Южно-Уральский государственный
гуманитарно-педагогический университет

Л. С. Носова

МАШИННОЕ ОБУЧЕНИЕ

Учебно-методическое пособие

Челябинск
2022

УДК 681.14(076)(021)
ББК 32.973.2я73
Н84

Рецензенты:
канд. пед. наук, доцент Н. А. Давыдова;
канд. пед. наук О. Н. Иванова

Носова, Людмила Сергеевна
Н84 Машинное обучение: учебно-методическое пособие / Л. С. Носова ;– [Челябинск] : Изд-во Библиотека А. Миллера, 2022. – 77 с. : ил.
ISBN 978-5-93162-705-2

В данном пособии представлены учебно-методические рекомендации к выполнению лабораторных работ по дисциплине «Машинное обучение». Лабораторные работы описывают основные методы работы задач машинного обучения, реализованных с помощью языка программирования Phyton и аналитической платформы Knime и могут использоваться для освоения на практических занятиях, а также в процессе самостоятельной работы студентов и магистрантов. Учебно-методическое пособие предназначено для студентов и преподавателей высших учебных заведений, а также может быть полезно учителям средних образовательных учреждений для повышения квалификации и формирования ИКТ-компетентности. Учебное пособие соответствует требованиям ФГОС ВО.

УДК 681.14(076)(021)
ББК 32.973.2я73

ISBN 978-5-93162-705-2 © Носова Л. С., 2022

Содержание

Пояснительная записка 5

1 Лабораторные работы 8

1.1 Лабораторная работа № 1
«Введение в анализ данных на Phyton» 8

1.2 Лабораторная работа №2
«Основные понятия задачи регрессии» 26

1.3 Лабораторная работа №3
«К ближайших соседей (KNN)» 35

1.4 Лабораторная работа №4
«Понижение размерности пространства признаков» 41

1.5 Лабораторная работа №5
«Сокращение размерности. Факторный анализ» 44

1.6 Лабораторная работа №6
«Сокращение размерности в аналитической платформе Knime» 49

1.7 Лабораторная работа №7
«Визуализация данных и решающие деревья в Phyton».....54

1.8 Лабораторная работа №8
«Визуализация в аналитической платформе KNIME» 60
.....
1.9 Лабораторная работа №9
«Кластеризация в аналитической платформе Knime» 62
.....
1.10 Лабораторная работа №10
«Классификация в аналитической платформе Knime» 66
.....
1.11 Лабораторная работа №11
«Итоговая проектная работа» 68
.....
2 Задачи для самостоятельного решения 70
.....
3 Контрольные вопросы 73
.....
4 Список литературы.....75
.....

Пояснительная записка

Дисциплина «Машинное обучение» относится к модулю части, формируемой участниками образовательных отношений, Блока 1 «Дисциплины/модули» основной профессиональной образовательной программы по направлению подготовки 09.03.02 «Информационные системы и технологии» (уровень образования бакалавр). Дисциплина является дисциплиной по выбору.

Общая трудоемкость дисциплины составляет 2 зачетных единиц, 72 час.

Изучение дисциплины «Машинное обучение» основано на знаниях, умениях и навыках, полученных при изучении обучающимися следующих дисциплин: «Алгоритмы дискретной математики», «Алгоритмы и структуры данных», «Компьютерная алгебра», «Программирование», «Технологии программирования». Дисциплина «Машинное обучение» формирует знания, умения и компетенции, необходимые для освоения следующих дисциплин: «Большие данные», для проведения следующих практик: «производственная практика (научно-исследовательская работа)», «производственная практика (преддипломная)», «производственная практика (технологическая (проектно-технологическая)) практика».

Цель изучения дисциплины: освоение обучающимися современных технологий для обработки и анализа данных и эффективных методов машинного обучения с применением современных программных средств, а также формирование целостной системы знаний в области создания, накопления, обработки и использования данных для решения практических задач.

Задачи дисциплины:

1) иметь представление о теоретических основах машинного обучения, принципах построения алгоритмов;

2) осуществлять математическую и информационную постановку задач по обработке информации;

3) владеть методами интеллектуального анализа информации.

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы представлен в таблице (Таблица 1).

Таблица 1 — Планируемые результаты

ПК-1 способность проводить исследования на всех этапах жизненного цикла программных средств	
ПК.1.1 Знать современные методики проведения исследований на всех этапах жизненного цикла программных средств	З. Знать понятие больших данных и их свойства; постановку задачи классификации и регрессии; принципы работы алгоритмов машинного обучения
ПК.1.2 Уметь проводить исследования на всех этапах жизненного цикла программных средств	У. Уметь выполнять постановку задачи машинного обучения; уметь использовать алгоритмы машинного обучения для анализа данных
ПК.1.3 Иметь навыки владения современным программным обеспечением для проведения исследований на всех этапах жизненного цикла программных средств	В. Владеть навыками предобработки данных, на основе языка программирования Phyton и аналитической платформы Knime

В данном пособии собраны лабораторные работы, раскрывающие возможности использования методов машинного обучения для обработки данных на языке программирования Phyton в среде Anaconda и аналитической платформе Knime. Данный материал может использоваться для освоения на практических занятиях, а также в процессе самостоятельной работы студентов инженерных специальностей.

1 Лабораторные работы

1.1 Лабораторная работа №1. Введение в анализ данных на Phyton

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

Ход работы

Часть 1. 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.

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

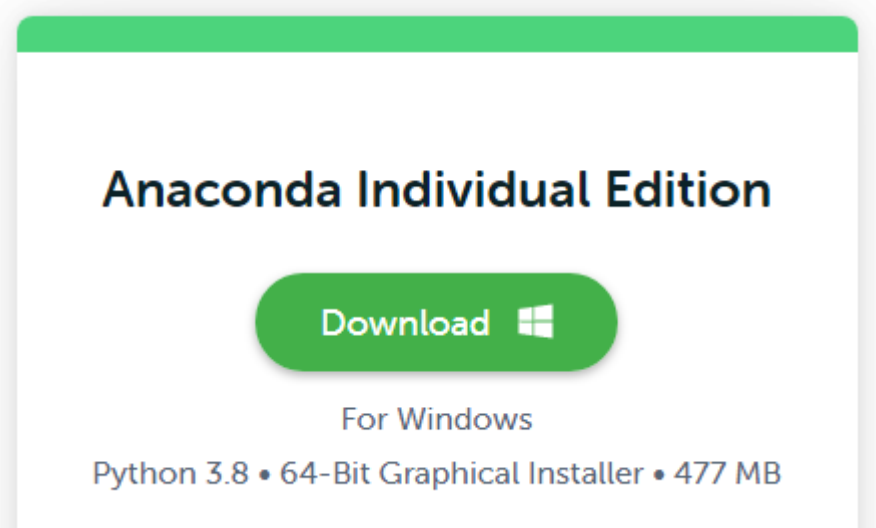

Рисунок 1.1 — Установка Anaconda

3. 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 была использована кириллица**, только английские символы. Дождитесь окончания установки.

4. Запустите программу Anaconda Navigator – 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:

1. JupyterLab.
2. Jupyter Notebook.
3. QtConsole.
4. Spyder.
5. Glueviz.

6. Orange.

7. RStudio.

8. Visual Studio Code.

9. PyCharm CE.

Мы будем работать с Jupyter Notebook (рисунок 1.2).

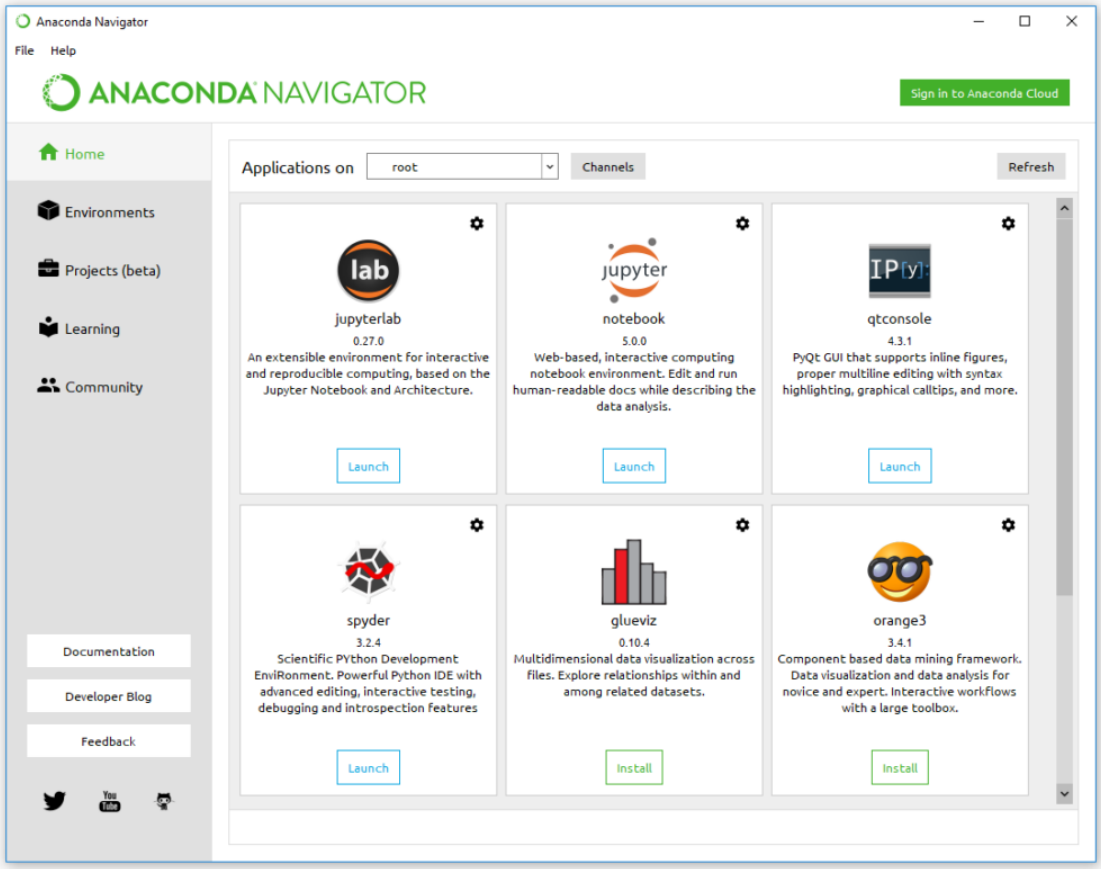

Рисунок 1.2 — Интерфейс Anaconda Navigator

5. 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 ноут-буки – 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 процесса работы, но при этом позволяет заново выполнить код, присутствующий на нем. Ноутбуки можно экспортировать в форматы PDF, HTML.

