнут листовой узел, из которого уже можно получить необходимые TID.

В реальности эта простая на вид процедура осложняется рядом обстоятельств. Например, индекс может содержать неуникальные ключи и одинаковых значений может оказаться достаточно много, чтобы они не поместились на одну страницу. Продолжая наш пример, кажется, что из внутреннего узла следовало бы спуститься по ссылке, которая ведет от значения 49. Но, как видно на картинке, так мы пропустим один из ключей 49 в предыдущей листовой странице. Поэтому, обнаружив на внутренней странице точное равенство ключа, нам приходится спускаться на одну позицию левее, а затем просматривать индексные записи нижележащего уровня слева направо в поисках интересующего ключа.

Рассмотрим пример создания индекса на тестовой таблице.

```
create index on test using btree(id),
```

Предложение «using btree» можно не использовать, т.к. по умолчанию строится B-дерево.

Индекс GiST

GiST — сокращение от «generalized search tree». Это сбалансированное дерево поиска, точно так же, как и рассмотренный ранее индекс b-tree.

Разница в том, что индекс b-tree жестко привязан к семантике сравнения: поддержка операторов «больше», «меньше», «равно». Но в современных базах хранятся и такие типы данных, для которых эти операторы просто не имеют смысла: геоданные, текстовые документы, картинки.

Тут на помощь и приходит индексный метод GiST. Он позволяет задать принцип распределения данных произвольного типа по сбалансированному дереву, и метод использования этого представления для доступа по некоторому оператору. Например, в GiST-индекс можно «уложить» R-дерево для пространственных данных с поддержкой операторов взаимного расположения (находится слева, справа; содержит и т. п.), или RD-дерево для множеств с поддержкой операторов пересечения или вхождения.

За счет расширяемости в PostgreSQL вполне можно создать совершенно новый метод доступа с нуля: для этого надо реализовать интерфейс с механизмом индексирования. Но это требует продумывания не только логики индексации, но и страничной структуры, эффективной реализации блокировок, поддержки журнала упреждающей записи — что подразумевает очень высокую квалификацию разработчика и большую трудоемкость. GiST упрощает задачу, беря на себя низкоуровневые проблемы и предоставляя свой собственный интерфейс: несколько функций, относящихся не к технической сфере, а к прикладной области. В этом смысле можно говорить о том, что GiST является каркасом для построения новых методов доступа.

Задание.

1. Создать индексы для своей БД.
2. Построить планы выполнения запроса по индексированным колонкам.

Порядок выполнения работы

1. Изучить теоретический материал
2. Создать индексы для своей БД
3. Составить запросы получения данных с участием колонок, входящих в индекс. Получить план выполнения запроса

Содержание отчета

1. Программный код создания индексов
2. Запросы получения данных с участием колонок, входящих в индекс.
3. Планы выполнения запросов

Практическое занятие № 40
Чтение данных и запись файлов различных форматов

Цель занятия. Изучить основные функции библиотеки pandas для работы с внешними файлами. Научиться загружать и выгружать данные в популярном формате CSV

Краткие теоретические сведения

Форматы файлов данных

Данные записанные во внешних файлах могут быть представлены в различных форматах. Наиболее популярные из них:

- CSV
- JSON
- PARQUET

Файл CSV – это особый вид файла, который позволяет структурировать большие объемы данных. Это обычным текстовым файлом. Обычно каждая запись начинается с новой строки, а каждый новый элемент строки отделен от предыдущего запятой или другим разделителем. в первой строке обычно указывается, какая информация будет находиться в каждом столбце. Данные CSV можно легко экспортировать в электронные таблицы, базы данных, системы машинного обучения.

Пример CSV файла. Учащиеся

```
Фамилия,Имя,Класс,Год рождения
Иванов,Алексей,11а,2005
Михайлов,Виктор,11а,2006
…
```

JSON – это текстовый формат данных, используемый практически во всех скриптовых языках программирования. Данные представлены в виде иерархического словаря и структурированы в парах «ключ-значение». Могут включать массивы значений, заключенные в квадратные скобки

Пример JSON файла. Учащиеся

```
{
  "class": "11а",
  "pupils": [
    {
      "fam": "Иванов",
      "name": "Алексей",
      "age": 16},
    {
      "fam": " Михайлов ",
      "name": "Виктор",
      "age": 15},
```

```
        ...
        ]
    "class": "11,",
        ...
}
```

PARQUET - это формат данных, ориентированный на колоночный формат хранения больших данных. Он разработан для повышения эффективности чтения и записи фреймов данных и упрощения обмена данными между языками анализа данных. Отличие от строко ориентированного хранения рассмотрим на простом примере. Например, если структура данных включает 3 колонки (A, B, C), то в формате CSV, они записываются строка за строкой:

```
A1, B1, C1
A2, B2, C2
```

В формате PARQUET они сохраняются по столбцам:

```
A1, A2, A3
B1, B2, B3
C1, C2, C3
```

Загрузка данных из файлов CSV

В задачах анализа данных и машинного обучения более популярным является формат CSV. Рассмотрим основные операции с такими файлами.

Для того чтобы загрузить данные из файла в формате CSV в набор данных DataFrame необходимо использовать функцию `read_csv`:

```
pd.read_csv(путь к файлу)
```

Если разделителем является не запятая, то ее надо указать дополнительным параметром `sep='разделитель'`.

Если в файле нет первой строки с названиями столбцов, то надо дополнительно задать параметр: `header=None`. Но в этом случае надо также задать параметр с именами столбцов `names=[список имен]`.

Для выполнения операций загрузки будем использовать данные с сайта Kaggle об успеваемости учащихся при сдаче эзаменов[1]. Предварительно необходимо скачать файл на компьютер (требуется регистрация на сайте с указанием своей электронную почту).

Прочитаем весь файл

```
import pandas as pd
df=pd.read_csv('d:\data\StudentsPerformance.csv')
```

Выведем размер и структуру набора данных, используя функцию `count()`:

```
df.count()
```

```
gender                         1000
race/ethnicity                 1000
parental level of education    1000
lunch                          1000
test preparation course        1000
math score                     1000
reading score                  1000
writing score                  1000
dtype: int64
```

Бывают ситуации, когда набор данных очень большой и его трудно загрузить. Часто требуется выборочная загрузка, когда вам нужны не все данные, содержащиеся в наборе данных, а только некоторые столбцы или некоторые записи, удовлетворяющие некоторым критериям.

В этом случае можно использовать варианты выборочной загрузки:

- загрузка заданных столбцов;
- загрузка заданного набора строк.

Загрузка отдельных столбцов

В этом случае используется параметр функции загрузки `usecols`, в котором в виде списка задается набор столбцов.

Пример. Загрузим уровень образования и оценки, т.е. столбцы с именами:

```
parental level of education
math score
reading score
writing score
```

```
df1=pd.read_csv('d:\data\StudentsPerformance.csv',\
        usecols=['parental level of education','math score',\
                 'reading score','writing score'])
```

```
df1.head(3)
```

	parental level of education	math score	reading score	writing score
0	bachelor's degree	72	72	74
1	some college	69	90	88
2	master's degree	90	95	93

Выборочная загрузка строк

В этом случае загружается только несколько строк набора данных. Обычно выбирается начальная строка и заданное число строк. Следует

учесть, что если первая строка не загружается, то названия столбцов необходимо передать функции чтения в качестве дополнительного параметра.

В команде загрузки используются параметры:

`skiprows` - количество пропускаемых строк (по умолчанию 0);

`nrows` - количество загружаемых строк

`columns` - - имена загружаемых колонок

Пример. Пропустим 5 строк и загрузим 100 строк

```
srows=5   # Пропуск 5 строк
crows=100 # Загружаем 100 следующих строк
# Задаем имена загружаемых колонок
columns=['parental level of education','math score',\
                 'reading score','writing score']
df1=pd.read_csv('d:\data\StudentsPerformance.csv',\
                skiprows=srows,nrows=crows,names=columns)
```

```
df1.count()
```

```
parental level of education    100
math score                     100
reading score                  100
writing score                  100
dtype: int64
```

Для загрузки данных из файлов других форматов используются другие методы pandas:

read_json - для загрузки данных из файлов в формате JSON

`read_parquet` - для загрузки данных из файлов в формате PARQUET

Сохранение данных в файле

Часто требуется сохранить данные обработки во внешнем файле. Это выполняется методом набора данных `to_csv`, которому в качестве параметра передается путь к внешнему файлу.

Выполним сохранение набора данных:

```
df1.to_csv('d:\data\exams.csv')
```

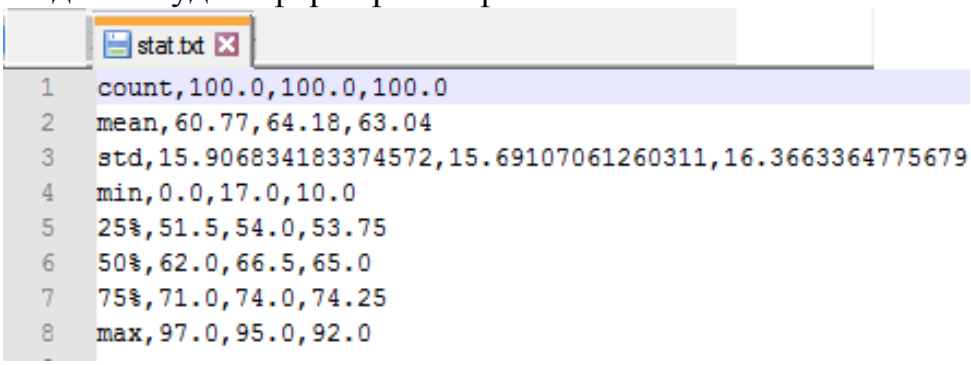
Пример. Сформируем статистику по набору данных и выведем ее в текстовый файл без заголовочной строки.

Если надо сохранить данные без заголовочной строки, используется дополнительный параметр `header=None`.

```
stat=df1.describe()
```

```
stat.to_csv('d:\data\stat.txt',header=None)
```

На диске будет сформирован файл



Задание

1. Найти набор данных в репозитории Kaggle[2] или [3] и скачать его на свой компьютер
2. Выполнить загрузку в разных вариантах: полной, отдельных столбцов, части строк
3. Выполнить обработку данных и сохранить результаты во внешнем файле

Список источников

1. Набор данных Kaggle .- https://www.kaggle.com/spscientist/students-performance-in-exams
2. Репозиторий Kaggle. - https://www.kaggle.com/datasets
3. Репозиторий наборов данных для машинного обучения.- https://archive.ics.uci.edu/ml/index.php

Порядок выполнения работы

1. Изучить теоретический материал
2. Выполнить пункты задания

Содержание отчета

1. Описание наборов данных для загрузки
2. Результаты выборки загруженных данных
3 Результаты обработки загруженных данных

Практическое занятие № 41
Средства аналитической обработки PySpark

Цель занятия. Изучить основные возможности библиотеки аналитической обработки данных PySpark, его основные методы и функции.

Краткие теоретические сведения

Apache Spark является мощным средством аналитической обработки данных для крупномасштабных мощных приложений распределенной обработки данных и машинного обучения. Освоить функционал Apache Spark можно с использованием PySpark - библиотеки Spark, написанной на Python.

Для работы с PySpark в приложении Jupyter Notebook необходимо загрузить библиотеку pyspark в экосистему Anaconda. Загрузку следует выполнять также как для библиотек numpy, pandas, matplotlib.

Подключение библиотеки

```
import pyspark
```

```
pyspark.__version__
```

```
'3.1.2'
```

Создание сессии

SparkSession является точкой входа в PySpark. Для создания SparkSession используется метод builder данного объекта.

- getOrCreate() возвращает уже существующий SparkSession; если он не существует, создается новый SparkSession.
- master() задает имя менеджера кластера в качестве аргумента, а в автономном (локальном) режиме используется local[x]. Здесь X - положительное целое число, указывающее, сколько разделов будет создано. Желательно X должно соответствовать количеству ядер ЦП. «*» обозначает использование всех ядер процессора
- appName() задает имя приложения.

Пример создания SparkSession:

```
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder\
        .master("local[*]")\
        .appName('PySpark_Test')\
        .getOrCreate()
```

Загрузка данных

Для загрузки данных используется функция `spark.read`. Данные можно считывать из файлов различных форматов: CSV, JSON, Parquet и других.

В качестве примера загрузим данные о прокате велосипедов в Сеуле января 2017 года по июль 2018 года, которые доступны в репозитории наборов данных для машинного обучения [1]. Скачаем данные (файл `SeoulBikeData` в формате CSV) и положим их в папку на локальном компьютере (например, `d:\data`).

Загрузим данные методом `spark.read.csv` и выведем информацию о схеме данных с помощью метода PrintSchema. Загрузка данных может занять 5-10 минут.

```
data = spark.read.csv(
    'd:\data\SeoulBikeData.csv',
    sep=',',
    header=True,
)

data.printSchema()
```

```
root
 |-- Date: string (nullable = true)
 |-- Rented Bike Count: string (nullable = true)
 |-- Hour: string (nullable = true)
 |-- Temperature(C): string (nullable = true)
 |-- Humidity(%): string (nullable = true)
 |-- Wind speed (m/s): string (nullable = true)
 |-- Visibility (10m): string (nullable = true)
 |-- Dew point temperature(C): string (nullable = true)
 |-- Solar Radiation (MJ/m2): string (nullable = true)
 |-- Rainfall(mm): string (nullable = true)
 |-- Snowfall (cm): string (nullable = true)
 |-- Seasons: string (nullable = true)
 |-- Holiday: string (nullable = true)
 |-- Functioning Day: string (nullable = true)
```

Все данные прочитаны как строки. На самом деле отдельные поля (колонки таблицы) имеют различные типы. Поэтому перед обработкой данных надо определить структуру фрейма данных. Мы можем определить ее с помощью класса StructType, который представляет собой коллекцию объектов полей StructField. Каждый объект поля определяет имя столбца (строка), его тип (DataType), допускает ли он значение NULL (True/False), и необязательное описание данных (метаданные).

Зададим структуру данных в соответствии с описанием набора данных:

```
from pyspark.sql.types import *
data_schema = [
                StructField('Дата', DateType(), True),
                StructField('Кол-во', IntegerType(), True),
                StructField('Время', IntegerType(), True),
                StructField('Температура,C', DoubleType(), True),
                StructField('Влажность,%', IntegerType(), True),
                StructField('Скорость ветра,м/с', DoubleType(), True),
                StructField('Видимость-10м', IntegerType(), True),
                StructField('Темп.точки росы,C', DoubleType(), True),
                StructField('Солн.излуч.,МДж/м2', DoubleType(), True),
                StructField('Осадки, мм', DoubleType(), True),
                StructField('Снег, мм', DoubleType(), True),
                StructField('Сезон', StringType(), True),
                StructField('Праздник', StringType(), True),
                StructField('Раб.время', StringType(), True),
            ]

final_struc = StructType(fields = data_schema)
```

Выполним повторную загрузку с учетом указанной структуры (параметр `schema`)

```
data = spark.read.csv(
    'd:\data\SeoulBikeData.csv',
    sep=',',
    header=True,
    schema=final_struc
)

data.printSchema()
```

```
root
 |-- Дата: date (nullable = true)
 |-- Кол-во: integer (nullable = true)
 |-- Время: integer (nullable = true)
 |-- Температура,C: double (nullable = true)
 |-- Влажность,%: integer (nullable = true)
 |-- Скорость ветра,м/с: double (nullable = true)
 |-- Видимость-10м: integer (nullable = true)
 |-- Температура точки росы,C: double (nullable = true)
 |-- Солнечное излучение - МДж/м2: double (nullable = true)
 |-- Осадки, мм: double (nullable = true)
 |-- Снег, мм: double (nullable = true)
 |-- Сезон: string (nullable = true)
 |-- Праздник: string (nullable = true)
 |-- Рабочее время: string (nullable = true)
```

Методы просмотра данных

Существуют следующие методы просмотра данных:

- схемы данных (методы schema и printSchema),
- типов данных (метод dtypes),
- строк данных (метод show),
- строк данных с именами колонок (метод head),
- первой строки данных (метод first),
- начальных строк (метод take),
- вывод описательной статистики (метод describe),
- список с названием столбцов (метод columns),
- количества строк (метод count),
- количества различных строк (метод distinct),

Рассмотрим методы просмотра на примерах.

Вывод информации о схеме данных

Схема отображает структуру данных (имя и тип столбца, возможность незаполненных значений). Метод `printSchema` был использован выше. Метод `schema` выводит схему в виде списка:

```
data.schema
```

```
StructType(List(StructField(Дата,DateType,true),StructField(Кол-во,IntegerType,
true),StructField(Время,IntegerType,true),StructField(Температура,C,DoubleType,
true),StructField(Влажность,%,IntegerType,true),StructField(Скорость ветра,м/с,
DoubleType,true),StructField(Видимость-10м,IntegerType,true),StructField(Темп.т
очки росы,C,DoubleType,true),StructField(Солн.излуч.,МДж/м2,DoubleType,true),St
ructField(Осадки, мм,DoubleType,true),StructField(Снег, мм,DoubleType,true),Str
uctField(Сезон,StringType,true),StructField(Праздник,StringType,true),StructFie
ld(Раб.время,StringType,true)))
```

Метод `columns` возвращает список, содержащий названия столбцов. Метод `dtypes` возвращает список кортежей с именами столбцов и типами данных:

```
data.dtypes
```

```
[('Дата', 'date'),
 ('Кол-во', 'int'),
 ('Время', 'int'),
 ('Температура,C', 'double'),
 ('Влажность,%', 'int'),
 ('Скорость ветра,м/с', 'double'),
 ('Видимость-10м', 'int'),
 ('Темп.точки росы,C', 'double'),
 ('Солн.излуч.,МДж/м2', 'double'),
 ('Осадки, мм', 'double'),
 ('Снег, мм', 'double'),
 ('Сезон', 'string'),
 ('Праздник', 'string'),
 ('Раб.время', 'string')]
```

Вывод строк набора данных

Метод `head(n)` возвращает n строк в виде списка:

```
data.head(2)
```

```
[Row(Дата='01/12/2017', Кол-во=254, Время=0, Температура,C=-5.2, Влажность,%=3
7, Скорость ветра,м/с=2.2, Видимость-10м=2000, Темп.точки росы,C=-17.6, Солн.из
луч.,МДж/м2=0.0, Осадки, мм=0.0, Снег, мм=0.0, Сезон='Winter', Праздник='No Hol
iday', Раб.время='Yes'),
 Row(Дата='01/12/2017', Кол-во=204, Время=1, Температура,C=-5.5, Влажность,%=3
8, Скорость ветра,м/с=0.8, Видимость-10м=2000, Темп.точки росы,C=-17.6, Солн.из
луч.,МДж/м2=0.0, Осадки, мм=0.0, Снег, мм=0.0, Сезон='Winter', Праздник='No Hol
iday', Раб.время='Yes')]
```

Метод `show()` по умолчанию отображает первые 20 строк, но можно задать в качестве параметра количество выводимых строк:

```
data.show(5)
```

```
+----------+------+-----+------------+----------+-----------------+------------+--------------
--+-----------------+----------+--------+------+----------+---------+
|      Дата|Кол-во|Время|Температура,C|Влажность,%|Скорость ветра,м/с|Видимость-10м|Темп.точки рос
ы,C|Солн.излуч.,МДж/м2|Осадки, мм|Снег, мм| Сезон|  Праздник|Раб.время|
+----------+------+-----+------------+----------+-----------------+------------+--------------
--+-----------------+----------+--------+------+----------+---------+
|01/12/2017|   254|    0|        -5.2|        37|              2.2|        2000|            -1
7.6|              0.0|       0.0|     0.0|Winter|No Holiday|      Yes|
|01/12/2017|   204|    1|        -5.5|        38|              0.8|        2000|            -1
7.6|              0.0|       0.0|     0.0|Winter|No Holiday|      Yes|
|01/12/2017|   173|    2|        -6.0|        39|              1.0|        2000|            -1
7.7|              0.0|       0.0|     0.0|Winter|No Holiday|      Yes|
|01/12/2017|   107|    3|        -6.2|        40|              0.9|        2000|            -1
7.6|              0.0|       0.0|     0.0|Winter|No Holiday|      Yes|
|01/12/2017|    78|    4|        -6.0|        36|              2.3|        2000|            -1
8.6|              0.0|       0.0|     0.0|Winter|No Holiday|      Yes|
+----------+------+-----+------------+----------+-----------------+------------+--------------
--+-----------------+----------+--------+------+----------+---------+
only showing top 5 rows
```

Метод `take(n)` возвращает первые n строк. Метод `first()` возвращает первую строку данных:

```
data.first()
```

```
Row(Дата='01/12/2017', Кол-во=254, Время=0, Температура,C=-5.2, Влажность,%=37,
Скорость ветра,м/с=2.2, Видимость-10м=2000, Темп.точки росы,C=-17.6, Солн.излу
ч.,МДж/м2=0.0, Осадки, мм=0.0, Снег, мм=0.0, Сезон='Winter', Праздник='No Holid
ay', Раб.время='Yes')
```

Метод `count()` возвращает общее число строк в наборе данных:

```
data.count()
```

```
8760
```

Отбор и анализ данных

PySpark имеет методы и функции для формирования запросов на получение данных:

- `select`
- `filter`
- `groupBy`
- `agg`

Метод `Select` используется для выбора одного или нескольких столбцов, используя их имена. Отберем 4 столбца и выведем методом `show` первые 5 строк:

```
data.select(['Дата','Время','Скорость ветра,м/с','Сезон']).show(5)
```

```
+----------+-----+------------------+------+
|      Дата|Время|Скорость ветра,м/с| Сезон|
+----------+-----+------------------+------+
|01/12/2017|    0|               2.2|Winter|
|01/12/2017|    1|               0.8|Winter|
|01/12/2017|    2|               1.0|Winter|
|01/12/2017|    3|               0.9|Winter|
|01/12/2017|    4|               2.3|Winter|
+----------+-----+------------------+------+
only showing top 5 rows
```

Метод `Filter` отбирает (фильтрует) данные на основе заданных условий. Он похож на раздел `Where` в операторе `SELECT` языка SQL реляционных БД. Можно указать несколько условий, используя операторы & (логическое И), |(Логическое ИЛИ) и ~ (логическое отрицание). Например, отберем данные по сезону Winter (зима) и температурой больше 0. Выведем только несколько столбцов и первые 5 найденных строк:

```
from pyspark.sql.functions import col, lit
data.select(['Дата','Влажность,%','Скорость ветра,м/с','Сезон'])\
   .filter((col('Сезон')=='Winter')&(col('Температура,C')>0)).show(5)
```

```
+----------+----------+------------------+------+
|      Дата|Влажность,%|Скорость ветра,м/с| Сезон|
+----------+----------+------------------+------+
|01/12/2017|         23|               1.4|Winter|
|01/12/2017|         25|               1.6|Winter|
|01/12/2017|         26|               2.0|Winter|
|01/12/2017|         36|               3.2|Winter|
|01/12/2017|         54|               4.2|Winter|
+----------+----------+------------------+------+
only showing top 5 rows
```

В фильтре можно задать предикат `Between`, который отбирает значения столбца, принадлежащие указанному диапазону. Отберем данные с влажностью в диапазоне от 30 до 40:

```
data.select(['Дата','Влажность,%','Температура,C','Скорость ветра,м/с'])\
   .filter(data['Влажность,%'].between(30,40)).show(5)
```

```
+----------+----------+-------------+------------------+
|      Дата|Влажность,%|Температура,C|Скорость ветра,м/с|
+----------+----------+-------------+------------------+
|01/12/2017|         37|         -5.2|               2.2|
|01/12/2017|         38|         -5.5|               0.8|
|01/12/2017|         39|         -6.0|               1.0|
|01/12/2017|         40|         -6.2|               0.9|
|01/12/2017|         36|         -6.0|               2.3|
+----------+----------+-------------+------------------+
only showing top 5 rows
```

Метод groupBy позволяет сгруппировать данные по выбранным столбцам и выполняет различные операции, такие как вычисление суммы, среднего, минимального, максимального значения и т. д. Например, выведем максимальную температуру и влажность по сезонам:

```
data.select(['Сезон','Влажность,%','Температура,C'])\
    .groupBy('Сезон')\
    .max()\
    .show()
```

```
+------+----------------+------------------+
| Сезон|max(Влажность,%)|max(Температура,C)|
+------+----------------+------------------+
|Spring|              98|              29.4|
|Summer|              98|              39.4|
|Autumn|              97|              30.5|
|Winter|              97|              10.3|
+------+----------------+------------------+
```

PySpark предоставляет встроенные стандартные функции агрегирования данных по сгруппированным данным. Например, выведем минимальные, максимальные и средние значения температуры и влажности, сгруппировав данные по сезонам:

```
from pyspark.sql import functions as fun
data.groupBy('Сезон')\
   .agg(fun.min('Влажность,%').alias('Мин.влаж.'),
        fun.max('Влажность,%').alias('Макс.влаж.'),
        fun.min('Температура,C').alias('Мин.темп.'),
        fun.max('Температура,C').alias('Макс.темп.'),
        fun.avg('Температура,C').alias('Ср.темп.')
       ).show()
```

```
+------+---------+----------+---------+----------+------------------+
| Сезон|Мин.влаж.|Макс.влаж.|Мин.темп.|Макс.темп.|          Ср.темп.|
+------+---------+----------+---------+----------+------------------+
|Spring|        0|        98|     -6.6|      29.4| 13.046693840579712|
|Summer|       21|        98|     16.3|      39.4| 26.582789855072505|
|Autumn|       13|        97|     -3.0|      30.5|   14.1208333333333|
|Winter|       14|        97|    -17.8|      10.3|-2.5404629629629607|
+------+---------+----------+---------+----------+------------------+
```

PySpark — отличный инструмент для специалистов по данным, поскольку он обеспечивает масштабируемый анализ

Задание

1. Найти в репозитории наборов данных для машинного обучения[2] набор данных. Загрузить его в среду Python.
2. Выполнить основные операции просмотра, отбора и анализа данных

Список источников

1. Набор данных о прокате велосипедов.- https://archive.ics.uci.edu/ml/machine-learning-databases/00560/
2. Репозиторий наборов данных.- https://archive.ics.uci.edu/ml/index.php

Порядок выполнения работы

1. Изучить теоретический материал
2. Выполнить пункты задания

Содержание отчета

1. Описание наборов данных для загрузки
2. Результаты выборки загруженных данных
3 Результаты обработки загруженных данных: просмотра, отбора и анализа

ч.4 Введение в машинное обучение

В данной части сборника рассмотрены базовые понятия, методы и средства машинного обучения, Программные средства анализа данных и машинного обучения. Приведены примеры решения прикладных задач в области обработки и анализа больших данных с использованием средств машинного обучения

11 класс 2-е полугодие

Оборудование для проведения занятий

1. Рабочая станция ученика (Intel i5, 8Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)
2. Рабочая станция учителя (Intel i7, 16Гб ОЗУ, SSD 500Гб, видеокарта Radeon RX VEGA 56 или аналогичная с 8Гб видеопамяти, монитор с разрешением 1920x1080, клавиатура, мышь)

Программное обеспечение для проведения занятий (в том числе системное ПО)

1. ОС Windows 10
2. MS Office 2016
3. PyCharm CE (Anaconda)
4. Python 3.7

Рекомендации учителю по проведению занятий.

