

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное образовательное учреждение высшего образования**  
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

## **МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ**

### **УЧЕБНОЕ ПОСОБИЕ (ЛАБОРАТОРНЫЙ ПРАКТИКУМ)**

Направление подготовки:  
09.03.02 «Информационные системы и технологии»

Квалификация выпускника: бакалавр

Ставрополь, 2022

УДК 004.41 (075.8)  
ББК 22.18я73  
Н 63

Печатается по решению  
учебно-методического совета  
Северо-Кавказского федерального  
университета

**Н 63 Методы машинного обучения:** учебное пособие (лабораторный практикум) для студентов направления 09.03.02 «Информационные системы и технологии» / Николаев Е.И. – Ставрополь: Изд-во СКФУ, 2022. – 86 с.

Учебное пособие (лабораторный практикум) по дисциплине «Методы машинного обучения» для студентов направления 09.03.02 «Информационные системы и технологии». Пособие охватывает практические аспекты построения информационных систем на основе методов искусственного интеллекта с применением современных технологий разработки информационных систем и инструментария анализа данных. Основное внимание уделяется теории обучения машин (машинальное обучение, machine learning). Пособие предназначено для студентов, обладающих теоретическими знаниями в области проектирования приложений и практическими навыками программирования (предпочтительно языки C++, C#, Python, Java, R). Цель учебного пособия: сформировать у студентов практические навыки разработки информационных систем на основе методов машинного обучения, искусственного интеллекта, анализа данных; сформировать систему компетенций.

УДК 004.41 (075.8)  
ББК 22.18я73

**Автор:**

канд. техн. наук, доцент **Е.И. Николаев**

**Рецензенты**

© Издательство Северо-Кавказского  
федерального университета, 2022

## ОГЛАВЛЕНИЕ

<b>ВВЕДЕНИЕ.....</b>	<b>4</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 1. ПЕРВИЧНЫЙ АНАЛИЗ ДАННЫХ .....</b>	<b>6</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 2. ВИЗУАЛИЗАЦИЯ ДАННЫХ.....</b>	<b>16</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 3. МЕТРИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ .....</b>	<b>32</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 4. ЛОГИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ .....</b>	<b>42</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 5. РАЗРАБОТКА ЕДИНОГО ПОДХОДА К ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ ..</b>	<b>52</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 6. ПОСТРОЕНИЕ ПАЙПЛАЙНА ОДНОМЕРНОЙ РЕГРЕССИИ.....</b>	<b>60</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 7. ИСПОЛЬЗОВАНИЕ РАЗРАБОТАННОГО ПАЙПЛАЙНА ДЛЯ МНОГОМЕРНОЙ РЕГРЕССИИ .....</b>	<b>65</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 8. ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ .....</b>	<b>72</b>
<b>ЛАБОРАТОРНАЯ РАБОТА 9. КЛАСТЕРИЗАЦИЯ .....</b>	<b>78</b>
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>83</b>
<b>СПИСОК ЛИТЕРАТУРЫ .....</b>	<b>84</b>

## ВВЕДЕНИЕ

**1. Цели и задачи освоения дисциплины.** Учебное пособие по дисциплине «Методы машинного обучения» для студентов направления 09.03.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов искусственного интеллекта. Основное внимание уделяется теории обучения машин (машинае обучение, machine learning).

Основная задача науки и реальной жизни – получение правильных предсказаний о будущем поведении сложных систем на основании их прошлого поведения. Многие задачи, возникающие в практических приложениях, не могут быть решены заранее известными методами или алгоритмами. Это происходит по той причине, что нам заранее не известны механизмы порождения исходных данных или же известная нам информация недостаточна для построения модели источника, генерирующего поступающие к нам данные. Как говорят, мы получаем данные из «черного ящика». В этих условиях ничего не остается, как только изучать доступную нам последовательность исходных данных и пытаться строить предсказания совершенствуя нашу схему в процессе предсказания. Подход, при котором прошлые данные или примеры используются для первоначального формирования и совершенствования схемы предсказания, называется методом машинного обучения (Machine Learning). Машинае обучение – чрезвычайно широкая и динамически развивающаяся область исследований, использующая огромное число теоретических и практических методов. Данное пособие ни в какой мере не претендует на какое-либо исчерпывающее изложение содержания данной области. Основная цель – дать студентам теоретическое представление о современных математических проблемах в области систем искусственного интеллекта, а также познакомить с путями их решения.

Пособие предназначено для студентов, обладающих теоретическими знаниями в области проектирования приложений и практическими навыками программирования (предпочтительно языки С, С++, С#, Python, R). Цель учебного пособия: сформировать у студентов целостный взгляд на современные тенденции в областях машинного обучения, искусственного интеллекта, анализа данных; сформировать систему компетенций.

**2. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы.**

ПК-2 – Способен использовать современные инструментальные средства и технологии программирования при разработке прикладного программного обеспечения вычислительных средств и систем различного функционального назначения.

# ЛАБОРАТОРНАЯ РАБОТА 1. ПЕРВИЧНЫЙ АНАЛИЗ ДАННЫХ

## 1. Цели и задачи

Цель лабораторной работы: изучение программных средств для организации рабочего места специалиста по анализу данных и машинному обучению.

Основные задачи:

- получение программного доступа к данным, содержащимся в источниках различного типа;
- выполнение предварительного анализа данных и получение обобщенных характеристик наборов данных;
- исследование простых методов визуализации данных;
- изучение основных библиотек Python для работы с данными.

## 2. Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с базовыми принципами языка Python, используя следующие источники: [1-5]. Особое внимание необходимо уделить репозитарию [5] с исходными кодами.

## 3. Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

### 3.1 Учебная задача

Необходимо организовать подготовку данных для построения модели (допустим модели классификации). В качестве данных выбран набор данных об ирисах Фишера. Это, пожалуй, самый известный набор данных, с которого многие начинают исследование алгоритмов машинного обучения.

Данный набор данных предназначен для построения модели классификации. Данные о 150 экземплярах ириса (рис. 1.1), по 50 экземпляров из трёх видов – Ирис щетинистый (*Iris setosa*), Ирис виргинский (*Iris virginica*) и Ирис разноцветный (*Iris versicolor*). Для каждого экземпляра измерялись четыре характеристики (в сантиметрах):

- 1) длина наружной доли околоцветника (sepal length);
- 2) ширина наружной доли околоцветника (sepal width);
- 3) длина внутренней доли околоцветника (petal length);
- 4) ширина внутренней доли околоцветника (petal width).

На основании этого набора данных требуется построить правило классификации, определяющее вид растения по данным измерений. Это задача многоклассовой классификации, так как имеется три класса – три вида ириса.

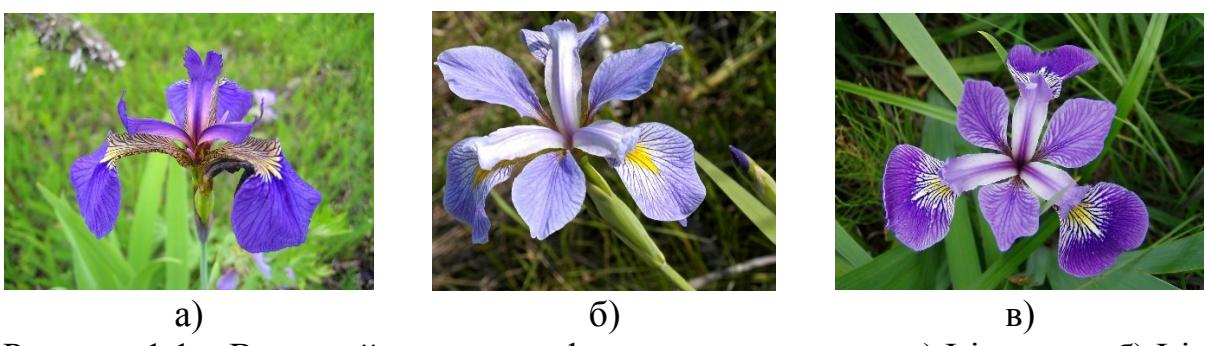


Рисунок 1.1 – Внешний вид классифицируемых ирисов: а) *Iris setosa*; б) *Iris virginica*; в) *Iris versicolor*

1. Необходимо скачать набор данных из репозитория Center for Machine Learning and Intelligent Systems (необходим только один текстовый файл с данными измерений): <http://archive.ics.uci.edu/ml/datasets/Iris>.

Файл Iris.data при просмотре выглядит следующим образом (рис. 1.2):

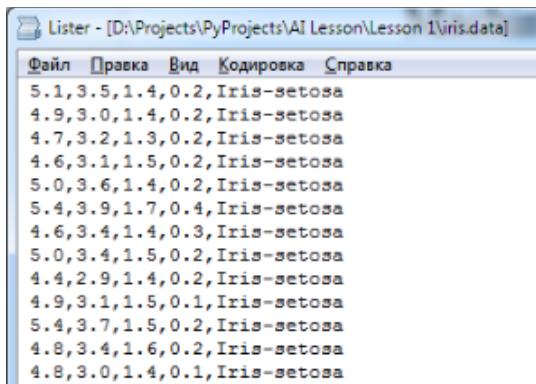


Рисунок 1.2 – Внешний вид данных файла Iris.data

2. Использовать текстовые редакторы для просмотра и анализа данных из определенных наборов – нерациональный вариант. Поэтому запустим Jupyter Notebook и начнем работать с загруженным набором с использованием среды Python. Используем метод `genfromtxt()` из пакета `scipy`.

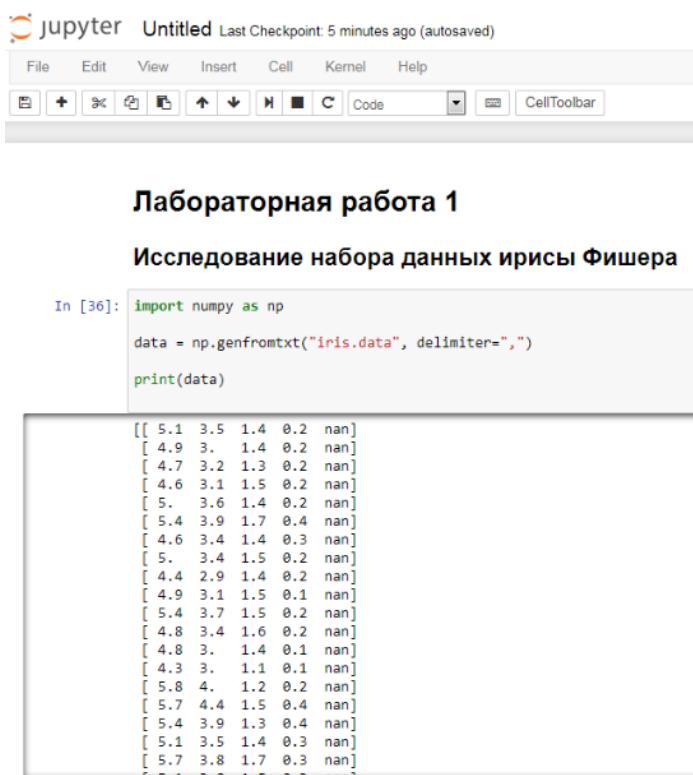


Рисунок 1.3 – Загрузка данных файла Iris.data

Метод `genfromtxt()` возвращает массив `numpy` (тип `numpy.ndarray`). Следует обратить внимание, что пятый столбец содержит неопределенные значения `numpy.NaN` (объясните – почему?).

3. Производить вывод всего источника данных – нерациональный путь. В реальных задачах данных может оказаться слишком много, поэтому

чаще всего используют подвыборку данных для поверхностного обзора исследуемой обучающей выборки (рис. 1.4).

```
In [37]: print ( "Data type : ", type(data) )
print ( "Data shape : ", data.shape )
print ( data[:10] )

Data type : <class 'numpy.ndarray'>
Data shape : (150, 5)
[[ 5.1  3.5  1.4  0.2  nan]
 [ 4.9  3.  1.4  0.2  nan]
 [ 4.7  3.2  1.3  0.2  nan]
 [ 4.6  3.1  1.5  0.2  nan]
 [ 5.  3.6  1.4  0.2  nan]
 [ 5.4  3.9  1.7  0.4  nan]
 [ 4.6  3.4  1.4  0.3  nan]
 [ 5.  3.4  1.5  0.2  nan]
 [ 4.4  2.9  1.4  0.2  nan]
 [ 4.9  3.1  1.5  0.1  nan]]
```

Рисунок 1.4 – Начальное исследование Iris.data

Из представленного фрагмента видно, что data – это двумерный массив размером 150x5, или можно сказать, что это одномерный массив, каждый элемент которого также одномерный массив размером 5 элементов.

4. В рамках данной задачи необходимо все-таки получить значения пятого столбца. Для этого желательно использовать другой подход (рис. 1.5):

```
In [69]: data1 = np.genfromtxt("iris.data", delimiter=",", dtype=None)
print(data1.shape)
print(type(data1))
print(type(data1[0]))
print(type(data1[0][4]))
print(data1[:10])

(150,)
<class 'numpy.ndarray'>
<class 'numpy.void'>
<class 'numpy.bytes_ '>
[(5.1, 3.5, 1.4, 0.2, b'Iris-setosa') (4.9, 3.0, 1.4, 0.2, b'Iris-setosa')
 (4.7, 3.2, 1.3, 0.2, b'Iris-setosa') (4.6, 3.1, 1.5, 0.2, b'Iris-setosa')
 (5.0, 3.6, 1.4, 0.2, b'Iris-setosa') (5.4, 3.9, 1.7, 0.4, b'Iris-setosa')
 (4.6, 3.4, 1.4, 0.3, b'Iris-setosa') (5.0, 3.4, 1.5, 0.2, b'Iris-setosa')
 (4.4, 2.9, 1.4, 0.2, b'Iris-setosa') (4.9, 3.1, 1.5, 0.1, b'Iris-setosa')]
```

Рисунок 1.5 – Загрузка данных разного типа в массив

Сразу же желательно (на первоначальных этапах изучения Python) проводить анализ типов различных значений. На рис. 1.6 представлен еще один вариант загрузки данных в массив numpy.ndarray.

```
In [70]: dt = np.dtype("f8, f8, f8, f8, U30")
data2 = np.genfromtxt("iris.data", delimiter=",", dtype=dt)
print(data2.shape)
print(type(data2))
print(type(data2[0]))
print(type(data2[0][4]))
print(data2[:10])

(150,)
<class 'numpy.ndarray'>
<class 'numpy.void'>
<class 'numpy.str_'>
[(5.1, 3.5, 1.4, 0.2, 'Iris-setosa') (4.9, 3.0, 1.4, 0.2, 'Iris-setosa')
 (4.7, 3.2, 1.3, 0.2, 'Iris-setosa') (4.6, 3.1, 1.5, 0.2, 'Iris-setosa')
 (5.0, 3.6, 1.4, 0.2, 'Iris-setosa') (5.4, 3.9, 1.7, 0.4, 'Iris-setosa')
 (4.6, 3.4, 1.4, 0.3, 'Iris-setosa') (5.0, 3.4, 1.5, 0.2, 'Iris-setosa')
 (4.4, 2.9, 1.4, 0.2, 'Iris-setosa') (4.9, 3.1, 1.5, 0.1, 'Iris-setosa')]
```

Рисунок 1.6 – Загрузка данных в массив с типом, определяемым пользователем

Поясните различие в структурах данных, получаемых с использованием представленных листингов.

5. Было загружено 150 элементов данных, но даже при такой маленькой выборке невозможно что-либо сказать о наборе данных. Для получения дополнительной информации необходимо визуализировать загруженные данные. В нашем случае каждый элемент данных представлен вещественными признаками – это существенно упрощает визуализацию (рис. 1.7). Но сложность заключается в том, что приходится работать с элементами 4-мерного пространства, поэтому строится не графическое представление распределения, а отдельные проекции.

