**Методические указания   
к практическим работам по курсу**

**ОСНОВЫ машинного обучения**

[Введение в анализ данных на Python 2](#_Toc128237783)

[Основные понятия задачи регрессии 19](#_Toc128237784)

[Метод k ближайших соседей (KNN) 28](#_Toc128237785)

[Понижение размерности пространства признаков 34](#_Toc128237786)

[Сокращение размерности. Факторный анализ 37](#_Toc128237787)

[Визуализация данных и решающие деревья в Python 42](#_Toc128237788)

# Введение в анализ данных на Python

**Цель**: изучить особенности установки и настройки дистрибутива Anaconda, особенности интерфейса и принципов работы в Jupyter Notebook, работа с библиотекой Pandas языка программирования Python, ввод в анализ данных.

**Ход работы**

**Знакомство с интерфейсом Anaconda**

1. Для работы с данными удобнее использовать дистрибутив Anaconda.

Anaconda – дистрибутив языков программирования Python и R, включающий набор популярных свободных библиотек, объединённых проблематиками науки о данных и машинного обучения. Основная цель – поставка единым согласованным комплектом наиболее востребованных соответствующим кругом пользователей тематических модулей (таких как NumPy, SciPy, Astropy и других) с разрешением возникающих зависимостей и конфликтов, которые неизбежны при одиночной установке.

1. Зайдите на сайт https://www.anaconda.com/products/individual, скачайте версию для индивидуального использования Open Source, нажав по кнопке “Download” (рис. 1.1).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.1 – Установка Anaconda

1. Откроется окно регистрации (заполнять форму не надо), при этом файл будет скачен. Запустите установку. Пройдите по шагам установки, выбирая версию “Just me (recommended)”. При выборе папки для установки обратите внимание, чтобы **не было использована кириллица**, только английские символы. Дождитесь окончания установки.
2. Запустите программу Anaconda Navigator (рис. 1.2) – это графический интерфейс (GUI), включённый в дистрибутив Anaconda, позволяющий запускать приложения, устанавливать дополнительные пакеты и т.д. без использования командной строки Anaconda Prompt.

По умолчанию в Anaconda Navigator доступны следующие приложения:

* JupyterLab.
* Jupyter Notebook.
* QtConsole.
* Spyder.
* Glueviz.
* Orange.
* RStudio.
* Visual Studio Code.
* PyCharm CE.

Мы будем работать с Jupyter Notebook.

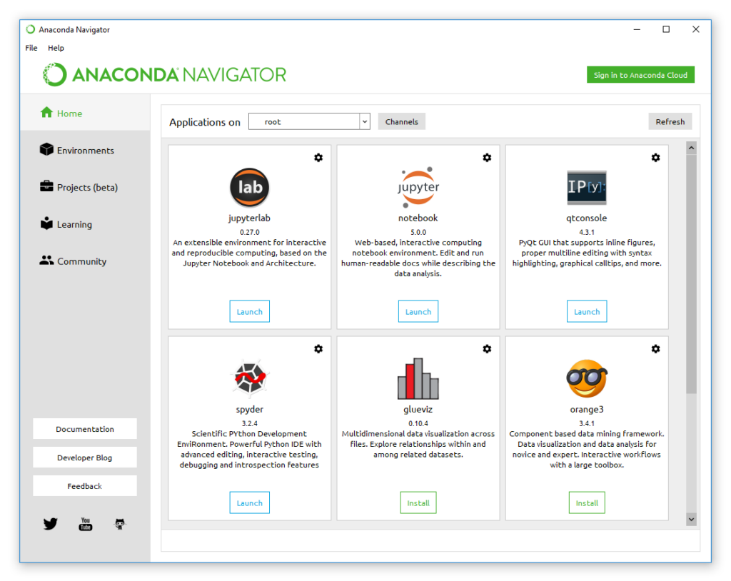


Рисунок 1.2 – Интерфейс Anaconda Navigator

1. Выберите в Anaconda Navigator программу Jupyter Notebook и нажмите по кнопке “Launch”.

В Jupyter notebook можно разрабатывать, документировать и выполнять приложения на языке Python, он состоит из двух компонентов: веб-приложение, запускаемое в браузере, и ноутбуки – файлы, в которых можно работать с исходным кодом программы, запускать его, вводить и выводить данные и т.п.

Веб-приложение позволяет:

* редактировать Python код в браузере, с подсветкой синтаксиса, автоотступами и автодополнением;
* запускать код в браузере;
* отображать результаты вычислений с медиа представлением (схемы, графики);
* работать с языком разметки Markdown и LaTeX.

Ноутбуки (или блокноты) – это файлы, в которых сохраняются исходный код, входные и выходные данные, полученные в рамках сессии. Фактически, он является записью процессе работы, но при этом позволяет заново выполнить код, присутствующий на нем. Ноутбуки можно экспортировать в форматы PDF, HTML.

1. Откроется новая вкладка в браузере с адресом http://localhost:8889/tree следующего вида (рис. 1.3):

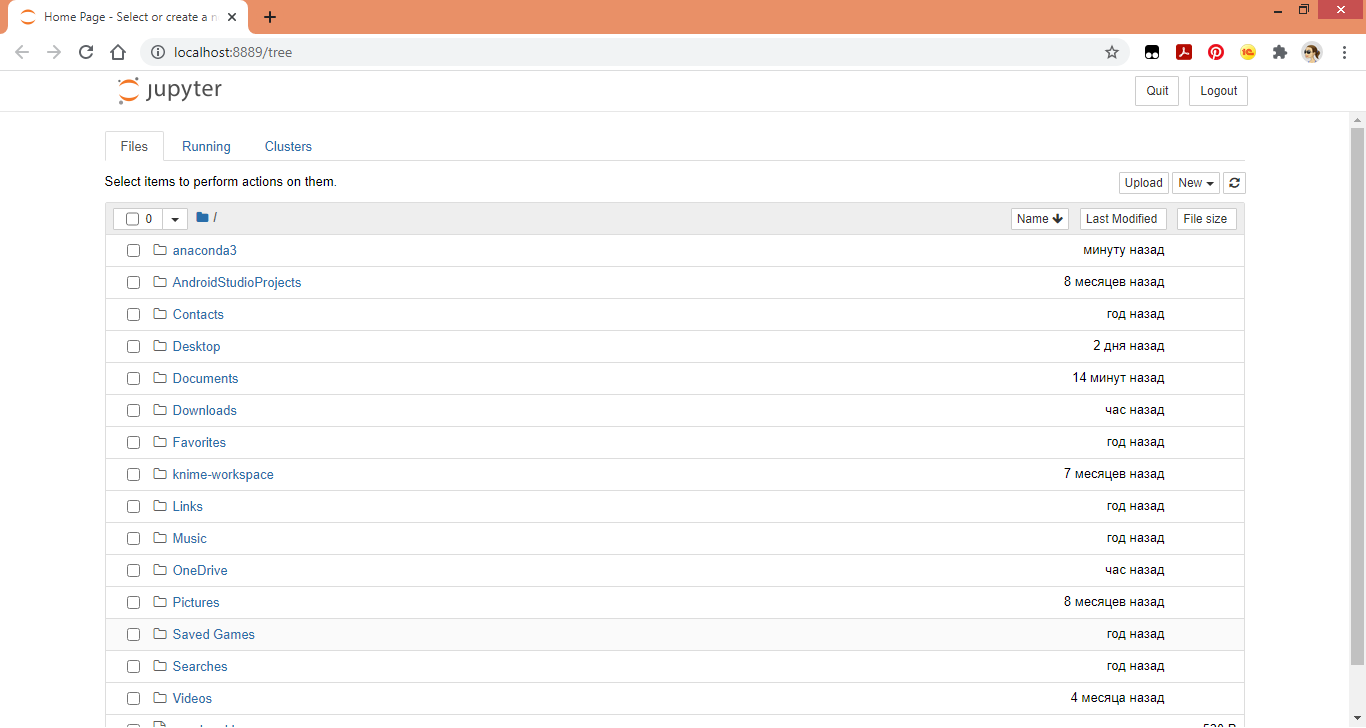


Рисунок 1.3 – Интерфейс Jupyter notebook

1. Запустите Jupyter notebook и создайте папку для лабораторных работ, для этого нажмите на New в правой части экрана и выберите в выпадающем списке Folder (рис. 1.4).

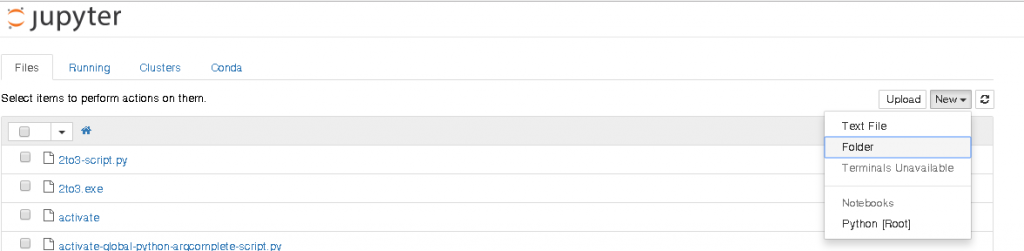


Рисунок 1.4 – Меню “New”

1. По умолчанию создастся папка с именем “Untitled folder”. Переименуйте ее: поставьте галочку напротив имени папки и нажмите на кнопку “Rename” вверху слева (рис. 1.5).