Практические занятия данного цикла связаны с решением типовых задач средствами машинного обучения.

В рамках практических занятий рассматриваются методические рекомендации по решению типовых задач. Для программной реализации методов машинного обучения используется Python и дополнительные высокоуровневые библиотеки.

В первой части данного цикла используется библиотека `scikit-learn`. Она использовалась на одном из занятий второго цикла. Если же она не была установлена, то описание процесса установки дано в описании практического занятия 43.

В задачах с использованием нейросетевых технологий используются еще две высокоуровневые библиотеки `tensorflow` и `keras`. Последние версии `tensorflow` уже содержат внутри пакета и библиотеку `keras`. Поэтому достаточно будет установить только библиотеку `tensorflow`.

Все практические занятия включают краткие теоретические сведения по теме занятия, типовые задачи с разбором их решения, вопросы для проверки усвоения материала.

В начале занятия необходимо проверить выполнение заданий для самостоятельного выполнения, выданных на предыдущем практическом занятии.

В ходе проведения занятия рекомендуется изложить учащимся краткие теоретические сведения, разобрать решение типовых задач, описанных в практических занятиях. Для усвоения материала необходимо выдать задания для самостоятельной проработки материала, разобранного на практическом занятии

Практическое занятие № 42
Введение в машинное обучение

Цель занятия. Получить представление о парадигме машинного обучения, составляющих машинного обучения, основной задачи машинного обучения

Краткие теоретические сведения

Область машинного обучения возникла из вопроса: может ли компьютер выйти за рамки того, «что мы и сами знаем, как выполнять», и самостоятельно научиться решать некоторую определенную задачу? Может ли компьютер без помощи программиста, задающего правила обработки данных, автоматически определить эти правила, исследуя данные?

Этот вопрос открыл двери в новую парадигму программирования. В классическом программировании, в парадигме символического ИИ, люди вводят правила (программу) и данные для обработки в соответствии с этими правилами и получают ответы (рис.1). В машинном обучении люди вводят данные и ответы, соответствующие этим данным, а на выходе получают правила. Эти правила затем можно применить к новым данным для получения оригинальных ответов.

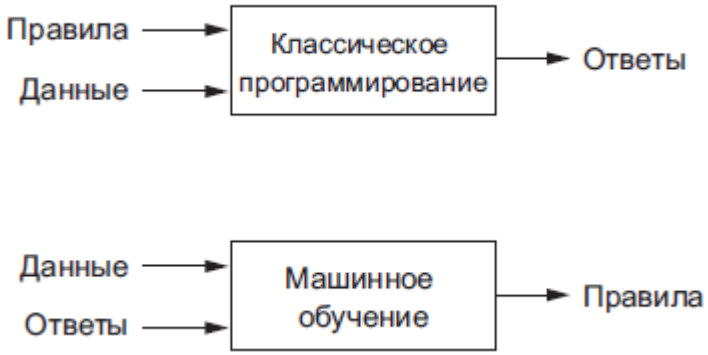


Рис. 1. Машинное обучение: новая парадигма программирования

В машинном обучении система *обучается*, а не программируется явно. Ей передаются многочисленные примеры, имеющие отношение к решаемой задаче, а она находит в этих примерах статистическую структуру, которая позволяет системе выработать правила для автоматического решения задачи. Например, чтобы автоматизировать задачу определения фотографий, сделанных в отпуске, можно передать системе машинного обучения множество примеров фотографий, уже классифицированных людьми, и система изучит статистические правила классификации конкретных фотографий.

Расцвет машинного обучения начался только в 1990-х, но эта область быстро превратилась в наиболее популярный и успешный раздел ИИ, и эта

тенденция была подкреплена появлением более быстродействующей аппаратуры и огромных наборов данных. Машинное обучение тесно связано с математической статистикой, но имеет несколько важных отличий. В отличие от статистики, машинное обучение обычно имеет дело с большими и сложными наборами данных (например, состоящими из миллионов фотографий, каждая из которых состоит из десятков тысяч пикселов), к которым практически невозможно применить классические методы статистического анализа, такие как байесовские методы. Как результат, машинное и в особенности глубокое обучение не имеют мощной математической платформы и основываются почти исключительно на инженерных решениях. Это практическая дисциплина, в которой идеи чаще доказываются эмпирически, а не теоретически.

Для машинного обучения нам нужны три составляющие:

– *Контрольные входные данные* — например, в задаче распознавания речи такими контрольными входными данными могут быть файлы с записью речи разных людей, а в задаче классификации изображений - сами изображения;

– *Примеры ожидаемых результатов* — в задаче распознавания речи это могут быть транскрипции звуковых файлов, составленные людьми, а в задаче классификации изображений ожидаемым результатом могут быть метки, такие как «собака», «кошка» и другие;

– *Способ оценки качества работы алгоритма* — это необходимо для определения, насколько далеко отклоняются результаты, возвращаемые алгоритмом, от ожидаемых. Оценка используется как сигнал обратной связи для корректировки работы алгоритма. Этап корректировки и называют обучением.

Модель машинного обучения трансформирует исходные данные в значимые результаты, «обучаясь» на известных примерах входных данных и результатов. То есть главной задачей машинного обучения является *значимое преобразование данных*, или, иными словами, обучение *представлению* входных данных, приближающему к ожидаемому результату. Давайте определим, что называют представлением данных? По сути, это другой способ *представления*, или *кодирования*, данных. Например, цветное изображение можно закодировать в формате RGB (red-green-blue — красный-зеленый-синий) или HSV (hue-saturation-value — тон-насыщенность-значение). Это два разных представления одних и тех же данных. Некоторые задачи трудно решаются с данными в одном представлении, но легко — в другом. Например, задача «выбрать все красные пикселы в изображении» легче решается с данными в формате RGB, тогда как задача «сделать изображение менее насыщенным» проще решается с данными в формате HSV. Главная задача моделей машинного обучения как раз и заключается в поиске такого представления входных данных (преобразований к этому представлению), которое сделает данные более пригодными для решения

задачи анализа данных, например, классификации изображений.

Рассмотрим конкретный пример. Имеется система координат с осями X и Y и несколько точек (белых и черных кружочков) в этой системе координат (x, y), как показано на рис. 2. Нужно разработать алгоритм, принимающий координаты (x, y) точки и возвращающий наиболее вероятный цвет, черный или белый. В данном примере:

– исходными данными являются координаты точек;

– результатом является цвет точек;

– мерой качества алгоритма может быть, например, процент правильно классифицированных точек.

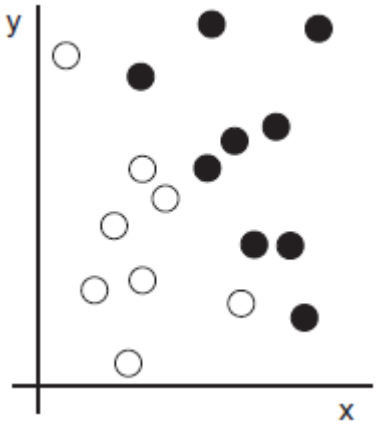


Рис. 2. Пример некоторых данных

В данном примере нужно получить новый способ представления исходных данных, позволяющий четко отделять белые точки от черных. Таким преобразованием, может быть изменение системы координат, как показано на рис. 3: смещение центра и поворот.

Рис. 3. Изменение системы координат

Координаты точек в этой новой системе координат можно назвать новым представлением данных, которое позволяет привести задачу классификации данных «черный/белый» к простому правилу: «черные точки имеют координату x > 0» или «белые точки имеют координату x < 0».

В данном примере изменение координат проводилось вручную. Но если реализовать систематический поиск разных вариантов изменения системы координат с использованием процента правильно классифицированных точек в качестве обратной связи, мы получим версию машинного обучения. Обучение модели описывает автоматический процесс поиска лучшего представления.

Все алгоритмы машинного обучения заключаются в автоматическом поиске таких преобразований, которые смогут привести данные к виду, более пригодному для решения конкретной задачи. Алгоритмы машинного просто выполняют поиск в предопределенном наборе операций, который называют *пространством гипотез*.

Таким образом, машинное обучение — это поиск значимого представления некоторых входных данных в предопределенном пространстве возможностей с использованием сигнала обратной связи. Эта простая идея позволяет решать чрезвычайно широкий круг интеллектуальных задач: от распознавания речи до беспилотного автомобиля.

Машинное обучение принято разделять на три вида (категории):

- обучение с учителем (контролируемое обучение);
- обучение без учителя (неконтролируемое обучение);
- обучение с подкреплением.

В системах обучения с учителем данные, подготовленные для анализа, изначально содержат правильный ответ, поэтому цель алгоритма — не настроить параметры модели, чтобы ответы формируемые системой были близки к правильным ответам. Результатом становится способность выстраивать корректные прогнозы и модели.

В системах обучения без учителя данные не имеют правильных ответов. Задачей алгоритма в этом случае выявить закономерности, содержащиеся в данных. На следующем этапе на основе выявленных закономерностей система интерпретирует и систематизирует данные.

В системах обучения с подкреплением принципы обучения из психологических экспериментов. Система пытается найти оптимальные действия и учится на ошибках. После каждой попытки системе подается положительный сигнал (награда) или отрицательный сигнал, в зависимости от полученного результата (правильного или ошибки). В конечном итоге система будет настраиваться, чтобы чаще достигать положительного результата.

Контрольные вопросы

1. Объясните различие парадигмы машинного обучения от стандартной процедуры решения задачи

2. Какие из перечисленных задач можно отнести к машинному обучению:

- определение спама по содержанию письма
- решение системы линейных алгебраических уравнений
- кредитный скоринг: по анкете заемщика принять решение о выдаче/отказе кредита
- распознавание лиц и других паттернов на изображениях
- сборка автомобиля на конвейере;
- медицинская диагностика
- выполнение программы компьютером
- выявление эмоциональной окраски текста

3. Какой процесс называют обучением ?

4. Назовите основные составляющие машинного обучения.

5. Укажите основные виды машинного обучения и их отличительные особенности.

6. Укажите, к какому виду машинного обучения относятся следующие задачи:

- распознавание лиц;
- определение покупательских предпочтений;
- группировка клиентов компании по их характеристикам;
- поведение игрока в видеоигре;
- определение спама в почтовых отправлениях;
- выявление эмоциональной окраски текста;
- поиск роботом пути в лабиринте.

Порядок выполнения работы

1. Изучить теоретический материал
2. Ответить на контрольные вопросы и обосновать эти ответы

Содержание отчета

1.Ответы на контрольные вопросы с обоснованием

Практическое занятие № 43
Логистическая регрессия на Python

Цель занятия. Получить представление о задачах классификации, методе классификации логистической регрессии. Освоить средства Python для решения задачи классификации

Краткие теоретические сведения

Классификация — это задача из области анализа данных, в которой необходимо отнести некоторый объект к определенному классу или категории на основе характеристик этого объекта.

Выделяют задачи:

– бинарной классификации, когда необходимо выбрать принадлежность к одному из двух классов. В качестве примера можно привести задачу скорринга, когда на основе анкетных данных и кредитной истории рассчитывается платежоспособность клиента и принимается решение о выдаче (класс 1) или невыдаче (класс 0) кредита.

– мультиклассовой классификации, когда необходимо разбить объекты на 3 и более класса. В качестве примера можно привести задачу определения по содержанию текста его эмоциональной окраски: положительной (класс 1), отрицательной (класс 2) или нейтральной (класс 3).

Логистическая регрессия является одним из основных методов классификации [1]. Он принадлежит к группе линейных классификаторов, так как моделью классификатора по сути является прямая линия, разделяющая область признаков на две части.

Важное место в понимании механизма логистической регрессии занимают функции сигмоиды и натурального логарифма.

Функция сигмоида представляет S-образную кривую (рис.1) от некоторой переменной x.

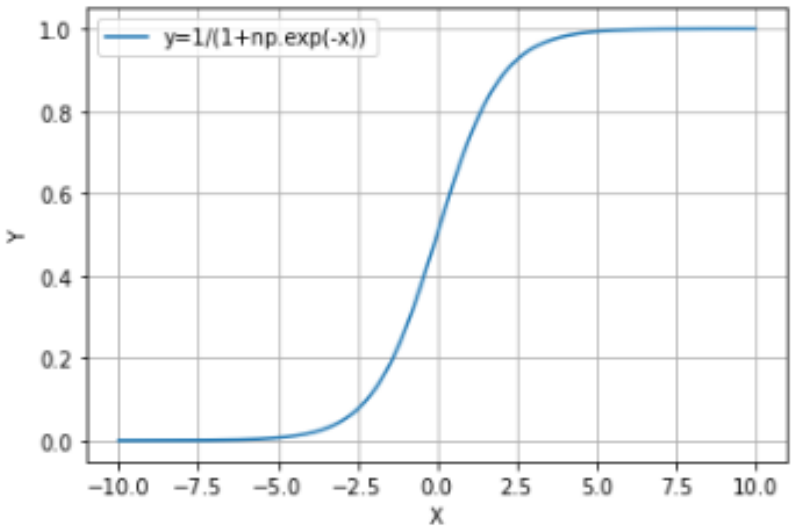


Рис.1 Функция сигмоиды

Сигмоида в большей части своей области имеет значения, очень близкие к 0 или 1. Это позволяет использовать ее в задаче классификации.

Функция логарифма на отрезке от 0 до 1 для аргументов x и 1-x показана на рис.2

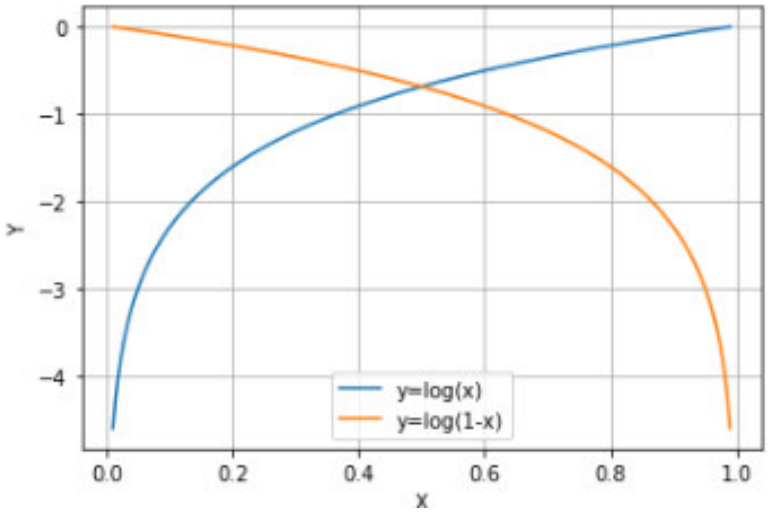


Рис.2 Функция логорифма

Когда аргумент x приближается к нулю, функция натурального логарифма стремится к отрицательной бесконечности, при x = 1, логарифм равен 0. Для аргумента 1-x все наоборот.

Цель метода классификации — найти функцию логистической регрессии p(x) такую, чтобы значения p(xi) для каждого наблюдения i были как можно ближе к фактическому значению yi, которые могут быть только 0 или 1. Поэтому здесь хорошо подходит функция сигмоиды.

Логистическая регрессия использует линейную функцию

$f(x) = b_0 + b_1x_1 + \ldots + b_кx_к$

Переменные b_0, b_1, b_k являются оценками коэффициентов регрессии,

Функция логистической регрессии p(x) является сигмоидой:

$$p(x) = \frac{1}{(1 - e^{-x})}$$

Необходимо определить наилучшие коэффициенты b_j (j=0,…k), чтобы функция сигмоиды была как можно ближе ко всем значениям y_i (i = 1,…,n). Процесс вычисления оптимальных весов по имеющимся наблюдениям называется обучением модели.

Для получения оптимальных весов используют метод максимального правдоподобия, который представляется уравнением.

$$LLF = \sum_{i=1}^{n} (y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i)))$$

Когда $y_i=0$, LLF равен $\log(1—p(x_i))$. Данная функция стремится к нулю, если $p(x_i)$ близко к $y_i=0$. Этого мы и добиваемся. И, наоборот, когда $y_i=1$ LLF равен $\log(p(x_i))$. И здесь, если $p(x_i)$ близко к $y_i=1$, то $\log(p(x_i))$ близко к 0. Это тоже нам подходит.

После определения оптимальных весов мы получаем функцию, хорошо аппроксимирующую функцию сигмоиды.

Бинарная классификация может дать следующие результаты:

- Истинно отрицательные: правильно предсказанные нули;
- Истинно положительные: правильно предсказанные единицы;
- Ложно отрицательные: неверно предсказанные нули;
- Ложно положительные: неверно предсказанные единицы.

Обычно точность классификации оценивают путем сравнения фактических и расчетных результатов, подсчитывая правильные и неправильные предсказания.

Самый простой показатель точности классификации равен отношению количества правильных предсказаний к общему количеству наблюдений. Другие показатели определяются следующим образом:

- Прогнозирующая ценность положительного результата — это отношение количества истинных положительных результатов к сумме количества истинных и ложных положительных результатов.
- Прогнозирующая ценность отрицательного результата — это отношение количества истинно отрицательных результатов к сумме количества истинных и ложных отрицаний.
- Чувствительность— это отношение количества истинных положительных результатов к количеству фактических положительных результатов.
- Специфичность— это отношение количества истинных негативов к количеству реальных негативов.

Критерий оценки классификатора выбирают исходя из поставленной задачи.

Рассмотрим бинарную классификацию по одной переменной на следующем примере (таблица 1)

Таблица 1. Данные для бинарной классификации

x	0	1	2	3	4	5	6	7	8	9
y	0	0	0	0	1	1	1	1	1	1

Для построения логистической регрессии на Python используют библиотеку numpy, класс `LogisticRegression` библиотеки `scikit-learn` [2], а для визуализации решения - библиотеку matplotlib.

Откроем новый блокнот и приступим к выполнению работы.

Если ранее не была установлена библиотека `scikit-learn`, выполним установку сейчас. Это можно сделать через Anaconda (как описано в практическом занятии 19), а можно в Jupyter Notebook. Для этого необходимо в файле нового блокнота ввести команду:

```
!pip install scikit-learn
```

```
!pip install scikit-learn

Collecting scikit-learn
  Downloading scikit_learn-1.0.1-cp38-cp38-win_amd64.whl (7.2 MB)
Collecting joblib>=0.11
  Downloading joblib-1.1.0-py2.py3-none-any.whl (306 kB)
Requirement already satisfied: numpy>=1.14.6 in e:\programs\anaconda3\envs\bigdat
(1.21.2)
Requirement already satisfied: scipy>=1.1.0 in e:\programs\anaconda3\envs\bigdata
7.1)
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.0.0-py3-none-any.whl (14 kB)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.1.0 scikit-learn-1.0.1 threadpoolctl-3.0.0
```

Далее импортируем необходимые библиотеки в файл, открытый в блокноте.

```
import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
```

Теперь создадим массивы входной и выходной переменных:

```
x=np.arange(10)
y=np.array([0,0,0,0,1,1,1,1,1,1])
```

Преобразуем массив `x` в вектор-столбец методом изменения формы `rehape`:

```
x=x.reshape(-1,1)
```

```
print("x:",x)
print("y:",y)
```

```
x: [[0]
 [1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
y: [0 0 0 0 1 1 1 1 1 1]
```

На следующем шаге создадим модель логистической регрессии. Для этого создадим модель, являющуюся экземпляром класса `LogisticRegression`:

```
model=LogisticRegression(random_state=0)
```

При создании модели был использован дополнительный параметр `random_state`, определяющий принцип использования генератора случайных чисел. Дополнительно необходимо задать 2 параметра

- `solver` - определяет, какой решатель используется для подбора модели (по умолчанию `liblinear`);
- `penalty` - определяет использование регуляризации для борьбы с переобучением (по умолчанию используется регуляризатор `l2`).

Решатель `liblinear` всегда должен использовать регуляризатор. Оставим значения этих параметров по умолчанию.

После того, как определена структура модели, перейдем к ее обучению, вызвав метод модели `fit`, передав ей в качестве параметров массивы входной и выходной переменной.

Получим из модели ее коэффициенты: атрибут `intercept_` возвращает свободный коэффициент, а атрибут `coef_` - коэффициент при входной переменной:

```
print('b0=', np.round(model.intercept_,3))
print('b1=', np.round(model.coef_,3))
```

```
b0= [-4.126]
b1= [[1.181]]
```

Как только модель определена, можно проверить ее точность с помощью метода модели `predict_proba()`, который возвращает матрицу вероятностей того, что прогнозируемый результат равен нулю или единице (вероятности округлим до 2 знаков после запятой):

```
np.round(model.predict_proba(x),2)
```

```
array([[0.98, 0.02],
       [0.95, 0.05],
       [0.85, 0.15],
       [0.64, 0.36],
       [0.35, 0.65],
       [0.14, 0.86],
       [0.05, 0.95],
       [0.02, 0.98],
       [0.  , 1.  ],
       [0.  , 1.  ]])
```

Можно также получить фактические прогнозы с помощью метода модели `predict()`:

```
model.predict(x)
```

```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

Так как 10 из 10 наблюдений классифицированы правильно, точность модели составляет 10/10=1. Эту оценку можно также получить, вызвав метод `score`:

```
model.score(x,y)
```

```
1.0
```

Можно также получить больше информации о точности модели с помощью матрицы ошибок. В случае бинарной классификации матрица ошибок показывает следующие числа:

- истинные минусы в верхнем левом углу;
- ложные минусы в нижнем левом углу;
- ложные плюсы в правом верхнем углу;
- истинные плюсы в правом нижнем углу.

Для создания матрицы ошибок можно использовать функцию `confusion_matrix()`, которой надо передать в качестве аргументов фактические и прогнозные значения:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y,model.predict(x))
```

```
array([[4, 0],
       [0, 6]], dtype=int64)
```

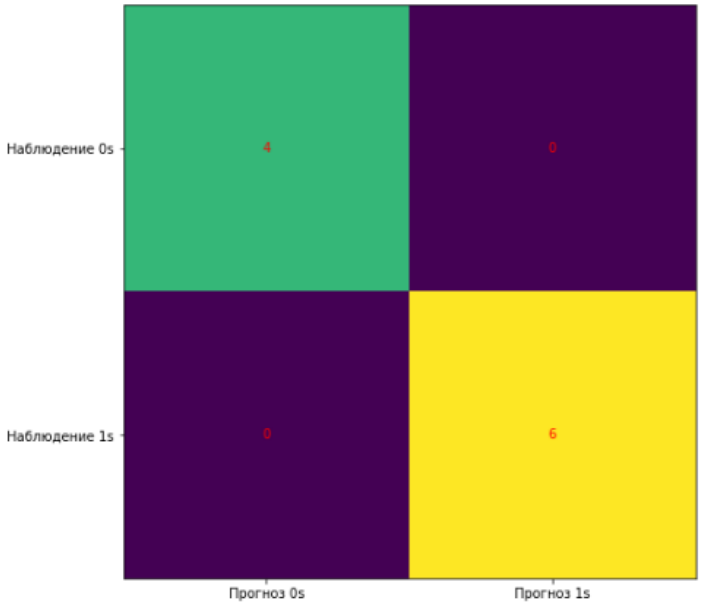
Полученная матрица показывает следующее:

- 4 истинно отрицательных прогноза: первые четыре наблюдения — это правильно предсказанные нули.
- Никаких ложноотрицательных прогнозов
- Никаких ложноположительных прогнозов
- 6 истинно положительных предсказаний: последние шесть наблюдений предсказаны правильно.

