Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Макушев Андрей Евгеньевич
Должность: Ректор
Дата подписания: 04.10.2022 14:15:52
Уникальный программный ключ:
4c46f2d9ddda5fab9e57e85d11e5a4257b6ddfe

На 1-м практическом занятии необходимо познакомить обучающихся

- с возможностями языка Python (см. Введение в Python (руководство по быстрому старту).pdf)

- с работой в Google Colab, которая не требует установки Python и библиотек (см. Методические указания по работе с Google Colab.pdf).

В рамках занятия обучающиеся должны узнать и научиться:

- Как зайти в Google Colab (https://colab.research.google.com/);
- Как загрузить готовый блокнот в Google Colab;
- Как создать новый блокнот, оформлять текст, вставлять картинки, вставлять и запускать код;
- как скачать блокнот на локальный диск и как сохранить блокнот на своем гугл-диске;
- как организовать работу с блокнотами на гугл-диске.

Требования:
- доступ в интернет с рабочего места обучающегося (можно с планшета);
- должна быть у каждого гугловская учетка (создать почту на gmail.com).

СРС:
на онлайн-курсе "**Введение в анализ данных**" пройти 1-ю часть темы "Среда решения задач анализа данных. Основы программирования на Python"

На 2-м практическом занятии необходимо познакомить обучающихся

- с основами работы в Python (см. Материалы для практики),

Лучше всего, если обучающиеся вместе с объяснением вами материала будут набирать код, имеющийся в материале, и запускать его.

В рамках занятия обучающиеся должны разобраться с темами:

- базовые типы данных ;
- работа с переменными;
- преобразование типов данных - явное и неявное;
- основные операции с числовыми и строковыми данными;
- как оформляются блоки кода с помощью отступов;
- как описываются основные управляющие структуры.

Требования:
- доступ в интернет с рабочего места обучающегося (можно с планшета);
- должен уметь работать с блокнотами в среде Google Colab

СРС:
на онлайн-курсе "Введение в анализ данных" пройти 1-ю часть темы "Среда решения задач анализа данных. Основы программирования на Python".
Выполнить практические задания в среде VPL:
- Встроенные операции с числами (VPL)
- Встроенные операции со строками (VPL)
- Управляющие структуры в Python (VPL)

На этом практическом занятии необходимо познакомить обучающихся

с основами работы с коллекциями и функциями в Python (см. Материалы для практики и СРС),

Лучше всего, если обучающиеся вместе с объяснением вами материала будут набирать код, имеющийся в материале, и запускать его.

В рамках занятия обучающиеся должны разобраться с темами:

- Упорядоченные коллекции;
- работа с множествами;
- работа со словарями;
- как создаются и вызываются функции;
- Работа с модулями и пакетами в Python;
- Знакомство с пакетом sklearn.

Требования:
- доступ в интернет с рабочего места обучающегося (можно с планшета);
- должен уметь работать с блокнотами в среде Google Colab
- понимать базовые типы данных и как работают управляющие структуры

СРС:
на онлайн-курсе "Введение в анализ данных" пройти тему "Основные структуры данных, массивы. Функции".
Выполнить практические задания в среде VPL:
- Работа со списками (VPL)

Тема 1. ВВЕДЕНИЕ в Python.

План уроков темы

- ❑ История Python. Секрет популярности. Особенности программирования.
- ❑ Установка языка. Дзен Python-а. Создание и настройка виртуальных окружений.
- ❑ Установка Jupyter. Работа в Jupyter notebook локально.
- ❑ Работа в облаке с Google Colab. Работа с Google Disk

Python (Пайтон)

Python – это интерпретируемый, объектно-ориентированный язык программирования высокого уровня с динамической типизацией, автоматическим управлением памятью и удобными высокоуровневыми структурами данных, такими как словари (хэш-таблицы), списки, кортежи.

https://it-black.ru/istoriya-yazyka-python/

История развития Python

Начало разработки – 1989 г

Первый релиз Python - 1991 г.

Python 1.0 появился в январе 1994 года.

Python 2.0 появился в 2000 году.

Самая свободная лицензия - никаких ограничений в использовании языка

Гуидо ван Россум (Guido van Rossum) – автор Python и «великодушный пожизненный диктатор» проекта

https://ru.wikipedia.org/wiki/

Версия 3.0, поддержка на платформах

Python 3.0 - 3 декабря 2008 года

Версии **2.x** и **3.x** не полностью совместимы!

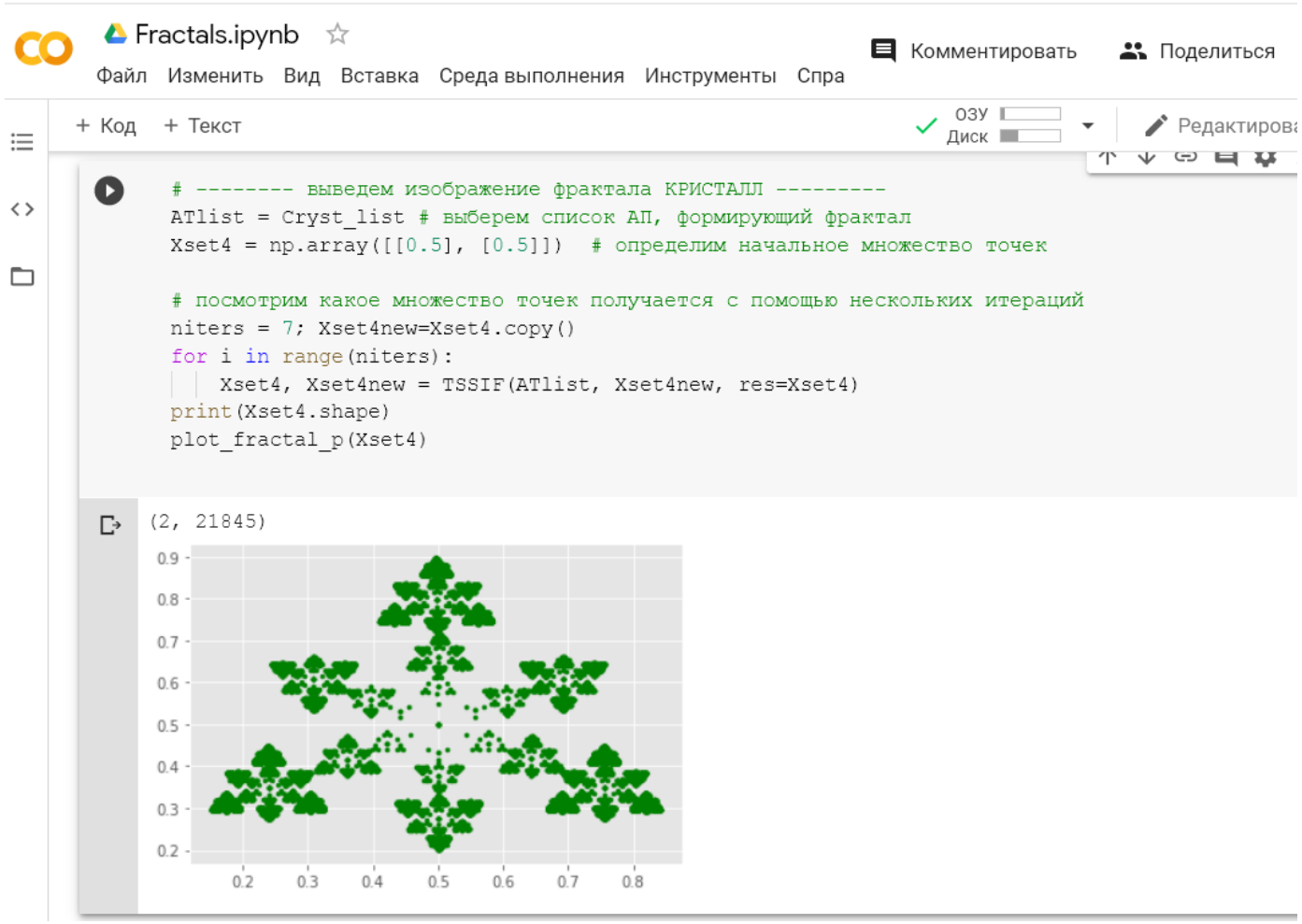
Интерпретатор языка поддерживается на всех платформах

Python – скрипты (***.py**), запуск программ в стандартном и интерактивном режимах

Jupyter notebook

Jupyter Notebook, блокноты (***.ipynb**) – удобный формат представления отчетов

Файл **.ipynb** предста вляет собой текстовый файл в формате JSON.



Популярность Python

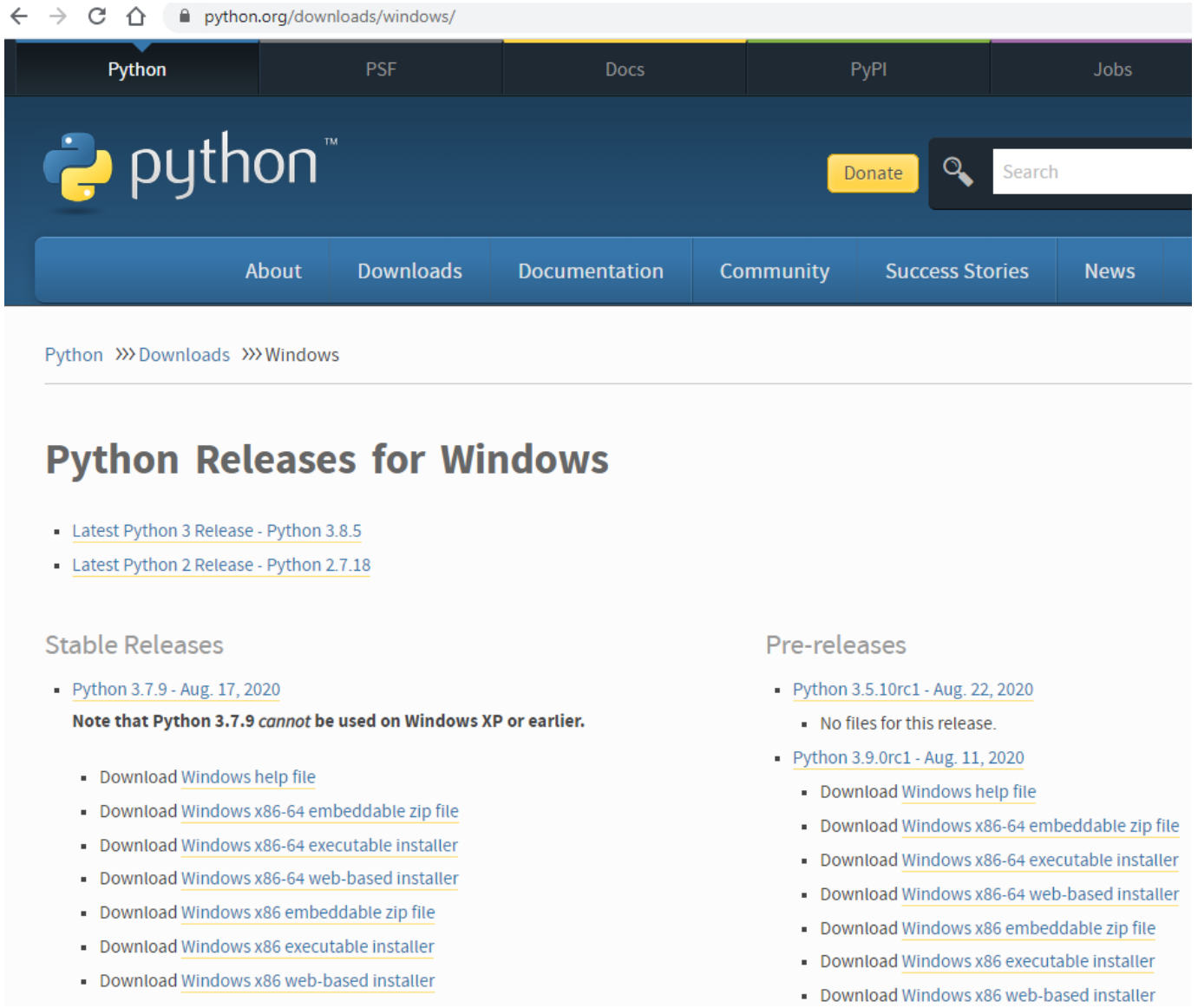
- Компания **Google** использует Python в своей поисковой системе.
- **Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm и IBM,** используют Python для тестирования аппаратного обеспечения.
- Служба коллективного использования видеоматериалов YouTube в значительной степени реализована на Python.
- Популярная программа **BitTorrent** для обмена файлами в пиринговых сетях написана на языке Python.
- Популярный веб-фреймворк **App Engine** от компании **Google** использует Python в качестве прикладного языка программирования.
- **NASA, Los Alamos, JPL и Fermilab** используют Python для научных вычислений.

План уроков темы

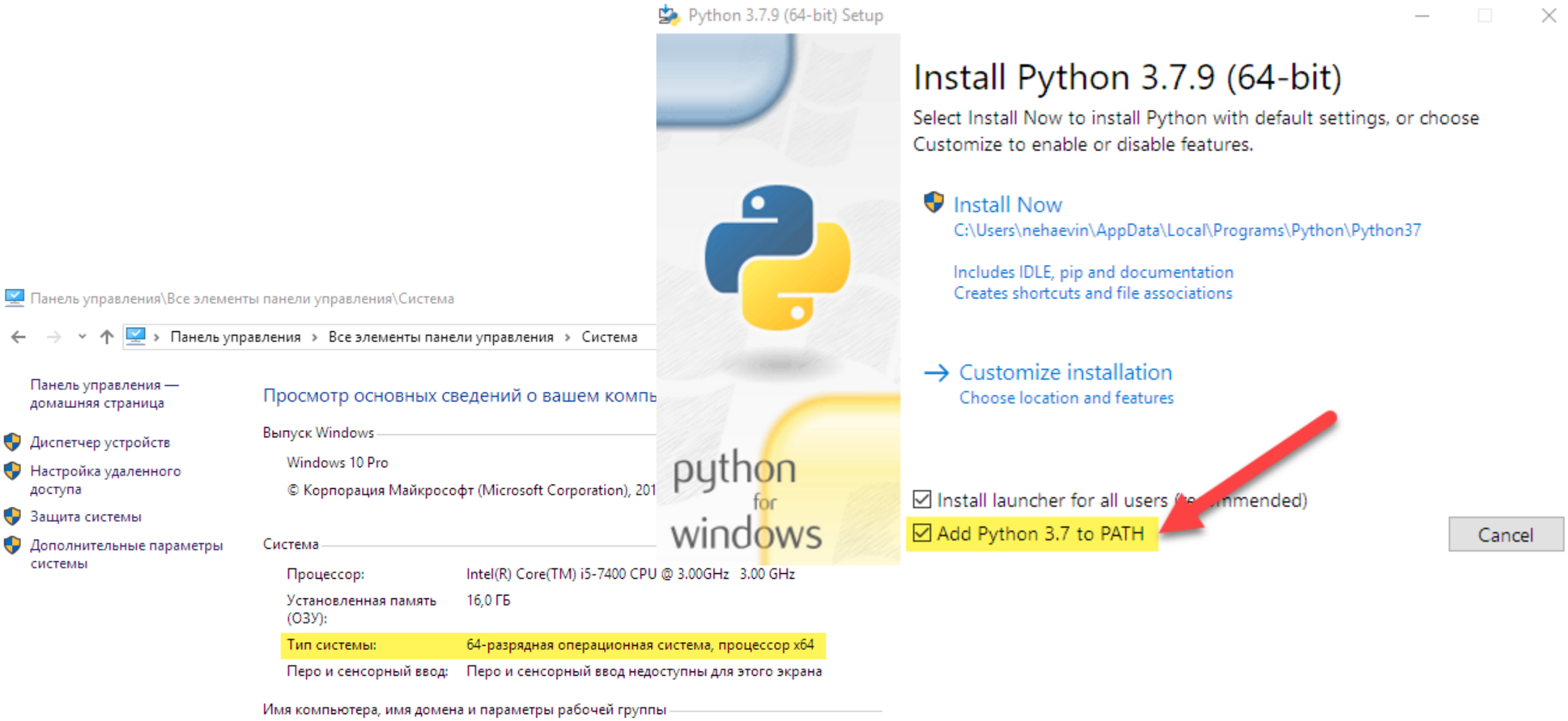
- ❑ История Python. Секрет популярности. Особенности программирования.
- ❑ **Установка языка. Дзен Python-а. Создание и настройка виртуальных окружений.**
- ❑ Установка Jupyter. Работа в Jupyter notebook локально.
- ❑ Работа в облаке с Google Colab. Работа с Google Disk

Установка языка Python под Windows

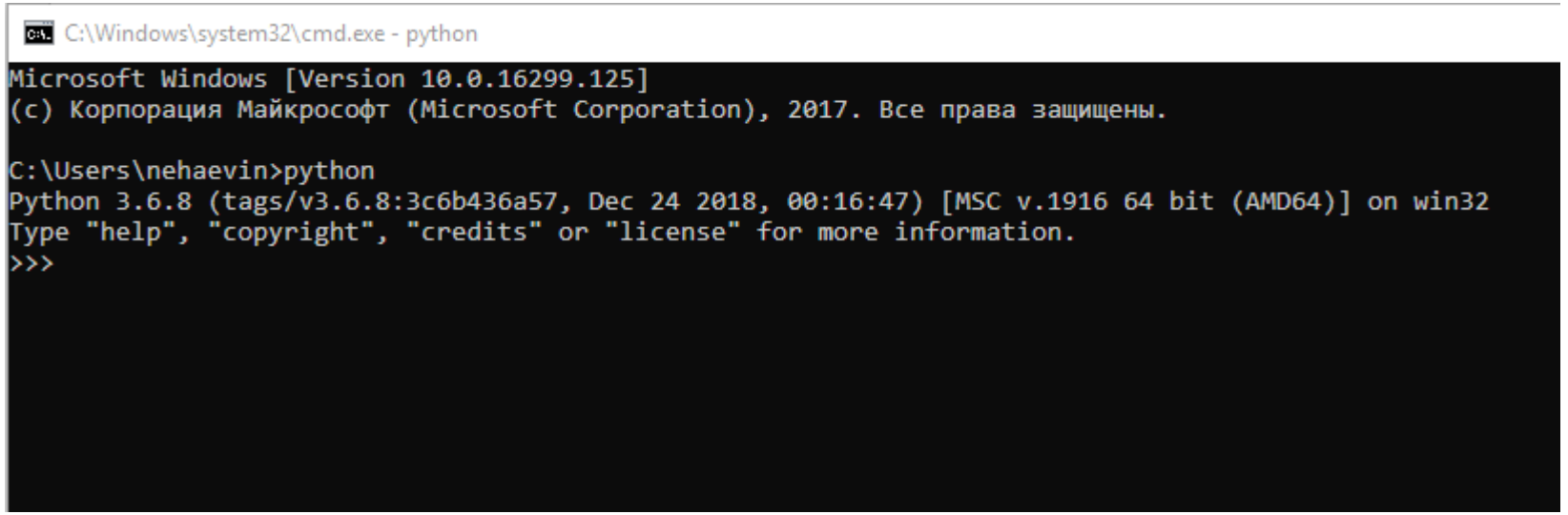
- Переходим на сайт https://www.python.org/
- Выбираем пункт меню “**downloads**”
- Выбираем установку “*…for Windows*”
- Выбираем стабильную версию под свою конфигурацию



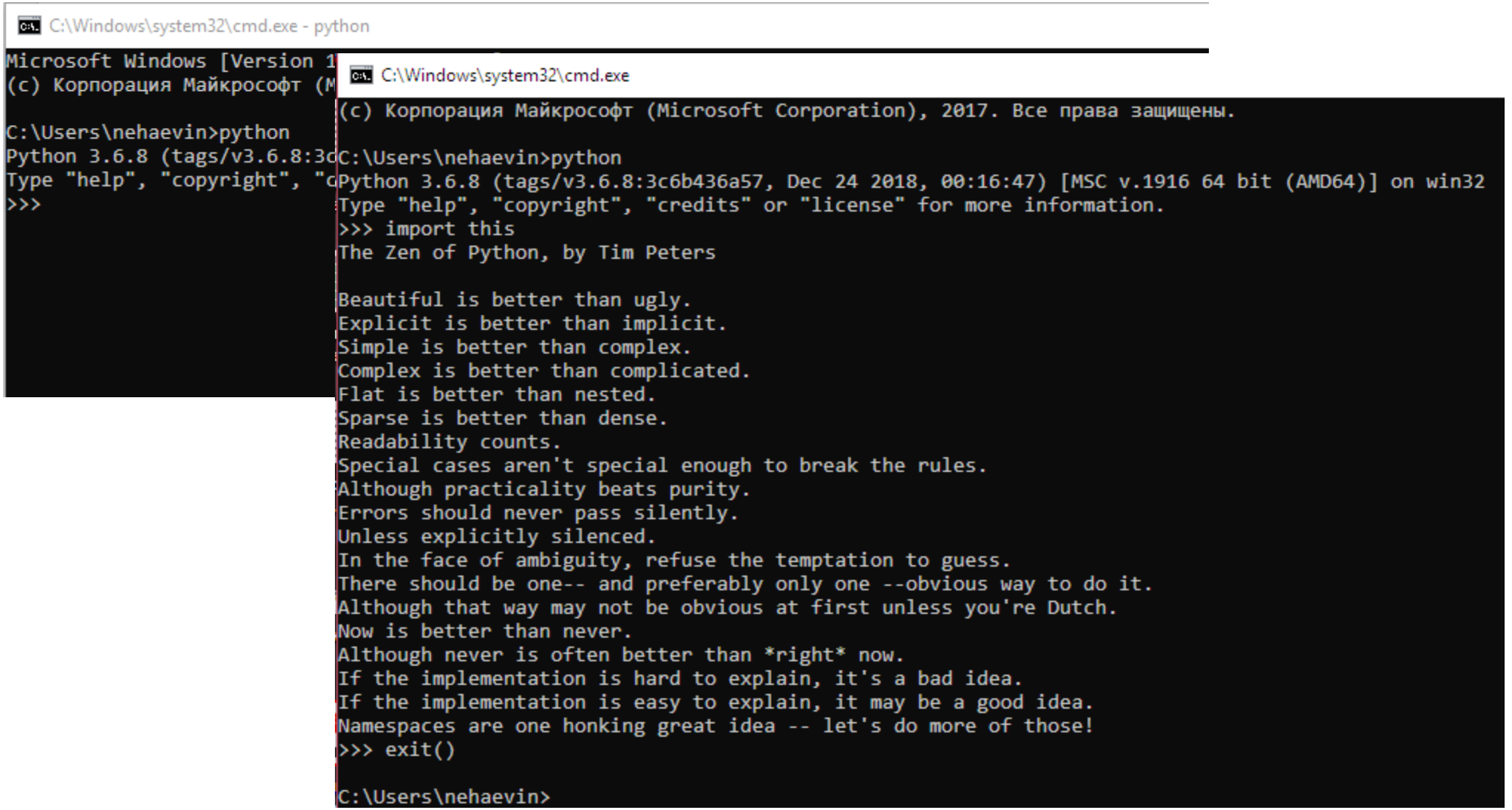
Выбор версии и разрядности



Запуск и проверка установки Python



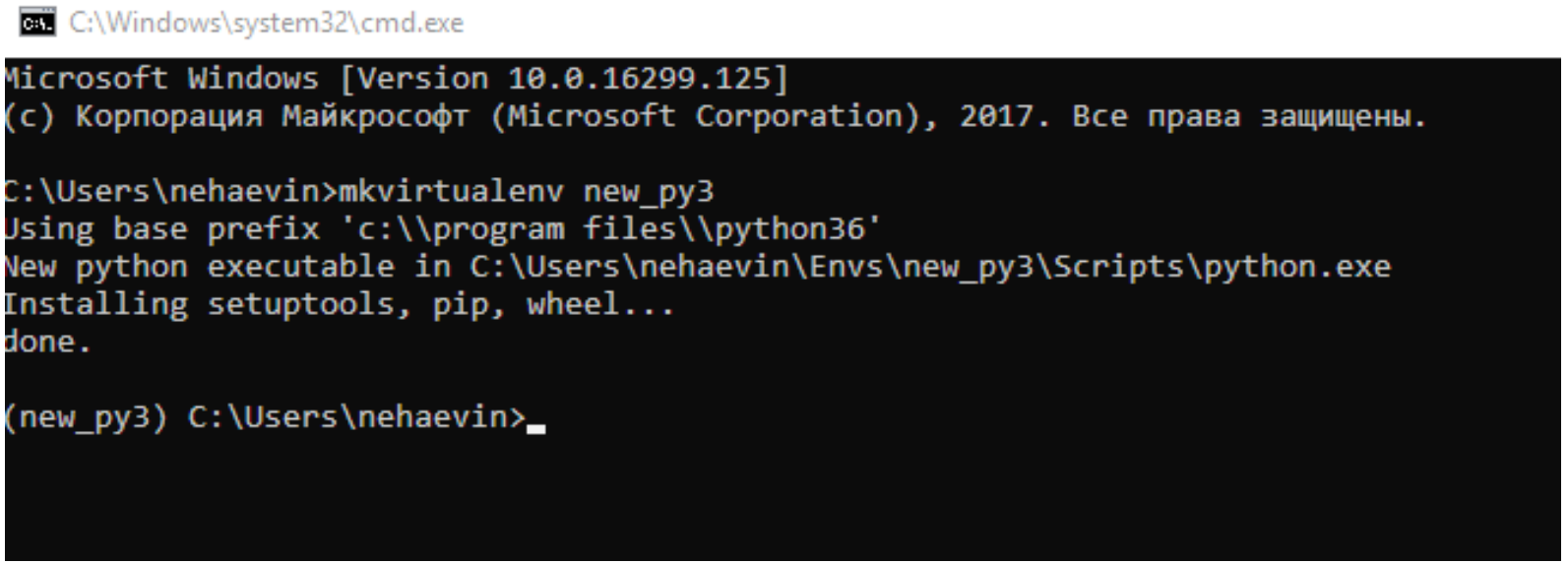
Запуск и проверка установки Python



Установка и работа с вирт-ми окружениями

Создание виртуального окружения:

- …> **mkvirtualenv myenv** (создать новое окружение с именем «myenv», например. “py3”)



Установка виртуального окружения

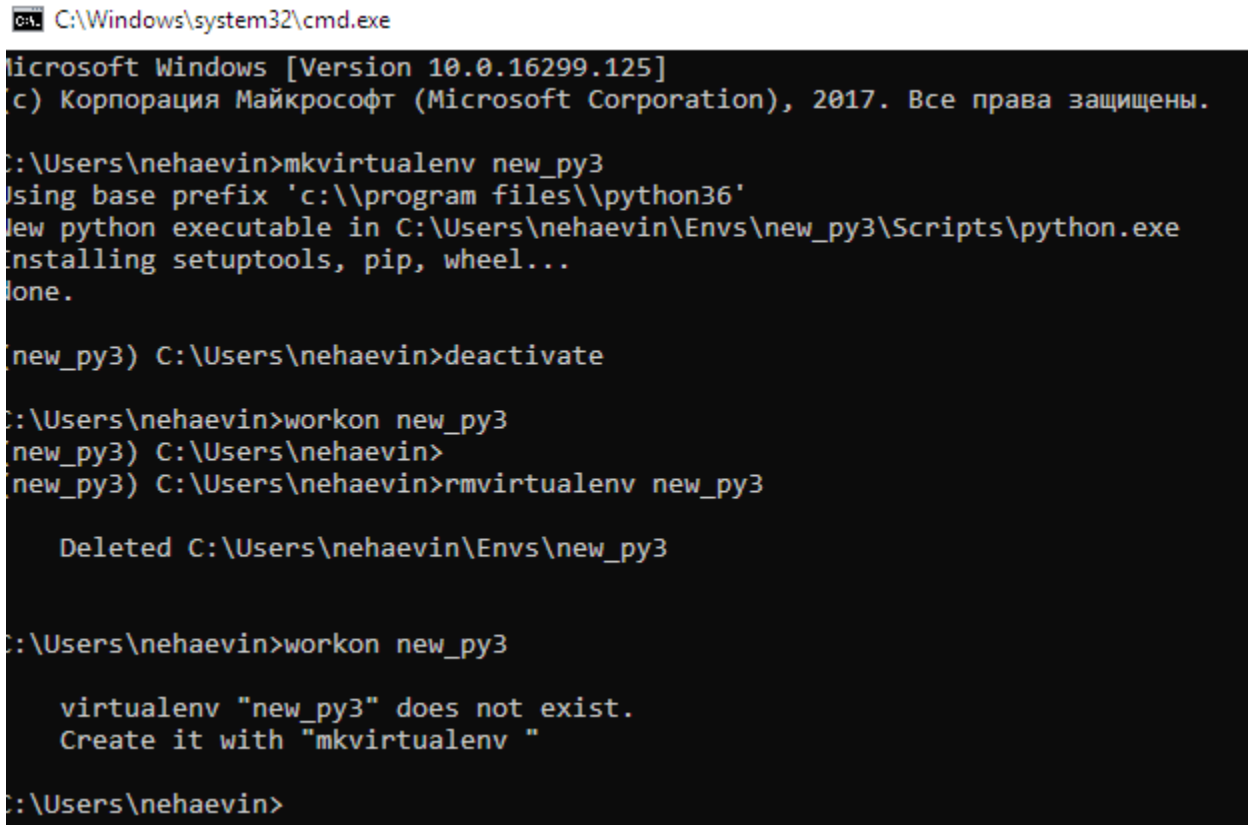
Установка пакетов для работы с виртуальными окружениями:

- …> **pip install virtualenv**
- …> **pip install virtualenvwrapper-win**

Работа с виртуальными окружениями

Работа с виртуальными окружениями:

- …> **rmvirtualenv myenv** (удалить окружение с именем «myenv»)
- …> **workon myenv** (войти в окружение с именем «myenv»)
- …> **deactivate** (выйти из текущего виртуального окружения)



Установка необходимых библиотек

- Войдите в созданное окружение (например с именем “py3”):

C:\Users\nehaevin>workon py3

- С помощью команды «pip install <имя_пакета>» установите необходимые пакеты, например пакет работы с массивами **numpy**:

(py3) C:\Users\nehaevin>pip install numpy

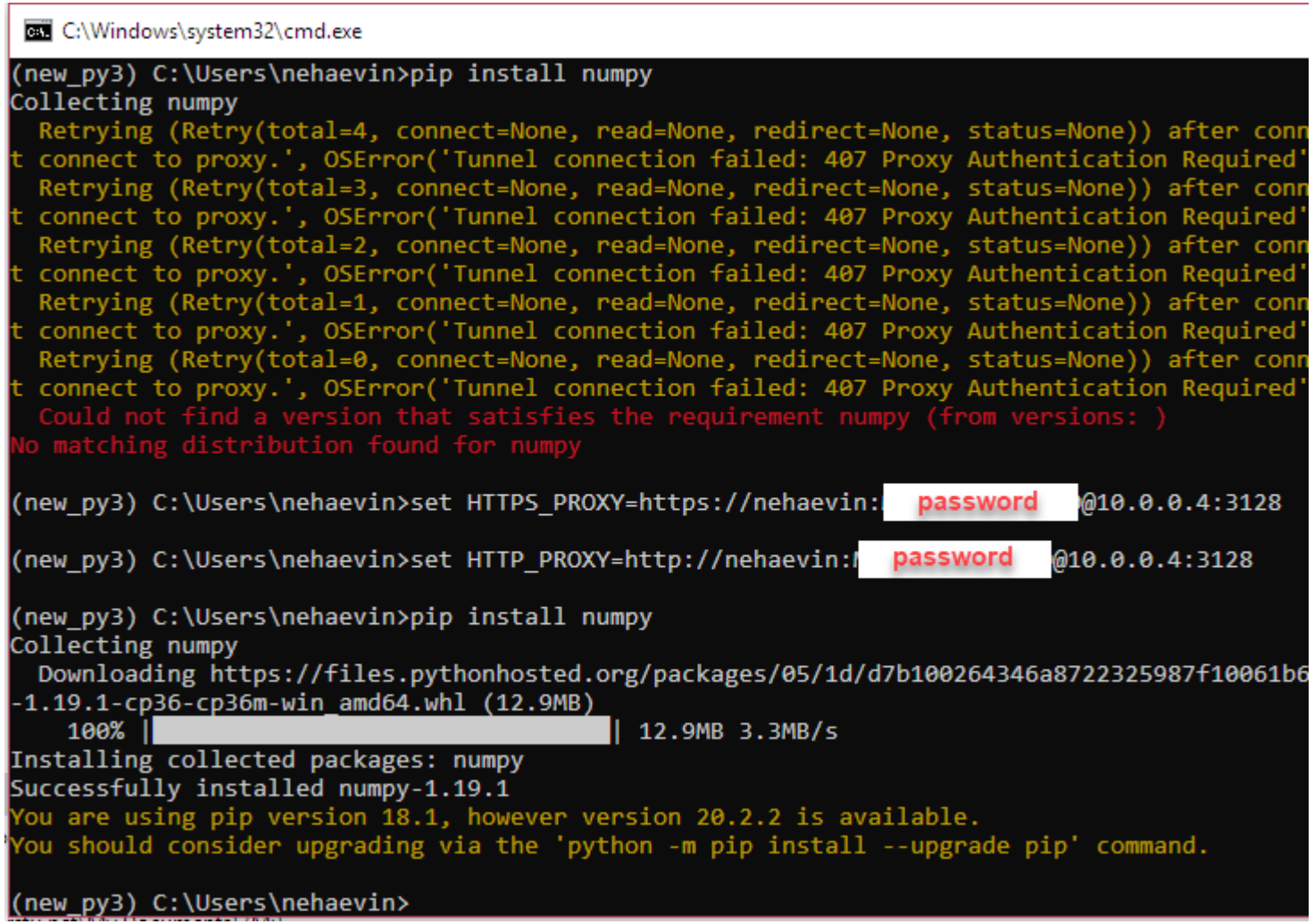
- Теперь вы можете с легкостью установить нужную Вам библиотеку в соответствующее окружение или создать новое окружение для нового проекта.

Важные замечания

- Можно устанавливать библиотеки, не устанавливая и не переходя в окружение. Но при этом они по умолчанию будут доступны в любом окружении и вы не сможете в будущем поменять с легкостью окружение под новые задачи и новые библиотеки

Важные замечания

- Если вы устанавливается питон и библиотеки в корпоративной сети с прокси-сервером, то вам необходимо перед установкой указать адрес прокси и логин-пароль соответствующими командами:



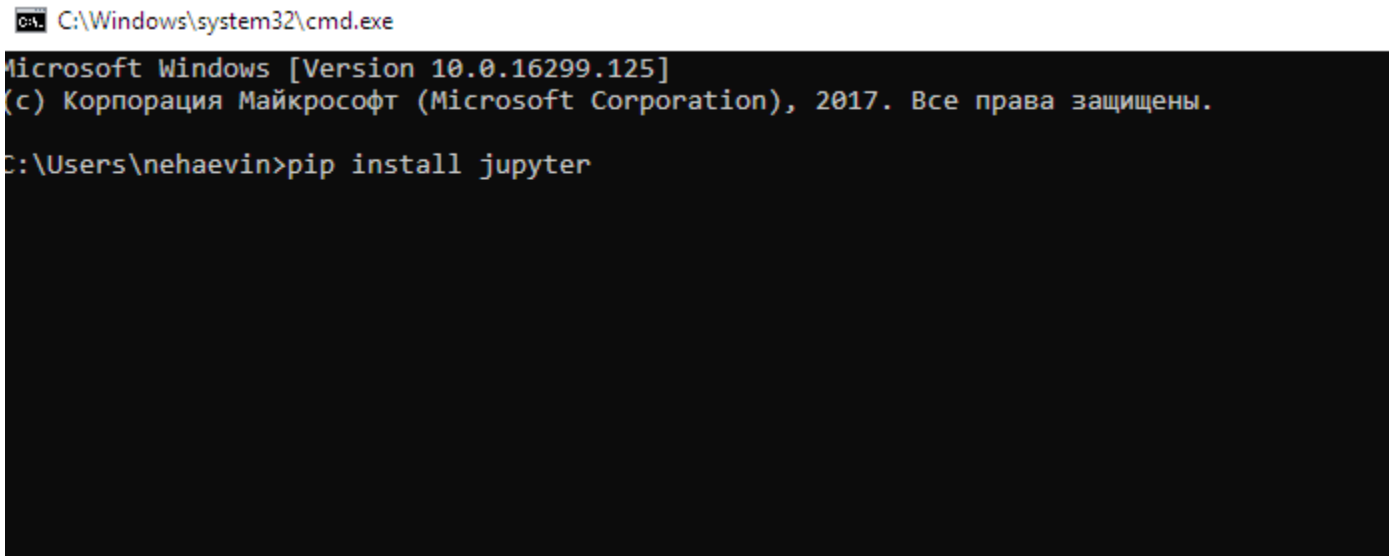
План уроков темы

- ❑ История Python. Секрет популярности. Особенности программирования.
- ❑ Установка языка. Дзен Python-а. Создание и настройка виртуальных окружений.
- ❑ **Установка Jupyter. Работа в Jupyter notebook локально.**
- ❑ Работа в облаке с Google Colab. Работа с Google Disk

Установка Jupyter

1. Откроем командную строку
Если вы не находитесь в командной строке, то войдите в окно программы "Выполнить" (например, с помощью сочетания клавиш **Win+R**). В появившемся диалоговом окне программы "Выполнить" наберите слово cmd и нажмите Enter. Запустится терминал.