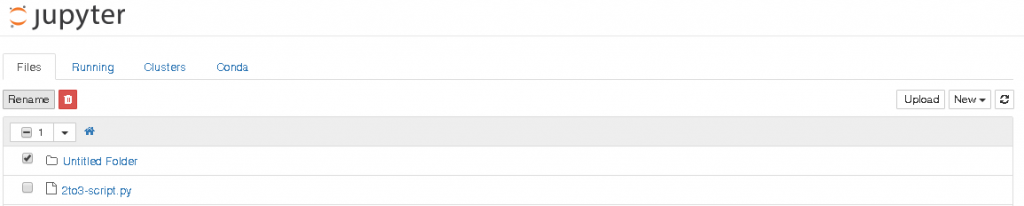


Рисунок 1.5 – Кнопка “Rename”

1. Зайдите в эту папку и создайте в ней ноутбук, воспользовавшись кнопкой New, выберите Notebook -> Python 3 (рис. 1.6).

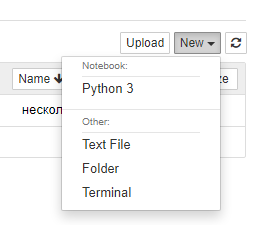


Рисунок 1.6 – Меню “New”

1. В результате будет создан ноутбук (рис. 1.7).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.7 – Ноутбук

1. Код на языке Python или текст в нотации Markdown нужно вводить в ячейки (рис. 1.8):

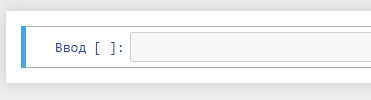


Рисунок 1.8 – Ячейки для ввода

1. Если это код Python, то на панели инструментов нужно выбрать из списка свойство “Код” (рис. 1.9).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.9 – Свойство Код

1. Если это Markdown текст – выставить “Markdown” (рис. 1.10).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.10 – Свойство Markdown

1. Для начала решим простую арифметическую задачу: выставите свойство “Код”, введите в ячейке “2 + 3” без кавычек и нажмите Ctrl+Enter или Shift+Enter, в первом случае введенный вами код будет выполнен интерпретатором Python, во втором – будет выполнен код и создана новая ячейка, которая расположится уровнем ниже так, как показано на рисунке 1.11. Можно воспользоваться кнопкой «Запуск» на панели инструментов.

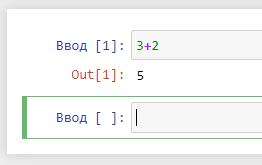


Рисунок 1.11 – Пример

1. Если у вас получилось это сделать, выполните еще несколько примеров (рис. 1.12).

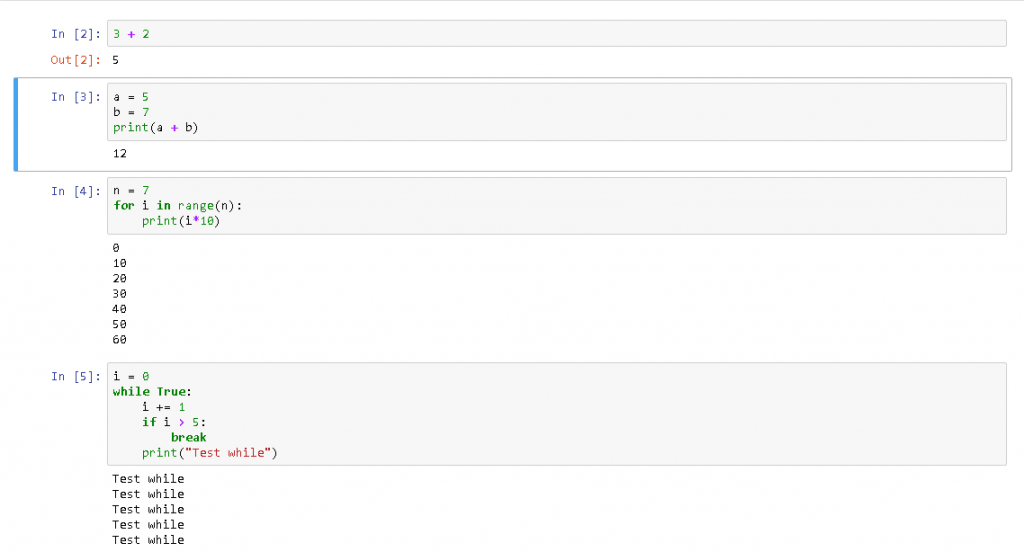


Рисунок 1.12 – Примеры для самостоятельной работы

**Основные элементы интерфейса Jupyter**

У каждого ноутбука есть имя, оно отображается в верхней части экрана. Для изменения имени нажмите на его текущее имя и введите новое (рис. 1.13).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.13 – Имя ноутбука

Из элементов интерфейса можно выделить, панель меню (рис. 1.14):



Рисунок 1.14 – Панель главного меню

панель инструментов (рис. 1.15):



Рисунок 1.15 – Панель инструментов

и рабочее поле с ячейками (рис. 1.16):

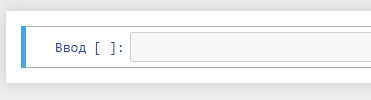


Рисунок 1.16 – Рабочее поле

Ноутбук может находиться в одном из двух режимов – это режим редактирования (Edit mode) и командный режим (Command mode). Текущий режим отображается на панели меню в правой части, в режиме правки появляется изображение карандаша, отсутствие этой иконки значит, что ноутбук находится в командном режиме (рис. 1.17).



Рисунок 1.17 – Режим работы

Для открытия справки по сочетаниям клавиш нажмите “Help” → “Keyboard Shortcuts” (рис. 1.18).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.18 – Меню “Help”

В самой правой части панели меню находится индикатор загруженности ядра Python. Если ядро находится в режиме ожидания, то индикатор представляет собой окружность (рис. 1.19).



Рисунок 1.19 – Индикатор загруженности ядра

Если оно выполняет какую-то задачу, то изображение измениться на закрашенный круг.

**Запуск и прерывание выполнения кода**

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

Для добавления новой ячейки используйте Insert → Insert Cell Above и Insert → Insert Cell Below.

Для запуска ячейки используете команды из меню Cell, либо следующие сочетания клавиш:

* Ctrl+Enter – выполнить содержимое ячейки.
* Shift+Enter – выполнить содержимое ячейки и перейти на ячейку ниже.
* Alt+Enter – выполнить содержимое ячейки и вставить новую ячейку ниже.

Можно сделать ноутбук доступным для других пользователей.

Существует несколько способов поделиться своим ноутбуком с другими людьми, причем так, чтобы им было удобно с ним работать:

* передать непосредственно файл ноутбука, имеющий расширение “.ipynb”, при этом открыть его можно только с помощью Jupyter Notebook;
* сконвертировать ноутбук в html;
* использовать https://gist.github.com/;
* использовать http://nbviewer.jupyter.org/.

**Вывод изображений в ноутбуке**

Печать изображений может пригодиться в том случае, если используется библиотеку matplotlib для построения графиков. По умолчанию, графики не выводятся в рабочее поле ноутбука. Для того чтобы графики отображались, необходимо ввести и выполнить следующую команду:

*%matplotlib inline*

Пример вывода графика представлен на рисунке 1.20.

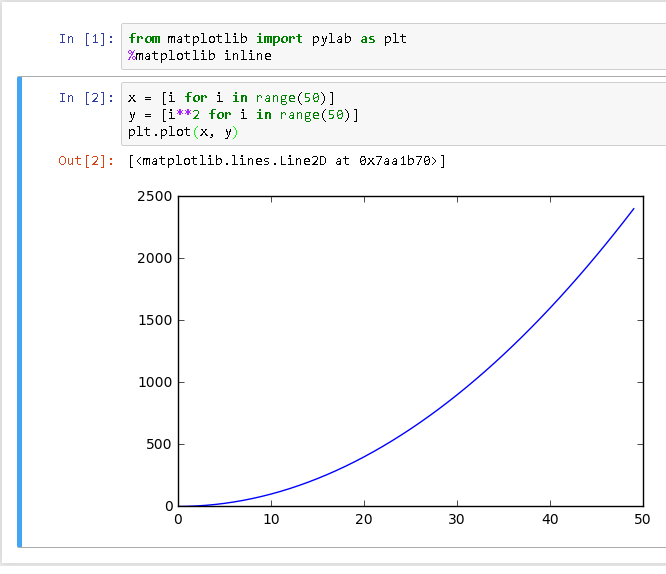


Рисунок 1.20 – Графики в Jupyter Notebook

**Команды и «магия» в Jupyter Notebook**

Важной частью функционала Jupyter Notebook является поддержка «магии». Под «магией» в IPython понимаются дополнительные команды, выполняемые в рамках оболочки, которые облегчают процесс разработки и расширяют возможности обработки данных. Список доступных команд можно получить с помощью команды (рис. 1.21):

*%lsmagic*

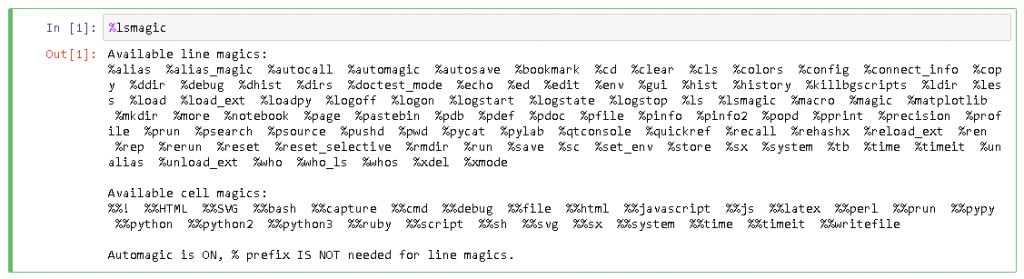


Рисунок 1.21 – Команды «магии»

Для работы с переменными окружения используется команда (рис. 1.22):

*%env.*



Рисунок 1.22 – Использование команды %env

Запуск Python кода из “.py” файлов, а также из других ноутбуков – файлов с расширением “.ipynb”, осуществляется с помощью команды (рис. 1.23):

*%run*.