Часто бывает полезно визуализировать матрицу ошибок. Это можно сделать с помощью функции `imshow()` из библиотеки matplotlib, которая принимает матрицу ошибок в качестве аргумента:

```
cm = confusion_matrix(y, model.predict(x))
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Прогноз 0s', 'Прогноз 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Наблюдение 0s', 'Наблюдение 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
plt.show()
```

Будет выведено



Задание

1. Задайте свои данные для классификации
2. Постройте модель, обучите ее и оцените качества предсказания

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.
2. Класс логистической регрессии в scikit-learn .- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить пункты задания

Содержание отчета

1. Программный код (блокнот) решения задачи классификации методом логистической регрессии
2. Программный код (блокнот) выполнения пунктов задания

Практическое занятие № 44
Метрические методы классификации.
Метод ближайших соседей

Цель занятия. Получить представление о метрических методах и алгоритмах решения задач классификации. Научиться применять метод ближайших соседей решений в задачах классификации. Освоить средства Python для реализации метрических алгоритмов решения задачи классификации

Краткие теоретические сведения

Продолжим рассмотрение методов классификации. На данном занятии рассмотрим один из популярных метрических методов решения задачи классификации – метод ближайших соседей (KNN - k-nearest neighbor).

Данный метод определяет принадлежность нового объекта к тому или иному классу по ближайшему окружению (по классам ближайших соседей).

Алгоритм KNN является довольно простым и включает следующие шаги:

1. Выбрать число k и метрику расстояния.
2. Найти k ближайших соседей образца (объекта), который надо классифицировать.
3. Присвоить метку класса путем мажоритарного голосования (большинством голосов).

На рисунке 1 показано, как определяется класс объекта на основе мажоритарного голосования [1].

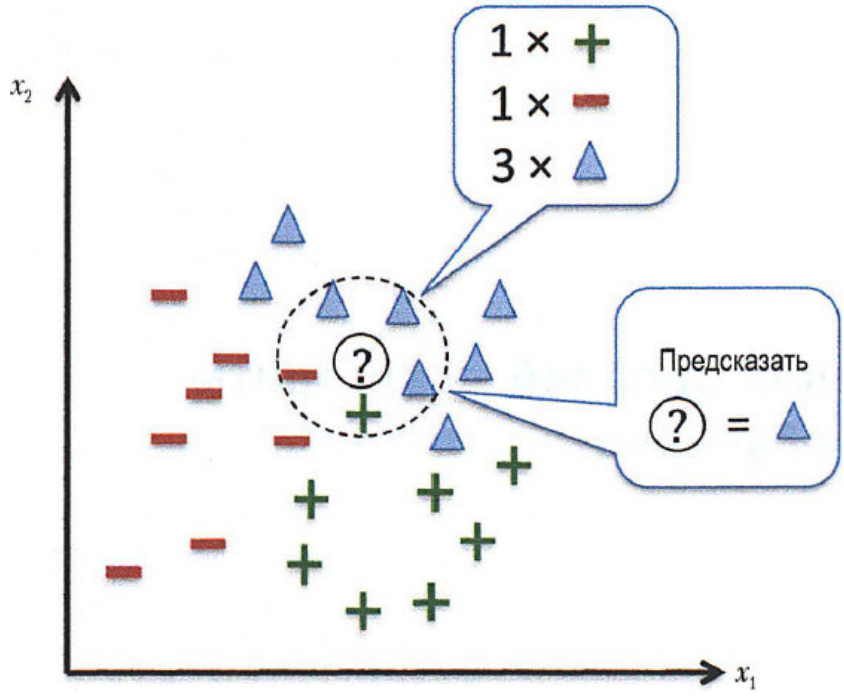


Рис.1 Определение класса объекта методом мажоритарного голосования

Алгоритм KNN, выбрав метрику расстояний, находит в наборе данных k образцов (объектов), которые являются самыми близкими к нему (самыми похожими на него). Метка класса новой точки данных затем определяется мажоритарным голосованием среди его k ближайших соседей.

Основное преимущество такого подхода состоит в том, что классификатор немедленно адаптируется по мере сбора новых тренировочных данных.

Однако его оборотная сторона: вычислительная сложность классифицирования новых образцов растет линейно вместе с числом образцов в тренировочном наборе данных.

Качество классификации методом KNN зависит от нескольких параметров и, прежде всего, от:

- числа соседей k
- метрики расстояния между объектами

Величину k часто определяют опытным путем или исходя из анализа набора данных.

Существует несколько видов метрик близости (расстояний). Часто используются такие метрики расстояний: евклидово расстояние, манхэттенское расстояние (часто называемое расстоянием городских кварталов) и расстояние Минковского. Расстояние Минковского:

$$d(x^{(i)}, x^{(j)}) = \sqrt[p]{\sum_k \left| x_k^{(i)} - x_k^{(j)} \right|^p}$$

является обобщением евклидова и манхэттенского расстояния При p=2 оно становится евклидовым расстоянием, а при p=1 - манхэттенским расстоянием.

Метрику расстояния выбирают исходя из признаков в наборе данных. Для образцов с вещественными значениями, нередко используется простая евклидова мера расстояния. При использовании большинства метрик значения признаков надо масштабировать (приводить к одному масштабу). Иначе признак с диапазоном изменения значений в тысячу единиц будет доминировать (вносить больший вклад) над признаком с диапазоном изменения сотни или десятка единиц.

Для построения дерева решений будем использовать библиотеку `scikit-learn`. Создадим новый блокнот и вначале импортируем необходимые библиотеки для анализа данных на графике:

```
import numpy as np
import matplotlib.pyplot as plt
```

Будем использовать известный набор данных цветков ириса. Этот набор данных часто используется для тестирования алгоритмов классификации машинного обучения и имеется в библиотеке scikit-learn.

```
from sklearn import datasets
# Загрузка данных
iris=datasets.load_iris()
```

Набор содержит 150 образцов цветков ириса трех сортов. В качестве признаков будем использовать длину и ширину лепестков. Создадим массив признаков `x` (3 и 4 колонки загруженного набора данных) и массив меток класса (сортов цветка) `y`.

```
# Массив признаков
x = iris.data[:,[2,3]]
# Массив меток класса
y = iris.target
```

```
x.shape # Количество образцов
```

```
(150, 2)
```

```
np.unique(y) # Количество классов
```

```
array([0, 1, 2])
```

В машинном обучении принято делить исходный набор данных на обучающий и проверочный наборы. На первом наборе модель обучается, а на втором проверяется ее точность. При помощи функции `train_test_split` из модуля model_selection библиотеки scikit_learn произвольным образом разделим массивы `x` и `y` на обучающие данные в размере 70% (105 образцов) и контрольные данные в размере 30% (45 образцов) от общего объема.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
   x,y,test_size=0.3,random_state=1, stratify=y)
```

`Функция train_test_split` перед разделением набора данных перемешивает строки исходного набора, так как образцы цветков одного вида в исходном наборе располагаются группами. Если этого не сделать, то в обучающий набор попадут образцы только двух классов из трех. Параметр `random_state=1` обеспечивает одинаковое распределение при повторном решении задачи. А параметр `stratify=y` обеспечивает пропорциональное соотношение образцов каждого класса, как и в исходном наборе. В этом можно убедиться, вызвав функцию `np.bincount`, которая подсчитывает количество образцов каждого класса.

```
print('Количество объектов в y:', np.bincount(y))
print('Количество объектов в y_train:', np.bincount(y_train))
```

```
Количество объектов в y: [50 50 50]
Количество объектов в y_train: [35 35 35]
```

Далее необходимо выполнить подготовку данных. Многие алгоритмы машинного обучения в целях улучшения качества требуют приведение

признаков к одному масштабу. Выполним стандартизацию признаков, воспользовавшись для этого классом `StandardScaler` из модуля `preprocessing` библиотеки `scikit-learn`. При стандартизации из значения каждого признака вычитают среднее и полученную разность делят на стандартное отклонение.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(x_train)
x_train_std = sc.transform(x_train)
x_test_std = sc.transform(x_test)
```

Для построения модели ближайших соседей используют класс `KNeighborsClassifier` библиотеки scikit-learn [2].

Создадим и обучим модель. В качестве меры расстояния выберем евклидово (Минковского с параметром p=2). Правильный выбор числа k крайне важен для нахождения хорошего равновесия между переобучением и недообучением. Зададим параметр k=5:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(x_train_std, y_train)
```

Визуализируем полученную границу разделения классов. Для этого понадобится функция для построения границ `plot_decision_regions` из библиотеки `mlxtend`. Инсталлируем библиотеку, используя команду:

```
!pip install mlxtend
```

```
!pip install mlxtend
```

```
Collecting mlxtend
  Downloading mlxtend-0.19.0-py2.py3-none-any.whl (1.3 MB)
Requirement already satisfied: setuptools in e:\programs\anaconda3
Requirement already satisfied: scipy>=1.2.1 in e:\programs\anacond
Requirement already satisfied: numpy>=1.16.2 in e:\programs\anacon
2)
Requirement already satisfied: scikit-learn>=0.20.3 in e:\programs
(1.0.1)
Requirement already satisfied: matplotlib>=3.0.0 in e:\programs\an
4.3)
Requirement already satisfied: joblib>=0.13.2 in e:\programs\anacc
0)
Requirement already satisfied: pandas>=0.24.2 in e:\programs\anacc
4)
Requirement already satisfied: pillow>=6.2.0 in e:\programs\anacon
0->mlxtend) (8.4.0)
Requirement already satisfied: python-dateutil>=2.7 in e:\programs
b>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in e:\programs\ana
3.0.0->mlxtend) (3.0.4)
Requirement already satisfied: cycler>=0.10 in e:\programs\anacond
->mlxtend) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in e:\programs\an
3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: six in e:\programs\anaconda3\envs\b
3.0.0->mlxtend) (1.16.0)
Requirement already satisfied: pytz>=2017.3 in e:\programs\anacond
lxtend) (2021.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in e:\programs
arn>=0.20.3->mlxtend) (3.0.0)
Installing collected packages: mlxtend
Successfully installed mlxtend-0.19.0
```

Выполним построение границ обученного дерева решений. В функцию передаем обучающие данные, метки классов (их надо преобразовать к целому типу) и обученное дерево. Границы решений приведены на рис.2.

```
from mlxtend.plotting import plot_decision_regions
plot_decision_regions(x_combined_std,y_combined,clf=knn)
plt.xlabel('длина лепестка [станд.]')
plt.ylabel('ширина лепестка [станд.]')
plt.show()
```

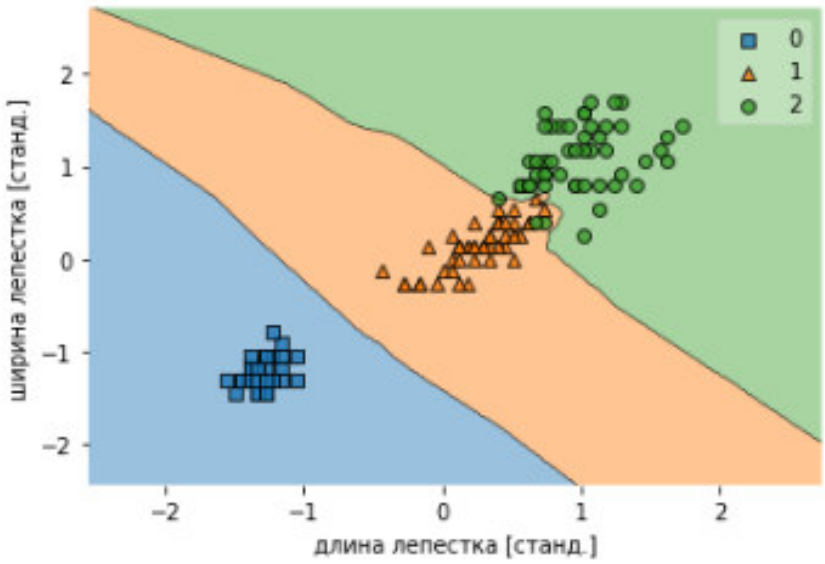


Рис.2 Границы классификатора KNN

Оценим точность модели на обучающих данных, используя метрику `accuracy_score(реальные значения, предсказание)`.

```
from sklearn.metrics import accuracy_score
# Предсказание по модели на обучающих данных
yp=knn.predict(x_train_std)
# Точность прогноза
score = accuracy_score(y_train, yp)
print(np.round(score,5))
```

```
0.97143
```

Также оценим точность на проверочных данных

```
from sklearn.metrics import accuracy_score
# Предсказание по модели на проверочных данных
yp=knn.predict(x_test_std)
# Точность прогноза
score = accuracy_score(y_test, yp)
print(np.round(score,5))
```

```
1.0
```

Как видно точность предсказания довольно высокая, а на проверочных данных - 100%.

Задание

1. Проведите эксперимент с моделью KNN

а) измените параметр k

б) выберите другую метрику

2. Проведите сравнительный анализ полученных результатов

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.
2. Класс метода KNN в scikit-learn .- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить эксперимент с моделью в соответствии с заданием
4. Выполнить сравнительный анализ полученных результатов

Содержание отчета

1. Программный код (блокнот) решения задачи классификации методом ближайших соседей
2. Результаты эксперимента с моделью и выводы сравнительного анализа

Практическое занятие № 45
Логические методы классификации.
Деревья принятия решений

Цель занятия. Получить представление о логических методах и алгоритмах решения задач классификации. Научиться применять деревья принятия решений в задачах классификации. Освоить средства Python для реализации логических алгоритмов решения задачи классификации

Краткие теоретические сведения

Продолжим рассмотрение методов классификации. На данном занятии рассмотрим один из популярных логических методов решения задачи классификации – дерева принятия решений. Из самого названия метода следует, что для разделения данных на классы (точнее стоящих за ними объектов) используется последовательность вопросов. Деревья решений и само принятие решений используются в повседневной жизни в самых разных областях человеческой деятельности. В качестве простого примера можно привести дерево принятия решения для определения вида занятий человека в отдельно взятый день, представленный на рис.1 [1].

Рис.1 Пример дерева принятия решений

При использовании метода выбора решения движение начинают с корня дерева и на каждом шаге данные разбивают по признаку, который ведет к самому большому приросту информации. Данная процедура продолжается, пока мы не дойдем до класса объектов. На практике может образоваться очень глубокое дерево со многими узлами, что легко может привести к переобучению. В силу этого дерево обычно подрезается путем установления предела для его максимальной глубины.

Дерево решений представляет набор логических правил вида: «значение признака соответствует, больше либо меньше некоторого значения", которые объединяются в иерархическую древовидную структуру. Преимуществом деревьев решений является их легкая интерпретируемость и понятность человеку.

Для того чтобы разделить узлы в самых информативных признаках, нужно определить целевую функцию. Целевую функцию надо оптимизировать в результате обучения. В задаче дерева принятия решений целевой функцией будет прирост информации при каждом разделении объектов и ее надо максимизировать. В деревьях решений обычно используются три критерия разделения: мера неоднородности Джини, энтропия и ошибка классификации. Все критерии при построении целевой функции используют меру доли объектов, которые принадлежат каждому классу в узле разделения. На практике ошибка классификации почти не используется, а неоднородность Джини и энтропия работают почти одинаково.

Рассмотрим пример применения дерева решений для искусственно полученных данных. Создадим 2 класса объектов, которые получим путем генерации чисел из двух нормальных распределений с разными средними и одинаковыми дисперсиями.

Для построения дерева решений будем использовать библиотеку `scikit-learn`. Создадим новый блокнот и вначале импортируем необходимые библиотеки для анализа данных на графике:

```
import numpy as np
import matplotlib.pyplot as plt
```

Создадим объекты первого класса, сгенерировав 200 нормально распределенных случайных чисел со средним = 0 (параметр `loc`) и стандартным отклонением = 1 (параметр `scale`). Метками объектов класса будет массив нулей.

```
# Создание первого класса
np.seed = 7
train_data_1 = np.random.normal(size=(200, 2),loc=0,scale=1)
train_labels_1 = np.zeros(200)  # Метка класса =0
```

Создадим объекты второго класса, сгенерировав 100 нормально распределенных случайных чисел со средним = 2 и стандартным отклонением = 1. Метками объектов класса будет массив единиц.

```
# Создание второго класса
train_data_2 = np.random.normal(size=(100, 2), loc=2, scale=1)
train_labels_2 = np.ones(100) # Метка класса =1
```

Объединим объекты в один массив обучающих данных и их меток. Для обучающих данных используем метод `vstack`, а для меток - `hstack`.

```
# Объединим массивы двух классов
train_data = np.vstack((train_data_1,train_data_2))
train_labels = np.hstack((train_labels_1,train_labels_2))
```

Проверим форму массивов исходных данных:

```
print(train_data.shape)
print(train_labels.shape)
```

```
(300, 2)
(300,)
```

Построим график разброса объектов (каждый объект задается координатами x и y). Задача классификации состоит в нахождении границы, разделяющей объекты 2 классов (на графики кружки разных цветов). В нашем случае это будет простая прямая линии (в многомерном пространстве гиперплоскость). График разброса объектов приведен на рис.2.

```
import matplotlib.pyplot as plt
```

```
plt.rcParams['figure.figsize'] = (10,8) # размер грфика,в дюймах(ширина,высота)
plt.scatter(train_data[:, 0], train_data[:, 1], # координаты объектов
            c=train_labels, # цвет объекта опред. меткой
            s=100, # размер маркера (объекта)
            cmap='autumn',       # цветовая карта
            edgecolors='black' # цвет границы маркера
           );
plt.plot(range(-2,5), range(4,-3,-1)); # линия разделения объектов
plt.show()
```

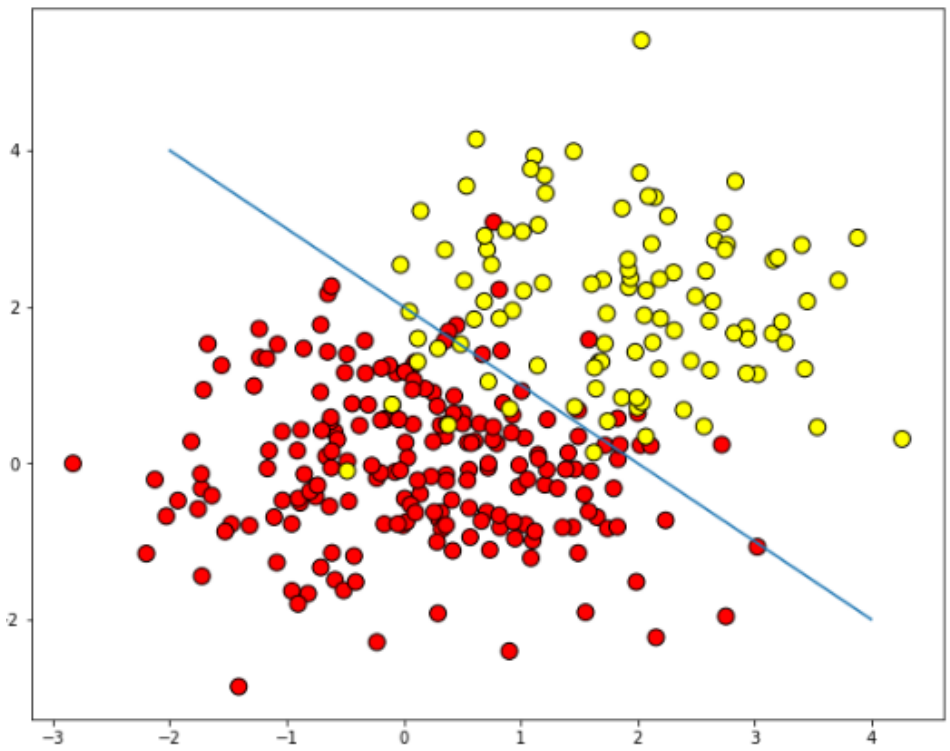


Рис.2 График разброса объектов с границей

Попробуем разделить эти два класса, используя обучение дерева решений. Для построения и обучения модели дерева принятия решений используют библиотеку класс `DecisionTreeClassifier` библиотеки scikit-learn [2]. В объекте дерева используется параметр `max_depth`, определяющий глубину дерева. В качестве целевой функции будем использовать критерий Джини.

```
# Импортируем класс модели Дерево решений
from sklearn.tree import DecisionTreeClassifier
# Создаем экземпляр класса
tree = DecisionTreeClassifier (criterion='gini', max_depth=3, random_state=1)
# Обучение дерева
tree.fit(train_data, train_labels)
```

Визуализируем полученную границу разделения классов. Для этого нам понадобится функция для построения границ `plot_decision_regions` из библиотеки `mlxtend`.

Выполним построение границ обученного дерева решений. В функцию передаем обучающие данные, метки классов (их надо преобразовать к целому типу) и обученное дерево. Границы решений приведены на рис.3.

```
plot_decision_regions(train_data, train_labels.astype(np.int_),
    clf=tree)
plt.show()
```

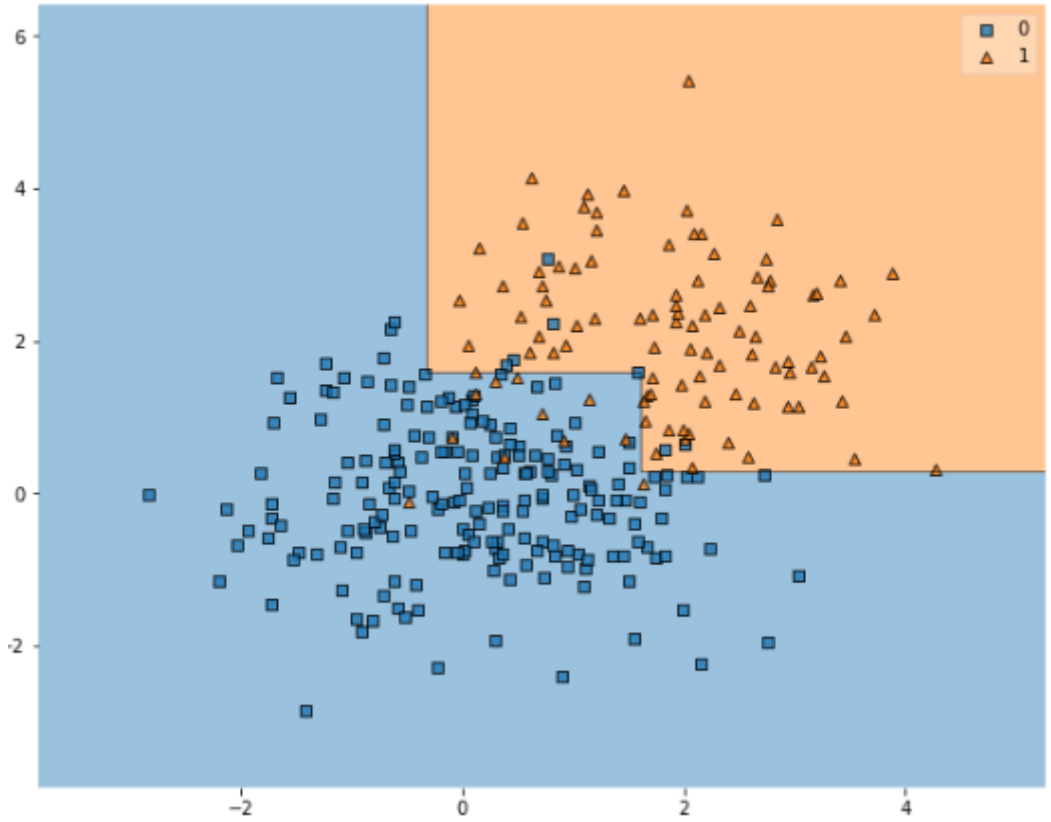


Рис.3 Границы дерева решений

Задание

1. Проведите эксперимент с моделью принятия решений

а) изменив глубину дерева решений (например, 5)

б) выбрав критерий энтропии (параметр модели `criterion='entropy'`)

2. Проведите сравнительный анализ полученных результатов

3. Сгенерируйте новый набор данных и решите задачу классификации по методике практического занятия

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.

2. Класс дерева принятия решений в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Порядок выполнения работы

1. Изучить теоретический материал

2. Ввести и выполнить программный код из описания практического занятия

3. Выполнить эксперимент с моделью в соответствии с заданием

4. Выполнить сравнительный анализ полученных результатов

5. Сгенерируйте новый набор данных и решите задачу классификации по методике практического занятия

Содержание отчета

1. Программный код (блокнот) решения задачи классификации методом дерева принятия решений

2. Результаты эксперимента с моделью и выводы сравнительного анализа

3. Программный код (блокнот) решения задачи на новом наборе данных

Практическое занятие № 46
Ансамблевые методы классификации

Цель занятия. Получить представление об ансамблевых методах и алгоритмах решения задач классификации. Научиться применять ансамблевые методы в задачах классификации. Освоить средства Python для реализации ансамблевых алгоритмов решения задачи классификации

Краткие теоретические сведения

Продолжим рассмотрение методов классификации. На данном занятии рассмотрим использования наборов методов классификации для повышения эффективности и точности предсказания – ансамблевых методов.

Цель ансамблевых методов - объединить различные классификаторы в их сочетание (мета-классификатор) для получения более точной и надежной модели. Каждый отдельный классификатор (его часто называют «слабым учеником») может решать задачу неидеально. Результаты могут быть либо смещенными, либо иметь большой разброс. Задачей ансамблевых методов является уменьшение смещения и разброса моделей слабых учеников.

Исходные классификаторы для объединения могут быть однородными (одна модель классификации с разными параметрами) и разнородными (разные модели классификации). Существуют 3 основных типа алгоритмов ансамблевых методов:

- **стекинг** (стек оценщиков с финальным классификатором) - подразумевает объединение разнородных моделей;
- **бэггинг** (bagging - упаковка) - объединение однородных модель, которые обучаются параллельно, а затем объединяются по определенному принципу (например, усреднение)
- **бустинг (**boosting — улучшение) - однородные модели обучают последовательно адаптивным способом (каждый последующий обучается с учетом опыта предыдущего)

Рассмотрим отдельные ансамблевые модели и реализуем их на Python с использованием библиотеки `scikit-learn`, так как в ней есть готовые классы ансамблевых методов. Создадим новый блокнот и вначале импортируем необходимые библиотеки:

```
import numpy as np
```

Загрузим набор данных цветков ириса:

```
from sklearn import datasets
# Загрузка данных
iris=datasets.load_iris()
```

Оставим на наборе данных только 2 класса объектов, исключив первый класс, и выберем для обучения первые 2 признака оставшихся объектов:

```
# Из набора исключаем образцы 1-го класса
# Массив признаков
x = iris.data[50:,[1,2]]
# Массив меток класса
y = iris.target[50:]
```

Выполним предварительную подготовку данных. Вначале перекодируем массив с метками класса, приведя значения классов к 0 и 1. Для этого используем класс `LabelEncoder` из модуля `preprocessing` библиотеки `scikit-learn`:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
# Преобразуем метки классов к значениям 0,1
yt=le.fit_transform(y)
print(yt)
```

Выполним стандартизацию признаков, воспользовавшись для этого классом `StandardScaler` из модуля `preprocessing` библиотеки `scikit-learn`. При стандартизации из значения каждого признака вычитают среднее и полученную разность делят на стандартное отклонение.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(x_train)
x_train_std = sc.transform(x_train)
x_test_std = sc.transform(x_test)
```

Далее разделим исходный набор данных на обучающий и проверочный наборы. При помощи функции `train_test_split` из модуля model_selection библиотеки scikit_learn разделим массивы `x` и `y` на обучающие данные и контрольные данные в размере 50% от общего объема.