2. Установите пакет Jupyter:
Его необходимо установить в окружение по умолчанию, т.е. вам необходимо выйти из виртуального окружения, если вы в нем находитесь.
Наберите команду установки:
> pip install jupyter

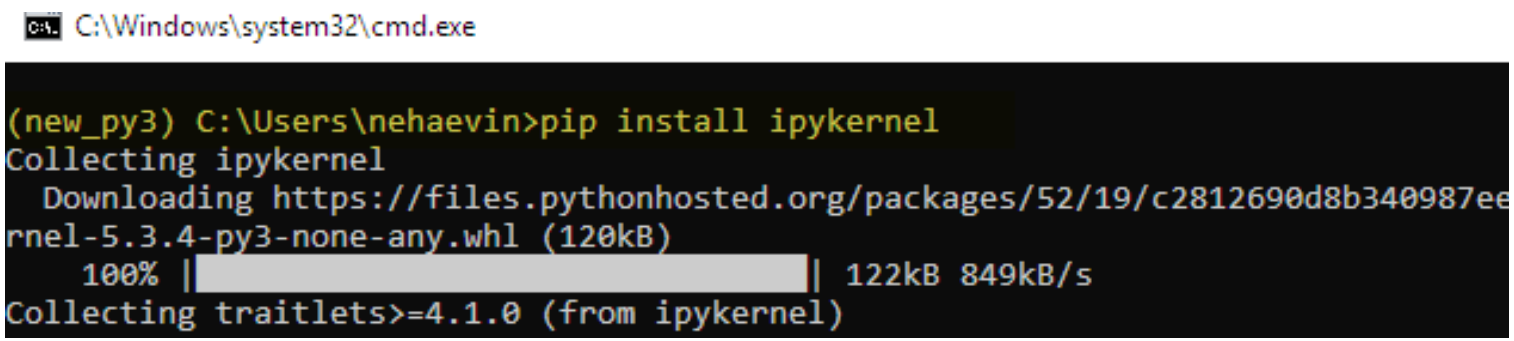


Установка Jupyter

3. Войдите в виртуальное окружение. Введите команды:

C:\Users\user>workon py3

Установите пакет ядра для работы с jupyter notebook:
(py3) C:\Users\user> pip install ipykernel



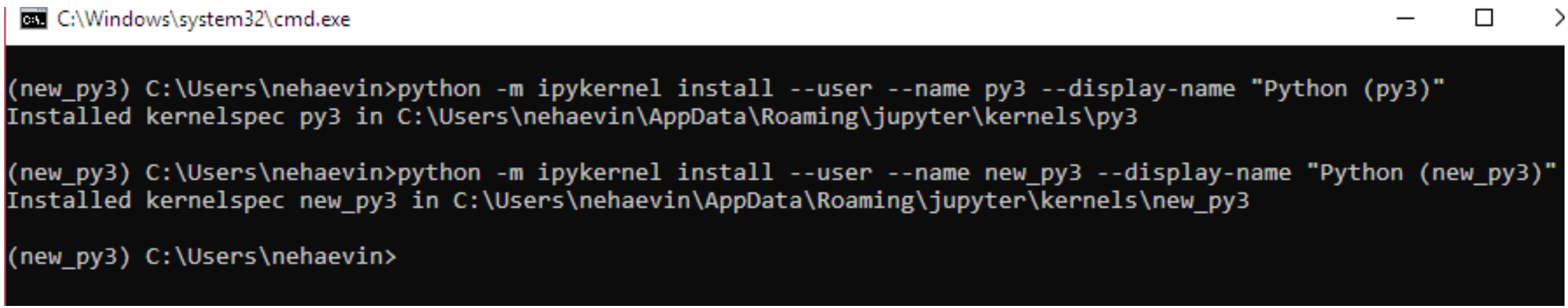
Установка Jupyter

3. Войдите в виртуальное окружение. Введите команды:

C:\Users\user>workon py3

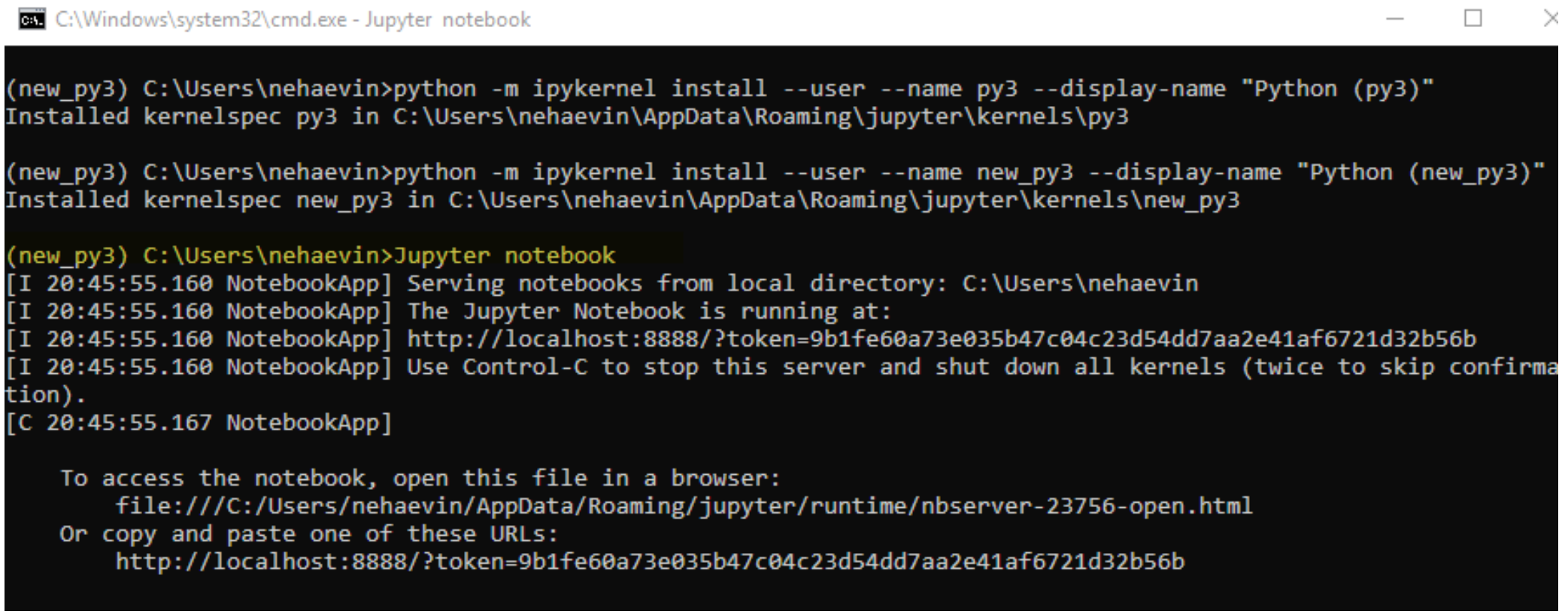
Установите пакет ядра для работы с jupyter notebook:
(py3) C:\Users\user> pip install ipykernel

Связываем имя окружения (“py3”) с питоном и пользователем:
(py3) C:\Users\user> python -m ipykernel install --user --name py3 --display-name "Python (py3)"



Запуск Jupyter

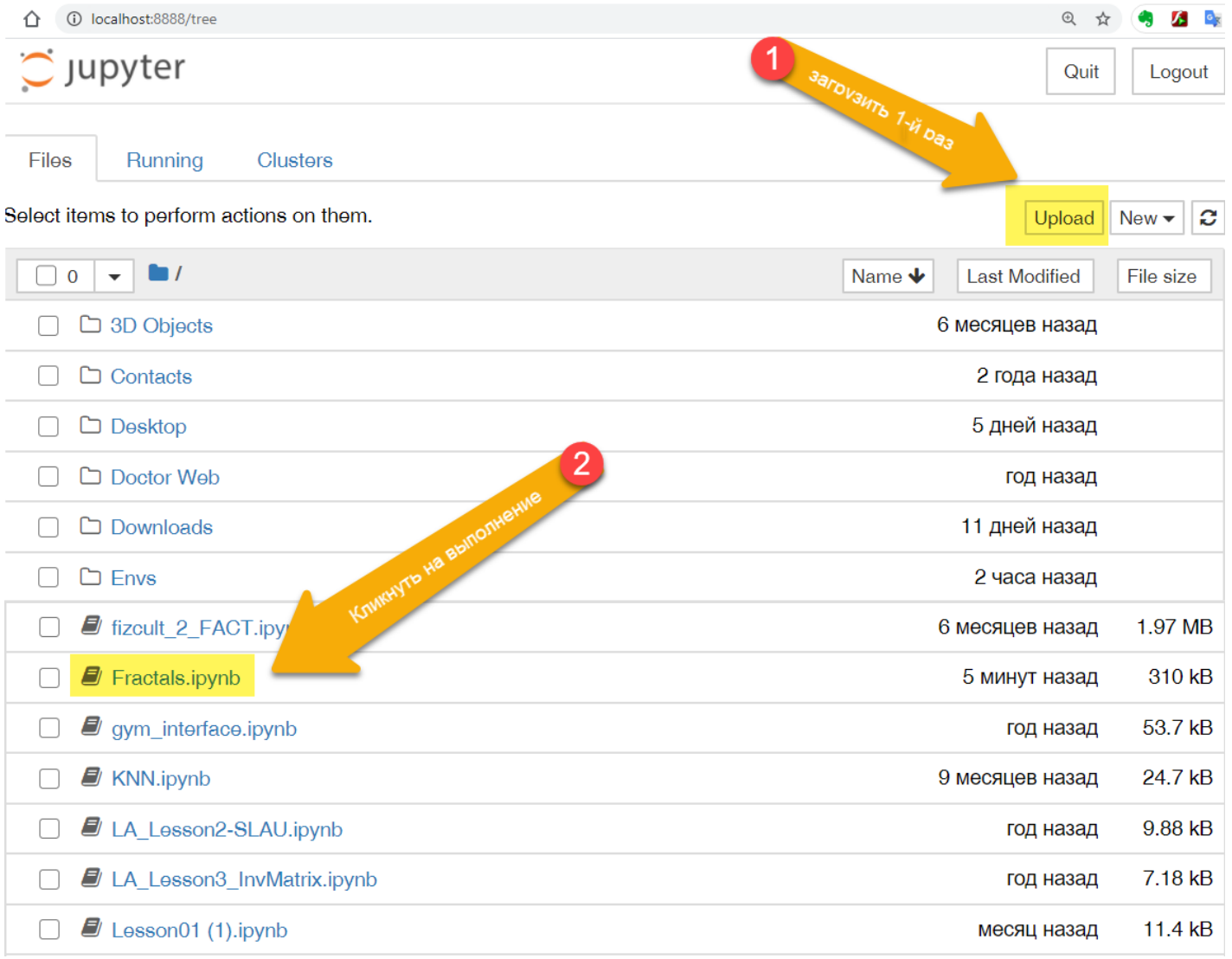
Запуск осуществляется командой:
(new_py3) C:\Users\nehaevin>**Jupyter notebook**



Web - оболочка Jupyter

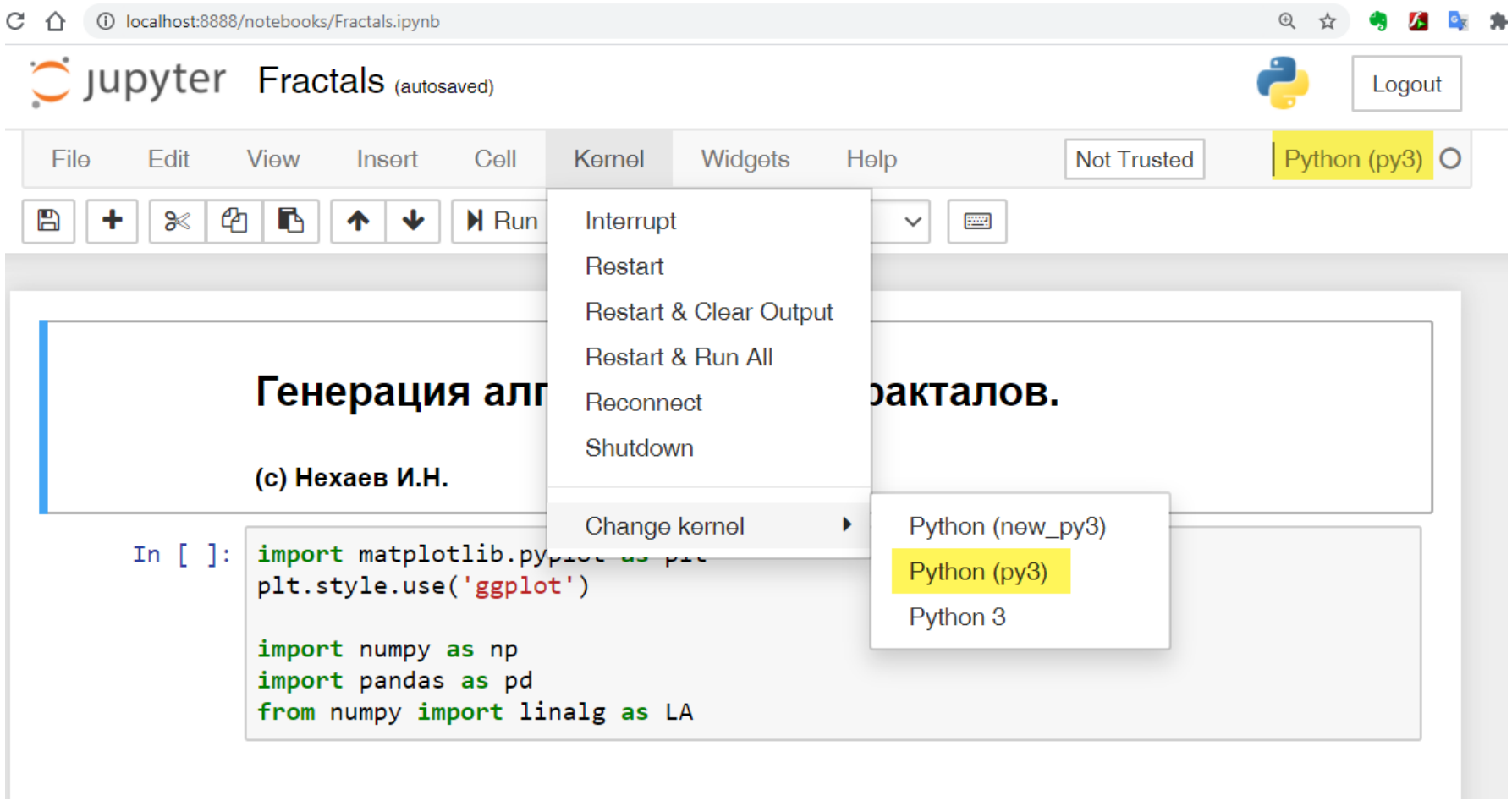
После загрузки сервиса, в браузере по умолчанию появится окно Jupyter

Теперь мы можем загрузить и начать работать с блокнотами



Работа с блокнотом

Перед тем как начать работать с блокнотом необходимо указать нужное окружение



План уроков темы

- ❑ История Python. Секрет популярности. Особенности программирования.
- ❑ Установка языка. Дзен Python-а. Создание и настройка виртуальных окружений.
- ❑ Установка Jupyter. Работа в Jupyter notebook локально.
- ❑ **Работа в облаке с Google Colab. Работа с Google Disk**

Python - – это интерпретируемый, объектно-ориентированный язык программирования высокого уровня с динамической типизацией, автоматическим управлением памятью и удобными высокоуровневыми структурами данных, такими как словари (хэш-таблицы), списки, кортежи

https://it-black.ru/istoriya-yazyka-python/

https://docs.python.org/3/

ДОПУСК к ЭКЗАМЕНУ

Экзамен рекомендуется проводить аудиторно в два этапа:

На 1-м этапе студент выполняет итоговый тест и не может пользоваться ничем, кроме ручки и тетради. На тест из 15 заданий дается 75 минут.

На 2-м этапе студент, успешно прошедший итоговый тест (не менее 50% баллов), допускается к решению практической задачи с использованием программной системы VPL. Преподаватель сам может определить сложность задачи и выбрать задачу из пула задач.

Успешно сдавшие тест и выполнившие практическое задание считаются успешно сдавшими экзамен.

К сдаче итогового теста раздела допускаются те обучающиеся, которые

1) выполнили все тесты раздела, каждый не менее, чем на 67% от максимального балла за тест.

2) выполнили все практические задания раздела (перечень необходимых к выполнению заданий определяется преподавателем);

У вас 3 попытки. Засчитывается последняя. Тестирование организуется преподавателем курса с использованием оффлайн или онлайн прокторинга.

Надо набрать не менее 50% баллов за тест, чтобы он считался успешно пройденным.

Экзамен также может быть заменен на защиту выполненного студентом практико-ориентированного задания, кейса курса.

Успешной работы!

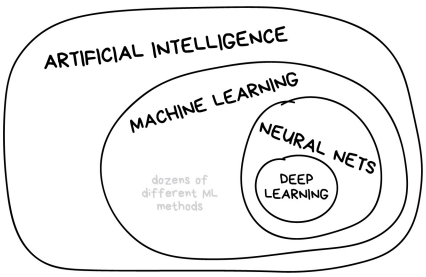
Поволжский государственный технологический университет

Нехаев Игорь Николаевич

Конспект

РАЗДЕЛ 1. ВВЕДЕНИЕ в МАШИННОЕ ОБУЧЕНИЕ

Лекция 1. Машинное обучение и искусственный интеллект.
Основные понятия. Иерархия задач машинного обучения



Аннотация: В лекции рассматриваются определение машинного обучения, принципы создания искусственного интеллекта.
Рассматриваются основные понятия и пирамида задач машинного обучения

Ключевые слова: машинное обучение, искусственный интеллект, данные, иерархия задач машинного обучения, метрики качества, алгоритмы, предобработка, дата-инженер, дата-аналитик, дата-сайентист

Йошкар-Ола - 2022

Содержание

1 Что такое машинное обучение

Первую программу на основе алгоритмов, способных самообучаться, разработал Артур Самуэль (Arthur Samuel) в 1952 году, предназначена она была для игры в шашки. Самуэль дал и первое определение термину «машинное обучение»: это «область исследований разработки машин, не являющихся заранее запрограммированными». Более точное определение термину «обучение» дал намного позже Т. М. Митчелл (см. [1])

Определение 1.1 (Машинное обучение по Т. М. Митчеллу, (machine learning, ML))**.** *Будем считать, что компьютерная программа обучается на опыте (данных)* ***E*** *относительно некоторой задачи* ***T*** *и некоторого критерия качества работы* ***P****, если её работа по решению* ***T****, как измерено* ***P****, улучшается с опытом (увеличением данных)* ***E****.*

Определение 1.2 (Задача машинного обучения, ML Problem)**.** *Пусть имеется:*
- некоторая задача ***Task*** *(например, улучшить продажи какого-то интернет-магазина);*
- есть критерий качества работы ***Productivity*** *нашей системы машинного обучения, решающей поставленную задачу* ***Task*** *(например, количество заходов на продающий сайт, количество продаж, суммарная прибыль за месяц и.т.п.);*
- есть накапливаемый опыт (данные) ***Experience*** *(сколько всего было заходов на сайт, сколько раз пользователь кликал на страницу с товаром, сколько товаров откладывал в корзину, сколько оплатил, ...)*
Тогда будем считать задачей машинного обучения - создание такой компьютерной программы, которая использует накапливаемые данные ***Experience*** *для того, чтобы все лучше (с точки зрения выбранного критерия* ***Productivity****) решать некоторую практическую задачу* ***Task****.*

2 Искусственный интеллект

Существует много определений **ИИ** (искусственный интеллект). (англ. **AI** - Artifical Intelligence). Различают два типа определений - определение сильного ИИ и слабого ИИ.

Определение 2.1 (сильный ИИ)**.** *Искусственный интеллект — это направление в информатике и информационных технологиях, задачей которого является воссоздание с помощью вычислительных систем и иных искусственных устройств разумных рассуждений и действий.*

Таким образом под сильным ИИ понимается вычислительная система, которая способна рассуждать и решать задачи как человек.

Существует **тест Тьюринга** - эмпирический тест, идея которого была предложена Аланом Тьюрингом в статье «Вычислительные машины и разум», опубликованной в 1950 году в философском журнале Mind. Тьюринг задался целью определить, может ли машина мыслить.

Стандартная интерпретация этого теста звучит следующим образом: «Человек на основании ответов на вопросы должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор» (см. [2]).

Определение 2.2 (слабый ИИ)**.** *Искусственный интеллект — это способность системы правильно интерпретировать внешние данные, извлекать уроки из таких данных и использовать полученные знания для достижения конкретных целей и задач при помощи гибкой адаптации.*

Рис. 1: Связь искусственного интеллекта с машинным обучением

Способность **ИИ** самостоятельно получать знания в процессе работы из данных тесно связана с решением задач машинного обучения (см. рис. 1). Это направление в настоящее время является центральным в развитии **ИИ** (см. [1]).

В наиболее продвинутых алгоритмах ML используются нейронные сети (см. [3], [4]).

3 Иерархия задач машинного обучения

Иерархию задач, которые необходимо решать при решении задач машинного обучения, создания искусственного интеллекта хорошо отображает рис. 2 (см. [5]).

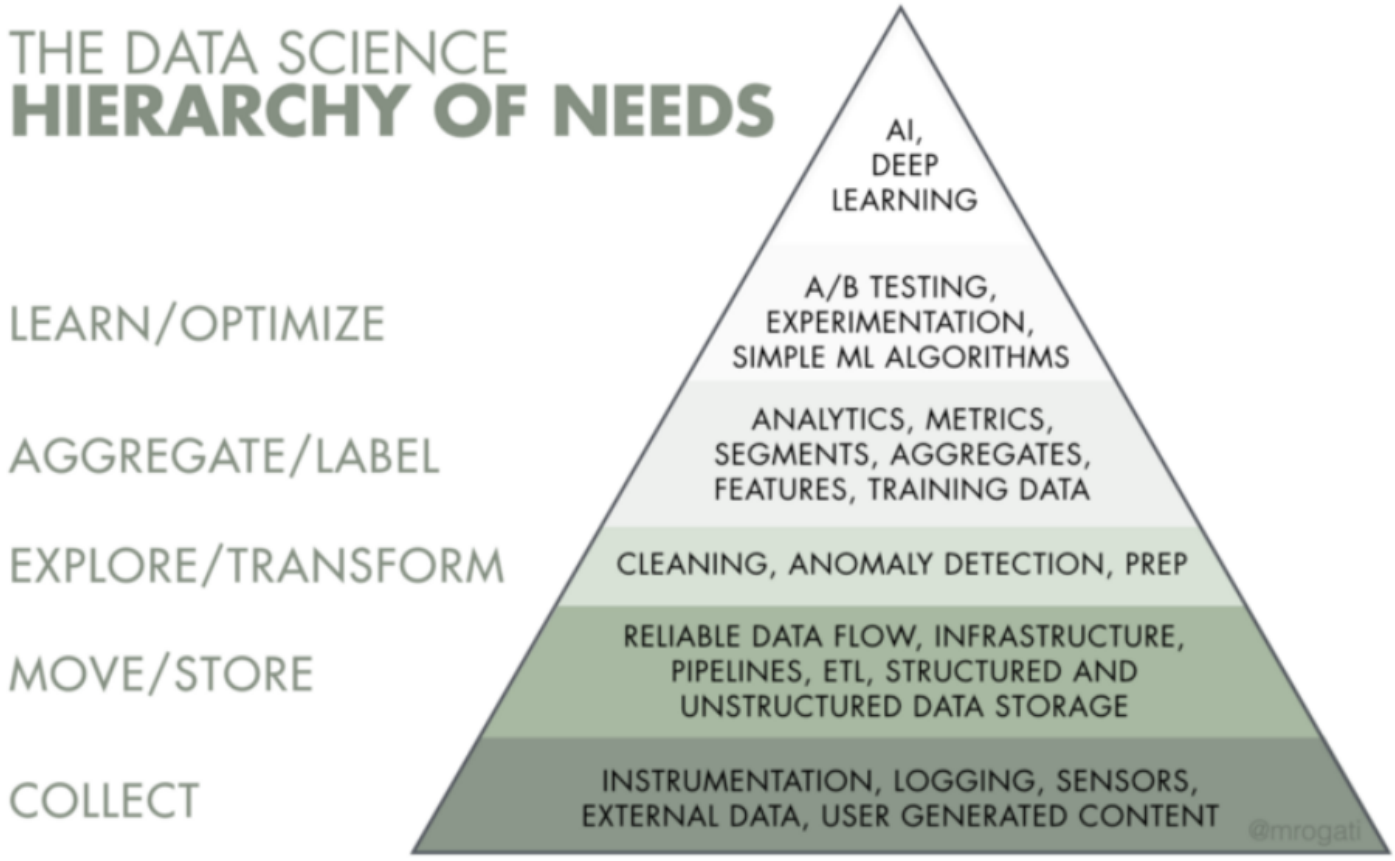


Рис. 2: Иерархия задач машинного обучения

На рис. 2 мы видим большое количество задач, которые надо решить, прежде чем удастся создать "умную"программу, решающую задачу машинного обучения или построения искусственного интеллекта. Эти задачи располагаются на самой вершине пирамиды задач.

Более того, ширина горизонтального сечения слоя пирамиды отражает тот факт, что трудоемкость задач внизу гораздо больше, чем трудоемкость задач вверху пирамиды. Что же это за задачи?

Рассмотрим задачи, начиная с онования пирамиды.

Задача 3.1 (Collect data: сбор данных)**.** В самом низу пирамиды находятся задачи по получению данных с раздичных сенсоров, логгирование процессов, сбор данных с различных внешних источников и в том числе из опросов экпертов и пользователей систем.

Например, в рамках проекта "Умный Дом"требуется поставить в доме и вокруг него различные сенсоры, датчики, вебкамеры, фиксировать звуки и команды людей, собирать отзывы о качестве работы системы.

Задача 3.2 (Move/Store: первоначальная очистка и накопление, хранение, структурирование)**.** После получения данных требуется эти данные где-то накапливать и хранить, предварительно проверяя их надежность и отсеивая неинформативные данные.

Это задача связана с использованием облачных сервисов, на которых накапливаются данные. При этом имеет смысл уже на этом этапе предобрабатывать сырые данные. Например, если весь видеопоток со всех камер накапливать в течении нескольких суток, то это может занять сотни терабайтов информации. И при этом событийных моментоа, ради которых ведется фиксация может быть очень немного. Поэтому имеет смысл кодировать и запоминать только изменения в картинках, а не сами картинки.

Задача 3.3 (Explore/Transform: разведка/преобразование)**.** на этом этапе происходит очистка данных и их первичная предобработка перед последующим анализом. Поэтому на этом этапе могут выявляться аномалии, которые могут внести серьезные искажения в результаты дальнейшего анализа данных с целью их фильтрации или сглаживания.

Например, нет смысла хранить результаты сбоев видеокамер и других процессов, которые не несут никакой информации. Недостоверные или искаженные данные. Резкие звуки или хлопки, хрипы, которые могут сопровождать запись голоса нужно вырезать или уменьшить, отфильтровать.

Задача 3.4 (Aggregate/Label: агрегирование/разметка)**.** на данном этапе происходит анализ и подготовка данных для использвоания в алгоритмах машинного обучения. Данные анализируются, кластеризуются, агрегируются, размечаются

Например, чтобы обучить "Умный дом"правильно реагировать на команды хозяина, необходимо собрать образцы голосовых команд, разметить их (т.е. указать, какая команда была произнесена), закодировать и проаназировать насколько хорошо произнесены команды и насколько их можно отличить от других команд, или от команд, произнесенных другими людьми.

На данном этапе необходимо определить основные метрики, которые могут быть испльзуемы для того, чтобы понять насколько хорошо решение той или иной задачи. Т.е. выделить числовые критерии качества решения задач.

Задача 3.5 (Learn/optimize: обучение/оптимизация)**.** данные для обучения подготовлены, метрики определены и теперь можно начинать экспериментировать, пробовать применять различные алгоритмы машинного обучения, сравнивать, подбирать их параметры, оценивать, проверять действие тех или иных факторов.

Задача данного этапа - исследовать применимость методов машинного обучения, выбрать наилучший, оптимзировать параметры методов, оценить возможный результат.

Например, на данном этапе мы подбираем метод или приблизиительную архитектуру нейросети, оцениваем качество работы алгоритмов, сравнимаем и выбираем среди них потенциально лучшие.

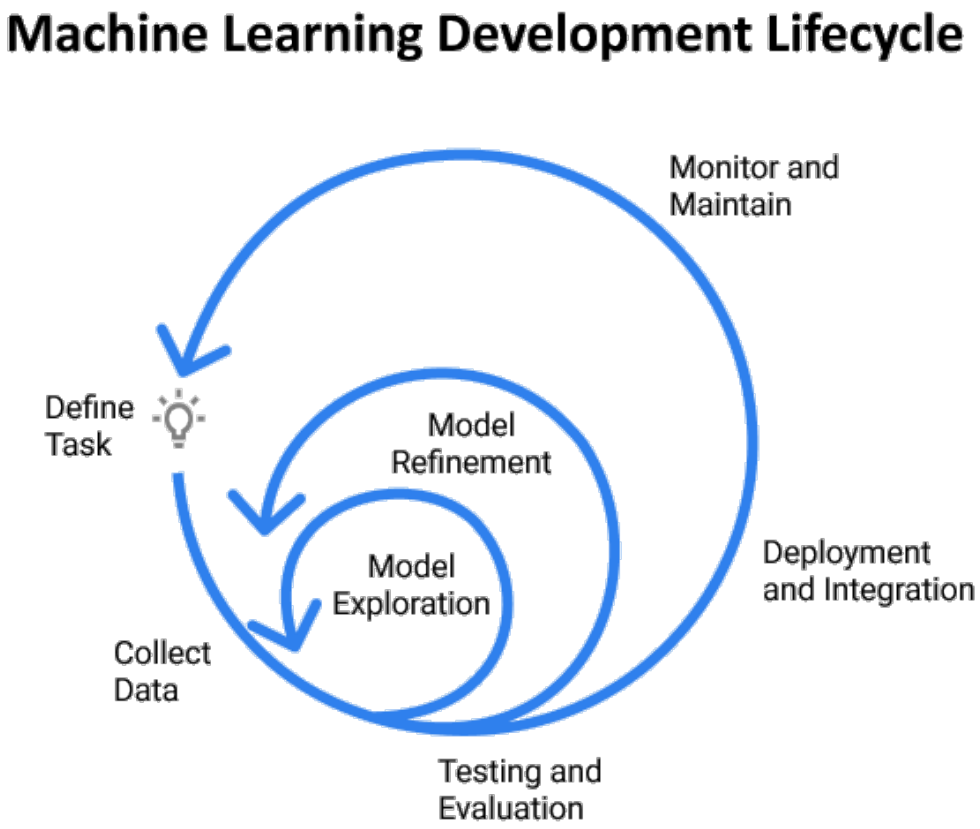


Рис. 3: Цикл решения задач машинного обучения:

Задача 3.6 (AI, Deep Learning: ИИ, глубокое обучение)**.** это вишенка на тортике, здесь мы тюним модели, доводим до совершенства результаты обучения и внедряем модели. Настраиваем вычислительную систему и запускаем систему в работу. Это самый наукоемкий этап, который требует очень хорошего знания тонкостей глубокого обучения.

На данном этапе мы замыкаем контур обратной связи, мы создаем вычислительную систему, которая решает поставленную задачу, накапливает, анализирует и готовит для обучения новые данные, корректирует на основе этих данных модель, тестирует ее и заменяет старую модель, если новая модель лучше решает поставленную задачу (см. рис. 3, [6])

4 Дата-инженер, Дата-аналитик и Дата-сайентист

Если мы вернемся к рис. 2, то увидим большое количество задач, которые надо решить, прежде чем удастся создать "умную"программу, решающую задачу машинного обучения или построения искусственного интеллекта. Вряд ли это под силу одному человеку. Поэтому возник целый ряд профессий, связанных с решением задач машинного обучения.

Кто же участвует в решении задач машинного обучения?

- **Дата-инженер** - это самый востребованный специалист в наше время, так как он и создает фундамент для решения многих задач, связанных с данными, не только для решения

задач ML. Инженерия данных связана с данными, а именно с их получением, доставкой, хранением и обработкой. Соответственно, основная задача инженеров — обеспечить надежную инфраструктуру для данных. Если мы посмотрим на иерархию потребностей (рис.2), инженерия данных занимает первые 2–3 этапа: сбор, перемещение и хранение, подготовка данных (см. [7], [8]).

- **Дата-аналитик** - это человек, который проводит первичный анализ данных, статистический анализ данных, оценивает возможность решения разных задач, умеет визуализировать важные свойства данных. В иерархии потребностей (рис.2), анализ данных занимает 3–4 этажи: первичная обработка и анализ данных, подготовка данных, агрегирование, разметка данных или подготовка данных для разметки.

- **Дата-сайентист** - Data Scientist (ученый по данным) — это специалист по интеллектуальной обработке данных, который может и то, что делает аналитик данных. И плюс к этому он имеет какое-то особенное умение или особо узкую специализацию (занимается распознаванием изображений или создает рекомендательные системы). В иерархии потребностей (рис.2), он занимает самые высокие 4–6 этажи. Его отличает хорошая математическая подготовка, навыки подготовки данных, знание и умение применять алгоритмы машинного обучения.

——

С точки зрения рассмотренных выше задач и профессий, авторы видят основную роль курса в том, чтобы:

- дать понимание спектра задач, которые можно решить с помощью алгоритмов машинного обучения;

- дать понимание подходов к решению этих задач, этапов решения, критериев успешности решения, принципов работы алгоритмов машинного обучения и анализа данных;

- научить применять современные пакеты программ для решения некоторого ряда задач машинного обучения и анализа данных.

В конце курса слушатель будет понимать как и уметь применять готовые инструменты для решения некоторых задач машинного обучения и анализа данных.

5 Вопросы для самоконтроля и контроля

1. Какую задачу можно назвать задачей машинного обучения?

2. Каким требованиям должна отвечать программная система, чтобы ее можно было отнести к реализации машинного обучения?

3. Чем отличается сильный ИИ от слабого ИИ?

4. Как можно протестировать сильный ИИ?
5. Перечислите иерархию задач машинного обучения
6. Приведите примеры задач, которые приходится решать прежде чем можно будет применить алгоритмы машинного обучения.
7. Какие задачи решает дата-инженер?
8. Какие задачи решает аналитик данных?
9. Кто такой дата-сайентист?

6 Практические задания

Задача 6.1 (Постановка задачи ML)**.** Сформулируйте постановку задачи ML для концепции "Умный Дом". Постановка задачи должна содержать описание:

- цели/задачи **Task** (например, улучшить продажи какого-то интернет-магазина);
- критерия качества решения задачи **Productivity** (например, количество заходов на продающий сайт, количество продаж, суммарная прибыль за месяц и.т.п.);
- используемых для решения задачи данных **Experience**, их происхождение, откуда они берутся и как размечаются

Список источников

[1] Технологии и концепции Industry 4.0 [Электронный ресурс]. URL: https://www.it.ua/ru/knowledge-base/technology-innovation (Дата обращения: 09.02.2022)

[2] Википедия. [Электронный ресурс] Тест Тьюринга (Дата обращения: 09.02.2022)

[3] Блог РБК-Тренды. Что такое машинное обучение и как оно работает. [Электронный ресурс] https://trends.rbc.ru/trends/industry (Дата обращения: 09.02.2022)

[4] Vas3k blog. Машинное обучение для людей [Электронный ресурс] https://vas3k.ru/blog/machine-learning/ (Дата обращения: 09.02.2022)

[5] Whilejean. The AI Hierarchy of Needs. [Электронный ресурс]. URL: https://medium.com/@whilejean0/the-ai-hierarchy-of-needs-270a5caa74c (Дата обращения: 09.02.2022)

[6] Jeremy Jordan. Building machine learning products: a problem well-defined is a problem half-solved. [Электронный ресурс] https://www.jeremyjordan.me/ml-requirements/ (Дата обращения: 09.02.2022)

[7] MaxRokatansky. OTUS corporate blog. Кто такие дата-инженеры, и как ими становятся? [Электронный ресурс] https://habr.com/en/company/otus/blog/452670/ (Дата обращения: 09.02.2022)

[8] amokryshev. Основные функции ETL-систем [Электронный ресурс] https://habr.com/en/post/248231/ (Дата обращения: 09.02.2022)

[9] Открытый курс машинного обучения [Электронный ресурс] https://ods.ai/tracks/open-ml-course (Дата обращения: 09.02.2022)

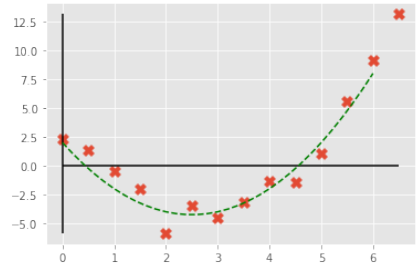
Поволжский государственный технологический университет

Нехаев Игорь Николаевич

Конспект

РАЗДЕЛ 1. ВВЕДЕНИЕ в МАШИННОЕ ОБУЧЕНИЕ

Лекция 2. Виды и постановки задач машинного обучения.
Обучение с учителем: регрессионный анализ и классификация



Аннотация: В лекции рассматриваются классы задач машинного обучения, постановки задач регрессионного анализа и классификации. Рассматриваются примеры различных задач машинного обучения и принципы их решения.

Ключевые слова: виды задач машинного обучения, модель данных, признаки, объектно-признаковая матрица, решающая функция, обучающая выборка, алгоритм машинного обучения, функция потерь, регрессионный анализ, классификация, примеры задач

Йошкар-Ола - 2022

Содержание

1 Принцип машинного обучения

Машинное обучение похоже на обучение человека (см. [1], [2]).

- Человек учится на своем опыте. Программа обучается на данных, которые она может сама загружать из онлайн-источников и специальных репозиториев.
- Чем больше опыт человека, тем лучше он решает новые задачи. Чем больше накоплено данных и больше их используется для обучения параметров выбранной модели, тем лучше модель описывает реальность и может быть использована для более точного решения соответствующих задач.
- обучение у человека происходит методом проб и ошибок, корректировок своих представлений после получения результатов своих решений и действий, обобщением накопленного опыта; обучение модели происходит аналогично - модели предъявляют новые данные, описывающие объект/ситуацию/процесс и модель пробует дать ответ на вопрос, если он не совпадает с реальностью, модель корректируется так, чтобы в следующий раз на этих данных дать правильный ответ; так происходит накопление опыта.

Идея применения машинного обучения состоит в организации двух циклов (см. рис. 1):

- **цикл построения или обучения модели** состоит из шагов: собираем и загружаем данные (Collect and load data), готовим данные для обучения модели (Create pipeline), обучаем модель (Train model), оцениваем и улучшаем модель.

- **цикл применения модели** состоит из шагов: загружаем модель (load), делаем прогноз (Make predictions).

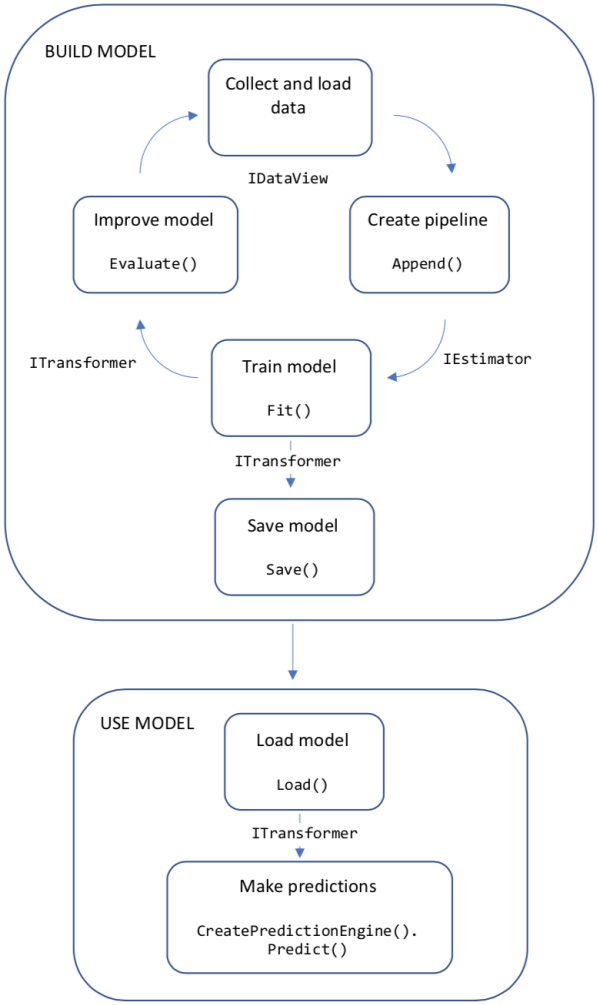


Рис. 1: Принцип машинного обучения: есть цикл обучения модели (Build Model) и есть цикл использвания модели (Use Model) ([3])

Рассмотрим концептуальную модель данных и модели машинного обучения (см. [4]).