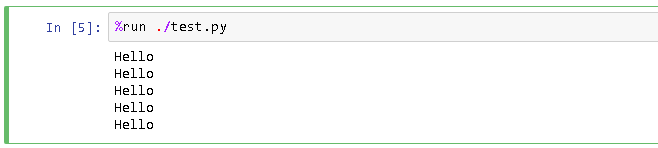


Рисунок 1.23 – Использование команды %run

Для измерения времени работы кода используется:

*%%time* и *%timeit*.

*%%time* позволяет получить информацию о времени работы кода в рамках одной ячейки (рис. 1.24).

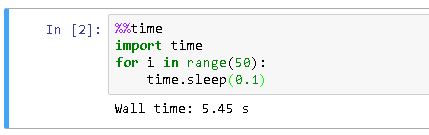


Рисунок 1.24 – Использование команды %%time

*%timeit* запускает переданный ей код 100000 раз (по умолчанию) и выводит информацию среднем значении трех наиболее быстрых прогонах (рис. 1.25).

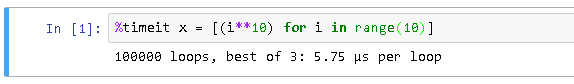


Рисунок 1.25 – Результаты работы команды %timeit

Информацию по остальным командам можно найти в следующих источниках:

1. https://ipython.org/ipython-doc/3/interactive/magics.html.
2. https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/.

Интересные примеры ноутбуков, в которых довольно полно раскрыты возможности Jupyter Notebook можно найти в ресурсах, перечисленных ниже.

1. http://nb.bianp.net/sort/views/.
2. https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-and-IPython-Notebooks.
3. https://blog.dominodatalab.com/interactive-dashboards-in-jupyter/.
4. http://www.clawpack.org/notebooks.html.
5. В самом конце панели откройте список и выберите пункт “Markdown”:
6. Выделенный блок превратится в блок текста. Набранный текст можно оформить *курсивом* или сделать его **жирным**. Больше информации по Markdown (средства оформления текста) можно найти по ссылке: https://gist.github.com/Jekins/2bf2d0638163f1294637.
7. На рисунке 1.26 представлены способы оформления текстовых блоков ноутбука (рис. 1.26):

[Изображение выглядит как текст

Автоматически созданное описание](https://neuralnet.info/wp-content/uploads/2017/12/jupyterlab_blocks_example.png)

Рисунок 1.26 – Примеры оформления текстовых блоков

1. Скопируйте на портале содержимое папки Lab1 (рис. 1.27).

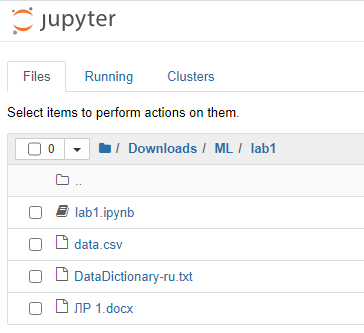


Рисунок 1.27 – Содержимое папки Lab1

1. Запустите блокнот lab1.ipynb.
2. Выполните все задания в блокноте.
3. Сохраните ноутбук с помощью меню “File” → “Save as” (рис. 1.30), указав в названии файла номер лабораторной работы и вашу фамилию.

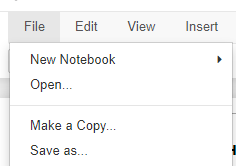


Рисунок 1.30 – Меню “File”

1. Результаты работы покажите преподавателю.

**Блокнот lab1.ipynb**

Лабораторная работа 1. Работа с Pandas

Pandas – это библиотека Python, предоставляющая широкие возможности для анализа данных (рис. 1.27). С ее помощью очень удобно загружать, обрабатывать и анализировать табличные данные с помощью SQL-подобных запросов.



Рисунок 1.27 – Подключение библиотеки Pandas

Основными структурами данных в Pandas являются классы Series и DataFrame. Первый из них представляет собой одномерный индексированный массив данных некоторого фиксированного типа. Второй – это двумерная структура данных, представляющая собой таблицу, каждый столбец которой содержит данные одного типа. Можно представлять её как словарь объектов типа Series.

С помощью библиотеки Pandas займемся анализом данных. Будем работать с данными о клиентах банка, который интересуется, произойдет ли просрочка платежа на 90 и более дней при выдаче кредита.

1. Прочтите данные из файла data.csv. Вводите код в блоки (рис. 1.28). Функции, которые могут пригодиться при решении:

pd.read\_csv(…, delimiter=’,’)



Рисунок 1.28 – Блоки для ввода

1. Выведите описание прочтенных данных. Функции, которые могут пригодиться при решении:

.describe()

3 Отобразите несколько первых и несколько последних записей. Функции, которые могут пригодиться при решении:

.head(), .tail()

Какие параметры можно передать этим функциям?

4. Прочтите в файле DataDictionary-ru.txt, что означают столбцы матрицы (рис. 1.28). Какому типу принадлежит каждый столбец (вещественный, целый, категориальный)?

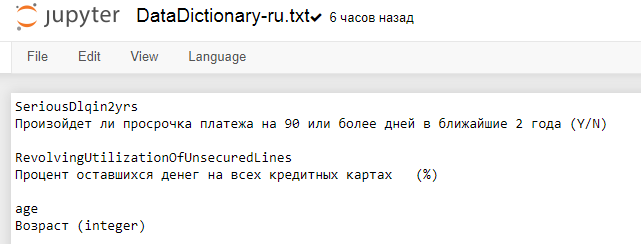


Рисунок 1.28 – Фрагмент содержимого файла

1. Заметьте, что столбец DebtRatio содержит неправдоподобные данные. Только значения, соответствующие известному месячному доходу, являются отношениями. Остальные – абсолютные значения месячных выплат процентов.

Исправьте данные, сделав все значения столбца DebtRatio абсолютными (умножьте их на MonthlyIncome). Для того чтобы программа быстро работала на полных данных, нужно не использовать цикл.

Функции, которые могут пригодиться при решении:

Обращение к элементам DataFrame:

* элемент: data.loc[i, 'названиеСтолбца'];
* столбец: data['названиеСтолбца'];
* подматрица: data.loc[a:b, списокНазванийСтолбцов].

Условная индексация:

* data.loc[data['столбец'] > 20, списокНазванийСтолбцов]

лучше писать так:

* i = data['столбец'] > 20 # вектор True и False
* data.loc[i, 'названиеСтолбца']

У подматриц номера строк наследуются от исходной.

* pandas.isnull(скаляр или массив) – проверка, является ли значение неопределенным (NaN);
* pandas.notnull(скаляр или массив) – проверка, является ли значение определенным (не NaN).

1. Поменяйте имя столбца на Debt. Функции, которые могут пригодиться при решении:

*.rename(columns={'староеИмя':'новоеИмя'}, inplace=True)*

1. Вычислите средний ежемесячный доход и присвойте всем клиентам с неизвестным доходом полученное число. Функции, которые могут пригодиться при решении:

*.mean()*

Другие описательные статистики можно найти по ссылке: https://pandas.pydata.org/pandas-docs/stable/reference/frame.html#computations-descriptive-stats.

1. Используя метод *groupby*, оцените вероятности невозврата кредита (SeriousDlqin2yrs=1) для различных значений количества иждивенцев (NumberOfDependents).

Проделайте аналогичную процедуру для различных значений столбца NumberRealEstateLoansOrLines.

Подсказка: *data['столбец1'].groupby(data['столбец2']).mean()* – расчет средних значений столбца1 по группам из столбца2.

1. Для визуализации данных можно ипользовать следующие функции (рис. 1.29).

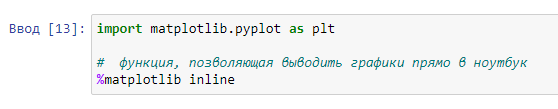


Рисунок 1.29 – Функции для визуализации данных

Matplotlib позволяет удобно визуализировать табличные данные.

Функции, которые могут пригодиться при решении:

Рисование:

* plt.plot(x, y);
* plt.show();
* plt.scatter(x, y);
* plt.hist().

Рисование нескольких графиков на одном:

* fig, ax = plt.subplots();
* ax.hist(...);
* ax.hist(...);
* plt.show();

Логарифмическая шкала:

* ax.set\_xscale('log') или ax.set\_yscale('log').

Ограничение области графика:

* ax.axis([x1, x2, y1, y2]).

9. Постройте график рассеяния на осях age и Debt. Синим цветом отметьте клиентов без серьезных задолженностей (SeriousDlqin2yrs = 0) и красным – должников (SeriousDlqin2yrs = 1).

# Основные понятия задачи регрессии

**Цель**: ознакомиться с особенностями решения задачи регрессии в машинном обучении и ее реализации на языке программирования Python в Jupyter Notebook.

**Ход работы**

1. Скачайте с портала содержимое папки Lab2. Разместите ее в ранее созданной папке для лабораторных работ.
2. Запустите программу Anaconda Navigator (рис. 2.1).