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

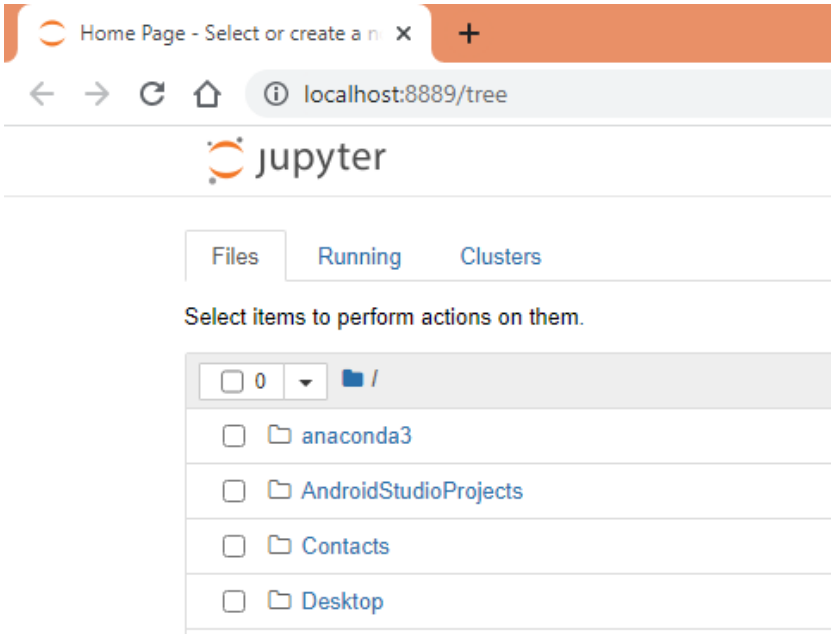

Рисунок 1.3 – Интерфейс Jupyter notebook

7. 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).

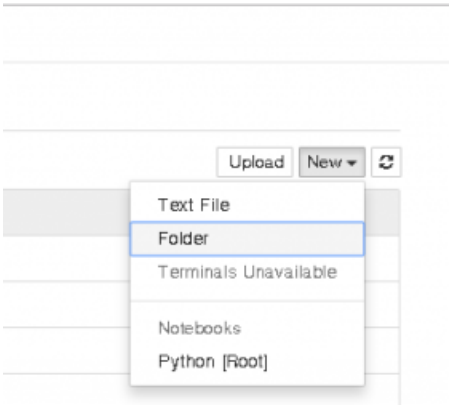

Рисунок 1.4 – Меню “New”

8. 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”

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

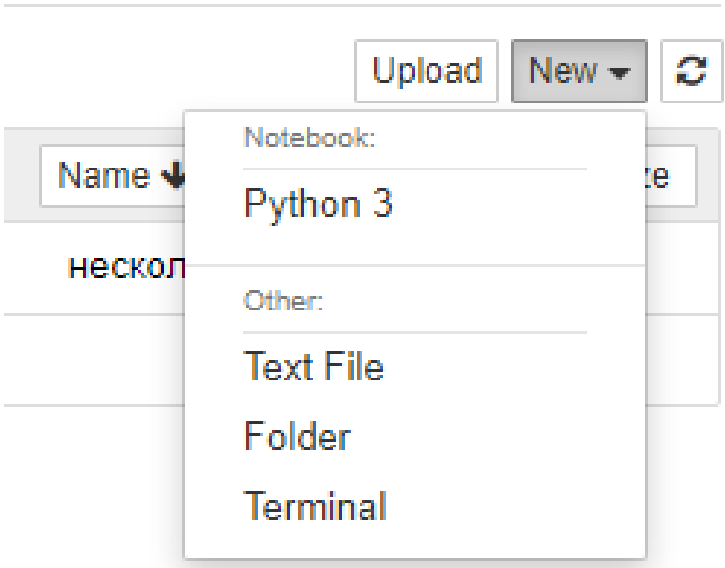

Рисунок 1.6 – Меню “New”

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

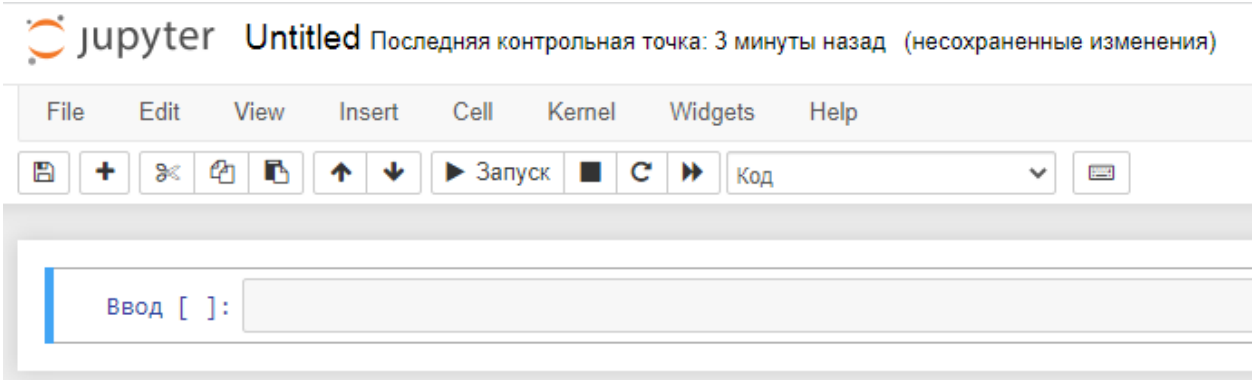

Рисунок 1.7 – Ноутбук

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

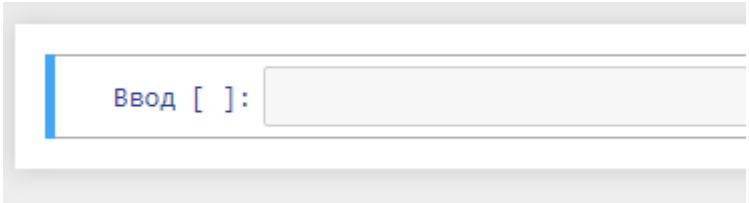

Рисунок 1.8 – Ячейки для ввода

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

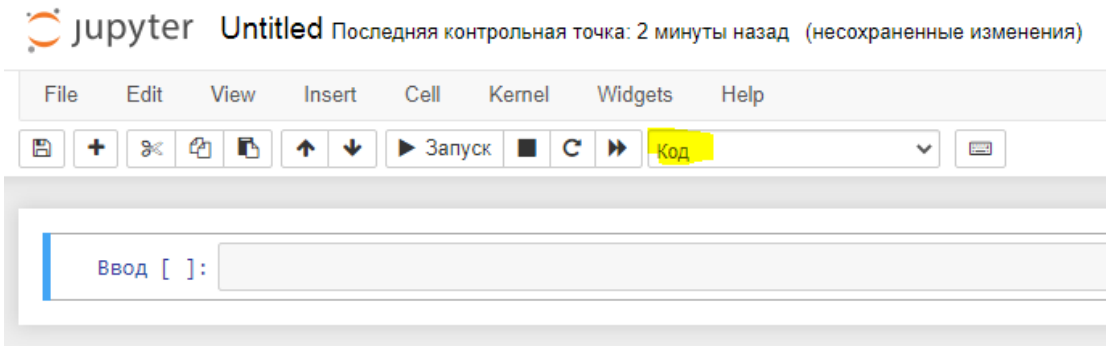

Рисунок 1.9 – Свойство Код

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

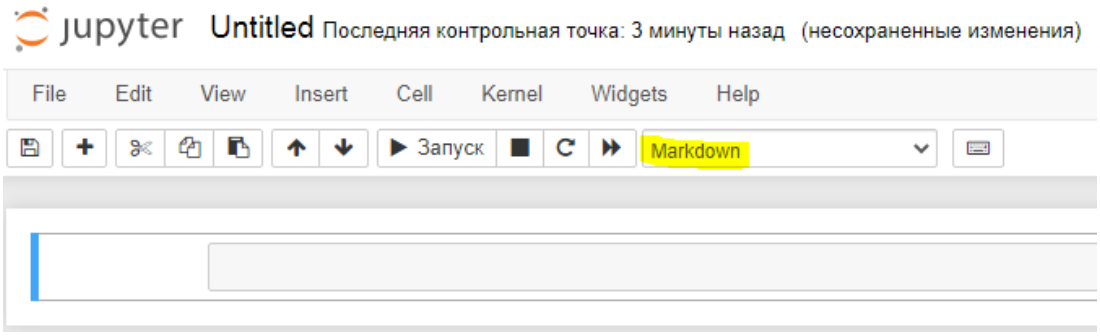

Рисунок 1.10 – Свойство Markdown

14. 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.

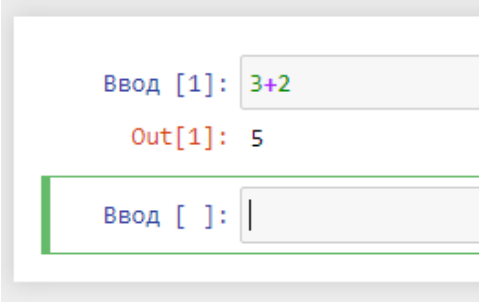

Рисунок 1.11 – Пример

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

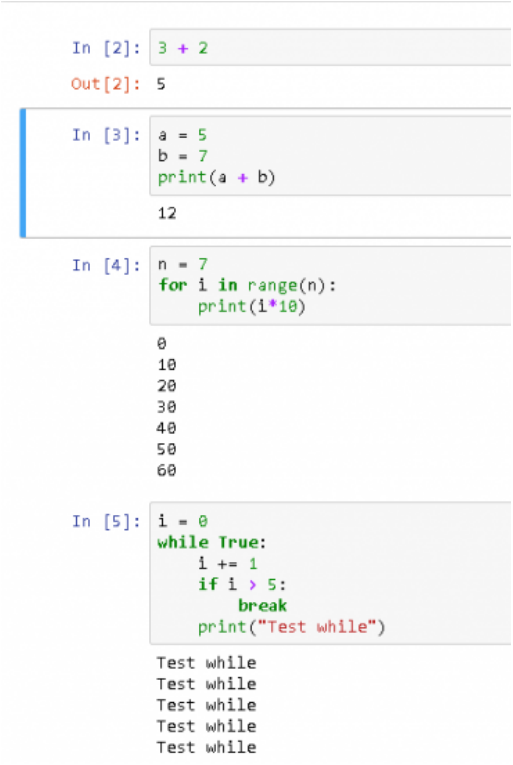

Рисунок 1.12 – Примеры для самостоятельной работы

Часть 2. 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).