6. Уже из графического распределения на рис. 1.7 видно, что тип ирисов Setosa хорошо отделяется. На данном графике представлено отображение в плоскости признаков ('Sepal Width', 'Sepal Length'). Но исследователь имеет возможность построить столько графиков, сколько необходимо для глубокого анализа данных. Изменим ячейку In[72] в соответствии с листингом, представленным на рис. 1.8.

```
In [72]: import matplotlib as mpl
import matplotlib.pyplot as plt

# Данные из отдельных столбцов
sepal_length = [] # Sepal Length
sepal_width = [] # Sepal Width
petal_length = [] # Petal Length
petal_width = [] # Petal Width

# Выполняем обход всей коллекции data2
for dot in data2:
    sepal_length.append(dot[0])
    sepal_width.append(dot[1])
    petal_length.append(dot[2])
    petal_width.append(dot[3])

# Строим графики по проекциям данных
# Учитываем, что каждые 50 типов ирисов идут последовательно
plt.figure(1)
setosa, = plt.plot(sepal_length[:50], sepal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], sepal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], sepal_width[100:150], 'bs', label='Verginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')

plt.show()
```

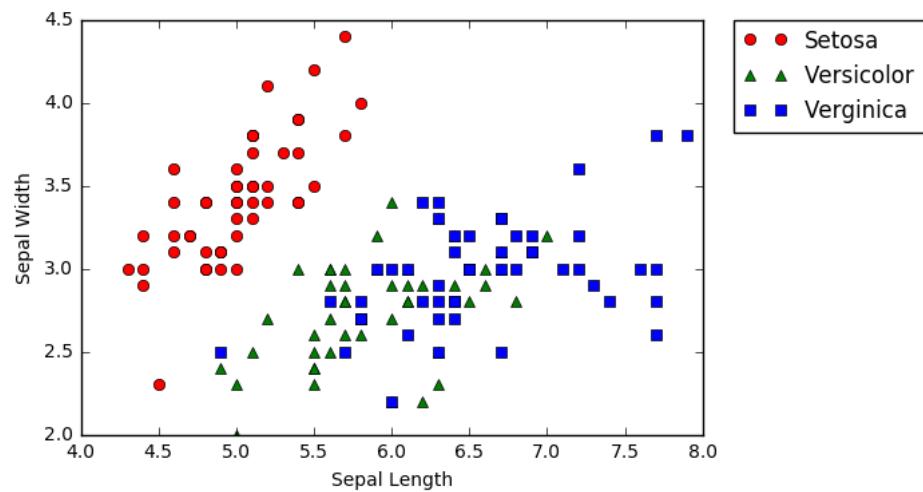


Рисунок 1.7 – Простейший анализ данных по графическому представлению

```
In [74]: import matplotlib as mpl
import matplotlib.pyplot as plt

# Данные из отдельных столбцов
sepal_length = [] # Sepal Length
sepal_width = [] # Sepal Width
petal_length = [] # Petal Length
petal_width = [] # Petal Width

# Выполняем обход всей коллекции data2
for dot in data2:
    sepal_length.append(dot[0])
    sepal_width.append(dot[1])
    petal_length.append(dot[2])
    petal_width.append(dot[3])

# Строим графики по проекциям данных
# Учитываем, что каждые 50 типов ирисов идут последовательно
plt.figure(1)
setosa, = plt.plot(sepal_length[:50], sepal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], sepal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], sepal_width[100:150], 'bs', label='Verginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')

plt.figure(2)
setosa, = plt.plot(sepal_length[:50], petal_length[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], petal_length[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], petal_length[100:150], 'bs', label='Verginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Petal Length')

plt.figure(3)
setosa, = plt.plot(sepal_length[:50], petal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], petal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], petal_width[100:150], 'bs', label='Verginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')

plt.show()
```

Рисунок 1.8 – Построение простейшего графика для отображения различных проекций данных

Вывод данного кода представлен на рис. 1.9.

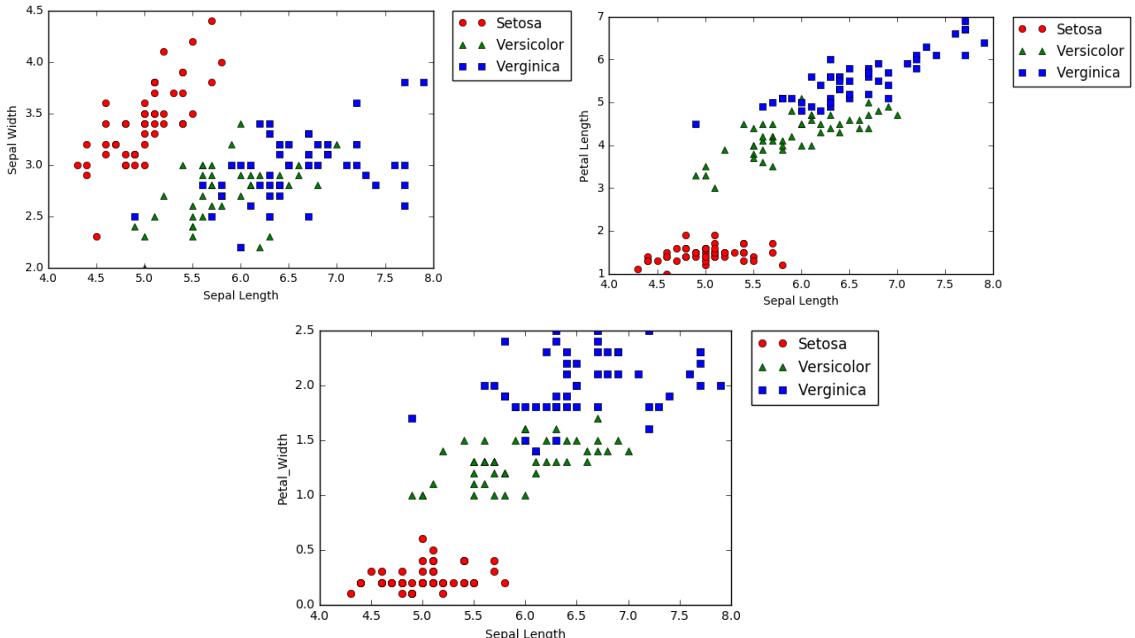


Рисунок 1.9 – Вывод графика для отображения различных проекций данных

Из графиков 1.9 уже хорошо видно, что множество Setosa хорошо отделимо, а множества Versicolor и Virginica представляют собой множества, разделение которых является непростой задачей.

Постройте другие проекции исходных данных. Сколько всего различных проекций можно построить для данного набора данных?

### 3.2 Важные замечания

1. Несмотря на кажущуюся простоту и «понятность» данных в результате визуализации, исследователь не должен делать поспешных выводов (например, было бы ошибочно делать вывод по рис. 1.9 о том, что ирисы Setosa те, у которых petal width менее 0,75). Следует помнить, что цель первичного исследования данных – получение представления о структуре и природе данных, а не построение модели предсказания, классификации и т.п.

1. В качестве среды разработки используйте языки программирования Python, Java или C#. По согласованию с преподавателем студент может самостоятельно выбрать язык программирования и среду разработки (при этом студенту необходимо критически обосновать свой выбор).

2. При выборе набора данных (data set) на ресурсах [9, 10] необходимо согласовать свой выбор с другими студентами группы и преподавателем, так как работа над одинаковыми наборами данных недопустима.

3. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

### 3.3 Индивидуальное задание

1. Подберите набор данных на ресурсах [9, 10] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.

2. Проведите первичный анализ данных. В результате анализа данных студент должен предоставить следующую информацию о наборе данных:

2.1. Описание набора данных, пояснения, позволяющие лучше понять природу данных. Назначение набора данных и возможные модели, которые можно построить на основе данного набора данных (практические задачи, решаемые с использованием данного обучающего набора данных). Описание каждого признака и его тип.

2.2. Форма набора данных: количество элементов набора, количество признаков, количество пропущенных значений, среднее значение отдельных признаков, максимальные и минимальные значения отдельных признаков и прочие показатели. Предположения, которые можно сделать, проведя первичный анализ.

2.3. Графические представления, позволяющие судить о неоднородности исследуемого набора данных. Построение графиков желательно произвести по нескольким проекциям.

### 4. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

## **5. Контрольные вопросы**

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?

2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.

3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?

## **6. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-3, 9, 10].

## ЛАБОРАТОРНАЯ РАБОТА 2. ВИЗУАЛИЗАЦИЯ ДАННЫХ

### 1. Цели и задачи

Цель лабораторной работы: изучение программных средств для визуализации наборов данных.

Основные задачи:

- установка и настройка matplotlib, seaborn;
- изучение основных типов графиков библиотеки matplotlib;
- изучение основных типов графиков библиотеки seaborn;
- получение навыков анализа данных по визуальным представлениям данных.

### 2. Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с базовыми принципами языка Python, используя следующие источники: [1-5]. Особое внимание необходимо уделить репозитарию [5] с исходными кодами.

### 3. Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

#### 3.1 Учебная задача

Выполним анализ набора данных «Предсказание ухода клиента». Данный набор данных используется в качестве учебного набора при изучении методов прогнозирования. Набор представляет собой данные об активности клиентов телекоммуникационной компании (количество часов разговоров, видеозвонков, ночные и дневные разговоры и прочие). Набор данных подходит для обучения моделей логистической регрессии, моделей

классификации (CNN, kNN, Logic tree). Набор данных можно получить в репозитории [5] или на портале Kaggle [4].

Рассмотрим основные признаки, представленный в наборе. Загрузим набор данных с использованием pandas и выведем признаки набора данных (рисунок 2.1).

```

1 import numpy as np
2 import pandas as pd
3 from matplotlib import pyplot as plt
4 import seaborn as sns
5 %matplotlib inline

1 data_path = "../datasets/telecom_churn/telecom_churn.csv"
2 data = pd.read_csv(data_path)
3 data.head()

```

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	To d char
0	KS	128	415	No	Yes	25	265.1	110	45.
1	OH	107	415	No	Yes	26	161.6	123	27.
2	NJ	137	415	No	No	0	243.4	114	41.

Рисунок 2.1 – Загрузка данных и получение и превичный анализ признаков

Набор данных `telecom_churn.csv` содержит большое количество признаков. Для детального изучения воспользуемся методом `info()` класса `DataFrame` (рисунок 2.2).

```
1 data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
State                      3333 non-null object
Account length              3333 non-null int64
Area code                   3333 non-null int64
International plan          3333 non-null object
Voice mail plan             3333 non-null object
Number vmail messages       3333 non-null int64
Total day minutes           3333 non-null float64
Total day calls              3333 non-null int64
Total day charge             3333 non-null float64
Total eve minutes            3333 non-null float64
Total eve calls              3333 non-null int64
Total eve charge             3333 non-null float64
Total night minutes          3333 non-null float64
Total night calls             3333 non-null int64
Total night charge            3333 non-null float64
Total intl minutes            3333 non-null float64
Total intl calls              3333 non-null int64
Total intl charge             3333 non-null float64
Customer service calls       3333 non-null int64
Churn                       3333 non-null bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
```

Рисунок 2.2 – Информация о признаках набора данных

Графики, используемые при анализе данных, делят не по библиотекам, с использованием которых они строятся, а по типам признаков, для анализа которых предназначены графики.

### 3.1.1. Визуализация количественных признаков

Для представления распределения простого количественного признака подходит обычная гистограмма, содержащаяся во всех библиотеках (рисунок 2.3).

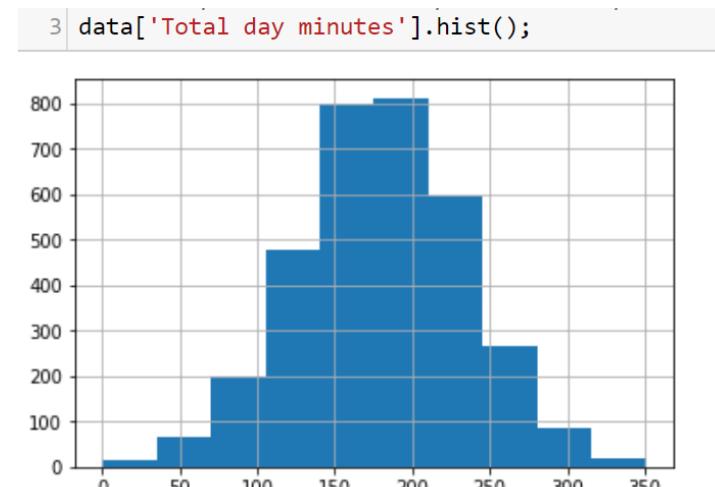


Рисунок 2.3 – Информация о признаках набора данных

Для построения гистограммы вызывается метод **hist()** класса DataFtrame. На самом деле используется метод из библиотеки matplotlib. Метод **hist()** можно использовать для построения гистограмм по нескольким признакам (рисунок 2.4). При этом неколичественные признаки игнорируются.

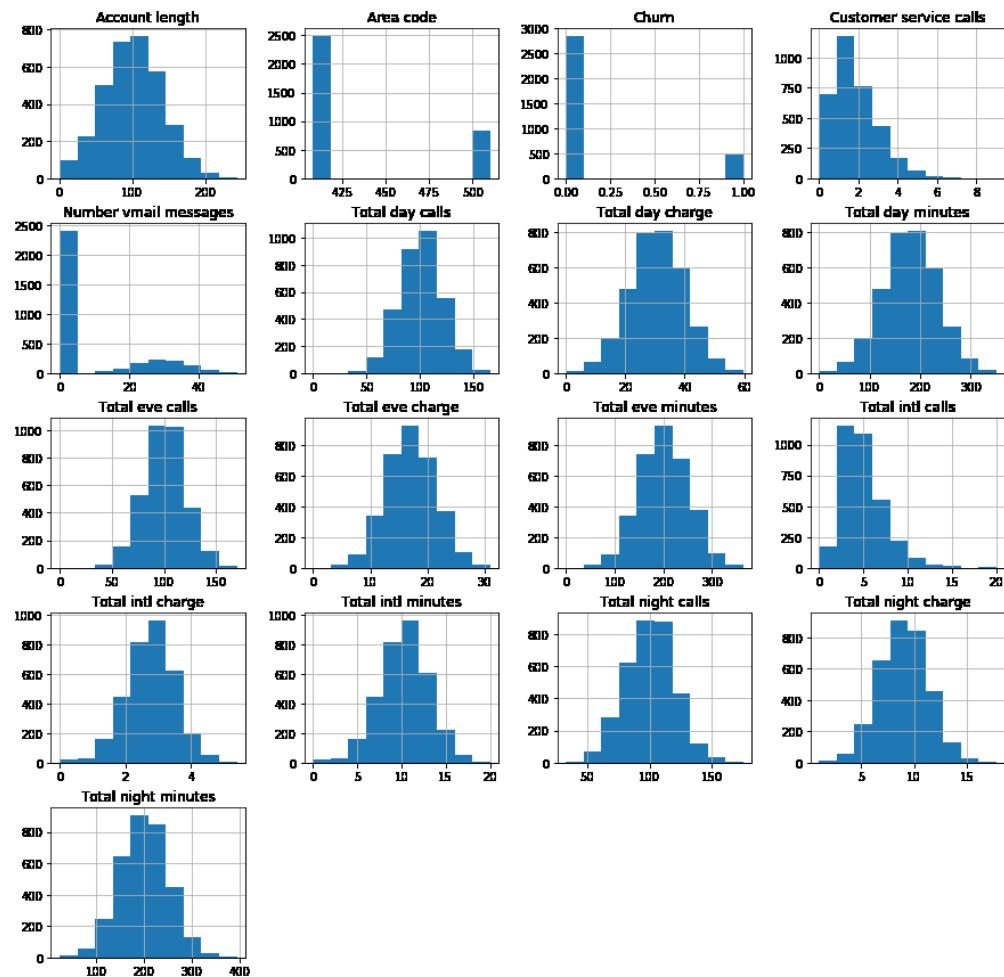


Рисунок 2.4 – Применение метода **hist()** для визуализации распределения нескольких признаков