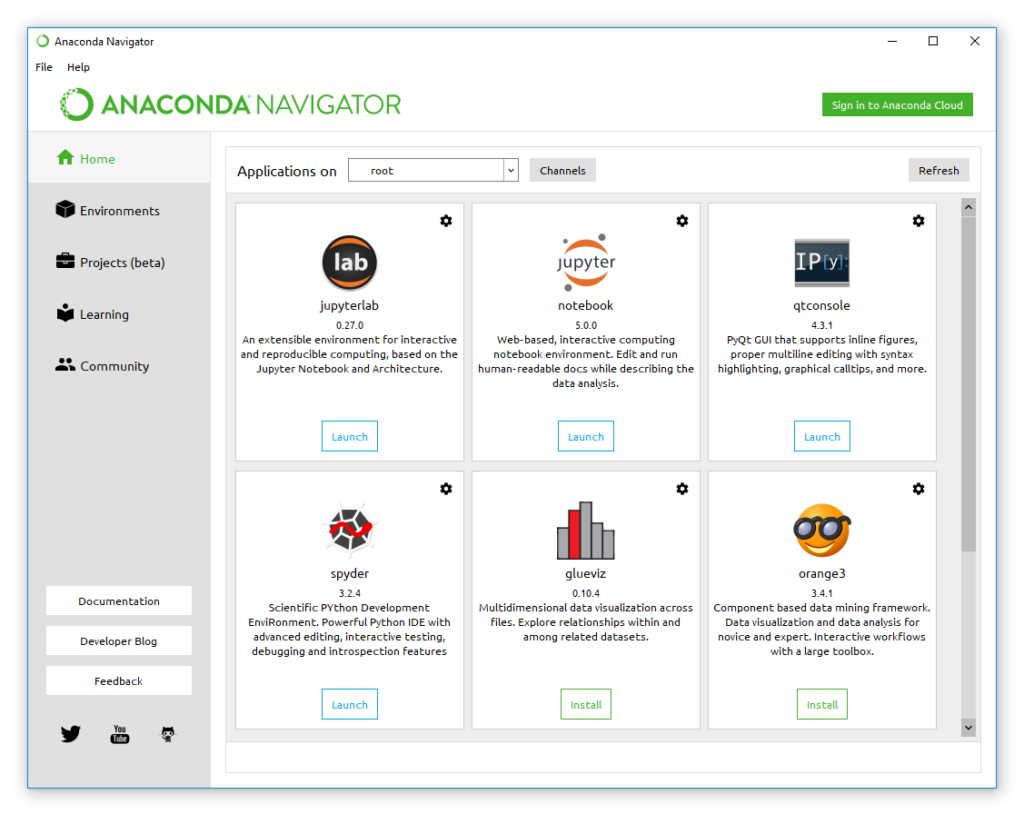


Рисунок 2.1 – Интерфейс Anaconda Navigator

1. Выберите в Anaconda Navigator программу Jupyter Notebook и нажмите по кнопке “Launch”.
2. Откроется новая вкладка в браузере с адресом http://localhost:8889/tree следующего вида (рис. 2.2):

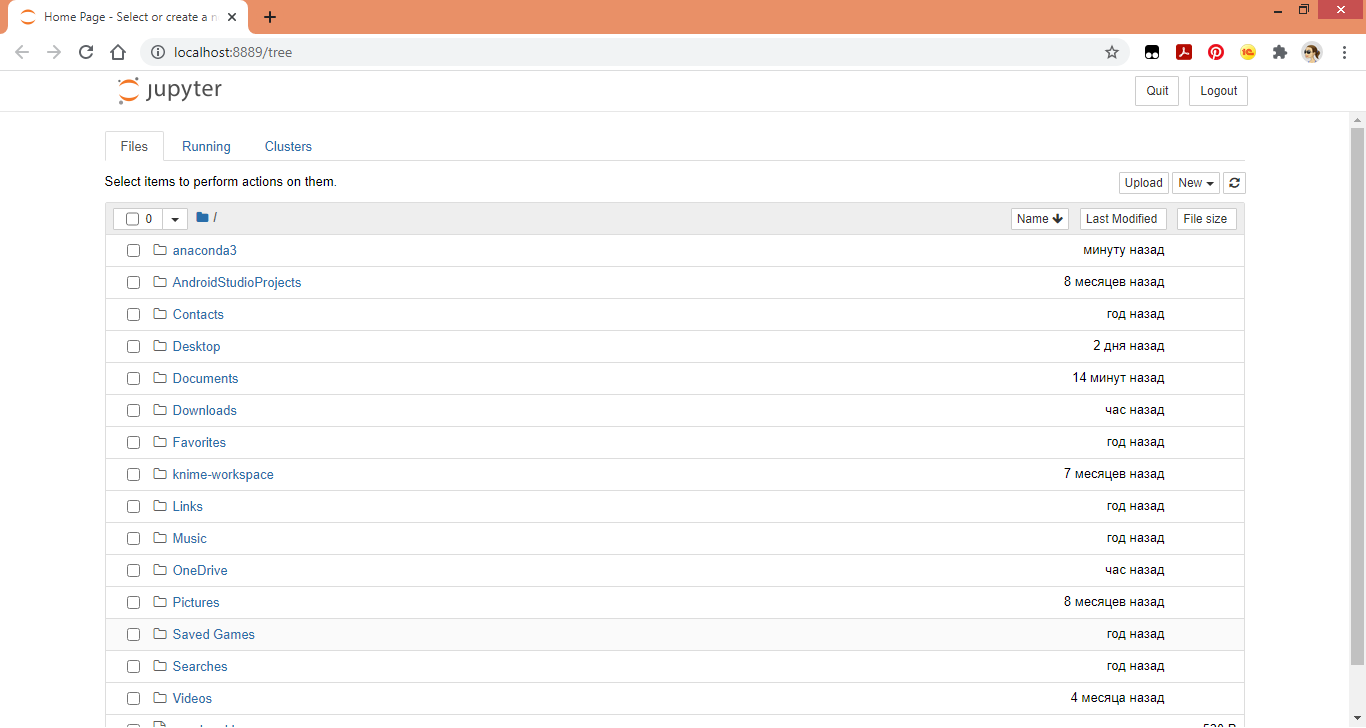


Рисунок 2.2 – Интерфейс Jupyter notebook

1. Запустите Jupyter notebook и откройте ранее созданную папку для лабораторных работ на вкладке Files и в ней папку Lab2.
2. Откройте в папке ноутбук lab2.ipynb. Далее продолжайте работу в этом файле.
3. Сохраните ноутбук с помощью меню “File” → “Save as”, указав в названии файла номер лабораторной работы и вашу фамилию.
4. Результаты работы покажите преподавателю.

**Блокнот lab2.ipynb**

Попробуем решить задачу регрессии, и предсказать число прокатов велосипедов в зависимости от погоды. Описание всех признаков:

* season: 1 – весна, 2 – лето, 3 – осень, 4 – зима;
* yr: 0 – 2011, 1 – 2012;
* mnth: от 1 до 12;
* holiday: 0 – нет праздника, 1 – есть праздник;
* weekday: от 0 до 6;
* workingday: 0 – нерабочий день, 1 – рабочий день;
* weathersit: оценка благоприятности погоды от 1 (чистый, ясный день) до 4 (ливень, туман);
* temp: температура в Цельсиях;
* atemp: температура по ощущениям в Цельсиях;
* hum: влажность;
* windspeed(mph): скорость ветра в милях в час;
* windspeed(ms): скорость ветра в метрах в секунду;
* cnt: количество арендованных велосипедов (это целевой признак, его мы будем предсказывать).

Предсказывать мы будем cnt, все остальные значения – нецелевые.

1. Загрузите данные о прокате. Посмотрите на «сырые» данные из файла bikes\_rent.csv, разделитель ','. Код вводите в блоки (рис. 2.3).

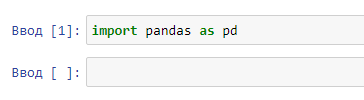


Рисунок 2.3 – Блоки для ввода кода

1. Как видно, все данные в дата-сете разного масштаба. Необходимо привести все к одному масштабу, чтобы не было большого разброса значений. Это необходимо в связи с тем, что, как в линейной регрессии используются взвешенные суммы признаков, соответственно, если их значения будут сильно отличаться друг от друга, то это приведет к расхождениям при обучении градиентным спуском.

Изучим распределение признаков, используя графики из библиотеки seaborn. Seaborn это некая «обертка» над matplotlib, позволяющая строить чуть более красивые графики.

Можно воспользоваться функцией для построения графиков kdeplot. Данный график показывает распределение плотности случайной величины, за которую мы берем столбец таблицы. Результат работы на рис. 2.4.

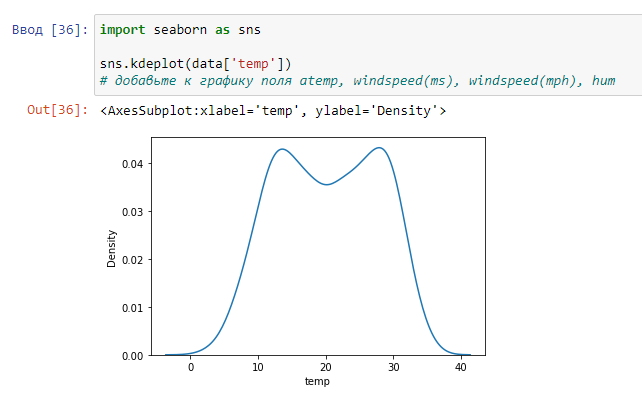


Рисунок 2.4 – Результат работы функции

С одной стороны, по графику видно, что величины укладываются в один масштаб. Однако, если отразить на графике вместе не только вещественные (тип float) признаки, но и категориальные, то увидим, что все данные имеют разный масштаб, и в таком случае предсказать что-то не представляется возможным.

Обратите внимание, что мы визуализируем только нецелевые признаки – так как значение cnt будем предсказывать, то его тут не учитываем.

Метод DataFrame.drop удаляет нужную колонку из датафрейма, и возвращает новый датафрейм, уже без удаляемой колонки. Так как целевой признак cnt, то именно его необходимо удалить. Фрагмент кода на рис. 2.5.

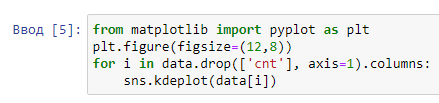


Рисунок 2.5 – Фрагмент кода

Новый график представлен на рис. 2.6.