```
from sklearn.model_selection import train_test_split
# Разбиваем данные на 2 набора (по 50%):
#   обучающий (train) и контрольный (test)
x_train, x_test, y_train, y_test = train_test_split(
   x,yt,test_size=0.5,random_state=1, stratify=y)
```

Для сравнения вначале попробуем использовать одиночные модели. Обучим модели на обучающем наборе и оценим точность прогноза на проверочном наборе.

Сначала используем модель ближайших соседей (класс KNeighborsClassifier [2]):

```
# Метод ближайших соседей
from sklearn.neighbors import KNeighborsClassifier
# Создание модели
knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
# Обучение модели
knn.fit(x_train_std, y_train)
# Оценка точности модели
score = knn.score (x_test_std,y_test)
print('Точность:',score)
```

```
Точность: 0.9
```

Затем используем метод дерева принятия решений (класс DecisionTreeClassifier [3]):

```
# Метод дерева принятия решений
from sklearn.tree import DecisionTreeClassifier
# Создание модели
dtr = DecisionTreeClassifier (criterion='gini', max_depth=3, random_state=1)
# Обучение модели
dtr.fit(x_train_std,y_train)
# Оценка точности модели
score = dtr.score (x_test_std,y_test)
print('Точность:',score)
```

```
Точность: 0.88
```

Наконец используем модель опорных векторов (класс SVC [4]):

```
# Метод опорных векторов
from sklearn.svm import SVC
# Создание модели
svc=SVC(random_state=0)
# Обучение модели
svc.fit(x_train_std,y_train)
# Оценка точности модели
score = svc.score (x_test_std,y_test)
print('Точность:',score)
```

```
Точность: 0.92
```

Из выбранных моделей классификации лучшую точность показал метод опорных векторов. Попытаемся улучшить качество модели классификации с использованием ансамблевых моделей разных типов.

Метод стекинга

Вначале выполним объединение разнородных моделей методом стекинга. Создадим ансамбль из моделей ближайших соседей, дерева принятия решений и опорных векторов:

```
# Создание набора одиночных моделей
estimators = [('sv', SVC()),
              ('dt', DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=1)),
              ('kn', KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski'))
             ]
```

Создадим ансамблевую модель, включающую подготовленный набор разнородных первичных моделей и финальную мета-модель, в качестве

которой выберем модель логистической регрессии. Для создания стекинговой ансамблевой модели используем класс `StackingClassifier` из пакета ансамблей `sklearn.ensemble` [5]:

```
# Создание ансамбля
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
ansClf = StackingClassifier(estimators=estimators,
                            final_estimator=LogisticRegression(random_state=0))
```

Выполним обучение ансамблевой модели. На вход всех первичных моделей подаётся обучающий набор, результат каждый из этих моделей поступает на вход мета-модели, которая вырабатывает окончательный прогноз:

```
# Обучение модели
ansClf.fit(x_train_std, y_train)
# Точность прогноза
score=ansClf.score(x_test_std,y_test)
print(np.round(score,5))
```

```
0.92
```

Точность ансамблевой модели оказался на уровне лучшей исходной одиночной модели.

Метод беггинга

Этот метод используются для уменьшения дисперсию базовой оценки. При использовании метода беггинга последовательно делается выборка из обучающего набора так, чтобы каждая новая выборка выполняла роль независимого обучающего набора. На каждой из этих выборок обучается выбранная модель «слабого ученика». Результаты затем агрегируются: либо выбирается результат лучшей модели либо результаты всех моделей усредняются.

Создадим ансамблевую модель, включающую базовую модель «слабого ученика». В качестве базовой модели выберем модель логистической регрессии. Для создания беггинговой ансамблевой модели используем класс `BaggingClassifier` из пакета ансамблей `sklearn.ensemble` [6]:

```
# Бэггинг
from sklearn.ensemble import BaggingClassifier
bagClf = BaggingClassifier(base_estimator=LogisticRegression(), # Базовая модель
                           n_estimators=50,          # Кол-во выборок (оценок)
                           random_state=12)
```

Выполним обучение ансамблевой модели. На вход базовой модели последовательно подаётся «независимая» выборка из обучающего набора, результаты обучения усредняются. После обучения оценим точность прогноза:

```
# Обучение модели
bagClf.fit(x_train_std, y_train)
# Точность прогноза
score=bagClf.score(x_test_std,y_test)
print(np.round(score,5))
```

```
0.94
```

Точность данной ансамблевой модели оказалась выше предыдущей.

Метод бустинга

Этот метод похож на предыдущий вариант ансамблевого метода. Отличие данного варианта в том, что базовая модель обучается на данных последовательно, то есть каждое новое обучение учитывает результат предыдущего. Недостатком бустинга является невозможность распараллелить вычисления. Существует два наиболее распространённых алгоритма бустинга - адаптивный и градиентный.

Адаптивный метод начинает с подгонки классификатора к исходному набору данных, а затем подгоняет дополнительные копии классификатора к тому же набору данных, но при этом веса неправильно классифицированных экземпляров корректируются таким образом, что последующие классификаторы больше сосредотачиваются на сложных случаях.

Для ансамблевой адаптивной бустинговой модели будем использовать базовую модель логистической регрессии. Для ансамблевой модели используем класс `AdaBustingClassifier` из пакета ансамблей `sklearn.ensemble`. Кроме базовой модели также задается количество итераций обучения. Это максимальное количество итераций и в случае идеальной подгонки процедура обучения прекращается досрочно.

```
# Бустинг
from sklearn.ensemble import AdaBoostClassifier
bustClf = AdaBoostClassifier(base_estimator=LogisticRegression(random_state=12),
                             n_estimators=100, # Максимальное кол-во обучений
                             random_state=12)
```

Выполним обучение ансамблевой модели и оценим точность прогноза:

```
# Обучение модели
bustClf.fit(x_train_std, y_train)
# Точность прогноза
score=bustClf.score(x_test_std,y_test)
print(np.round(score,5))
```

```
0.94
```

Точность данного варианта ансамблевой модели оказалась на уровне предыдущего.

Задание

1. Проведите эксперименты с ансамблевыми моделями

а) в модели стекинга поменяйте финальную модель классификации

б) в модели беггинга выберите другую базовую модель и поменяйте количество обучений

в) в модели бустинга выберите другую базовую модель и поменяйте максимальное количество обучений

2. Проведите сравнительный анализ полученных результатов

3. Скачайте набор данных для классификации из репозитория Kaggle и решите задачу классификации по методике практического занятия

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.
2. Класс ближайших соседей в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
3. Класс дерева принятия решений в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
4. Класс опорных векторов в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
5. Класс стекинговой ансамблевой модели в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html
6. Класс беггинговой ансамблевой модели в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
7. Класс бустинговой ансамблевой модели в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
8. Наборы данных классификации.- https://www.kaggle.com/datasets?tags=13302-Classification

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить эксперимент с моделью в соответствии с заданием
4. Выполнить сравнительный анализ полученных результатов
5. Создать и обучить модель классификации на наборе данных из репозитория Kaggle по методике практического занятия

Содержание отчета

1. Программный код (блокнот) решения задачи классификации ансамблевыми методами

2. Результаты эксперимента с моделью и выводы сравнительного анализа

3. Программный код (блокнот) решения задачи на новом наборе данных

Практическое занятие № 47
Методы кластеризации

Цель занятия. Получить представление о методах обучения моделей без учителя и задаче кластеризации данных. Научиться применять методы k-средних и DBSCAN в задачах кластеризации. Освоить средства Python для реализации методов кластеризации

Краткие теоретические сведения

Ранее мы рассмотрели модели классификации, которые относятся к моделям машинного обучения с учителем. В задачах такого типа данные, подаваемые на вход модели, содержат не только признаковые переменные, но и известные для них ответы - метки. Другой вид моделей машинного обучения - обучение без учителя. Данные, подаваемые на вход таких моделей, обычно не размечены, то есть передаются только входные переменные-признаки без соответствующих меток-ответов. К такому классу моделей относятся модели кластеризации. При кластеризации также необходимо разбить образцы данных на группы, но на основе выявления скрытых закономерностях в признаковых переменных. Кластеризация имеет целью найти в данных группирование, при котором объекты в одном кластере (заранее не известной группе) будут похожими друг на друга, чем на объекты из других кластеров.

Наиболее популярным алгоритмом кластеризации данных является метод k-средних [1]. Это итеративный алгоритм кластеризации, основанный на минимизации суммарных квадратичных отклонений точек кластеров от центроидов (средних координат) этих кластеров.

Первоначально выбирается желаемое количество кластеров k. Далее случайным образом из входных данных выбираются k элементов выборки, в соответствие которым ставятся k кластеров, в каждый из которых теперь включено по одной точке (начальный центроид кластера). Далее ищется ближайший сосед текущего центроида, добавляется к кластеру и пересчитываем положение центроида с учетом координат новых точек. Алгоритм заканчивает работу, когда координаты каждого центроида перестают меняться. Центроид каждого кластера в результате представляет собой набор значений признаков, описывающих усредненные параметры выделенных классов. Метод k-средних чувствителен к форме кластеров. Лучшие результаты метод дает для кластеров выпуклой формы, и плохо работает на вытянутых данных.

Другой популярный метод кластеризации DBSCAN, в котором кластеры формируются на основе плотности точек. Хорошо подходит для данных, содержащих кластеры одинаковой плотности. Вначале находятся образцы (точки) с высокой плотностью и на основе них строятся кластеры.

Метод не требует предварительных предположений о числе кластеров, но для него нужно настроить два других параметра:

- **eps** - максимальное расстояние между соседними точками
- **min_samples** - минимальное число точек в окрестности (количество соседей) одного кластера.

Более высокие значения min_samples или малые значения eps указывают на более высокую плотность, необходимую для формирования кластера. Кластеры в отличие от метода k-средних могут иметь любую форму

В scikit-learn есть соответствующие значения параметров по умолчанию, но часто их приходится настраивать самостоятельно. Параметр eps имеет важное значение, так как контролирует локальное окружение точек. Если выбрано слишком маленькое расстояние, большая часть данных вообще не будет кластеризована (и помечена как «шум»). Если же будет выбрано слишком большое значение расстояния, близкие кластеры будут объединены в один кластер, и в конечном итоге весь набор данных будет возвращен как единый кластер.

Для решения задачи классификации будем использовать классический набор данных ирисов. Этот набор включает 150 записей с пятью атрибутами: длина чашелистика (sepal length), ширина чашелистика (sepal width), длина лепестка (petal length), ширина лепестка (petal width) и класс, соответствующий одному из трех видов: Iris Setosa, Iris Versicolor или Iris Virginica, обозначенных соответственно 0, 1, 2. Будем использовать данный набор в режиме неконтролируемого обучения. Значения классов будем использовать не на этапе обучения моделей, а на этапе ее проверки.

Рассмотрим реализацию упомянутых методов кластеризации на Python с использованием библиотеки `scikit-learn`. Создадим новый блокнот и вначале импортируем необходимые библиотеки:

```
import numpy as np
```

Загрузим набор данных цветков ириса:

```
from sklearn import datasets
# Загрузка данных
iris=datasets.load_iris()
```

Набор данных будем использовать в двух вариантах:

1) набор данных data2 и метки target2, включающие только 2 наиболее различимых класса цветков (первые 100 экземпляра набора данных и первые 100 меток):

```
data2=iris.data[:100,:]
target2=iris.target[:100]
```

```
data2.shape
```

```
(100, 4)
```

2) полный набор данных data3 и меток target3

```
data3=iris.data
target3=iris.target
data3.shape
```

```
(150, 4)
```

Построим точечные графики длины и ширины лепестка цветка для набора данных с 2 классами (рис.1)

```
# Разделение набора данных
x_axis = data2[:, 2]  # petal Length
y_axis = data2[:, 3]  # petal Width
# Построение графика точек разброса
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
plt.scatter(x_axis, y_axis, c=target2)
plt.show()
```

и полного набора (рис.2)

```
x_axis = data3[:, 2]  # petal Length
y_axis = data3[:, 3]  # petal Width
# Построение графика точек разброса
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
plt.scatter(x_axis, y_axis, c=target3)
plt.show()
```

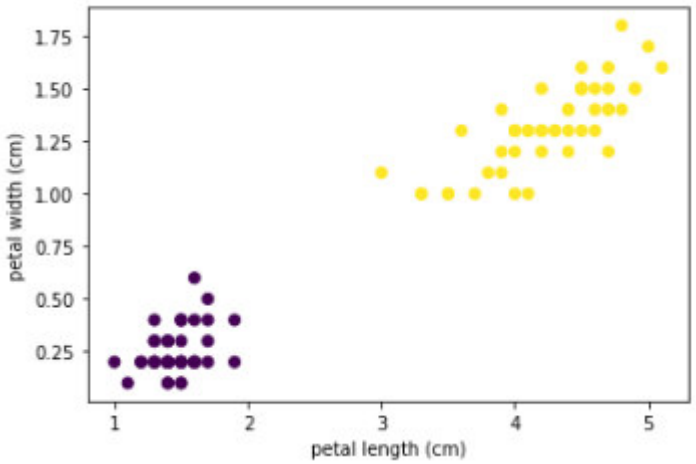


Рис.1 Размеры лепестков цветков классов 0 и 1

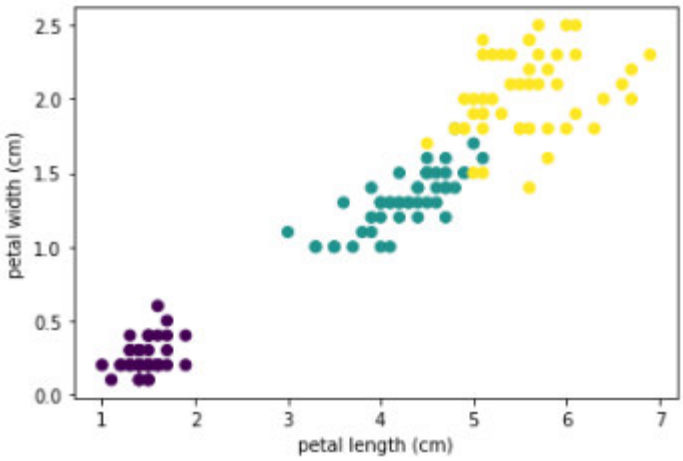


Рис.2 Размеры лепестков цветков трех классов

Кластеризация методом k-средних

Начнем с метода k-средних. Для создания модели k-средних используем класс `KMeans` из пакета моделей кластеризации `sklearn.cluster` [2]:

```
from sklearn.cluster import KMeans
```

Начнем с неполного набора данных data2. При создании модели необходимо указать параметр k.

Зададим вначале k = 3 (в неполном наборе только 2 класса):

```
# Укажем неправильное число кластеров
model = KMeans(n_clusters=3)
```

Обучим модель и выполним по ней предсказание принадлежности данных к кластерам:

```
# Проводим кластеризацию
model.fit(data2)
# Предсказание на всем наборе данных
yp = model.predict(data2)
```

Выведем график, на котором у каждой точке цвет будет определяться номером кластера, к которому она отнесена (рис.3).

```
x_axis = data2[:, 2]  # petal Length
y_axis = data2[:, 3]  # petal Width
# Построение графика точек разброса
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
plt.scatter(x_axis, y_axis, c=yp)
plt.show()
```

Как видно из графика. Вытянутый второй класс при неправильном выборе параметра k разделился на 2 кластера.

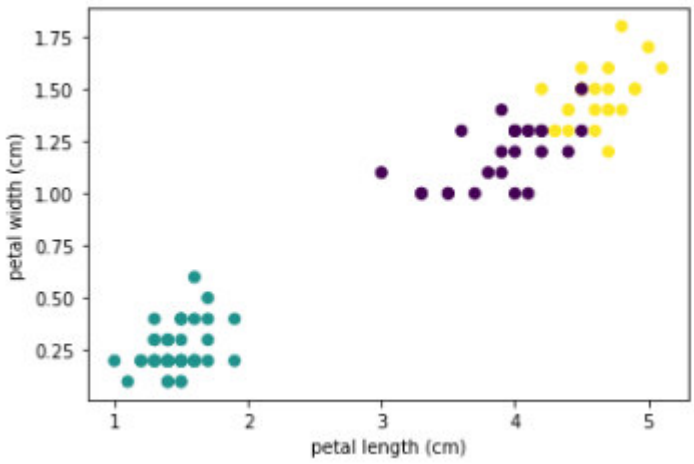


Рис.3 Результаты кластеризации k-средних данных двух классов при выборе k=3

Теперь зададим k = 2, что соответствует двум классам в сокращенном наборе данных:

```
model = KMeans(n_clusters=2)
```

Обучим модель и выполним по ней предсказание принадлежности данных к кластерам. Выведем график, на котором у каждой точке цвет будет определяться номером кластера, к которому она отнесена (рис.4).

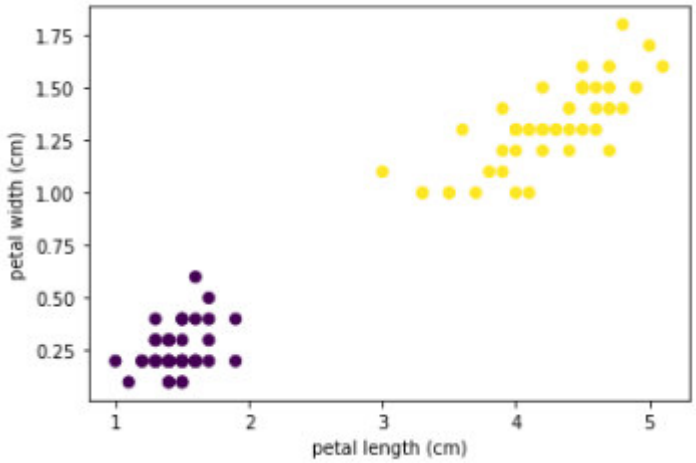


Рис.4 Результаты кластеризации k-средних данных двух классов при выборе k=2

Как видно из графика кластеризация выполнена правильно.

Создадим теперь модель на полном наборе данных data3. Зададим k=3 (что соответствует количеству классов в данных).

```
# Укажем правильное число кластеров
model = KMeans(n_clusters=3)
```

Обучим модель и выполним по ней предсказание принадлежности данных к кластерам. Выведем график, на котором у каждой точке цвет будет определяться номером кластера, к которому она отнесена (рис.5).

```
# Проводим кластеризацию
model.fit(data3)
# Предсказание на всем наборе данных
yp = model.predict(data3)
```

```
x_axis = data3[:, 2]  # petal Length
y_axis = data3[:, 3]  # petal Width
# Построение графика точек разброса
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
plt.scatter(x_axis, y_axis, c=yp)
plt.show()
```

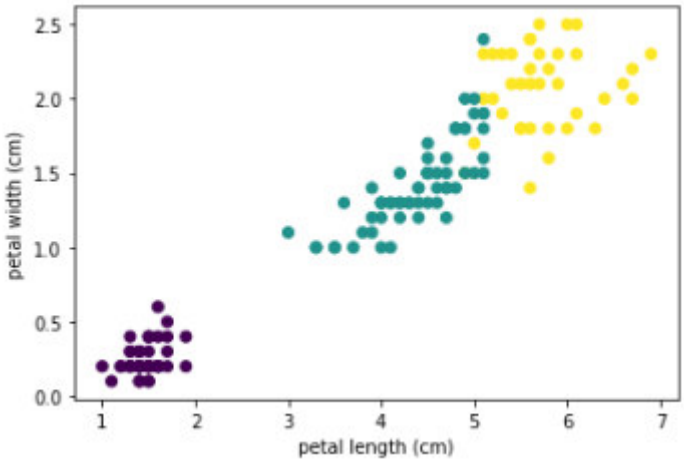


Рис.5 Результаты кластеризации k-средних полного набора данных при выборе k=3

Метод DBSCAN

Выполним кластеризацию методом плотности DBSCAN. Для создания модели DBSCAN используем одноименный класс из пакета методов кластеризации `sklearn.cluster` [3]:

```
from sklearn.cluster import DBSCAN
```

Выполним кластеризацию на неполном наборе данных data2.

Создадим модель DBSCAN с параметрами eps=0.5, `min_samples=4` и обучим ее:

```
# Определяем модель
dbscan = DBSCAN(eps=0.5, min_samples=4)
# Обучаем
dbscan.fit(data2)
```

Принадлежность каждой точки к кластеру можно посмотреть в поле labels_ Выведем точки на графике (рис.6) , где цвет будет определяться номером кластера.