Аналогичный тип графика можно получить с использованием **matplotlib** (рисунок 2.5). Если необходимо построить график распределения, аналогичный представленному на рисунке 2.3, то нужно выполнить дополнительные расчеты (рисунок 2.6).

```
1 plt.bar(data.index, data['Total day minutes'])
2 plt.show()
```

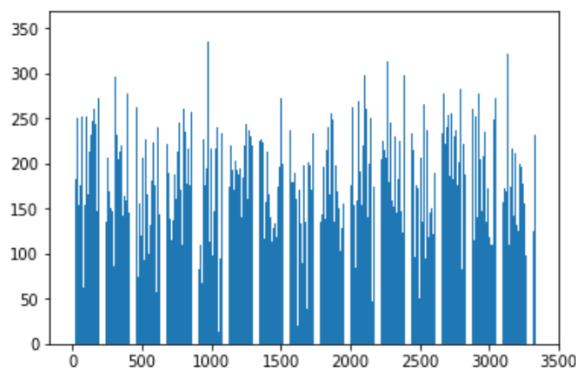


Рисунок 2.5 – Построение гистограммы с использованием matplotlib

```
1 hist = data['Total day minutes'].value_counts()
2 plt.bar(hist.index, hist);
```

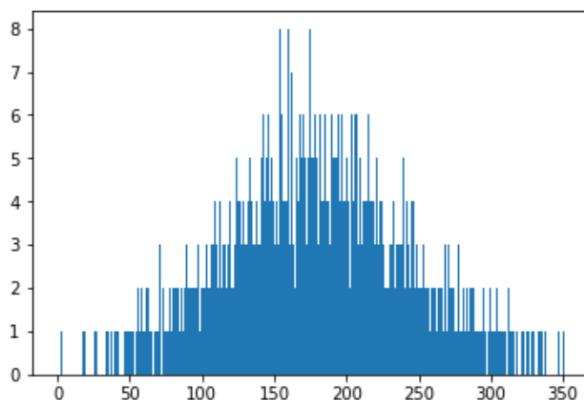


Рисунок 2.6 – Использование matplotlib для представления распределения значений признака

Один из эффективных типов графиков для анализа количественных признаков – это «ящие с усами» (boxplot). На рисунке 2.7 показан код и реализованный график. Для анализа нескольких признаков графики boxplot также эффективны. На рисунке 2.8 представлен код и результат построения графиков для анализа пяти штатов с максимальным объемом дневных звонков.

```
5 sns.boxplot(data['Total day minutes']);
```

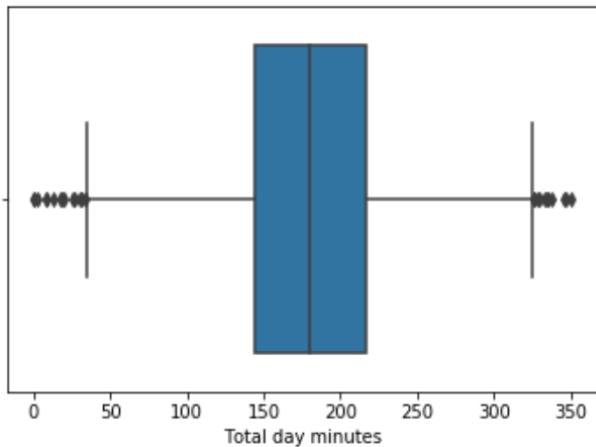


Рисунок 2.7 – График «ящик с усами» для отдельного признака

```
1 top_data = data[['State', 'Total day minutes']]
2 top_data = top_data.groupby('State').sum()
3 top_data = top_data.sort_values('Total day minutes', ascending=False)
4 top_data = top_data[:5].index.values
5 sns.boxplot(y='State',
6               x='Total day minutes',
7               data=data[data.State.isin(top_data)], palette='Set3');
```

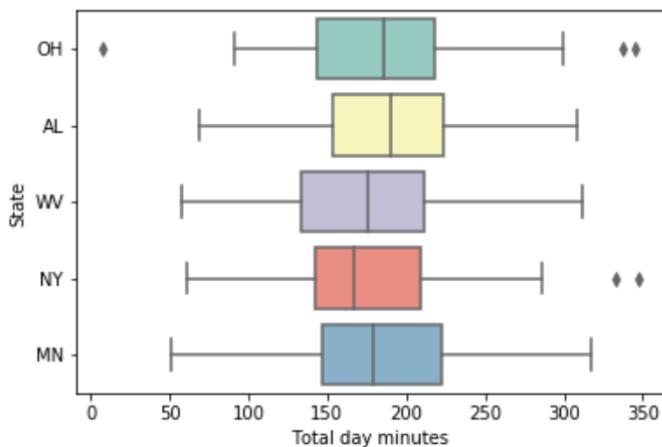


Рисунок 2.8 – Использование boxplot для анализа признака для пяти штатов

График boxplot состоит из коробки, усов и точек. Коробка показывает интерквартильный размах распределения, то есть соответственно 25% (первая квартиль,  $Q_1$ ) и 75% ( $Q_3$ ) перцентили. Черта внутри коробки обозначает медиану распределения (можно получить с использованием метода `median()` в `pandas` и `numpy`). Усы отображают весь разброс точек кроме выбросов, то есть минимальные и максимальные значения, которые попадают в промежуток ( $Q_1 - 1,5 \cdot IQR$ ,  $Q_3 + 1,5 \cdot IQR$ ), где  $IQR = Q_3 - Q_1$  – интерквартильный размах. Точками на графике обозначаются выбросы (outliers), то есть те

значения, которые не вписываются в промежуток значений, заданный усами графика (рисунок 2.9).

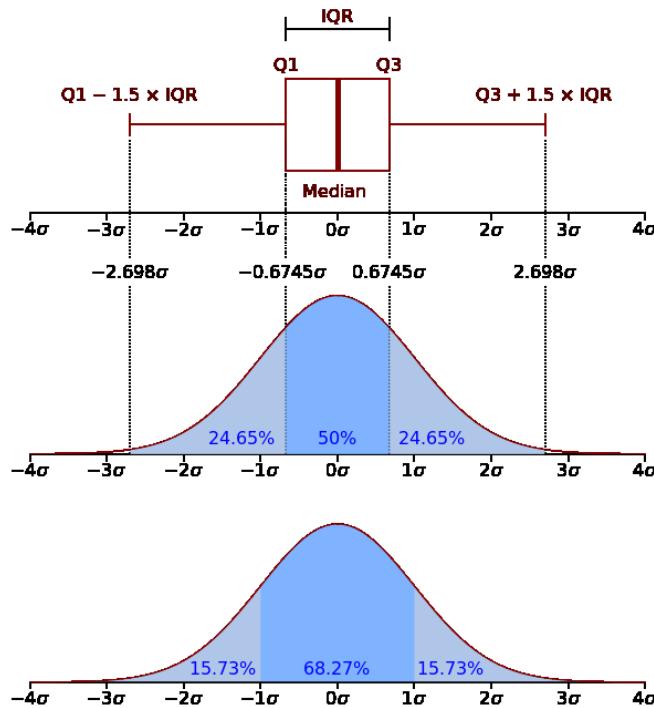


Рисунок 2.9 – Структура графика типа «ящик с усами»

### 3.1.2. Категориальные признаки

Типичным категориальным признаком в анализируемом наборе данных является «Штат» (State). Под категориальный признак подходит также «Отказ» (Churn) (хотя он является логическим). На рисунке 2.10 представлены графики типа countplot() из библиотеки seaborn, которые строят гистограммы, но не по сырым данным, а по расчитанному количеству разных значений признака.

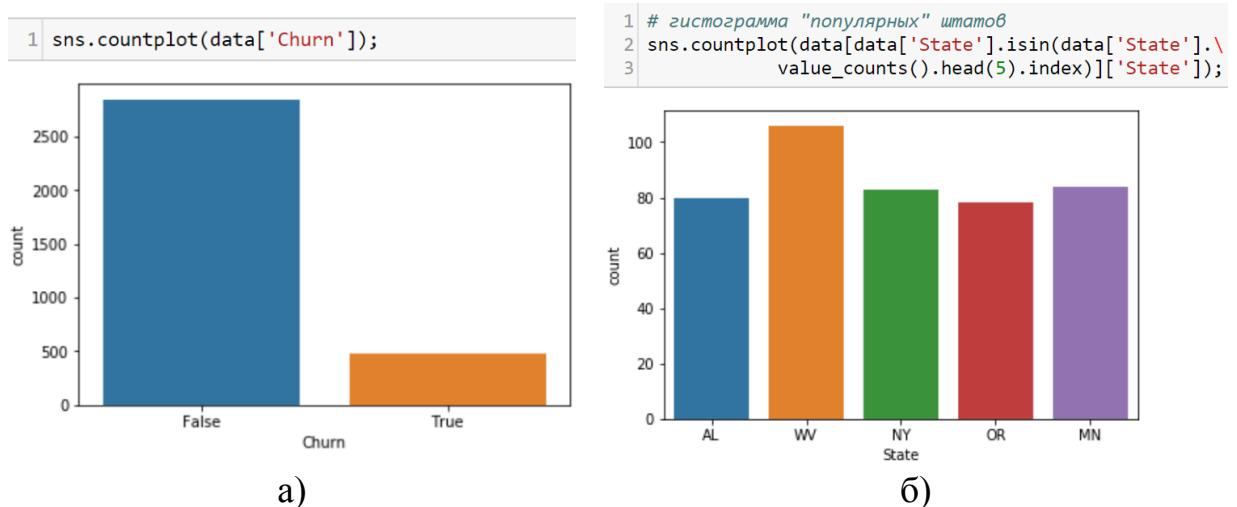


Рисунок 2.10 – График countplot: а) визуализация распределения признака Churn; б) визуализация пяти популярных штатов

### 3.1.3. Визуализация соотношения количественных признаков

Одним из вариантов визуализации соотношения количественных признаков является диаграмма по нескольким признакам (рисунки 2.4, 2.8). Рассмотрим пример демонстрирующий сравнение распределений показателей, связанных с финансовыми затратами клиентов. Упрощенно, можно сказать, что это все показатели, содержащие подстроку «charge» в имени показателя. На рисунке 2.11 представлен код для отбора требуемых показателей.

```

1 # Отбор числовых признаков, содержащих слово 'charge'
2 feats = [f for f in data.columns if 'charge' in f]
3 feats

['Total day charge',
 'Total eve charge',
 'Total night charge',
 'Total intl charge']

```

Рисунок 2.11 – Отбор показателей, связанных с затратами клиентов

После отбора интересующих показателей можно построить диаграммы для сравнения (рисунок 2.12).

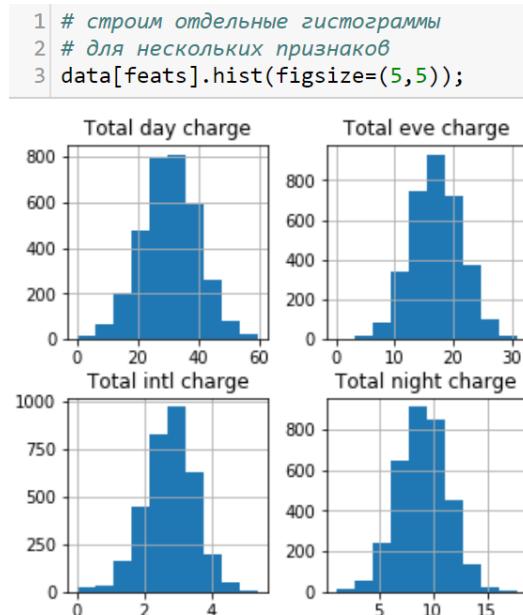


Рисунок 2.12 – Диаграммы для сравнения распределения числовых показателей

Часто используют попарное сравнение признаков для обеспечения широкого взгляда на набор данных (рисунок 2.13). На диагональных графиках рисунка 2.13 представлены гистограммы распределения отдельного признака, на внедиагональных позициях – попарные распределения.

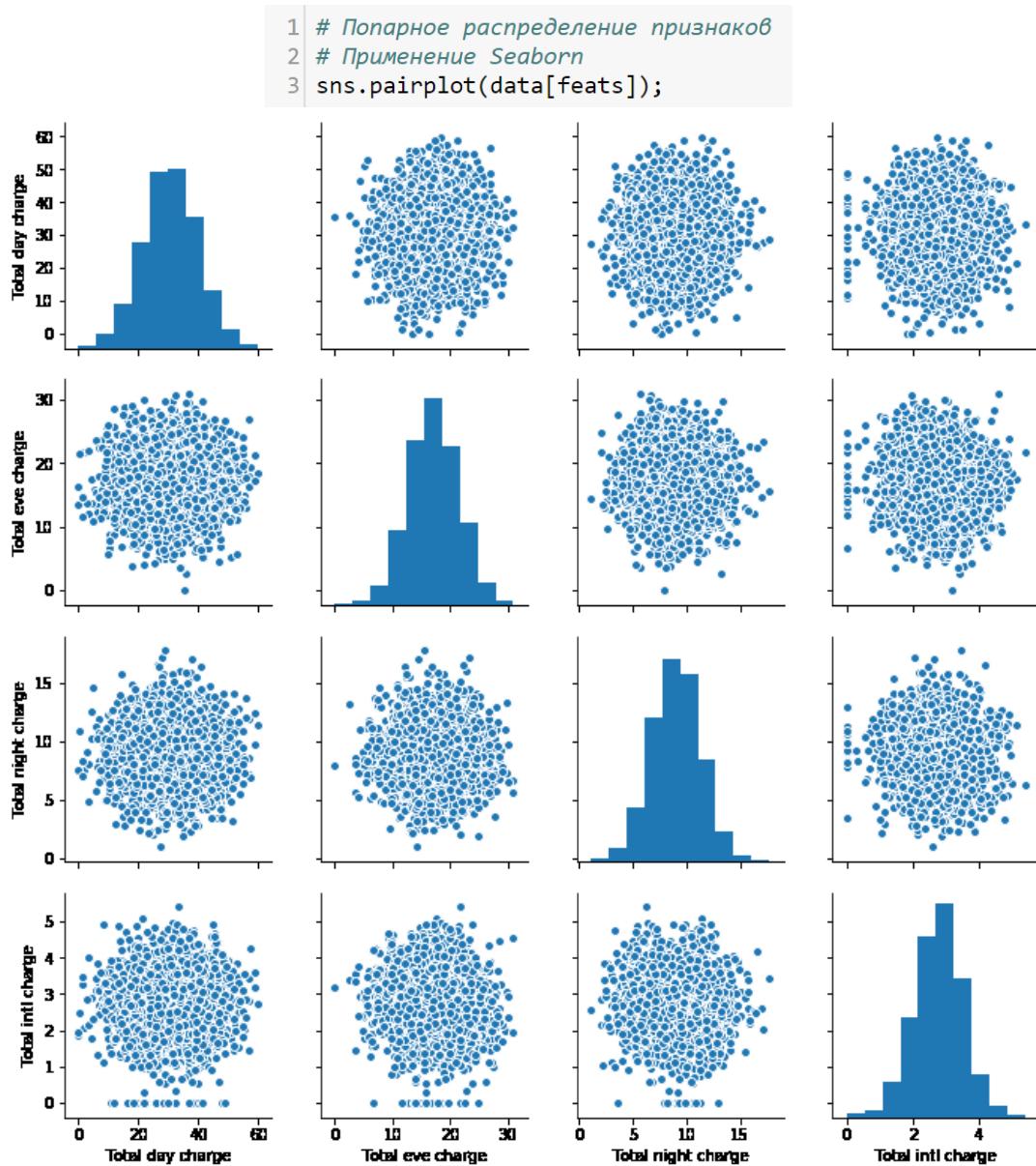


Рисунок 2.13 – Попарное распределение признаков

Можно реализовать более сложные графики. Например, если требуется добавить к существующим признакам, целевой признак Churn (количество отказов) и раскрасить разные типы элементов, то можно воспользоваться попарными распределениями, но с отображением подмножеств отказов (рисунок 2.14).