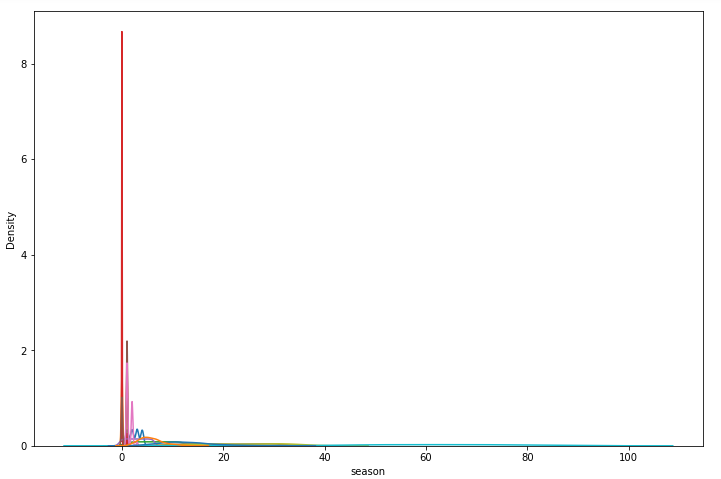


Рисунок 2.6 – График функции

Напишем небольшую собственную функцию StandardScaler. Суть ее работы: все приводится к нулевому среднему значению, и единичному отклонению.

Формула: xi = (xi – mean(X))/std(X),

где mean – среднее, std – среднеквадратичное отклонение.

Соответственно после масштабирования Mean(X) = 0 и Std(X) = 0. Реализация на рис. 2.7.

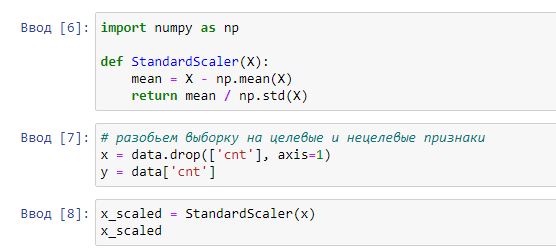


Рисунок 2.7 – Код

Фрагмент масштабированного датафрейма представлен на рис. 2.8.



Рисунок 2.8 – Масштабированные значения

Снова визуализируем график с помощью *plt.figure*, только теперь для *x\_scaled* (рис. 2.9). Программный код на рис. 2.10. Полученный график на рис. 2.11.

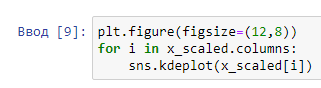


Рисунок 2.10 – Программный код

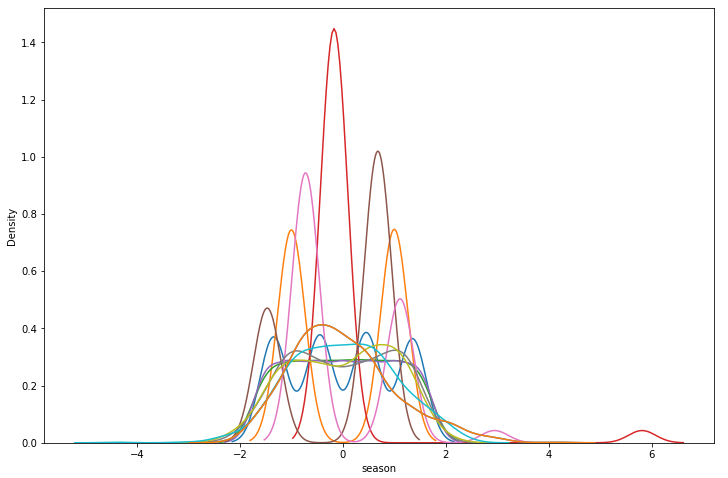


Рисунок 2.11 – График масштабированных данных

Проверим среднее и отклонение. Выведите на печать значения mean и std из np для значений x\_scaled (рис. 2.12).

Поле values классов DataFrame или Series вернет numpy-массив.

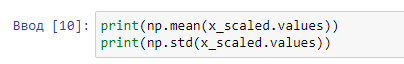


Рисунок 2.12 – Программный код

Среднее должно быть околонулевым значением, а отклонение равно единице.

**Регрессия**

1. Переименуем переменные для удобства, и превратим данные в numpy-массивы (рис. 2.13).

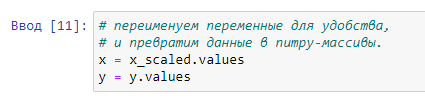


Рисунок 2.13 – Программный код

Импортируем из библиотеки sklearn все модели машинного обучения:

*from sklearn import \**

Импортируем класс LinearRegression из sklearn.linear\_model Класс sklearn.linear\_model.LinearRegression используется для линейной регрессии и прогнозов.

Трактуя задачу о предсказании наличия недостатков как задачу регрессии, натренируем линейную модель LinearRegression на подмножестве признаков. Функции, которые могут пригодиться при решении:

* создание модели: *model = linear\_model.LinearRegression()*;
* тренировка: *model.fit(x, y)*.
  1. Выполните предсказание для всех объектов обучающей выборки и присвойте результат переменной prediction. Функции, которые могут пригодиться при решении:

*model.predict()*.

Рассчитайте TSS – это общая сумма квадратов отклонений; RSS – это объяснённая сумма квадратов отклонений.

*RSS = ((y – prediction) \*\* 2).sum() TSS = ((y – y.mean()) \*\* 2).sum()*.

Проверьте правильность предсказаний, создав новую переменную, которая рассчитывается по формуле:

*1-RSS/TSS*.

Сравните расчет правильности предсказания модели по встроенной функции *score* для модели, выведя их на экран.

* 1. Преобразуйте получившийся вектор предсказаний *prediction* к значениям {0,1}. Это можно сделать, например, используя list comprehensions: https://docs.python.org/3/tutorial/datastructures.html#list-comprehensions.

*predictionClass = [1 if prediction[i] > 0.5 else 0 for i in range(prediction.shape[0])].*

* 1. Постройте отчет по качеству классификации и матрицу ошибок. Как изменятся отчет и матрица ошибок, если изменить порог в задании 3.2 (по умолчанию его значение равно 0.5)?

Функции, которые могут пригодиться при решении:

*print(metrics.classification\_report(...)), print(metrics.confusion\_matrix(...))*.

Первый параметр функций – y (исходная целевая функция), второй – предсказанный класс. Функция *classification\_report* для каждого класса объектов считает точность (precision) в этом классе и полноту (recall). Полнота – это процент объектов данного класса, которые ваш метод предсказания тоже отнес к этому классу, среди всех объектов данного класса. Точность (precision) – то же самое, только среди всех объектов, предсказанных для этого класса.

Функция *confusion\_matrix* возвращает матрицу с количествами объектов. Номера столбцов матрицы – это номера предсказанных классов, строки – это номера правильных классов. Например, элемент M[0,1] – это количество элементов, где на самом деле 0, а вы предсказали 1.

# Метод k ближайших соседей (KNN)

**Цель**: ознакомиться с особенностями метода ближайших соседей (kNN) в машинном обучении и его реализации на языке программирования Python в Jupyter Notebook.

**Ход работы**

1. Скачайте с портала содержимое папки Lab3. Разместите ее в ранее созданной папке для лабораторных работ.
2. Запустите программу Anaconda Navigator (рис. 3.1).

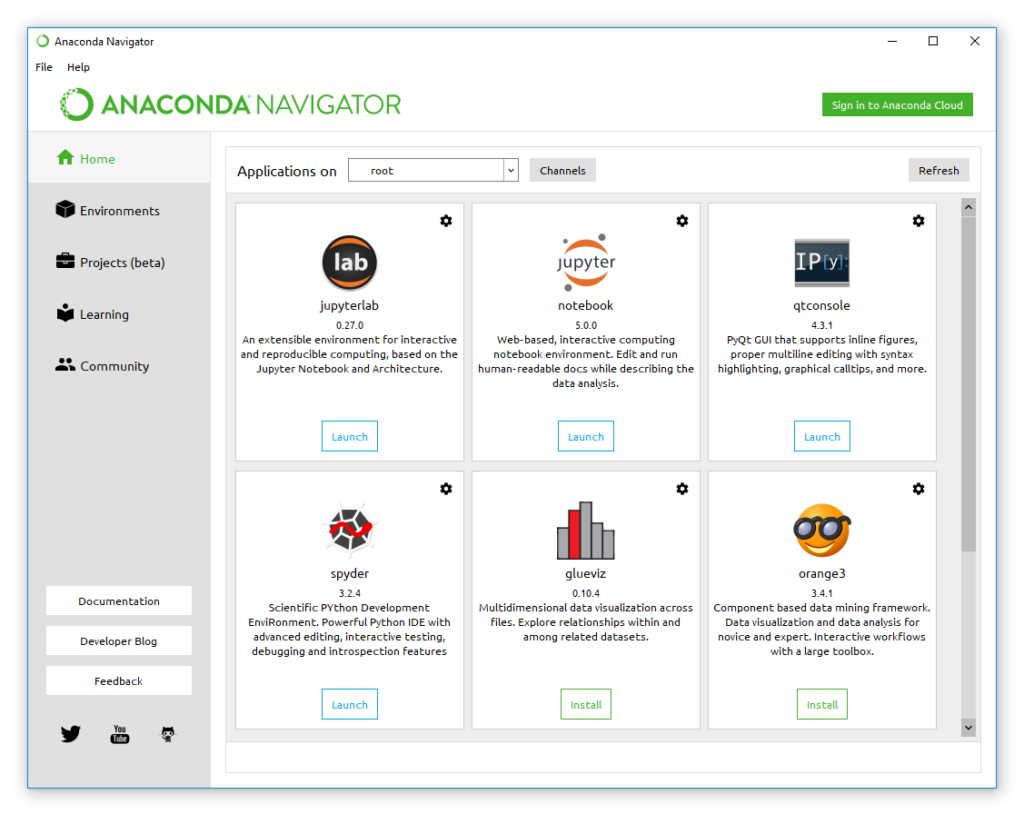


Рисунок 3.1 – Интерфейс Anaconda Navigator

1. Выберите в Anaconda Navigator программу Jupyter Notebook и нажмите по кнопке “Launch”.
2. Откроется новая вкладка в браузере с адресом http://localhost:8889/tree следующего вида (рис. 3.2):