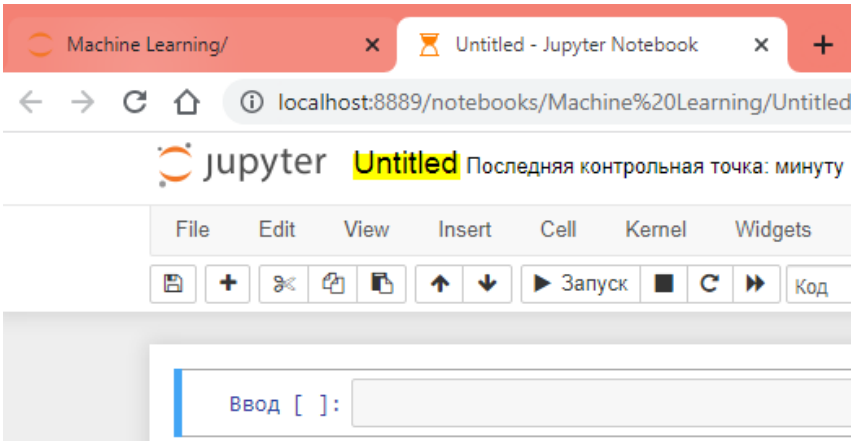

Рисунок 1.13 – Имя ноутбука

Из элементов интерфейса можно выделить

– панель меню (рис. 1.14):

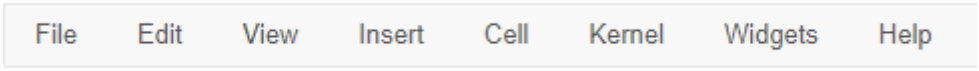

Рисунок 1.14 – Панель главного меню

– панель инструментов (рис. 1.15):

Рисунок 1.15 – Панель инструментов

– рабочее поле с ячейками (рис. 1.16):

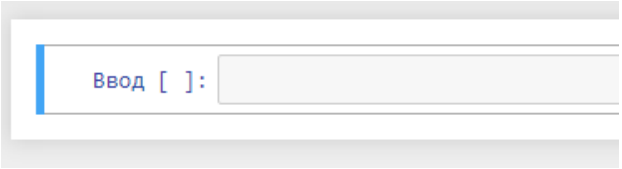

Рисунок 1.16 – Рабочее поле

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).

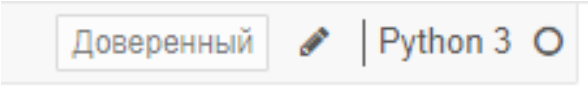

Рисунок 1.17 – Режим работы

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

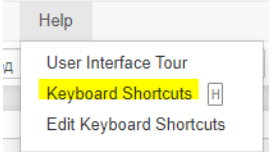

Рисунок 1.18 – Меню “Help”

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 изменится на закрашенный круг.

Часть 3. 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/.

Часть 4. Вывод изображений в ноутбуке

Печать изображений может пригодиться в том случае, если используется библиотека 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.

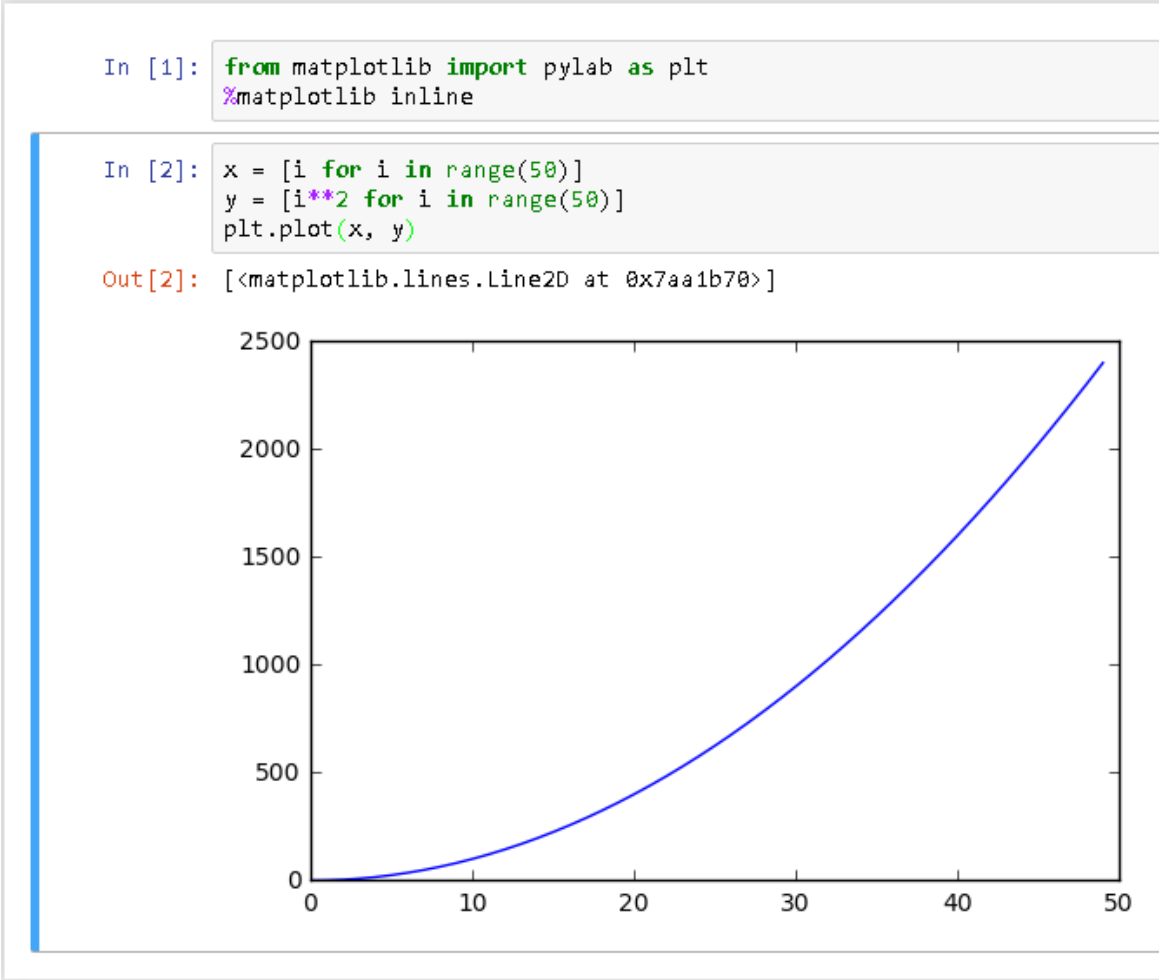

Рисунок 1.20 – Графики в Jupyter Notebook

Часть 5. 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

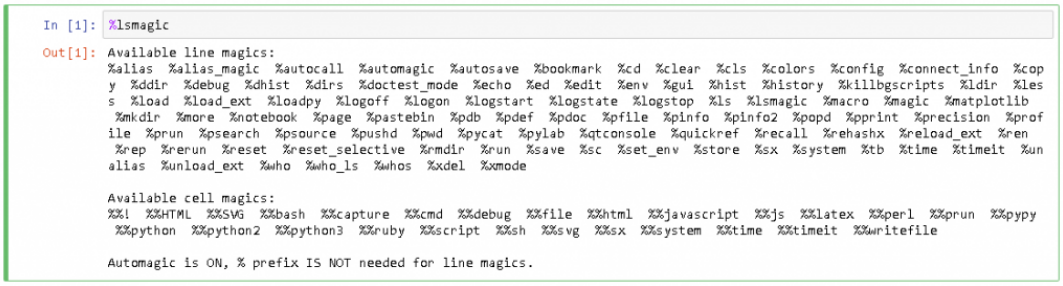

Рисунок 1.21 – Команды «магии»

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.

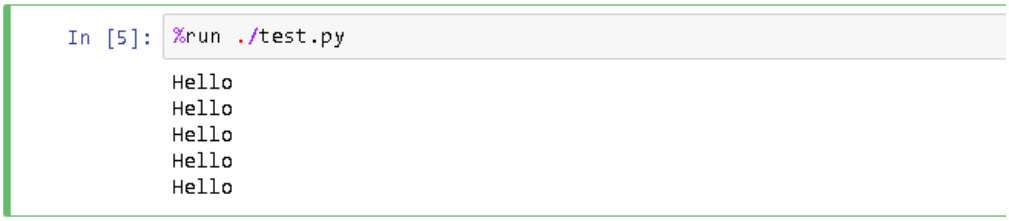

Рисунок 1.23 – Использование команды %run

Для измерения времени работы кода используется:

%%time и *%timeit*.

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

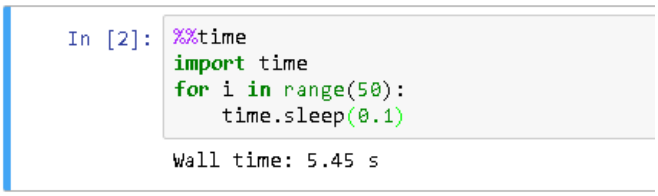

Рисунок 1.24 – Использование команды %%time

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

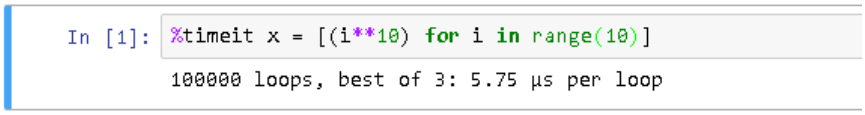

Рисунок 1.25 – Результаты работы команды %timeit

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.

16. В самом конце панели откройте список и выберите пункт “Markdown”:

17. 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.

18. На рисунке 1.26 представлены способы оформления текстовых блоков ноутбука (рис. 1.26):

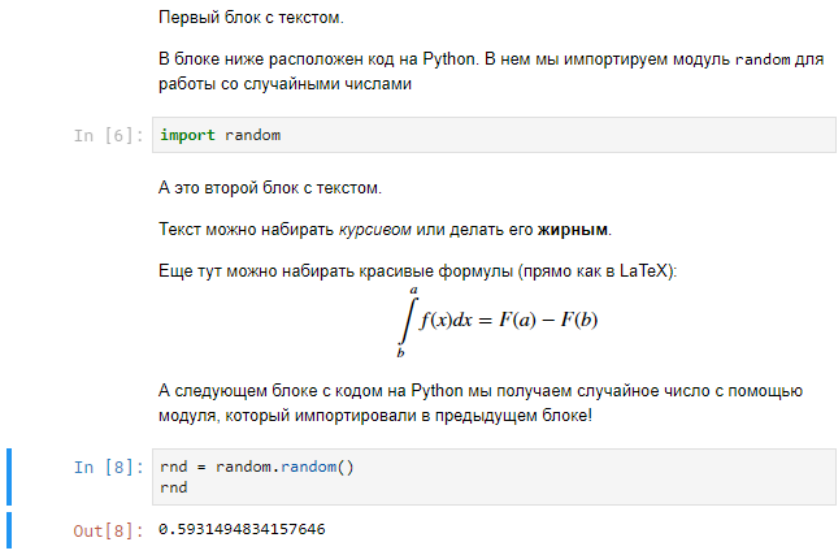

Рисунок 1.26 – Примеры оформления текстовых блоков

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

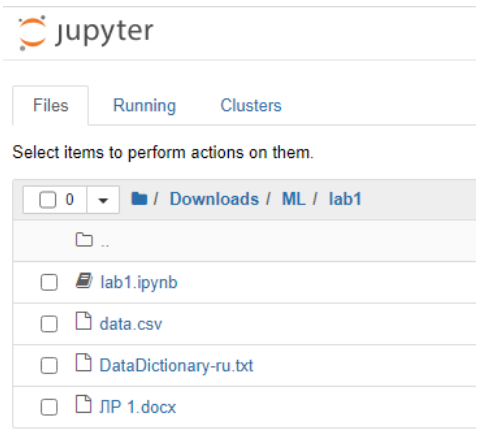

Рисунок 1.27 – Содержимое папки Lab1

20. Запустите блокнот lab1.ipynb.