До сих пор использовались возможности библиотеки seaborn, а также методы pandas (которые производят визуализацию, обращаясь к библиотеке matplotlib). Библиотека matplotlib наиболее известная и широко применяемая при анализе данных в рамках стека технологий python.

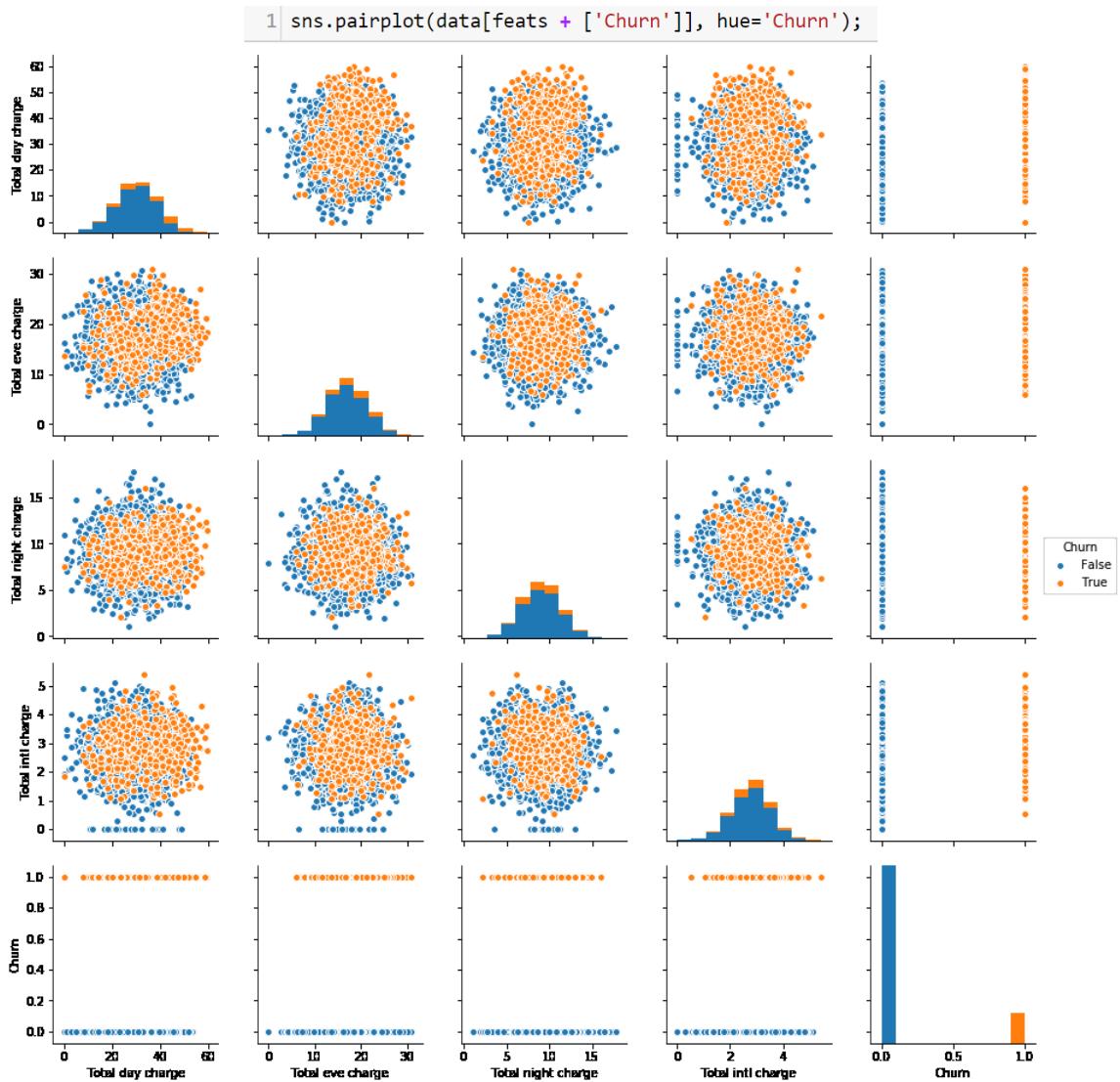


Рисунок 2.14 – Попарное распределение признаков с визуализацией отказов

На рисунке 2.15 показан пример использования графика scatter библиотеки matplotlib, предназначенного для вывода множества точек.

```

1 plt.scatter(data['Total day charge'],
2             data['Total intl charge'],
3             color='lightblue', edgecolors='blue')
4 plt.xlabel('Дневные начисление')
5 plt.ylabel('Международн. начисление')
6 plt.title('Распределение по 2 признакам');

```

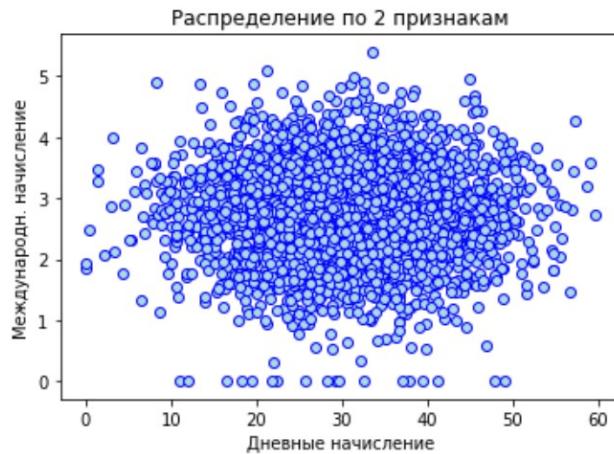


Рисунок 2.15 – График scatter библиотеки matplotlib

На рисунке 2.16 показан пример более тонкой настройки параметров графика.

```

1 # Раскрашивание данных
2 # Цвет в зависимости от ухода клиента
3 c = data['Churn'].map({False: 'lightblue', True: 'orange'})
4 edge_c = data['Churn'].map({False: 'blue', True: 'red'})
5 # Настойка графика
6 plt.scatter(data['Total day charge'], data['Total intl charge'],
7             color=c, edgecolors=edge_c
8             )
9 plt.xlabel('Дневные начисление')
10 plt.ylabel('Международн. начисление');

```

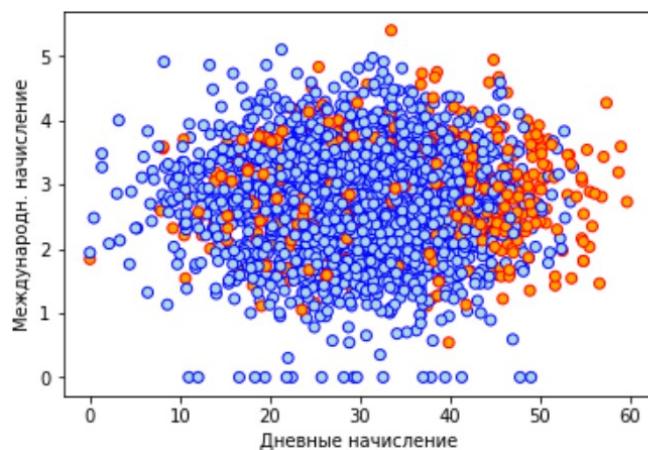


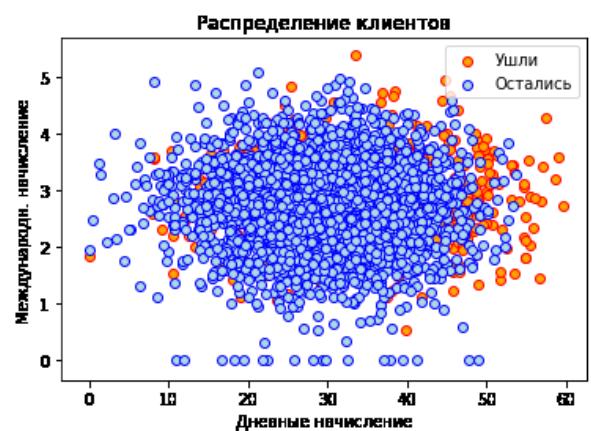
Рисунок 2.16 – Настройка графика: цвет точки зависит от целевого значения признака

График на рисунке 2.16 можно построить различными способами, например, можно добавлять множества точек отдельными подмножествами, указывая параметры визуализации для каждого подмножества (рисунок 2.17).

```

4 # Ушедшие клиенты
5 data_churn = data[data['Churn']]
6 # Оставшиеся клиенты
7 data_loyal = data[~data['Churn']]
8
9 plt.scatter(data_churn['Total day charge'],
10             data_churn['Total intl charge'],
11             color='orange',
12             edgecolors='red',
13             label='Ушли'
14         )
15 plt.scatter(data_loyal['Total day charge'],
16             data_loyal['Total intl charge'],
17             color='lightblue',
18             edgecolors='blue',
19             label='Остались'
20         )
21 plt.xlabel('Дневные начисления')
22 plt.ylabel('Международн. начисление')
23 plt.title('Распределение клиентов')
24 plt.legend();

```



a)

б)

Рисунок 2.17 – Построение отдельных подмножеств с легендой; а) исходный код; б) полученный график

В реальных задачах машинного обучения при первичном анализе данных необходимо выявить корреляции признаков обучающей выборки. В пакете Pandas имеется встроенный инструмент для этого – метод **corr()** класса DataFrame. На рисунке 2.18 показан фрагмент вывода этой функции.

```

1 # Применяется функция corr() из Pandas
2 data.corr()

```

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total
Account length	1.000000	-0.012463	-0.004628	0.006216	0.038470	0.006214	-0.006757	0.01
Area code	-0.012463	1.000000	-0.001994	-0.008264	-0.009646	-0.008264	0.003580	-0.01
Number vmail messages	-0.004628	-0.001994	1.000000	0.000778	-0.009548	0.000776	0.017562	-0.00
Total day minutes	0.006216	-0.008264	0.000778	1.000000	0.006750	1.000000	0.007043	0.01

Рисунок 2.18 – Определение коррелирующих признаков набора данных

Полученная матрица имеет размер  $17 \times 17$ . Это незначительный размер (в реальных задачах машинного обучения размеры матриц корреляции имеют порядки  $10^6 - 10^{10}$  и более), но даже для матрицы рассматриваемого набора

данных проанализировать корреляцию признаков вручную – трудоемкая задача. Например, можно использовать скрипты, для выделения больших коэффициентов корреляции. Но лучше использовать специальный тип графика – heatmap (рисунок 2.19).

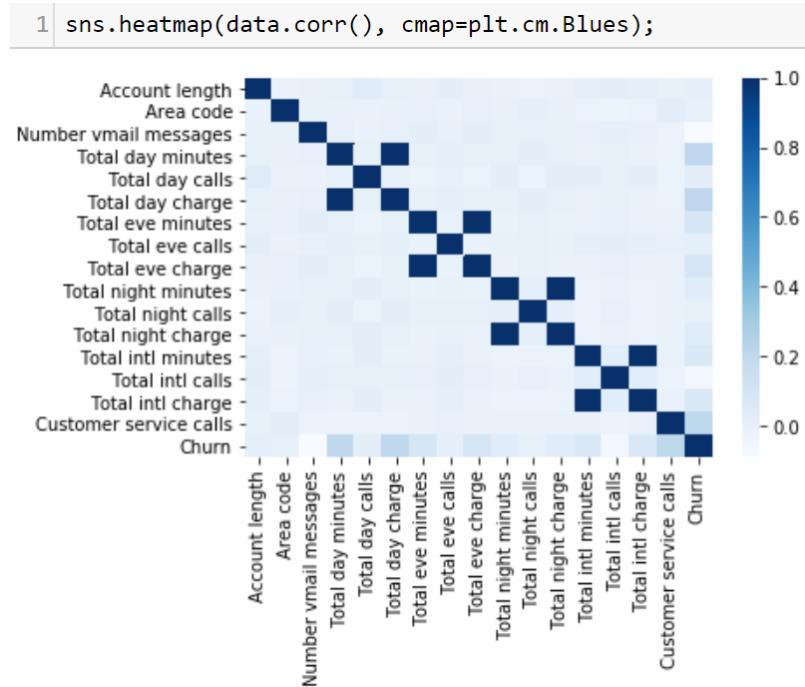


Рисунок 2.19 – Визуализация матрица корреляции с использованием графика типа heatmap

Коррелирующие признаки обычно удаляются и не рассматриваются в процессе обучения.

### 3.2 Важные замечания

- Статья о типах графиков при первичном анализе данных:  
<https://medium.com/open-machine-learning-course/open-machine-learning-course-topic-2-visual-data-analysis-in-python-846b989675cd>
- В качестве среды разработки используйте языки программирования Python, Java или C#. По согласованию с преподавателем студент может самостоятельно может выбрать язык программирования и среду разработки (при этом студенту необходимо критически обосновать свой выбор).

2. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем, так как работа над одинаковыми наборами данных недопустима.

3. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

### **3.3 Индивидуальное задание**

1. Подберите набор данных на ресурсах [3, 4] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.

2. Проведите первичный анализ данных. Особое внимание следует уделить графическому представлению распределений признаков, визуализации взаимосвязей, позволяющие судить о наборе данных. Построение графиков желательно произвести по нескольким проекциям. При анализе данных использовать как можно более разнообразные типы графиков.

## **4. Содержание отчета и его форма**

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.
4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

## 5. Контрольные вопросы

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?
2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.
3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?
4. Перечислите функции Python, которые были изучены в рамках данной лабораторной работы и которые используются для визуализации данных.
5. Какая библиотека python предназначена для управления наборами данных: numpy, pandas, sklearn, opencv, matplotlib?
6. Какая стратегия является нежелательной при обработке пропусков в данных?
  - а) замена пропущенных значений в столбце медианным значением по данному столбцу;
  - б) удаление строк, содержащих пропуски в данных;
  - в) замена пропущенных значений в столбце средним арифметическим значением по данному столбцу;
  - г) замена пропущенных значений в столбце наиболее часто встречающимся значением по данному столбцу;
7. Обоснуйте ответ на следующую проблему предварительной обработки данных: имеется независимая категориальная переменная у, которая представляет собой категориальный признак, определенный на домене {C#, Java, Python, R}. Нужно ли применять к данному целевому признаку OneHotEncoder?

8. Поясните принцип разбиения набора данных на обучающую и тестовую выборку. Какое соотношение «тестовая:обучающая» наиболее оптимально: 20:80, 50:50, 25:75, 5:95, 40:30?

9. Какой код лучше использовать при загрузке данных из csv-файла?

- а) dataset = read\_csv("data.csv")
- б) dataset = import("data.csv")
- в) dataset = read.csv("data.csv")
- г) dataset = import.csv("data.csv")
- д) dataset = read\_xls("data.csv")

## **6. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-4].

## ЛАБОРАТОРНАЯ РАБОТА 3. МЕТРИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ

### 1. Цели и задачи

Цель лабораторной работы: изучение принципов построения информационных систем с использованием метрических методов классификации.

Основные задачи:

- изучение инструментария Python для реализации алгоритмов метрической классификации;
- изучение методов оптимизации параметров метрической классификации;
- освоение модификаций kNN-метода.

### 2. Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с теорией построения метрических классификаторов, используя следующие источники: [1-5]. Особое внимание необходимо уделить репозитарию [5] с исходными кодами.