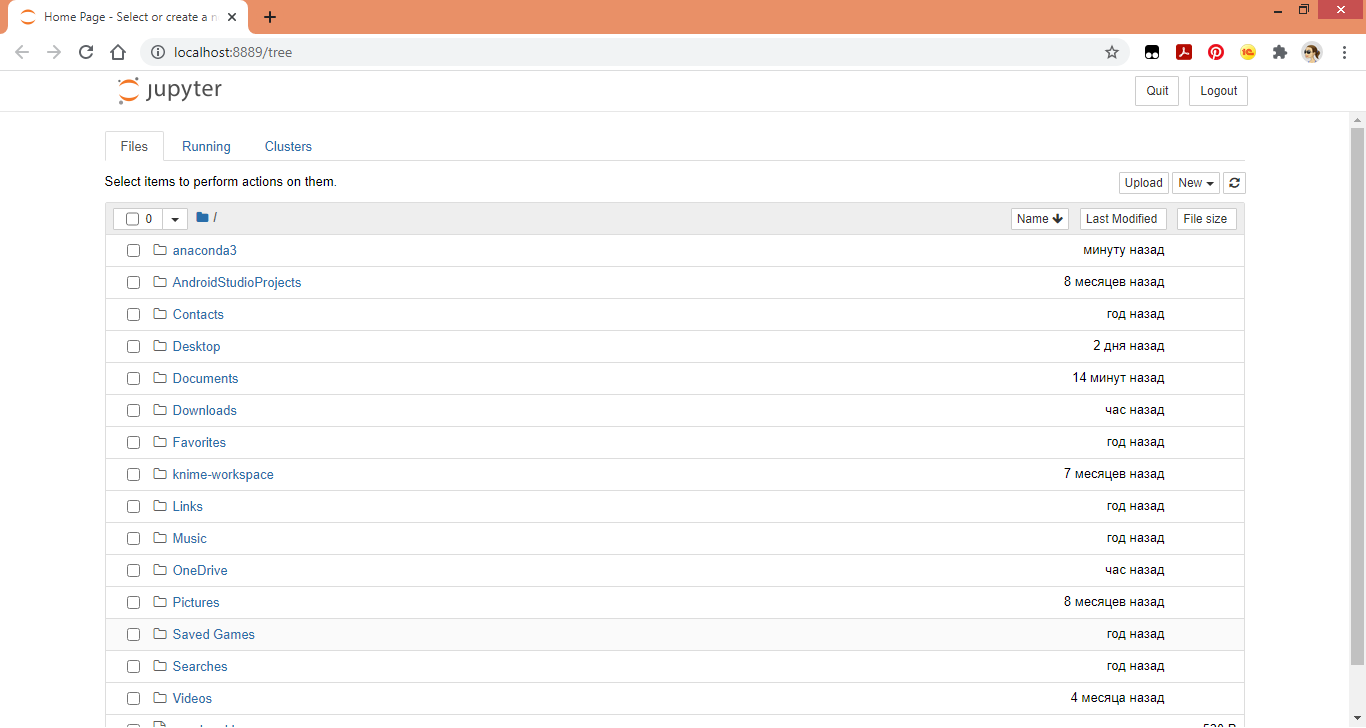


Рисунок 3.2 – Интерфейс Jupyter notebook

1. Запустите Jupyter notebook и откройте ранее созданную папку для лабораторных работ на вкладке Files и в ней папку Lab3.
2. Откройте в папке ноутбук lab3.ipynb. Далее продолжайте работу в этом файле.
3. Сохраните ноутбук с помощью меню “File” → “Save as”, указав в названии файла номер лабораторной работы и вашу фамилию.
4. Результаты работы покажите преподавателю.

**Блокнот lab3.ipynb**

**Теория**

К ближайших соседей (k-nearest neighbors) – алгоритм, который хранит данные и определяет класс для объекта по схожести c другими, класс определяется по классам соседей.

kNN – это метрический алгоритм, предполагающий, что объекты одного класса в пространстве находятся близко друг к другу. В зависимости от выбора k – количества ближайших соседей и метрики расстояния, качество может сильно отличаться.

Алгоритм ленивый, то есть производит вычисления при предсказании, а не при обучении, тренировочную выборку хранит в себе.

Идея алгоритма:

1. Взять новый объект и вычислить все расстояния по некоторой метрике от него до других объектов
2. Выбрать k ближайших соседей к этому объекту
3. Класс объекта – это класс наиболее часто встречающегося объекта среди k соседей.

В алгоритм можно внести изменение добавив веса для каждого объекта или класса. Например, при выборе класса смотрят не на большинство соседей, а на какую-то взвешенную сумму.

Идея ближайшего соседа расширяется и на другие задачи, например, в рекомендательных системах простым начальным решением может быть рекомендация какого-то товара (или услуги), популярного среди ближайших соседей человека, которому хотим сделать рекомендацию.

Качество классификации методом ближайших соседей зависит от нескольких параметров:

1. Число соседей.
2. Метрика расстояния между объектами (часто используются метрика Хэмминга, евклидово расстояние, косинусное расстояние и расстояние Минковского). При использовании большинства метрик значения признаков надо масштабировать. Условно говоря, чтобы признак «Зарплата» с диапазоном значений до 100 тысяч не вносил больший вклад в расстояние, чем «Возраст» со значениями до 100. То есть алгоритм «страдает» от «проклятия размерностей».
3. Веса соседей.

**Задание 1. Прокат ведосипедов**

1. Реализуйте алгоритм KNN, проверьте его работу. Для этого импортируйте библиотеку pandas. Загрузите набор данных с предыдущей лабораторной работы 2 (аренда велосипедов). Можно загружать наборы данных из сети, указав ссылку на файл в формате http://.

Например,

*link = 'https://путь к файлу данных'*

*data = pd.read\_csv(link).*

1. Используйте методику корреляции регрессионного анализа для выбора важных переменных из набора данных, т.е. посмотрите как в наборе данных признаки коррелируют между собой.

Воспользуйтесь методами *loc* и *corr*.

*corr\_matrix = НаборДанных.loc[:,numeric\_col].corr()*,

где numeric\_col – числовые показатели набора данных.

Для лучшей визуализации корреляционной зависимости можно построить тепловую карту. В библиотеке seaborn метод heatmap.

Сильно коррелирующие между собой данные можно исключить из анализа.

Закончите суждение: Судя по корреляционной матрице – это ....

Таким образом, можно бросить любую одну из двух переменных из набора данных методом *drop*

*axis=0, тогда отрбасываем...*

*axis=1, тогда отрбасываем...*

1. Разделите наборы данных на тренировочную выборку (80%) и тестовую (20%). Так как предсказывать будем целевой признак cnt, то поместите его в массив y. Остальные данные без целевого показателя в массив x (известным нам методом *drop*).

Импортируйте train\_test\_split из модуля sklearn.model\_selection.

Создайте четыре выборки X\_train, X\_test, Y\_train, Y\_test с помощью методa *train\_test\_split*.

1. Постройте модель.

Для этого импортируйте KNeighborsRegressor из sklearn.neighbors. Используйте KNeighborsRegressor для работы с числовыми признаками. Для работы с номинальными признаками можно использовать KNeighborsClassifier Укажите количество соседей, например 3. Обучите модель методом fit на тренировочной выборке.

Сделайте предсказание.

1. Проведите оценку модели.

Для этого создайте свою функцию MAPE, где будут в массиве храниться метрики ошибок.

*import numpy as np def MAPE(Y\_actual,Y\_Predicted):*

*mape = np.mean(np.abs((Y\_actual – Y\_Predicted)/Y\_actual))\*100 return mape*

Запомните в массиве все метрики ошибок для тестовых данных Y\_test и полученного предсказания и выведите результат.

Посчитайте качество модели в процентном соотношении как

*100 – количество ошибок.*

Для вывода процентов можно воспользоваться выводом:

*print('Accuracy of KNN model: {:0.2f}%.'.format(Качество)).*

После измените количество соседей в модели, измените проценты тренировочной и тестовой выборок.

**Задание 2. Классификация вина**

* 1. Решите задачу классификации вина на основе готовых моделей.

Для этого импортируйте datasets из sklearn. Определите новый набор данных методом load\_wine()

* 1. Выведите название нецелевых признаков feature\_names и отдельно целевой призак target\_name.
  2. Разделите наборы данных на тренировочную выборку (70%) и тестовую (30%). Набор исходных данных wine.data (выведите его содержимое) Целевой признак wine.target (выведите его содержимое).
  3. Постройте модель.

Для этого мпортируйте KNeighborsClassifier Количество соседей равно пяти. Обучите модель методом fit на тренировочной выборке.

Выполните предсказание.

* 1. Выведите значение accuracy\_score из модуля metrics библиотеки sklearn.
  2. Повторите снова расчеты уже для количества соседей =7. Как изменилось значение Accuracy?
  3. Выберите оптимальное значения для K с помощью метода «Локтя». Метод включает в себя итерацию по различным значениям K и выбор значения с наименьшей частотой ошибок при применении к тестовым данным. Заведите список ошибок для каждого значение К error\_rates = [].

В цикле for i in np.arange(1, 101): перебирая все значения К от 1 до 100 проводите построение модели, ее обучение и предсказание. Добавлять значение np.mean в список можно методом append, при условии, что предсказанное значение не равно тестовому.