2 Модели машинного обучения

Определение 2.1 (Модель данных для обучения)**.** *Пусть*

- *имеется конечное или бесконечное множество изучаемых однородных объектов (объекты реального мира/ситуации/процессы)* $\boldsymbol{O}$*;*
- *для некоторых объектов* $o^{(k)}, k = 1,..,K$ *мы можем получить их информационные описания*

$$\overline{x}^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}\right), k = 1,..,K$$

,

в виде векторов признаков, их характеризующих (n признаков);

- *для некоторых из этих объектов* $o^{(j)}, j = 1,..,J$ *мы также имеем значение интересующего нас показателя (ответа)*

$$y_j, j = 1,..,J$$

.

Таким образом, у нас есть множество описаний объектов $\bar{x}^{(j)}, j = 1,..,J$ *и могут быть известны сответствующие им (объектам) - метки* $y_j, j = 1,..,J$.

Определение 2.2 (Объектно-признаковая матрица)**.** *Если мы объединим все описания объектов в одной матрице* $\boldsymbol{X}$*, каждая строка которой содержит описание какого-то одного объекта, то мы получим объектно-признаковую матрицу. Число строк данной матрицы совпадает с числом описываемых матрицей объектов, а число столбцов совпадает с числом признаков, с помощью которых мы описываем эти объекты.*

Все известные ответы мы также можем объединить в одной матрице-столбце $\boldsymbol{Y}$.

Признаковые описания объектов сильно зависят от специфики задачи. Например, если объекты - это картинки, то признаки - это яркости пикселей. Если картинка цветная, то яркости пикселей в зависимости от каждого цвета. Если объектами являются тексты, то признаками могут быть частоты появления слов из словаря.

Различают несколько видов признаков.

Определение 2.3 (Виды признаков, описывающих объекты)**.** *Обозначим множество возможных значений признака* f_i *через* D_i*. Тогда*

- f_i *- бинарный признак, если* $D_i = \{0, 1\}$*;*
- f_i *- номинальный или категориальный признак, если* D_i *- перечислимое множество неупорядоченных значений (Иванов, Сидоров, Петров, ...);*
- f_i *- порядковый признак, если* D_i *- перечислимое множество упорядоченных значений (черный, серый, белый);*
- f_i *- количественный признак, если* $D_i = R$ *(числа);*

Определение 2.4 (Обучающая выборка)**.** *Под обучающей выборкой мы будем понимать некоторое множество L объектов, для которых известно их признаковое описание* $\bar{x}^{(i)}, i = 1,..,L$ *и, возможно, их метки* $y_i, i = 1,..,L$ *(**L** - **L**earning, обучение).*

Эти описания мы будем объединять и обозначать матрицей X^L*, каждая строка которой содержит описание какого-то объекта выборки. Все известные ответы мы также будем обозначать* Y^L.

Также нам будут нужны и другие выборки для анализа полученной модели - тестовая или контрольная. Их объектно-признаковую матрицу также будем обозначать буквами X^T или X^K, а соответствующие метки Y^T или Y^K.

Мы бы хотели получить ответы - значение интересующего нас показателя (распознать объект, оценить ожидаемую прибыль, узнать прогноз погоды, спрогнозировать изменение цены, ...), когда будет предъявлен новый объект, который еще не встречался нам ранее.

Определение 2.5 (Решающая функция или модель машинного обучения)**.** *Таким образом, у нас есть множество описаний объектов $\bar{x}^{(k)}, k = 1,..,K$ и могут быть известны соответствующие им (объектам) - метки $y_k, i = 1,..,K$.*

Функция fm, которая отображает с какой-то точностью множество описаний объектов $\boldsymbol{X}$ в множество ответов $\boldsymbol{Y}$ называется решающей функцией или моделью машинного обучения. Предполагается, что модель должна хорошо описывать некоторую неизвестную функцию f:

$$f : X \to Y$$

Определение 2.6 (Алгоритм машинного обучения)**.** *Алгоритм $\boldsymbol{a}$, который по известной обучающей выборке $X^L(, Y^L)$ строит модель fm называется алгоритмом машинного обучения:*

$$fm = a(X^L, Y^L)$$

Такоим образом решение задачи машинного обучения - это построение алгоритма машинного обучения, который на основании обучающей выборки строит модель, которая затем используется для поиска ответов на новые вопросы.

3 Классы задач машинного обучения

В зависимости от того, какие вопросы ставятся перед моделью и в зависимости от того, известны ли правильные ответы для входных данных или нет, различают следующие типы задач машинного обучения (см. рис.2, [5]).

- "Обучение с учителем"или "Обучение по прецедентам"(supervised learning). В этом классе задач известны (прецеденты) - описания объектов $X^{(i)}$ и ответы для них y_i (метки) для достаточно большого набора объектов. В этом случае говорят, что имеются размеченные данные. Подразумевается, что Учитель знает ответы на задания и обучает модель, показывая ей правильные ответы, чтобы она также могла находить эти ответы в будущем. Это очень похоже на обучение в школе. Когда ответы на учебные задачи заранее известны учителю.

- "Обучение без учителя". В этом классе задач известны только описания объектов $X^{(i)}$ для достаточно большого набора объектов. Подразумевается, что модель должна сама разметить данные (найти ответы), чтобы свести задачу к задаче обучения с учителем.

Рис. 2: Типология классических задач машинного обучения ([5])

- "Обучение с подкреплением"(Reinforcement Learning). Это нечто среднее между обучением с учителем и без. В этом классе задач также известны только описания объектов $X^{(i)}$. Подразумевается, что модель также должна сама искать ответы для объектов. Но, после того, как модель найдет ответ, учитель/жизнь может поощрить Вас за верный ответ или оштрафовать за плохой ответ (подкрепление). Это больше похоже на обучение в реальной жизни. Когда правильный ответ заранее никто не знает и можно понять насколько ответ был правильным только после принятия решения. Жизнь сама подскажет - правильным было решение или нет.

Рассмотрим более подробно каждый вид задачи в отдельности и приведем примеры ([4]).

4 Принцип решения задач обучения с учителем (supervised learning)

Самыми простыми задачами машинного обучения можно считать задачи обучения с учителем. Алгоритмы решения таких задач хорошо проработаны и, как правило, качество решения таких задач зависит только от качества данных.

Рис. 3: Примеры классических задач машинного обучения ([7])

- в этих задачах для части исходных объектов известны ответы (метки).
- мы обучаем модели на этой части объектов, для которых известны ответы и используем обученную модель для прогноза ответов на новых объектах, для которых неизвестны ответы (метки)..
- самой большой проблемой для решения этих задач является разметка данных.

Основной принцип решения задач обучения с учителем состоит в следующих шагах:

- формируется обучающая выборка X^L, Y^L, содержащая как можно больше разнообразных примеров объектов.
- выбирается структура модели - решающей функции $f_m(\bar{p}, x) : X \to Y$; здесь $\bar{p}$ - параметры функции;
- выбирается лосс $\mathfrak{L}(\bar{p}, X^L, Y^L)$ - функция потерь, которая вычисляет усредненный штраф за расхождение между значением решающей функции $f_m(\bar{p}, x)$ и реальной меткой объекта y для всех объектов с описанием (x, y) в обучающей выборке;
- выбирается и применяется алгоритм машинного обучения a, который оптимизирует параметры решающей функции $\bar{p}$ на обучающей выборке (X^L, Y^L) за счет минимизации функции потерь $a : \mathfrak{L}(\bar{p}, X^L, Y^L) \to min$.
- построенная модель $f_m(a, x)$ используется для прогноза на новых данных.

Рассмотрим разновидности и примеры таких задач (см. рис. 3).

5 Регрессионный анализ

Специфика задачи регрессионного анализа заключается в том, что ответы в таких задачах являются числами, т.е. $Y = \Re$.

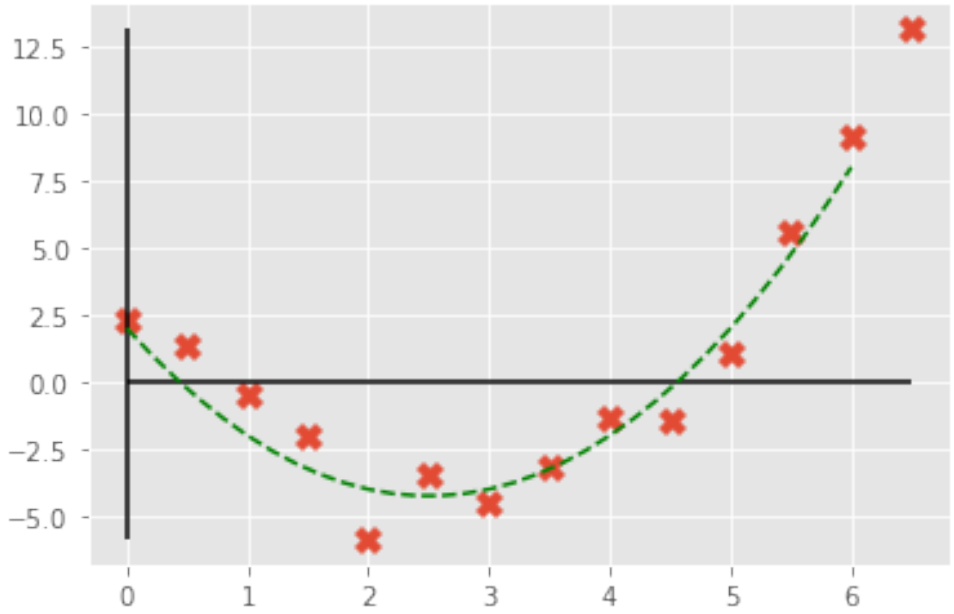


Рис. 4: Построение регрессионной зависимости по отдельным наблюдениям x и y

В качестве функции потерь чаще всего берется средний квадрат отклонения значений решающей функции от меток объекта:

$$\mathfrak{L}(\bar{p}, X^L, Y^L) = MSE(Y_m^L, Y^L) = \frac{1}{L}(Y_m^L - Y^L)^2 = \frac{1}{L}\sum_{i=1}^{L}(f_m(\bar{p}, x^{(i)}) - y_i)^2.$$

Здесь Y_m^L - это вектор ответов модели f_m на обучающей выборке:

$$Y_m^L = f_m(\bar{p}, X^L).$$

Рассмотрим конкретные примеры.

Пример 5.1 (Регрессионный анализ: оцениваем итоговый балл на курсе)**.** *Такая задача возникает когда нам нужно уметь прогнозировать какую-то численный показатель, величину (прибыль) по известным входным параметрам (вложения) (см. рис. 4). Например, по начальным результатам обучения спрогнозировать итоговый балл обучающегося на курсе. Таким образом, в качестве вектора признаков $\bar{x}^{(i)}$, описывающих объект (обучающегося) выступают его результаты обучения на начальном этапе (первой половине курса), т.е. его баллы по каждому виду работ. А в качестве ответа/метки обучающегося y_i - его итоговый балл в конце курса. Чтобы получить такие размеченные данные, необходимо иметь результаты обучения как минимум одного потока обучающихся на курсе. Обучив модель на этих прошлых данных (регрессия - возврат в прошлое), мы сможем использовать модель для прогноза баллов для будущих потоков.*

Пример 5.2 (Прогноз: заказ продуктов)**.** *Немного помечтаем. Мы хотим, чтобы Умный Дом сам заказывал требуемые продукты, когда они заканчиваются. Но тогда ему необходмио уметь оценивать имеющиеся запасы, динамику потребления и прогнозировать день когда определенные продукты закончатся, чтобы к этому моменту заказать новые.*

6 Классификация объектов

Специфика задачи классификации объектов заключается в том, что ответы в таких задачах являются бинарными (см. рис. 5) или категориальными (номинальными или порядковыми) признаками.

В качестве функции потерь чаще всего берется категориальная кросс-энтропия, которая штрафует за отклонение значений решающей функции от меток объекта:

$$\mathfrak{L}(\bar{p}, X^L, Y^L) = CCE(Y_m^L, Y^L) = -\frac{1}{L}\sum_{i=1}^{L}\log p(f_m(\bar{p}, x^{(i)}) = y_i).$$

Здесь $p(f_m(\bar{p}, x^{(i)}) = y_i)$ - это вероятность того, что ответ решающей функции совпадает с меткой i-го объекта.

Рассмотрим конкретные примеры.

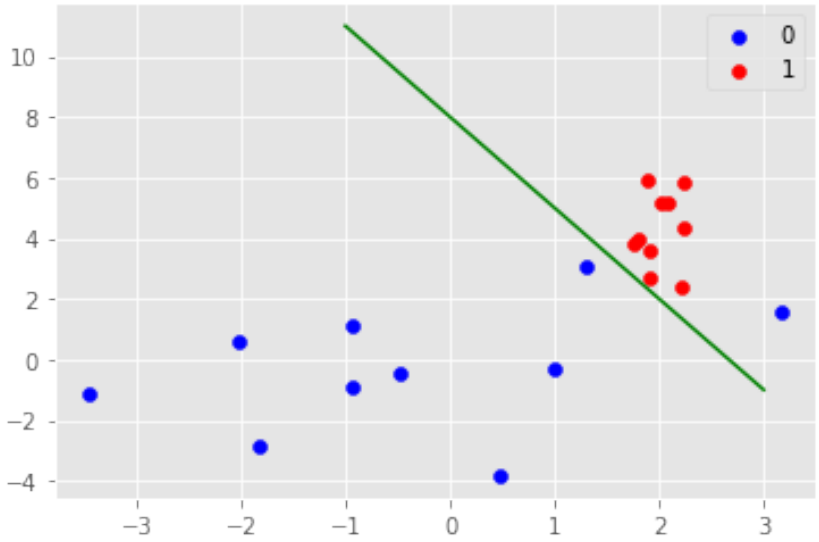


Рис. 5: Пример бинарной классификации: задача отделить красные точки от синих - провести границу

Пример 6.1 (Классификация: обучающихся на курсе)**.** *чаще нам необходимо оценить не итоговый балл обучающегося на курсе, а как он закончит курс - будет ли он успевающим (отличником, хорошистом, троечником) или неуспевающим. Этот пример отличается от прогноза итогового балла (см. пример 5.1) тем, что в качестве метки рассматривается номер класса, соответствующего итогу обучения на курсе, а не его итоговый балл. Для получения таких размеченных данных нам также необходимо иметь результаты обучения как минимум одного потока обучающихся на курсе.*

Можно перечислить огромное количество задач, которое относится к классическим задачам обучения с учителем (см. [4]):

- медицинская диагностика: необходимо определить диагноз заболевания;
- задача кредитного скоринга: необходимо решить - выдавать кредит клиенту или нет;
- предсказание оттока клиентов: предсказать уйдет или не удет клиент в следущем месяце в другую компанию;
- задача биометрической идентификации: необходимо идентифицировать человека по отпечатку пальца или сетчатке глаза, ...;
- прогнозирование стоимости недвижимости;
- прогнозирование объема продаж...

У каждой задачи есть своя специфика, касающаяся того, сколько данных имеется, с какими ошибками они получены, есть ли выбросы, как оценить качество решения задачи. Подумайте над тем, что является объектом в этих задачах, что является меткой объекта, какие признаки могут использоваться для описания объектов.

7 Вопросы для самоконтроля и контроля

1. Из каких циклов состоит применение машинного обучения?
2. Что понимается под данными, применяемыми для обучения? Опишите модель данных.
3. Что вляется исходными данными для алгоритма машинного обучения и что является результатом применения алгоритма машинного обучения?
4. Сколько строк и столбцов содержит объектно - признаковая матрица? От чего это зависит?
5. Опишите шаги применения машинного обучения рот решении задач с учителем.
6. Какие существуют типы признаков? Перечислите их и опишите чем они отличаются друг от друга.
7. Какие существуют типы задач машинного обучения?
8. Перечислите виды задач обучения с учителем и опишите чем они отличаются друг от друга.
9. Дайте примеры задач регрессионного анализа.
10. Приведите примеры задач классификации.

8 Практические задания

Задача 8.1 (Постановка задачи ML)**.** Придумайте постановку задачи регрессионного анализа для концепции "Умный Дом". Постановка задачи должна содержать описание:

- цели/задачи **Task**;
- критерия качества решения задачи **Productivity**;
- используемых для решения задачи данных **Experience**, их происхождение, откуда они берутся и как размечаются

Задача 8.2 (Постановка задачи ML)**.** Придумайте постановку задачи классификации для концепции "Умный Дом". Постановка задачи должна содержать описание:

- цели/задачи **Task**;
- критерия качества решения задачи **Productivity**;
- используемых для решения задачи данных **Experience**, их происхождение, откуда они берутся и как размечаются

Список источников

[1] Ян Лекун. Как учится машина. Революция в области нейронных сетей и глубокого обучения. (Библиотека Сбера: Искусственный интеллект). — М.: Альпина нон-фикшн, 2021. — ISBN 978-5-907394-29-2.

[2] Blog "toward data science". All Machine Learning Models Explained in 6 Minutes [Электронный ресурс] https://towardsdatascience.com/all-machine-learning-models-explained-in-6-minutes-9fe30ff6776a (Дата обращения: 09.02.2022)

[3] Microsoft Docs. Что такое ML.NET и принципы работы этой системы [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work (Дата обращения: 09.02.2022)

[4] Машинное обучение (курс лекций, К.В.Воронцов) [Электронный ресурс] http://www.machinelearning.ru/wiki/index.php (Дата обращения: 09.02.2022)

[5] Блог РБК-Тренды. Что такое машинное обучение и как оно работает. [Электронный ресурс] https://trends.rbc.ru/trends/industry (Дата обращения: 09.02.2022)

[6] Открытый курс машинного обучения [Электронный ресурс] https://ods.ai/tracks/open-ml-course (Дата обращения: 09.02.2022)

[7] Vas3k blog. Машинное обучение для людей [Электронный ресурс] https://vas3k.ru/blog/machine-learning/ (Дата обращения: 09.02.2022)

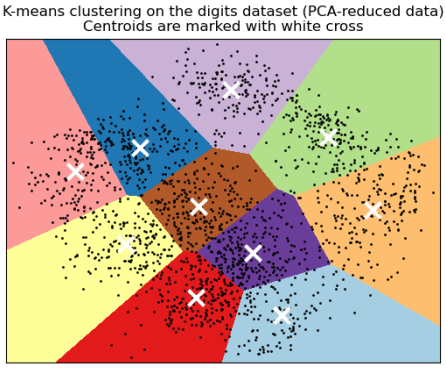
Поволжский государственный технологический университет

Нехаев Игорь Николаевич

Конспект

РАЗДЕЛ 1. ВВЕДЕНИЕ в МАШИННОЕ ОБУЧЕНИЕ

Лекция 3. Обучение без учителя: анализ структуры данных.
Фреймворки для решения задач машинного обучения



Аннотация: В лекции рассматриваются классы задач машинного обучения,
постановки задач обучения без учителя и обучения с подкреплением.
Рассматриваются фреймворки решения задач машинного обучения и схемы их решения.

Ключевые слова: обучение без учителя, кластеризация, снижение размерности
обучение с подкреплением, примеры задач, фреймворки решения задач ML, стек технологий

Йошкар-Ола - 2022

Содержание

1 Классы задач машинного обучения

В этой лекции мы рассмотрим задачи машинного обучения, в которых ответы для входных данных неизвестны. Это задачи обучения без учителя и задачи обучения с подкреплением (см. рис.1, [1]).

Напомним краткие характеристики данных задач.

- "Обучение без учителя". В этом классе задач известны только описания объектов $X^{(i)}$ для достаточно большого набора объектов. Подразумевается, что модель должна сама разметить данные (найти ответы), чтобы свести задачу к задаче обучения с учителем.
- "Обучение с подкреплением"(Reinforcement Learning). Это нечто среднее между обучением с учителем и без. В этом классе задач также известны только описания объектов $X^{(i)}$. Подразумевается, что модель также должна сама искать ответы для объектов. Но, после того, как модель найдет ответ, учитель/жизнь может поощрить Вас за верный ответ или оштрафовать за плохой ответ (подкрепление). Это больше похоже на обучение в реальной жизни. Когда правильный ответ заранее никто не знает и можно понять насколько ответ был правильным только после принятия решения. Жизнь сама подскажет - правильным было решение или нет.

Рассмотрим более подробно каждый вид задачи в отдельности и приведем примеры ([2]).

Рис. 1: Типология классических задач машинного обучения ([1])

2 Обучение без учителя (unsupervised learning)

В задачах обучения без учителя нам не известны метки (классы), которым принадлежат объекты. Нам нужно проанализировать структуру данных, чтобы понять как лучше подготовить данные и свести данную задачу к задачам обучения с учителем. Т.е. решение этих задач, как правило, предшествуют решению задач с обучения с учителем, играют вспомогательную роль.

Перечислим задачи анализа данных, с которыми обычно сталкивается дата-машинист, прежде чем сможет реализовать машинное обучение и решение реальной задачи

- предобработка данных: заполнение пропусков, приведение к удобному формату;
- добавление новых более удобных предикторов, меток объектов;
- графический и статистический анализ распределений объектов по имеющимся параметрам, по каждому параметру отдельно;
- выявить структуру данных:
  - снизить размерность данных с сохранением структуры,
  - графически проанализировать структуру данных,

Рис. 2: Примеры классических задач машинного обучения ([4])

– кластеризовать объекты, чтобы выделить классы типовых, похожих друг на друга объектов, проанализировать их генезис, чтобы подумать как их можно использовать для решения поставленных задач.

К задачам анализа структуры данных, в частности, относятся (см. рис. 1):

- задача кластеризации данных; эту задачу приходится решать в том числе, чтобы разметить объекты;
- задача снижения размерности данных, задача многомерного шкалирования, факторный анализ;
- поиск правил, объясняющих причинную взаимосвязь между различными признаками в данных.

Рассмотрим виды и примеры таких задач (см. рис. 2).

2.1 Задачи кластеризации объектов

Определение 2.1 (Кластеризация объектов)**.** ***Кластеризация*** *— объединение похожих объектов в группы, или кластеры. Кластеризацию также называют кластерным анализом или анализом пространственной структуры объектов.*

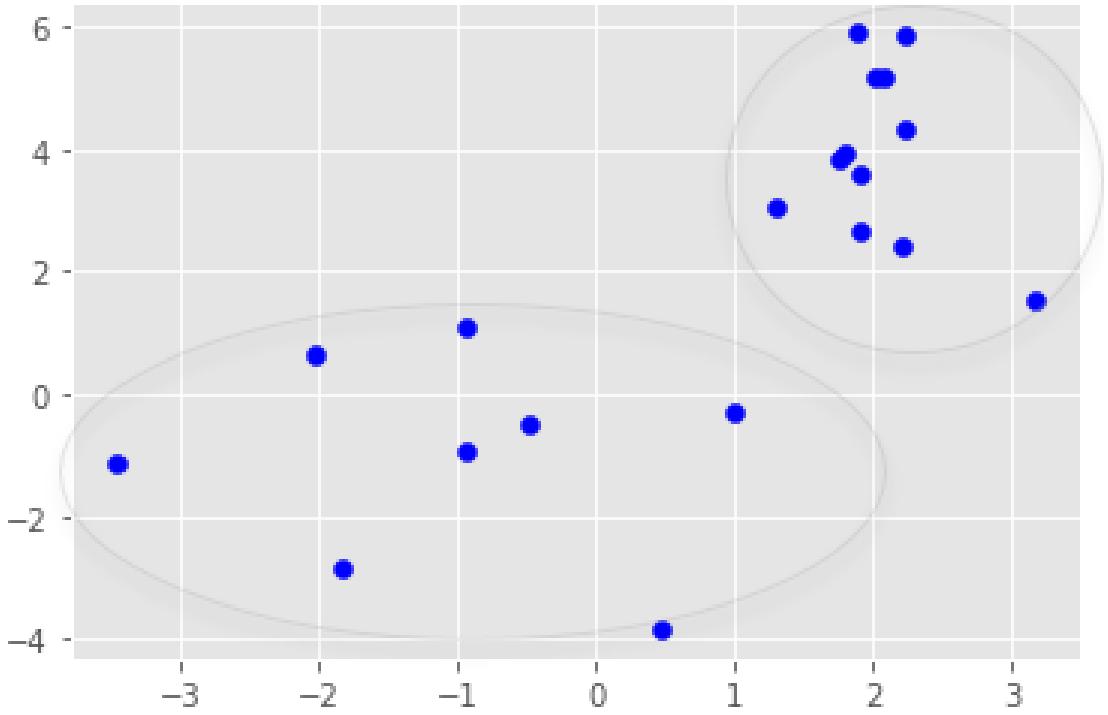


Рис. 3: Задача выделения кластеров схожих объектов

Результаты кластеризации зависят от метода и от того, как оценивать схожесть или различие объектов, т.е. расстояние между ними. Чем дальше объекты друг от друга, тем меньше сходства и больше различия. Кластеризация отличается от задачи классификации тем, что классы здесь заранее неизвестнеы и задачей кластеризации и является определить какие объекты в какие классы (кластеры) должны объединиться (см. рис. 3). Результат кластерного анализа и может служить разметкой данных для дальнейшего прмиенения методов обучения с учителем.

Пример 2.1 (Кластерный анализ: выявить кластеры типичных слушателей на онлайн-курсе)**.** *Вы имеете большую статистику поведения слушателей на онлайн-курсе - как они проходят курс, как набирают баллы, в какой последовательности смотрят материалы. Вам хочется выделить обучающихся с похожим (типичным) поведением для того, чтобы выстроить для них эффектиное сопровождение.*

Пример 2.2 (Кластерный анализ: выявить кластеры типичных покупателей в супермаркетах или на сайте интернет-магазина)**.** *Вы имеете большую статистику поведения слушателей на онлайн-курсе - как они проходят курс, как набирают баллы, в какой последовательности смот-*

рят материалы. Вам хочется выделить обучающихся с похожим (типичным) поведением для того, чтобы выстроить для них эффектиное сопровождение.

Иногда важно не только выделить типичные объекты в кластеры, но и найти и выделить аномалии - нетипичные объекты, которые непохожи на других объектов. Можно считать эту задачу поиска аномалий - дополнением к задаче кластеризации. Некоторые методы кластеризации одновременно размечают и аномальные объекты.

Пример 2.3 (Поиск аномалий: обнаружение попыток подбора паролей)**.** *Компания предоставляет услуги онлайн-платежей. На счетах ее клиентов хранятся денежные средства. Клиенты заходят в свои аккаунты и совершают платежные операции. Бывает, что клиент забывает пароль и начинает вводить разные вариации, вспоминая свой пароль. А бывает, что злоумышленник пытается подобрать пароль к аккаунтам клиентов. Имеются разные способы подбора парлей. Злоумышленник может подбирать варианты пароля к одному аккаунту или, найдя где-то базу паролей, проверяет пару логин-пароль в данной платежной системе. Задача машинного обучения заключается в том, чтобы, на основе анализа потока событий, выявить такие аномалии поведений клиентов, чтобы вовремя заблокировать ip-адреса злоумышленника, с которых идет перебор паролей.*

2.2 Задачи снижения размерности описания объектов

Снижение размерности описания объектов может быть полезно в нескольких отношениях:

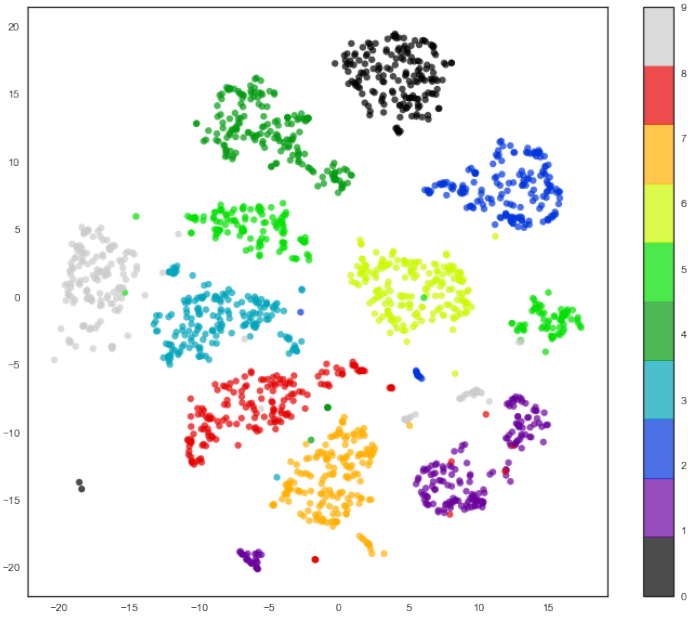


Рис. 4: Снижение размерности: каждая точка на плоскости представляет результат кодирования изображения цифры размером 28 x 28 пикселей с помощью метода снижения размерности tSNE; цвет соответствует изображенной цифре; видно, что изображения различных цифр достаточно хорошо отделимы ([3])

- визуализировать пространственную структуру данных; конечно, для этого необходимо спроектировать пространственную структуру описаний объектов из многомерного пространства в 2-х или 3-х мерное пространство с сохранением этой структуры (см. рис. 4);
- выявить главные признаки описания объектов (факторы), а все второстепенные шумы убрать; для этого используются методы факторного анализа (см. рис. 5);
- снизить требуемые объемы памяти и время расчета, используемые для хранения и обучения модели.

Рис. 5: Выделение главное фактора: красная линия показывает напрвление оси главного фактора. Зная значение главного фактора - расположение точки на его оси (одно число) можно получить координаты точки на плоскости (два числа)

Пример 2.4 (Снижение размерности: кодирование многомерных объектов и визуальный анализ)**.** *Предположим, что мы имеем определенную статистику поведения сотрудников компании на обучающих корпоративных курсах. Вам хочется на основе этой статистики выделить сотрудников, которые не будут прогрессировать в работе. Но вы не знаете какая информация для этого будет полезна. На основе какой информации можно разделить сотрудников на кластеры, чтобы выделить интересующий нас кластер. В этом случае полезно взглянуть на пространственную структуру объектов на основе имеющейся информации и понять есть ли там интересующие нас кластеры. Для этого необходимо воспользоваться методом снижения размерности описания объектов, сохраняющем пространственную структуру объектов*

2.3 Задачи поиска правил, описывающих закономерности в описании объектов

Задача поиска правил тесно связана с задачей кластеризации или поиска типичных объектов. Дело в том, что типичные объекты схожи между собой тем, что они подчиняются одинаковым закономерностям. Например, если мы имеем описание студентов курса "Математика и Python"в

виде векторов их оценок за всевозможные виды работ на курсе (балл.1, ..., балл.N), то разные кластеры обучающихся будут образованы в зависимости от того, что понимают/умеют и не понимают/не умеют слушатели курса. Например, если слушатель не смог установить на компьютер Python и пакет блилиотек анаконду, то он не продвинется далее 2-й темы курса. Потому как далее требуется выполонять задания по программированию. Если он плохо знает дифференциальное исчисление, то ему не осилить разделы, связанные с оптимизацией и обучением моделей. Таким образом, по результатам обучения на курсе можно сформулировать правила вида:

ПРАВИЛО: **ЕСЛИ** балл.1 (установка Python) = 0 **ТО** балл.n (n>2) = 0

Таким образом ассоциативные правила представляют собой механизм нахождения логических закономерностей между признаками объектов.

Пример 2.5 (Поиск правил: выявление закономерностей поведения жителей Умного Дома)**.** *Умный Дом может собирать информацию о последовательности действий жителей (пришел домой, включил свет после захода солнца, попросил включить обогрев, когда температура опустилась ниже 18 градусов, ...) Мы хотим на основе данной информации построить правила, которые определяют некоторые привычки хозяев, чтобы в будущем им подсказывать или напоминать или автоматизировать эти действия*

Другие примеры подобных задач вы можете найти в [1], [4].

3 Обучение с подкреплением (Reinforcement Learning)

Этот вид обучения очень привлекателен, но является сравнительно мало практически применимым на данном этапе. Идут интенсивные поиски более совершенных методов обучения с подкреплением.

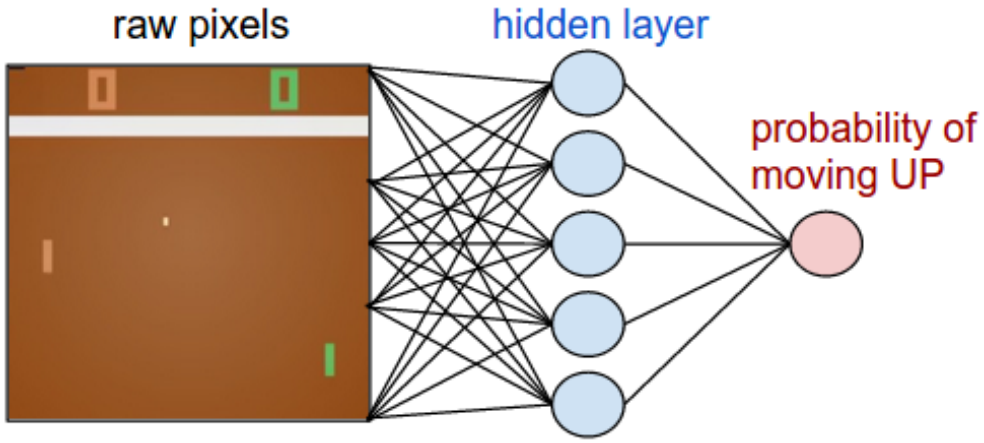


Рис. 6: Нейросетевой агент игры в пинг-понг: человек двигает коричневую ракетку (слева), а агент - зеленую ракетку (справа), агенту передается информация о том где находится шарик в данный момент и агент должен принять решение куда двигать ракетку, чтобы отбить шарик ([6])

Больше всего прмеров таких задача можно найти в области компьютерных игр (см. [5]). Но это не значит, что методы обучения с подкреплением совсем неприменимы для решения практиче-

ских задач. Просто пока эти методы проигрывают другим, более классическим методам решения практических задач.

Самые яркие примеры задач, в которых эти методы достигли уровня state-of-art:

- компьютерные игры приставки Atari на плоскости, в частности, игра в пинг-понг (см. рис. 6); нейросетевой метод обучения с подкреплением после 100 игр может играть наравне с человеком;
- задача обучения агентов виртуальных игровых миров, которые составляют конкуренцию человеку также способны обучаться на основе получаемых подкреплений - очков в игре, её выигрыша или проигрыша;
- игра в ГО - программа для игры в го AlphaGo в марте 2016 года выиграла со счетом 4:1 у профессионала 9-го дана (высшего ранга); обучение AlphaGo такому достижению произошло методом обучения с подкреплением после проведения 1 млн. игр сам с собою.

4 Библиотеки и фреймворки машинного обучения

Python стал популярным благодаря наличию большого числа мощных библиотек реализации машинного обучения и конструирования искусственного интеллекта.

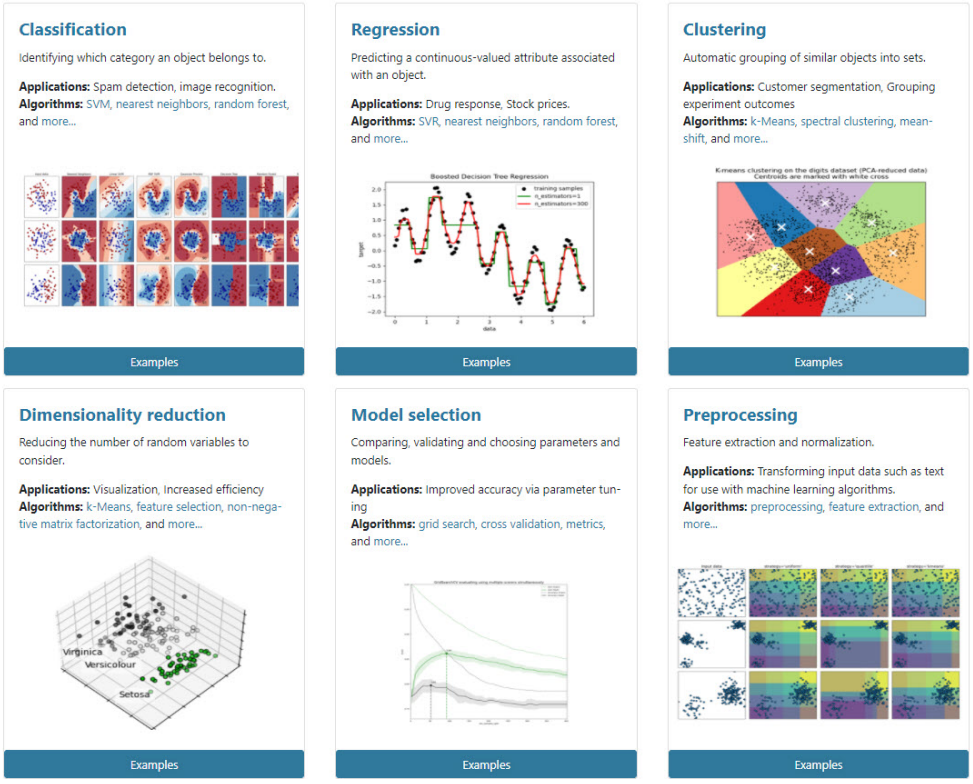


Рис. 7: Классы задач, для которых реализованы фреймворки в библиотеке sklearn

Основная библиотека по машинному обучению, которая реализует основной фреймворк машинного обучения - это библиотека **scikit-learn** или **sklearn** (https://scikit-learn.org/stable/index.html).

Данная библиотека содержит фремворки решения следующих задач (см. рис. 7):

- классификация объектов (Classification);
- построение регрессионных зависимостей (Regression);
- кластеризация (Clustering);
- Снижение размерности (Dimensionality reduction);
- Выбор лучшей модели (Model selection);
- Предобработка: отбор признаков и нормализация данных (Preprocessing).

Библиотека содержит много готовых данных и примеров применения методов машинного обучения к этим данных при решении перечисленных задач.