### 3. Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

#### 3.1 Учебная задача

В рамках данной задачи рассматривается построение классификатора с использованием метода ближайших соседей. В качестве набора данных используются данные об ирисах Фишера.

В рамках данной лабораторной работы рекомендуется использование библиотеки pandas. Pandas – это библиотека Python, предоставляющая широкие возможности для анализа данных. Данные, с которыми работают специалисты Data Science, часто хранятся в табличном формате (.csv, .tsv, .xlsx, ...). С помощью библиотеки Pandas данные удобно загружать, обрабатывать и анализировать с помощью SQL-подобных запросов. Pandas предоставляет широкие возможности визуального анализа табличных данных в связке с библиотеками Matplotlib и Seaborn.

Основными структурами данных в Pandas являются классы Series и DataFrame. Первый из них представляет собой одномерный индексированный массив данных некоторого фиксированного типа. Второй – это двухмерная структура данных, представляющая собой таблицу, каждый столбец которой содержит данные одного типа. Можно представлять её как словарь объектов типа Series. Структура DataFrame отлично подходит для представления реальных данных: строки соответствуют признаковым описаниям отдельных объектов, а столбцы соответствуют признакам.

1. На рис. 3.1 представлен код в Python Notebook для загрузки исходного набора данных.

```
In [8]: import pandas as pd
import numpy as np

data_source = 'iris.data'
d = pd.read_table(data_source, delimiter=',')
d.head(5)
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

Рисунок 3.1 – Использование pandas для загрузки данных

Следует обратить внимание, что первая строка набора данных интерпретировалась как шапка таблицы (название столбцов). Данную неточность необходимо исправить следующим образом (рис. 3.2). В таком случае столбцы получат порядковые номера в качестве названий столбцов.

In [11]:	import pandas as pd import numpy as np  data_source = 'iris.data' d = pd.read_table(data_source, delimiter=',', header=None) d.head()																																				
Out[11]:	<table border="1"><thead><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th></tr></thead><tbody><tr><td>0</td><td>5.1</td><td>3.5</td><td>1.4</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>1</td><td>4.9</td><td>3.0</td><td>1.4</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>2</td><td>4.7</td><td>3.2</td><td>1.3</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>3</td><td>4.6</td><td>3.1</td><td>1.5</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>4</td><td>5.0</td><td>3.6</td><td>1.4</td><td>0.2</td><td>Iris-setosa</td></tr></tbody></table>		0	1	2	3	4	0	5.1	3.5	1.4	0.2	Iris-setosa	1	4.9	3.0	1.4	0.2	Iris-setosa	2	4.7	3.2	1.3	0.2	Iris-setosa	3	4.6	3.1	1.5	0.2	Iris-setosa	4	5.0	3.6	1.4	0.2	Iris-setosa
	0	1	2	3	4																																
0	5.1	3.5	1.4	0.2	Iris-setosa																																
1	4.9	3.0	1.4	0.2	Iris-setosa																																
2	4.7	3.2	1.3	0.2	Iris-setosa																																
3	4.6	3.1	1.5	0.2	Iris-setosa																																
4	5.0	3.6	1.4	0.2	Iris-setosa																																

Рисунок 3.2 – Добавление шапки DataFrame по умолчанию

Исследователь также может дать символьные имена столбцам при загрузке (рис. 3.3).

In [19]:	import pandas as pd import numpy as np  data_source = 'iris.data' d = pd.read_table(data_source, delimiter=',', header=None, names=['sepal_length','sepal_width', 'petal_length','petal_width','answer']) d.head()																																				
Out[19]:	<table border="1"><thead><tr><th></th><th>sepal_length</th><th>sepal_width</th><th>petal_length</th><th>petal_width</th><th>answer</th></tr></thead><tbody><tr><td>0</td><td>5.1</td><td>3.5</td><td>1.4</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>1</td><td>4.9</td><td>3.0</td><td>1.4</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>2</td><td>4.7</td><td>3.2</td><td>1.3</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>3</td><td>4.6</td><td>3.1</td><td>1.5</td><td>0.2</td><td>Iris-setosa</td></tr><tr><td>4</td><td>5.0</td><td>3.6</td><td>1.4</td><td>0.2</td><td>Iris-setosa</td></tr></tbody></table>		sepal_length	sepal_width	petal_length	petal_width	answer	0	5.1	3.5	1.4	0.2	Iris-setosa	1	4.9	3.0	1.4	0.2	Iris-setosa	2	4.7	3.2	1.3	0.2	Iris-setosa	3	4.6	3.1	1.5	0.2	Iris-setosa	4	5.0	3.6	1.4	0.2	Iris-setosa
	sepal_length	sepal_width	petal_length	petal_width	answer																																
0	5.1	3.5	1.4	0.2	Iris-setosa																																
1	4.9	3.0	1.4	0.2	Iris-setosa																																
2	4.7	3.2	1.3	0.2	Iris-setosa																																
3	4.6	3.1	1.5	0.2	Iris-setosa																																
4	5.0	3.6	1.4	0.2	Iris-setosa																																

Рисунок 3.3 – Добавление шапки DataFrame с символьными именами столбцов

2. После загрузки данных можно визуализировать полученный набор данных. Для визуализации будем использовать библиотеку seaborn (рис. 3.4).

```
In [18]: import seaborn as sb  
%matplotlib inline  
sb.pairplot(d)
```

Рисунок 3.4 – Добавление шапки DataFrame по умолчанию

В результате будет выведен график, отображающий распределение объектов попарно по различным признакам (рис. 3.5).

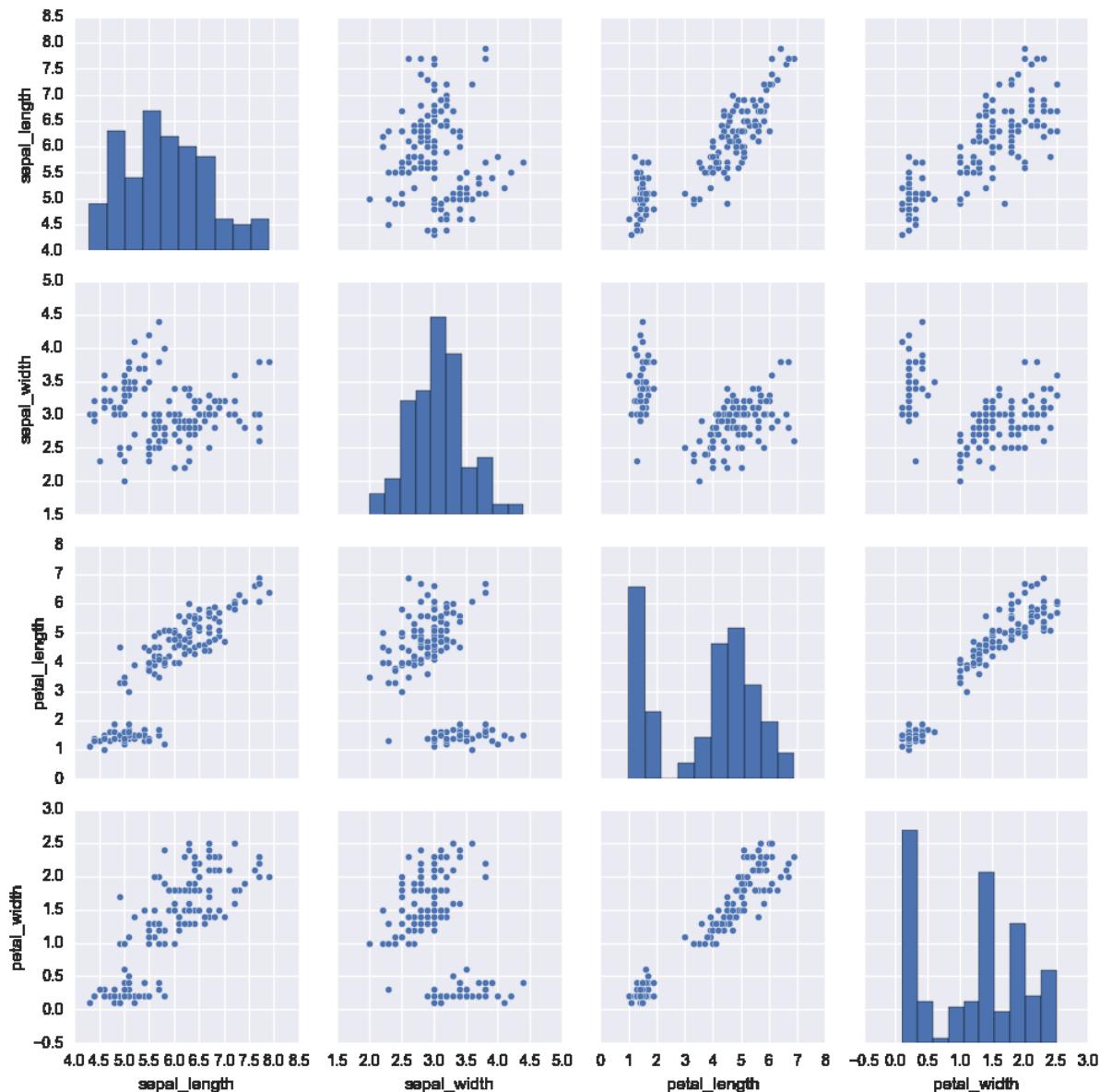


Рисунок 3.5 – Попарное признаковое распределение ирисов

На графике попарного распределения видно преимущество символического обозначения столбцов – график легче интерпретировать. Отдельные классы не отмечаются различными цветами, но видно, что на отдельных подграфиках множества точек разделены. Следует обратить внимание на подграфики, расположенные по диагонали. Подумайте, что они отображают?

3. Для придания отдельным классам своих цветов необходимо указать, по какому признаку разделяются точки (рис. 3.6).

```
In [23]: import seaborn as sb
%matplotlib inline
sb.pairplot(d, hue='answer')
```

Рисунок 3.6 – Построение графика с указанием признака отдельных классов

Результат представлен на рис. 3.7.

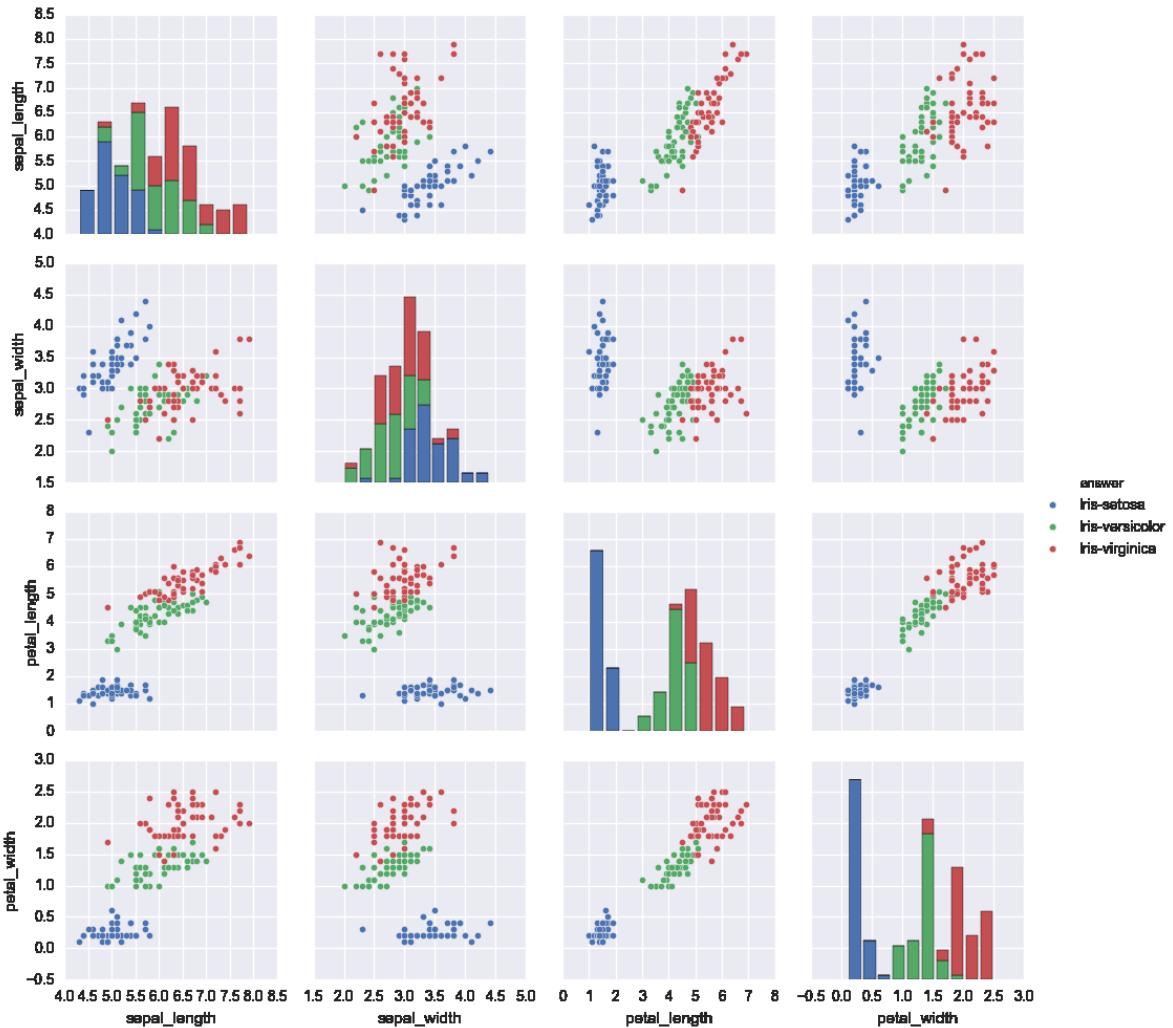


Рисунок 3.7 – Парное признаковое распределение ирисов с разделением на классы

Можно изменить маркеры каждого класса. Для этого необходимо использовать код: `sb.pairplot(d, hue='answer', markers=["o", "s", "D"])`.

4. Перейдем к построению модели. Модель метрической классификации должна обеспечивать следующий алгоритм работы: пользователь вводит новое признаковое описание объекта (объект отсутствует

в обучающей выборке), а алгоритм классификации относит новый объект к одному из классов ирисов.

5. Существует несколько вариаций метода ближайших соседей. Каждая модель предполагает наличие различных параметров для оптимизации. Воспользуемся библиотекой scikit для построения классификатора (рис. 3.8).

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

X_train = d[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]

y_train = d['answer']

K = 3

# Создание и настройка классификатора
knn = KNeighborsClassifier(n_neighbors=K)
# построение модели классификатора (процедура обучения)
knn.fit(X_train, y_train)

# Использование классификатора
# Объявление признаков объекта
X_test = np.array([[1.2, 1.0, 2.8, 1.2]])
# Получение ответа для нового объекта
target = knn.predict(X_test)
print(target)

['Iris-versicolor']
```

Рисунок 3.8 – Основные этапы решения задачи классификации методом ближайших соседей с использованием библиотеки scikit

Для прелставленного объекта X\_test попробуйте поменять значение признаков и проследите за изменением значения выходного класса.

6. Модель построена и выдает ответ для новых (отсутствующих в исходной выборке) объектов. Но, анализируя код на рис. 3.8, следует отметить следующие недостатки такого подхода:

- в качестве количества ближайших соседей выбрано значение K=3, выбор данного значения не обосновывается, но в данном методе именно данный параметр должен оптимизироваться;

- отсутствует какое-либо графическое представление модели, нет визуализации процесса принятия решения.

Исправим данные недостатки.