```
x_axis = data2[:, 2]  # petal Length
y_axis = data2[:, 3]  # petal Width
# Построение графика точек разброса
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
plt.scatter(x_axis, y_axis, c=dbscan.labels_)
plt.show()
```

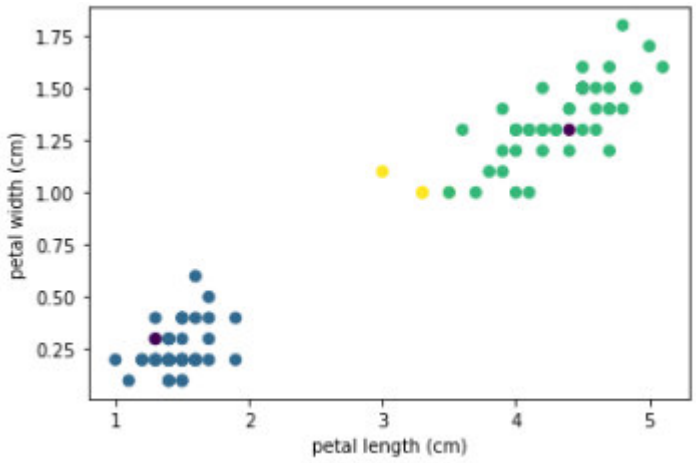


Рис.6 Результаты кластеризации DBSCAN набора данных двух классов

Как видно из графика было ошибочно выделено несколько шумовых точек.

Выполним кластеризацию на полном наборе данных data3.

Создадим модель DBSCAN с такими же параметрами, как для неполного набора данных eps=0.5, `min_samples=4` и обучим ее:

```
# Полный набор
# Определяем модель
dbscan = DBSCAN(eps=0.3, min_samples=4)
# Обучаем
dbscan.fit(data3)
```

Выведем точки на графике (рис.7) , где цвет будет определяться номером кластера.

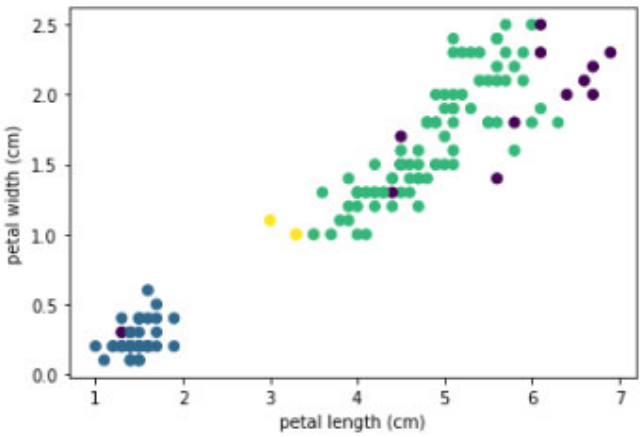


Рис.6 Результаты кластеризации DBSCAN на полном наборе данных

Из-за высокой плотности точек 2 и 3 класса оба этих класса были отнесены к одному кластеру. И еще часть точек были идентифицированы как шум (значения кластера = -1).

Таким образом, метод DBSCAN дает ошибки для данных нечетко выраженных классов. Можно поэкспериментировать с параметрами метода для улучшения кластеризации.

Задание

1. Проведите эксперименты с моделью DBSCAN

а) при фиксированном значении параметра eps меняйте параметр `min_samples`, что добиться лучшей кластеризации

б) при фиксированном значении параметра `min_samples` меняйте параметр eps, что добиться лучшей кластеризации

2. Проведите сравнительный анализ полученных результатов

3. Скачайте набор данных для классификации из репозитория Kaggle и решите задачу кластеризации по методике практического занятия

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.
2. Класс модели k-средних в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
3. Класс модели DBSCAN в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
4. Наборы данных классификации.- https://www.kaggle.com/datasets?tags=13302-Classification

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить эксперимент с моделью в соответствии с заданием
4. Выполнить сравнительный анализ полученных результатов
5. Создать и обучить модель кластеризации на наборе данных из репозитория Kaggle по методике практического занятия

Содержание отчета

1. Программный код (блокнот) решения задачи кластеризации
2. Результаты эксперимента с моделью и выводы сравнительного анализа
3. Программный код (блокнот) решения задачи на новом наборе данных

Практическое занятие № 48
Задача детектирования аномалий

Цель занятия. Получить представление о задачах поиска аномалий (выпадающих значений) в исходных данных. Освоить средства Python для решения задачи поиска и исключения аномалий в данных

Краткие теоретические сведения

Аномалиями (выбросами) в выборке называют точки данных, которые не принадлежат определенной популяции. Эти точки данных явно выделяются среди других данных.

Например, в ряде 40,24,22,39,19,28,1300, 10,38 число 1300 явно выделяется.

Аномалии легко определить (часто также говорят детектировать или идентифицировать), когда наблюдения представляют простой набор чисел небольшого объема. Однако часто имеется тысячи наблюдений и еще эти наблюдения многомерные. Это относится к наборам данных для машинного обучения.

Необходимо иметь специальные методы и алгоритмы обнаружения аномальных значений. В библиотеке `scikit-learn` имеется набор инструментов машинного обучения, которые можно использовать для обнаружения выбросов. Эта стратегия реализуется с помощью неконтролируемого обучения (обучения без учителя). Методы неконтролируемого обучения используются в задачах кластеризации и выбросы при работе этих алгоритмов определяются как шум.

Методы производят подгонку модели под набор данных обучения:

```
модель.fit(обучающий набор)
```

Наблюдения могут быть идентифицированы как выбросы с помощью метода `predict`:

```
модель.predict(обучающий набор)
```

Нормальные данные помечены как 1, а выбросы - как «-1». Можно 2 этих метода совместить:

```
модель.fit_predict(обучающий набор)
```

Существует много методов неконтролируемого обучения. В данном занятии рассмотри 2 таких метода: метод на основе плотности данных DBSCAN и метод изолированного леса [1].

В качестве наборов данных будем использовать случайным образом сгенерированные данные:

1) простой набор из 6 точек с явными выбросами (2 последние точки):

(-1, -1), (-2, -1), (-3, -2), (0, 0), (-10, 8), (4, 4)

2) выборка нормально распределенных случайных чисел, к которой будут добавлены равномерно распределенные случайные числа из большего диапазона.

Создадим новый блокнот и импортируем необходимые библиотеки:

```
import numpy as np
import matplotlib.pyplot as plt
```

Создадим первый набор данных `x1`:

```
# Набор произвольных точек с явными выбросами
x1 = np.array([[-1, -1], [-2, -1], [-3, -2], [0, 0], [-10, 8], [4, 4]])
```

и случайную выборку `x`:

```
# Генерация выборки
gen = np.random.RandomState(42)
# Генерация матрицы 150x2 нормальных СЧ
#  Приводим к ст.откл=0.5 и среднему=3
x_n = 0.5 * gen.randn(150, 2)+3
# Генерация шума: РРСЧ на (-4,4)
x_vyb = gen.uniform(low=-4, high=4, size=(15, 2))
# Добавляем шум к выборке
x=np.r_[x_n,x_vyb]
```

Сначала будем детектировать выбросы для первого набора данных. Первым будем использовать метод DBSCAN.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise, плотностной алгоритм пространственной кластеризации с присутствием шума) – популярный алгоритм кластеризации. Метод выполняет пространственную кластеризацию зашумленных данных на основе анализа плотности. Находятся образцы (точки) с высокой плотностью и на основе них строятся кластеры. Хорошо подходит для данных, содержащих кластеры одинаковой плотности. Метод не требует предварительных предположений о числе кластеров, но нужно настроить два других параметра:

`eps` - максимальное расстояние между соседними точками;

`min_samples` - минимальное число точек в окрестности (количество соседей), когда можно говорить, что эти экземпляры данных образуют один кластер.

В scikit-learn есть соответствующие значения параметров по умолчанию, но часто их приходится настраивать самостоятельно.

Для создания модели DBSCAN используем одноименный класс из пакета методов кластеризации `sklearn.cluster` [2]:

```
from sklearn.cluster import DBSCAN
```

Создадим модель DBSCAN с параметрами eps=3, `min_samples=2`:

```
db = DBSCAN(min_samples = 2, eps = 3)
```

Обучим модель на наборе данных и помощью свойства `labels_` определим метки кластеризации (напомним, что шум идентифицируется меткой «-1»):

```
db.fit(x1)
yp=db.labels_
yp
```

```
array([ 0,  0,  0,  0, -1, -1], dtype=int64)
```

Как видно, выбросы определились правильно.

Перейдем к методу изолированного леса IsolationForest. Метод IsolationForest «изолирует» наблюдения, случайным образом выбирает объект, а затем случайным образом выбирает значение разделения между максимальным и минимальным значениями выбранного объекта. Поскольку рекурсивное разделение может быть представлено древовидной структурой, количество разделений, необходимых для выделения образца, эквивалентно длине пути от корневого узла до конечного узла. Эта длина пути, усредненная по лесу таких случайных деревьев, является мерой нормальности. Случайное разбиение приводит к заметным более коротким путям для аномалий. Следовательно, когда лес случайных деревьев в совокупности дает более короткие пути для определенных выборок, они, скорее всего, являются аномалиями.

Для создания модели `IsolationForest` используем одноименный класс из пакета ансамблевых моделей `sklearn.ensemble` [3]:

```
from sklearn.ensemble import IsolationForest
```

Создадим модель `IsolationForest`:

```
clf = IsolationForest(n_estimators=10, warm_start=True)
```

В модели использованы параметры `n_estimator` - количество оценщиков в ансамбле, `warm_start` - следующий оценщик использует результаты предыдущего.

Оценим количество выбросов методом модели `fit_predict` и оценим количество выбросов:

```
yp=clf.fit_predict(x1)
print(yp)
```

```
[ 1  1  1  1 -1 -1]
```

Как видно, и этот метод на простых данных с явными выбросами определил их правильно.

Перейдем теперь к более сложному набору данных - зашумленной выборке. Также начнем с метода DBSCAN. Создадим модель с измененными параметрами, обучим модель на наборе данных и помощью свойства `labels_` определим метки, идентифицируемые как шум. Выведем общее

количество таких меток методом `count`. Таких меток оказалось 39 (в выборку закладывалось только 15 зашумленных значений):

```
db = DBSCAN(eps=0.3, min_samples=10)
db.fit(x)
yp=db.labels_
print(list(yp).count(-1))
```

```
39
```

Также визуализируем данные, выведя точки выборки на график.

```
# Построение графика точек разброса
plt.xlabel('x')
plt.ylabel('y')
plt.scatter(x[:,0], x[:,1], c=yp)
plt.show()
```

График кластеризации представлен на рис.1. Цвет точек (отображаемых кружками) определяется по меткам кластеризации. Темным цветом отмечены точки, определенные как шум.

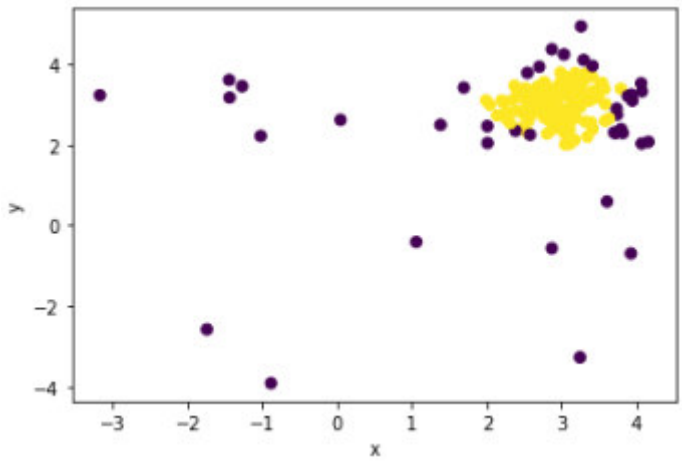


Рис.1 Выделение шума методом DBSCAN

Перейдем к методу изолированного леса IsolationForest. Создадим модель `IsolationForest`:

```
clf = IsolationForest(n_estimators=10, warm_start=True)
```

Оценим количество выбросов методом модели `fit_predict` и оценим количество выбросов:

```
yp=clf.fit_predict(x)
print(list(yp).count(-1))
```

```
15
```

Количество определенных точек шума совпало с количеством добавленных. На рис. 2 приведен график точек разброса.

```
# Построение графика точек разброса
plt.xlabel('x')
plt.ylabel('y')
plt.scatter(x[:,0], x[:,1], c=yp)
plt.show()
```

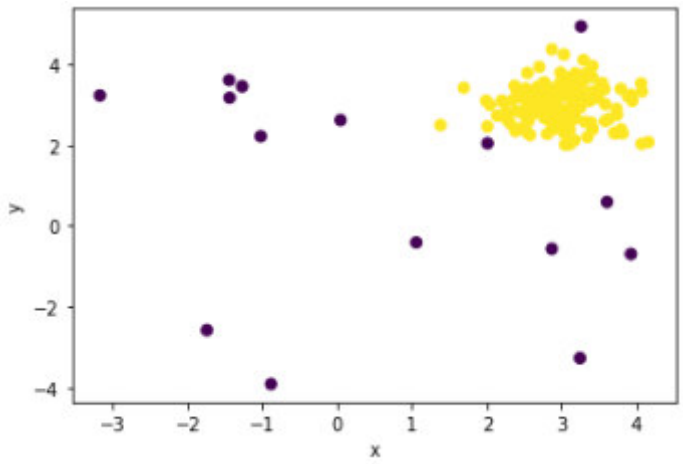


Рис.2 Выделение шума методом IsolationForest

Задание

1. Сгенерируйте простой набор данных из 20 точек с 4 выбросами

2. Сгенерируйте выборку и 300 нормально распределенных точек со смещением m и разбросом s, массив шума из равномерно распределенных чисел на интервале (a, b)

3. Выполните детектирование выбросов методами DBSCAN и IsolatedForest

4. Сделайте выводы из полученных результатов.

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.

2. Класс модели DBSCAN в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

3. Класс модели изолированного леса в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Порядок выполнения работы

1. Изучить теоретический материал

2. Ввести и выполнить программный код из описания практического занятия

3. Выполнить пункты задания

4. Выполнить сравнительный анализ полученных результатов

Содержание отчета

1. Программный код (блокнот) решения задачи поиска выбросов
2. Программный код (блокнот) решения задачи на новых наборах данных

Практическое занятие № 49
Методы снижения размерности многомерных данных

Цель занятия. Получить представление о задаче снижения размерности исходного признакового пространства, методах снижения размерности. Освоить средства Python для решения задачи снижения размерности

Краткие теоретические сведения

Под уменьшением размерности в машинном обучении подразумевается уменьшение числа признаков набора данных. Удаление избыточных или слабо информативных признаков может повысить эффективность модели, уменьшить объем памяти и ускорить работу алгоритмов машинного обучения.

Одним из основных методов уменьшения размерности является анализ главных компонент (PCA principal component analysis).

Анализ главных компонент (PCA) – это статистический метод, который создает новые признаки данных путем анализа имеющихся признаков [1]. Это выполняется объединением признаков из пространства большого количества измерений в меньшее. Главные компоненты (новые измерения) представляют собой линейные комбинации исходных переменных, вычисляемые с помощью собственных векторов. Предполагается, что новые компоненты ортогональны или не связаны друг с другом

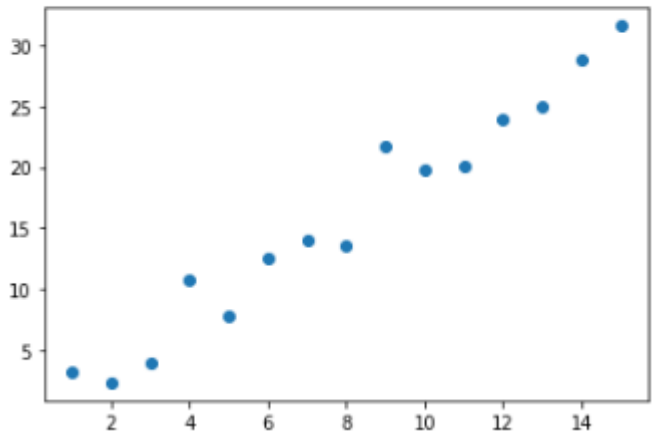
Рассмотрим идею метода на примере случайного набора данных. Сгенерируем 15 наблюдений с 2 признаками.

```
import numpy as np
x = np.arange(1,16)
y = np.round(2 * x + np.random.randn(15)*2,3)
X = np.vstack((x,y))
print(X)
```

```
[[ 1.     2.     3.     4.     5.     6.     7.     8.     9.    10.
  11.    12.    13.    14.    15.   ]
 [ 3.203  2.323  4.013 10.756  7.723 12.546 14.06  13.516 21.688 19.859
  20.131 23.946 24.986 28.8   31.622]]
```

Выведем график точек разброса

```
plt.scatter(x,y)
plt.show()
```



Из него видно, что мы имеем 2 сильно коррелированных признака. Методом главных компонент вычислим новый признак, который заменит 2 исходных путем незначительной потери информации.

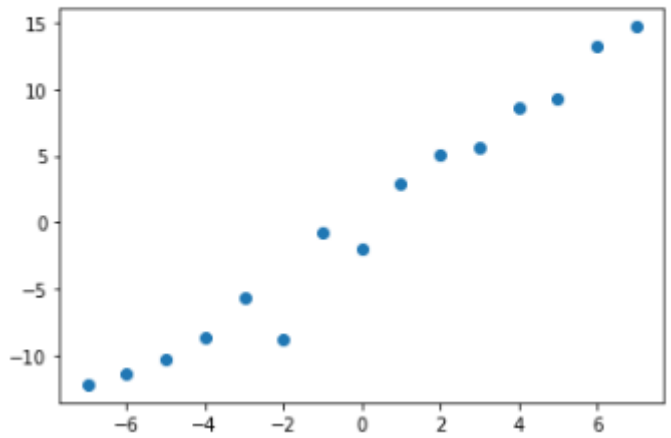
Для упрощения вычислений выполним центрирование исходных признаков, вычтя из значений признаков среднее.

```
m=(x.mean(),y.mean())
x=x-x.mean()
y=y-y.mean()
Xst=np.vstack((x,y))
print(Xst)
print("Вектор средних: ", m)
```

```
[[ -7.          -6.          -5.          -4.          -3.
   -2.          -1.           0.           1.           2.
    3.           4.           5.           6.           7.        ]
 [-12.14786667 -11.29886667 -10.20286667  -8.64986667  -5.68086667
   -8.75186667  -0.73686667  -2.04686667   2.93613333   5.11213333
    5.55813333   8.66513333   9.33213333  13.23213333  14.68013333]]
Вектор средних:  (8.0, 15.925866666666666)
```

Выведем график и увидим, что характер связи не изменился, но признаки стандартизированы.

```
plt.scatter(x,y)
plt.show()
```



Для описания формы случайного вектора необходима ковариационная матрица. Вычислим ковариационную матрицу, используя метод cov.

```
cvm = np.cov(Xst)
print (cvm)
```

```
[[20.         40.5025    ]
 [40.5025     84.72645241]]
```

В этой матрице вариация по x равна 20, по y - 84.73, а ковариация - 40.5.

Теперь надо найти такой вектор, при котором бы дисперсия проекции нашей выборки на него была максимальной. Для этого матрицу необходимо разложить на собственные вектора и собственные значения. Направление максимальной дисперсии у проекции всегда совпадает с собственным вектором, имеющим максимальное собственное значение, равное величине этой дисперсии.

В библиотеке numpy реализована функция `numpy.linalg.eig(X)`, Она возвращает 2 массива – массив собственных значений и массив собственных векторов-столбцов. И векторы нормированы, их длина равна 1. Эти 2 вектора задают новый базис для выборки такой, что его оси совпадают с полуосями аппроксимирующего эллипса нашей выборки.

Умножив массив собственных векторов на исходную матрицу признаков, получим новый признак

```
s,vecs = np.linalg.eig(cvm)
v = (vecs[:,1])
Xn = np.dot(v,Xst)
print(Xn)
```

```
[ 13.98153773  12.78298681  11.36184525   9.52886593   6.41982039
   8.75388216   1.09749897   1.84458856  -3.07942686  -5.47383927
  -6.30921597  -9.54262389 -10.57716065 -14.5252017  -16.26355746]
```

Посмотрим, какие потери получены из-за перехода от 2-х признаков к одному. Для этого возьмем последнюю точку и развернем ее в исходное пространство признаков:

```
Xv = np.dot(Xn[14],v) + m
print ('Восстановленное: ', Xv)
print ('Исходное: ', X[:,14])
```

```
Восстановленное:  [15.04966161 30.58260461]
Исходное:  [15.    30.606]
```

Легко подсчитать, что потеря информации составила менее 1 процента.

Все расчеты метода главных компонент мы выполнили с использованием матричных операций и статистических расчетов. Но в библиотеке sklearn есть класс PCA, который позволяет в несколько операторов выполнить расчеты [2]. Используем ее и посмотрим, насколько будут сильно различаться результаты:

```
from sklearn.decomposition import PCA
# настройка объекта PCA на число компонент
pca = PCA(n_components = 1)
# определение вектора главной комп.
Xpca = pca.fit_transform(X.T)
print(Xpca)
```

```
[[-13.98153773]
 [-12.78298681]
 [-11.36184525]
 [ -9.52886593]
 [ -6.41982039]
 [ -8.75388216]
 [ -1.09749897]
 [ -1.84458856]
 [  3.07942686]
 [  5.47383927]
 [  6.30921597]
 [  9.54262389]
 [ 10.57716065]
 [ 14.5252017 ]
 [ 16.26355746]]
```

Как видно результаты не отличаются.

Рассмотрим реализацию метода главных компонентов на реальном наборе данных, который используется в задачах машинного обучения. Это набор данных для классификации грибов на съедобные и несъедобные [3]. Набор данных содержит 22 признака. Первый столбец определяет класс (e-съедобный, p - ядовитый).

Вначале импортируем все необходимые библиотеки:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
```

Загрузим скачанный набор данных:

```
m_data = pd.read_csv('d:\data\mushrooms.csv')
m_data.head(2)
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring
0	p	x	s	n	t	p	f	c	n	k	...	s
1	e	x	s	y	t	a	f	c	b	k	...	s

Для применения метода главных компонент необходимо преобразовать все колонки набора данных в числовой формат. Для этого используется специальный класс кодировщика `LabelEncoder`:

```
# Создание экземпляра кодера
encoder = LabelEncoder()
# Выполним преобразование для всех колонок
for col in m_data.columns:
    m_data[col] = encoder.fit_transform(m_data[col])
```

Выделим массив признаков Xd и массив классов грибов y_label:

```
Xd = m_data.iloc[:,1:23]
y_label = m_data.iloc[:, 0]
```

```
Xd.head(2)
```

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...
0	5	2	4	1	6	1	0	1	4	0	...
1	5	2	9	1	0	1	0	0	4	0	...

2 rows × 22 columns

```
y_label.head(2)
```

```
0    1
1    0
Name: class, dtype: int32
```

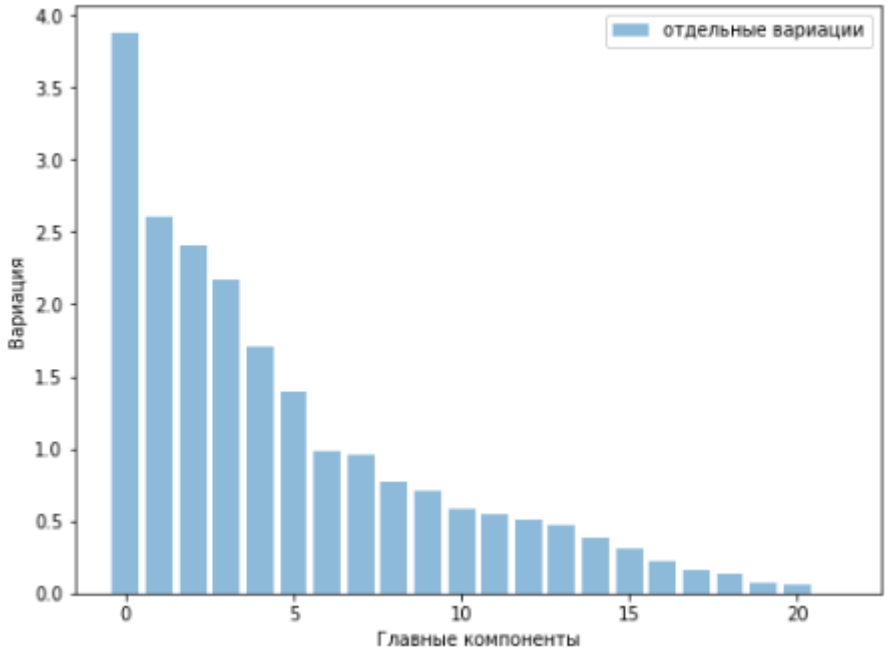
Выполним стандартизацию кодированных признаков:

```
# Стандартизация признаков
scaler = StandardScaler()
Xst = scaler.fit_transform(Xd)
print(Xst[0])
```

```
[ 1.02971224  0.14012794 -0.19824983  1.18591657  0.88193766  0.16289645
 -0.43886364  1.49468272 -0.22899776 -1.14480575  1.78146019  0.68377765
  0.58638466  0.62244139  0.63199138  0.          0.14203663 -0.25613174
  0.94808086 -0.67019486 -0.5143892   2.03002809]
```

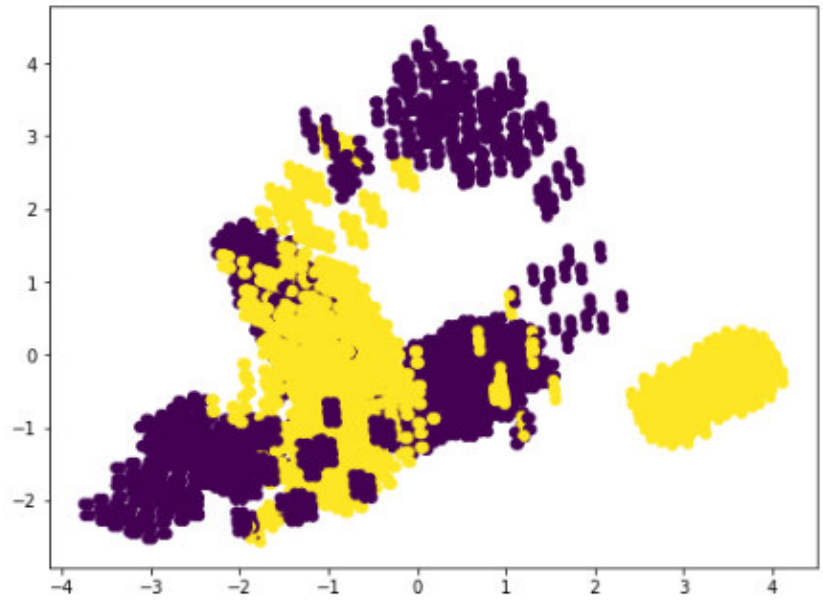
Теперь мы применим метод PCA для получения списка признаков и построим график признаков с наибольшей объяснительной силой:

```
pca = PCA()
pca.fit_transform(Xst)
pca_variance = pca.explained_variance_

plt.figure(figsize=(8, 6))
plt.bar(range(22), pca_variance, alpha=0.5, align='center', \
        label='отдельные вариации')
plt.legend()
plt.ylabel('Вариация')
plt.xlabel('Главные компоненты')
plt.show()
```



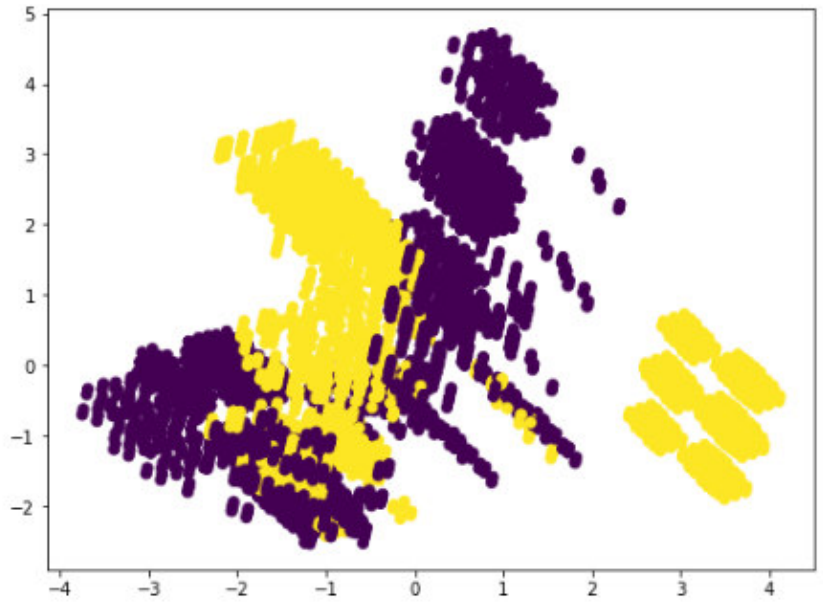
Выполним классификацию (метод `fit`) по всему пространству признаков и построим точечную диаграмму классификации объектов на основе этих признаков.