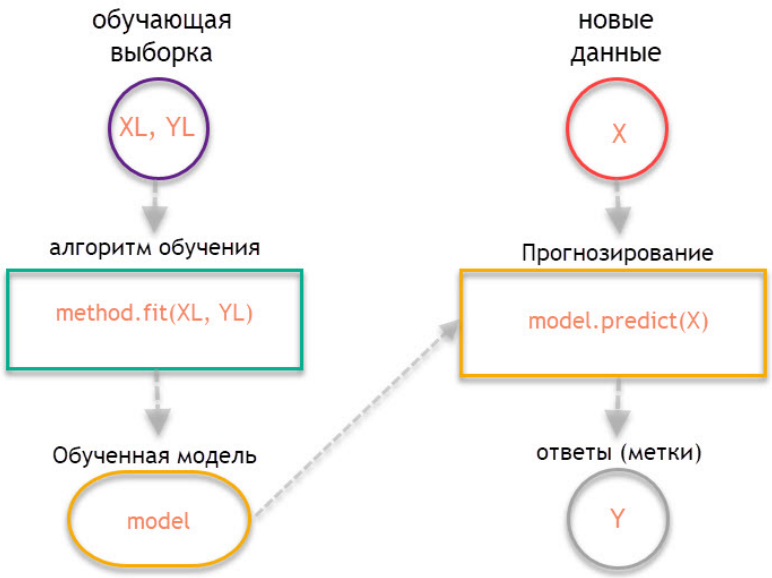


Рис. 8: Схема решения классических задач машинного обучения в библиотеке sklearn

Рассмотрим фреймворки решения задач классификации, регрессии, кластризации и снижения размерности.

Схема решения этих задач представлена на рис. 8. На рисунке схематически представлена реализация двух циклов машинного обучения - собственно обучение модели и ее использование.

Обучение модели заключается в применении выбранного метода/алгоритма обучения. Каждый алгоритм в библиотеке реализуется своим классом. Поэтому для решения задачи необходимо:

1. Создать объект данного класса:

$$method = Method();$$

2. создать и обучить модель - для этого обратиться к методу fit алгоритма/метода и передать ему обучающую выборку

$$model = method.fit(X^L, Y^L);$$

3. после обучения модели мы можем воспользоваться ею для прогноза на новых данных:

$$Y = model.predict(X)$$

Конкретные примеры применения различных методов необходимо рассмотреть на практике.

——

С точки зрения рассмотренных выше задач и профессий, авторы видят основную роль курса в том, чтобы:

- дать понимание спектра задач, которые можно решить с помощью алгоритмов машинного обучения;
- дать понимание подходов к решению этих задач, этапов решения, критериев успешности решения, принципов работы алгоритмов машинного обучения и анализа данных;
- научить применять современные пакеты программ для решения некоторого ряда задач машинного обучения и анализа данных.

В конце курса слушатель будет понимать как и уметь применять готовые инструменты для решения некоторых задач машинного обучения и анализа данных.

5 Вопросы для самоконтроля и контроля

1. Какие существуют типы задач машинного обучения? Перечислите их и опишите чем они отличаются друг от друга.
2. Какие существуют виды задач обучения без учителя? Перечислите их и опишите чем они отличаются друг от друга.
3. Зачем нужна кластеризация данных? Приведите примеры задач кластеризации.
4. Зачем нужно снижать размерность описания данных? Приведите примеры.
5. Для решения какой задачи может использоваться поиск правил?
6. Для решения каких задач могут применяться алгоритмы обучения с подкреплением?
7. Какие типы задач можно решать испльзованием библиотеки sklearn?
8. Опишите основной фрейворк решения классических задач машинного обучения.

6 Практические задания

Задача 6.1 (Постановка задачи ML)**.** Придумайте постановку задачи кластеризации для концепции "Умный Дом".

Задача 6.2 (Применить фрейворк ML)**.** Посмотрите описание методов при решении задачи классификации. Выберите метод классификации. Примените фреймворк для обучения модели выбранным методом при решениии задачи классификации цветков ирисов по их описаниям. Проанализируйте полученный результат.

Список источников

[1] Блог РБК-Тренды. Что такое машинное обучение и как оно работает. [Электронный ресурс] https://trends.rbc.ru/trends/industry (Дата обращения: 09.02.2022)

[2] Машинное обучение (курс лекций, К.В.Воронцов) [Электронный ресурс] http://www.machinelearning.ru/wiki/index.php (Дата обращения: 09.02.2022)

[3] Открытый курс машинного обучения [Электронный ресурс] https://ods.ai/tracks/open-ml-course (Дата обращения: 09.02.2022)

[4] Vas3k blog. Машинное обучение для людей [Электронный ресурс] https://vas3k.ru/blog/machine-learning/ (Дата обращения: 09.02.2022)

[5] Arthur Juliani. New Professions Lab. Введение в обучение с подкреплением: от многорукого бандита до полноценного RL агента [Электронный ресурс] https://habr.com/en/company/newprolab/blog/343834/ (Дата обращения: 09.02.2022)

[6] Andrej Karpathy blog. Deep Reinforcement Learning: Pong from Pixels [Электронный ресурс] http://karpathy.github.io/2016/05/31/rl/ (Дата обращения: 09.02.2022)

[7] Scikit-learn. Machine Learning in Python [Электронный ресурс] https://scikit-learn.org/stable/index.html (Дата обращения: 09.02.2022)

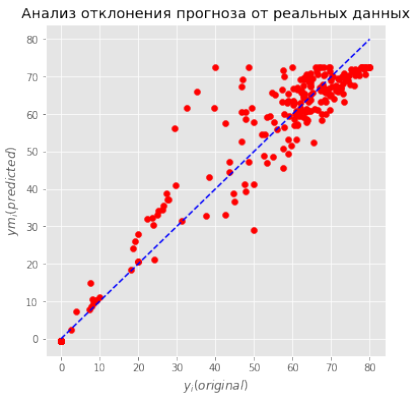
Поволжский государственный технологический университет

Нехаев Игорь Николаевич

Конспект

РАЗДЕЛ РЕШЕНИЕ ЗАДАЧ МАШИННОГО ОБУЧЕНИЯ

Лекция 4. Решение задач регрессионного анализа



Аннотация: В лекции рассматриваются постановка задачи построения линейной регрессии методы построения линейной регрессии, методы регуляризации - L1 и L2 регуляризация, метрики точности модели, вопросы выбора структуры модели.

Ключевые слова: регрессионный анализ, линейная регрессия, базисные функции, матрица Вандермонда, матрица Грама, метод наименьших квадратов, нормальная система уравнений, L2-регуляризация, L1-регуляризация, функция потерь МНК

Йошкар-Ола - 2022

Содержание

1 Постановка задачи построения линейной регрессии

Регрессия - дословно, это возвращение назад. Под регрессионной зависимостью мы подразумеваем зависимость, построенную по полученным ранее данным. Т.е. к задачам регрессионного анализа данных относятся задачи, в которых на основании ранее полученных данных наблюдения требуется построить зависимости одних показателей от других.

В этой теме мы рассмотрим задачи построения линейных регрессионных зависимостей. (см. [1–3]).

Рассмотрим общую постановку задачи регрессионного анализа или задачу аппроксимации функции по известным данным.

Определение 1.1 (Задача регрессионного анализа)**.** *Пусть в результате наблюдений некоторого процесса мы получили с вами набор данных вида* $(x^{(i)}, y_i), i = 1..n$,

где

- $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, .., x_m^{(i)})$ *- вектор значений входных или предиктивных переменных,*
- *от которых зависят* y_i *- значения зависимой или выходной переменной некоторого процесса,*
- *и мы считаем, что имеется некоторая функция* $f(\bar{x})$*, такая что связывает выходную переменную с входными,*

- *и нам известны значения этой функции $f(\bar{x})$ в заданных точках с ошибками:*

$$\hat{y}_i = f(\bar{x}^{(i)}) + \varepsilon_i,\ i = 1..n.$$

Нам надо попытаться восстановить связь между $\bar{x}$ и y. Т.е. задача - сконструировать аппроксимирующую (регрессионную) функцию $\hat{f}(\bar{x})$ такую, которая хорошо описывает имеющиеся данные с учетом ошибок:

$$\hat{f}(x^{(i)}) \approx \hat{y}_i,\ i = 1..n.$$

Для того, чтобы начать решать поставленную задачу необходимо разобраться с двумя вопросами - из какого класса функций мы будем выбирать апроксимирующую функцию $\hat{f}(\bar{x})$ и какой критерий использовать для выбора.

Рассмотрим постановку задачи построения линейной регрессии методом наименьших квадратов.

Определение 1.2 (Задача построения линейной регрессии)**.** *Если в качестве класса функций K_f, из которого выбирается аппроксимирующая функция $\hat{f}(\bar{x})$ мы используем класс обобщенных полиномов по некоторой выбранной системе функций $\varphi_0(x), \varphi_1(x), ..., \varphi_m(x)$, то можно говорить о **задаче построения линейной регрессии**, так как, можно считать, что по исходным данным мы будем строить модельную линейную функцию вида:*

$$y_M(x) = \sum_{j=0}^{m} a_j * X_j$$

где $X_j = \varphi_j(x)$ - называют обобщенными переменными. Их значения становятся известными, как только мы выбрали значение исходной переменной x.

В качестве критерия, по которому подбирают коэффициенты линейной регрессии $a_j, j = 0..m$ используют критерий наименьших квадратов. В этом случае говорят о решении задачи построения линейной регрессии с использованием критерия о наименьших квадратов.

Определение 1.3 (Задача о построении линейной регрессии методом наименьших квадратов)**.** *Задачу построения линейной регрессии с использованием критерия наименьших квадратов можно записать в следующем виде:*

$$\Phi 2(\bar{a}) = \sum_{i=0}^{n} (y_i - P_m(x_i, \bar{a}))^2 \to min$$

где

$$P_m(x_i, \bar{a}) = \sum_{j=0}^{m} a_j * \varphi_j(x_i)$$

Или

$$||X * \bar{a} - Y||_2 \to min,$$

Здесь матрица X - матрица значений базисных функций в узлах ($X_{i,j} = \varphi_j(x_i)$) (матрица Вандермонда);

вектор Y - вектор значений аппроксимируемой функции в узлах;

вектор $\bar{a}$ - вектор неизвестных коэффициентов.

Рассмотрим конкретный пример.

Пример 1.1 (Прогноз итогового балла на курсе)**.** *Пусть имеется выборка из 1000 студентов их результатов выполнения первых 5 заданий курса - 5-мерных векторов $x = (x_1, ..., x_5)$ и соответствующих значений y - итоговой суммы баллов обучения на курсе.*

Мы хотим построить линейную регрессию вида $y = \sum_{j=0}^{5} a_j * x_j$ ($x_0 = 1$), чтобы по результатам выполнения первых 5 заданий курса попытаться спрогнозировать их итоговый балл.

Попробуйте определить размерность соответствующей матрицы Вандермонда X, если мы не добавляем дополнительные признаки, кроме указанных в примере предикторов.

2 Решение задачи. Нормальная система уравнений

Чтобы найти решение задачи

$$||X * \bar{a} - Y||_2^2 \to min.$$

необходимо найти производную (градиент) квадрата нормы по вектору $\bar{a}$ b приравнять к нулевому вектору.

Распишем квадрат нормы в виде скалярного произведения, воспользуемся дистрибутивностью и приведем подобные:

$$||X * \bar{a} - Y||_2^2 = (X * \bar{a} - Y, X * \bar{a} - Y).$$

Вычислим производную по вектору $\bar{a}$ от полученного выражения, т.е. найдем градиент данной функции. Учтем, что скалярное произведение векторов можно переписать в "матричном"виде так $(a,b) = (b,a) = a^T * b = b^T * a$

$$(X * \bar{a} - Y, X * \bar{a} - Y)'_{\bar{a}} = 2 * X^T * (X * \bar{a} - Y).$$

Приравняв градиент к нулевому вектору, и разделив на 2 левую и правую часть, получаем нормальную систему уравнений:

$$2 * X^T * (X * \bar{a} - Y) = 2 * X^T * X * \bar{a} - 2 * X^T * Y = \bar{0}$$

$$X^T * X * \bar{a} = X^T * Y \ (NormalSystem)$$

Определение 2.1 (Матрица Грама)**.** *Матрицу попарных скалярных произведений базисных векторов называются матрицей Грама:* $\Gamma = X^T * X$*. Также называется ковариационной матрицей параметров x:* $cov(X)$*.*

Если базисные функции линейно-независимы на множестве узлов, то и $det() \neq 0$ и нормальная система имеет единственное решение.

Если столбцы матрицы X (вектора значений базисных функций) не просто линейно-зависимы, а ортогональны, то матрица $\Gamma = X^T * X$ будет диагональной.

3 Пример построения линейной регрессии

Для решения системы уравнений воспользуемся библиотекой numpy, модулем linalg. Для визуализации будем пользоваться библиотекой matplotlib:

import numpy as np

import matplotlib.pyplot as plt

plt.style.use('ggplot')

Сгенерируем данные следующим образом. В качестве восстанавливаемой функции возьмем параболу и добавим случайные ошибки в интервале $(-\varepsilon; +\varepsilon)$ (см. рис. 1):

$$\hat{y}_i = 2 - 5 * x_i + x_i^2 + \varepsilon_i, \ i = 1..n.$$

Конечно, на практике мы не знаем эту скрытую зависимость. Но по ее виду мы можем предположить ее структуру - какие базисные функции входят в нее.

Постулируем модель:

$$y_m(x) = a_0 + a_1 * x + a_2 * x^2.$$

Найдем ее коэффициенты методом наименьших квадратов, составив нормальное уравнение.

Для этого:

1. Сформируем матрицу значений базисных функций X. Модель содержит три базисные функции. Поэтому матрица Вандермонда будет иметь три столбца. Строк в ней будет столько,

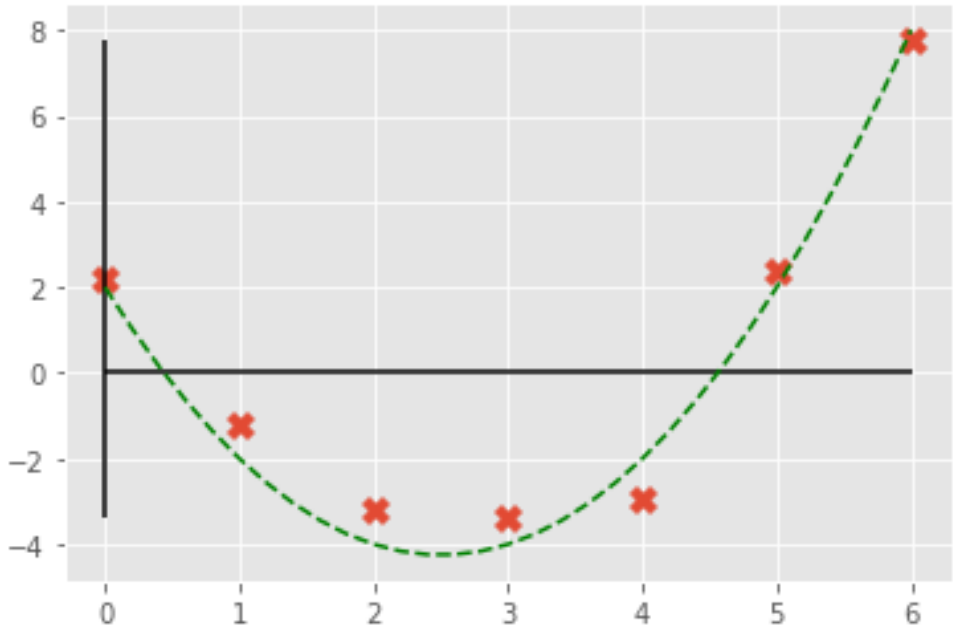


Рис. 1: Сгенерированные экспериментальные данные; пунктиром отмечена зависимость, использованная для генерации ([1])

сколько точек мы имеем:

$$Xm = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \end{bmatrix};$$

2. Вычислим матрицу Грама $\Gamma = X^T * X$ и правую часть нормальной системы уравнений $X^T * Y$:

$$\Gamma = \begin{bmatrix} 7 & 21 & 91 \\ 21 & 91 & 441 \\ 91 & 441 & 2275 \end{bmatrix};$$

$$X^T * Y = \begin{pmatrix} 1.6 \\ 29 \\ 247.6 \end{pmatrix},$$

3. решим нормальную систему уравнений:

$$X^T * X * \bar{a} = X^T * Y$$

$$\bar{a} = \begin{pmatrix} 2.49 \\ -4.96 \\ 0.97 \end{pmatrix};$$

4. проанализируем полученную линейную регрессию; вычислим значения регрессионной зависимости в узлах

$$Ym = Xm * \bar{a}$$

оценим критерии ошибки модели MSE и MAE

$$err = Y - Ym$$

Оценим среднеквадратическую ошибку (Mean Square Error)

$$MSE = \sum_{i=0}^{n} (err_i)^2)/len(err)$$

MSE=0.256

Оценим среднюю абсолютную ошибку (Mean Absolute Error)

$$MAE = sum(np.abs(err))/len(err)$$

MAE=0.406

5. построенную модель $f_m(a, x)$ теперь можно использовать для прогноза на новых данных. например, можно построить график модельной функции и сравнить его с графиком исходной функции, использованной для генерации точек (см. рис. 2)

При построении линейных регрессионных зависимостей с большим количеством базисных функций и большой выборкой матрица нормальной системы может быть плохо обусловленной, так как $cond(\Gamma) = cond(X) * cond(X)$. Поэтому на практике эти системы уравнений не решают, а используют методы оптимизации для решения задачи минимизации квадрата ошибки:

$$||X * \bar{a} - Y||_2^2 \to min.$$

Однако и у этого метода могут быть проблемы в случае большого количества данных. Оптимизируемая функция может содержать большое количество локальных минимумов и седловых точек и метод может застрять в одной из них.

На помощь приходит регуляризация, добавление дополнительного слагаемого в минимизируемую функцию.

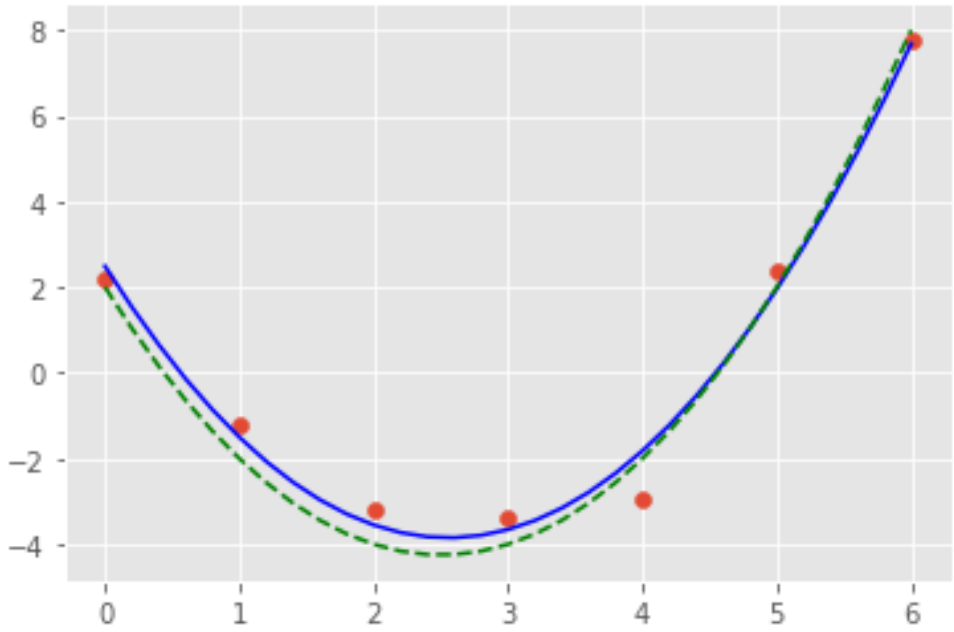


Рис. 2: Сравнение регрессионной зависимости, построенной по точкам (синяя сплошная) и зависимости, использованной для генерации данных (зеленым пунктиром)

4 Пример неустойчивости метода НК

Посмотрим как небольшие отклонения от истинной кривой растворимости, связанные с точностью измерений, могут повлиять на построение модели.

Пример 4.1 (Восстановление состава смеси по её кривой растворимости)**.** *Кривая растворимости смеси из (m+1) компонентов подчиняется модели вида:*

$$y(t) = a_0 * exp(-k_0 * t) + a_1 * exp(-k_1 * t) + ... + a_m * exp(-k_m * t)$$

где k_i - это константа растворимости i-го компонента.

Смоделируем растворение смеси из 5 компонентов. Зададим их константы растворимости:
kvect = np.array([0.9, 0.6, 0.4, 0.2, 0.1])
и сформируем 5 базисных функций с этими k - константами растворимости

$$exp(-k_0 * t), exp(-k_1 * t), exp(-k_2 * t), exp(-k_3 * t), exp(-k_4 * t)$$

Задача: по известным данным об оставшейся массе смеси (концентрации) в растворе определить/восстановить состав смеси.

Сгенерируем данные на основе следующего состава веществ

$$mvect = np.array([3, 2, 5, 0.0, 0.0]),$$

т.е. двух последних компонентов нет в смеси, а массы остальных составляют 3,2 и 5 грамм.

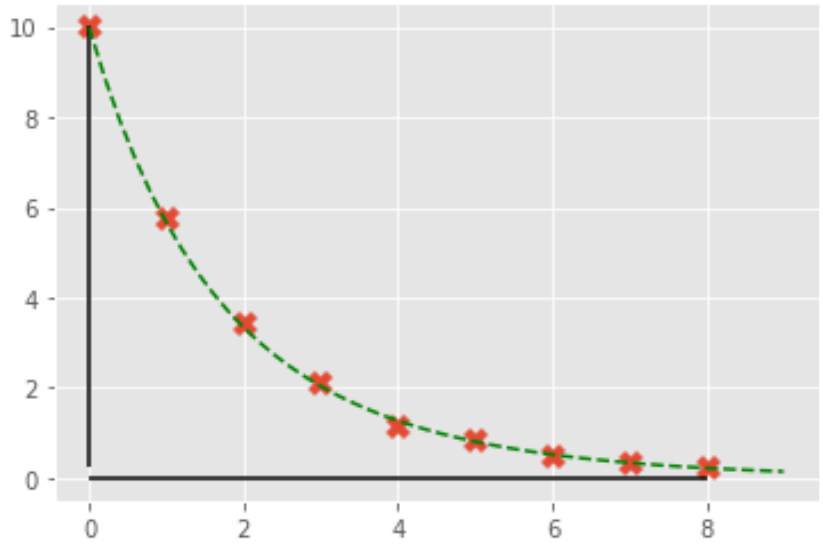


Рис. 3: Изменение массы смеси во времени; точки обозначают измеренные нами экспериментальные данные

Сгенерируем данные об изменении массы смеси в растворе в заисимости от времени (измерения делаем каждую минуту) с небольшими отклонениями (см. рис. 3)

Попробуем восстановить модель:

$$y_m(t) = a_0 * exp(-k_0 * t) + a_1 * exp(-k_1 * t) + ... + a_4 * exp(-k_4 * t)$$

Найдем ее коэффициенты методом НК. Результат будет такой:

Реальные массы компонент: [3. 2. 5. 0. 0.]

Полученные массы компонент: [4.27 -5.12 15.22 -6.6 2.25]

Так как масса не может быть отрицательной, то обнуляем отрицательные массы:

acoefs[acoefs < 0] $= 0$

Проанализируем полученную линейную регрессию (см. рис. 4):

MSE=32.073, MAE=4.632.

Видим, что небольшие ошибки наблюдения могут привести к совершенно неприемлемому результату.

Попробуем регуляризировать задачу - сформулируем доп. ограничение.

5 L2-регуляризация

Мы видели из примера, что небольшие ошибки в значениях выходного параметра приводят к резким изменениям коэффициентов модели и к неоправданному росту их значений.

Введем штраф за увеличение абсолютных величин параметров модели - нормы вектора $\bar{a}$ и добавим его к минимизируемой функции:

(МНК + L_2) $||X * \bar{a} - Y||_2^2 + \gamma * ||\bar{a}||^2 \to min.$

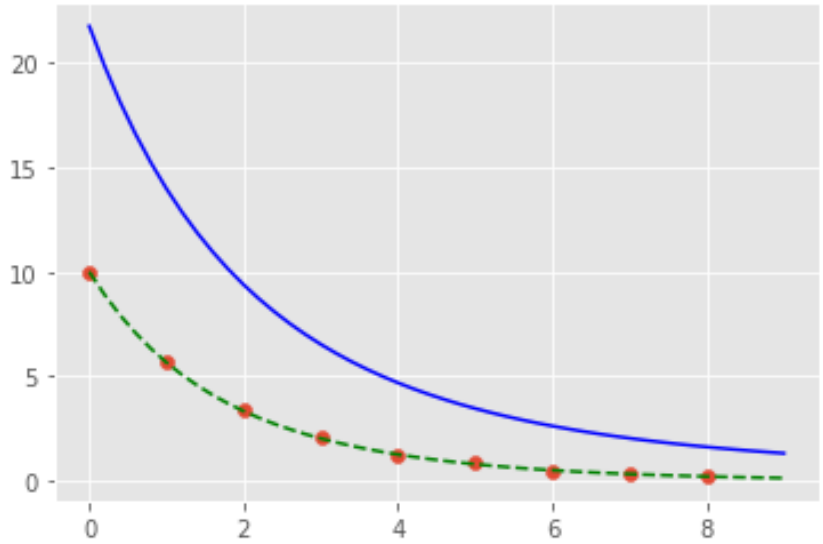


Рис. 4: Сравнение истинной (зеленый пунктир) и найденной регрессионной зависимости с использованием МНК (синяя сплошная)

Если мы продифференцируем данную функцию по $\bar{a}$ и, приравняем к нулевому вектору, то получим такую систему уравнений, которая равносильна задаче (МНК+L2):

$(NormalSystem + L2)$ $(X^T * X + \gamma * E_m) * \bar{a} = X^T * Y$

———

Здесь E_m - это единичная матрица размера m*m, где m - количество базисных функций (коэффициентов) модели.

Добавление к ковариационной матрице $X^T * X$ единичной матрицы, умноженной на *параметр регуляризации* γ формирует диагональ матрицы коэффициентов нормальной системы или "гребень"матрицы, что улучшает устойчивость решения. Поэтому такой метод иногда называется **"гребневой регрессией"**. А такой способ повышения устойчивости задачи по отношению к погрешности исходных данных называется L_2 - регуляризацией (так как добавляется квадрат нормы).

Вернемся к задаче из примера 4.1 и попробуем для её решения применить гребневую регрессию.

При значении параметр регуляризации $\gamma = 0.05$ получим следующий результат:

Реальные массы компонент: [3. 2. 5. 0. 0.]

Полученные массы компонент: [2.93689911 3.06835241 2.79970786 1.42824622 -0.3146425]

Обнулим отрицательную массу последнего компонента и оценим точность полученной модели:

$$MSE = 0.060, MAE = 0.238$$

Видим, что в этом случае мы имеем более реальный результат (см. рис. 5).

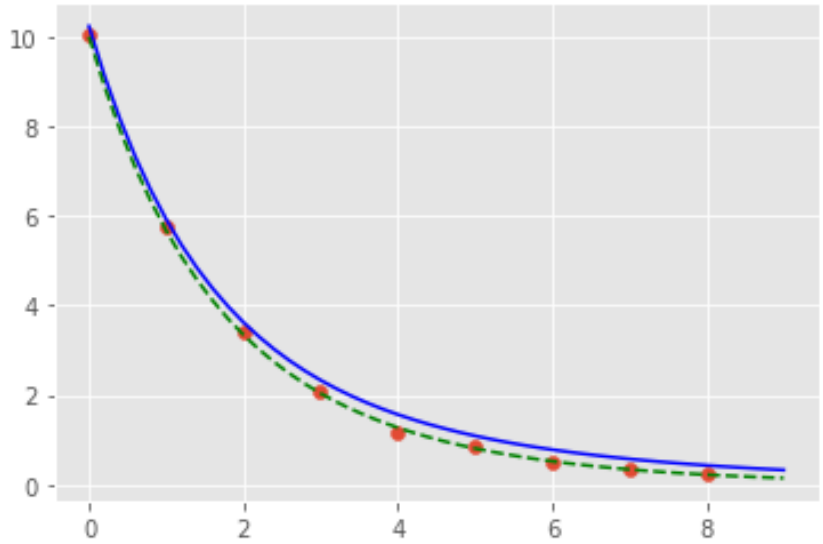


Рис. 5: Сравнение истинной (зеленый пунктир) и вычисленной регрессионной зависимости с использованием L2-регуляризации (синяя сплошная)

6 L1-регуляризация

Рассмотрим другой метод регуляризации - метод Лассо (LASSO - Least Absolute Shrinkage and Selection Operator) или L_1 - регуляризацию. Данный вид регуляризации позволяет регулировать состав базисных функций - обнуляются те коэффициенты модели, базсиные функции которых не имеют достаточного влияния на результат.

Введем ограничение на абсолютные величины коэффициентов модели:

$$\begin{cases} ||X * \bar{a} - Y||_2^2 \to min; \\ \sum\limits_{j=1}^{m} |a_j| \leqslant \sigma; \end{cases}$$

Можно считать, что этой задаче эквивалентна задача поиска минимума критерия (при соответствующем параметре регуляризации α):

(МНК + L_1) $||X * \bar{a} - Y||_2^2 + \alpha * ||\bar{a}||_1 \to min$.

Поэтому такой способ регуляризации называют L_1 - регуляризацией.

Интересно, что этот способ регуляризации приводит к отбору признаков - стурктуры модели, т.е. при уменьшении параметра σ все больше коэффициентов модели обнуляется, т.е. из модели убираются соответствующие базисные функции.

Продемонстрируем это на нашем примере.

При значении параметра регуляризации $\sigma = 0.001$ получим следующий результат:

Реальные массы компонент: [3. 2. 5. 0. 0.]

Полученные массы компонент: [1.22 5.53 2.93 0.32 0.]

Впервые мы получили результат без отрицательных масс. Оценим точность полученной модели:

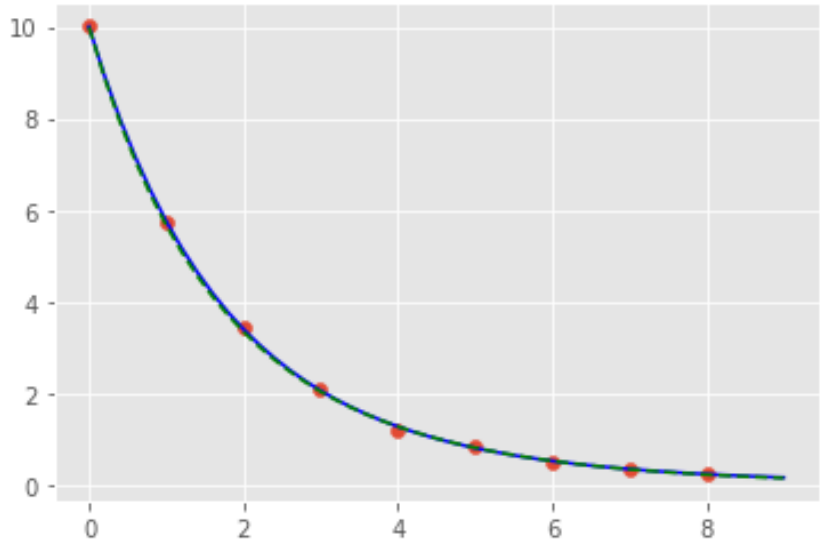


Рис. 6: Сравнение истинной (зеленый пунктир) и вычисленной регрессионной зависимости с использованием L1-регуляризации (синяя сплошная)

$$MSE = 0.002, MAE = 0.03$$

Но также видим, что несмотря на хорошее приближение (см. рис. 6) структура модели отличается от реальной и найденные составы компонентов далеки от реальных. Тем не менее данный метод регуляризации может дать хорошие результаты, если мы увеличим количество экмпериментальных данных (будем делать пробы каждые полминуты, а не каждую минуту) или улучшим точность оценки концентрации.

Более подробный материал вы можете найти в открытом онлайн-курсе [1].

7 Вопросы для самоконтроля и контроля

1. Какие данные необходимы для обучения регресионной модели? Опишите модель данных.
2. Какой вид имеет линейная регрессионная зависимость?
3. Какой критерий оптимизации (качества аппроксимирующей функции) используется при подборе коэффициентов в методе наименьших квадратов (МНК)?
4. С какими проблемами можно столкнуться при использовании МНК?
5. Что такое L2-регуляризация?
6. Какой метод построения регрессионной зависимости называется гребневым?
7. Что такое L1-регуляризация? Чем она отличается от L2-регуляризации?

8. Дайте пример задачи построения линейной регрессии.

9. Дайте пример решения задачи построения линейной регрессии с помощью МНК.

8 Практические задания

Задача 8.1 (Построение регрессионной зависимости по МНК)**.** Пусть имеются следующие данные наблюдения переменных (x,y):

(-1.0, 0.0), (0.0, 0.0), (1.0, 1.0), (2.0, 4.0).

Мы хотим построить линейную регрессию вида

$$y_M(x, \bar{a}) = a_0 + a_1 * x^2$$

с использованием критерия наименьших квадратов.

- Составьте и решите нормальную систему уравнений, найдите оптимальный вектор $\bar{a} = (a_0, a_1)$;
- Вычислите среднеквадратическую ошибку **MSE**;
- Вычислите среднюю абсолютную ошибку **MAE**.

Задача 8.2 (Построение регрессионной зависимости с использованием L2-регуляризации)**.** Пусть имеются следующие данные наблюдения переменных (x,y):

(-1.0, 0.0), (0.0, 0.0), (1.0, 1.0), (2.0, 4.0).

Мы хотим построить линейную регрессию вида

$$y_M(x, \bar{a}) = a_0 + a_1 * x^2$$

с использованием критерия наименьших квадратов.

- Составьте и решите нормальную систему уравнений для гребневой регрессии с параметром $\gamma = 0.5$, найдите вектор $\bar{a} = (a_0, a_1)$;
- Вычислите среднеквадратическую ошибку **MSE**;
- Вычислите среднюю абсолютную ошибку **MAE**.

Список источников

[1] Машинное обучение и нейросетевой анализ данных в Python [Электронный ресурс] https://mooped.net/course/view.php?id=393 (Дата обращения: 09.02.2022)

[2] Машинное обучение (курс лекций, К.В.Воронцов) [Электронный ресурс] http://www.machinelearning.ru/wiki/index.php (Дата обращения: 09.02.2022)

[3] Открытый курс машинного обучения [Электронный ресурс] https://ods.ai/tracks/open-ml-course (Дата обращения: 09.02.2022)

Лекция 3. "Классификация объектов"

- **Постановка задачи классификации**
- Оценка качества классификации.
- Метод kNN - k ближайших соседей
- Логистическая регрессия
- Пример классификации обучающихся на онлайн-курсе

Постановка задачи классификации

==========

Линейные регрессии могут использоваться и для решений задачи классификации объектов. Но в этом случае гиперплоскость строится так, чтобы отделить один класс объектов от объектов других классов, провести границу между объектами разных классов.

Начнем с более простой задачи - бинарной классификации

1. Бинарная классификация. Линейно-разделимые классы

Пусть имеются объекты, которые разделяются на два класса. Одному классу мы ставим в соответствие метку "0", другому классу - метку "1". Например, все письма можно разделить на два класса спам (метка "1") и не спам ("0"). Мы хотим построить автоматический классификатор, который по векторному представлению объекта (письма) определяет - к какому классу он принадлежит.

Нам дана некоторая выборка данных - для заданного набора точек

$$x^{(i)} = (x_1^{(i)}, x_2^{(i)}, .., x_m^{(i)}), i = 1..n$$

имеются соответствующие им значения $y_i \in \{0, 1\}$, которые показывают к какому классу принадлежит i-й объект - к классу "нулей" или классу "единиц":

$$y_i = \begin{cases} 1, & \text{если } class(\bar{x}^{(i)}) = 1 \\ 0, & \text{если } class(\bar{x}^{(i)}) = 0 \end{cases}$$

Нам надо построить классификатор, функцию $h(\bar{x})$, такую, что для любого объекта $\bar{x}$, в том числе и не входящего в исходную выборку, определяет к какому классу он принадлежит:

$$h(\bar{x}) = \begin{cases} 1, & \text{если } class(\bar{x}) = 1 \\ 0, & \text{если } class(\bar{x}) = 0 \end{cases}$$

Метрикой качества такой функции может быть точность классификации - % правильно классифицированных объектов (accuracy).

в дальнейшем входной вектор мы будем обозначать просто x вместо $\bar{x}$.

2. Линейная регрессия как классификатор

Линейную регрессию тоже можно использовать как классификатор.

Напоминаем, что линейная регрессия предлагает строить модель по имеющимся данным в виде:

$$u(x, W) = \bar{a}^T * \bar{x} + b = (b, \bar{a}^T) * \begin{pmatrix} 1 \\ \bar{x} \end{pmatrix} = \bar{W}^T * \bar{X}$$

При этом вектор коэффициентов $\bar{W}$ выбирают из условия минимизации среднего квадрата отклонений MSE(Mean of Square Errors):

$(MSE)\ \Phi(W) = \frac{1}{n} * \sum_{i=1..n}(y_i - u(x^{(i)}, W))^2 \to min$

Для разделения двух классов мы можем использовать и линейную регрессию в качестве пороговой функции:

$$\begin{cases} class(x) = 0 & \text{если } u(x, W) < 0 \\ class(x) = 1 & \text{если } u(x, W) > 0 \end{cases}$$

Но проблема линейной регрессии заключается в том, что она будет сильно реагировать на удаленные от границы объекты и передвигать ее поближе к ним, несмотря на ухудшение качества классификации.

Продемонстрируем это на примере.