7. Займемся обоснованием выбора оптимального значения количества ближайших соседей. Для этого будем использовать простейшую оценку качества hold-out (рис. 3.9).

```
from sklearn.model_selection import train_test_split

X_train, X_holdout, y_train, y_holdout = \
    train_test_split(d[['sepal_length', 'sepal_width',
                        'petal_length', 'petal_width']],
                     d['answer'],
                     test_size=0.3,
                     random_state=17)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_holdout)
accur = accuracy_score(y_holdout, knn_pred)
print('accuracy: ', accur)

accuracy:  0.977777777778
```

Рисунок 3.9 – Оценка точности классификатора с использованием методики hold-out

В качестве эксперимента попробуйте поменять значение количества соседей и рассмотрите изменение точности классификатора.

8. Еще одна оценка качества – cross validation (CV) error. На рис. 3.10 показан алгоритм получения оценки точности классификации CV и процедура выбора оптимального значения количества соседей в алгоритме kNN на основе данной оценки.

```

from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt

# Значения параметра K
k_list = list(range(1,50))
# Пустой список для хранения значений точности
cv_scores = []
# В цикле проходим все значения K
for K in k_list:
    knn = KNeighborsClassifier(n_neighbors=K)
    scores = cross_val_score(knn, d.ix[ :, 0:4 ], d[ 'answer' ], cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

# Вычисляем ошибку (misclassification error)
MSE = [1-x for x in cv_scores]

# Строим график
plt.plot(k_list, MSE)
plt.xlabel('Количество соседей (K)');
plt.ylabel('Ошибка классификации (MSE)')
plt.show()

# Ищем минимум
k_min = min(MSE)

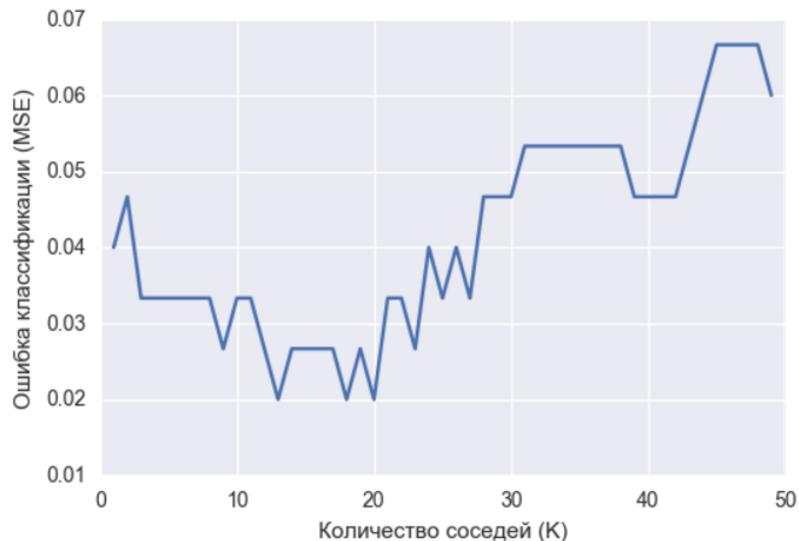
# Пробуем найти прочие минимумы (если их несколько)
all_k_min = []
for i in range(len(MSE)):
    if MSE[i] <= k_min:
        all_k_min.append(k_list[i])

# печатаем все K, оптимальные для модели
print('Оптимальные значения K: ', all_k_min)

```

Рисунок 3.10 – Реализация процедуры выбора оптимального параметра на основе cross validation error

Вывод для данного кода представлен на рис. 3.11.



Оптимальные значения K: [13, 18, 20]

Рисунок 3.11 – Визуализация выбора оптимального параметра на основе cross validation error

### 3.2 Важные замечания

1. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем с целью недопустимости выбора одинаковых вариантов.
2. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.
3. При выборе набора данных следует отдавать предпочтение тем наборам, которые имеют следующие характеристики: содержат не более 5 признаков на объект; все признаки – числовые; желательно отсутствие пропусков в данных.

### 3.3 Индивидуальное задание

1. Студент самостоятельно выбирает набор данных на ресурсах [3, 4] для построения классификатора с использованием метода ближайших соседей и согласует свой выбор с преподавателем.
2. Выполните построение модели классификации на основе метода ближайших соседей. В ходе решения задачи необходимо решить следующие подзадачи:
  - 2.1 Построение классификатора с заданием K (количества ближайших соседей) пользователем;
  - 2.2 Вычисление оценки hold-out для различных значений K, а также для различных долей обучающей и тестирующей подвыборок;
  - 2.3 Вычисление оценки cross validation для различных значений K, а также для различных значений fold (количества подмножеств при кросс-валидации).
  - 2.4 Вычислите оптимальные значения K. Обоснуйте свой выбор. Продемонстрируйте использование полученного классификатора.

#### **4. Содержание отчета и его форма**

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.
4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

#### **5. Контрольные вопросы**

1. Поясните особенности основных методов метрической классификации: метод ближайшего соседа, метод k ближайших соседей.
2. Поясните основные принципы и этапы реализации метода kNN.
3. Поясните принцип выбора количества соседних объектов, по которым определяется принадлежность целевого объекта к результирующему классу.
4. В чем заключается метод парзеновского окна?
5. Поясните принцип метода потенциальных функций.
6. Назовите, какие параметры оптимизируют в методах kNN?

#### **6. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-5].

## **ЛАБОРАТОРНАЯ РАБОТА 4. ЛОГИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ**

### **1. Цели и задачи**

Цель лабораторной работы: изучение принципов построения информационных систем с использованием логических методов классификации.

Основные задачи:

- освоение технологии внедрения алгоритмов на основе решающих списков в приложения;
- освоение технологии внедрения алгоритмов на основе решающих деревьев в приложения;
- изучение параметров логической классификации;
- освоение модификаций логических методов классификации.

### **2. Теоретическое обоснование**

Перед выполнением лабораторной работы необходимо ознакомиться с теорией построения логических классификаторов, используя следующие источники: [1-5].

### **3. Методика и порядок выполнения работы**

#### **3.1 Учебная задача**

В рамках учебной задачи необходимо произвести построение классификатора на основе логического дерева. В качестве набора данных используется набор данных об ирисах Фишера.

1. Подключим библиотеки, которые потребуются для загрузки и первичного анализа данных (рис. 4.1).

```

import numpy as np
import pandas as pd

%matplotlib inline

import seaborn as sns
from matplotlib import pyplot as plt

data_source = 'iris.data'
d = pd.read_table(data_source, delimiter=',',
                   header=None,
                   names=['sepal_length','sepal_width',
                          'petal_length','petal_width','answer'])
dX = d.ix[ : , 0:4 ]
dy = d['answer']
print(dX.head())
print(dy.head())

```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
0	Iris-setosa			
1	Iris-setosa			

Рисунок 4.1 – Использование pandas для загрузки набора данных

2. Для построения дерева классификации воспользуемся специальным классом `sklearn.tree.DecisionTreeClassifier`. Оценим точность модели методом hold-out (рис. 4.2). Следует обратить внимание, что если в методе ближайших соседей производилась оптимизация по одному параметру  $K$  – количеству ближайших соседей, то при создании модели `DecisionTreeClassifier` необходимо указать два параметра: максимальную глубину дерева (`max_depth`) и количество признаков разделения дерева (`max_features`).

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Подмножества для hold-out
X_train, X_holdout, y_train, y_holdout = \
train_test_split(dX, dy, test_size=0.3, random_state=12)

# Обучение модели
tree = DecisionTreeClassifier(max_depth=5,
                               random_state=21,
                               max_features=2)
tree.fit(X_train, y_train)

# Получение оценки hold-out
tree_pred = tree.predict(X_holdout)
accur = accuracy_score(y_holdout, tree_pred)
print(accur)

```

0.977777777778

Рисунок 4.2 – Обучение модели классификации и оценка ее точности методом hold-out

3. Произведем оценку точности модели по методу cross validation (рис. 4.3), а также сделаем выводы об оптимальном значении параметра `max_depth`.

```

from sklearn.model_selection import cross_val_score

# Значения параметра max_depth
d_list = list(range(1,20))
# Пустой список для хранения значений точности
cv_scores = []
# В цикле проходим все значения K
for d in d_list:
    tree = DecisionTreeClassifier(max_depth=d,
                                   random_state=21,
                                   max_features=2)
    scores = cross_val_score(tree, dX, dy, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())
|
# Вычисляем ошибку (misclassification error)
MSE = [1-x for x in cv_scores]

# Строим график
plt.plot(d_list, MSE)
plt.xlabel('Макс. глубина дерева (max_depth)');
plt.ylabel('Ошибка классификации (MSE)')
plt.show()

# Ищем минимум
d_min = min(MSE)

```

```

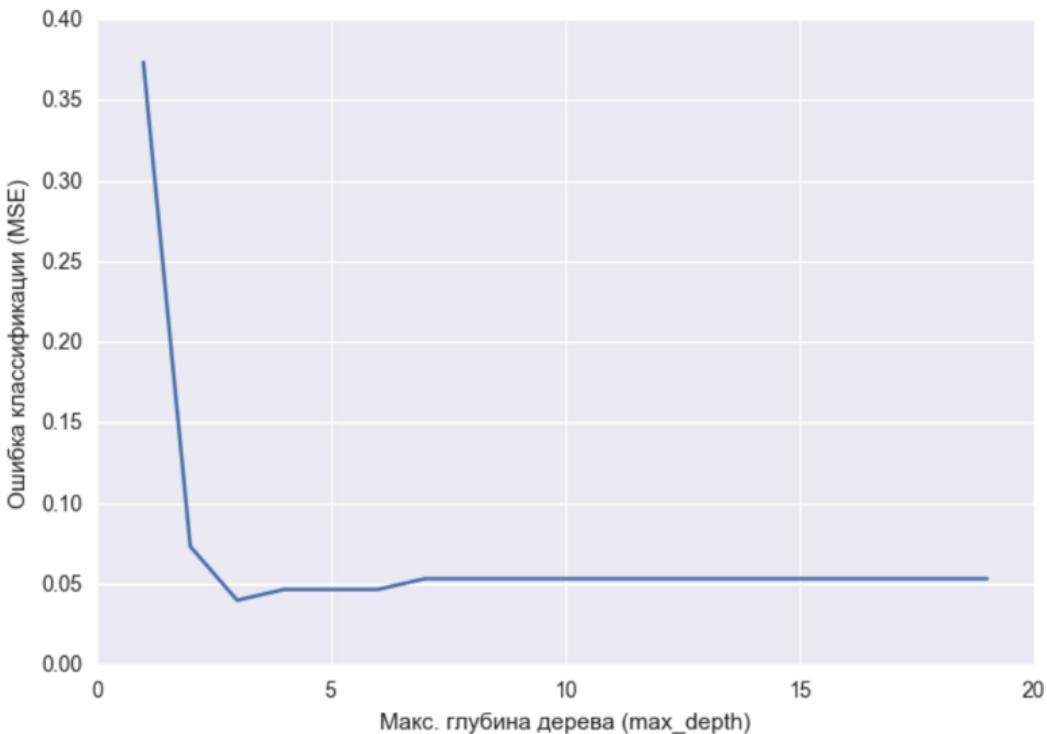
# Пробуем найти прочие минимумы (если их несколько)
all_d_min = []
for i in range(len(MSE)):
    if MSE[i] <= d_min:
        all_d_min.append(d_list[i])

# печатаем все K, оптимальные для модели
print('Оптимальные значения max_depth: ', all_d_min)

```

Рисунок 4.3 – Оценка точности модели методом cross validation и нахождение оптимального значения параметра max\_depth

В результате работы данного кода будет получен вывод (рис. 4.4).



Оптимальные значения max\_depth: [3]

Рисунок 4.4 – Вывод зависимости значения MSE от параметра max\_depth

4. Оптимальное значение параметра max\_depth модели получено, но в модели присутствует еще один параметр max\_features, который был установлен в значение 2 (не изменялся и не оптимизировался). Для проведения cross validation по всем параметрам воспользуемся классом GridSearchCV пакета sklearn.model\_selection (рис. 4.5).

```

from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn import tree

dtc = DecisionTreeClassifier(max_depth=10, random_state=21, max_features=2)

tree_params = { 'max_depth': range(1,20), 'max_features': range(1,4) }
tree_grid = GridSearchCV(dtc, tree_params, cv=10, verbose=True, n_jobs=-1)
tree_grid.fit(dx, dy)

print('\n')
print('Лучшее сочетание параметров: ', tree_grid.best_params_)
print('Лучшие баллы cross validation: ', tree_grid.best_score_)

# Генерируем графическое представление дерева
tree.export_graphviz(tree_grid.best_estimator_,
                     feature_names=dx.columns,
                     class_names=dy.unique(),
                     out_file='iris_tree.dot',
                     filled=True, rounded=True)

```

Fitting 10 folds for each of 57 candidates, totalling 570 fits

Лучшее сочетание параметров: {'max\_features': 2, 'max\_depth': 3}  
 Лучшие баллы cross validation: 0.96

[Parallel(n\_jobs=-1)]: Done 570 out of 570 | elapsed: 3.9s finished

Рисунок 4.5 – Нахождение оптимальных параметров модели логической классификации

Поясните вывод данного фрагмента. Поясните значение таких величин как fold, candidate, fit. Какие значения принимают данные величины в данном коде и почему?

Следует обратить внимание, что в результате оценки оптимальных параметров, фактически, было построено оптимальное дерево классификации. Доступ к данному дереву производится через поле `best_estimator_` класса `GridSearchCV`. В коде (рис. 4.5) производится экспорт полученного дерева в формат `.dot`. Полученный формат можно преобразовать в `.png` через сервис сайта <http://webgraphviz.com>. Полученное дерево представлено на рис. 4.6.

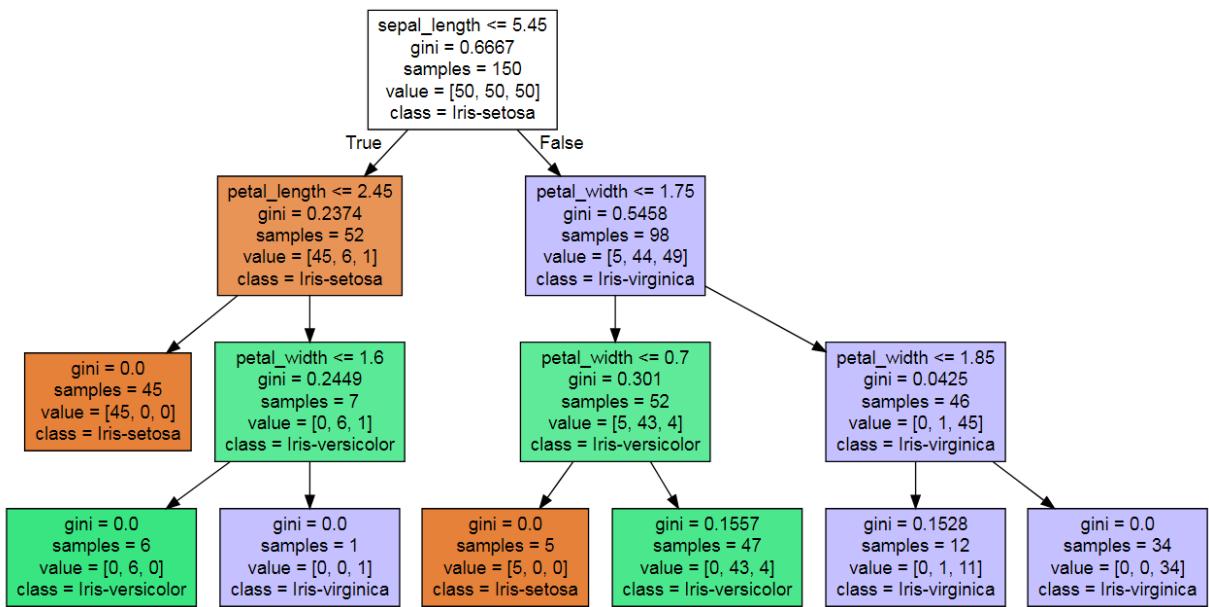


Рисунок 4.6 – Нахождение оптимальных параметров модели логической классификации

Поясните значения переменных в узлах полученного дерева: gini, samples, value.

5. Оптимальные параметры определены, можно обучить модель и использовать ее для классификации (рис. 4.7).