```
pca.fit(Xst)
xd = pca.transform(Xst)

plt.figure(figsize=(8,6))
plt.scatter(xd[:,0], xd[:,5], c=m_data['class'])
plt.show()
```



Теперь выполним классификацию на основе первых 6 значимых компонентах и посмотрим, как изменится классификация:

```
pca1 = PCA(n_components=6)
pca1.fit(Xst)
xd1 = pca1.transform(Xst)

plt.figure(figsize=(8,6))
plt.scatter(xd1[:,0], xd1[:,1], c=m_data['class'])
plt.show()
```



Как видно результат классификации стал более ярко выраженным

Задание

1. Сгенерируйте набор данных из 20 строк по 2 признакам. Используя метод главных компонент. Получите новый признак

2. Скачайте из репозитория Kaggle [4] набор данных для классификации и выполните преобразование исходного набора признаков к меньшему количеству значимых признаков

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.
2. Класс модели главных компонент в scikit-learn.- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
3. Набор данных для классификации грибов.- https://www.kaggle.com/uciml/mushroom-classification
4. Наборы данных классификации.- https://www.kaggle.com/datasets?tags=13302-Classification

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Сгенерировать новый набор данных и решить задачу снижения размерности по методике практического занятия
4. Скачать из репозитория Kaggle [4] набор данных для классификации и выполнить преобразование исходного набора признаков к меньшему количеству значимых признаков

Содержание отчета

1. Программный код (блокнот) решения задачи классификации методом дерева принятия решений
2. Программный код (блокнот) решения задачи на новом наборе данных
3. Программный код (блокнот) решения задачи на новом наборе данных из репозитория Kaggle

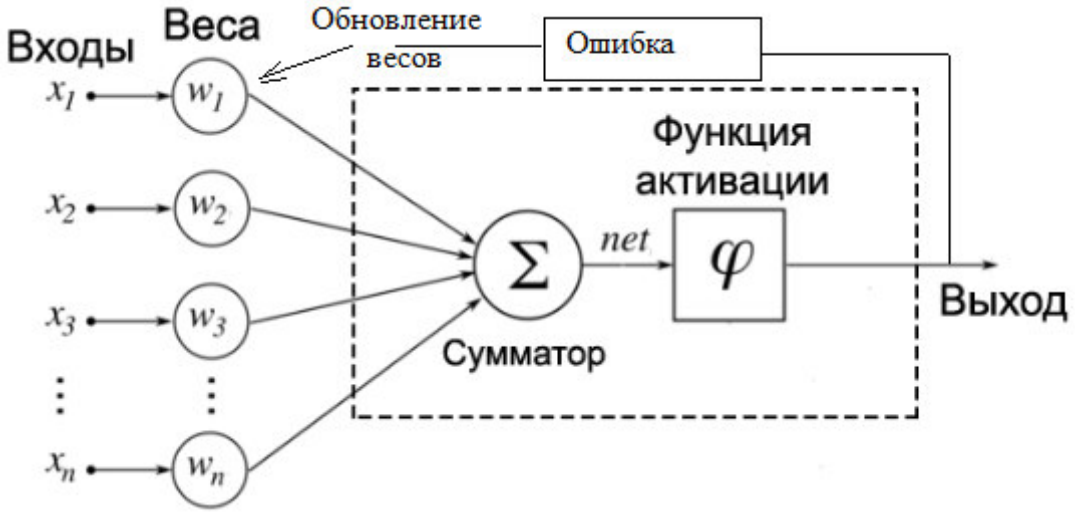
Практическое занятие № 50
Математические основы нейронных сетей. Модель персептрона

Цель занятия. Получить представление о математических основах нейросетевого подхода к машинному обучению. Выполнить программную реализацию Персептрона на Python

Краткие теоретические сведения

Стараясь понять, каким образом работает биологический мозг, с целью разработки искусственного интеллекта Уоррен Маккалок и Уолтер Питтс в 1943 г. впервые опубликовали концепцию упрощенной клетки головного мозга - нейрона Маккалока-Питтса. Эту нервную клетку описали в виде простого логического элемента с бинарными выходами. Это была модель преобразования входного сигнала в выходной. Искусственная нейронная сеть может быть представлена в виде совокупности искусственных нейронов, связанных между собой. Множественные входные сигналы поступают в клетку, и если накопленный сигнал превышает определенный порог, то генерируется выходной сигнал, который передается дальше. В 1957 году Фрэнк Розенблатт, опираясь на модель нейрона, предложил модель, которую он назвал персептроном, и сформулировал алгоритм автоматического обучения для принятия решения о том, активировать нейрон или нет.

Модель персептрона (рис.1) иллюстрирует то, каким образом персептрон получает входы x и смешивает их с весами w для вычисления чистого входа. Чистый вход затем передается функции активации (в данном случае единичная ступенчатая функция), которая генерирует бинарный выход -1 или +1. Во время фазы обучения этот выход используется для вычисления ошибки распознавания и для обновления весов.



В данной работе рассматривается реализация модели персептрона средствами Python [1]. Будем использовать объектно-ориентированный

подход, определив персептрон как класс языка Python. Это позволит инициализировать новые объекты персептрона, которые смогут обучаться на данных при помощи метода fit (выполнить подгонку модели) и делать прогнозы при помощи метода predict (распознать).

Персептрон будем использовать для классификации цветков ириса: определения вида цветка по их параметрам (длине чашелистика и длине лепестка).

Создадим новый блокнот и загрузим необходимые библиотеки:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
```

Создадим класс персептрона.

Параметры класса

eta - параметр скорости обучения (от 0.0 до 1.0)

n_iter - номер прохода (итерации) по обучающему набору

random_state - начальное значение генератора СЧ

Атрибуты (поля) класса:

w_ - одномерный массив весовых коэффициентов

errors_ - список ошибок классификации в каждой эпохе

```
class Perceptron(object):
    #  Метод инициализации полей класса
    def __init__(self, eta=0.01, n_iter=50, random_state=1):
        self.eta = eta
        self.n_iter = n_iter
        self.random_state = random_state
    #  Метод обучения
    def fit(self, X, y):
        # инициализация весов случайными значениями
        rgen = np.random.RandomState(self.random_state)
        self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
        self.errors_ = []
        #  Цикл по числу итераций (проходов)
        for _ in range(self.n_iter):
            errors = 0
            for xi, target in zip(X, y):
                update = self.eta * (target - self.predict(xi))
                self.w_[1:] += update * xi
                self.w_[0] += update
                errors += int(update != 0.0)
            self.errors_.append(errors)
        return self
    #   метод расчета чистого входа
    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    #  метод определения метки класса по ступенчатой функции активации
    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, -1)
```

Класс содержит методы:

а) обучения модели `fit`. Параметры метода:

n_samples - количество образцов

n_features - число признаков (входных переменных)

x - входной массив формы [n_samples, n_features]

y - массив целевых значений (ответов) формы = [n_samples]

б) вычисления прогноза `predict`.

Имея класс Perceptron, можно создавать объекты данного класса, задавая темп обучения eta и число эпох (проходов по обучающему набору) n_iter. При помощи метода выполнения подгонки (обучения) fit.

Тестировать реализацию персептрона будем на наборе данных цветков ириса. Загрузим данные из репозитория Калифорнийского университета:

```
df = pd.read_csv('https://archive.ics.uci.edu/ml/'
        'machine-learning-databases/iris/iris.data', header=None)
```

```
#  Вывод последних 5 строк
df.tail()
```

	0	1	2	3	4
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

Набор данных содержит данные о трех видах цветков ириса. Т.к. персептрон реализует бинарный выход, оставим в наборе два класса цветков: ирис щетинистый (Iris-setosa) и ирис разноцветный (Iris-versicolor). Правило персептрона позволяет использовать произвольное количество входных признаков. Для упрощения расчетов оставим только два входных признака: длину чашелистика (sepal length) и длину лепестка (petal length). Создадим массив входов `x` из первых 100 строк набора и колонок с индексами 0 и 2:

```
# Выделение первого и третьего столбцов набора данных (длина
#  чашелистика и длины лепестка) в качестве признаков входного массива
#  и первые 100 строк: 2 вида цветков ириса
x = df.iloc[0:100, [0, 2]].values
```

Кроме того, столбец признаков содержит названия цветков. Создадим массив выходов из пятой колонки набора данных и перекодируем его в целочисленные метки 1 и -1:

```
# Выделение последнего столбца с индексом 4 в качестве выходного массива
y = df.iloc[0:100, 4].values
```

```
# Перевод выхода в целочисленные метки
y = np.where(y == 'Iris-setosa', -1, 1)
```

Построим точечный график параметров цветков (рис.2):

```
#  Построение точечного графика параметров цветков
plt.scatter(x[:50, 0], x[:50, 1],
            color='red', marker='o', label='щетинистый')
plt.scatter(x[50:100, 0], x[50:100, 1],
            color='blue', marker='x', label='разноцветный')
plt.xlabel('длина чашелистика [см]')
plt.ylabel('длина лепестка [см]')
plt.legend(loc='upper left')
plt.show()
```

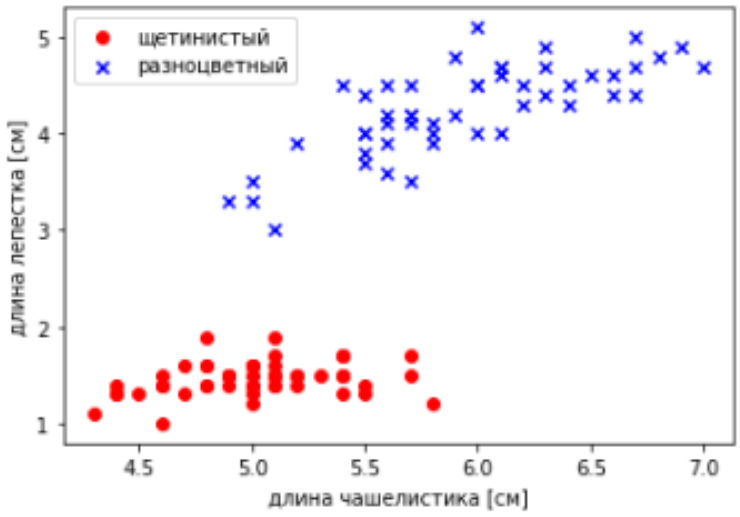


Рис.2 График параметров цветков

Подготовив данные, перейдем к созданию модели и ее обучению. Создадим объекта класса персептрона, передав ему в качестве параметров eta=0.1, n_iter=10 (число эпох).

```
#  Создание объекта класса персептрона
ppn = Perceptron(eta=0.1, n_iter=10)
```

Вызовем метод обучения модели, передав входной и выходной массивы набора данных:

```
#  Обучение модели
ppn.fit(x, y)
```

Выведем график, который показывает, как изменялась ошибка модели в ходе повторений обучения (рис.3)

```
#  Вывод графика ошибок классификации
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.xlabel('Эпохи')
plt.ylabel('Число случаев ошибочной классификации')
plt.show()
```

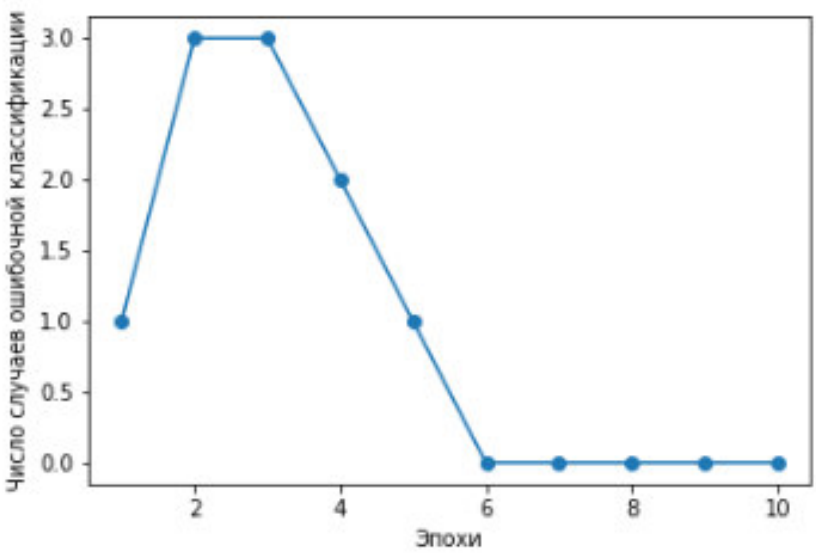


Рис. 3. График изменения ошибки в процессе обучения.

Из графика видно, что ошибка классификации вначале растет, затем падает, а после шестой эпохи перестает меняться. Т.е. персептрон выполняет правильные предсказания уже после шестой эпохи

Построим еще один вспомогательный график построения областей классификации (рис.4).

```
def plot_decision_regions(X, y, classifier, resolution=0.02):
    # определение маркеров
    markers = ('s', 'x', 'o', '^', 'v')
    # определение цветовой палитры и цветовой карты областей
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    # график поверхности решений
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    # вывод образцов классов
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    c=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')
# Вывод графика областей решения
plot_decision_regions(X, y, classifier=ppn)
plt.xlabel('длина чашелистика [см]')
plt.ylabel('длина лепестка [см]')
plt.legend(loc='upper left')
plt.show()
```

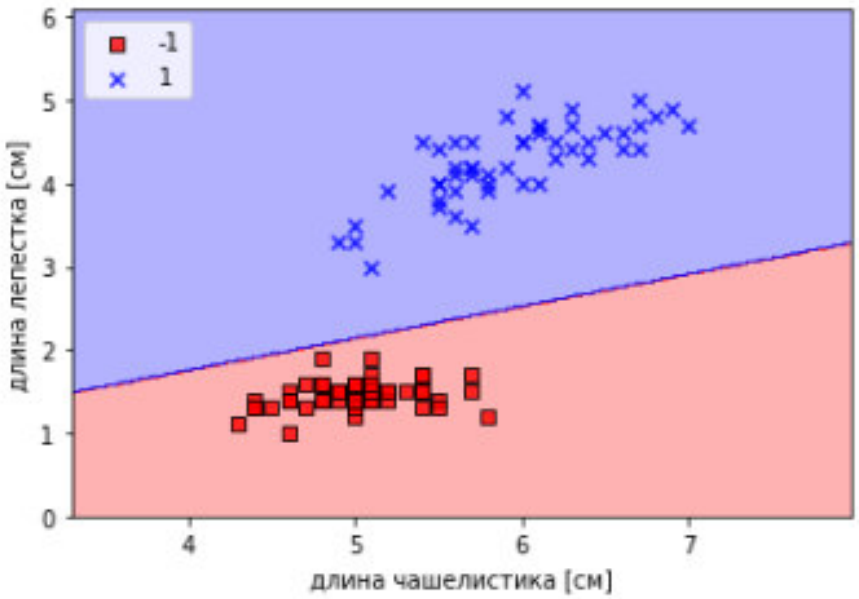


Рис.4 График областей классификации

Задание

1. Выполните эксперимент на модели персептрона:

- выберите другие признаковые переменные

- выберите другие 2 класса данных

2. Обучите модели и сравните с результатами приведенными в описании

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить эксперимент с моделью в соответствии с заданием
4. Выполнить сравнительный анализ полученных результатов

Содержание отчета

1. Программный код (блокнот) реализации персептрона
2. Результаты эксперимента с моделью и выводы сравнительного анализа

Практическое занятие № 51
Модель полносвязной нейронной сети для задачи бинарной классификации

Цель занятия. Получить представление о задаче бинарной классификации. Освоить методику бинарной классификации. Выполнить программную реализацию бинарной классификации на Python с использованием высокоуровневых библиотек tensorflow и keras

Краткие теоретические сведения

Смысловой анализ (или анализ мнений) относится к популярной подобласти более широкой области обработки естественного языка (NLP). Он занимается анализом направленности документов [1,2]. Распространенной задачей в анализе мнений является классификация документов на основе выраженных мнений или эмоций авторов относительно определенной темы.

На данном занятии будем рассматривать модель, которая будет классифицировать обзор фильма как *позитивный* или *негативный* на основе текста обзора. Это пример *бинарной* классификации (по двум классам), важной, и широко применяющейся задачи машинного обучения.

Для создания и обучения модели используется высокоуровневый API tf.keras [3], содержащийся в библиотеке моделей в TensorFlow [4].

Предполагается, что установлена библиотека Tensorflow версии 2.0 или выше (она включает библиотеку keras в качестве составной части). Установка библиотеки в среду разработки Anaconda выполните, как было описано на практическом занятии 19.

Подключение необходимых библиотек

Создадим новый блокнот и загрузим необходимые библиотеки и проверим их версию (в вашем случае версия может быть другой):

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
print("Версия tensorflow:",tf.__version__)
print("Версия keras:",keras.__version__)
```

```
Версия tensorflow: 2.3.0
Версия keras: 2.4.0
```

Загрузка набора данных IMDB

Библиотека keras содержит несколько наборов данных для типовых задач машинного обучения [5]. Набор данных IMDB включает 50 000 самых разных отзывов к кинолентам в базе фильмов (Internet Movie Database). Набор разбит на 25 000 обучающих и 25 000 контрольных отзывов, каждый набор на 50 % состоит из отрицательных и на 50 % из положительных

отзывов. Деление исходных данных на два набора (обучающий и контрольный) выполняют потому, что никогда не следует тестировать модель машинного обучения на тех же данных, которые использовались для ее обучения. Если модель прекрасно справляется с обучающими данными, это еще не значит, что она так же хорошо будет справляться с данными, которые прежде никогда не видела. Например, возможно, что модель просто *запомнит* соответствия между обучающими образцами и их целями, что совершенно бесполезно для задачи предсказания по данным, которые модель никогда не видела прежде.

Набор данных IMDB поставляется в составе TensorFlow и может быть загружен при помощи метода `load_data`. Он уже подготовлен таким образом, что обзоры (последовательности слов) конвертированы в последовательность целых чисел. Каждое число представляет конкретное слово в массиве. Кодировка слов выполняется в соответствии со словарем на 10000 наиболее употребимых слов.

Для загрузки набора данных добавим следующий код:

```
# Загрузка набора данных
imdb = keras.datasets.imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

Метод возвращает сразу 4 набора данных:

`train_data` - набор входных данных для обучения;

`train_label` - набор выходных данных (меток) для обучающего набора;

`test_data` - набор входных данных для проверки обученной модели;

`test_label` - набор выходных данных (меток) для проверочного набора;

Аргумент `num_words=10000` оставляет в тексте обзора фильма только 10000 наиболее часто встречающих слов. Все редкие слова из набора исключаются. Это позволяет ограничить объем данных разумными пределами.

Изучение загруженных данных

Проведем первичный анализ загруженных данных.

Выведем информацию о размере загруженного набора данных:

```
print("Обучающий набор. строк=", len(train_data), "колонок=",len(train_data[0]))
```

```
Обучающий набор. строк= 25000 колонок= 218
```

Данные каждого обзора, входящие в набор данных, представляют массив целых чисел, которые представляют код слова из словаря.

Разные обзоры фильмов содержат разное количество слов. Следующий код ниже показывает количество слов в первом и втором образцах (обзорах) набора данных:

```
print(len(train_data[0]), len(train_data[1]))
```

```
218 189
```

Вот пример того, как представлен первый обзор:

```
print(len(train_data[0])
```

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36,
 ...
8, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32]
```

Каждая метка набора данных является целым числом 0 или 1:

0 - негативный обзор, 1 - позитивный.

```
print("Кол-во меток обучающего набора:", len(train_labels))
print("Метки обзоров:", train_labels)
```

```
Кол-во меток обучающего набора: 25000
Метки обзоров: [1 0 0 ... 0 1 0]
```

Конвертируем целые числа представления обзора обратно в текст. Для этого используем следующую вспомогательную функцию `decode_review`, которая обращается к словарю для обратного представления индекса в слово:

```
# Назначим словарь, который будет отображать слова из массива данных
word_index = imdb.get_word_index()
# Зарезервируем первые несколько значений
word_index = {k:(v+3) for k,v in word_index.items()}
word_index["<PAD>"] = 0
word_index["<START>"] = 1
word_index["<UNK>"] = 2  # Вместо редких слов, не вошедших в набор из 10000, будет указано UNK
word_index["<UNUSED>"] = 3
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
def decode_review(text):
    return ' '.join([reverse_word_index.get(i, '?') for i in text])
```

С использованием функции decode_review отобразим оригинального текста первого обзора фильма (удаленные слова представлены UNK):

```
decode_review(train_data[0])
```

```
"<START> this film was just brilliant casting location scenery story direction
everyone's really suited the part they played and you could just imagine being
there robert <UNK> is an amazing actor and now the same being director <UNK> fa
ther came from the same scottish island as myself so i loved the fact there was
a real connection with this film the witty remarks throughout the film were gre
at it was just brilliant so much that i bought the film as soon as it was relea
sed for <UNK> and would recommend it to everyone to watch and the fly fishing w
as amazing really cried at the end it was so sad and you know what they say if
you cry at a film it must have been good and this definitely was also <UNK> to
the two little boy's that played the <UNK> of norman and paul they were just br
illiant children are often left out of the <UNK> list i think because the stars
that play them all grown up are such a big profile for the whole film but these
children are amazing and should be praised for what they have done don't you th
ink the whole story was so lovely because it was true and was someone's life af
ter all that was shared with us all"
```

Подготовка данных

Поскольку нейросеть может принимать только данные одинаковой длины, то необходимо скорректировать данные. Обзоры фильмов из массива целых чисел конвертируем в тензоры одинакового размера перед загрузкой в модель нейронной сети. Эта конвертация может быть сделана двумя способами:

– Конвертировать массивы в векторы 0 и 1 в формате *one-hot encoding*. Например, последовательность [3, 5] станет 10000-мерным вектором (определяется количеством возможных слов в словаре), полностью состоящим из нулей кроме позиций 3 и 5, которые будут заполнены единицами. Такой подход очень требователен к объему памяти, так как размер входной матрицы будет `num_words * num_reviews`;
– Сделать все массивы одинаковыми по длине, а затем создать тензор целых чисел с указанием `max_length * num_reviews`. Для такого подхода можно использовать *Embedding* ("Встроенный") слой, который может использовать эти параметры в качестве первого слоя сети.

Будем использовать второй способ. Для приведения всех обзоров фильмов к одинаковой длине используется функция `pad_sequences`, чтобы привести все длины к одному значению (256 элементов) дополнением последовательности до нужного размера нулями:

```
train_data = keras.preprocessing.sequence.pad_sequences(train_data,
value=word_index["<PAD>"],
padding='post',
maxlen=256)
test_data = keras.preprocessing.sequence.pad_sequences(test_data,
                                    value=word_index["<PAD>"],
padding='post',
maxlen=256)
```

В функции `pad_sequences` используются параметры:

value - значение, которым дополняется каждый обзор (вектор train_data[i]) до одинаковой длины. В примере word_index["<PAD>"] = 0
padding - место дополнения (pre - в начале, post - в конце последовательности)
maxlen - длина последовательности

Убедимся, что обзоры с индексами 0 и 1 после преобразования будут иметь одинаковую длину 256 элементов:

```
print(len(train_data[0]), len(train_data[1]))
```

```
256 256
```

Массив слов каждого обзора после преобразования будет дополнен нулями до фиксированного размера 256. В этом можно убедиться при выводе первого обзора:

```
print(train_data[0])
```

```
[   1   14   22   16   43  530  973 1622 1385   65  458 4468   66 3941
    4  173   36  256    5   25  100   43  838  112   50  670    2    9
    ...
  103   32   15   16 5345   19  178   32    0    0    0    0    0    0
    0    0    0    0]
```

Создание модели

Модель нейтронной сети включает стек слоев. При создании модели необходимо определить:

- сколько слоев будет использовано в модели
- сколько *скрытых нейронов* будет использовано в каждом слое

В рассматриваемом примере, входные данные состоят из массива слов (целых чисел). Получаемые предсказания представлены в виде меток 0 или 1. Сконструируем следующую модель и покажем ее структуру:

```
# Размер входных данных - кол-во слов, использ. в обзорах фильмов
vocab_size = 10000

model = keras.Sequential()
model.add(keras.layers.Embedding(vocab_size, 16, input_shape=(None,)))
model.add(keras.layers.GlobalAveragePooling1D())
model.add(keras.layers.Dense(16, activation=tf.nn.relu))
model.add(keras.layers.Dense(1, activation=tf.nn.sigmoid))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 16)          160000
_________________________________________________________________
global_average_pooling1d (Gl (None, 16)                0
_________________________________________________________________
dense (Dense)                (None, 16)                272
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 160,289
Trainable params: 160,289
Non-trainable params: 0
```

В модель включены следующие слои:

1) Первый слой Embedding принимает преобразованный массив обзоров и ищет соответствующий вектор для каждой пары слово/число. Модель обучается на этих векторах. Векторы увеличивают размерность получаемого массива на 1.

При добавлении слоя необходимо указать 3 аргумента:

- `input_dim`: это размер словаря текстовых данных (в нашем случае это `vocab_size = 10000`).

- `output_dim`: это размер векторного пространства, в которое будут преобразованы слова входных векторов. Он определяет размер выходного вектора этого слоя для каждого слова (в нашем случае выбрано 16). Можно проверить различные значения для конкретной задачи.
- `input_length`: Это длина входных последовательностей (число элементов, число образцов). В нашем случае число элементов 256, но как правило используется None и модель будет определять размер последовательности автоматически. Как правило, не указывают и число образцов, которое также определяется автоматически. Это делает модель универсальной под любой размер входных данных.

2) Следующий слой GlobalAveragePooling1D возвращает получаемый вектор заданной длины для каждого примера, усредняя размер ряда. Это позволит модели легко принимать данные разной длины

3) Следующий полносвязный слой Dense с 16 скрытыми нейронами принимает на вход вектор предыдущего слоя

4) Последний слой также является полносвязным, но всего с одним выходящим узлом. При помощи функции активации sigmoid (Сигмоида) на выходе будем получать число с плавающей запятой между 0 и 1, которое будет показывать вероятность отнесения отзыва к одному из классов

Созданная модель имеет 2 промежуточных или *скрытых* слоя, между входом и выходом данных. Количество выходов (узлов или нейронов) является размером репрезентативного пространства слоя. Другими словами, количество степеней свободы, которая разрешена сети во время обучения.

Если модель имеет больше скрытых узлов и больше слоев, то тогда нейросеть может обучиться более сложным представлениям. Однако в этом случае это будет дороже с точки зрения вычислительных ресурсов и может привести к переобучению, т.е. обучению распознавания нежелательных образцов (тех, которые улучшают показатели на обучающих данных, но не на проверочных).

Определение функции потерь и оптимизатора

Для модели необходимо указать функцию потерь и оптимизатор для обучения. Поскольку задача является примером бинарной классификации и модель будет выдавать вероятность (выходной слой имеет один выход с сигмоидой в качестве функции активации), то мы воспользуемся функцией потерь `binary_crossentropy` (перекрестная энтропия). Она измеряет "дистанцию" между распределениями вероятностей (между эталоном и предсказаниями). Это не единственный выбор для функции потерь: можно, например, выбрать `mean_squared_error` (среднеквадратическая ошибка). Обычно перекрестная энтропия лучше справляется с вероятностями.

Функция потерь среднеквадратичской ошибки часто используется в задачах регрессии, как будет показано на последнем занятии.

В качестве оптимизатора будем использовать оптимизатор Адама. Выполним настройку (компиляцию) модели:

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Создание проверочного набора данных

Во время обучения следует проверить точность модели на данных, которых она еще не видела. Для этого создадим *проверочный набор* данных, выделив 10000 примеров из оригинального тренировочного сета в отдельный набор. Настройка модели производится с использованием только данных для обучения, и только потом используется проверочный набор для оценки точности.