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

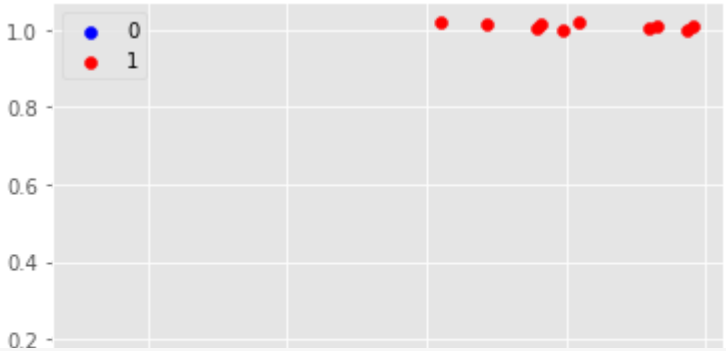
3. Пример использования линейной регресси в качестве классификатора

Сгенерируем два множества точек, линейно-отделимых друг от друга.

```
np.random.seed(42)
err_y = 0.02
# сгенерируем множество "нулей" как нормальное распределение N(m,s)
n0 = 10; m = -1; s = 1
x0 = np.random.randn(n0) * s + m
t0 = err_y*(np.random.random(n0) - 0.5)
y0 = np.zeros(n0)

# сгенерируем множество "единиц" - точек "внутри" отрезка ab
n1 = 10; a = 1; b = 3
par1 = np.random.random(n1)
t1 =  err_y*np.random.random(n1) + 1.0
x1 = par1*a + (1-par1)*b
y1 = np.ones(n1)
y1.shape
```

```
(10,)
```

```
# отобразим графически расположение точек
plt.scatter(x0, t0, c='b', label='0')
plt.scatter(x1, t1, c='r', label='1')
plt.legend();
```

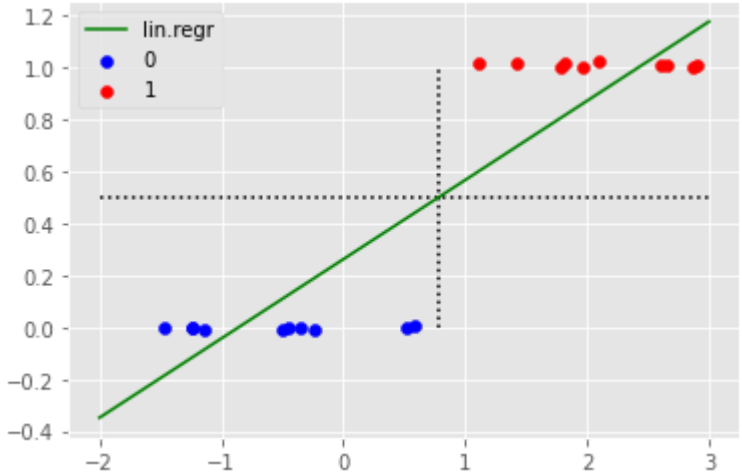



```
from sklearn.linear_model import LinearRegression
linear_regression = LinearRegression( )

# сформируем выборку и обучим линейный классификатор:
X = np.hstack([x0,x1]).reshape(-1,1)
X.shape
y = np.hstack([y0,y1])
labels = np.hstack([y0,y1])
y.shape

model = linear_regression.fit(X,y)
print('смещение:', model.intercept_)
print('коэфф-ты:', model.coef_)
```

```
смещение: 0.260834946900754
коэфф-ты: [0.30450282]
```

```
# нарисуем границу линию регрессии ym = a*x + b
b = model.intercept_; a = model.coef_[0]
xline_0 = np.arange(-2, 3.1, 0.1)
xline_1 = a*xline_0 + b
plt.plot(xline_0, xline_1, 'g', label='lin.regr')

plt.hlines(y=0.5, xmin=-2, xmax=3, linestyles='dotted')
plt.vlines(x=(0.5-b)/a, ymin=0, ymax=1, linestyles='dotted')

# оборазим графически расположение точек
plt.scatter(x0, t0, c='b', label='0')
```

```
plt.scatter(x1, t1, c='r', label='1')
plt.legend();
```




```
# прогноз и точность классификации
classes = (model.predict(X) > 0.5)*1
print(classes)
print('Точность классификации= {}%'.format(sum(classes == labels)*100/len(classes)))
```

```
[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
Точность классификации= 100.0%
```

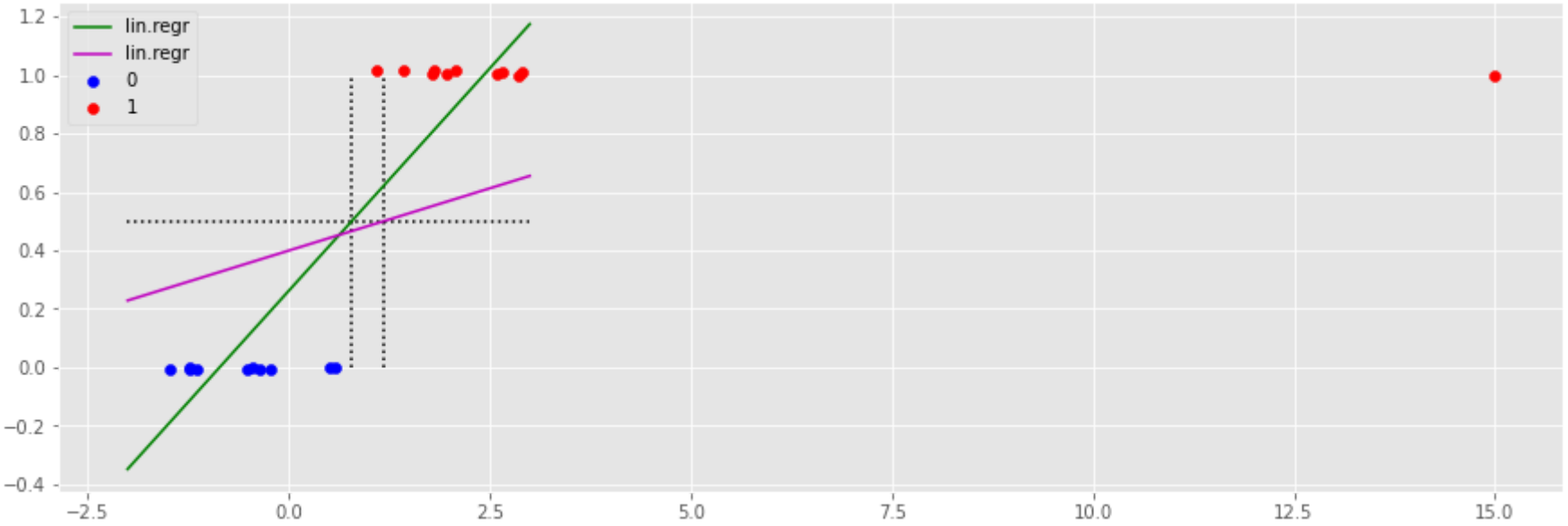
Добавим одну удаленную точку и посмотрим как изменится граница

```
X_new = np.vstack([X, 15])
y_new = np.hstack([y, 1])
t1_new = np.hstack([t1, 1])
x1_new = np.hstack([x1, 15])

y_new.shape, X_new.shape
```

```
((21,), (21, 1))
```

```
# построим модель
model_1 = linear_regression.fit(X_new,y_new)
```

```
# нарисуем границу линию регрессии ym = a*x + b
plt.figure(figsize=(15, 5))
b1 = model_1.intercept_; a1 = model_1.coef_[0]
xline1_0 = np.arange(-2, 3.1, 0.1)
xline1_1 = a1*xline_0 + b1
plt.plot(xline_0, xline_1, 'g', label='lin.regr')
plt.plot(xline1_0, xline1_1, 'm', label='lin.regr')

plt.hlines(y=0.5, xmin=-2, xmax=3, linestyles='dotted')
plt.vlines(x=(0.5-b1)/a1, ymin=0, ymax=1, linestyles='dotted')
plt.vlines(x=(0.5-b)/a, ymin=0, ymax=1, linestyles='dotted')

# оборазим графически расположение точек
plt.scatter(x0, t0, c='b', label='0')
plt.scatter(x1_new, t1_new, c='r', label='1')

plt.legend();
```




```
# прогноз и точность классификации
from sklearn import metrics
```

```
classes = (model_1.predict(X_new) > 0.5)*1
print(classes)
print('Точность классификации= {:5.2f}%'.format(100*metrics.accuracy_score(y_new, classes)))
```

```
[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1]
Точность классификации= 95.24%
```

ВЫВОДЫ

- задача классификации объектов отличается от задачи построения линейной регрессии тем, что линия строится как линия отделения объектов одного класса от всех остальных;
- линейная регрессия может использоваться как пороговая функция в бинарной классификации:

$$\begin{cases} class(x) = 0 & \text{если } u(x, W) < 0 \\ class(x) = 1 & \text{если } u(x, W) > 0 \end{cases}$$

- линейная регрессия не устойчива при решении задачи классификации - наиболее удаленная точка с координатами может сильно сместить границу и ухудшить точность классификации.

Выход - построение логистической регрессии

Лекция 3. "Классификация объектов"

- Постановка задачи классификации
- **Оценка качества классификации**
- Метод kNN - k ближайших соседей
- Логистическая регрессия
- Пример классификации обучающихся на онлайн-курсе

Оценка качества классификации

==========

Как правило, основной метрикой качества работы алгоритма классификации является его точность (accuracy), т.е. % правильно классифицированных объектов из выборки.

Но:

- есть много нюансов даже с применением метрики точность (accuracy); она может плохо работать на несбалансированных выборках;
- метод может переобучиться на основной выборке и показывать плохие результаты на новых объектах;
- в зависимости от специфики задачи нам иногда важнее не точность, а чтобы наш классификатор не выбросил в спам важное для нас письмо, пусть даже при этом он пропустит несколько спамных писем в основной ящик;

Поэтому надо, во-первых, знать приемы оценки точности классификации и знать несколько разных метрик оценки точности. Познакомимся с ними.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

plt.style.use('ggplot')
```

1. Матрица (не)соответствия (Confusion matrix)

Пусть мы сравниваем каких-либо два метода на какой-то задаче классификации и мы знаем результаты работы данных методов ('lab1', 'lab2) и истинные метки объектов ('target')

Проанализируем результаты работы методов и сравним их между собой

```
target = [0,0,0,0,0,0,0, 1,1,1]
lab1 =   [0,0,0,1,0,1,1, 1,1,1]
lab2 =   [0,0,0,0,0, 0,0,0,0,1]

df = pd.DataFrame(np.array([target, lab1, lab2]).T, columns=['target','lab1','lab2'])
df
```

	target	lab1	lab2
0	0	0	0
1	0	0	0
2	0	0	0
3	0	1	0
4	0	0	0
5	0	1	0
6	0	1	0
7	1	1	0
8	1	1	0
9	1	1	1

рассмотрим следущие понятия:

TP (True Positive) - это количество правильно распознанных объектов как принадлежащих классу "единичек";

TN (True Negative) - это количество правильно распознанных объектов как принадлежащих классу "нулей";

FP (False Positive) - это количество неправильно распознанных объектов ("нулей") как "единичек";

FN (False Negative) - это количество неправильно распознанных объектов ("единичек") как "нулей";

```
true_positive = sum(df.lab1[df.target == 1] == 1)
print('true_positive=', true_positive)

true_negative = sum(df.lab1[df.target == 0] == 0)
print('true_negative=', true_negative)

false_positive = sum(df.lab1[df.target == 0] == 1)
print('false_positive=', false_positive)
```

```
false_negative = sum(df.lab1[df.target == 1] == 0)
print('false_negative=', false_negative)
```

```
true_positive= 3
true_negative= 4
false_positive= 3
false_negative= 0
```

```
from sklearn.metrics import confusion_matrix

conf_mat1 = confusion_matrix(df.target, df.lab1)
print(conf_mat1)

conf_mat2 = confusion_matrix(df.target, df.lab2)
print(conf_mat2)

true_negative, false_positive, false_negative, true_positive = tuple(conf_mat.flatten().ast
```

```
[[4 3]
 [0 3]]
[[7 0]
 [2 1]]
```

2. Метрики точности

Теперь мы можем сформулировать основные метрики, используемые при оценке точности классификации

2.1. Accuracy

Самая простая и понятная метрика - это доля правильных ответов алгоритма:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

```
from sklearn.metrics import accuracy_score

acc1 = accuracy_score(df.target, df.lab1)
acc2 = accuracy_score(df.target, df.lab2)
print('Точность методов: ', acc1, acc2)
```

```
Точность методов:  0.7 0.8
```

2.2. Precision & Recall

представим себе, что объекты класса 1 - это зловредные вирусы, которые могут разрушить вашу файловую систему. А анитивирусные программы - классификаторы, которые определяют можно запустить данную программу или нет. Тогда какой вы выберете антивирус - который пропустит 2 из 3-х вирусов или который все вирусы обнаружит, но также и некоторые хорошие программы не даст запустить?

Или вам более важно, чтобы метод поменьше "врал" насчет того, что данный объект "единичка"?

- доля правильно распознанных "единичек":

$$recall = \frac{TP}{TP + FN}$$

- доля корректного обнаружения "единичек":

$$precision = \frac{TP}{TP + FP}$$

```
from sklearn.metrics import precision_score, recall_score

recall1 = recall_score(df.target, df.lab1)
recall2 = recall_score(df.target, df.lab2)
print('доля правильно распознанных "единичек" разными методами: ', recall1, round(recall2,2

precision1 = precision_score(df.target, df.lab1)
precision2 = precision_score(df.target, df.lab2)
print('доля корректного обнаружения "единичек" разными методами: ', precision1, precision2)
```

```
доля правильно распознанных "единичек" разными методами:  1.0 0.33
доля корректного обнаружения "единичек" разными методами:  0.5 1.0
```

2.3. F1 - метрика

- **F1** - это метрика, которая объединяет обе метрики - и precision и recall (гармоническое среднее):

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

```
from sklearn.metrics import f1_score

f1_1 = f1_score(df.target, df.lab1)
f1_2 = f1_score(df.target, df.lab2)
print('F1 у разных методов: ', round(f1_1,2), f1_2)
```

```
F1 у разных методов:  0.67 0.5
```

2.4. ROC AUC

Также часто используется для оценки качества работы классификатора метрика ROC AUC.

Идея метрики такая - пронумеровать все объекты, например, в порядке роста номеров классов вдоль оси OX и проходя по всем объектам откладывать на плоскости значения *true_positive_rate* против *false_positive_rate*, соединяя их линией.

$$true\_positive\_rate = \frac{TP}{TP + FN}$$

$$false\_positive\_rate = \frac{FP}{FP + TN}$$

Площадь под образованной кривой и даст нам значение данного критерия. Если на всех объектах мы не будем ошибаться, то $false\_positive\_rate = 0$ и кривая резко поднимется вверх до 1 и площадь под нею будет = 1.

Если же мы с вероятностью 0.5 будем определять правильный ответ, то площадь под кривой и будет равна 0.5. И смысла в применении такого метода классификации нет.

```
from sklearn.metrics import roc_auc_score

roc_auc_1 = roc_auc_score(df.target, df.lab1)
roc_auc_2 = roc_auc_score(df.target, df.lab2)
print('ROC_AUC у разных методов: ', round(roc_auc_1,2), round(roc_auc_2,2))
```

```
ROC_AUC у разных методов:  0.79 0.67
```

3. Измеряем точность на тестовой выборке

Если вы думаете, что хорошей мыслью является взять все данные и обучить на них классификатор, то вы серьезно ошибаетесь. Есть много примеров, когда классификатор показывает замечательную точность на обучающей выборке и плохую точность на новых данных.

Поэтому, чтобы правильно подобрать параметры классификаторов, необходимо оценивать его точность на тестовой выборке, не используемой при обучении.

```
from sklearn.datasets import load_breast_cancer

X, y = load_breast_cancer(return_X_y=True)
counts = pd.value_counts(y)
print(counts)
```

```
1    357
```

```
0    212
dtype: int64
```

```
# импортируем функцию деления выборки и классификаторы
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

Разбиваем исходную выборку на две подвыборки - обучающую и тестовую.

Проверим точность работы классификаторов по обучающей выборке и по тестовой.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3) #,
X_train.shape

logistic_regression = LogisticRegression(random_state=42, max_iter=1000)
kneighbors_classifier = KNeighborsClassifier()

logistic_regression.fit(X_train, y_train)
kneighbors_classifier.fit(X_train, y_train)

acc_1 = accuracy_score(y_train, logistic_regression.predict(X_train))
acc_2 = accuracy_score(y_train, kneighbors_classifier.predict(X_train))

F1_1 = f1_score(y_train, logistic_regression.predict(X_train))
F1_2 = f1_score(y_train, kneighbors_classifier.predict(X_train))

print('-------- Train Data -----------')
print('train accuracy for logistic=', round(acc_1, 3))
print('train accuracy for kneighbors=', round(acc_2, 3))

print('train F1 for logistic=', round(F1_1, 3))
print('train F1 for kneighbors=', round(F1_2, 3))

acc_3 = accuracy_score(y_test, logistic_regression.predict(X_test))
acc_4 = accuracy_score(y_test, kneighbors_classifier.predict(X_test))

F1_3 = f1_score(y_test, logistic_regression.predict(X_test))
F1_4 = f1_score(y_test, kneighbors_classifier.predict(X_test))

print('\n -------- Test Data -----------')
print('test accuracy for logistic=', round(acc_3, 3))
print('test accuracy for kneighbors=', round(acc_4, 3))

print('test F1 for logistic=', round(F1_3, 3))
print('test F1 for kneighbors=', round(F1_4, 3))
print()
```

```
-------- Train Data -----------
```

```
train accuracy for logistic= 0.955
train accuracy for kneighbors= 0.942
train F1 for logistic= 0.964
train F1 for kneighbors= 0.954

 -------- Test Data -----------
test accuracy for logistic= 0.942
test accuracy for kneighbors= 0.924
test F1 for logistic= 0.955
test F1 for kneighbors= 0.941

/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: Convergenc
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

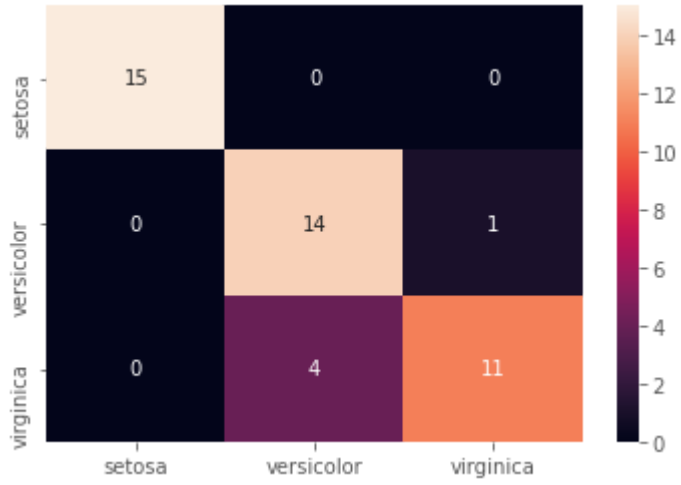
Случай мульти-классификации

```
from sklearn import datasets
import seaborn as sns

iris = datasets.load_iris()

iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df['Species'] = np.array([iris.target_names[cls] for cls in iris.target])
sns.pairplot(iris_df, hue='Species');
```


```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)

x_train, x_test, y_train, y_test = train_test_split(
    iris.data, iris.target,
    test_size=0.3, stratify= iris.target, random_state=42
)
rf_model = random_forest.fit(x_train, y_train)
predictions = rf_model.predict(x_test)
print('Accuracy: {:.2f}'.format(accuracy_score(y_test, predictions)))
```

```
Accuracy: 0.89
```

```
confusion_scores = confusion_matrix(y_test, predictions)
confusion_df = pd.DataFrame(confusion_scores, columns=iris.target_names, index= iris.target
sns.heatmap(confusion_df, annot=True);
```



ВЫВОДЫ

- $accuracy$ - доля правильно расклассифицированных объектов - это самый простой и понятный показатель точности классификации;
- если выборка не сбалансирована, то $accuracy$ - может оказаться плохой идеей;
- в зависимости от специфики задачи могут быть важнее такие показатели как **precision** - точность классификатора на одном классе или **recall** - доля объектов какого-то класса, правильно распознанных;
- **F1** и **ROC_AUC** - используемые на практике интегральные показатели качества классификации; с одним минусом -они плохо интерпретируемы;
- для выбора классификатора или параметров классификатора следует качество его работы оценивать на отдельной (тестовой) выборке, которая не использовалась при обучении модели.

Более подробно с разными метриками вы можете познакомиться здесь: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Лекция 3. "Классификация объектов"

- Постановка задачи классификации
- Оценка качества классификации.
- **Метод kNN - k ближайших соседей**
- Логистическая регрессия
- Пример классификации обучающихся на онлайн-курсе

Метод kNN - k ближайших соседей

==========

Рассмотрим очень простой метод классификации - метод k ближайших соседей. Метод относится к метрическим алгоритмам классификации, основанные на измерении и использовании расстояний между объектами.

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

1. Идея МЕТОДа k-БЛИЖАЙШИХ СОСЕДЕЙ

Цель метода k-ближайших соседей — классифицировать объекты на основе их сходства (например, функции расстояния).

Метод k-ближайших соседей не нуждается в предварительной фазе обучения и начинает классифицировать точки данных на основе большинства голосов их соседей.

- метод для новой не классифицированной точки x находит k ближайших соседей - ранее классифицированных точек $(x_i, y_i), i = 1..k,\ y_i \in \{1, .., m\}$; здесь m - это количество классов;
- Объект классифицируется в соответствии с большинством голосов ближайших соседей; т.е. если большинство соседей принадлежат классу j, то метод считает, что объект также принадлежит данному классу:

$$y(x) = arg\max_{j=1..m} \sum_{i=1..k} (y_i = j)$$

Рассмотрим простой пример бинарной классификации с помощью данного метода.

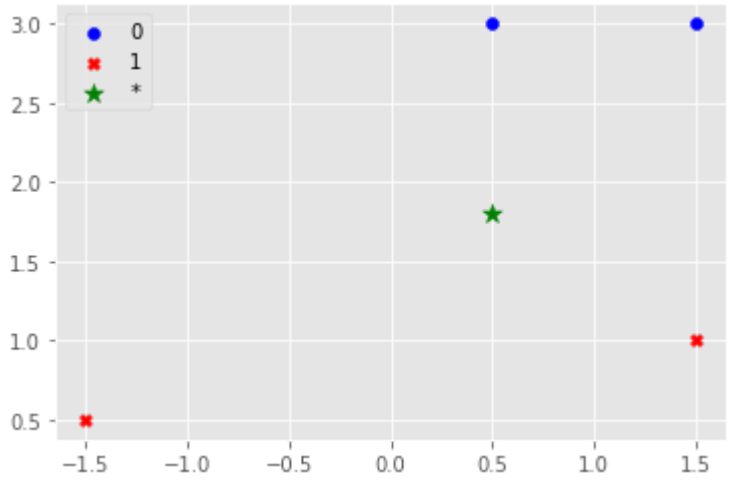
2. Пример использования метода kNN

Сгенерируем два множества точек.

```
np.random.seed(10)
# сгенерируем множество "нулей" как нормальное распределение N(m,s)
n0 = 10; m = np.array([0, 0]); s = 1
x0 = np.random.randn(n0, 2) * s #+ m
y0 = np.zeros(n0)

# сгенерируем множество "единиц" - точек "около" отрезка AB
n1 = 10; A = np.array([0, 2]); B = np.array([+2, 4])
t1 = np.random.random((n1,1))
t2 = (np.random.random((n1,2))-0.5)*1
x1 = t1*A + (1-t1)*B + t2
y1 = np.ones(n1)
```

Добавим новую точку $xnew$ и рассмотрим ее классификацию методом kNN при различным значениях k.

```
xnew =  np.array([0.5, 1.8])

# отобразим графически расположение точек
plt.scatter(x0.T[0], x0.T[1], c='b', marker='o', label='0')
plt.scatter(x1.T[0], x1.T[1], c='r', marker='X', label='1')
plt.scatter(xnew[:1], xnew[1:], c='g', marker='*', s=100, label='*')
plt.legend(loc= 'upper left');
```

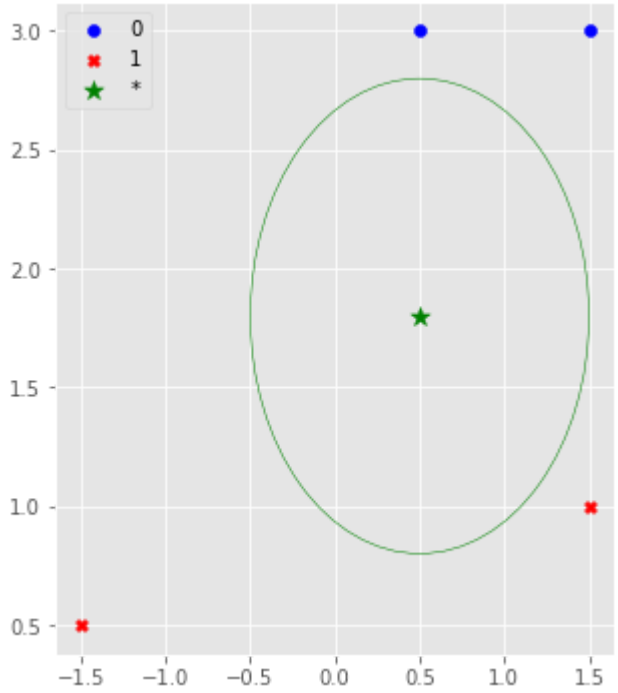


Рассмотрим задачу присвоения зеленой звездочки классу 0 или классу 1.

При $k = 1$ метод k-ближайших соседей присвоит зеленой звездочке класс 0, так как ближайшая точка к нему - синяя.

Увеличим количество ближайших соседей до k=3.

```
# отобразим графически расположение точек и нарисуем круг вокруг зел.точки
plt.figure(figsize=(5,6))
plt.scatter(x0.T[0], x0.T[1], c='b', marker='o', label='0')
plt.scatter(x1.T[0], x1.T[1], c='r', marker='X', label='1')
plt.scatter(xnew[:1], xnew[1:], c='g', marker='*', s=100, label='*')
circle=plt.Circle(xnew,1.0, color='g', fill=False)
plt.gcf().gca().add_artist(circle)
plt.legend(loc= 'upper left');
```



Как вы можете увидеть на рисунке: внутри круга находятся два объекта класса 1 и один объект класса 0. Метод k-ближайших соседей классифицирует зеленую звездочку как объект класса 1, поскольку они составляют большинство.

---

3. Применение метода kNN с использованием sklearn

Сгенерируем множество точек для классификации их методом kNN

```
np.random.seed(10)
# сгенерируем множество точек как нормальное распределение N(m,s)
```

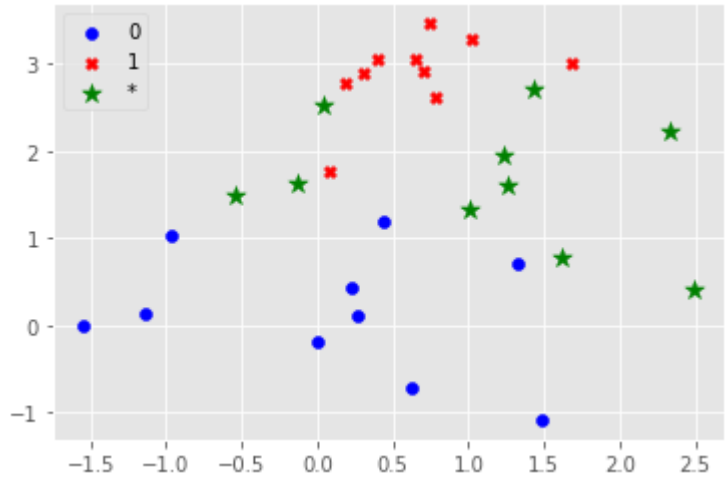
```
n = 10; mnew = np.array([1, 1.5]); snew = 1.0
xnew = mnew + snew*np.random.randn(n, 2)
xnew
```

```
array([[ 2.3315865 ,  2.21527897],
       [-0.54540029,  1.49161615],
       [ 1.62133597,  0.77991444],
       [ 1.26551159,  1.60854853],
       [ 1.00429143,  1.32539979],
       [ 1.43302619,  2.70303737],
       [ 0.03493433,  2.52827408],
       [ 1.22863013,  1.94513761],
       [-0.13660221,  1.63513688],
       [ 2.484537  ,  0.42019511]])
```

```
# сформируем выборку для обучения:
X = np.vstack([x0,x1])
y = np.hstack([y0,y1])
X.shape, y.shape
```

```
((20, 2), (20,))
```

```
# отобразим графически расположение точек
plt.scatter(x0.T[0], x0.T[1], c='b', marker='o', label='0')
plt.scatter(x1.T[0], x1.T[1], c='r', marker='X', label='1')
plt.scatter(xnew[:, 0], xnew[:, 1], c='g', marker='*', s=100, label='*')
plt.legend(loc= 'upper left');
```



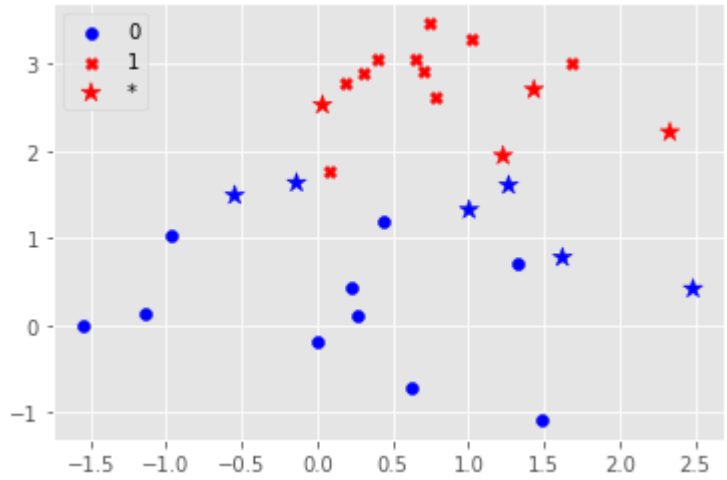
Применим метод при k=3

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X, y)
```

```
# прогноз и точность классификации
y_pred = model.predict(X)
score = metrics.accuracy_score(y, y_pred)

print('Точность классификации= {:5.2f}%'.format(100*score))
```

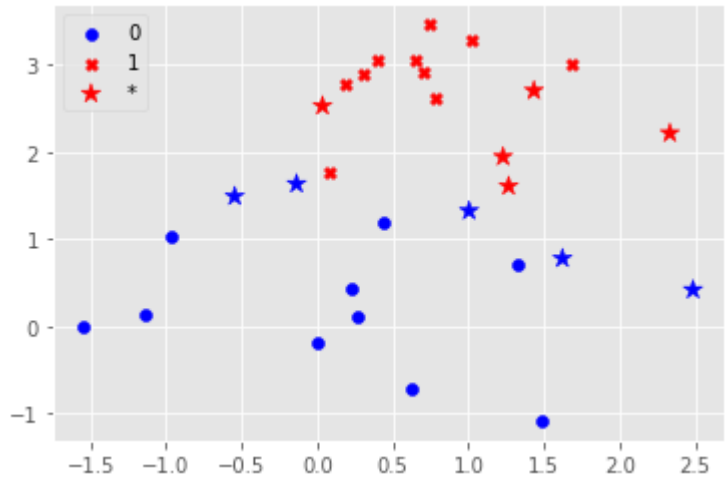
```
   Точность классификации= 100.00%
```

```
# сделаем прогноз для новых данных и оценим разметку графически
ynew = model.predict(xnew)
ycolors = {0:'b', 1:'r'}
yc_new = [ycolors[y] for y in ynew]
# отобразим графически расположение точек
plt.scatter(x0.T[0], x0.T[1], c='b', marker='o', label='0')
plt.scatter(x1.T[0], x1.T[1], c='r', marker='X', label='1')
plt.scatter(xnew[:, 0], xnew[:, 1], c=yc_new, marker='*', s=100, label='*')
plt.legend(loc= 'upper left');
```



Применим метод при k=5

```
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X, y)

# сделаем прогноз для новых данных и оценим разметку графически
ynew = model.predict(xnew)
ycolors = {0:'b', 1:'r'}
yc_new = [ycolors[y] for y in ynew]

# отобразим графически расположение точек
plt.scatter(x0.T[0], x0.T[1], c='b', marker='o', label='0')
plt.scatter(x1.T[0], x1.T[1], c='r', marker='X', label='1')
plt.scatter(xnew[:, 0], xnew[:, 1], c=yc_new, marker='*', s=100, label='*')
plt.legend(loc= 'upper left');
```



Если есть тестовая выборка, то можно подобрать параметр k, добиваясь наилучшей точности классификации на тестовой выборке.

ВЫВОДЫ

- kNN очень прост в реализации;
- kNN - интуитивно понятен, вывод на основе прецедентов, очень просто интерпретировать его решение;

НО:

- неустойчив к погрешностям (к шуму, выбросам);
- неоднозначность классификации при равенстве голосующих соседей за разные классы;
- низкое качество классификации;
- для классификации объекта приходится хранить и использовать всю выборку.

Лекция 3. "Классификация объектов"

- Постановка задачи классификации
- Оценка качества классификации.
- Метод kNN - k ближайших соседей
- **Логистическая регрессия**
- Пример классификации обучающихся на онлайн-курсе

Логистическая регрессия

==========

Начнем с более простой задачи - бинарной классификации

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

1. Логистическая регрессия как классификатор

Логистическая регрессия предлагает строить модель в том же виде, что и линейная регрессия, но есть два новшества:

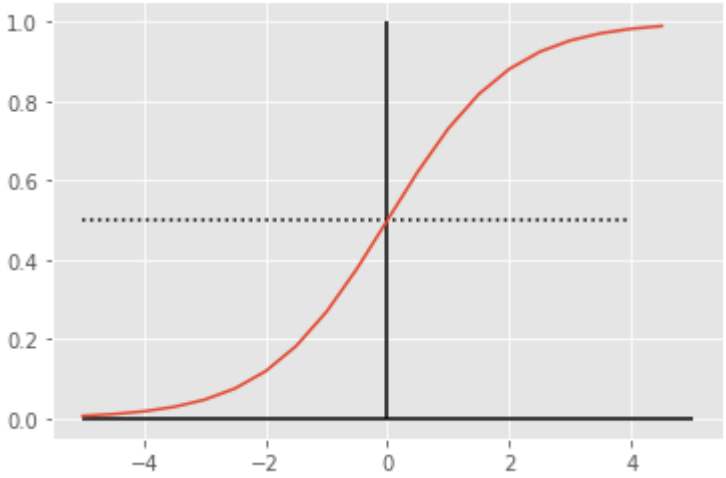
1. выход модели $y_m(x, W)$ получать, подавая выход линейной функции на вход логистической функции:

$$u = W^T * x,$$

$$y_M(x, W) = \frac{1}{1 + e^{-u}} = \frac{1}{1 + e^{-W^T * x}}$$

```
# логистическая функция
def sigmoid(u):
```

```
    return 1.0/(1.0 + np.exp(-u))

u_list = np.arange(-5, 5, 0.5)
plt.hlines(0, -5, 5); plt.vlines(0, 0, 1.0)
plt.hlines(y=0.5, xmin=-5, xmax=4, linestyles='dotted')
plt.plot(u_list, sigmoid(u_list));
```



и второе отличие:

2. вектор коэффициентов $\bar{W}$ выбирают из условия минимизации кросс-энтропии по формуле (для двух классов):

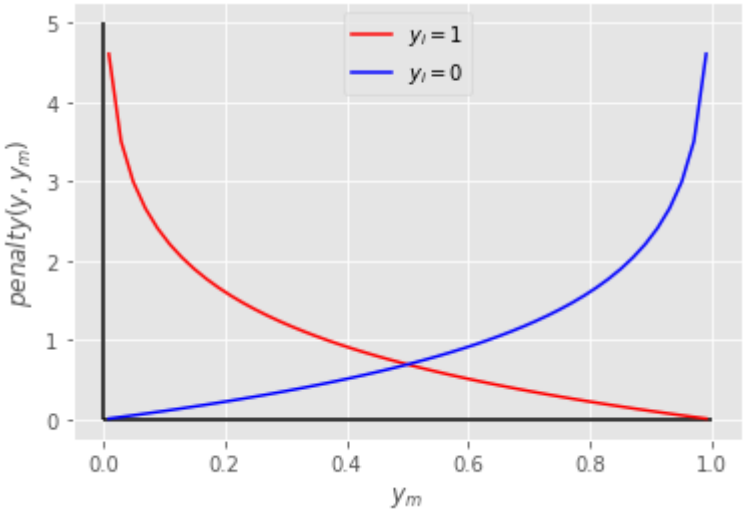
$$(BinCrossEntropy)\ \Phi_{CE}(W) = \frac{1}{n} * \sum_{i=1..n} -y_i * log(y_M(x^{(i)}, W)) - (1 - y_i) * log(1 - y_M(x^{(i)}, W)) \rightarrow min$$

Это связано с тем, что и значения y_i и значения $y_M(x^{(i)}, W)$ находятся на отрезке $[0, 1]$ и поэтому $(y_i - y_M(x^{(i)}, W))^2$ будет слабо меняться с изменением W.

Посмотрим как кросс-энтропийная функция функция потерь штрафует за ошибку в прогнозе

```
# построим графики слагаемых кросс-энтропийной функции потерь
ym_list = np.arange(0.01, 1, 0.02)
plt.hlines(0, 0, 1); plt.vlines(0, 0, 5.0)
```

```
# случай, когда объект принадлежит классу 1, т.е. y_i = 1
plt.plot(ym_list, -np.log(ym_list), 'r', label='$y_i=1$')
# случай, когда объект принадлежит классу 0, т.е. y_i = 0
plt.plot(ym_list, -np.log(1-ym_list), 'b', label='$y_i=0$')

plt.xlabel('$y_m$'); plt.ylabel('$penalty(y, y_m)$')
plt.legend();
```



2. Пример использования логистической регрессии в качестве классификатора

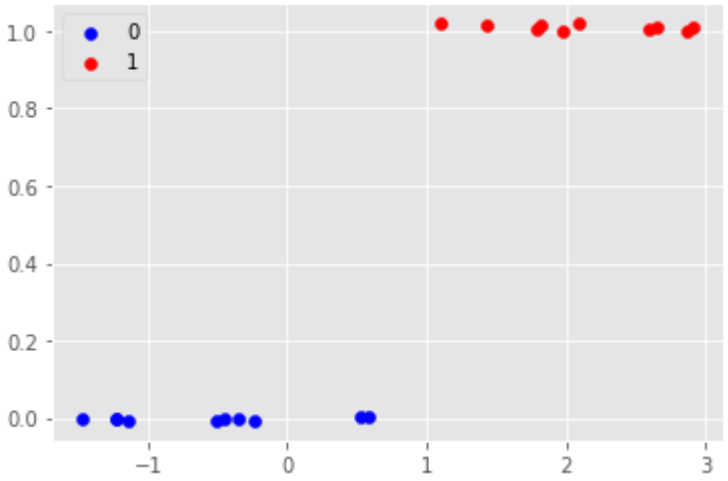
Исходные данные возьмем те же

```
np.random.seed(42)
err_y = 0.02
# сгенерируем множество "нулей" как нормальное распределение N(m,s)
n0 = 10; m = -1; s = 1
x0 = np.random.randn(n0) * s + m
t0 = err_y*(np.random.random(n0) - 0.5)
y0 = np.zeros(n0)

# сгенерируем множество "единиц" - точек "внутри" отрезка ab
```

```
n1 = 10; a = 1; b = 3
par1 = np.random.random(n1)
t1 =  err_y*np.random.random(n1) + 1.0
x1 = par1*a + (1-par1)*b
y1 = np.ones(n1)
y1.shape
```