* 1. Постройте график методом plot для списка ошибок, предварительно подключив необходимые модули и библиотека.
  2. На основании графика можно выбрать какое количество соседей К дает минимальное число ошибок.
  3. Сделайте вывод.

# Понижение размерности пространства признаков

**Цель**: изучить особенности задачи понижения размерности пространства признаков в машинном обучении и ее реализации на языке программирования Python в Jupyter Notebook..

**Ход работы**

1. Скачайте с портала содержимое папки Lab4. Разместите ее в ранее созданной папке для лабораторных работ.
2. Запустите программу Anaconda Navigator.
3. Выберите в Anaconda Navigator программу Jupyter Notebook и нажмите по кнопке “Launch”.
4. Откроется новая вкладка в браузере с адресом http://localhost:8889/tree.
5. Запустите Jupyter notebook и откройте ранее созданную папку для лабораторных работ на вкладке Files и в ней папку Lab4.
6. Откройте в папке ноутбук lab4.ipynb. Далее продолжайте работу в этом файле.
7. Сохраните ноутбук с помощью меню “File” → “Save as”, указав в названии файла номер лабораторной работы и вашу фамилию.
8. Результаты работы покажите преподавателю.

**Блокнот lab4.ipynb**

**Задание 1. Подготовка данных**

1. Ознакомимся с методами понижения размерности данных из библиотеки Scikit Learn.

Для этого загрузите датасет по ссылке: https://www.kaggle.com/uciml/glass. Данные представлены в виде csv таблицы.

1. Импортируйте две библиотеки pandas и numpy.
2. Прочитайте данные из загруженной таблицы, выведите данные на экран. Разберитесь в данных.
3. Разделите данные на описательные признаки и признак отображающий класс. Выведите их (пример на рис. 4.1)

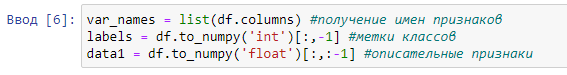


Рисунок 4.1 – Работа с данными

1. Проведите нормировку данных к интервалу [0;1]. Для этого импортируйте метод preprocessing из библиотеки sklearn.
2. Используйте функцию minmax\_scale.
3. Постройте диаграммы рассеяния для пар признаков. Самостоятельно определите соответствие цвета на диаграмме и класса в датасете. Для этого импортируйте matplotlib.pyplot. Пример кода на рис. 4.2

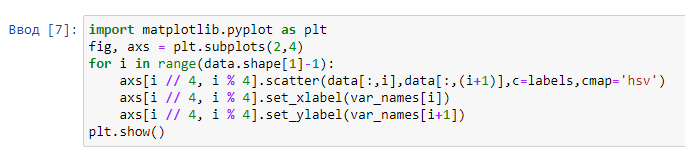


Рисунок 4.2 – Пример фрагменты кода

**Задание 2. Метод главных компонент**

* 1. Используя метод главных компонент (PCA). Проведите понижение размерности пространства до размерности 2. Для этого импортируйте метод PCA из библиотеки sklearn.decomposition Установите размерность =2.
  2. Методами fit и transform сформируйте данные. Выведите из на экран, проведите анализ.
  3. Выведите значение объясненной дисперсии в процентах (explained\_variance\_ratio\_) и собственные числа, соответствующие компонентам (singular\_values\_).
  4. Постройте диаграмму рассеяния после метода главных компонент. Пример кода на рис. 4.3. Проанализируйте результаты.

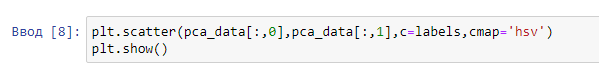


Рисунок 4.3 – Фрагмент кода

* 1. Изменяя количество компонент, определите количество, при котором компоненты объясняют не менее 85% дисперсии данных.

# Сокращение размерности. Факторный анализ

**Цель**: изучить особенностями факториального анализа в машинном обучении и его реализацию на языке программирования Python в Jupyter Notebook.

**Ход работы**

1. Скачайте с портала содержимое папки Lab5. Разместите ее в ранее созданной папке для лабораторных работ.
2. Запустите программу Anaconda Navigator.
3. Выберите в Anaconda Navigator программу Jupyter Notebook и нажмите по кнопке “Launch”.
4. Откроется новая вкладка в браузере с адресом http://localhost:8889/tree.
5. Запустите Jupyter notebook и откройте ранее созданную папку для лабораторных работ на вкладке Files и в ней папку Lab5.
6. Откройте в папке ноутбук lab5.ipynb. Далее продолжайте работу в этом файле.
7. Сохраните ноутбук с помощью меню “File” → “Save as”, указав в названии файла номер лабораторной работы и вашу фамилию.
8. Результаты работы покажите преподавателю.

**Блокнот lab5.ipynb**

CRISP (Cross-Industry Standard Process for Data Mining) – это модель жизненного цикла исследования данных для систем машинного обучения.

Методология CRISP разбивает процесс машинного обучения на 6 этапов.

1. Понимание проблемы (Business Understanding). Этот этап направлен на определение целей и требований к проекту со стороны бизнеса. Составляется план проекта.
2. Понимание данных (Data Understanding). Эта фаза включает в себя поиск источников, сбор данных, формирование гипотез о скрытых закономерностях в данных. Также в ней выявляются ошибки в данных, выбросы, пропуски, оценивается качество данных.
3. Подготовка данных (Data preparation). Здесь из общего массива данных выбираются данные нужные для модели. Производятся преобразования данных (one-hot encoding, например), перевод в нужные форматы, комбинирование с целью получения новых данных.
4. Моделирование (Modeling). В этом этапе используются разнообразные методики моделирования и алгоритмы, производятся тесты модели. Из-за смены моделей возможен частый возврат на предыдущий этап.
5. Оценка модели (Evaluation). Нужно убедиться, что модель покрывает все поставленные бизнесом цели. Вычисляются оценки ее качества, производится ревью процесса.
6. Развертывание (Deployment). Последний этап часто представляет собой простое формирование отчета. Но во многих случаях включает в себя автоматизацию какого-либо процесса анализа данных, внедрения построенной модели в какую-либо информационную систему для решения задач бизнеса.
7. Сокращение размерности – это промежуточный шаг между моделированием данных и их подготовкой.

Три метода:

1. Метод главных компонент PCA.
2. Метод сингулярного разложения SVD
3. TSNE (нет аналога названия на русской языке).

**Пример 1**

Рассмотрим пример из массива случайных числовых данных pca\_example.xls. Пример кода на рис. 5.1.

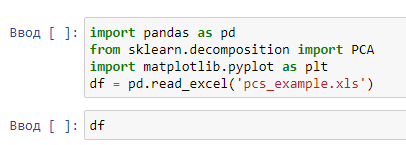


Рисунок 5.1 – Фрагмент кода

В датасете 6 переменных и 1000 строк. Визуализация при этом будет сложна: для каждый пары необходимо будет построить диаграмму рассеивания, посмотреть связаны они или нет. И это только для того, чтобы представить эти данные в двумерном пространстве. Можно упростить, используя метод главных компонент. Код на рис. 5.2.

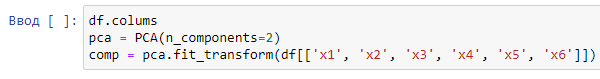


Рисунок 5.2 – Программный код

Получается два вектора (т.к. у нас две компоненты). Их можно сохранить в отдельной переменной и сделать диаграмму рассеивания уже по ней (рис. 5.3).



Рисунок 5.3 – Программный код

Полученный график можно использовать для анализа данных, поиска зависимости и т.д.

Можно посмотреть дисперсию: в первой компоненте находится примерно 99,9% дисперсии, во второй ближе к нулю (рис. 5.4).



Рисунок 5.4 – Дисперсия

**Пример 2**

Реализуем визуализацию данных по сайту hh.ru для вакансии «Преподаватель английского языка» и проведем ее анализ.

*import requests*

*vac = pd.DataFrame(requests.get(‘https://api.hh.ru/vacancies?area=1&*

*text=преподаватель+английского+языка&per\_page=100’).json()[‘items’])*

Можно посмотреть отдельно требования (рис. 5.5).

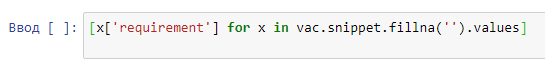


Рисунок 5.5 – Требования

Желательно воспользоваться переводом текста в числовые данные, воспользуетесь для этого библиотекой sklearn (рис. 5.6).

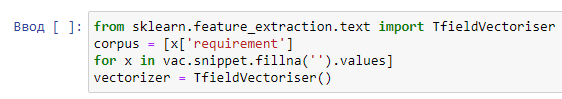


Рисунок 5.6 – Программный код

Взвешенная частота слова в числовом векторе без смысла, убираются служебные символы, остается словарь. Предварительно надо избавиться от пустых значений в тексте и от значений None.

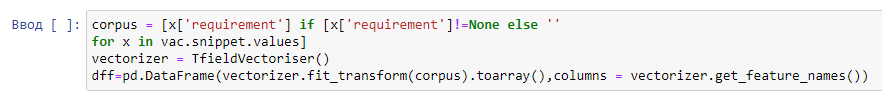


Рисунок 5.7 – Программный код

Оцените размер датасета, количество строк и столбцов.

Применим метод главных компонент PCA (рис. 5.8).

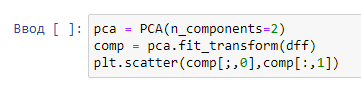


Рисунок 5.8 – Программный код

Оцените полученный график вакансий. Есть ли группы, есть ли выбросы?

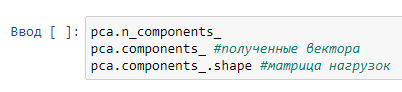


Рисунок 5.8 – Программный код

Применим метод TSNE.