```
x_val = train_data[:10000]
partial_x_train = train_data[10000:]

y_val = train_labels[:10000]
partial_y_train = train_labels[10000:]
```

Обучение модели

Начнем обучение модели с 40 эпох при помощи мини-батчей (пакетов) по 512 образцов. Это означает, что выполняется 40 итераций (или проходов) по всем образцам данных в `partial_x_train` и `partial_y_train`:

```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=40,
                    batch_size=512,
                    validation_data=(x_val, y_val),
                    verbose=1)
```

Ход обучения с его результатами выводится в консоль:

```
Epoch 1/40
30/30 [==============================] - 1s 36ms/step - loss: 0.6918 - accuracy: 0.5601 - val_loss:
0.6897 - val_accuracy: 0.5871
Epoch 2/40
30/30 [==============================] - 1s 22ms/step - loss: 0.6855 - accuracy: 0.6824 - val_loss:
0.6811 - val_accuracy: 0.7073

 . . .
Epoch 40/40
30/30 [==============================] - 1s 23ms/step - loss: 0.0904 - accuracy: 0.9769 - val_loss:
0.3151 - val_accuracy: 0.8806
```

Построим временной график точности и потерь. Метод model.fit() возвращает объект History, который содержит все показатели, которые были записаны в лог во время обучения:

```
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Здесь всего четыре показателя, по одному для каждой отслеживаемой метрики во время обучения и проверки. Можно использовать их, чтобы построить графики потерь (loss) и точности (accuracy) обоих стадий для сравнения.

График потерь во время обучения представлен на рис. 1, а точности на рис.2. На графиках точками отмечены потери и точность модели во время обучения, а линией - во время проверки.

```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
# "bo" означает "blue dot", синяя точка
plt.plot(epochs, loss, 'bo', label='Потери обучения')
# "b" означает "solid blue line", непрерывная синяя линия
plt.plot(epochs, val_loss, 'b', label='Потери проверки')
plt.title('Потери во время обучения и проверки')
plt.xlabel('Эпохи')
plt.ylabel('Потери')
plt.legend()
plt.show()
```

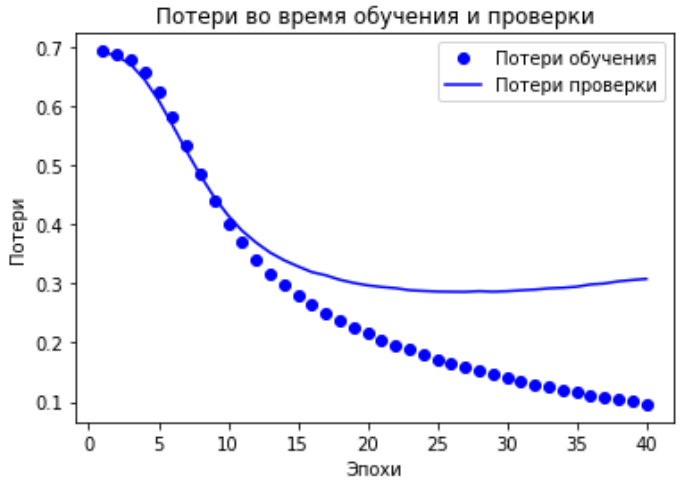


Рис. 1 График потерь во время обучения

```
plt.plot(epochs, acc, 'bo', label='Точность обучения')
plt.plot(epochs, val_acc, 'b', label='Точность проверки')
plt.title('Точность во время обучения и проверки')
plt.xlabel('Эпохи')
plt.ylabel('Точность')
plt.legend()
plt.show()
```

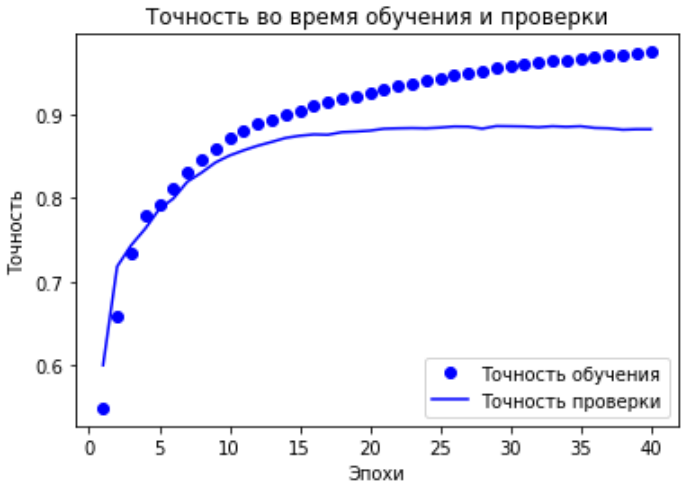


Рис. 2 График точности во время обучения

Обратите внимание на то, что потери во время обучения *уменьшаются*, а точность – *увеличивается* с каждой следующей эпохой. Это вполне ожидаемо, поскольку использовался градиентный спуск *gradient descent* - он минимизирует показатели потерь с каждой итерацией настолько быстро, насколько это возможно.

Но если посмотреть на потери и точность во время проверки модели: после приблизительно 20 эпох они находятся на пике, а далее уменьшаются. Это явный пример переобучения: модель показывает более лучшие показатели на данных для обучения, нежели на новых, которых она еще не видела. После этого момента модель начинает переоптимизироваться и обучается представлениям, которые *свойственны* только данным обучения. Таким образом, модель не учится *обобщать* новые, проверочные данные.

Именно здесь можно предотвратить переобучение просто прекратив тренировку сразу после 20 эпох обучения. Это можно сделать автоматически при помощи `callback`, функции обратного вызова.

Оценка точности модели

Теперь, когда обучение прошло успешно, проверим точность модели на 10000 образцах из контрольного набора данных. Метод проверки будет возвращать 2 значения: потери *loss* (число, которое показывает ошибку, чем оно ниже, тем лучше), и точность *accuracy (*чем она выше, тем лучше).

```
results = model.evaluate(test_data, test_labels)
print(results)
```

```
782/782 [==============================] - 1s 930us/step - loss: 0.3341 - accuracy: 0.8715
[0.33406832814216614, 0.8714799880981445]
```

Как видно, этот достаточно наивный подход к построению модели достиг точности около 87%. Если бы мы использовали более сложные методы, то модель приблизилась бы к отметке в 95%.

Сохранение и восстановление модели

Сохраним модель целиком в единый файл, который будет содержать все веса обученной модели, конфигурацию модели и оптимизатор конфигурации. Это позволит впоследствии восстановить модель и использовать ее для решения задачи классификации без предварительного обучения.

В Keras есть встроенный формат для сохранения модели при помощи стандарта HDF5. Сохраним модель.

```
model.save('путь\model_one_class.h5')
```

Создадим новую модель из сохраненного файла:

```
new_model = keras.models.load_model('путь\ model_one_class.h5')
```

Проверим модель, загруженную из файла:

```
loss, acc = new_model.evaluate(test_data, test_labels)
print("Восстановленная модель, точность: {:5.2f}%".format(100*acc))
```

```
782/782 [==============================] - 1s 930us/step - loss: 0.3341 - accuracy: 0.8715
Восстановленная модель, точность: 87.15%
```

Видно, что восстановленная модель дает такие же результаты, как и ранее обученная модель.

Задание.

1. Постройте модель заданной архитектуры, обучите и проверьте ее точность. Сохраните модель. Восстановите модель и сравните точность обеих моделей.

Измените параметры архитектуры модели сети и выберите оптимальные значения

- В данном примере использовался один скрытый слой Dense. Попробуйте использовать два или три и посмотрите, как это повлияет на точность на этапах обучения и проверки.
- Попробуйте использовать слои с большим или с меньшим количеством скрытых нейронов: 32 нейрона, 64 нейрона и т. д.
- Попробуйте вместо `binary_crossentropy` использовать функцию потерь `mean_squared_error`
- Попробуйте вместо `relu` в промежуточном слое использовать функцию активации `tanh` (она была популярна на заре становления нейронных сетей).

Список источников

1. Рашка С., Мирджалили В. Python и машинное обучение.- СПб.: ООО «Диалектика», 2019.
2. Шолле Ф. Глубокое обучение на Python.- СПб.: Питер, 2019

3. Библиотека keras.- https://keras.io/getting_started/
4. Библиотека tensorflow.- https://www.tensorflow.org/resources/learn-ml
5. Наборы данных для машинного обучения keras.- https://keras.io/api/datasets/

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить эксперимент с моделью в соответствии с заданием
4. Выполнить сравнительный анализ полученных результатов

Содержание отчета

1. Программный код (блокнот) решения задачи бинарной классификации
2. Результаты эксперимента с моделью и выводы сравнительного анализа

Практическое занятие № 52
Модель полносвязной нейронной сети для задачи многомерной классификации

Цель занятия. Получить представление о задаче многомерной классификации. Освоить методику многомерной классификации. Выполнить программную реализацию многомерной классификации на Python с использованием высокоуровневых библиотек tensorflow и keras

Краткие теоретические сведения

На предыдущем занятии рассматривалась классификация входных данных на два взаимоисключающих класса с использованием полносвязной нейронной сети. Рассмотрим создание и обучение модели нейросети, которая классифицирует изображения одежды. Так как теперь количество классов больше двух, эта задача относится к категории задач *многомерной классификации* [1]. И, поскольку каждый экземпляр данных должен быть отнесен только к одному классу, эта задача является примером *однозначной многомерной классификации*. Если бы каждый экземпляр данных мог принадлежать нескольким классам (в данном случае темам), эта задача была бы примером *многозначной многомерной классификации*.

Как и на предыдущем занятии для построения и обучения моделей В работе используется библиотека tensorflow.keras - высокоуровневый API в TensorFlow [2,3].

Подключение необходимых библиотек

Создадим новый блокнот и загрузим необходимые библиотеки:

```
# TensorFlow и tf.keras
import tensorflow as tf
from tensorflow import keras
# Вспомогательные библиотеки
import numpy as np
import matplotlib.pyplot as plt
```

Загрузка набора данных Fashion MNIST

Набор данных Fashion MNIST [4] содержит 70,000 монохромных изображений в 10 категориях. На каждом изображении содержится по одному предмету одежды в низком разрешении 28 на 28 пикселей. Fashion MNIST является альтернативным набором вместо классического набора данных рукописных цифр MNIST, который часто используют в программах машинного обучения для компьютерного зрения.

Набор данных Fashion MNIST поставляется в составе TensorFlow и может быть загружен при помощи метода load_data. Набор данных Fashion MNIST разбит следующим образом:

– 60,000 изображений для обучения нейросети (train_images);

10,000 изображений для проверки модели нейросети, чтобы оценить, насколько правильно сеть обучилась их классифицировать (test_images).

Для загрузки набора данных добавим следующий код:

```
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

Процедура загрузки возвращает четыре массива NumPy:

- массивы *обучающего набора* train_images и train_labels;
- массивы *проверочного набора* test_images и test_labels.

Изображения (`images`) являются 28x28 массивами NumPy, где значение пикселей варьируется от 0 до 255. *Метки* (`labels`) - это массив целых чисел от 0 до 9. Они соответствуют *классам* одежды изображенной на картинках:

Label	Класс
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Каждому изображению соответствует единственная метка. Так как *названия классов* не включены в набор данных, заполним массив названиями для дальнейшего использования при построении изображений:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Изучение данных

Проанализируем формат данных перед обучением модели. Воспользовавшись свойством shape, можно увидеть, что в обучающем наборе данных 60000 изображений, каждое размером 28 x 28 пикселей:

```
train_images.shape
```

```
(60000, 28, 28)
```

Соответственно, в обучающем наборе 60000 меток и каждая метка это целое число от 0 до 9:

```
train_labels
```

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

Проверочный набор содержит 10,000 изображений, каждое имеет такой же размер 28 на 28 пикселей, а в проверочном наборе 10000 меток:

```
print("Проверочный набор: ",test_images.shape)
print("Метки пров.набор: ",len(test_labels))
```

```
Проверочный набор:  (10000, 28, 28)
Метки пров.набор:  10000
```

Подготовка данных

Данные должны быть подготовлены перед обучением нейросети. Если проанализировать первое изображение в обучающем наборе, то можно увидеть, что значения пикселей находятся в диапазоне от 0 до 255. Это видно по изображению первого образца (рис.1).

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```

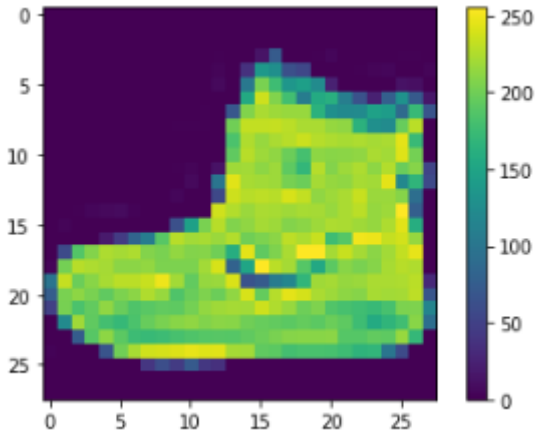


Рис.1 Изображение одного образца

Необходимо масштабировать эти значения - привести к диапазону от 0 до 1. Для этого необходимо разделить значения на 255. Важно, чтобы *обучающий набор* и *проверочный набор* были в одинаковом масштабе:

```
train_images = train_images/255.0
test_images = test_images/255.0
```

Чтобы убедиться, что данные в правильном формате для построения и обучения нейросети, выведем на экран первые 9 изображений из *обучающего набора* и отобразим под ними наименования их классов (рис.2).

```
plt.figure(figsize=(5,5))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

Рис.2 Изображений из обучающего набора с метками

Построение модели

Построение модели нейронной сети требует правильной конфигурации каждого слоя, и последующей компиляции модели.

Базовым строительным блоком нейронной сети является *слой*. Слои извлекают образы из данных, которые в них подаются.

Большая часть глубокого обучения состоит из соединения в последовательность простых слоев. Большинство слоев, таких как `tf.keras.layers.Dense`, имеют параметры, которые настраиваются во время обучения.

Создадим модель и выведем ее структуру:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
```

Первый слой этой сети tf.keras.layers.Flatten, пробразует формат изображения из двумерного массива (28 на 28 пикселей) в одномерный (размерностью 28 * 28 = 784 пикселя). Слой извлекает строки пикселей из массива изображения (ранг 2) и выстраивает их в вектор (ранг 1). Этот слой не имеет параметров для обучения; он только переформатирует данные.

После преобразования пикселей, нейросеть содержит два слоя tf.keras.layers.Dense. Это полносвязные нейронные слои. Первый Dense слой состоит из 128 узлов (или нейронов), на выходе слоя используется функция активации *relu*. Второй слой (и последний) имеет 10 узлов, на выходе слоя используется функция активации *softmax*, которая возвращает массив из 10 вероятностных оценок дающих в сумме 1. Т.е. каждый узел выходного слоя содержит оценку указывающую вероятность принадлежности изображения к одному из 10 классов.

Компилирование модели

Прежде чем модель будет готова для обучения, нужно указать еще несколько параметров. Они добавляются на шаге компилирования модели *compile*:

- *Функция потерь (Loss function)* — измеряет точность модели во время обучения. Требуется минимизировать эту функцию чтобы "направить" модель в верном направлении.
- *Оптимизатор (Optimizer)* — определяет метод обновления модели на основе входных данных и функции потерь.
- *Метрики (Metrics)* — используются для мониторинга обучения и и тестирования модели. Будем использовать метрику *accuracy* равную доле правильно классифицированных изображений.

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

В модели использованы следующие параметры.

Оптимизатор `adam`, реализующий алгоритм Адама. Это метод стохастического градиентного спуска, основанный на адаптивной оценке моментов первого и второго порядка, эффективен по скорости, требует небольшого объема памяти, инвариантен к диагональному масштабированию

градиентов и хорошо подходит для задач с большим объемом данных и параметров

Функции потерь `sparse_categorical_crossentropy` - вычисляет потерю кроссэнтропии между метками и прогнозами. Эта функция используется, когда существует два или более классов меток. Метки должны быть представлены как целые числа. Это вероятностная функция. Другой популярной вероятностной функцией является `categorical_crossentropy`, которая вычисляет потерю кроссэнтропии между метками и прогнозами. Эта функция также используется, когда существует два или более классов меток. Метки должны быть представлены в формате `one_hot` (представление категориальных данных в виде двоичных векторов).

В качестве метрики используется точность модели.

Выделение проверочного набора

Для контроля точности модели создадим проверочный набор, выбрав 10000 образцов из набора обучающих данных 60000 образцов.

```
x_val = train_images[:10000]
partial_x_train = train_images[10000:]
y_val = train_labels[:10000]
partial_y_train = train_labels[10000:]
```

Обучение модели

Обучение модели нейронной сети требует выполнения следующих шагов:

– Передать обучающий и проверочные наборы на вход модели. В этом примере обучающие данные: массивы `partial_x_train` и `partial_y_train`, проверочные данные: `x_val` и `y_val`.

– Модель учится связывать изображения с правильными классами. Проведем обучение модели в течение 20 эпох.

– В заключение обученная модель используется для выполнения прогнозов на проверочных данных `test_images`. Проверяется, соответствуют ли предсказанные классы меткам из массива `test_labels`.

Для начала обучения вызывается метод `model.fit`:

```
history = model.fit(partial_x_train,
   partial_y_train,
   epochs=20,
   batch_size=512,
   validation_data=(x_val, y_val))
```

Ход обучения с его результатами выводится в консоль:

```
Epoch 1/20
98/98 [==============================] - 3s 27ms/step - loss: 0.7406 - accuracy: 0.7524 - val_loss:
0.5110 - val_accuracy: 0.8267
Epoch 2/20
98/98 [==============================] - 2s 23ms/step - loss: 0.4743 - accuracy: 0.8387 - val_loss:
0.4537 - val_accuracy: 0.8431
      . . .
Epoch 20/20
98/98 [==============================] - 2s 22ms/step - loss: 0.2647 - accuracy: 0.9054 - val_loss:
0.3238 - val_accuracy: 0.8851
```

В процессе обучения модели отображаются метрики потери (loss) и точности (accuracy) на обучающих данных, метрики потери (val_loss) и точности (val_accuracy) на проверочных данных. Эта модель достигает на обучающих данных точности равной приблизительно 0.91 (91%).

Вызов `model.fit()` возвращает объект History. Этот объект имеет поле `history` — словарь с данными обо всем происходившем в процессе обучения. Выведем обозначение основных метрик:

```
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Словарь содержит четыре элемента— по одному на метрику, — за которыми осуществлялся мониторинг в процессе обучения и проверки.

Выведем графики кривых потерь и точности. С использованием библиотеки Matplotlib выведем графики потерь (рис. 3) и графики точности на этапах обучения и проверки (рис. 4). Результаты могут отличаться от приведенных на рисунках, что обусловлено различием в случайных числах, использовавшихся для инициализации сети.

Выведем графики кривых потерь:

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Потери на этапе обучения')
plt.plot(epochs, val_loss, 'b', label=' Потери на этапе проверки')
plt.title('Потери при обучении и проверке')
plt.xlabel('Эпохи')
plt.ylabel('Потери')
plt.legend()
plt.show()
```

и точности:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'bo', label='Точность на этапе обучения')
plt.plot(epochs, val_acc, 'b', label=' Точность на этапе проверки')
plt.title('Точность при обучении и проверке')
plt.xlabel('Эпохи')
plt.ylabel('Точность')
plt.legend()
plt.show()
```

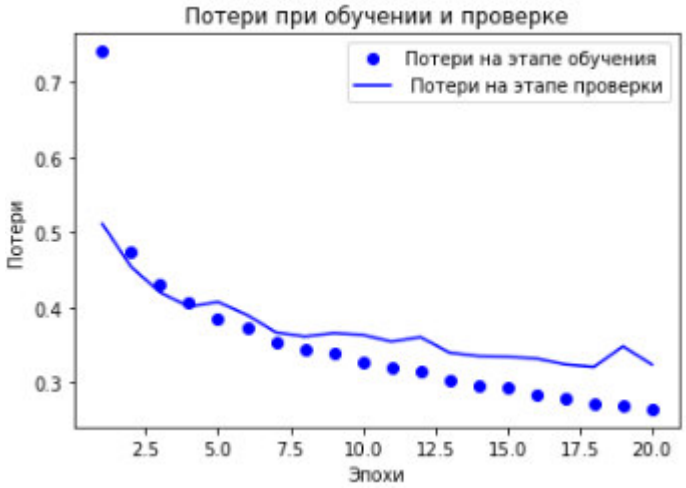


Рис.3. Потери на этапах обучения и проверки

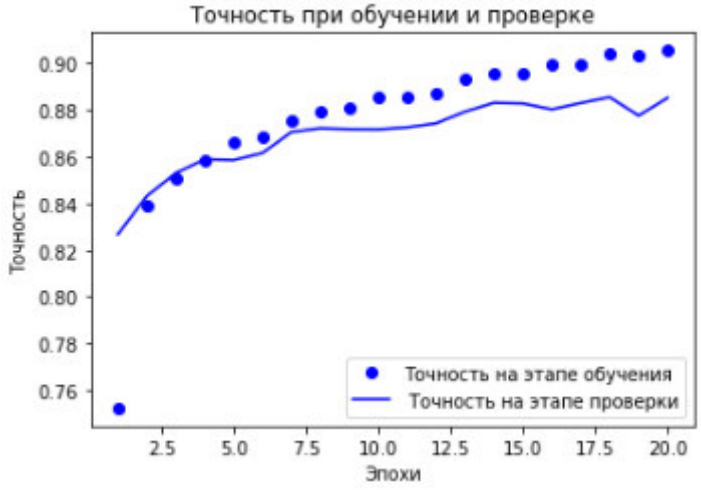


Рис.4. Точность на этапах обучения и проверки

Если посмотреть на потери и точность во время проверки модели, то после приблизительно 12 эпох они находятся на пике, а далее практически не изменяются и становятся хуже показателей обучения (которые постоянно улучшаются). Это явный пример переобучения: модель показывает более лучшие показатели на данных для обучения, нежели на новых, которых она еще не видела. После этого момента модель начинает переоптимизироваться и обучается представлениям, которые *свойственны* только данным обучения. Таким образом, модель не учится *обобщать* новые, проверочные данные.