21. Выполните все задания в блокноте.

22. Сохраните ноутбук с помощью меню “File” – “Save as” (рис. 1.30), указав в названии файла номер лабораторной работы и вашу фамилию.

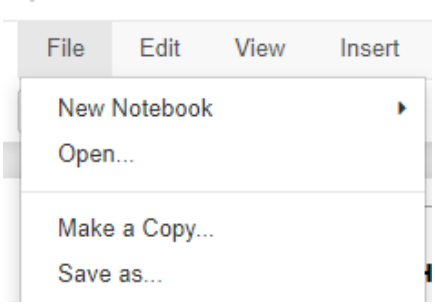

Рисунок 1.30 – Меню “File”

23. 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.

```
Ввод [1]: import pandas as pd
```

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=',')

```
Ввод [ ]: # место для кода
```

Рисунок 1.28 – Блоки для ввода

2. 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. 1.29). Какому типу принадлежит каждый столбец (вещественный, целый, категориальный)?

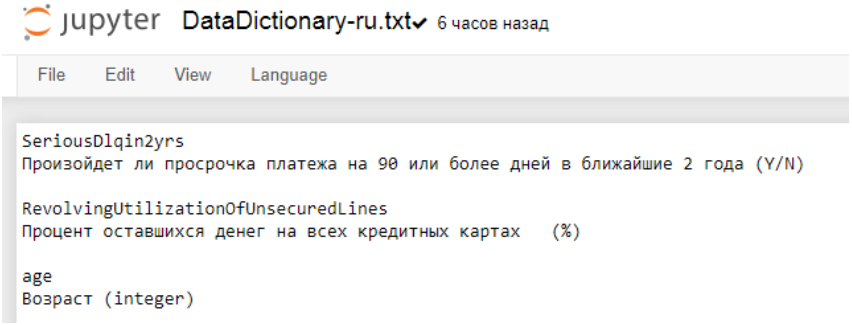

Рисунок 1.29 – Фрагмент содержимого файла

5. 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).

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

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

7. 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.

8. 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.

9. Для визуализации данных можно использовать следующие функции (рис. 1.30).

```
Ввод [13]: import matplotlib.pyplot as plt

           #  функция, позволяющая выводить графики прямо в ноутбук
           %matplotlib inline
```

Рисунок 1.30 – 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).

1.2 Лабораторная работа №2. Основные понятия задачи регрессии

Цель: ознакомиться с особенностями решения задачи регрессии в машинном обучении и ее реализации на языке программирования Phyton 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).

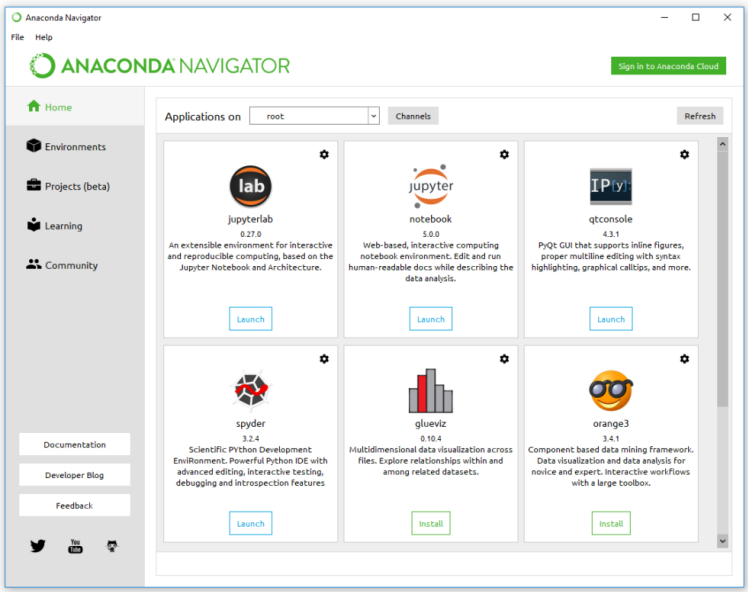

Рисунок 2.1 – Интерфейс Anaconda Navigator

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 следующего вида (рис. 2.2):

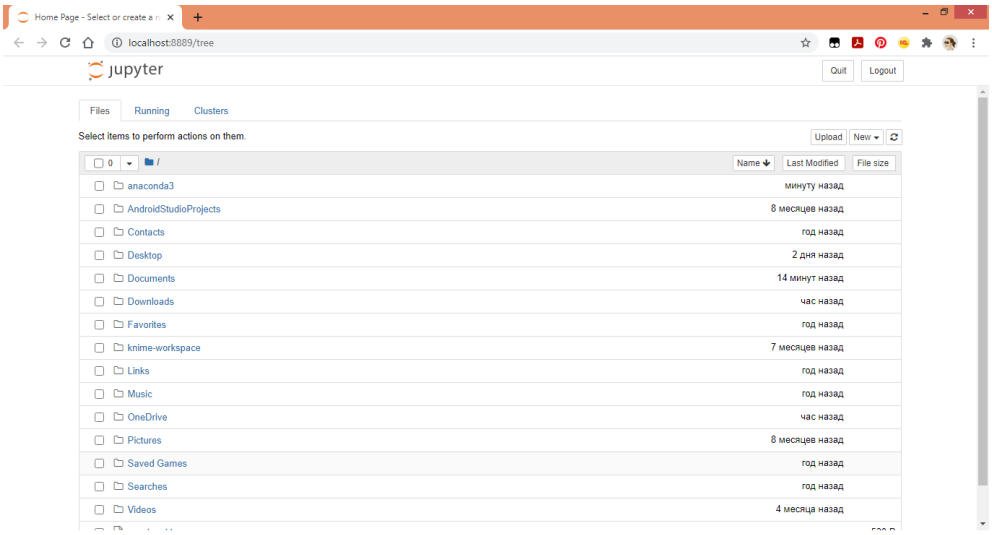

Рисунок 2.2 – Интерфейс Jupyter notebook

5. Запустите Jupyter notebook и откройте ранее созданную папку для лабораторных работ на вкладке Files и в ней папку Lab2.

6. Откройте в папке ноутбук lab2.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 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

2. 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.

```
Ввод [36]: import seaborn as sns

           sns.kdeplot(data['temp'])
           # добавьте к графику поля atemp, windspeed(ms), windspeed(mph), hum

Out[36]: <AxesSubplot:xlabel='temp', ylabel='Density'>
```

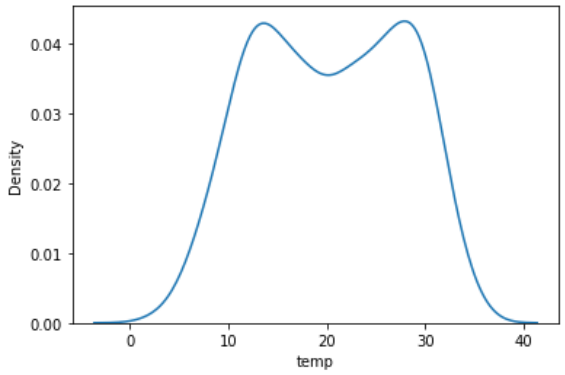

Рисунок 2.4 – Результат работы функции

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.

```
Ввод [5]: from matplotlib import pyplot as plt
          plt.figure(figsize=(12,8))
          for i in data.drop(['cnt'], axis=1).columns:
              sns.kdeplot(data[i])
```

Рисунок 2.5 – Фрагмент кода

Новый график представлен на рис. 2.6.

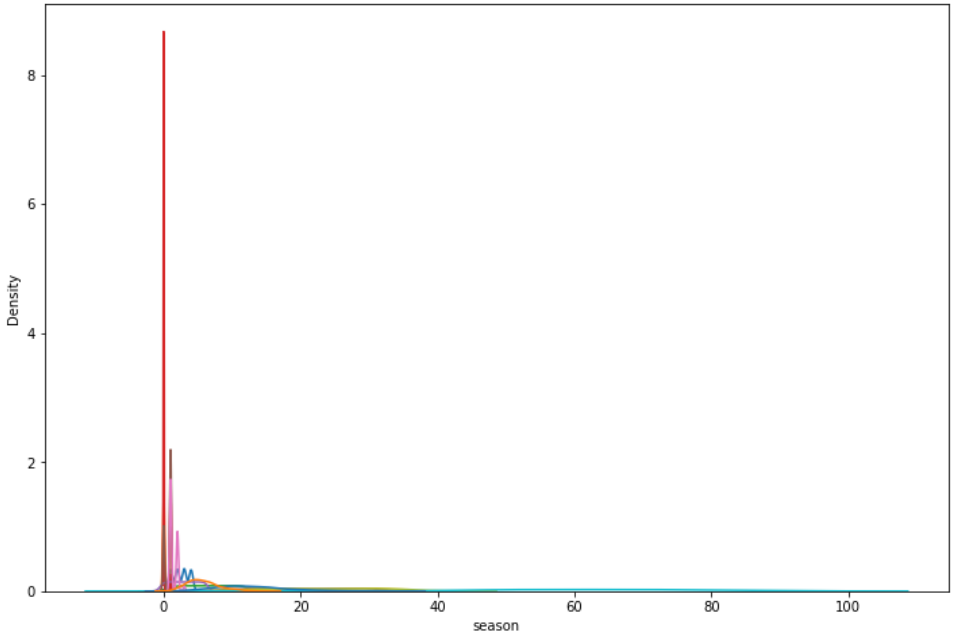

Рисунок 2.6 – График функции

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.

```
Ввод [6]: import numpy as np

          def StandardScaler(X):
              mean = X - np.mean(X)
              return mean / np.std(X)
```

```
Ввод [7]: # разобьем выборку на целевые и нецелевые признаки
          x = data.drop(['cnt'], axis=1)
          y = data['cnt']
```

```
Ввод [8]: x_scaled = StandardScaler(x)
          x_scaled
```

Рисунок 2.7 – Код

Фрагмент масштабированного датафрейма представлен на рис. 2.8.