```

# Построим области решения для оптимального дерева
# max_features = 2, max_depth = 3

dtc = DecisionTreeClassifier(max_depth=3,
                             random_state=21,
                             max_features=2)
dtc.fit(dx, dy)
res = dtc.predict([[5.1, 3.5, 1.4, 0.2]])
print(res)

['Iris-setosa']
  
```

Рисунок 4.7 – Использование модели логической классификации

6. В заключении построим еще одну визуализацию процесса логической классификации – покажем решающие границы модели классификации (рис. 4.8).

```

plot_markers = ['r*', 'g^', 'bo']
answers = dy.unique()

# Создаем подграфики для каждой пары признаков
f, places = plt.subplots(4, 4, figsize=(16,16))

fmin = dX.min()-0.5
fmax = dX.max()+0.5
plot_step = 0.02

# Обходим все subplot
for i in range(0,4):
    for j in range(0,4):

        # Строим решающие границы
        if(i != j):
            xx, yy = np.meshgrid(np.arange(fmin[i], fmax[i], plot_step),
                                  np.arange(fmin[j], fmax[j], plot_step))
            model = DecisionTreeClassifier(max_depth=3, random_state=21, max_features=2)
            model.fit(dX.ix[:, [i,j]], dy)
            p = model.predict(np.c_[xx.ravel(), yy.ravel()])
            p = p.reshape(xx.shape)
            p[p==answers[0]] = 0
            p[p==answers[1]] = 1
            p[p==answers[2]] = 2
            places[i,j].contourf(xx, yy, p, cmap='Pastell1')

        # Обход всех классов
        for id_answer in range(len(answers)):
            idx = np.where(dy == answers[id_answer])
            if i==j:
                places[i, j].hist(dX.iloc[idx].ix[:,i],
                                    color=plot_markers[id_answer][0],
                                    histtype = 'step')
            else:
                places[i, j].plot(dX.iloc[idx].ix[:,i], dX.iloc[idx].ix[:,j],
                                   plot_markers[id_answer],
                                   label=answers[id_answer], markersize=6)

            if j==0:
                places[i, j].set_ylabel(dX.columns[j])

            if i==3:
                places[i, j].set_xlabel(dX.columns[i])

```

Рисунок 4.8 – Построение решающих границ

Вывод данного кода представлен на рис. 4.9.

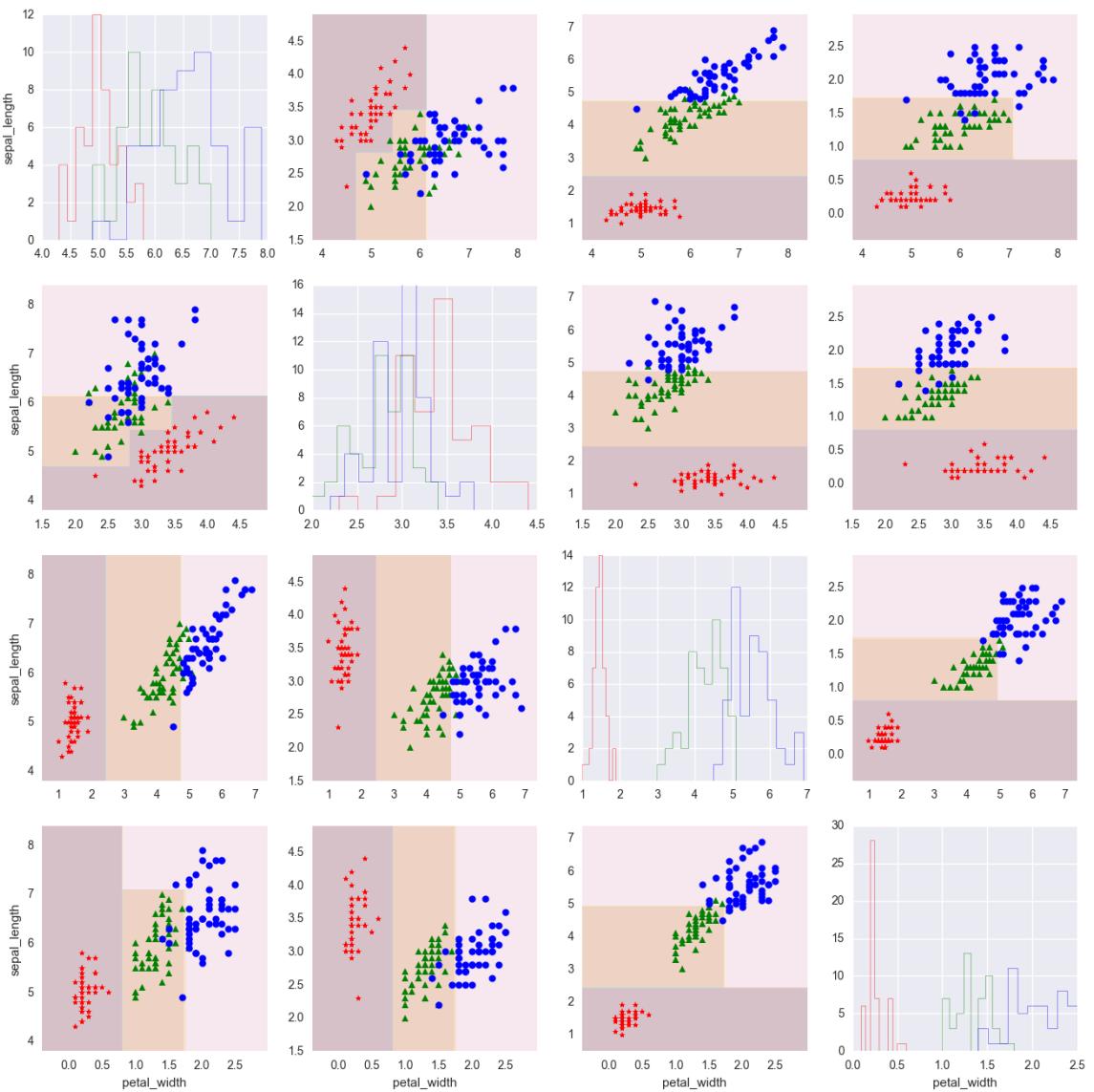


Рисунок 4.9 – Решающие границы логической модели классификации

7. Выполните индивидуальное задание.

### 3.2 Важные замечания

1. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем с целью недопустимости выбора одинаковых вариантов.
2. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

### 3.3 Индивидуальное задание

1. Студент самостоятельно выбирает набор данных на ресурсах [3, 4] для построения классификатора с использованием метода логической классификации и согласует свой выбор с преподавателем.

2. Выполните построение модели классификации на основе дерева классификации. В ходе решения задачи необходимо решить следующие подзадачи:

2.1 Построение логического классификатора с заданием `max_depth` (максимальной глубины) и `max_features` (максимального количества признаков) пользователем (установить любые); визуализация дерева решений для выбранных исследователем параметров (в формате .png)

2.2 Вычисление оценки cross validation (MSE) для различных значений `max_depth` (построить график зависимости);

2.3 Вычисление оценки cross validation (MSE) для различных значений `max_features` (построить график зависимости);

2.4 Вычислите оптимальные значения `max_depth` и `max_features`. Обоснуйте свой выбор. Продемонстрируйте использование полученного классификатора.

2.5 Выведите дерево в формате .png;

2.6 Выведите решающие границы полученной модели.

### 4. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.
4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

## **5. Контрольные вопросы**

1. Поясните принцип построения дерева решений.
2. Укажите статистическое определение информативности.
3. Поясните энтропийное определение информативности.
4. Что такое многоклассовая информативность? Для чего она применяется?
5. Поясните назначение и алгоритм бинаризации количественных признаков.
6. Поясните порядок поиска закономерностей в форме конъюнкций.

## **6. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1–5].

## ЛАБОРАТОРНАЯ РАБОТА 5. РАЗРАБОТКА ЕДИНОГО ПОДХОДА К ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ

### 1. Цели и задачи

Цель лабораторной работы: изучение теоретических принципов и инструментальных средств для построения пайплайна для предварительной обработки данных.

Основные задачи:

- предварительная обработка данных;
- изучение библиотек для предварительной обработки данных;
- масштабирование признаков;
- представление категориальных данных;
- построение пайплайна для предварительной обработки данных.

### 2. Теоретическое обоснование

В предыдущих работах уже были представлены алгоритмы, позволяющие представить последовательность решения задач методами машинного обучения. Но для различных моделей очень часто приходится повторять одни и те же действия. Это делает возможным выработку некоторого унифицированного подхода к последовательности действий, которые реализует исследователь. В данной работе рассмотрим унифицированную последовательность действий, которую обычно выполняет исследователь по предварительной обработке данных.

Для решения задач машинного обучения часто приходится повторять различные блоки кода, которые являются единообразными для разных задач, принадлежащих одному классу (регрессия, классификация, кластеризация и т.д.). Данное обстоятельство приводит к повторяющемуся шаблонному коду. Такой код называется boilerplate-код или просто boilerplate. С другой стороны, единообразная последовательность действий, которую выполняет

разработчик при решении задач машинного обучения часто называется пайплайном (machine learning pipeline).

Рассмотрим простейший пайплан для решения задачи регрессии. Для решения задачи регрессии необходимо реализовать (в общем случае) следующие стадии:

1. Загрузка набора данных.
2. Заполнение пропусков данных в соответствии с выбранной стратегией.
3. Масштабирование признаков.
4. Обработка категориальных признаков.
5. Разделение на тестовую и тренировочную выборку.
6. Обучение модели.
7. Интерпретация и визуализация результатов.

Заполнение пропусков в данных

```
missingvalues = SimpleImputer(missing_values = np.nan, strategy = 'mean',
verbose = 0)
```

```
missingvalues = missingvalues.fit(X[:, 1:3])
```

```
X[:, 1:3]=missingvalues.transform(X[:, 1:3])
```

Масштабирование признаков (standartisation и normalization)

```
from sklearn.preprocessing import StandardScaler
```

Разделение на тестовую и обучающую выборки:

```
from sklearn.model_selection import train_test_split
from sklearn.cross_validation import train_test_split
```

Перед выполнением лабораторной работы необходимо ознакомиться с базовыми принципами языка Python, используя следующие источники: [1-5].

### 3. Методика и порядок выполнения работы

Для тестирования универсального пайплайна будет использоваться модель линейоной регрессии (LinearRegression из библиотеки sklearn).

#### 3.1 Учебная задача

**Устовие.** Построить пайплайн, реализующий первичную обработку данных.

**Решение.** Для решения задачи необходимо написать скрипт на языке Python (рисунок 5.1).

##### 1. Подключение библиотек

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

##### 2. Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
dataset = pd.read_csv('Data.csv')
dataset.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 3].values
print ("Матрица признаков"); print(x)
print ("Зависимая переменная"); print(y)
```

```
Матрица признаков
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 nan]
 ['France' 35.0 58000.0]
 ['Spain' nan 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
Зависимая переменная
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

### 3. Обработка пропущенных значений

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
imputer = imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
print(X)

[['France' 44.0 72000.0]
['Spain' 27.0 48000.0]
['Germany' 30.0 54000.0]
['Spain' 38.0 61000.0]
['Germany' 40.0 63777.77777777778]
['France' 35.0 58000.0]
['Spain' 38.77777777777778 52000.0]
['France' 48.0 79000.0]
['Germany' 50.0 83000.0]
['France' 37.0 67000.0]]
```

### 4. Обработка категориальных данных

#### 4.1 Замена категории кодом (LabelEncoder)

```
from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
print("Зависимая переменная до обработки")
print(y)
y = labelencoder_y.fit_transform(y)
print("Зависимая переменная после обработки")
print(y)

Зависимая переменная до обработки
[ 'No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
Зависимая переменная после обработки
[0 1 0 0 1 1 0 1 0 1]
```

#### 4.2 Применение OneHotEncoder

```
from sklearn.preprocessing import OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 0] = labelencoder_X.fit_transform(X[:, 0])
onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()
print("Перекодировка категориального признака")
print(X)

Перекодировка категориального признака
[[ 1.0000000e+00  0.0000000e+00  0.0000000e+00  4.4000000e+01
   7.2000000e+04]
 [ 0.0000000e+00  0.0000000e+00  1.0000000e+00  2.7000000e+01
   4.8000000e+04]
 [ 0.0000000e+00  1.0000000e+00  0.0000000e+00  3.0000000e+01
   5.4000000e+04]
 [ 0.0000000e+00  0.0000000e+00  1.0000000e+00  3.8000000e+01
   6.1000000e+04]]
```

Рисунок 5.1 – Код Python, отражающий общий пайплайн для предварительной обработки данных

На рисунке 5.1 показан устаревший метод работы с трансформатором OneHotEncoder. На рисунке 5.2 показан метод, который используется в настоящее время. Данный подход предполагает применение класса ColumnTransformer, который осуществляет преобразование значений столбцов с использованием переданных ему классов-трансформеров. В

качестве классов-трансформеров можно использовать классы FeatureHasher, MinMaxScaler, SimpleImputer, OneHotEncoder, Normalizer и прочие.

```

1  # создаем копию "грязного" объекта: спропусками и некодированными категориями
2  X_dirty = X.copy()
3  X_dirty
executed in 5ms, finished 17:12:54 2022-09-01

array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, nan],
       ['France', 35.0, 58000.0],
       ['Spain', nan, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)

1  # Современный метод трансформации признаков
2  from sklearn.preprocessing import OneHotEncoder
3  from sklearn.impute import SimpleImputer
4  from sklearn.compose import ColumnTransformer
5
6  # создаем список трансформеров
7  transformers = [
8      ('onehot', OneHotEncoder(), [0]),
9      ('imp', SimpleImputer(), [1, 2])
10 ]
11
12 # Создаем объект ColumnTransformer и передаем ему список трансформеров
13 ct = ColumnTransformer(transformers)
14
15 # Выполняем трансформацию признаков
16 X_transformed = ct.fit_transform(X_dirty)
17 print(X_transformed.shape)
18 X_transformed
executed in 10ms, finished 17:21:47 2022-09-01

Out[35]: array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 4.4000000e+01,
   7.2000000e+04],
   [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 2.7000000e+01,
   4.8000000e+04],
   [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 3.0000000e+01,
   5.4000000e+04],
   [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 3.8000000e+01,
   6.1000000e+04],
   [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 4.0000000e+01,
   6.3777778e+04],
   [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 3.5000000e+01,
   5.8000000e+04],
   [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 3.8777778e+01,
   5.2000000e+04],
   [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 4.8000000e+01,
   7.9000000e+04],
   [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 5.0000000e+01,
   8.3000000e+04],
   [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 3.7000000e+01,
   6.7000000e+04]])

```

Рисунок 5.2 – Код Python, отражающий применение ColumnTransformer

Следует обратить внимание, что большинство методов трансформации приводит к получению результата в виде массива. Для конвертации массива в DataFrame можно воспользоваться следующим подходом:

```

1  # Можно преобразовать полученный многомерный массив обратно в Dataframe
2  X_data = pd.DataFrame(
3      X_transformed,
4      columns=['C1', 'C2', 'C3', 'Age', 'Salary'])
5  X_data

```

executed in 12ms, finished 17:34:48 2022-09-01

	C1	C2	C3	Age	Salary
0	1.0	0.0	0.0	44.000000	72000.000000
1	0.0	0.0	1.0	27.000000	48000.000000
2	0.0	1.0	0.0	30.000000	54000.000000
3	0.0	0.0	1.0	38.000000	61000.000000
4	0.0	1.0	0.0	40.000000	63777.777778
5	1.0	0.0	0.0	35.000000	58000.000000
6	0.0	0.0	1.0	38.777778	52000.000000
7	1.0	0.0	0.0	48.000000	79000.000000
8	0.0	1.0	0.0	50.000000	83000.000000
9	1.0	0.0	0.0	37.000000	67000.000000

Рисунок 5.3 – Преобразование массива в DataFrame

### 3.2 Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.
2. Реализуйте первичную обработку данных загруженного набора. Выполните полный спектр операций для загруженного набора данных: загрузка, визуализация, обработка пропущенных значений, обработка категориальных данных и разделение выборки на тестовую и тренировочную.