Оценка точности модели

Определим точность модели на контрольном (тестовом) наборе:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('\nТочность на проверочных данных:', test_acc)
```

```
313/313 [==============================] - 0s 2ms/step - loss: 0.3544 - accuracy: 0.8752

Точность на проверочных данных: 0.8751999735832214
```

Полученная на проверочном наборе точность оказалась немного ниже, чем на обучающем наборе. Этот разрыв между точностью на тренировке и тесте является примером *переобучения (overfitting)*. Переобучение возникает, когда модель машинного обучения показывает на новых данных худший результат, чем на тех, на которых она обучалась.

Использование модели для предсказания

Теперь, когда модель обучена, ее можно использовать для выполнения предсказаний на контрольном наборе:

```
predictions = model.predict(test_images)
```

В данной команде обученная модель выполняет предсказание класса одежды для каждого изображения в проверочном наборе. Выведем первое предсказание:

```
np.round(predictions[0],3)
```

```
array([0.   , 0.   , 0.   , 0.   , 0.   , 0.01 , 0.   , 0.035, 0.001,
       0.955], dtype=float32)
```

Прогноз представляет из себя массив из 10 чисел. Они описывают «уверенность» модели в том, насколько изображение соответствует каждому из 10 разных видов одежды. Можно посмотреть какой метке соответствует максимальное значение:

```
np.argmax(predictions[0])
```

```
9
```

Модель полагает, что на первой картинке изображен ботинок (ankle boot), или класс 9. Проверка показывает, что классификация верна:

```
test_labels[0]
```

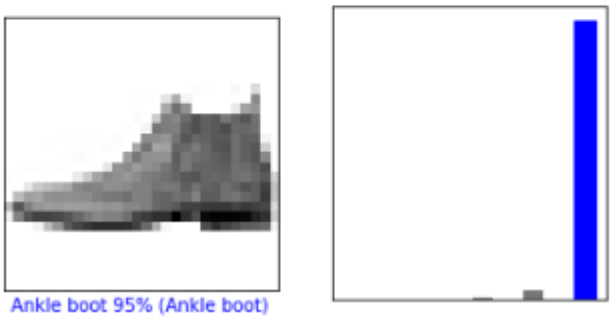
```
9
```

Построим график, чтобы взглянуть на полный набор из 10 предсказаний классов.

```
# График изображений
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(img, cmap=plt.cm.binary)
  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)
# Диаграмма вероятностей классов
def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array[i], true_label[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

Выведем изображение, предсказание и массив предсказаний для образца контрольного набора с индексом 0.

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
plt.show()
```



Сверим несколько изображений с их прогнозами. Цвет верных предсказаний синий, а неверных - красный. Число это процент уверенности (от 100) для предсказанной метки. Отметим, что модель может ошибаться, даже если она очень уверена (рис.5).

```
# Отображаем первые X тестовых изображений, их предсказанную и настоящую метки.
# Корректные предсказания окрашиваем в синий цвет, ошибочные в красный.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions, test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions, test_labels)
plt.show()
```

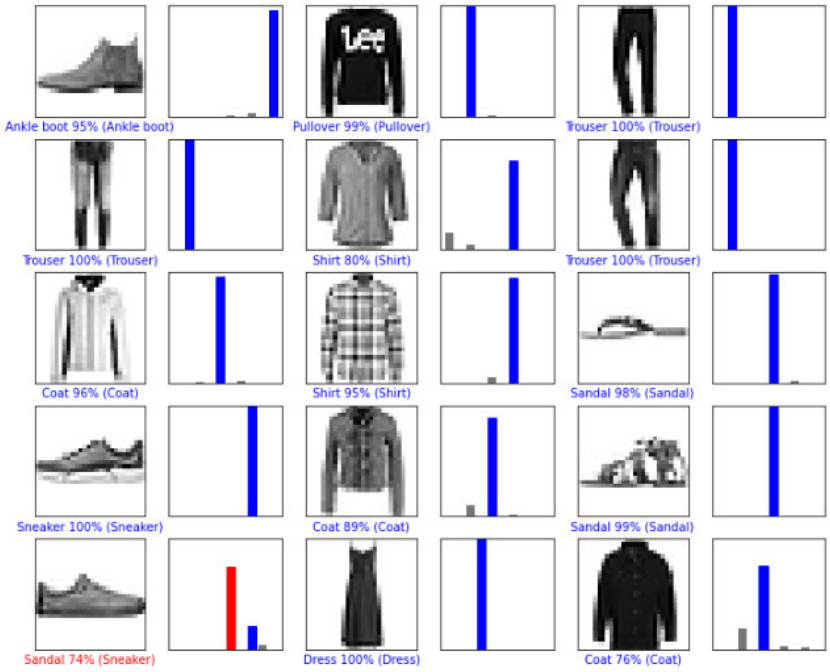


Рис.5. Сверка нескольких изображений с их прогнозами

Задание.

1. Постройте модель заданной архитектуры, обучите и проверьте ее точность.

Измените параметры архитектуры модели сети и выберите оптимальные значения

- В модели использовались два скрытых слоя. Попробуйте использовать один или три и посмотрите, как это повлияет на ее точность на этапах обучения и проверки.
- Попробуйте использовать первый слой с меньшим количеством скрытых нейронов: 64, 32, 16
- Попробуйте вместо функции потерь *sparse_categorical_crossentropy* использовать функцию *binary_crossentropy* или *mean_squared_error*.
- Попробуйте вместо *relu* использовать функцию активации *tanh* (она была популярна на заре становления нейронных сетей).

2. Создайте и обучите модель множественной классификации на одном из наборов данных: изображений рукописных цифр [5] или малых изображений [6] по методике практического занятия

Список источников

1. Шолле Ф. Глубокое обучение на Python.- СПб.: Питер, 2019
2. Библиотека keras.- https://keras.io/getting_started/
3. Библиотека tensorflow.- https://www.tensorflow.org/resources/learn-ml
4. Набор данных FASHION-MNIST.- https://github.com/zalandoresearch/fashion-mnist
5. Набор данных рукописных цифр.- https://keras.io/api/datasets/mnist/
6. Набор данных малых изображений.- https://keras.io/api/datasets/cifar10/

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить эксперимент с моделью в соответствии с заданием
4. Выполнить сравнительный анализ полученных результатов
5. Создать и обучить модель многомерной классификации изображений рукописных цифр [5] или малых изображений [6] по методике практического занятия

Содержание отчета

1. Программный код (блокнот) решения задачи многомерной классификации
2. Результаты эксперимента с моделью и выводы сравнительного анализа
3. Программный код (блокнот) решения задачи многомерной классификации на данных изображений рукописных цифр или малых изображений

Практическое занятие № 53
Модель полносвязной нейронной сети для задачи регрессии

Цель занятия. Получить представление о задаче регрессии в машинном обучении. Освоить методику создания и обучения регрессионной модели. Выполнить программную реализацию регрессионной модели на Python с использованием высокоуровневых библиотек tensorflow и keras

Краткие теоретические сведения

На двух предыдущих занятиях рассматривались задачи классификации, цель которых состояла в предсказании одной дискретной метки для образца входных данных. Другим распространенным типом задач машинного обучения является *регрессия*, которая заключается в предсказании не дискретной метки, а значения на непрерывной числовой прямой: например, предсказание температуры воздуха на завтра по имеющимся метеорологическим данным или предсказание размера прибыли по данным за предшествующие периоды [1]. Не надо путать регрессию с алгоритмом логистической регрессии. Логистическая регрессия не является регрессионным алгоритмом — это алгоритм классификации.

Как и на предыдущем занятии для построения и обучения моделей В работе используется библиотека tensorflow.keras - высокоуровневый API в TensorFlow [2,3].

Подключение необходимых библиотек

Создадим новый блокнот и загрузим необходимые библиотеки:

```
# TensorFlow и tf.keras
import tensorflow as tf
from tensorflow import keras
# Вспомогательные библиотеки
import numpy as np
import matplotlib.pyplot as plt
```

Загрузка набора данных с ценами на жилье в Бостоне

Будем использовать набор данных Boston Housing, взятый из библиотеки StatLib, которая хранится в Университете Карнеги-Меллона. Образцы содержат 13 атрибутов домов в разных местах пригорода Бостона в конце 1970-х годов. Целью задачи регрессии является предсказание медианную цену дома по таким данным о пригороде того времени, как уровень преступности, ставка местного имущественного налога и т. д.

Для загрузки набора данных добавим следующий код:

```
boston_housing=keras.datasets.boston_housing
(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
```

Выведем информацию о загруженных данных (формы обучающего и контрольного наборов):

```
print("Обучающий набор:",train_data.shape)
print("Тестовый набор:",test_data.shape)
```

```
Обучающий набор: (404, 13)
Тестовый набор: (102, 13)
```

Набор данных для данной задачи не столь объемен, как в задачах классификации. Он содержит 506 образцов, разбитых на 404 обучающих и 102 контрольных образца. Каждый образец имеет 13 числовых признака: уровень преступности, среднее число комнат в доме, удаленность от центральных дорог и т. д. И каждый признак во входных данных имеет свой масштаб (диапазон изменения). Например, некоторые признаки являются пропорциями и имеют значения между 0 и 1, другие — между 1 и 12 и т. д.

Целевой показатель — медианные значения цен на дома, занимаемые собственниками, в тысячах долларов (последнее значение):

```
train_targets
```

```
array([15.2, 42.3, 50. , 21.1, 17.7, 18.5, 11.3, 15.6, 15.6, 14.4, 12.1,
       17.9, 23.1, 19.9, 15.7,  8.8, 50. , 22.5, 24.1, 27.5, 10.9, 30.8,
        . . .
       11.8, 24.4, 13.8, 19.4, 25.2, 19.4, 19.4, 29.1])
```

Цены находятся в диапазоне от 10 до 50 тыс. долларов.

Подготовка данных

Сеть может автоматически адаптироваться к таким разнородным данным, однако это усложнит обучение. Поэтому на практике к таким данным принято применять нормализацию: для каждого признака во входных данных (столбца в матрице входных данных) из каждого значения вычитается среднее по этому признаку, и разность делится на стандартное отклонение, в результате признак центрируется по нулевому значению и имеет стандартное отклонение, равное единице. Такую нормализацию легко выполнить с помощью Numpy:

```
mn = train_data.mean(axis=0)
train_data_norm = train_data-mn
std = train_data.std(axis=0)
train_data_norm /= std

test_data_norm = test_data-mn
test_data_norm /= std
```

Например, данные первого образца до нормализации:

```
train_data[0]
```

```
array([  1.23247,   0.     ,   8.14   ,   0.     ,   0.538  ,   6.142  ,
        91.7    ,   3.9769 ,   4.     , 307.     ,  21.     , 396.9    ,
        18.72   ])
```

и после нормализации:

```
np.round(train_data_norm[0],4)
```

```
array([-0.2722, -0.4836, -0.4358, -0.2568, -0.1652, -0.1764,  0.8131,
        0.1167, -0.6262, -0.5952,  1.1485,  0.4481,  0.8252])
```

Обратите внимание на то, что величины, используемые для нормализации контрольных данных, вычисляются с использованием обучающих данных. Никогда не следует использовать в работе какие-либо значения, вычисленные по контрольным данным, даже для таких простых шагов, как нормализация данных.

Конструирование сети

Из-за небольшого количества образцов будет использоваться очень маленькая сеть с двумя четырехмерными промежуточными слоями. Вообще говоря, чем меньше обучающих данных, тем скорее наступит переобучение, а использование маленькой сети — один из способов борьбы с ним.

Создадим функцию построения модели:

```
from tensorflow.keras import models
from tensorflow.keras import layers
def build_model():
    model = models.Sequential()
    model.add(layers.Dense(64, activation='relu',
        input_shape=(train_data.shape[1],)))
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(1))
    model.compile(optimizer='rmsprop', loss='mse',
            metrics=['mae'])
    return model
```

Сеть заканчивается одномерным слоем, не имеющим функции активации (это линейный слой). Это типичная конфигурация для задачи регрессии, целью которой является предсказание одного значения на непрерывной числовой прямой. Применение функции активации могло бы ограничить диапазон выходных значений: например, если в последнем слое применить функцию активации *sigmoid*, сеть обучилась бы предсказывать только значения из диапазона между 0 и 1. В данном случае, с линейным последним слоем, сеть способна предсказывать значения из любого диапазона.

Обратите внимание на то, что сеть компилируется с функцией потерь mse — *mean squared error* (среднеквадратичная ошибка), вычисляющей

квадрат разности между предсказанными и целевыми значениями. Эта функция широко используется в задачах регрессии.

Также включен новый параметр в мониторинг на этапе обучения: mae — *mean absolute error* (средняя абсолютная ошибка). Это абсолютное значение разности между предсказанными и целевыми значениями. Например, значение MAE, равное 0,3, в этой задаче означает, что в среднем прогнозы отклоняются на 300 долларов США.

Оценка решения методом перекрестной проверки по K блокам

Чтобы оценить качество сети в ходе корректировки ее параметров (например, количество эпох обучения), можно разбить исходные данные на обучающий и проверочный наборы, как это делалось в предыдущих примерах. Но так как использовался небольшой набор данных, проверочный набор получился бы слишком маленьким (например, около 100 образцов). Как следствие, оценки при проверке могут сильно меняться в зависимости от того, какие данные попадут в проверочный и обучающий наборы: оценки при проверке могут иметь слишком большой разброс. Такой маленький объем не позволит надежно оценить качество модели.

Лучшей практикой в таких ситуациях является применение *перекрестной проверки по K блокам* (K-fold cross-validation), как показано на рис. 1. Суть ее заключается в разделении доступных данных на K блоков (обычно K = 4 или 5), создании K идентичных моделей и обучении каждой на K—1 блоках с оценкой по оставшемуся блоку. По полученным K оценкам вычисляется среднее значение, которое принимается как оценка модели. В программном коде такая проверка реализуется достаточно просто [1].

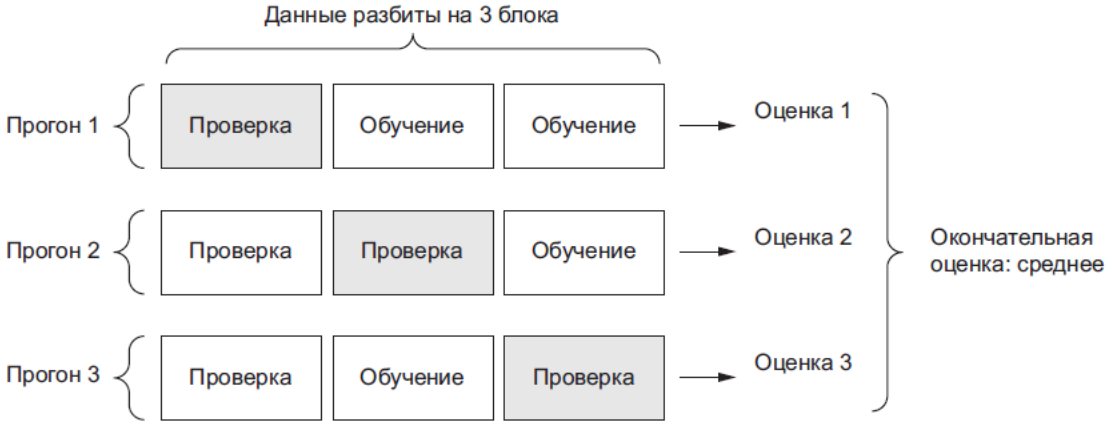


Рис. 1 Перекрестная проверка по трем блокам

Программный код для перекрестной проверки по K блокам:

```
k = 4
num_val_samples = len(train_data_norm) // k
num_epochs = 100
all_scores = []
for i in range(k):
    print('processing fold #', i)
    # Подготовка проверочных данных: данных из блока k
    val_data=train_data_norm[i*num_val_samples:(i+1)*num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1)
             * num_val_samples]
    # Подготовка обучающих данных
    partial_train_data = np.concatenate(
        [train_data_norm[:i * num_val_samples],
        train_data_norm[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
          [train_targets[:i * num_val_samples],
          train_targets[(i + 1) * num_val_samples:]],
          axis=0)
    # Конструирование модели Keras (уже скомпилированной)
    model = build_model()
    # Обучение модели (в режиме «без вывода сообщений»  verbose=0)
    model.fit(partial_train_data, partial_train_targets,
            epochs=num_epochs, batch_size=1, verbose=0)
    #  Оценка модели по проверочным данным
    val_mse,val_mae=model.evaluate(val_data,val_targets,
           verbose=0)
    all_scores.append(val_mae)
```

Обучение с перекрестной проверкой с num_epochs = 100 займет некоторое время. Выведем оценки модели на проверочных данных для каждой итерации перекрестной проверки:

```
np.round(all_scores,4)
```

```
array([2.1251, 2.4061, 2.3697, 2.5901])
```

и усредненное значение

```
np.round(np.mean(all_scores),4)
```

```
2.3728
```

Разные прогоны действительно показывают разные оценки, от 2.1 до 2.6. Средняя оценка 2.4 выглядит более достоверно, чем любая из оценок отдельных прогонов, — в этом главная ценность перекрестной проверки по K блокам. В данном случае средняя ошибка составила 2400 долларов, что довольно много, если вспомнить, что цены колеблются в диапазоне от 10 000 до 50 000 долларов.

Попробуем увеличить время обучения сети до 500 эпох. Чтобы получить информацию о качестве обучения модели в каждую эпоху, изменим цикл обучения и добавим сохранение оценки проверки перед началом эпохи.

```
num_epochs = 500
all_mae_histories = []
for i in range(k):
    print('processing fold #', i)
```

. . . (как в предыдущем случае)

```
    #  Обучение модели (в режиме «без сообщений» verbose=0)
    history = model.fit(partial_train_data, partial_train_targets,
          validation_data=(val_data, val_targets),
          epochs=num_epochs, batch_size=1, verbose=0)
    mae_history = history.history['val_mae']
    all_mae_histories.append(mae_history)
          [train_targets[:i * num_val_samples],
           train_targets[(i + 1) * num_val_samples:]],axis=0)
    # # Конструирование модели Keras (уже скомпилированной)
    model = build model()
```

Теперь можно вычислить средние значения метрики *mae* для всех прогонов. Создадим истории последовательных средних оценок проверки по k блокам:

```
average_mae_history = [
np.mean([x[i] for x in all_mae_histories])
                  for i in range(num_epochs)]
```

и сформируем график с оценками проверок:

```
plt.plot(range(1, len(average_mae_history) + 1), average_mae_history)
plt.xlabel('Эпохи')
plt.ylabel('Оценки MAE')
plt.show()
```

Результат показан на рис. 2.

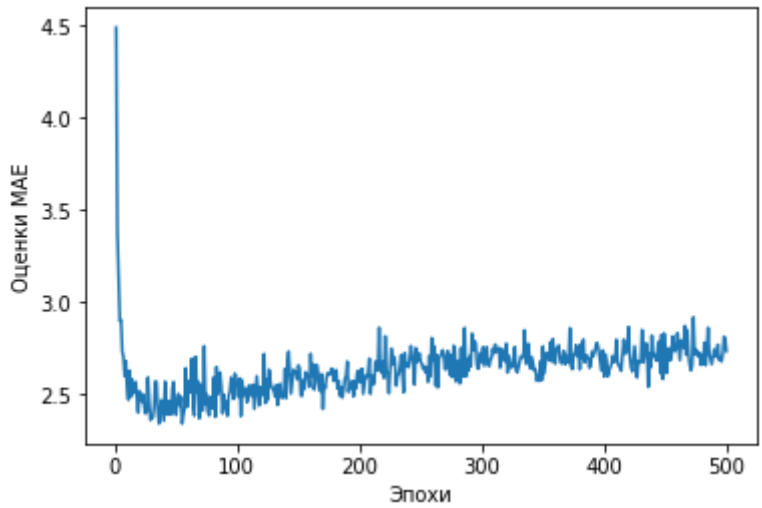


Рис. 2. Оценки MAE по эпохам

Из-за проблем с масштабированием, а также ввиду относительно высокой дисперсии затруднительно увидеть общую тенденцию. Внесем изменения в построение графика:

– опустим первые 10 замеров, которые имеют другой масштаб,

отличный от масштаба остальной кривой;

– заменим каждую оценку экспоненциальным скользящим средним по предыдущим оценкам, чтобы получить более гладкую кривую.

```
def smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            previous = smoothed_points[-1]
            smoothed_points.append(previous * factor + point *
                (1 - factor))
        else:
            smoothed_points.append(point)
    return smoothed_points
smooth_mae_history = smooth_curve(average_mae_history[10:])
plt.plot(range(1, len(smooth_mae_history) + 1), smooth_mae_history)
plt.xlabel('Эпохи')
plt.ylabel('Оценки MAE')
plt.show()
```

Результат показан на рис. 3.

Согласно этому графику, наилучшая оценка MAE достигается после 80 эпох. После этого момента начинается переобучение.

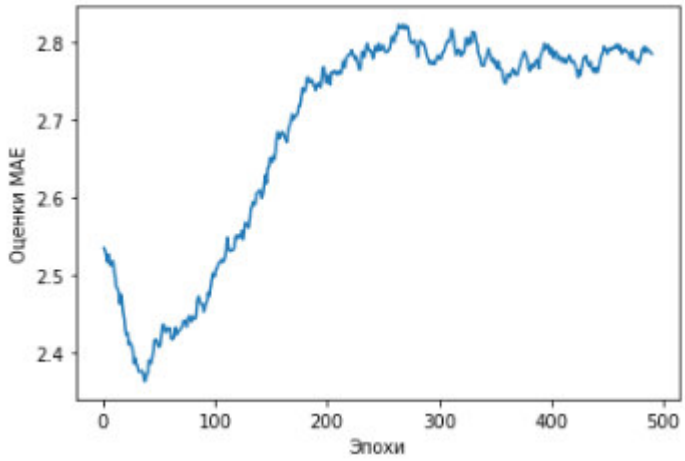


Рис. 3. Оценки MAE по эпохам за исключением первых 10 замеров

По окончании настройки других параметров модели (кроме количества эпох можно также скорректировать количество промежуточных слоев) можно обучить окончательную версию модели на всех обучающих данных, а затем оценить ее качество на контрольных данных.

```
model = build_model() # Получим новую скомпилированную модель
#  Обучение модели на всем объеме обучающих данных
model.fit(train_data_norm, train_targets,
     epochs=80, batch_size=16, verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data_norm, test_targets)
```

```
4/4 [==============================] - 0s 1ms/step - loss: 17.8266 - mae: 2.5956
```

Вот окончательный результат:

```
test_mae_score
2.595592737197876
```

Средняя ошибка все еще составляет около 2600 долларов.

Подведение итогов по задачи регрессии

Сделаем выводы по полученным результатам:

Регрессия выполняется с применением иных функций потерь, нежели классификация. Для регрессии часто используется функция потерь, вычисляющая среднеквадратичную ошибку mse (Mean Squared Error).

Также для регрессии используются иные метрики оценки; понятие точности неприменимо для регрессии, поэтому для оценки качества часто применяется средняя абсолютная ошибка mae (Mean Absolute Error).

Когда признаки образцов на входе имеют значения из разных диапазонов, их необходимо предварительно масштабировать.

При небольшом объеме входных данных надежно оценить качество модели поможет метод перекрестной проверки по K блокам.

При небольшом объеме обучающих данных предпочтительнее использовать маленькие сети с небольшим количеством промежуточных слоев (обычно с одним или двумя), чтобы избежать серьезного переобучения.

Общие выводы по использованию моделей классификации и регрессии:

– Исходные данные обычно приходится подвергать предварительной обработке перед передачей в нейронную сеть.

– Когда данные включают в себя признаки со значениями из разных диапазонов, их необходимо предварительно масштабировать.

– В процессе обучения нейронных сетей в некоторый момент наступает эффект переобучения, из-за чего падает качество результатов оценки сети на данных, которые она прежде не видела.

– При небольшом объеме входных данных следует использовать небольшие сети с одним или двумя промежуточными слоями, чтобы избежать серьезного переобучения.

– В том случае, когда данные делятся на большое число категорий, может возникнуть узкое место для информации, если слишком сильно ограничить размерность промежуточных слоев.

– При регрессии применяются иные функции потерь и метрики, нежели при классификации.

- При небольшом объеме входных данных надежно оценить качество модели поможет метод перекрестной проверки по K блокам.

Задание.

1. Постройте модель заданной архитектуры, обучите и проверьте ее точность. Сохраните модель. Восстановите модель и сравните точность обеих моделей.

Измените параметры архитектуры модели сети и выберите оптимальные значения.

Оцените качество модели методом перекрестной проверки по K блокам.

Используйте метод ранней остановки для избежания переобучения.

2. Создайте и обучите модель регрессии на наборе данных расхода топлива автомобилями в городском цикле в милях на галлон [5]

Список источников

1. Шолле Ф. Глубокое обучение на Python.- СПб.: Питер, 2019
2. Библиотека keras.- https://keras.io/getting_started/
3. Библиотека tensorflow.- https://www.tensorflow.org/resources/learn-ml
4. Набор данных цен на жилье в Бостоне.- https://keras.io/api/datasets/boston_housing/
5. Набор данных авто MPG.- https://archive.ics.uci.edu/ml/datasets/auto+mpg

Порядок выполнения работы

1. Изучить теоретический материал
2. Ввести и выполнить программный код из описания практического занятия
3. Выполнить эксперимент с моделью в соответствии с заданием
4. Выполнить сравнительный анализ полученных результатов
5. Создать и обучить модель регрессии на наборе данных расхода топлива автомобилями в городском цикле [5] по методике практического занятия

Содержание отчета

1. Программный код (блокнот) решения задачи регрессии
2. Результаты эксперимента с моделью и выводы сравнительного анализа
3. Программный код (блокнот) решения задачи регрессии для прогноза расхода топлива автомобилями в городском цикле