Out[8]:

	season	yr	mnth	holiday	weekday
0	-1.348213	-1.001369	-1.600161	-0.171981	1.498809
1	-1.348213	-1.001369	-1.600161	-0.171981	-1.496077
2	-1.348213	-1.001369	-1.600161	-0.171981	-0.996930
3	-1.348213	-1.001369	-1.600161	-0.171981	-0.497782
4	-1.348213	-1.001369	-1.600161	-0.171981	0.001366
...	...	...	...	...	...
726	-1.348213	0.998633	1.588660	-0.171981	0.500513
727	-1.348213	0.998633	1.588660	-0.171981	0.999661
728	-1.348213	0.998633	1.588660	-0.171981	1.498809
729	-1.348213	0.998633	1.588660	-0.171981	-1.496077
730	-1.348213	0.998633	1.588660	-0.171981	-0.996930

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.

```
Ввод [9]: plt.figure(figsize=(12,8))
          for i in x_scaled.columns:
              sns.kdeplot(x_scaled[i])
```

Рисунок 2.10 – Программный код

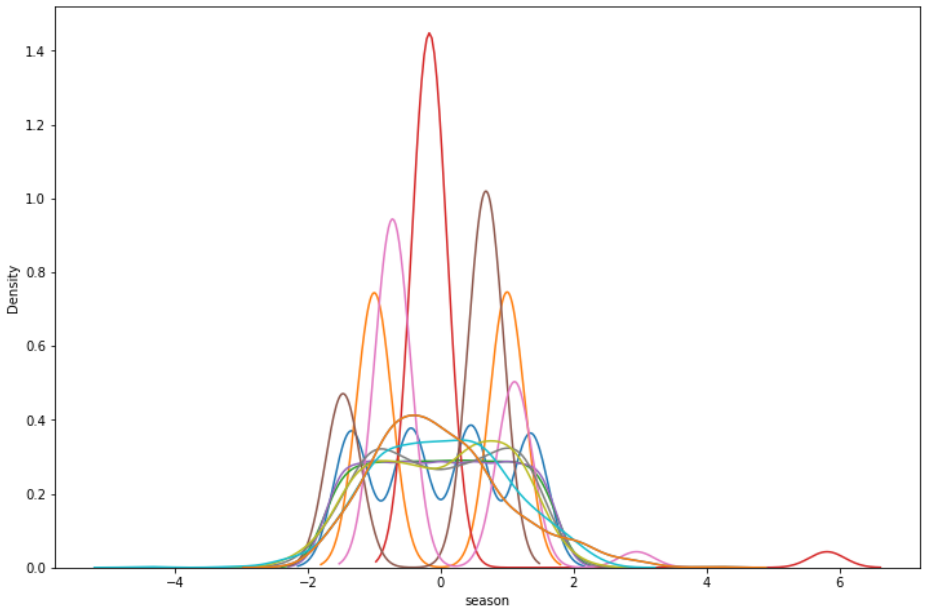

Рисунок 2.11 – График масштабированных данных

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.

```
Ввод [10]: print(np.mean(x_scaled.values))
           print(np.std(x_scaled.values))
```

Рисунок 2.12 – Программный код

Среднее должно быть околонулевым значением, а отклонение равно единице.

Регрессия

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

```
Ввод [11]: # переименуем переменные для удобства,
           # и превратим данные в numpy-массивы.
           x = x_scaled.values
           y = y.values
```

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)*.

3.1. 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.

3.2. 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])].

3.3. 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.

1.3 Лабораторная работа №3. К ближайших соседей (kNN)

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

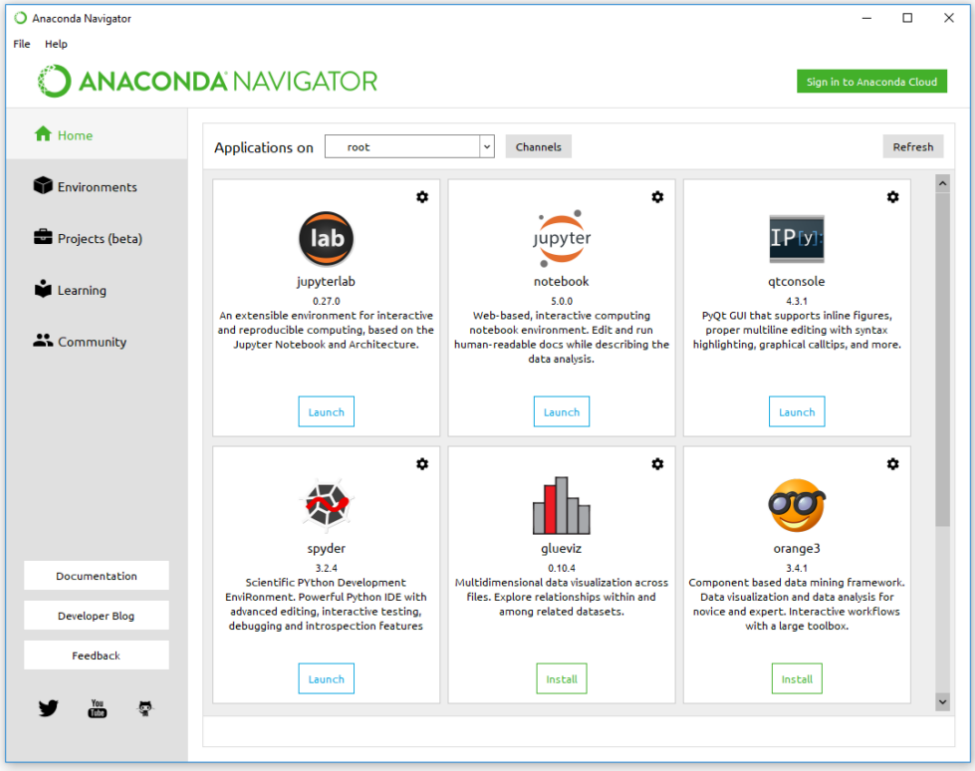

Рисунок 3.1 – Интерфейс Anaconda Navigator

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 следующего вида (рис. 3.2):

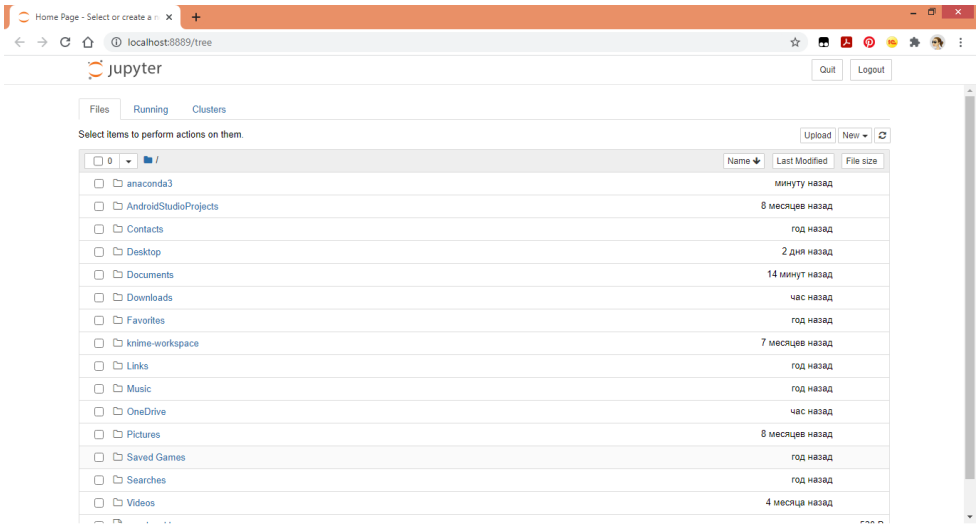

Рисунок 3.2 – Интерфейс Jupyter notebook

5. Запустите Jupyter notebook и откройте ранее созданную папку для лабораторных работ на вкладке Files и в ней папку Lab3.

6. Откройте в папке ноутбук lab3.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 lab3.ipynb

Теория

К ближайших соседей (k-nearest neighbors) – алгоритм, который хранит данные и определяет класс для объекта по схожести с 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 велосипедов

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).

2. 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 отбрасываем...

axis=1, тогда отбрасываем...

3. 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 метода *train_test_split*.

4. 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.

5. 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().

2. Выведите название нецелевых признаков feature_names и отдельно целевой признак target_name.

3. 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).

4. Постройте модель.

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

Выполните предсказание.

5. Выведите значение accuracy_score из модуля metrics библиотеки sklearn.

6. Повторите снова расчеты уже для количества соседей =7. Как изменилось значение Accuracy?

7. 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 значения 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.

8. Постройте график методом plot для списка ошибок, предварительно подключив необходимые модули и библиотеки.

На основании графика можно выбрать какое количество соседей К дает минимальное число ошибок.

9. Сделайте вывод.

1.4 Лабораторная работа №4. Понижение размерности пространства признаков

Цель: изучить особенности задачи понижения размерности пространства признаков в машинном обучении и ее реализации на языке программирования Phyton 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.

2. Импортируйте две библиотеки pandas и numpy.

3. Прочитайте данные из загруженной таблицы, выведите данные на экран. Разберитесь в данных.

4. Разделите данные на описательные признаки и признак отображающий класс. Выведите их (пример на рис. 4.1)

```
Ввод [6]: var_names = list(df.columns) #получение имен признаков
          labels = df.to_numpy('int')[:,-1] #метки классов
          data1 = df.to_numpy('float')[:,:-1] #описательные признаки
```

Рисунок 4.1 – Работа с данными

5. Проведите нормировку данных к интервалу [0;1]. Для этого импортируйте метод preprocessing из библиотеки sklearn.

6. Используйте функцию minmax_scale.

7. 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

```
Ввод [7]: import matplotlib.pyplot as plt
          fig, axs = plt.subplots(2,4)
          for i in range(data.shape[1]-1):
              axs[i // 4, i % 4].scatter(data[:,i],data[:,(i+1)],c=labels,cmap='hsv')
              axs[i // 4, i % 4].set_xlabel(var_names[i])
              axs[i // 4, i % 4].set_ylabel(var_names[i+1])
          plt.show()
```

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.

```
Ввод [8]: plt.scatter(pca_data[:,0],pca_data[:,1],c=labels,cmap='hsv')
          plt.show()
```

Рисунок 4.3 – Фрагмент кода

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

1.5 Лабораторная работа №5. Сокращение размерности. Факторный анализ

Цель: изучить особенностями факториального анализа в машинном обучении и его реализацию на языке программирования Phyton 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 русском языке).