```
(10,)
```

```
# отобразим графически расположение точек
plt.scatter(x0, t0, c='b', label='0')
plt.scatter(x1, t1, c='r', label='1')
plt.legend();
```

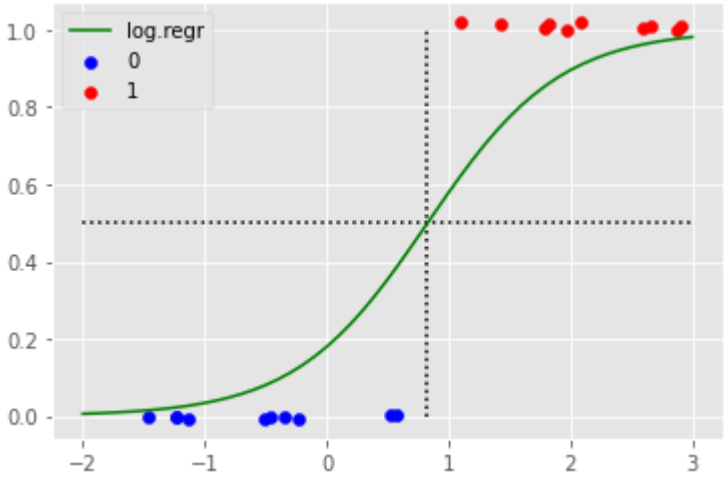



```
from sklearn.linear_model import LogisticRegression
log_regression = LogisticRegression( )

# сформируем выборку и обучим классификатор:
X = np.hstack([x0,x1]).reshape(-1,1)
X.shape
y = np.hstack([y0,y1])
labels = np.hstack([y0,y1])
y.shape
```

```
model = log_regression.fit(X,y)
print('смещение:', model.intercept_)
print('коэфф-ты:', model.coef_)
```

```
смещение: [-1.52435315]
коэфф-ты: [[1.83898584]]
```

```
# нарисуем границу линию регрессии ym = a1*x + a0
a0 = model.intercept_; a1 = model.coef_[0]
xline_0 = np.arange(-2, 3.1, 0.1)
xline_1 = sigmoid(a0 + a1*xline_0)
plt.plot(xline_0, xline_1, 'g', label='log.regr')

plt.hlines(y=0.5, xmin=-2, xmax=3, linestyles='dotted')
plt.vlines(x=-a0/a1, ymin=0, ymax=1, linestyles='dotted')

# оборазим графически расположение точек
plt.scatter(x0, t0, c='b', label='0')
plt.scatter(x1, t1, c='r', label='1')
plt.legend();
```




```
# прогноз и точность классификации
from sklearn import metrics
```

```
classes = (model.predict(X) > 0.5)*1
print(classes)
print('Точность классификации= {:5.2f}%'.format(100*metrics.accuracy_score(y, classes)))
```

```
[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
Точность классификации= 100.00%
```

Из графика видно, что добавление новой точки (15.0, 1) не поменяет картинку, так как штраф в этой точке за классификацию будет почти 0. Т.е. логистическая регрессия чувствительна, в основном, к приграничным точкам.

ВЫВОДЫ

- логистическая регрессия, также как и линейная регрессия строит линию отделения объектов одного класса от всех остальных;
- логистическая регрессия отличается от линейной регрессии тем, что она меняется в интервале от 0 до 1 и может быть использована как вероятностная мера того, что объект принадлежит данному классу.
- логистическая регрессия может использоваться как пороговая функция в бинарной классификации:

$$\begin{cases} class(x) = 0 & \text{если } y_M(x, W) < 0.5 \\ class(x) = 1 & \text{если } y_M(x, W) > 0.5 \end{cases}$$

- логистическая регрессия устойчива при решении задачи классификации - она реагирует только на приграничные точки.

Вопрос: а что делать в более сложных случаях? Когда классов объектов больше двух?

Ответ на этот вопрос простой:

в этом случае строят для каждого класса логистическую регрессию, которая отделяет объекты этого класса от всех остальных и показывает вероятность того, что объект принадлежит данному классу.

В качестве ответа классификатора выбирается тот класс, у которого вероятность принадлежности оказывается больше всех остальных.

Рассмотрим применение логистической регрессии для решения задачи мульти-классифмкации на конкретном примере

Лекция 3. "Классификация объектов"

- Постановка задачи классификации
- Оценка качества классификации.
- Метод kNN - k ближайших соседей
- Логистическая регрессия
- **Пример классификации обучающихся на онлайн-курсе**

Пример классификации обучающихся на онлайн-курсе

==========

что делать в случаях, когда классов объектов больше двух? Много данных?

- в этом случае строят для каждого класса логистическую регрессию, которая отделяет объекты этого класса от всех остальных и показывает вероятность того, что объект принадлежит данному классу.
- В качестве ответа классификатора выбирается тот класс, у которого вероятность принадлежности оказывается больше всех остальных.
- также как и в случае линейной регрессии применяют те же самые приемы регуляризации задачи (L_1, L_2);

Рассмотрим применение логистической регрессии для решения задачи мульти-классификации на примере классификации обучающихся на онлайн-курсе

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler, Normalizer, RobustScaler
from sklearn import metrics
```

```
np.set_printoptions(precision=2)
plt.style.use('ggplot')
```

1. Загрузка данных и первичный анализ

```
from google.colab import drive
drive.mount('/content/drive')

# DIR PATH to DATA
dir_path = "/content/drive/My Drive/Colab Notebooks/COMP_MATH/Cases/case-4/"
```

```
Mounted at /content/drive
```

```
# загрузим данные - результаты выполнения заданий на онлайн-курсе КПК "Академия онлайн-обучения"
#course_file_name = "online_acad_log.csv"
course_file_name = "linal_geom_log.csv"
csv_file_path = dir_path + course_file_name

data = pd.read_csv(csv_file_path, sep=';')
data.tail()
```

	0	1	2	label
142	4.50	5.00	4.83	1
143	5.00	4.50	4.50	1
144	4.00	3.83	5.00	1
145	3.75	5.00	5.00	1
146	5.00	4.50	5.00	0

```
# первичный анализ данных
data.describe()
```

	0	1	2	label
count	147.000000	147.000000	147.000000	147.000000
mean	2.911429	3.758912	3.518639	0.421769
std	2.000098	1.529196	1.639918	0.522444
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	3.830000	3.165000	0.000000
50%	4.000000	4.330000	4.080000	0.000000
75%	4.500000	4.580000	4.500000	1.000000
max	5.000000	5.000000	5.000000	2.000000

2. Подготовка данных для регрессионного анализа

```
# извлекаем данные в np.array
XY = data.values
XY[-3:, :]
```

```
array([[4.  , 3.83, 5.  , 1.  ],
       [3.75, 5.  , 5.  , 1.  ],
       [5.  , 4.5 , 5.  , 0.  ]])
```

```
# выделяем массив значений предиктивных переменных и выходной переменной
Xm = XY[:,:-1]
Y = XY[:,-1]

Xm.shape, Y.shape
```

```
((147, 3), (147,))
```

```
# Выделяем часть данных для обучения (70%), а часть - для проверки точности модели
```

```
x_train, x_test, y_train, y_test = train_test_split(Xm, Y, train_size=0.7, random_state=42)
x_train.shape, x_test.shape
```

```
((102, 3), (45, 3))
```

3. Построение линейной регрессии без дополнительных предикторов

Т.е. будем строить модель вида:

$$y_m(t) = \sigma(a_0 + a_1 * x_1 + a_2 * x_2 + a_3 * x_3)$$

где $\sigma(u) = \frac{1}{1+e^{-u}}$.

3.1. Построим логистическую регрессию с L2-регуляризацией

```
# Построим логистическую регрессию с L2-регуляризацией (по умолчанию)
log_reg = LogisticRegression( )
model = log_reg.fit(x_train, y_train)
print('a0=', model.intercept_)
print(model.coef_)
```

```
a0= [-6.09]
[[0.77 0.1  0.65]]
```

Проанализируем полученную модель

```
# проанализируем насколько хорошо
y_predicted = model.predict(x_test)

print('Точность на обучающей выборке= {}'.format(log_reg.score(x_train, y_train)))
print('Точность на тестовой выборке= {}'.format(log_reg.score(x_test, y_test)))

print('Матрица соответствия: \n', metrics.confusion_matrix(y_test, y_predicted))
```

```
Точность на обучающей выборке= 0.7843137254901961
```

```
Точность на тестовой выборке= 0.8444444444444444
Матрица соответствия:
 [[24  3  0]
 [ 2 14  0]
 [ 1  1  0]]
```

```
metrics.accuracy_score(y_true=y_test, y_pred=y_predicted)
```

```
0.8444444444444444
```

4. Добавление в модель предикторов

Т.е. добавим в модель слагаемые взаимодействия признаков/переменных:

$$y_m(t) = \sigma(a_0 + a_1 * x_1 + \ldots + a_4 * x_4 + a_5 * x_1 * x_1 + a_6 * x_1 * x_2 + \ldots)$$

4.1. Готовим данные

```
def extend_data_lin(data):
    new_data = data.drop(columns='label')
    nn = new_data.shape[1]
    for i in range(nn-1):
        for j in range(i, nn):
            new_data[new_data.columns[i]+'&'+new_data.columns[j]] = new_data.iloc[:,i]*new_data.iloc[:,j]
    new_data[new_data.columns[nn-1]+'&'+new_data.columns[nn-1]] = new_data.iloc[:,nn-1]*new_data.iloc[:,nn-1]
    new_data['label'] = data['label']
    return new_data

data1 = extend_data_lin(data)
print(data1.shape)
data1.tail(3)
```

(898, 15)

	0	1	2	3	0&0	0&1	0&2	0&3	1&1	1&2	1&3	2&2	2&3	3&3	label
895	8.52	9.04	9.33	10.0	72.5904	77.0208	79.4916	85.2	81.7216	84.3432	90.4	87.0489	93.3	100.0	0

```
data2 = data1.values
x = data2[:,:-1]
y = data2[:,-1]
x.shape, y.shape
```

((898, 14), (898,))

```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.75, random_state=42)
x_train.shape, x_test.shape
```

((673, 14), (225, 14))

4.2. Нормируем данные

теперь необходимо нормировать наши предикторные переменных, так как разброс взаимодействий переменных на порядок больше (0-100), чем разброс значений самих переменных (0-10).

```
# Используем MinMax нормирование, чтобы все переменных менялись в интервале от 0 до 1
x_scaler = MinMaxScaler()
x_scaler.fit(x_train)
x_train = x_scaler.transform(x_train)
x_test = x_scaler.transform(x_test)
x_train[-3:, :]
```

```
array([[1.  , 0.7 , 0.77, 1.  , 1.  , 0.7 , 0.77, 1.  , 0.49, 0.54, 0.7 ,
        0.59, 0.77, 1.  ],
       [0.91, 0.98, 0.61, 0.64, 0.84, 0.89, 0.56, 0.59, 0.96, 0.6 , 0.63,
        0.38, 0.39, 0.41],
       [0.36, 0.  , 0.  , 0.  , 0.13, 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ,
        0.  , 0.  , 0.  ]])
```

4.3. Построим логистическую регрессию с L2-регуляризацией

```
# Построим логистическую регрессию с L2-регуляризацией (по умолчанию)
log_reg = LogisticRegression(C=2.0)
model_3 = log_reg.fit(x_train, y_train)
print('a0=', model_3.intercept_)
print(model_3.coef_)

# проанализируем насколько хорошо
y_predicted = model_3.predict(x_test)

print('Точность на обучающей выборке= {}'.format(log_reg.score(x_train, y_train)))
print('Точность на тестовой выборке= {}'.format(log_reg.score(x_test, y_test)))

print('Матрица соответствия: \n', metrics.confusion_matrix(y_test, y_predicted))
```

```
a0= [-4.76]
[[ 0.82  1.09  1.74  1.62 -0.04 -0.13  0.63 -0.09 -0.36  0.07 -0.31  0.29
  -0.48  0.39]]
Точность на обучающей выборке= 0.687964338781575
Точность на тестовой выборке= 0.7155555555555555
Матрица соответствия:
 [[123  31]
 [ 33  38]]
```

ВЫВОДЫ:

Точность логистической регрессии можно улучшить за счет

- применения регуляризации: L_2 - регуляризация уменьшает переобученность, L_1 -регуляризация упрощает саму структуру модели;
- добавления дополнительных предикторов в виде взаимодействия исходных параметров;
- масштабирования параметров;

Лекция 4.1. "Снижение размерности. Кластеризация объектов"

- Задачи анализа данных
- **Линейный факторный анализ. Метод главных компонент**
- Снижение размерности: метод t-SNE
- Кластеризация объектов: метод k-means
- Иерархическая кластеризация

4.1.2. Линейный факторный анализ. Метод главных компонент

==========

Чтобы понять идею данного метода снижения размерности данных давайте рассмотрим пример ...

```
# Импорт необходимых библиотек
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')

# %matplotlib inline
import seaborn as sns
```

1. Пример постановки задачи снижения размерности данных

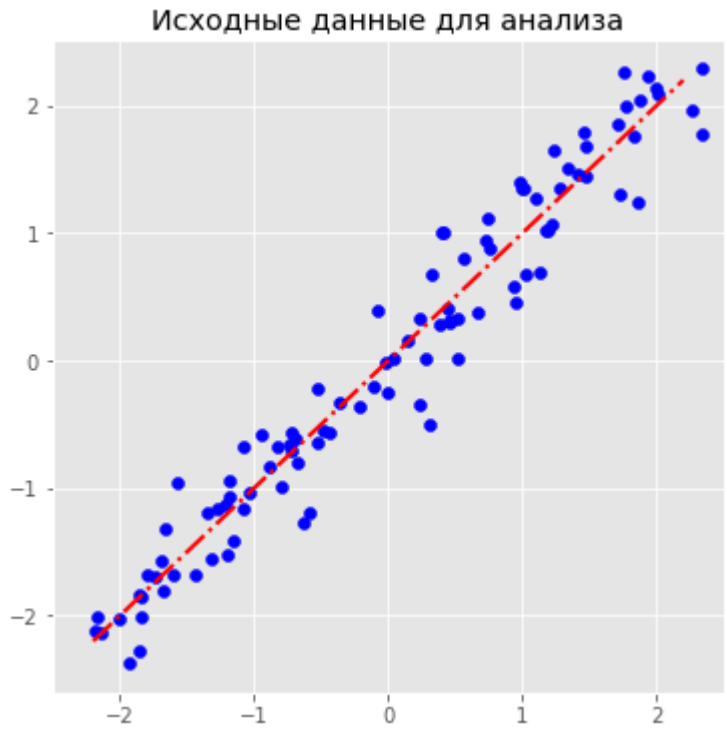
Сгенерируем данные на плоскости. При этом расположим их так, чтобы они лежали на прямой с небольшим отклонением от нее.

```
def linfun(x):
    return x

# Создание датасета
x1 = np.linspace(-2.2, 2.2, 100)
fx = linfun(x1)
dots = np.vstack([x1, fx]).T
noise = 0.2 * np.random.randn(*dots.shape)
dots += noise

# Цветные точки для отдельной визуализации позже
from itertools import cycle
size = 25
colors = ["r", "g", "c", "y", "m"]
```

```
idxs = range(0, x1.shape[0], x1.shape[0]//size)
vx1 = x1[idxs]
vdots = dots[idxs]
```

```
# Визуализация
plt.figure(figsize=(6, 6))
plt.xlim([-2.5, 2.5])
plt.scatter(dots[:, 0], dots[:, 1], c='b')
plt.plot(x1, fx,  color="red", linewidth=2, linestyle = 'dashdot')
plt.title('Исходные данные для анализа')
```

```
Text(0.5, 1.0, 'Исходные данные для анализа')
```



Хотя каждая точка на плоскости характеризуется двумя координатами, мы видим, что если не учитывать "шум", то каждую точку можно характеризовать ее "проекцией" на скрытую от нас линию, т.е. одной координатой.

Таким образом, появляется задача снижения размерности данных, называемая задачей факторного анализа, в которой:

- появляются новые оси координат, которые в дальнейшем будут называться факторами;
- но при этом новые признаки должны сохранить как можно больше изменчивости и вариативности наших исходных данных (структуру, взаимные расстояния, ...).

Если при этом новые оси являются прямыми линиями, т.е. новые признаки являются линейной комбинации исходных признаков, то такая задача называется задачей линейного

факторного анализа.

Для решения такой задачи очень часто используется метод главных компонент. В этом методе новые оси строятся так, чтобы они были ортогональными друг другу и называются главными компонентами.

2. Метод главных компонент

Метод главных компонент предлагает находить главные оси поочередно, начиная с той оси, вдоль которой имеется наибольшая дисперсия данных.

Эту задачу можно формализовать как задачу поиска собственного вектора матрицы ковариации с наибольшим собственным числом:

$$Cov(X) = X^T * X$$

где X - матрица исходных признаков.

Так как все остальные компоненты мы ищем так, чтобы они были ортогональны главному компоненту и друг другу, то в целом задачу поиска главнымх компонент можно сформулировать как задачу поиска собственных векторов матрицы ковариации. Ее собственные числа все больше нуля и упорядочивая собственные вектора по убыванию соответствующих им собственных значений, мы и получим новые компоненты в порядке убывания их важности.

Начнем с решения нашей демо задачи, чтобы продемонстрировать результат работы метода и его реализацию.

(демо-пример взят из статьи на хабре: https://habr.com/en/post/331500/)

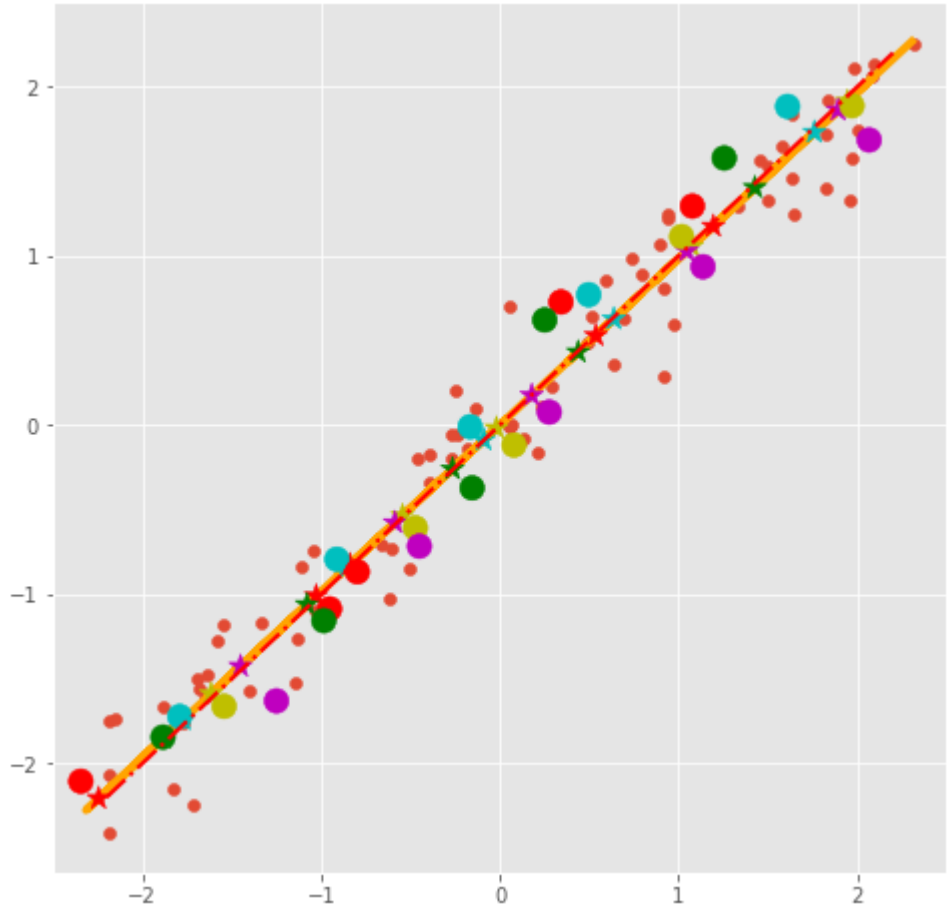
```
# Применение PCA
from sklearn.decomposition import PCA
pca = PCA(1)
pca_coords = pca.fit_transform(dots)
print('Доля объясненной вариации данных=', round(pca.explained_variance_ratio_[0]*100, 2))
print('Кординаты вектора главного компонента=', pca.components_[0])
pca_coords.shape
```

```
Доля объясненной вариации данных= 99.04
Кординаты вектора главного компонента= [-0.71299233 -0.70117183]
(100, 1)
```

```
# Восстановление данных по главному компоненту
pdots_pca = pca.inverse_transform(pca_coords)
vpdots = pdots_pca[idxs]
pdots_pca.shape
```

```
(100, 2)
```

```
# Визуализация
plt.figure(figsize=(8, 8))
plt.xlim([-2.5, 2.5])
plt.scatter(dots[:, 0], dots[:, 1], zorder=1)
plt.plot(pdots_pca[:,0], pdots_pca[:,1], color='orange', linewidth=4, zorder=4)
plt.plot(x1, fx,  color="red", linewidth=2, zorder=10, linestyle = 'dashdot')
plt.scatter(vpdots[:,0], vpdots[:,1], color=colors*5, marker='*', s=150, zorder=5)
plt.scatter(vdots[:,0], vdots[:,1], color=colors*5, s=150, zorder=6);
```



3. Применение и анализ метода главных компонент

Посмотрим как метод может быть применен к анализу реальных данных

```
# Загрузим таблицу с реальными данными
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
dir_path = "/content/drive/My Drive/Colab Notebooks/NEURO_NETs/4 Clusterization/data/"
df_wine = pd.read_csv(dir_path + 'winequality-red.csv', sep = ';')
df_wine.tail()
```

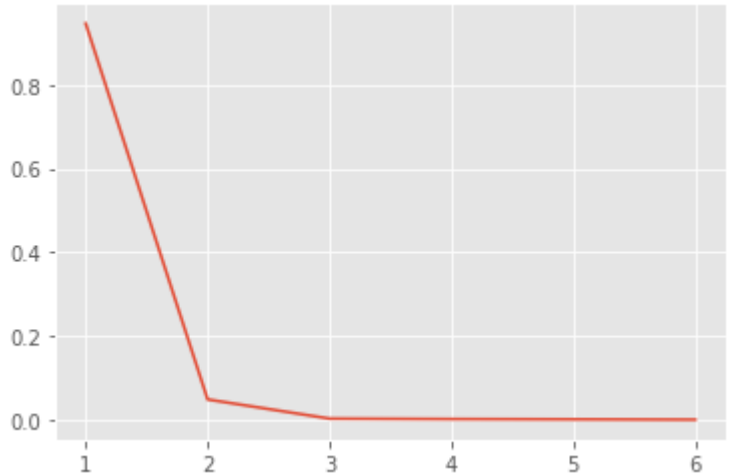
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	s
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	

```
# Выделим данные для кластеризации
df_wine.loc[:, 'quality_cat'] = (df_wine.quality > 5).astype(int)
df_wine = df_wine.drop('quality', axis=1)
X = df_wine.iloc[:, :-1]
y = df_wine.iloc[:, -1]
```

```
pca = PCA(n_components= 6)
pca.fit(X)
Z = pca.transform(X)
X.shape, Z.shape, pca.components_.shape
```

```
((1599, 11), (1599, 6), (6, 11))
```

```
plt.plot(list(range(1,7)), pca.explained_variance_ratio_);
```



ВЫВОДЫ

метод главных компонент

- **ВХОД**: исходные данные в виде матрицы исходных признаков X и количество компонент d, которые мы хотим получить.

1. Центрируем данные, то есть из каждого столбца вычитаем его среднее значение.
2. Считаем матрицу ковариаций S и находим d наибольших собственных чисел и соответствующих собственных векторов данной матрицы.
3. Составляем матрицу преобразования - матрицу состоящую из этих собственных векторов - и получаем новое признаковое описание.

Лекция 4.1. "Снижение размерности. Кластеризация объектов"

- Задачи анализа данных
- Линейный факторный анализ. Метод главных компонент
- **Снижение размерности: метод t-SNE**
- Кластеризация объектов: метод k-means
- Иерархическая кластеризация

4.1.3. Снижение размерности: метод t-SNE

==========

Линейный факторный анализ не всегда помогает эффективно снижать размерность на реальных данных.

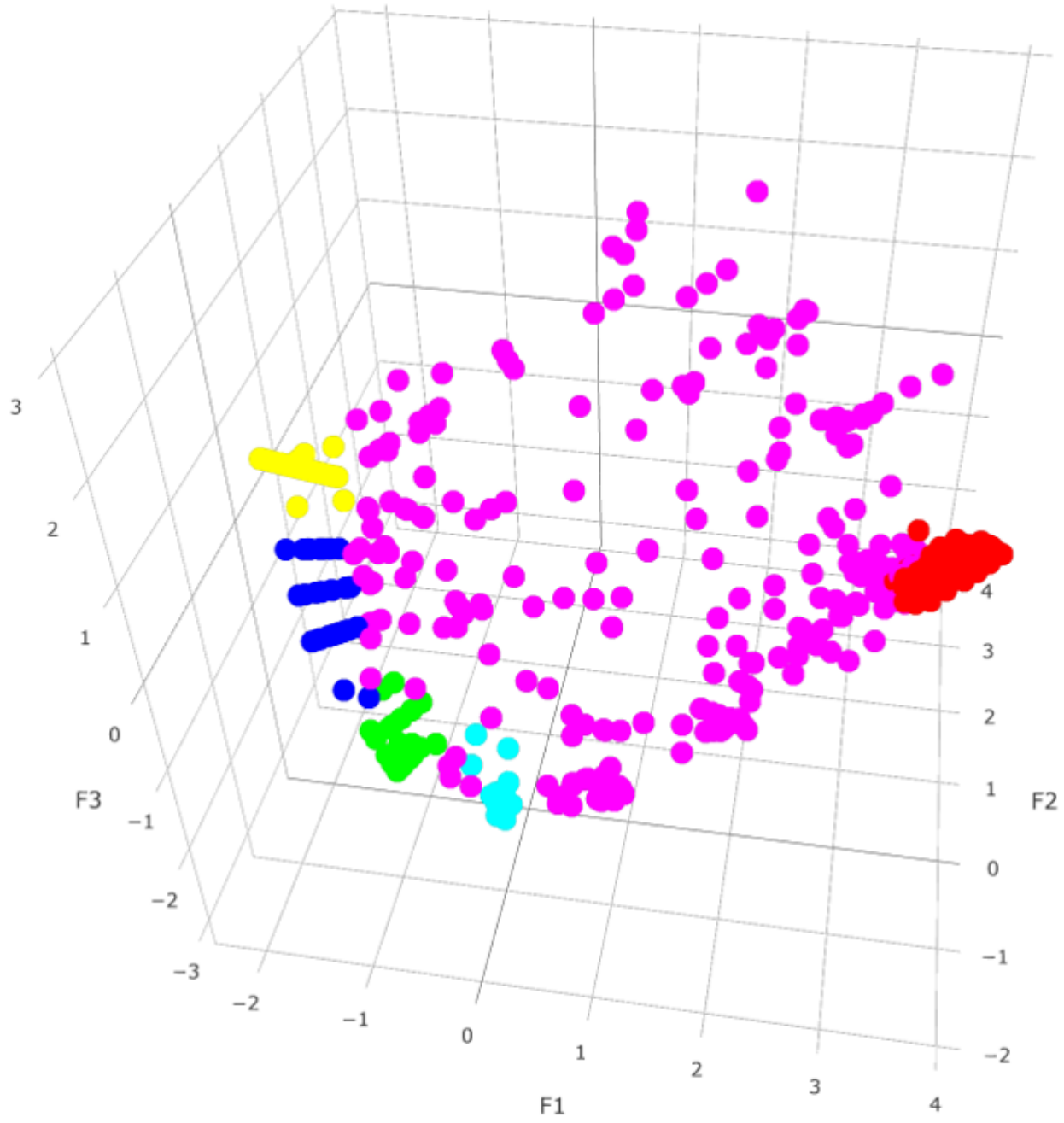


Рис.1. Распределение обучающихся на онлайн-курсе по результатам выполнения 8 заданий курса, спроецированным на 3-х мерное пространство с сохранением локальной структуры с помощью метода t-SNE

Линейный факторный анализ показывает, что здесь имеется 5 значимых общих факторов, которые объясняют примерно 95% вариации данных в 8-мерном пространстве.

Нелинейный же нейросетевой факторный анализ показывает, что два фактора вполне достаточно, чтобы объяснить более 90% вариации результатов обучения. А три фактора объясняют более 95% вариации.

---

2. Проблема анализа нелинейных факторов

- Методы линейного факторного анализа решают задачу снижения размерности путем проецирования; при этом, если выделенный набор факторов достаточно точно описывает исходные данные, то при переходе в новое признаковое пространство достаточно точно воспроизводятся пропорции в расстоянии между точками. Т.е., если расстояние между точками A,B в исходном пространстве было больше расстояния между точками C,D в два раза, то оно и в новом признаковом пространстве будет примерно в 2 раза больше;
- проблема только в том, что если точки лежат не на плоскости и не вдоль каких-то прямых в исходном признаковом пространстве, то PCA приодится использовать большое количество факторов, чтобы хорошо описать данные; при этом теряется выразительность самых главных компонент;

Продемонстрируем этот эффект на исскусственно сгенерированном примере.

```
# Импорт необходимых библиотек
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')

# %matplotlib inline
import seaborn as sns
```

3. Пример применения PCA на задаче с нелинейными факторами

Сгенерируем данные на квадратической поверхности (эллиптическом параболоиде) в 3-х мерном пространстве. При этом расположим их так, чтобы они лежали вдоль одной линии с небольшим отклонением от нее.

```
# Создание датасета точек на плоскости
xmin, xmax = -3.4, 3.4
x1 = np.linspace(xmin, xmax, 500)
fx = np.sin(x1)
dots = np.vstack([x1, fx]).T
```

```
noise = 0.2 * np.random.randn(*dots.shape)
dots += noise

# Цветные точки для отдельной визуализации позже
from itertools import cycle
size = 25
colors = ["r", "g", "c", "y", "m"]
idxs = range(0, x1.shape[0], x1.shape[0]//size)
vx1 = x1[idxs]
vdots = dots[idxs]
```

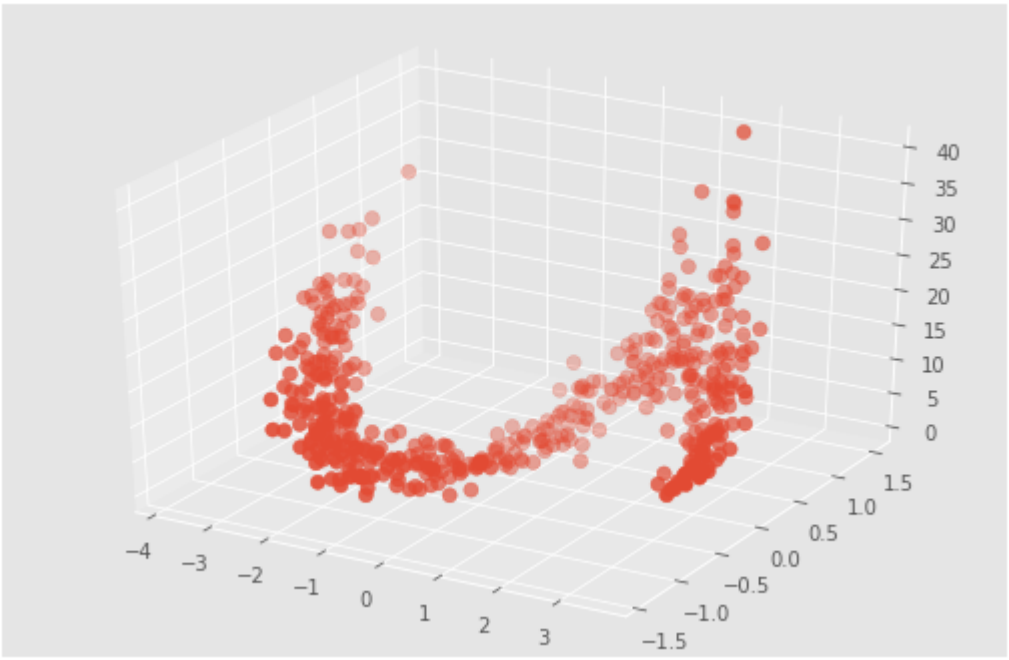
```
# Визуализация
plt.figure(figsize=(6, 6))
plt.xlim([xmin, xmax])
plt.scatter(dots[:, 0], dots[:, 1], c='b')
#plt.plot(x1, fx,  color="red", linewidth=2, linestyle = 'dashdot')
plt.title('Проекции точек на плоскость XOY')
```

```
Text(0.5, 1.0, 'Проекции точек на плоскость XOY')
```


```
# Добавим третье измерение
def zfun(xy):
    x, y = xy[0], xy[1]
    z = (x - 1.0)**2 + (y+1)**4
    noise = 0.2 * np.random.randn(*z.shape)
    z += noise
    return z

z = zfun(dots.T)
dots_3d = np.hstack([dots, z.reshape((len(z),1))])
dots_3d.shape
```

```
(500, 3)
```

```
# Посмотрим на получившиеся в 3D точки
from mpl_toolkits.mplot3d import Axes3D
XYZmat = dots_3d.T
# Настраиваем 3D график
fig = plt.figure(figsize=[9, 6])
ax = fig.gca(projection='3d')
# Задаем угол обзора
ax.view_init(30, -60)
# нарисуем точки
ax.scatter(XYZmat[0], XYZmat[1], XYZmat[2], s = 50, marker='o')
plt.show()
```



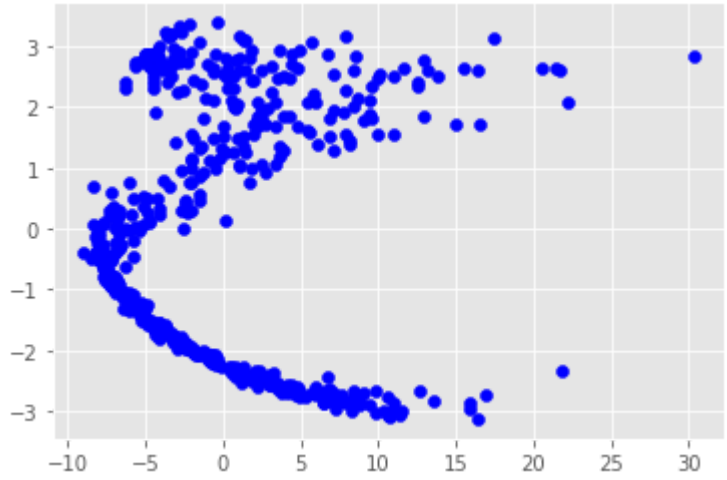
Хотя каждая точка характеризуется тремя координатами, мы видим, что если не учитывать "шум", то каждую точку можно характеризовать ее "проекцией" на скрытую от нас линию, т.е. одной координатой.

```
# Применение PCA
from sklearn.decomposition import PCA
pca = PCA(2)
pca_coords = pca.fit_transform(dots_3d)
print('Доля объясненной вариации данных=', np.round(pca.explained_variance_ratio_*100, 2))
print('Кординаты вектора главного компонента=', pca.components_)
pca_coords.shape
```

```
Доля объясненной вариации данных= [90.29  9.38]
Кординаты вектора главного компонента= [[-0.04160843  0.05159943  0.9978007 ]
```

```
 [ 0.96577099  0.25799442  0.02693108]]
(500, 2)
```

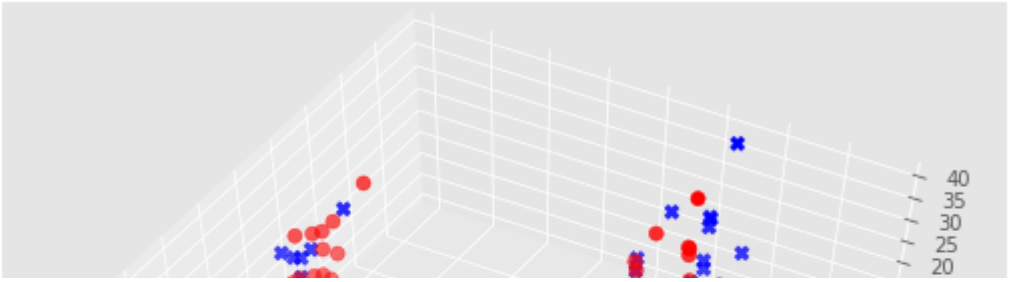
```
plt.scatter(pca_coords[:, 0], pca_coords[:, 1], c='b');
```




```
# Восстановление данных по главному компоненту
pdots_pca = pca.inverse_transform(pca_coords)
vpdots = pdots_pca[idxs]
pdots_pca.shape
```

```
(500, 3)
```

```
XYZmat = dots_3d.T
XYZpca = pdots_pca.T
# Настраиваем 3D график
fig = plt.figure(figsize=[9, 6])
ax = fig.gca(projection='3d')
# Задаем угол обзора
ax.view_init(60, -60)
# нарисуем точки
ax.scatter(XYZmat[0], XYZmat[1], XYZmat[2], s = 50, marker='o', c='r')
ax.scatter(XYZpca[0], XYZpca[1], XYZpca[2], s = 50, marker='X', c='b')
plt.show()
```



- видим, что PCA требуется больше факторов, чем нужно, чтобы описать с хорошей точностью исходные данные.
- один главный компонент плохо будет описывать разнообразие данных несмотря на то, что объясняет 90% вариации.

4. Метод t-SNE: t-distributed Stocastic Neighbor Embedding

Метод "переносит" структуру данных из исходного в новое сжатое пространство признаков с использованием следующего вероятностного описания окрестности произвольной точки **A**.

Пусть мы зафиксировали некоторую точку a из исходной выборки. Выберем некоторое множество точек $M(a, n)$ в ее окружении, наиболее близко к ней расположенных; здесь n - количество точек;

зададим условное вероятностное распределение на множестве данных точек $x^{(i)} \in M(a, n)$ следующим образом:

$$p(x^{(i)}|a) = \frac{e^{-(x^{(i)}-a)/\sigma^2}}{\sum_{x^{(k)} \in M(a,n)} e^{-(x^{(k)}-a)/\sigma^2}}$$

σ - это параметр размера окрестности, который определяется пользователем.

используя то же множество точек мы можем для какой-то из точек окружения $b \in M(a, n)$ определить $p(a|b)$;

$p(b|a), p(a|b)$ - характеризуют некоторую "близость" объектов a и b между собой; поэтому определим новую вероятностную меру близости между точками a и b в нашей подвыборке размера n таким образом, чтобы она была симметричной:

$$p(a, b) = \frac{p(b|a) + p(a|b)}{2}$$

если мы обозначим координаты данных точек в новом признаковом пространстве через $y^{(i)}$, то мы можем аналогично определить

$$q(y^{(i)}|a) = \frac{e^{-(y^{(i)}-y(a))/\sigma^2}}{\sum_{x^{(k)} \in M(a,n)} e^{-(y^{(k)}-y(a))/\sigma^2}}$$

и $q(y(a), y(b)) = \frac{q(y(b)|y(a))+q(y(a)|y(b))}{2}$.