Это достаточно сложный метод. Сводится к тому, что все данные исходные переносят в новое пространство, чтобы сохранились принципы близости или дальности значений. Далее считаются расстояния, вычисляется вероятность близости и ищется пространство, где эти вероятности сохраняются. Как пишут авторы данного методы, «мы пытаемся сжать пружину, изменив положение в пространстве».

Метод часто используется для анализа текстов.

Рассмотрим его работу для нашего корпуса вакансий преподавателей английского языка с сайта hh.ru (рис. 5.9).



Рисунок 5.9 – Программный код

Реализовать метод можно для примера в цикле, где изменяя перплексию (величина для оценки и изменения пространства) можно рассмотреть различные варианты (рис. 5.10).

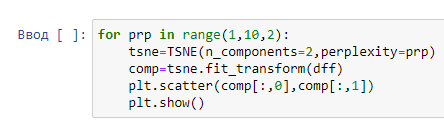


Рисунок 5.10 – Программный код

# Визуализация данных и решающие деревья в Python

**Цель**: изучить особенности визуализации информации для анализа данных в машинном обучении и ее реализацию на языке программирования Python в Jupyter Notebook.

**Ход работы**

1. Скачайте с портала содержимое папки Lab7. Разместите ее в ранее созданной папке для лабораторных работ.
2. Запустите программу Anaconda Navigator.
3. Выберите в Anaconda Navigator программу Jupyter Notebook и нажмите по кнопке “Launch”.
4. Откроется новая вкладка в браузере с адресом http://localhost:8889/tree.
5. Запустите Jupyter notebook и откройте ранее созданную папку для лабораторных работ на вкладке Files и в ней папку Lab7.
6. Откройте в папке ноутбук lab6.ipynb. Далее продолжайте работу в этом файле.
7. Сохраните ноутбук с помощью меню “File” → “Save as”, указав в названии файла номер лабораторной работы и вашу фамилию.
8. Результаты работы покажите преподавателю.

**Блокнот lab7.ipynb**

Задание 1. Рассмотрим вопросы визуализации на примере датасета tips – на нем можно решать задачу предсказания размера чаевых в ресторане

1. Загрузите библиотеки pandas и seaborn.
2. Найдите информацию, для чего используется библиотека seaborn.
3. Загрузите набор данных для предсказания размера чаевых в ресторане tips (в зависимости от размера счета, пола клиента, времени суток (день или вечер), дня недели, размера заказа и был ли сделан заказ в зале для курящих или нет). Зарузить датасет можно методом load\_dataset из библиотеки seaborn.
4. Изучите содержимое датасета.
5. Рассмотрим несколько вариантов построения графиков на примере задач.

Задача 1. Есть ли зависимость размера чаевых (tip) от общей стоимости заказа (total\_bill)? Можно использовать метод relplot из библиотеки seaborn.

1. Задача 2. Определите размер чаевых в зависимости от пола (col='sex').
2. Задача 3. Визуализируйте зависимость размера чаевых от пола человека, оплатившего заказ (не надо включать информацию о полной стоимости совершенного заказа).

Такой способ называется визуализацией разброса измеренных значений внутри каждой категории (в нашем случае категорией будет пол). Для этого используем метод catplot.

1. Задача 4. Визуализируйте распределение размера чаевых в зависимости от дня недели (day).
2. Задача 5. Визуализируйте, как на размер чаевых влияет не только пол человека, оплатившего заказ, но еще и время дня (hue='time').
3. График разброса по категориям удобен для небольших наборов данных, так как по мере увеличения количества точек они все равно начнут перекрывать друг друга и сливаться. Чтобы преодолеть эти трудности, лучше воспользоваться графиками, которые сами содержат некоторую информацию о распределении внутри категорий. Один из таких графиков – это «ящик с усами» или boxplot. Его можно построить с помощью той же функции catplot с параметром kind, установленным в значение 'box'.

Решите задачу 3 через «ящик с усами».

1. Решите задачу 4 через «ящик с усами».
2. Еще один тип графика, который позволяет судить о распределении значений – это violinplot (kind='violin'). Широкая и тонкая черная полоса внутри «виолончели» соответствует «ящику» и «усам». Если данные состоят только из двух подмножеств, то violinplot можно разделить пополам для каждой из них с помощью параметра split (split=True). Белая точка внутри – это медиана.

Для примера реализуйте зависимость чаевых (tip) от времени дня (time) и пола (sex).

1. Столбчатые диаграммы тоже могут быть очень удобны для визуализации (kind='bar'). Функция по умолчанию отображает среднее значение величины внутри каждой категории и отображает ее доверительный интервал.

Решите задачу 4 через столбчатую диаграмму.

1. Для того чтобы отразить количество наблюдений внутри каждой категории можно воспользоваться функцией countlot() или построить столбчатую диаграмму (kind='count').

Задача 6. Визуализируйте количество обслуженных столиков по каждому дню (day)

Изучите функцию pairplot и визуализируйте датасет.

1. Постройте pairplot зависимость данных от пола (hue ='sex'), измените маркеры (например, круг для м и квадрат для ж).
2. Изучите функцию heatmap (тепловая карта) и визуализируйте датасет data.corr() с отражением коэффициента корреляции (annot=True) в выбранной цветовой гамме (cmap, выбрать цвета https://matplotlib.org/2.0.2/examples/color/colormaps\_reference.html).

Задание 2. Перед решением задачи необходимо проверить данные на наличие пропусков и изучить типы переменных.

1. Для этого можно воспользоваться функцией info() для датасета.
2. Кодируем категориальные признаки.

Категориальный признак – это такой признак, который может принимать одно значение из ограниченного числа возможных. В наших данных четыре категориальных признака, три из которых являются бинарными. Нам необходимо закодировать их, то есть преобразовать в числовые. Бинарные признаки кодируются методом факторизации, который преобразует их значения в 0 и 1. Признак, принимающий более двух значений закодируем методом дамми-кодирования.

Рассмотрим на примере. Пусть имеется категориальный признак Category, принимающий одно из четырех возможных значений ['Human', 'Penguin', 'Octopus', 'Alien']. После применения дамми-кодирования мы получим четыре новых признака (по количеству возможных значений) Category\_Human, Category\_Penguin, Category\_Octopus, Category\_Alien. Для той строчки, у которой в исходных данных стояла категория Human, в столбце Category\_Human будет стоять 1, в остальных столбцах 0. Аналогично для другого значения. Фрагмент кода представлен на рис. 7.1, где #df – это имя переменной, хранящей датасет.

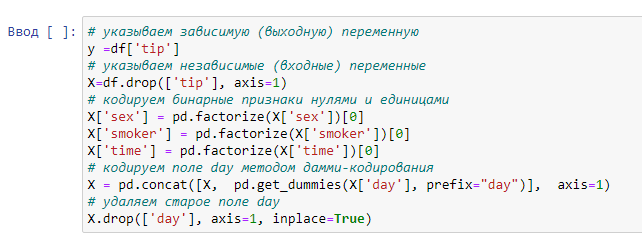


Рисунок 7.1 – Фрагмент кода

1. Посмотрите, как изменились данные после кодирования.
2. Из библиотеки scikit-learn импортируйте метод train\_test\_split и разделите выборку на обучающую и тестовую части в соотношении 3:1.
3. DecisionTreeRegressor – процедура для построения деревьев решений в задачах регрессии.

DecisionTreeClassifier – процедура для построения деревьев решений в задачах классификации.

plot\_tree – процедура для визуализации деревьев.

Программный код представлен на рис .7.2.

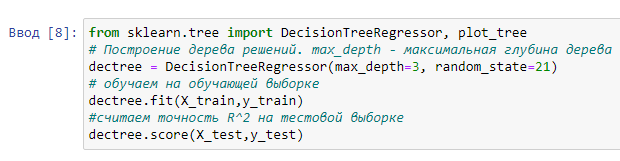


Рисунок 7.2 – Программный код

1. Рисуем дерево. matplotlib.pyplot – базовая библиотека для визуализации, используется здесь для задания общих размеров картинки. Процедура plot\_tree непосредственно рисует дерево (рис. 7.3).

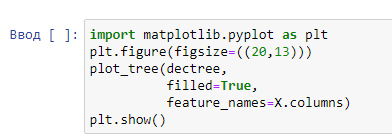


Рисунок 7.3 – Программный код

Левая стрелка – да, правая – нет. Samples – количество примеров из обучающей выборки, попавших в данный узел. Value – предсказанное значение чаевых. МSE – величина среднеквадратической ошибки (среднее значение квадрата разности между предсказанными чаевыми и теми, что были на самом деле). Далее программный код представлен на рис. 7.4.

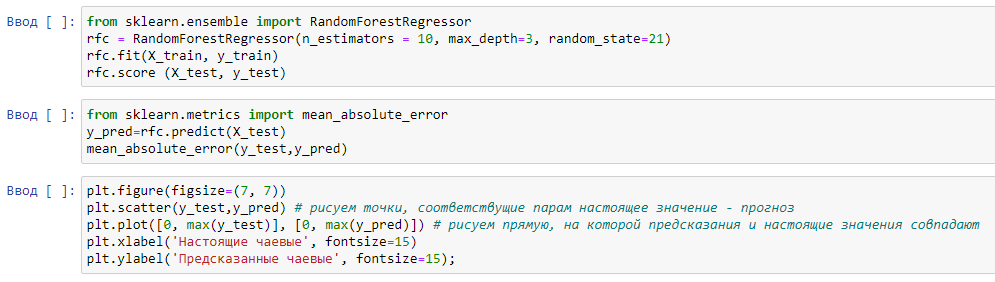


Рисунок 7.4 – Программный код

1. Сделайте вывод о данных на основе полученного дерева.