Пример 1

Рассмотрим пример из массива случайных числовых данных pca_example.xls. Пример кода на рис. 5.1.

```
Ввод [ ]: import pandas as pd
          from sklearn.decomposition import PCA
          import matplotlib.pyplot as plt
          df = pd.read_excel('pcs_example.xls')

Ввод [ ]: df
```

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.

```
Ввод [ ]: df.colums
          pca = PCA(n_components=2)
          comp = pca.fit_transform(df[['x1', 'x2', 'x3', 'x4', 'x5', 'x6']])
```

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).

```
Ввод [ ]: plt.scatter(comp[;,0],comp[:,1])
```

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).

```
Ввод [ ]: pca.explained_variance_ratio_
```

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).

```
Ввод [ ]: [x['requirement'] for x in vac.snippet.fillna('').values]
```

Рисунок 5.5 – Требования

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

```
Ввод [ ]: from sklearn.feature_extraction.text import TfieldVectoriser
          corpus = [x['requirement']
          for x in vac.snippet.fillna('').values]
          vectorizer = TfieldVectoriser()
```

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.

```
Ввод [ ]: corpus = [x['requirement'] if [x['requirement']!=None else ''
          for x in vac.snippet.values]
          vectorizer = TfieldVectoriser()
          dff=pd.DataFrame(vectorizer.fit_transform(corpus).toarray(),columns = vectorizer.get_feature_names())
```

Рисунок 5.7 – Программный код

Оцените размер датасета, количество строк и столбцов.

Применим метод главных компонент PCA (рис. 5.8).

```
Ввод [ ]: pca = PCA(n_components=2)
          comp = pca.fit_transform(dff)
          plt.scatter(comp[;,0],comp[:,1])
```

Рисунок 5.8 – Программный код

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

```
Ввод [ ]: pca.n_components_
          pca.components_ #полученные вектора
          pca.components_.shape #матрица нагрузок
```

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).

```
Ввод [ ]: from sklearm.manifold import TSNE
```

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).

```
Ввод [ ]: for prp in range(1,10,2):
              tsne=TSNE(n_components=2,perplexity=prp)
              comp=tsne.fit_transform(dff)
              plt.scatter(comp[:,0],comp[:,1])
              plt.show()
```

Рисунок 5.10 – Программный код

1.6 Лабораторная работа №6. Сокращение размерности в аналитической платформе Knime

Цель: изучить особенностями задачи сокращения размерности в машинном обучении и его реализацию с помощью аналитической платформы Knime.

Ход работы

Knime Analytics Platform

KNIME (/ naɪm /), Konstanz Information Miner – это бесплатная платформа для анализа данных, отчетности и интеграции с открытым исходным кодом. KNIME объединяет различные компоненты для машинного обучения и интеллектуального анализа данных.

Аналитическая платформа KNIME предоставляет пользователям возможности визуально создавать потоки данных (конвейеры), выборочно выполнять отдельные или все шаги анализа, а затем проверять результаты, модели, используя интерактивные виджеты и представления.

Скачать программу можно с сайта: последняя версия https://www.knime.com/knime-analytics-platform или версия 4.3.2 https://disk.yandex.ru/d/5iOlyzb9GaNnzg.

Задание 1. Работа с числовыми данными.

1. Запустите программу. При запуске запрашивается рабочая папка. Сохраняйте каждый проект в отдельной папке.

2. Создайте новый проект.

3. Добавьте в проект данные из файла excel, для этого разместите компонент Excel Reader и настройте его. В контекстном меню выберите пункт **Configure**…, выберите файл с числовым набором данных в области **File** кнопка **Browse**. Если загрузка произошла корректно, то в области **Preview** появятся данные.

4. Можно изменять диапазоны данных. Например, уберите столбец индекса. Для этого установите параметры как указано на рисунке 6.1.

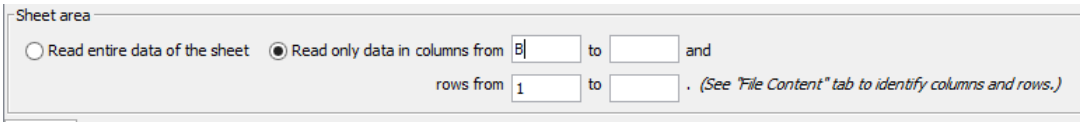

Рисунок 6.1 – Настройка параметров

5. Запустите узел на исполнение: в контекстном меню выберите пункт **Execute**. Зеленый кружок внизу узла говорит об отсутствии ошибок в выполнении.

6. Изучите загруженные данные: в контекстном меню выберите пункт **File – Table**. Посмотрите на вкладках спецификацию и свойства данных.

7. Добавьте компонент PCA (Analytics, Mining) или в поиске можно указать PCA. Выберите **PCA Compute**.

8. Установите связь между компонентами: удерживая левую кнопку мыши протяните от черного треугольника справа от первого компонента до черного треугольника слева от второго компонента.

9. Настройте компонент PCA. В контекстном меню выберите пункт **Configure**…, выберите какие параметры необходимо преобразовать. В нашем случае это все шесть. Запустите узел на исполнение: в контекстном меню выберите пункт **Execute**.

10. После исполнения оцените полученные модели матрицы ковариации (в контекстном меню выберите пункт **Covariance Matrix**), собственные вектора (**Spectral Decomposition**) и саму модель (**Transformation Model**).

11. Далее можно изучить модель, для этого добавьте компонент **PCA Apply** и соедините его через квадраты голубого цвета с предыдущим узлом.

12. Подайте исходные данные на компонент **PCA Apply**, соединив его с первым компонентом посредством черных треугольников.

13. Запустите узел на исполнение: в контекстном меню выберите пункт **Execute**.

14. Изучите полученные данные в контекстном меню пункт **Transformed data**, получились преобразованные результаты в столбце **PCA dimension 0**.

15. Далее можно производить настройки в пункте **Configure**…, перезапускать обработку и изучать полученные данные (рис. 6.2).

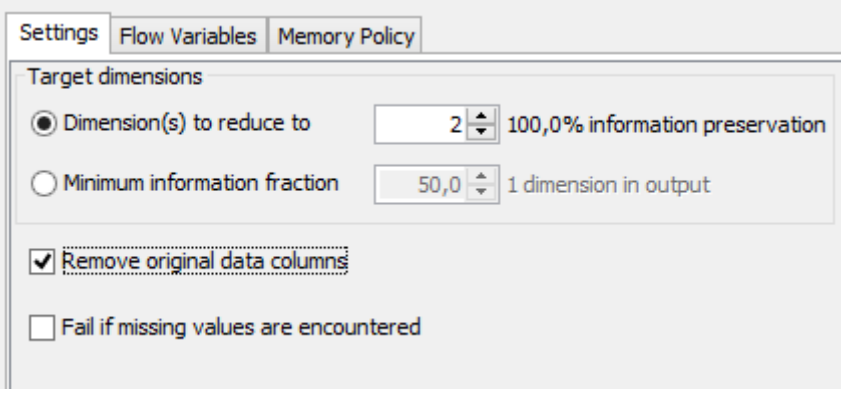

Рисунок 6.2. – Настройка параметров

16. Визуализируйте данные с помощью графика. Выберите два компонент **Scatter Matrix**. Соедините первый с исходными данными, второй – с полученной моделью.

17. Для первого компонента **Scatter Matrix** запустите выполнение. Изучите полученный график зависимости параметров в контекстном меню **View: Scatter Matrix**. Можно добавить компоненты и посмотреть корреляцию компонент.

18. Исходя из выводов, настройте конфигурацию компоненты **PCA Apply**, запустите на выполнение и изучите результат визуализации в **Scatter Matrix**. Сделайте вывод.

Задание 2. Работа с тестовыми данными.

1. Добавьте в проект следующие компоненты: **GET Request**, **JSON Path**, **Ungroup**.

2. Установите связи между компонентами.

3. Настройте компонент **GET Request**, указав путь до сайта и запрос: *https://api.hh.ru/vacancies?area=1&text=преподаватель+английского+языка&per_page=100*.

Запустите на исполнение, посмотрите результат в пункте **GET results**.

4. Настройте компонент **JSON Path**, в разделе **Configure**… в области **JSON-Cell Preview** выберите параметры, которые необходимы (например, id, name, requiment), с помощью **Add JSON Path**. В области **Outputs** замените значение 0 на *, чтобы отобразить все вакансии и установите тип **List** для всех.

5. Компонент **Ungroup** запустите на выполнение и изучите **Data table**. Таким образом получены данные для дальнейшей векторизации.

6. Для обработки текста необходимо импортировать расширение **Text processing** в меню **File**. Далее добавьте компонент **Strings To Document**. Свяжите его с предыдущим компонентом. В конфигурации установите настройки: **Title** – **requiement**, **Tokerization** – **OpenNLP Simple Tokenizer**. Запустите на выполнение и изучите полученные документы из строк.

7. Далее обработайте их компонентом **Document Vector**. Свяжите его с предыдущим компонентом, в конфигурации уберите настройку **As collection cell**.

8. На связь между последними компонентами добавьте узел **Bag Of Words Creator**. В конфигурации оставьте только колонку **Documents**. Запустите компоненты на исполнение. Изучите полученные данные.

9. Добавьте компонент **PCA**. Настройте конфигурацию, выбрав две компоненты, отбросив исходную информацию. Запустите на исполнение, проанализируйте полученные данные.

10. Постройте диаграмму рассеивания с помощью **Scatter Plot**. В конфигурации в качестве измерений укажите компоненту 0 и 1. Сделайте выводы.

Задание 3. Самостоятельная работа

Вернитесь к заданию 1. Действительно ли стоит прибегать к сокращению размерности данных? Как это проверить?

11. Сохраните файл в формате Knime, указав в названии файла номер лабораторной работы и вашу фамилию.

12. Результаты работы покажите преподавателю.

1.7 Лабораторная работа №7. Визуализация данных и решающие деревья в Phyton

Цель: изучить особенности визуализации информации для анализа данных в машинном обучении и ее реализацию на языке программирования Phyton 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). Загрузить 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.

Задача 2. Определите размер чаевых в зависимости от пола (col='sex').

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.

Задача 4. Визуализируйте распределение размера чаевых в зависимости от дня недели (day).

Задача 5. Визуализируйте, как на размер чаевых влияет не только пол человека, оплатившего заказ, но еще и время дня (hue='time').

6. 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».

8. Решите задачу 4 через «ящик с усами».

9. 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).

10. 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.

11. 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.

12. Постройте pairplot зависимость данных от пола (hue ='sex'), измените маркеры (например, круг для м и квадрат для ж).

13. 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.

```
Ввод [ ]: # указываем зависимую (выходную) переменную
y =df['tip']
# указываем независимые (входные) переменные
X=df.drop(['tip'], axis=1)
# кодируем бинарные признаки нулями и единицами
X['sex'] = pd.factorize(X['sex'])[0]
X['smoker'] = pd.factorize(X['smoker'])[0]
X['time'] = pd.factorize(X['time'])[0]
# кодируем поле day методом дамми-кодирования
X = pd.concat([X,  pd.get_dummies(X['day'], prefix="day")],  axis=1)
# удаляем старое поле day
X.drop(['day'], axis=1, inplace=True)
```

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.