Идея метода SNE состоит в том, чтобы подобрать координаты $y^{(i)}$ для точек нашей исходной выборки таким образом, чтобы минимизировать расхождение между двумя вероятностными распределениями минимизируя дивергенцию Кулбака-Лейблера:

$$\Phi(Y) = KL(P||Q) = \sum_i \sum_j p(x^{(i)}, x^{(j)}) \cdot log \frac{p(x^{(i)}, x^{(j)})}{q(y^{(i)}, y^{(j)})}$$

Метод t-SNE отличается от метода SNE тем, что использует t-распределение Стьюдента с 1-й степенью свободы

$$g(z) = \frac{1}{1+z^2}$$

для оценки

$$q(y^{(i)}|y(a)) = \frac{g(|y^{(i)} - y(a)|)}{\sum_{x^{(k)} \in M(a,n)} g(|y^{(k)} - y(a)|)}$$

Метод t-SNE оказался проще для реализации и лучше по своим результатам.

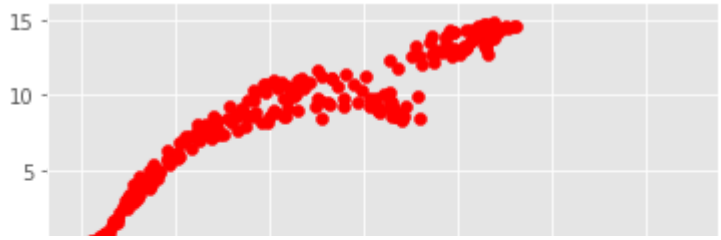
Посмотрим как метод применяется к анализу данных с использованием пакета sklearn

---

применим его к проецированию наших игрушечных данных

```
from sklearn.manifold import TSNE
```

```
tsne = TSNE(n_components=2, perplexity=50)
tsne.fit(dots_3d)
Z = tsne.embedding_
Z.shape

plt.scatter(Z[:, 0], Z[:, 1], c='r');
```



ВЫВОДЫ

- В отличие от PCA и методов многомерного шкалирования, которые стараются сохранить пропорции расстояний между точками в исходном и новом признаковом пространствах метод t-SNE старается сохранить окружение каждой точки;
- если точка **B** находилась в окрестности точки **A** в исходном признаковом пространстве, то и в новом признаковом пространстве она тоже будет находиться в окрестности этой точки. Отличие от методов шкалирования заключается в том, что метод не стремится сохранить углы между векторами.
- основным параметров метода является параметр **perplexity** - сколько точек в окружении любой точки учитывать при переносе в новое признаковое пространство; т.е., если **perplexity = 10**, значит метод будет стараться, чтобы 10 ближайших точек к точке **A** в исходном признаковом пространстве оказались также в окрестности точки **A** в новом признаковом пространстве; при этом методу будет все равно как будут взаимно располагаться далекие друг от друга точки в новом признаковом пространстве;
- метод может давать разные результаты при двух запусках с одинаковыми данными и параметрами;

Лекция 4.1. "Снижение размерности. Кластеризация объектов"

- Задачи анализа данных
- Линейный факторный анализ. Метод главных компонент
- Снижение размерности: метод t-SNE
- **Кластеризация объектов: метод k-means**
- Иерархическая кластеризация

4.1.4. Кластеризация объектов: метод k-means

==========

Предположим, что у вас есть N - большое количество слушателей онлайн-курса, описываемых большим количеством признаков M (результатами решения заданий курса, количеством просмотров видеоуроков курса, количеством попыток решений практических заданий и.т.п.).

Вы должны классифицировать этих слушателей, чтобы для каждого класса слушателей применить свое правило сопровождения обучения. Например, слушателям одного класса надо посылать напоминание о том, что истекает срок выполнения очередного задания. Другим слушателям надо послать поощрительное послание о том, что они молодцы и все успели вовремя сдать и пожелать им дальше также продолжать.

Но проблема заключается в том, что вы:

1. еще не знаете на какие классы могут делиться слушатели;
2. слушатели не размечены по этим классам, чтобы можно было обучить и применить какой-то метод классификации.

И мы вынуждены искать структуру в данных без использования каких-либо метод классов (обучение без учителя) с помошью кластеризации.

```
# Импорт необходимых библиотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')

from sklearn.manifold import TSNE
from sklearn.datasets import load_digits
```

1. Иллюстрация задачи кластеризации данных

Чтобы проиллюстрировать идею кластеризации рассмотрим следующий пример.

Предположим. что у вас есть большая коллекция картинок. Вам надо определить классы этих картинок. Но вы не знаете на сколько классов имеет смысл разделить эти картинки.

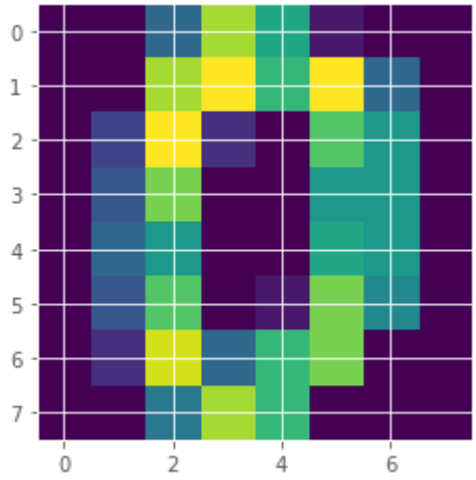
Попробуем применить t-SNE чтобы понять это

```
# загрузим картинки цифр
digit = load_digits()
img = digit.images
img.shape
```

```
(1797, 8, 8)
```

```
# Визуализация
plt.imshow(img[0]);
img[0]
```

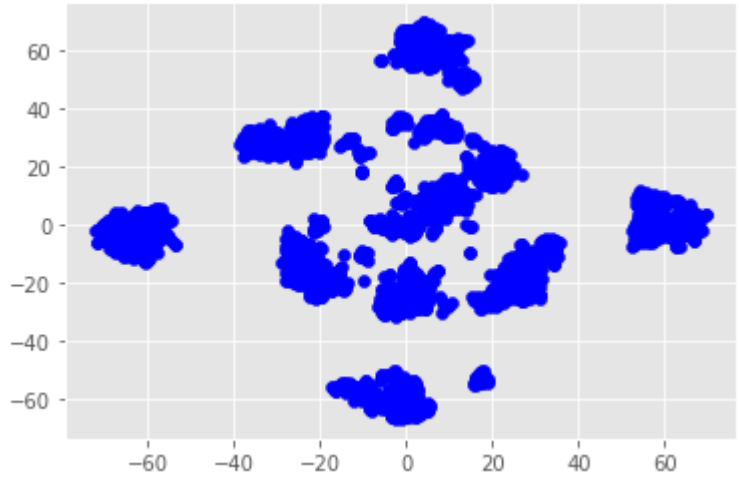
```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```



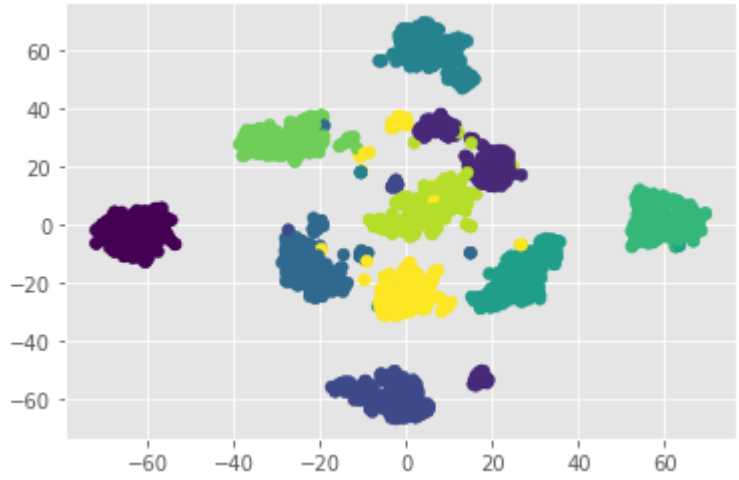

```
# применим t-SNE
X = img.reshape(-1, 64)
tsne = TSNE(n_components=2, perplexity=30)
tsne.fit(X)
Z = tsne.embedding_
Z.shape
```

```
(1797, 2)
```

```
# визуализируем получившееся разбиение
plt.scatter(Z[:, 0], Z[:, 1], c='b');
```




```
# визуализируем получившееся разбиение с учетом знания цифр на картинках
y = digit.target
plt.scatter(Z[:, 0], Z[:, 1], c=y);
```



2. Метод k-means

Самый простой метод k-средних предлагает характеризовать кластер C_k центроидом $c^{(k)}$ - центром масс объектов, которые входят в этот кластер.

Объект x входит в тот кластер, центроид которого ближе всего находится к данному объекту:

$$x \in C_k \Leftrightarrow k = arg\min_j ||x - c_j||$$

Таким образом, мы можем сформулировать следующую задачу разбиения C исходной выборки X на K кластеров как задачу минимизации функционала:

$$(*)\ \ \Phi(C) = \sum_{k=1..K} \sum_{x^{(i)} \in C_k} ||x^{(i)} - c_k||$$

Из математической статистики известно, что при фиксированном разбиении C минимум этого функционала получится при выборе центроидов c_k как центров масс объектов, относящихся к этому кластеру:

$$c_k = \frac{\sum_{x^{(i)} \in C_k} x^{(i)}}{|C_k|}$$

Однако, число различных разбиений на K классов на множестве этих объектов будет экспоненциально возрастать с ростом количества объектов.

---

3. Алгоритм k-средних

Данный алгоритм не претендует на нахождение глобального минимума. Но при "хорошем" выборе начальных центроидов он дает неплохие результаты (хороший локальный минимум).

ВХОД: выборка $\{x^{(i)}, i \in \{1, \ldots, N\}\}$, K - количество кластеров

1. Инициализируем центроиды $c_k, k = 1..K$.
2. В соответствии с центроидами (пере)определяем разбиение C исходной выборки по правилу, что объект x входит в тот кластер, центроид которого ближе всего находится к данному объекту.
3. Корректируем центроиды по формуле

$$c_k = \frac{\sum_{x^{(i)} \in C_k} x^{(i)}}{|C_k|}$$

4. Если разбиение не изменилось или если превышен лимит шагов, то выход; иначе - идти к шагу 2.

ВЫХОД: разбиение на кластеры исходной выборки объектов C

---

4. Применение метод k-means

Из алгоритма понятно, что результат работы алгоритма определяется:

- прежде всего количеством заданных кластеров K, которое должно определяться заранее пользователем;
- начальной инициализацией центроидов.

Чтобы проиллюстрировать последнее рассмотрим предыдущий пример с изображениями цифр.

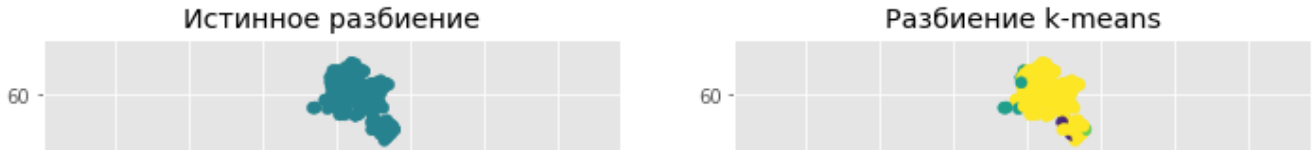
Мы будет случайно определять начальное расположение центроидов. Посмотрим как будет меняться финальное разбиение и насколько оно будет соответствовать реальному делению на классы.

```
# Применение k-means
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=10, init='random', n_init=10, random_state=10)
```

```
# применяем - обучаем
kmeans.fit(X)
# метки кластеров
labels = kmeans.labels_
labels.shape
```

```
(1797,)
```

```
# визуализируем получившееся разбиение с учетом знания цифр на картинках
# и сравним с разбиением на кластеры, полученным с помощью k-means
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
ax[0].scatter(Z[:, 0], Z[:, 1], c=y)
ax[1].scatter(Z[:, 0], Z[:, 1], c=labels)
ax[0].set_title('Истинное разбиение')
ax[1].set_title('Разбиение k-means')
fig.show()
```



5. Метод k-means++

Мы видим, что случайный выбор начального расположения центроидов не совсем хороший способ добиться устойчивого и надежного результата.

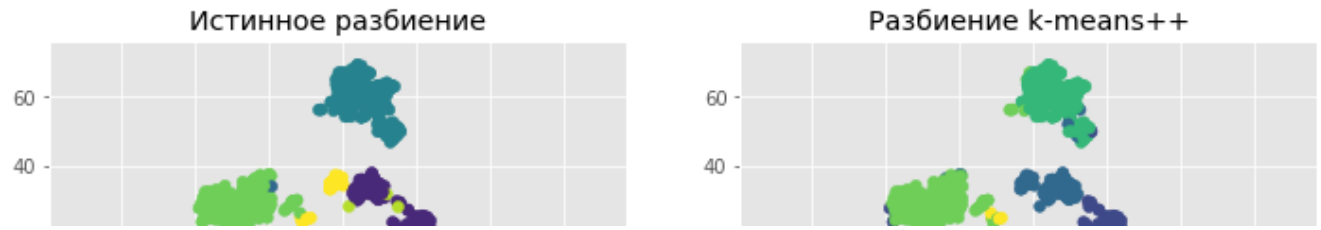
Поэтому метод был усовершенствован - добавлен специальный способ инициализации; метод назвали k-means++.

Идея метода состоит в том, что только 1-й центроид выбирают случайно. А вот следующий центроид мы будем выбирать так, чтобы он как можно дальше отстоял от выбранных ранее центроидов. Но делать мы это будем не жестко, а с вероятностями, пропорциональными этим расстояниям.

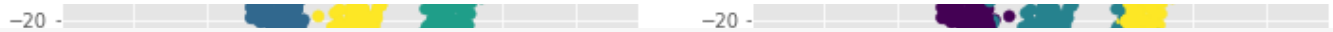
Посмотрим какой результат этот метод даст на рассмотренной выше задаче распознавания изображений цифр.

```
kmeans = KMeans(n_clusters=10, init='k-means++', n_init=10, random_state=42)
```

```
# применяем - обучаем
kmeans.fit(X)
# метки кластеров
labels = kmeans.labels_
# визуализируем получившееся разбиение с учетом знания цифр на картинках
# и сравним с разбиением на кластеры, полученным с помощью k-means
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
ax[0].scatter(Z[:, 0], Z[:, 1], c=y)
ax[1].scatter(Z[:, 0], Z[:, 1], c=labels)
ax[0].set_title('Истинное разбиение')
ax[1].set_title('Разбиение k-means++')
fig.show()
```

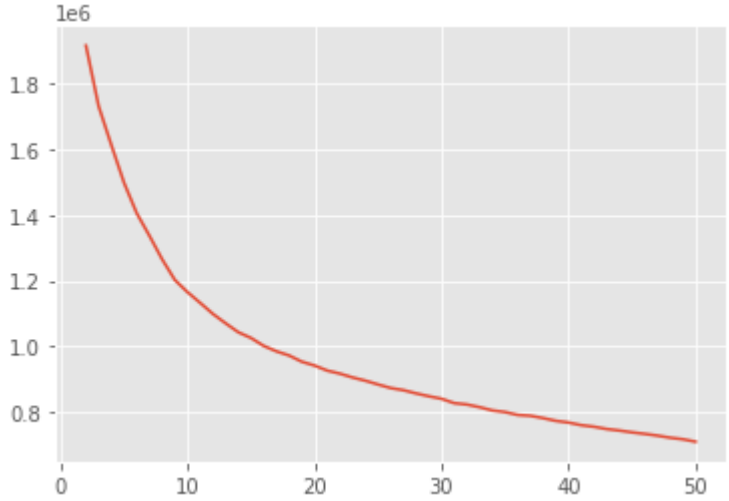


6. Выбор количества кластеров. Метод локтя




```
crit = []
for k in range(2, 51):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    crit.append(kmeans.inertia_)
```

```
plt.plot(range(2,51), crit);
```



Мы видим, что наиболее резкое снижение критерия (*) заканчивается при 10 кластерах и после отметки в 20 кластеров дальнейшее снижение критерия идет по прямой, с одинаковой скоростью. Поэтому можно считать, что выбор K в интервале 10 - 20 кластеров - это хороший выбор.

ВЫВОДЫ

- кластеризация объетов выполняется тогда, когда у вас нет нужных меток у объектов и вы хотите получить представление о том, на сколько классов и на какие классы следует делить объекты;

- хорошая идея перед кластеризацией визулизировать структуру объектов с помощью метода t-SNE;
- метод k-средних прост в реализации, но его результат сильно зависит от начального размещения центроидов;
- **k-means++** модификация метода **k-means**, в котором используется более "умная" инициалиазация центроидов;
- В следующем уроке мы рассмотрим более устойчивый способ кластеризации, который, в том числе, можно использовать и для инициализации центроидов в методе k-means.

Лекция 4.1. "Снижение размерности. Кластеризация объектов"

- Задачи анализа данных
- Линейный факторный анализ. Метод главных компонент
- Снижение размерности: метод t-SNE
- Кластеризация объектов: метод k-means
- **Иерархическая кластеризация**

4.1.5. Иерархическая кластеризация

==========

Если вам ничего не известно о структуре данных и вы хотели бы получить какое-то представление о том, на какое число кластеров имеет смысл разбивать, то лучше всего применить метод иерархической кластеризации.

1. Основная идея

Различают аггломеративную и дивизионную иерархические кластеризации.

- В дивизионной кластеризации в начале вы имеете один кластер, в который включены все объекты. На каждом шаге вы делите очередной кластер на два.
- В аггломеративной кластеризации все наоборот - в начале вы имеете столько кластеров - сколько у вас объектов. И каждый кластер состоит из одного объекта. На каждом шаге вы объединяте каких-то два ближайших кластера в один.

Более популярна аггломеративная кластеризация. Её мы и рассмотрим подробно.
Основные моменты, которые нужно определить, чтобы реализовать данный метод:

- какую метрику использовать для измерения расстояния между отдельными объектами;
- как измерять расстояние между кластерами объектов.

Если на первый вопрос ответы не так сложны - в боольшинстве случаев выбирается либо евклидово либо манхеттенское расстояние, то ответ на 2-й вопрос не очевиден.

Давайте рассмотрим простой пример.

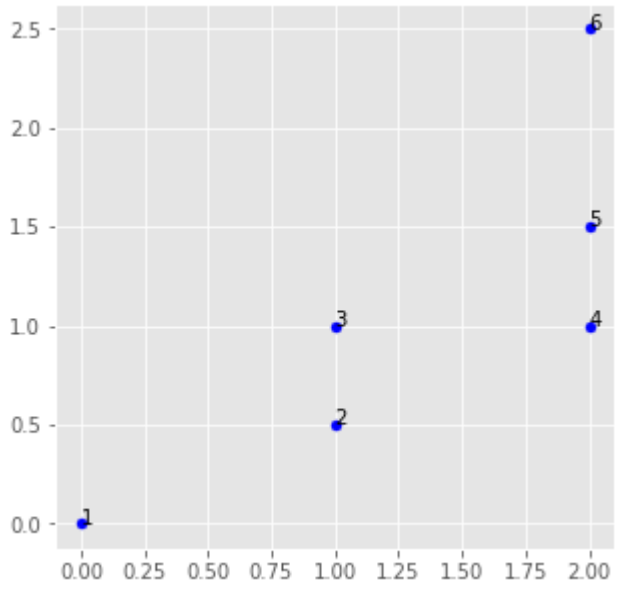
```
# Импорт необходимых библиотек
import numpy as np
import pandas as pd
```

```
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

2. Пример выбора метрики для оценки расстояний между кластерами

Чтобы проиллюстрировать идею кластеризации рассмотрим следующий простой пример.

```
# создадим массив из нескольких точек с координатами (x,y)
plt.figure(figsize=(5,5))
x = np.array([0., 1., 1., 2.0, 2., 2.0])
y = np.array([0., 0.5, 1.0, 1.0, 1.5, 2.5])
labels = [str(i) for i in list(range(1,len(x)+1))]
xy = np.vstack([x,y]).T
plt.scatter(x, y, c='b', s=25)
for i in range(len(labels)):
    plt.text(x[i], y[i], s = labels[i])
plt.show()
```



Будем для простоты использовать манхеттенскую метрику для оценки расстояний между объектами.

Тогда на 1-м шаге метода иерархической кластеризации в один кластер A объединятся точки №№ 2 и 3, так как между ними наименьшее расстояние = 0.5 (по манх.метрике).

На 2-м шаге в один кластер B объединятся точки 4 и 5. Между ними расстояние равно 1.0.

На 3-м шаге можно объединить точку 1 с кластером A. Или точку 6 с кластером B. Но с другой стороны, как посчитать расстояние между точкой №1 и вновь образованным кластером A? Ведь если считать расстояние до кластера как расстояние до ближайшей

точки, то оно равно 1.5. Если до самой дальней точки, то оно равно 2. А если до центра кластера, то расстояние будет равно 1.75.

В зависимости от способов рассчёта расстояний между кластерами (linkage) различают методы иерархической кластеризации.

- **Single Linkage** - расстояние между кластерами определяется по самым близким объектам;
- **Complete Linkage** - расстояние между кластерами определяется по самым дальним объектам;
- **Centroid Linkage** - расстояние между кластерами определяется как расстояние между центроидами кластеров;
- **Average Linkage** - расстояние между кластерами определяется как *групповое среднее расстояние* по всем парам объектов из разных кластеров;
- **Ward Linkage** - расстояние между кластерами определяется как расстояние между центроидами кластеров, но делается поправка на размер кластеров;

На практике чаще используют **Complete Linkage**, **Average Linkage** и **Ward Linkage**. Они монотонны, соответствуют аксиомам расстояния и дают более лучшие результаты.

3. Визуализация кластерной структуры. Дендрограмма

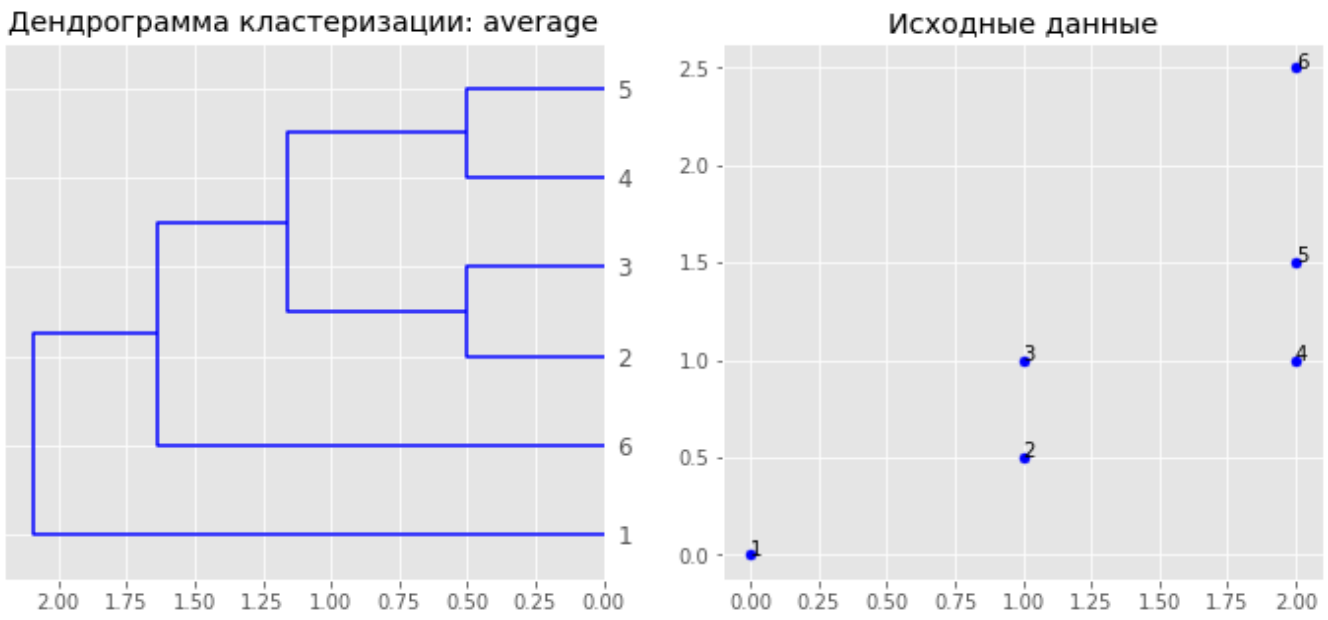
Очень информативна для принятия решений о том - какие метрики выбрать и на сколько кластеров делить - дендрограмма

применим разные метрики к нашей демо-задаче и посмотрим как объединяются объекты в кластеры

```
from scipy.cluster.hierarchy import linkage, fcluster, dendrogram
```

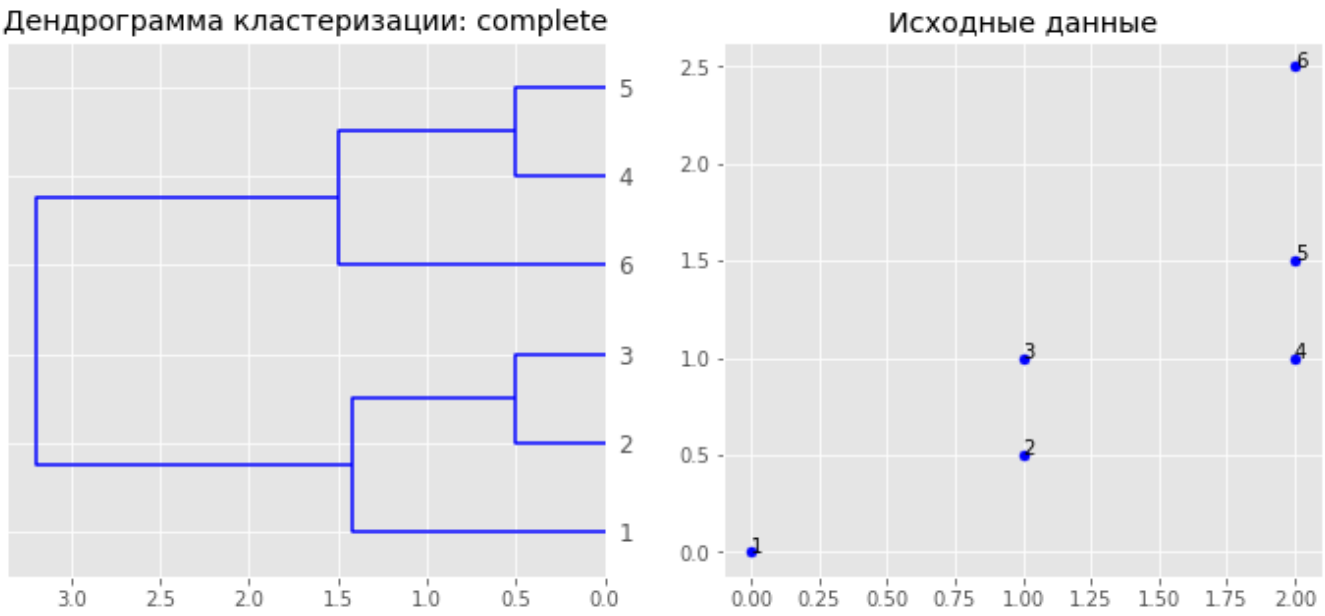
```
method='average'
Z = linkage(xy, method=method, metric = 'euclidean')
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
dend = dendrogram(Z, orientation='left', color_threshold=0.0, labels=labels, ax= ax[0])
ax[0].set_title('Дендрограмма кластеризации: '+ method)

ax[1].scatter(x, y, c='b', s=25)
for i in range(len(labels)):
    ax[1].text(x[i], y[i], s = labels[i])
```

```
ax[1].set_title('Исходные данные')
fig.show()
```

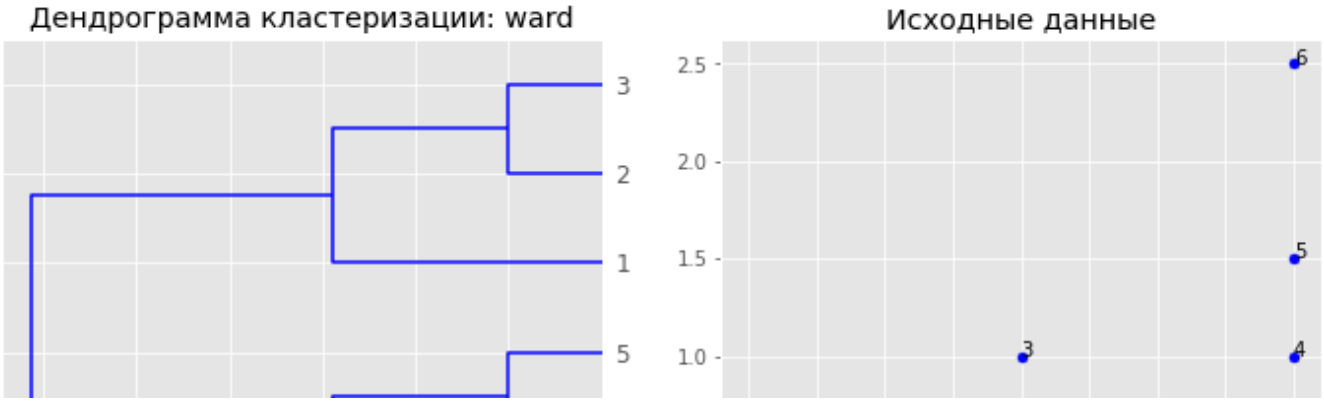



```
method='single'
Z = linkage(xy, method=method, metric = 'euclidean',)
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
dend = dendrogram(Z, orientation='left', color_threshold=0.0, labels=labels, ax= ax[0])
ax[0].set_title('Дендрограмма кластеризации: '+ method)

ax[1].scatter(x, y, c='b', s=25)
for i in range(len(labels)):
    ax[1].text(x[i], y[i], s = labels[i])
ax[1].set_title('Исходные данные')
fig.show()
```

```
method='complete'
Z = linkage(xy, method=method, metric = 'euclidean')
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
dend = dendrogram(Z, orientation='left', color_threshold=0.0, labels=labels, ax= ax[0])
ax[0].set_title('Дендрограмма кластеризации: '+ method)

ax[1].scatter(x, y, c='b', s=25)
for i in range(len(labels)):
    ax[1].text(x[i], y[i], s = labels[i])
ax[1].set_title('Исходные данные')
fig.show()
```




```
method='ward'
Z = linkage(xy, method=method, metric = 'euclidean',)
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
dend = dendrogram(Z, orientation='left', color_threshold=0.0, labels=labels, ax= ax[0])
ax[0].set_title('Дендрограмма кластеризации: '+ method)

ax[1].scatter(x, y, c='b', s=25)
for i in range(len(labels)):
    ax[1].text(x[i], y[i], s = labels[i])
ax[1].set_title('Исходные данные')
fig.show()
```



4. Применение метода к реальным данным

Рассмотрим предыдущий пример с изображениями цифр.

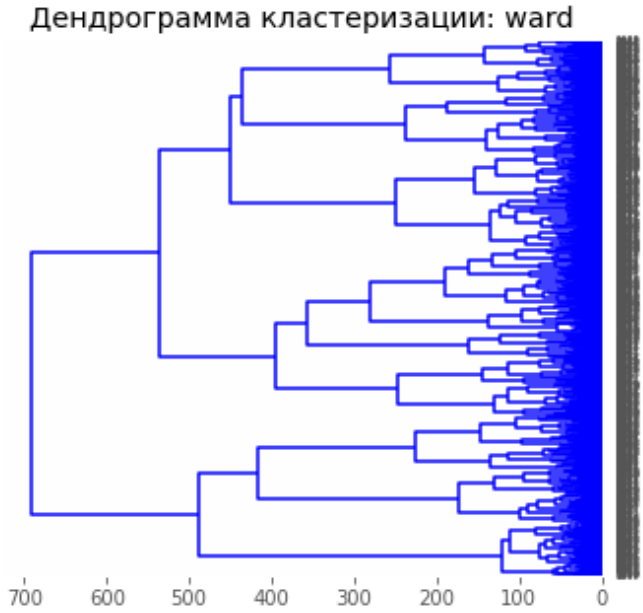
```
from sklearn.datasets import load_digits
from sklearn.manifold import TSNE

# загрузим картинки цифр
digit = load_digits()
img = digit.images
img.shape
```

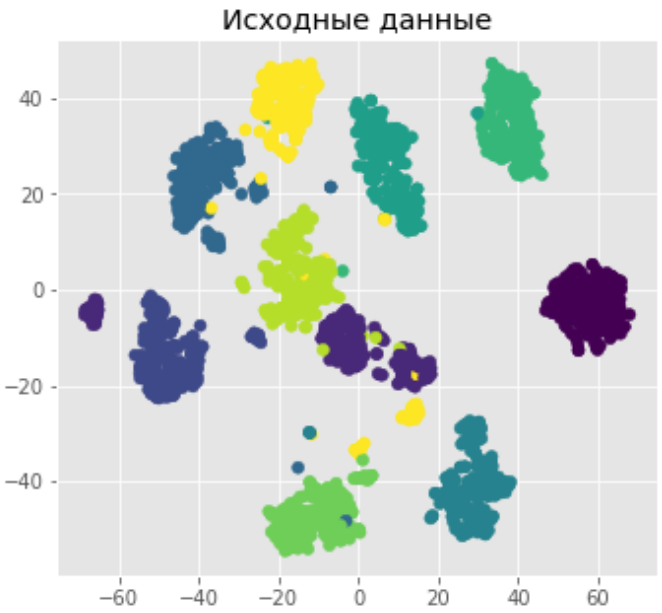
```
(1797, 8, 8)
```

```
# применим t-SNE
X = img.reshape(-1, 64)
tsne = TSNE(n_components=2, perplexity=30)
tsne.fit(X)
Z = tsne.embedding_
Z.shape
```

```
(1797, 2)
```

```
# применим иерархическую кластеризацию
method='ward'
Z2 = linkage(X, method=method, metric = 'euclidean',)
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
dend = dendrogram(Z2, orientation='left', color_threshold=0.0, ax= ax[0])
ax[0].set_title('Дендрограмма кластеризации: '+ method)

# визуализируем получившееся разбиение с учетом знания цифр на картинках
y = digit.target
ax[1].scatter(Z[:, 0], Z[:, 1], c=y)
ax[1].set_title('Исходные данные')
fig.show()
```



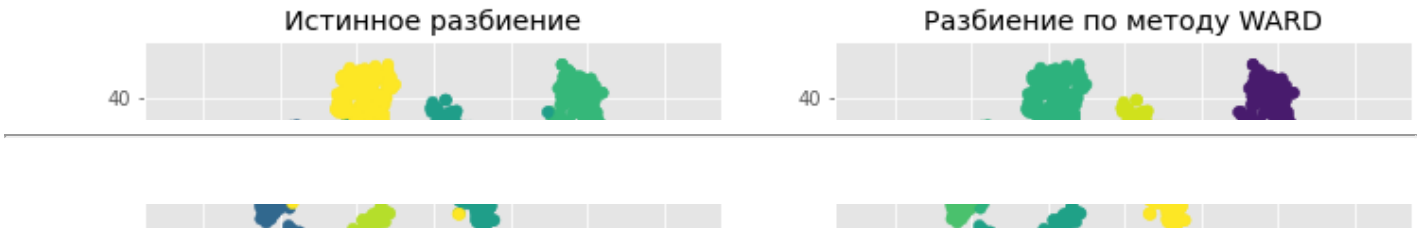


6. Выбор количества кластеров

```
labels = fcluster(Z2, 200, criterion='distance')
np.unique(labels)
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15],
      dtype=int32)
```

```
# визуализируем получившееся разбиение с учетом знания цифр на картинках
# и сравним с разбиением на кластеры, полученным с помощью иерархической кластеризации
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,5))
ax[0].scatter(Z[:, 0], Z[:, 1], c=y)
ax[1].scatter(Z[:, 0], Z[:, 1], c=labels)
ax[0].set_title('Истинное разбиение')
ax[1].set_title('Разбиение по методу WARD')
fig.show()
```



ВЫВОДЫ

- иерархическая кластеризация - это мощный инструмент анализа структуры данных;
- дендрограмма позволяет проанализировать имеющуюся структуру данных и определить оптимальное количество кластеров;
- метод сильно зависит от выбора метрики, определяющей расстояние между кластерами;

Уважаемые студенты!

Мы рады приветствовать Вас на курсе «Машинное обучение». Вы можете ознакомиться с построением курса и критериями его усвоения в книге ОПИСАНИЕ КУРСА.

В этом материале мы дадим краткое описание и рекомендации по изучению курса.

Курс состоит из четырех модулей. Каждый модуль включает в себя несколько тем.

Освоение каждой темы состоит из следующих работ:

- работа с лекциями, дополнительным теоретическим материалом;

- закрепление нового материала с использованием теста темы (5-10 тестовых заданий);

- выполнение практических заданий в среде VPL (Virtual Programming Lab) с автоматизированной проверкой программного кода на правильность и плагиат.

В конце модуля обучающийся выполняет итоговое задание раздела - кейсовое задание с использованием всего изученного материала.

Лекции дают систематизированные знания по дисциплине, концентрируют внимание на наиболее сложных и важных вопросах.

Подготовка к **занятиям практики** включает ознакомление с планом практического занятия; работу с конспектом лекций, работу с электронными ресурсами курса: прохождение теста темы, выполнение практических заданий текущей темы, работу с учебной и учебно-методической литературой курса (**Расширенный список литературы**). Также в начале курса вам дан список ссылок на полезные ресурсы для углубленного изучения по тематике курса (**Полезные ссылки**). Кроме этого в конце лекций также приводятся ссылки на ресурсы для более глубокого изучения темы.

В конце курса в **Дополнительном разделе** мы предоставим вам ссылки на дополнительные материалы для углубленного изучения.

При выполнении кейсов рекомендуется дополнительно познакомиться с рекомендуемыми научными статьями, рекомендованными в методических материалах темы.

Формой промежуточной аттестации по дисциплине является экзамен. Он может быть проведен в виде итогового теста.+ итоговое практическое задание (кейс).

УСПЕХОВ!