5. 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.

Ввод [8]:
```
from sklearn.tree import DecisionTreeRegressor, plot_tree
# Построение дерева решений. max_depth - максимальная глубина дерева
dectree = DecisionTreeRegressor(max_depth=3, random_state=21)
# обучаем на обучающей выборке
dectree.fit(X_train,y_train)
#считаем точность R^2 на тестовой выборке
dectree.score(X_test,y_test)
```

Рисунок 7.2 – Программный код

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

Ввод [ ]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=((20,13)))
plot_tree(dectree,
          filled=True,
          feature_names=X.columns)
plt.show()
```

Рисунок 7.3 – Программный код

Левая стрелка – да, правая – нет. Samples – количество примеров из обучающей выборки, попавших в данный узел. Value – предсказанное значение чаевых. MSE – 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.

Ввод [ ]:
```
from sklearn.ensemble import RandomForestRegressor
rfc = RandomForestRegressor(n_estimators = 10, max_depth=3, random_state=21)
rfc.fit(X_train, y_train)
rfc.score (X_test, y_test)
```

Ввод [ ]:
```
from sklearn.metrics import mean_absolute_error
y_pred=rfc.predict(X_test)
mean_absolute_error(y_test,y_pred)
```

Ввод [ ]:
```
plt.figure(figsize=(7, 7))
plt.scatter(y_test,y_pred) # рисуем точки, соответствущие парам настоящее значение - прогноз
plt.plot([0, max(y_test)], [0, max(y_pred)]) # рисуем прямую, на которой предсказания и настоящие значения совпадают
plt.xlabel('Настоящие чаевые', fontsize=15)
plt.ylabel('Предсказанные чаевые', fontsize=15);
```

Рисунок 7.4 – Программный код

7. Сделайте вывод о данных на основе полученного дерева.

1.8 Лабораторная работа №8. Визуализация в аналитической платформе KNIME

Цель: изучить особенности визуализации информации для анализа данных в машинном обучении и ее реализацию средствами аналитической платформы Knime.

Ход работы

1. Создайте в Excel таблицу с данными: Пол (Мужской/Женский), Рост (в см), Размер обуви, Длина волос (Короткие, Средние, Длинные), Хобби (Кино, Литература, Музыка, Спорт, Фотография, Танцы, Рисование).

2. Заполните таблицу данными сокурсников на основе проведенного опроса.

3. Сохраните таблицу в старом формате xls.

4. Запустите программу Knime. При запуске запрашивается рабочая папка. Сохраняйте каждый проект в отдельной папке.

5. Создайте новый проект.

6. Добавьте в проект данные из файла excel, для этого разместите компонент Excel Reader и настройте его. В контекстном меню выберите пункт **Configure**…, выберите файл с созданным набором данных в области **File** кнопка **Browse**. Если загрузка произошла корректно, то в области **Preview** появятся данные. Если определение данных произошло некорректно, можно в области **Adjust Settings** выбрать заголовок данных **Select the sheet to read** и в области **Column Names** указать номер ряда (row number). Затем в области **Preview** обновить данные **Refresh**.

7. Запустите узел на исполнение: в контекстном меню выберите пункт **Execute**. Зеленый кружок внизу узла говорит об отсутствии ошибок в выполнении.

8. Изучите загруженные данные: в контекстном меню выберите пункт **Output Table**. Посмотрите на вкладках спецификацию и свойства данных.

9. Добавьте компонент **Histogram**, свяжите его с данными. Настройте конфигурацию: **Histogram Column** – рост, **Aggregation Method** – **Occurrence Count**. На вкладке **Binning** укажите количество интервалов **Number of bins 10**. Запустите узел на исполнение и изучите **Interactive View**. При наведении на столбцы отражается частота появления значения.

10. Постройте еще несколько видов диаграмм. Для этого разместите компоненты: **Conditional Box Plot**, **Heatmap**, **Parallel Coordinates Plot**, **Scatter Plot**. Соедините все виды графиков с исходными данными.

11. В настройках диаграммы рассеивания **Scatter Plot** можно посмотреть зависимость роста от размера обуви. Запустите на исполнение и изучите. **Interactive View.**

12. В настройках диаграммы **Conditional Box Plot** установите зависимость пола (**Category Column**) от роста (**Selected Column**). Запустите на исполнение и изучите **Interactive View**.

13. В настройках диаграммы **Parallel Coordinates Plot** выберите рост и размер обуви. Запустите на исполнение и изучите **Interactive View**.

14. В настройках диаграммы **Heatmap** можно изменить цветовую гамму (вторая вкладка). Запустите на исполнение и изучите **Interactive View**.

15. В настройках диаграммы **Heatmap** измените зависимость от пола (**Choose a label column**). Запустите на исполнение и изучите **Interactive View**.

16. Добавьте еще один вид визуализации **Pie/Donut Chart**. В настройках укажите категорию «Хобби». Запустите на исполнение и изучите **Interactive View**.

17. Добавьте еще один вид визуализации **Bar Chart**. В настойках укажите **Category Column** – длина волос, можно указать **Aggregation Method** – **Average**. Запустите на исполнение и изучите **Interactive View**.

18. Сделайте выводы о качестве полученных данных на основе визуализации.

19. Сохраните файл в формате Knime, указав в названии файла номер лабораторной работы и вашу фамилию.

20. Результаты работы покажите преподавателю.

1.9 Лабораторная работа №9. Кластеризация в аналитической платформе Knime

Цель: изучить особенности задачи кластеризации данных в машинном обучении и ее реализацию средствами аналитической платформы Knime.

Ход работы

1. Запустите программу. При запуске запрашивается рабочая папка. Сохраняйте каждый проект в отдельной папке.

2. Создайте новый проект.

3. Добавьте в проект данных из файла excel, для этого разместите компонент **Excel Reader** и настройте его. В контекстном меню выберите пункт **Configure**…, выберите файл с числовым набором данных в области **File** кнопка **Browse**. Укажите в области **Sheet area**, что чтение необходимо начать со столбца B (индексный столбец A для анализа не нужен). Если загрузка произошла корректно, то в области **Preview** появятся данные.

4. Запустите узел на исполнение: в контекстном меню выберите пункт **Execute**. Зеленый кружок внизу узла говорит об отсутствии ошибок в выполнении.

5. Изучите загруженные данные: в контекстном меню выберите пункт **File – Table**. Посмотрите на вкладках спецификацию и свойства данных.

6. Для кластеризации данных добавьте узел **k-Means**. Установите связь между компонентами: удерживая левую кнопку мыши протяните от черного треугольника справа от первого компонента до черного треугольника слева от второго компонента.

7. Для визуализации данных добавьте узел **Silhouette Coefficient**. Установите связь между компонентами: удерживая левую кнопку мыши протяните от черного треугольника справа от первого компонента до черного треугольника слева от второго компонента.

8. Выполним конфигурацию k-Means: установите количество кластеров равным 4.

9. Запустите узлы на исполнение: в контекстном меню выберите пункт **Execute**.

10. Оцените модель кластеризации: размеченный вход **Labeled input** и кластерный профиль **Clusters**.

11. Оцените коэффициенты силуэтов **Mean Silhouette Coefficient**. Стремление среднего коэффициента к 1 говорит о качестве кластера.

12. Для визуализации **Silhouette Coefficient** добавьте узел **Bar Char**. Установите связь между узлами. В конфигурации убедитесь, что построение пойдет по колонке **Clusters**.

13. Однако для точного построения необходимо отфильтровать входные данные. Для этого на связь между **Silhouette Coefficient** и **Bar Char** переместите узел **Column Filter**. В конфигурации оставьте с правой части только две колонки: кластеры и коэффициенты.

14. Добавьте узел **Line Plot** и установите связь с **Column Filter**. Запустите на исполнение и изучить полученный график. В правом верхнем углу графика можно настроить параметры.

15. Предварительно отсортируйте данные для корректного построения графика. Для этого добавьте узел на связь между **Line Plot** и **Column Filter**. В конфигурации сортировки укажите сначала сортировку по кластеру по возрастанию, затем добавьте сортировку коэффициентов по убыванию.

16. Выполните сортировку, изучите полученные данные.

17. Запустите на исполнение график, оцените результат кластеризации данных.

18. Попробуйте провести анализ модели: измените количество кластеров, измените настройку центрирования (use static random seed 10000), измените число итераций.

19. Проведите кластеризацию еще одним методом DBSCAN. Для этого добавьте узел **DBSCAN**.

20. Для корректной работы метода предварительно укажите расстояния – необходимо добавить узел и установите связь с набором данных. В конфигурации выберите **Numeric Distances** метод для подсчета расстояния.

21. Свяжите компонент **Numeric Distances** и **DBSCAN**, а также **DBSCAN** с набором данных.

22. В конфигурации **DBSCAN** можно установить минимум точек =1, так как мы знаем о выбросе среди данных. Изучите полученные результаты.

23. В конфигурации **DBSCAN** можно увеличить расстояние между точками (**Epsilon**, например, 5).

24. Помочь в этом случае может использование метода главных компонент (PCA). Добавьте компонент **PCA**. Установите связь набора данных с компонентом, затем свяжите его с компонентами **Numeric Distances**, **DBSCAN** и **k-Means**.

25. В конфигурации укажите количество компонент 2. Запустите на исполнение и изучите результаты **DBSCAN** и **Line Plot**. Изменились ли результаты?

26. Проведите кластеризацию результатов опроса из предыдущей лабораторной работы.

27. Проведите кластеризацию из набора данных с сайта hh.ru. Необходимо ли заменять предварительно категориальные признаки?

28. Сохраните файл в формате Knime, указав в названии файла номер лабораторной работы и вашу фамилию.

29. Результаты работы покажите преподавателю.

1.10 Лабораторная работа №10. Классификация в аналитической платформе Knime

Цель: изучить особенности задачи классификации данных в машинном обучении и ее реализацию средствами аналитической платформы Knime.

Ход работы

1. Запустите программу. При запуске запрашивается рабочая папка. Сохраняйте каждый проект в отдельной папке.

2. Создайте новый проект.

3. Добавим в проект данных из файла Excel, для этого разместите компонент **Excel Reader** и настройте его. В контекстном меню выберите пункт **Configure**…, выберите файл с числовым набором данных в области **File** кнопка **Browse**. Укажите в области **Sheet area**, что чтение необходимо начать со столбца B (индексный столбец A для анализа не нужен). Если загрузка произошла корректно, то в области Preview появятся данные.

4. Запустите узел на исполнение: в контекстном меню выберите пункт **Execute**. Зеленый кружок внизу узла говорит об отсутствии ошибок в выполнении.

5. Изучите загруженные данные: в контекстном меню выберите пункт **File Table**. Посмотрите на вкладках спецификацию и свойства данных.