Тема 1. Инструмент решения задач

План уроков темы

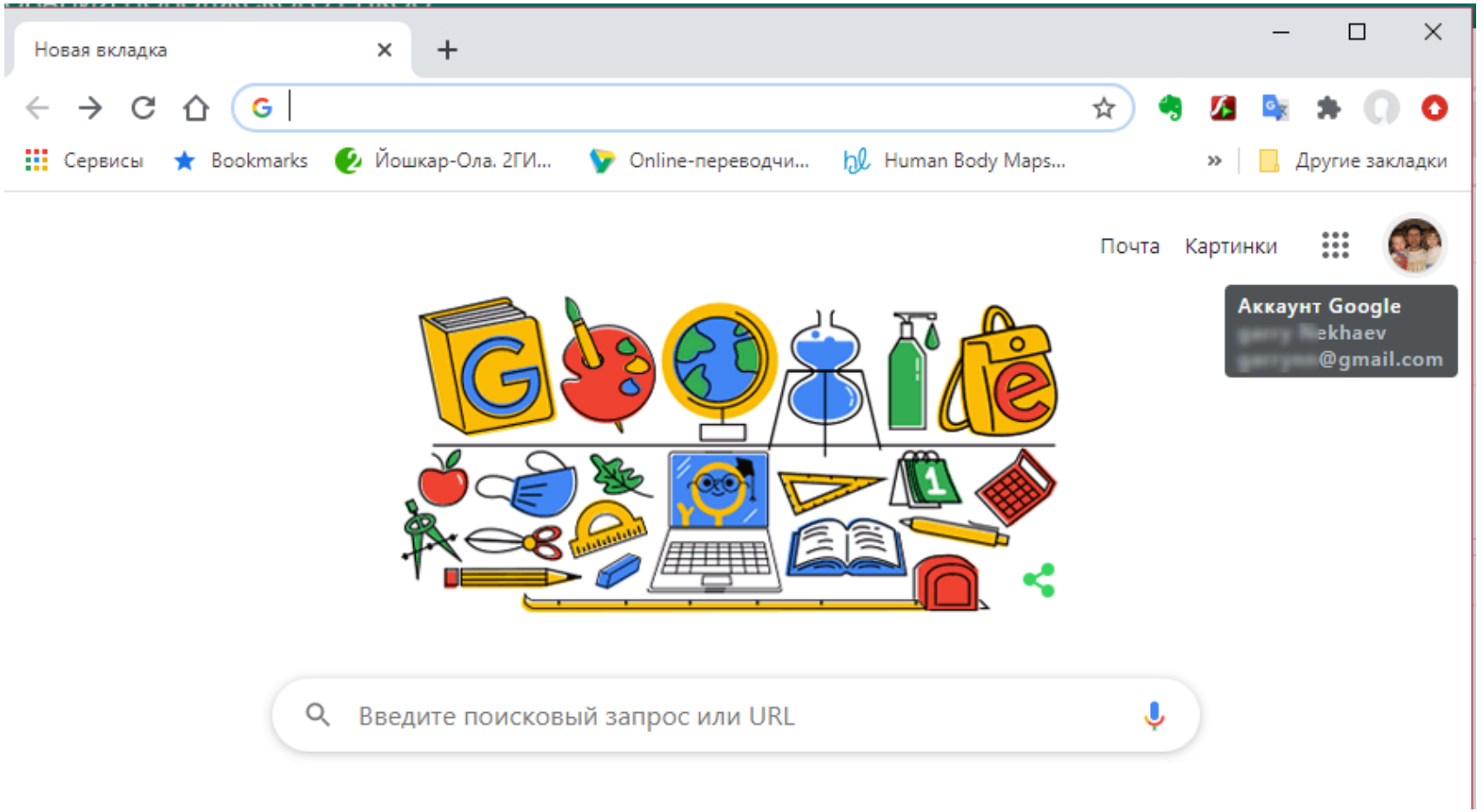
- ❑ История Python. Секрет популярности. Особенности программирования.
- ❑ Установка языка. Дзен Python-а. Создание и настройка виртуальных окружений.
- ❑ Установка Jupyter. Работа в Jupyter notebook локально.
- ❑ **Работа в облаке с Google Colab. Работа с Google Disk**

Создайте и зайдите в Google Account

Создать Google Account

Чтобы создать GA наберите в поисковой строке:
“**аккаунт Google**”

Если он уже у вас есть, то просто зайдите в него в любом браузере (лучше *Chrome* или *Mozilla*)

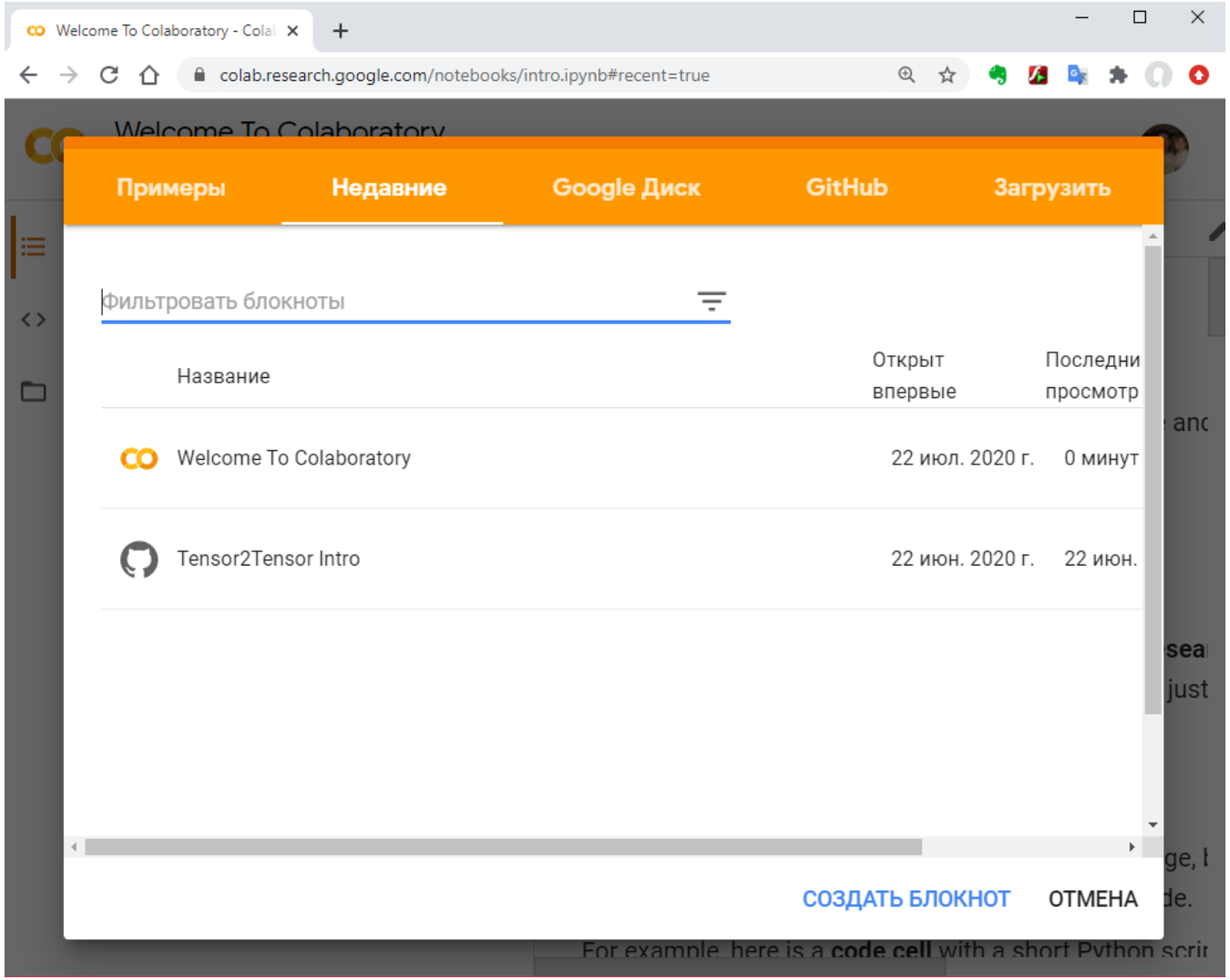


Зайдите в Google Colab

наберите в поисковой строке:

“**Google Colab**”

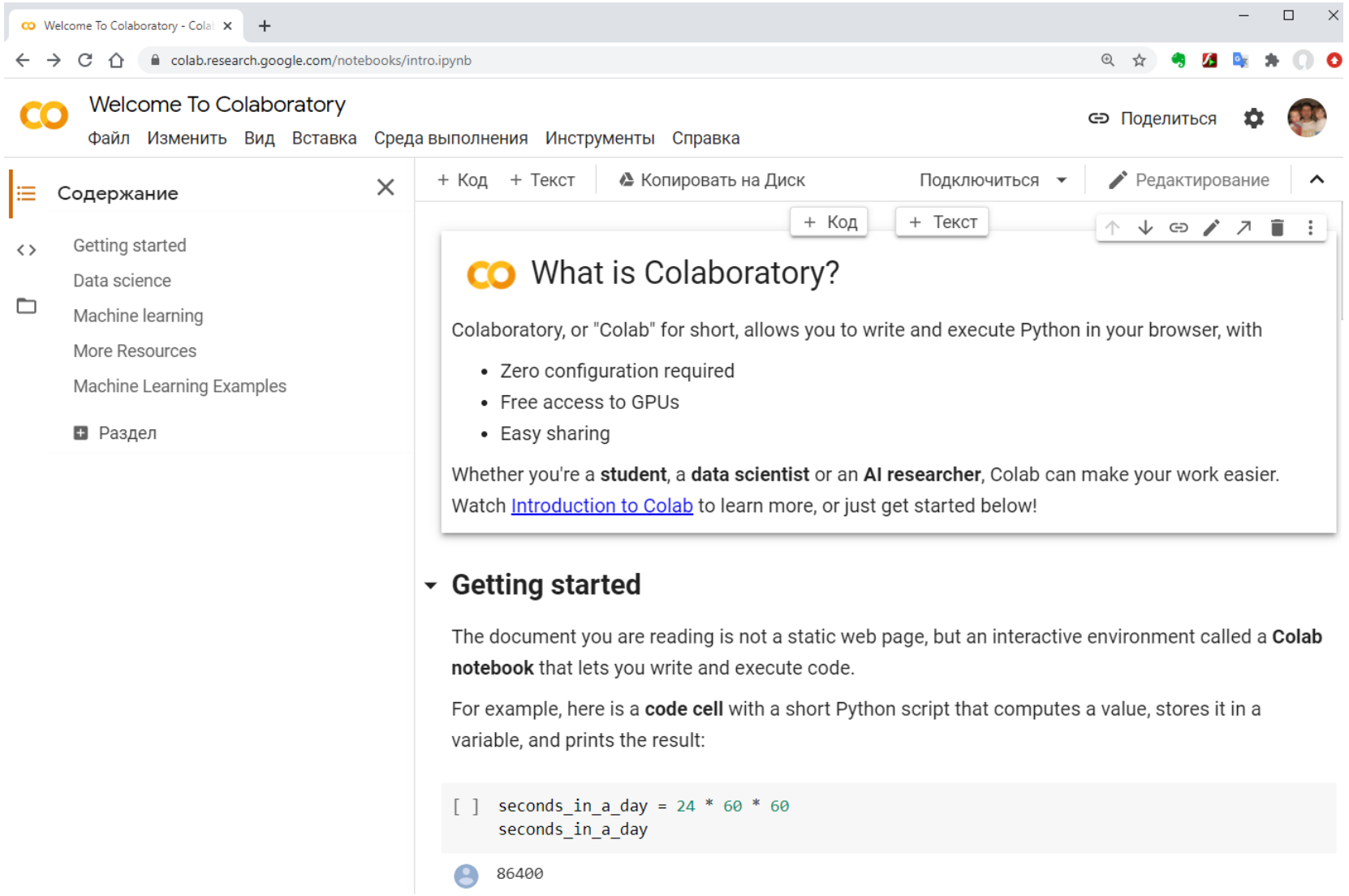
Зайдите в сервис и загрузите блокнот
«**Welcome To Colaboratory**»



Познакомьтесь с возможностями Colab

Вы можете писать и вычислять Python в браузере:

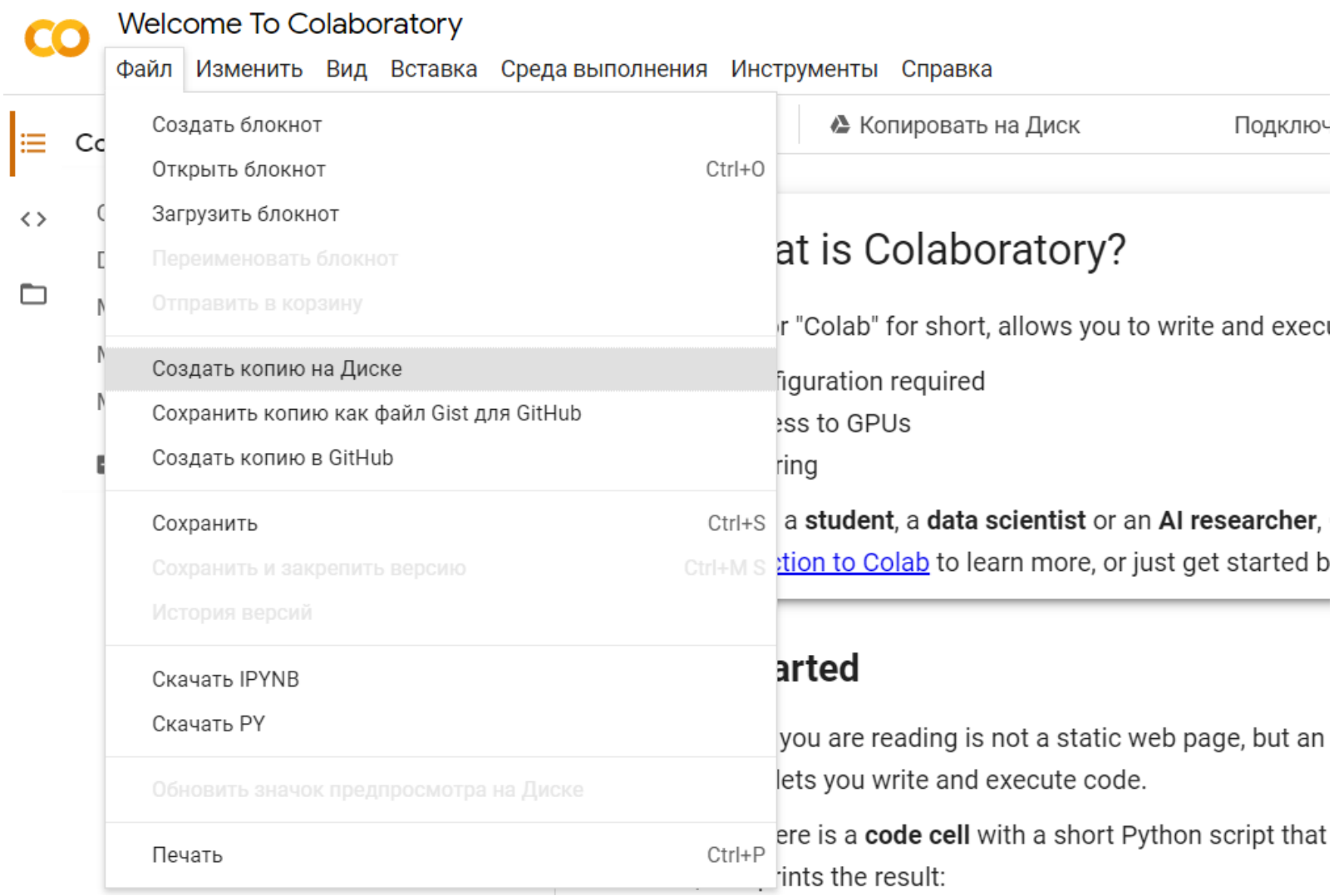
- без какой-либо настройки ПО;
- со свободным доступом к GPU;
- Легко расшаривая код и работая над ним совместно



Сохраните блокнот себе на диск

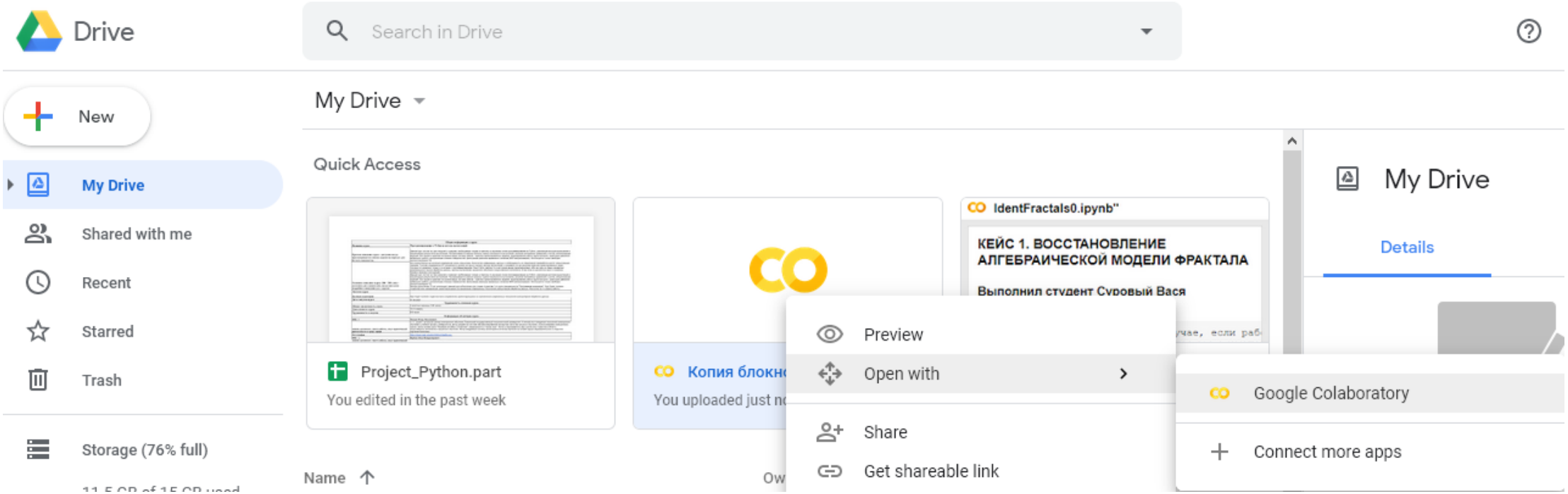
Вы можете:

- создавать новые блокноты,
- загружать блокноты из облака Colab и со своего Google диска;
- сохранять блокноты и копии на свой облачный диск;
- Экспортировать блокноты на локальный диск в формате ipynb или py
- Выводить на печать



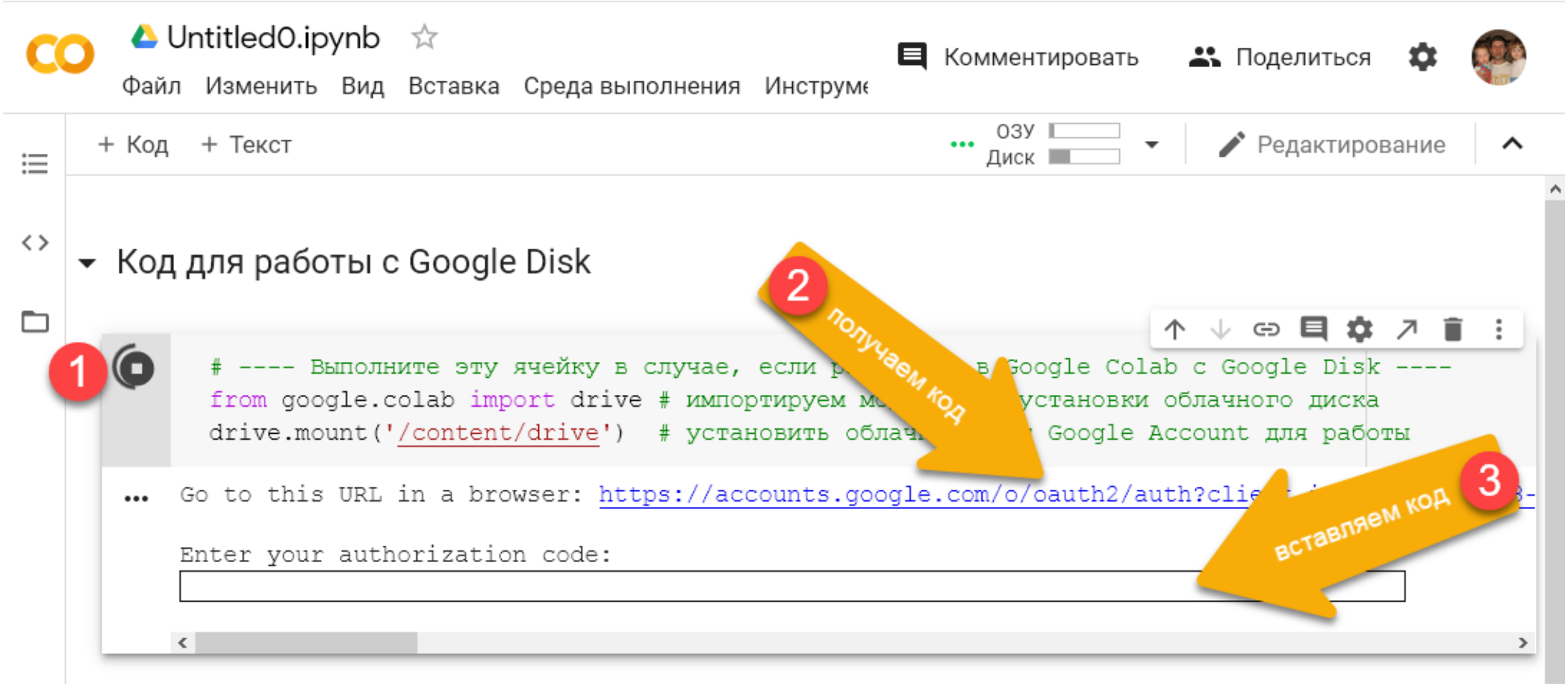
Свяжите расширение ipynb с Colab

1. **Зайдите на свой Google Disk**
2. **Найдите там файл «**Копия блокнота "Welcome To Colaboratory.ipynb**"»** (по умолчанию он находится в папке “**Colab Notebooks**”)
3. Кликните на него правой кнопкой мыши (контекстное меню) и укажите «Open with» «Google Colaboratory»

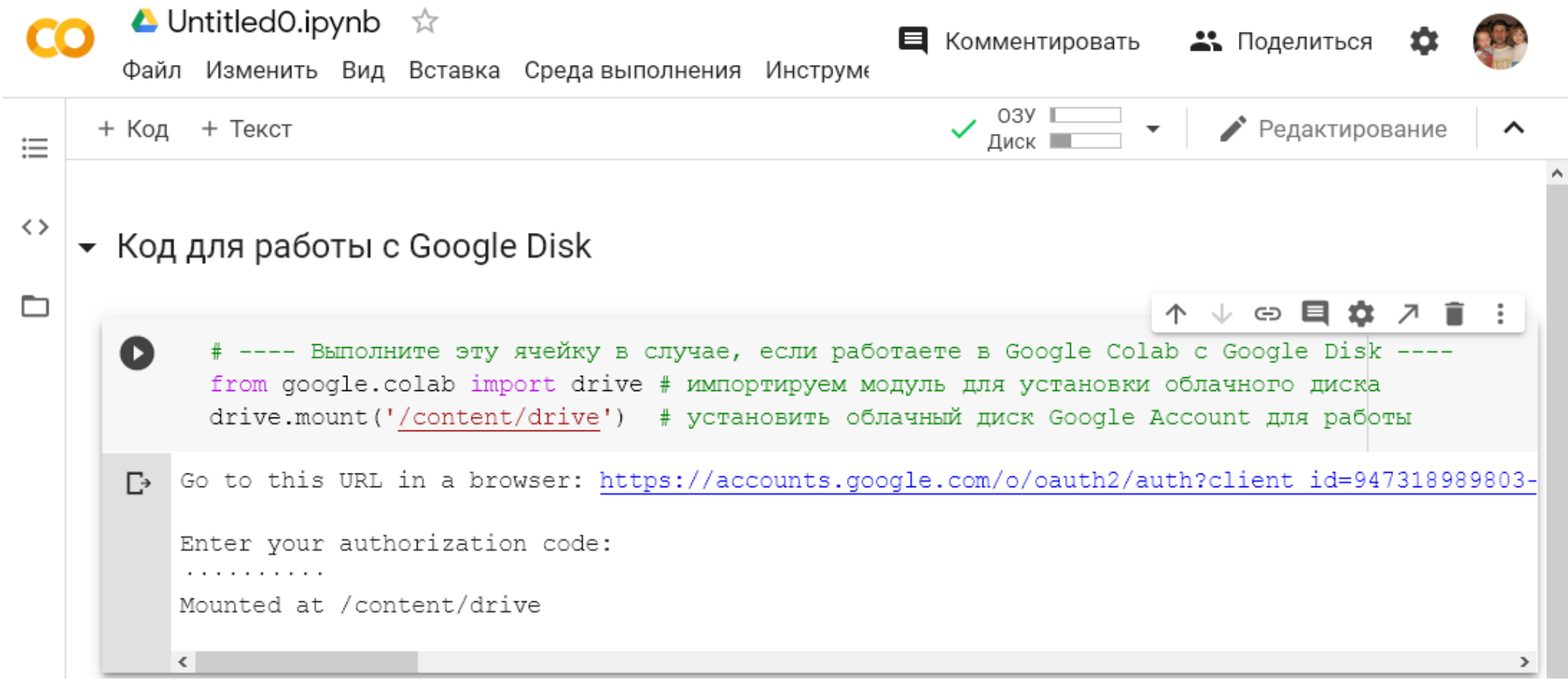


Для работы с Google Disk в Colab

1. Импортируйте из библиотеки **google.colab** модуль **drive** и вызовите функцию mount(‘/content/drive’)
2. Пройдите по появившейся ссылке и вставьте полученный код в соответствующее поле

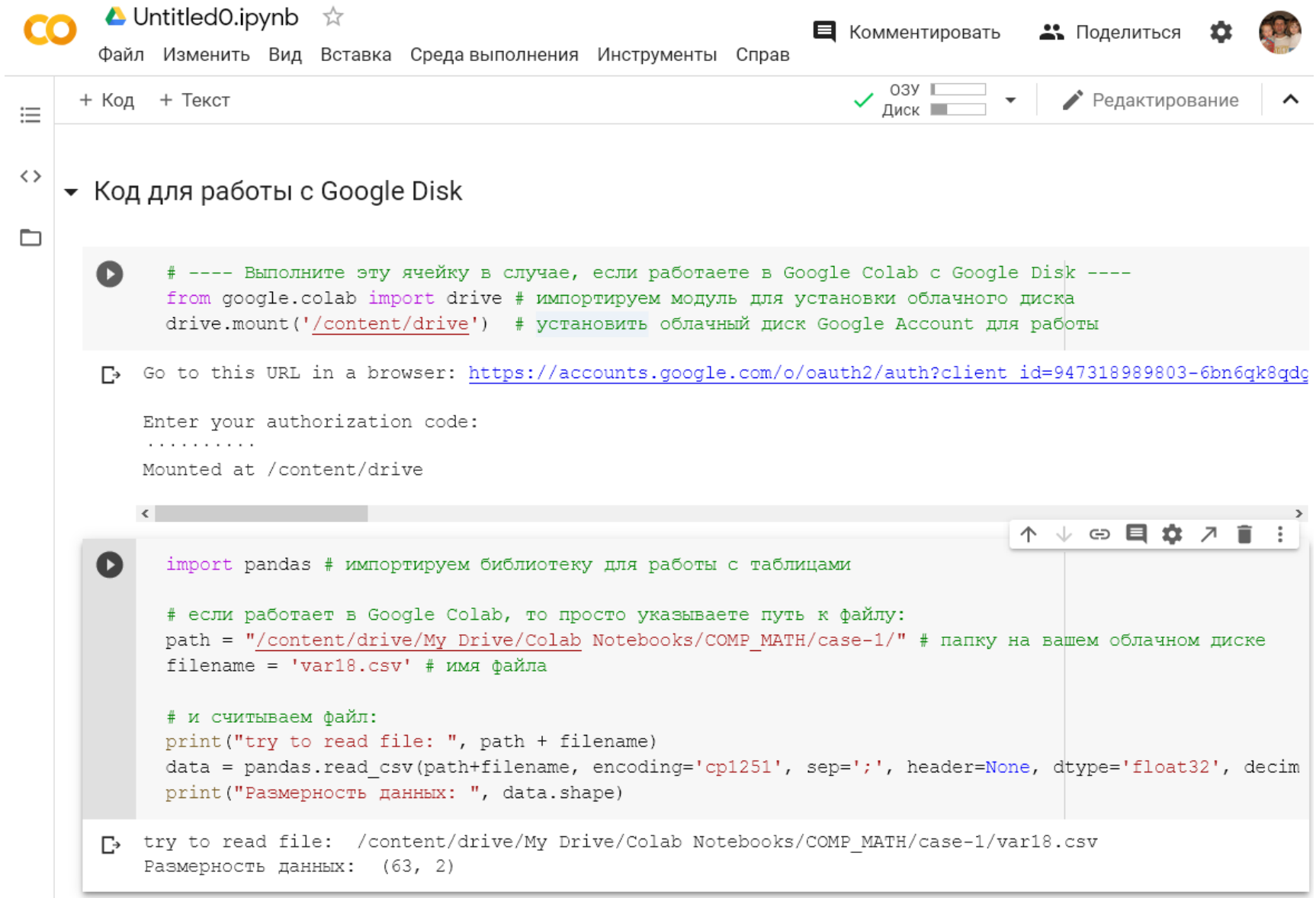


Установка Google Disk в Colab



Теперь вы можете работать со своим облачным диском как с обычным диском вашего компьютера

Работа с Google Disk в Colab



Например,

вы можете считать файл с данными, хранящемся в какой-нибудь папке на облачном диске;

Или импортировать из какой-либо папки дополнительные модули, предварительно добавив имя папки в пути с помощью **sys.path.append()**

Оформление отчетов

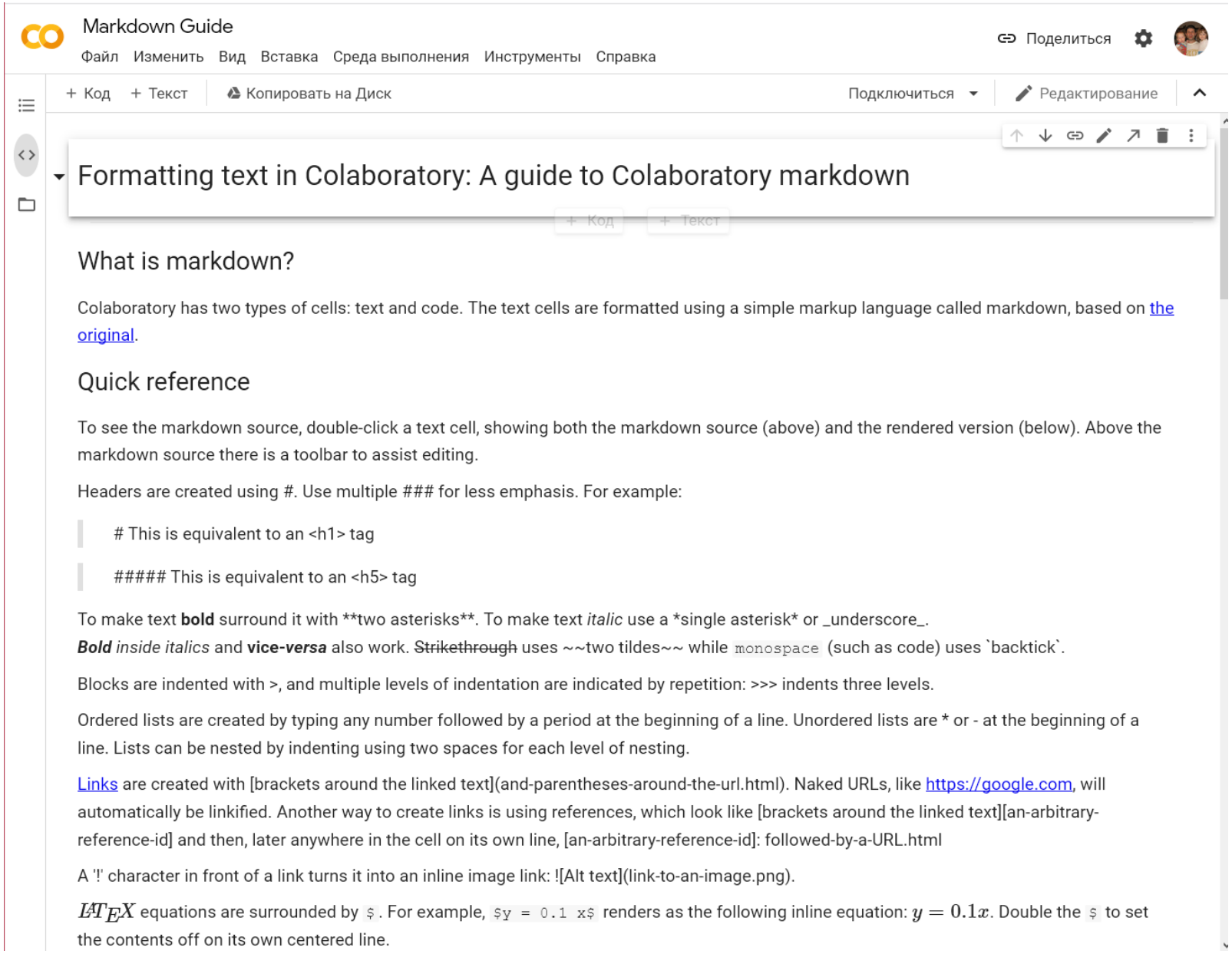
Оформление текста

Вставка формул и полная возможность использовать язык оформления **LaTex**;

Вставлять картинки

Вставлять ссылки

....



Итоги урока

Google Colab – это тот инструмент, о котором может только мечтать обучающийся, преподаватель и научный работник, исследователь.

Вы можете все свои данные, модули, блокноты хранить в облаке и в любой момент и в любом месте можете загрузить и выполнить их в **Google Colab** с использованием интернета и браузера.

Возможности оформления текста – такие же как у издательского учреждения, возможность же экспорта в формат **html** и в формат **pdf** делает процесс формирования отчетов легким и приятным.

Вы можете выгрузить блокнот на свой комп и продолжать работать с ним локально.

В ваших руках - очень мощный инструмент с неограниченными возможностями исследования данных и оформления результатов,

которым можно пользоваться без всякой установки ПО на компьютер или планшет.

Так давайте научимся им пользоваться!

Уважаемые преподаватели курса «Машинное обучение»!

Предлагаем Вашему вниманию взгляд разработчиков курса на организацию обучения студентов

Курс состоит из четырех модулей. Каждый модуль включает в себя несколько тем.

Освоение каждой темы предлагаем начать с аудиторной лекции, в которой обзорно рассматриваются самые важные теоретические положения темы. В помощь лектору в начале каждой темы имеется *конспект лекций*.

Рекомендуется придерживаться следующей организации работы студентов в рамках каждой темы:

- после проведения аудиторной лекции по теме студентам выдается задание пройти тест в рамках темы; лучше всего настраивать дедлайн по прохождению. теста - неделя после лекции и давать несколько (3-5) попыток и оценку по лучшей попытке, чтобы они не боялись работать с тестом; эта работа будет способствовать пониманию того, что им следует повторить и посмотреть дополнитеьно.

- попросить подготовиться к практическим занятиям - ознакомиться с дополнительным учебным материалом (методические указания или видеоуроки, скринкасты по освоению инструментальной части - python и функционала библиотек);

- практическое занятие лучше всего начинать с простого 15-минутного задания на проверку усвоенного материала при подготовке к практическим занятиям; всем, кто уложился в хронометраж, поставить баллы (собрать решения можно с помощью элемента задания в теме) и разобрать решение задачи. Тем самым устранить пробелы в подготовке к материалу практики, чтобы повысить эффективность проведения занятия;

- на практическом занятии разобрать необходимый материал для выполнения практических заданий темы;

- организовать консультирование при выполнении студентами практических заданий в среде VPL (Virtual Programming Lab) с автоматизированной проверкой программного кода на правильность как в рамках занятий так и за его пределами (с использованием мессенджеров или специализированного форума на курсе);

- обязательно проверять решения на плагиат, объявлять об аннулировании баллов при 100%-м плагиате на практических занятиях или в объявлении на курсе.

Тем, кто выполнит план по минимальному набору баллов на курсе (60% от максимума), предложить выполнить кейсовое задание курса (на отличную оценку).

При выполнении кейсов рекомендовать дополнительно познакомиться с научными статьями из списка источников.

Формой промежуточной аттестации по дисциплине является экзамен. Он может быть проведен в виде итогового теста.+ итоговое практическое задание (кейс).

УСПЕХОВ!

Про машинное обучение:

· **Курс лекций проф. Воронцова по машинному обучению**;

· Harrington. *Machine Learning in Action* — дается базовое знакомство с методами машинного обучения, без перегрузки математическими деталями

· Marshland. *Machine Learning: An Algorithmic Perspective* — приводятся и объясняются реализации разных методов машинного обучения на Python

· Richert, Coelho. *Building Machine Learning Systems with Python* — очень доступное изложение разных задач машинного обучения (анализ изображений, текстов, звука) с описанием того, как это сделать в Python (прямо с кодом)

· Есипов, Б. А. Методы исследования операций [Электронный ресурс] : учебное пособие / Б. А. Есипов. 2-е изд., испр. и доп. Санкт-Петербург: Лань, 2013. - 304 с. ISBN 978-5-8114-0917-4.

про нейросети:

· Хайкин, Саймон. Нейронные сети [Текст] : Полный курс / С. Хайкин ; [пер. с англ. Н. Н. Куссуль, А. Ю. Шелестова ; под ред. Н. Н. Куссуль]. 2-е изд. Москва [и др.]: Вильямс, 2006. - 1103 с.

· *Рутковская, Данута. Нейронные сети, генетические алгоритмы и нечеткие системы [Текст] / Д. Рутковская, М. Пилиньский, Л. Рутковский ; пер. с пол. И. Д. Рудинского. М.: Горячая линия - Телеком, 2007. - 383 с. ISBN 5-93517-103-1*

· *Роберт Каллан. Основные концепции нейронных сетей*

Полезные ссылки на онлайн-ресурсы:

Бесплатный курс Python на рус.яз. на Stepic, а также его продолжение - для начального знакомства

https://stepik.org/course/3356/syllabus - прекрасный практикум по Python и математике, если вы хотите усвоить тонкости.

https://docs.scipy.org/doc/ - оригинальная документация по NumPy и SciPy (англ.)

https://github.com/Kyubyong/numpy_exercises - упражнения по Numpy

http://www.labri.fr/perso/nrougier/from-python-to-numpy/ - очень подробный сайт про устройство **NumPy**. Полезно для оптимизации кода.

Погружение в Python - курс от физтеха на Coursera, если вам понравился язык и вы хотите серьезно его изучить.