6. Добавьте компонент **Number to String**, установите связь с данными и запустите узел на исполнение.

7. Добавьте компонент **Row Splitter** для разделения выборки. Установите связь с предыдущим компонентом и в конфигурации **Include rows by number** укажите первые 200.

8. Для классификации с помощью деревьев расположим следующие компоненты узла **Decision Tree**: **Learner**, **Predictor**, **to Image** и **to Ruleset** (правило «если – то»).

9. Установите связь **Row Splitter** с **Decision Tree Learner** и **Decision Tree Predictor**. Затем связь **Decision Tree Learner** и **Decision Tree Predictor**.

10. Установите связь **Decision Tree Learner** с компонентами **Decision Tree to Image** и **to Ruleset**.

11. Настройте конфигурацию компонента **Decision Tree Learner**. Здесь можно выбрать метрику (например, Gini index), разбиение дерева (например, методом MDL), минимум элементов в узле дерева, количество элементов для обучения, принудительное деление корня дерева (например, по первой переменной) и т.д.

12. Запустите компоненты на исполнение и изучите визуализацию дерева: **Decision Tree to Image**, пункт **View**.

13. Оцените точность классификации с помощью компонента **Scorer**. Установите связь компонентов **Decision Tree Predictor** и **Scorer**. Проверьте конфигурации правильность указания переменных и в пункте **View** оцените значение матрицы запутанности (она перевернутая, это необходимо учесть при оценке). Также можно произвести оценку точности **Accuracy Statistics**.

14. Запустите на исполнение компонент **Decision Tree to Ruleset**. И изучите правила в пункте **Rules table**.

Справка: для решения проблемы разбиения выборки используется метод кросс валидации (компонент **Cross Validation**).

15. Добавьте это компонент и свяжите его с исходными данными (компонентом **Number to String**). Для изменения

настроек компонента необходимо нажать по нему двойным щелчком мыши. Здесь можно увидеть принцип работы компонента. Установите в конфигурации X-Partitioner случайное воспроизведение Random Seed = 1000.

16. Запустите компонент на исполнение и оцените результаты предсказаний в пункте **Connected to Prediction table**. Изучите ошибки в пункте **Error rates**.

17. Оцените качество модели, добавив и связав компонент **Scorer**.

18 Проведите классификацию результатов опроса из лабораторной работы 8.

1.11 Лабораторная работа №11. Итоговая проектная работа

Цель: обобщить и систематизировать полученные знания в области машинного обучения и навыки работы в аналитической платформе Knime и языке программирования Phyton в Jupyter Notebook.

Ход работы

Итоговая проектная работа выполняется в парах по 2 человека.

Задание: разработать модель на основе выбранного набора данных (датасета).

Наборы данных можно найти на следующих сайтах:

– https://archive.ics.uci.edu/ml/datasets.php;

– https://www.kaggle.com/datasets.

Реализовать модель на языке программирования **Phyton** и в **Knime**.

Требования к проекту:

1. Представить описание выбранного датасета и поставленной задачи (своими словами).

2. Для выбранного набора данных провести подготовку данных, обосновать необходимость подготовки (например, наличие пустых знаний или значений типа NaN, ошибочных данных, выбросов и т.д.).

3. Представить предварительный анализ данных методами визуализации, для каждой визуализации представить пояснение/интерпретацию, возможные гипотезы.

4. Для поставленной задачи выбрать подходящую модель обучения, обосновать выбор модели.

5. Для обучения модели разделить набор данных на тестовую и тренировочную выборки.

6. Представить анализ полученной модели, выводы (какие гипотезы подтвердились).

2 Задачи для самостоятельного решения

Задача 1. В таблице 2 приведены сведения о выставленных на продажу квартирах. Каким является признак «Площадь»?

Таблица 2 — Продажа квартир

Квартира	Площадь	Район	Количество комнат	Этаж
Объект А	53	Централь-ный	2	3
Объект В	110	Ленинский	4	5
Объект С	48	Советский	1	4

Задача 2. Даны значения признака А для шести объектов выборки:

$$A=(-1,0,4,1,2,2).$$

Вычислить:

– среднее значение;

– медиану;

– моду;

– отклонение.

Задача 3. Отклонение значений признака P=(1,1,1,1,1,1) равно…

Задача 4. Укажите правильный ответ.

В симметричной выборке…

а) среднее значение близко к нулю;

б) отклонение близко к нулю;

в) разница между медианой и средним значением близка к нулю.

Задача 5. Найти значение пропущенного признака P3 на основе таблицы 3.

Таблица 3 — Признаки объектов

Объект	P1	P2	P3
A	1	1	0
B	2	0	1
C	0	1	5
D	4	1	-

Задача 6. Дан вектор признака P=(1,0,5,2,2). Нормализуем этот вектор по формуле, использующей минимальное и максимальное значение признака P. Значение первой координаты нормализованного вектора буде равно…

Задача 7. Дан вектор признака P=(1,0,5,2,2). Нормализуем этот вектор по формуле, использующей среднее значение и отклонение признака P. Значение последней координаты нормализованного вектора буде равно…

Задача 8. Таблица 4 содержит информацию об оценках, выставленных фильмам. Посчитайте, какую оценку поставит Саша фильму «Гарри Поттер» по метрике Манхэттен. Таблицу нормализовать не нужно. Значение округлите до одного знака после запятой.

Таблица 4 — Оценки фильмам

Пользователь/ Фильм	Супермен	Титаник	Матрица	Гарри Поттер
Вася	5	5	5	5
Петя	5	3	4	4
Маша	2	5	3	5
Саша	3	4	4	-

Задача 9. Первая и третья квартиль значений признака P равны 2, 4 соответственно. Какие из следующих значений будут считаться выбросами?

3	6,5	0,5	-1,5	0	7,5	8

Задача 10. Среднее значение, отклонение и медиана десяти значений признака P равны 10, 1.1 (одна целая одна десятая), и 9 соответственно. Какие из следующих значений будут выбросами (симметричность выборки)?

13	6	7	13,5	6,5	14

Задача 11. Представьте, что у вас есть выборка значений некоторого признака P, и среднее значение в ней равно 50. Для этой выборки был применен критерий Шовене поиска выбросов. Оказалось, что элементы выборки равные 42 и 57 были признаны выбросами. Какие из указанных ниже чисел критерий Шовене гарантированно определит как выбросы?

41	58	44	56

3 Контрольные вопросы

1. Понятия и терминология машинного обучения.

2. Классификация задач машинного обучения.

3. Обучение с учителем.

4. Метод коррекции ошибки.

5. Метод обратного распространения ошибки.

6. Обучение без учителя. Альфа-система подкрепления. Гамма-система подкрепления.

7. Задача классификации. Методы построения деревьев решения. Методика «разделяй и властвуй».

8. Задача классификации. Методы построения деревьев решений. Алгоритм покрытия.

9. Задача классификации. Методы построения деревьев решений. Алгоритм ID 3.

10. Задача классификации. Методы построения деревьев решений. Алгоритм C4.5.

11. Задача классификации. Методы построения правил классификации. Алгоритм построения 1 – правил.

12. Задача классификации. Методы построения правил классификации. Метод Naive Bayes.

13. Большие данные. Свойства больших данных.

14. Машинное обучение, формализация задачи машинного обучения.

15. Признаковое описание объекта.

16. Ответы и типы задач машинного обучения.

17. Машинное обучение. Модель алгоритмов.

18. Метод обучения. Этап обучения и этап применения.

19. Машинное обучение. Функционалы качества.

20. Сведение задачи обучения к задаче оптимизации.

21. Переобучение и обобщение.

22. Пример переобучения (Рунге).

23. Эмпирические оценки обобщающей способности.

24. Примеры задач машинного обучения: задачи классификации и регрессии; задачи ранжирования.

25. Эксперименты в машинном обучении: эксперименты на реальных и синтетических данных.

26. Метрические методы классификации. Формализация задачи.

27. Обобщенный метрический классификатор.

28. Логические методы классификации. Логическая закономерность.

29. Основы вопросы построения логических алгоритмов классификации.

30. Виды закономерностей.

31. Критерии информативности: простые критерии, статистический критерий, энтропийный критерий.

32. Линейные методы классификации.

33. Задача построения разделяющей поверхности.

4 Список литературы

1. Воронов, В.И. Data Mining – технологии обработки больших данных : учебное пособие / В.И. Воронов, Л.И. Воронова, В.А. Усачев. – Москва : Московский технический университет связи и информатики, 2018. – 47 с. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/81324.html (дата обращения: 21.11.2022). – Режим доступа: для авторизир. пользователей.

2. Неделько, В.М. Основы статистических методов машинного обучения : учебное пособие / В.М. Неделько. – Новосибирск : Новосибирский государственный технический университет, 2010. – 72 с. – ISBN 978-5-7782-1385-2. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/45418.html (дата обращения: 21.11.2022). – Режим доступа: для авторизир. пользователей.

3. Ракитский, А.А. Методы машинного обучения : учебно-методическое пособие / А.А. Ракитский. – Новосибирск : Сибирский государственный университет телекоммуникаций и информатики, 2018. – 32 с. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/90591.html (дата обращения: 21.11.2022). – Режим доступа: для авторизир. пользователей.

4. Сараев, П.В. Методы машинного обучения : методические указания и задания к лабораторным работам по курсу / П.В. Сараев. – Липецк : Липецкий государственный технический университет, ЭБС АСВ, 2017. – 48 с. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/83183.html (дата обращения: 21.11.2022). – Режим доступа: для авторизир. пользователей.

5. Сопов, Е.А. Многокритериальные нейроэволюционные системы в задачах машинного обучения и человеко-машинного взаимодействия : монография / Е.А. Сопов, И.А. Иванов. – Красноярск :

Сибирский федеральный университет, 2019. – 160 c. – ISBN 978-5-7638-3969-2. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/100054.html (дата обращения: 21.11.2022). – Режим доступа: для авторизир. пользователей.

6. Теория и практика машинного обучения : учебное пособие / В.В. Воронина, А.В. Михеев, Н.Г. Ярушкина, К.В. Святков. – Ульяновск : Ульяновский государственный технический университет, 2017. – 291 с. – ISBN 978-5-9795-1712-4. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/106120.html (дата обращения: 10.07.2022).

7. Чубукова, И.А. Data Mining : учебное пособие / И.А. Чубукова. – Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 469 с. – ISBN 978-5-4497-0289-0. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/89404.html (дата обращения: 21.11.2022). – Режим доступа: для авторизир. пользователей.

Учебное издание

Носова Людмила Сергеевна

МАШИННОЕ ОБУЧЕНИЕ

Подписано в печать 06.12.2022. Формат 60х84 1/16. Усл. печ. л. 4,48. Тираж 500 экз. Заказ 608.
Учебная типография Федерального государственного бюджетного образовательного учреждения высшего образования «Южно-Уральский государственный гуманитарно-педагогический университет. 454080, Челябинск, проспект Ленина, 69.