## 7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

## **8. Контрольные вопросы**

1. Какая библиотека python предназначена для управления наборами данных: numpy, pandas, sklearn, opencv, matplotlib?

2. Какая стратегия является нежелательной при обработке пропусков в данных?

а) замена пропущенных значений в столбце медианным значением по данному столбцу;

б) удаление строк, содержащих пропуски в данных;

в) замена пропущенных значений в столбце средним арифметическим значением по данному столбцу;

г) замена пропущенных значений в столбце наиболее часто встречающимся значением по данному столбцу;

3. Обоснуйте ответ на следующую проблему предварительной обработки данных: имеется независимая категориальная переменная  $y$ , которая представляет собой категориальный признак, определенный на домене {C#, Java, Python, R}. Нужно ли применять к данному целевому признаку OneHotEncoder?

4. Поясните принцип разбиения набора данных на обучающую и тестовую выборку. Какое соотношение «тестовая:обучающая» наиболее оптимально: 20:80, 50:50, 25:75, 5:95, 40:30?

5. Какой код лучше использовать при загрузке данных из csv-файла?

а) `dataset = read_csv("data.csv")`

- б) dataset = import("data.csv")
- в) dataset = read.csv("data.csv")
- г) dataset = import.csv("data.csv")
- д) dataset = read\_xls("data.csv")

## **9. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

## ЛАБОРАТОРНАЯ РАБОТА 6. ПОСТРОЕНИЕ ПАЙПЛАЙНА ОДНОМЕРНОЙ РЕГРЕССИИ

### 1. Цели и задачи

Цель лабораторной работы: разработка единого пайплайна для решения задачи регрессии.

Основные задачи:

- реализовать конвейер для выполнения всех стадий обработки данных при решении задачи одномерной регрессии;
- получение теоретических представлений о задаче регрессии;
- получение навыков использования пайплайна при решении задачи машинного обучения;
- получение навыков рефакторинга кода в задачах машинного обучения.

### 2. Теоретическое обоснование

Для решения задачи одномерной регрессии необходимо использовать универсальный пайpline предварительной обработки данных. К имеющемуся шаблонному коду необходимо добавить код для обучения модели, интерпретации и визуализации результатов.

Линейная регрессия – метод восстановления зависимости между двумя переменными. Пусть задана модель регрессии – параметрическое семейство функций  $g(x, \alpha)$ , где  $\alpha \in \mathbb{R}^p$  – вектор параметров модели. Определим функционал качества аппроксимации целевой зависимости на выборке  $X^\ell$  как сумму квадратов ошибок:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i)^2. \quad (6.1)$$

Обучение по методу наименьших квадратов (МНК) состоит в том, чтобы найти вектор параметров  $\alpha^*$ , при котором достигается минимум среднего квадрата ошибки на заданной обучающей выборке  $X^\ell$ :

$$\alpha^* = \arg \min_{\alpha \in \mathbb{R}^p} Q(\alpha, X^\ell). \quad (6.2)$$

Стандартный способ решения этой оптимизационной задачи – воспользоваться необходимым условием минимума. Если функция  $g(x, \alpha)$  достаточное число раз дифференцируема по  $\alpha$ , то в точке минимума выполняется система  $p$  уравнений относительно  $p$  неизвестных:

$$\frac{\partial Q}{\partial \alpha}(\alpha, X^\ell) = 2 \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i) \frac{\partial g}{\partial \alpha}(x_i, \alpha) = 0. \quad (6.3)$$

С использованием библиотек машинного обучения формулы (6.1) – (6.2) можно реализовать автоматически, но следует понимать, что конкретно реализует каждый метод.

### 3. Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

#### 3.1 Учебная задача

**Установка.** Построить пайплайн, реализующий решение задачи линейной одномерной регрессии.

**Решение.** Для решения задачи необходимо написать скрипт на языке Python (рисунок 6.1).

## 1. Подключение библиотек

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

## 2. Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
dataset = pd.read_csv('./Salary_Data.csv')
dataset.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
print ("Матрица признаков"); print(X[:5])
print ("Зависимая переменная"); print(y[:5])
```

Матрица признаков

```
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]]
```

Зависимая переменная

```
[ 39343.  46205.  37731.  43525.  39891.]
```

Lesson 7. Построение пайплайна регрессии Last Checkpoint: 5 minutes ago (autosaved)

View Insert Cell Kernel Widgets Help Trusted

Code

## 5. Разделение выборки на тестовую и тренировочную

```
# from sklearn.cross_validation import train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/4, random_state = 0)
```

## 6. Обучение линейной модели регрессии

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

## 7. Предсказание, обработка и визуализация результатов

```
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[ 41056.25705466 123597.70938378 65443.50433372 63567.56223533
 116093.94099022 108590.17259667 117031.91203942 64505.53328452]
```

```

plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

```



```

plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

```



Рисунок 6.1 – Код Python, отражающий общий пайплайн для задачи одномерной регрессии

### 3.2 Индивидуальное задание

- Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.
- Постройте модель регрессии на основе универсального пайплайна.

## 7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
  2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
  3. Ответы на контрольные вопросы.
  4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.
- Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

## **8. Контрольные вопросы**

1. Почему при реализации линейной модели регрессии нет необходимости выполнять масштабирование признаков?
2. Почему при реализации модели линейной регрессии в качестве функции потерь используется квадратичное отклонение, а не модуль отклонения?
3. Что именно реализовано в методе `fit(X, y)` класса `LinearRegression`?
4. Что такое р-значение? Как р-значение используется при оптимизации моделей регрессии?
5. Поясните назначение метода `predict` класса `LinearRegression`.
6. Поясните назначение метода `plot` и `scatter` класса `pyplot`.
7. По какой подвыборке необходимо оценивать точность модели машинного обучения: тестовой или тренировочной?

## **9. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

# ЛАБОРАТОРНАЯ РАБОТА 7. ИСПОЛЬЗОВАНИЕ РАЗРАБОТАННОГО ПАЙПЛАЙНА ДЛЯ МНОГОМЕРНОЙ РЕГРЕССИИ

## 1. Цели и задачи

Цель лабораторной работы: научиться применять разработанный пайплайн для тиражирования кода с целью решения широкого круга задач машинного обучения.

Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- получение навыков определения ключевых признаков в задачах машинного обучения;
- получение навыков реализации ключевых стратегий оптимизации моделей регрессии.

## 2. Теоретическое обоснование

При решении задач многомерной регрессии исследователю необходимо решить ряд подзадач:

1. Определить коррелированность признаков.
2. Определить, какие признаки существенны при построении модели регрессии.

Проблема определения значимых признаков связана с проблемой снижения размерности.

Важное значение при многомерной регрессии приобретает обработка категориальных признаков. Часто необходимо заменить категориальный признак на набор фиктивных переменных.

К проблеме выбора значимых переменных существует несколько стратегий (фактически это методы построения модели многомерной регрессии):

1. All-in. В данном подходе производится включение всех признаков в модель.
2. Backward Elimination. В подходе предполагается обучение модели с учетом всех признаков и удаление признаков по одному на основе их значимости до достижения ситуации, когда останутся только значимые признаки.
3. Forward Selection. Подход предполагает начальное тестирование модели с одним признаком (тестируется каждый признак). Затем добавляются по одному наиболее значимые признаки.
4. Bidirectional Elimination. Подход совмещает стратегии 2 и 3.
5. Score Comparison.

### **3. Методика и порядок выполнения работы**

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

#### **3.1 Учебная задача**

**Задание.** На основе разработанного пайплайна для линейной одномерной регрессии разработать многомерную модель регрессии.

**Решение.** Для разработки модели необходимо реализовать следующий код:

##### **1.1 Подключение библиотек**

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
executed in 11.9s, finished 21:21:46 2019-10-11

```

## 1.2 Загрузка данных и разделение на матрицу признаков и зависимую переменную

```

1 dataset = pd.read_csv('50_Startups.csv')
2 dataset.head()
executed in 14ms, finished 21:49:36 2019-10-11

```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

```

1 X = dataset.iloc[:, :-1].values
2 y = dataset.iloc[:, 4].values
3 print ("Матрица признаков"); print(X[:5])
4 print ("Зависимая переменная"); print(y[:5])
executed in 6ms, finished 21:50:21 2019-10-11

```

Матрица признаков

```

[[165349.2 136897.8 471784.1 'New York']
 [162597.7 151377.59 443898.53 'California']
 [153441.51 101145.55 407934.54 'Florida']
 [144372.41 118671.85 383199.62 'New York']
 [142107.34 91391.77 366168.42 'Florida']]

```

## 1.3 Обработка пропущенных значений

```

1 # from sklearn.preprocessing import Imputer
2 # imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
3 # imputer = imputer.fit(X[:, 1:3])
4 # X[:, 1:3] = imputer.transform(X[:, 1:3])
5 # print(X)

```

## 1.4 Обработка категориальных данных

### 1.4.1 Замена категории кодом (LabelEncoder)

```

1 # from sklearn.preprocessing import LabelEncoder
2 # labelencoder_y = LabelEncoder()
3 # print("Зависимая переменная до обработки")
4 # print(y)
5 # y = labelencoder_y.fit_transform(y)
6 # print("Зависимая переменная после обработки")
7 # print(y)

```

### 1.4.2 Применение OneHotEncoder

```

1 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
2 labelencoder = LabelEncoder()
3 X[:, 3] = labelencoder.fit_transform(X[:, 3])
4 onehotencoder = OneHotEncoder(categorical_features = [3])
5 X = onehotencoder.fit_transform(X).toarray()
6 print("Перекодировка категориального признака")
7 print(X[:,4,:])

```

```

executed in 8ms, finished 21:50:24 2019-10-11
Перекодировка категориального признака
[[0.000000e+00 0.000000e+00 1.000000e+00 1.6534920e+05 1.3689780e+05
 4.7178410e+05]
 [1.000000e+00 0.000000e+00 0.000000e+00 1.6259770e+05 1.5137759e+05
 4.4389853e+05]
 [0.000000e+00 1.000000e+00 0.000000e+00 1.5344151e+05 1.0114555e+05
 4.0793454e+05]
 [0.000000e+00 0.000000e+00 1.000000e+00 1.4437241e+05 1.1867185e+05
 3.8319962e+05]]

```

## 1.5 Для предотвращения мультиколлинеарности необходимо избавиться от одной из фиктивных переменных, добавленных в результате обработки категориальных признаков

```

1 X = X[:, 1:]
2 print(X[:4,:])

```

executed in 4ms, finished 21:50:50 2019-10-11

```

[[0.000000e+00 1.000000e+00 1.6534920e+05 1.3689780e+05 4.7178410e+05]
 [0.000000e+00 0.000000e+00 1.6259770e+05 1.5137759e+05 4.4389853e+05]
 [1.000000e+00 0.000000e+00 1.5344151e+05 1.0114555e+05 4.0793454e+05]
 [0.000000e+00 1.000000e+00 1.4437241e+05 1.1867185e+05 3.8319962e+05]]

```

## 1.6 Разделение выборки на тестовую и тренировочную

```

1 # from sklearn.cross_validation import train_test_split
2 from sklearn.model_selection import train_test_split
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

```

executed in 3ms, finished 21:51:04 2019-10-11

## 1.7 Обучение линейной модели регрессии

```

1 from sklearn.linear_model import LinearRegression
2 regressor = LinearRegression()
3 regressor.fit(X_train, y_train)

```

executed in 5ms, finished 21:51:10 2019-10-11

## 1.8 Обработка результатов, тюнинг модели

### 1.8.1 Предсказание

```

1 y_pred = regressor.predict(X_test)
2 print(y_pred)

```

executed in 3ms, finished 21:51:13 2019-10-11

```

[103015.20159795 132582.27760817 132447.73845176 71976.09851257
 178537.48221058 116161.24230165 67851.69209675 98791.73374686
 113969.43533013 167921.06569553]

```

Рисунок 7.1 – Код Python для построения модели многомерной регрессии

На данный момент произведено обучение модели на всем наборе признаков. Для оптимизации модели реализуем стратегию Back Elimination (рисунок 7.2).

### 1.8.2 Оптимизация модели

```

1 import statsmodels.formula.api as sm
2 X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)
3 X_opt = X[:, [0, 1, 2, 3, 4, 5]]
4 regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
5 regressor_OLS.summary()

```

executed in 22ms, finished 21:51:17 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	5.013e+04	6884.820	7.281	0.000	3.62e+04	6.4e+04
x1	198.7888	3371.007	0.059	0.953	-6595.030	6992.607
x2	-41.8870	3256.039	-0.013	0.990	-6604.003	6520.229
x3	0.8060	0.046	17.369	0.000	0.712	0.900
x4	-0.0270	0.052	-0.517	0.608	-0.132	0.078
x5	0.0270	0.017	1.574	0.123	-0.008	0.062

```

X_opt = X[:, [0, 1, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()

```

executed in 20ms, finished 21:52:56 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	5.011e+04	6647.870	7.537	0.000	3.67e+04	6.35e+04
x1	220.1585	2900.536	0.076	0.940	-5621.821	6062.138
x2	0.8060	0.046	17.606	0.000	0.714	0.898
x3	-0.0270	0.052	-0.523	0.604	-0.131	0.077
x4	0.0270	0.017	1.592	0.118	-0.007	0.061

X_opt = X[:, [0, 3, 4, 5]] regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit() regressor_OLS.summary()
executed in 19ms, finished 21:53:11 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	5.012e+04	6572.353	7.626	0.000	3.69e+04	6.34e+04
x1	0.8057	0.045	17.846	0.000	0.715	0.897
x2	-0.0268	0.051	-0.526	0.602	-0.130	0.076
x3	0.0272	0.016	1.655	0.105	-0.006	0.060

X_opt = X[:, [0, 3, 5]] regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit() regressor_OLS.summary()
executed in 20ms, finished 21:53:30 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	4.698e+04	2689.933	17.464	0.000	4.16e+04	5.24e+04
x1	0.7966	0.041	19.266	0.000	0.713	0.880

X_opt = X[:, [0, 3]] regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit() regressor_OLS.summary()
executed in 18ms, finished 21:53:49 2019-10-11

Dep. Variable:	y	R-squared:	0.947
Model:	OLS	Adj. R-squared:	0.945
Method:	Least Squares	F-statistic:	849.8
Date:	Fri, 11 Oct 2019	Prob (F-statistic):	3.50e-32
Time:	21:53:49	Log-Likelihood:	-527.44
No. Observations:	50	AIC:	1059.
Df Residuals:	48	BIC:	1063.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	4.903e+04	2537.897	19.320	0.000	4.39e+04	5.41e+04
x1	0.8543	0.029	29.151	0.000	0.795	0.913

Omnibus: 13.727	Durbin-Watson: 1.116
Prob(Omnibus): 0.001	Jarque-Bera (JB): 18.536
Skew: -0.911	Prob(JB): 9.44e-05
Kurtosis: 5.361	Cond. No. 1.65e+05

Рисунок 7.2 – Оптимизация модели многомерной регрессии

Выполните индивидуальное задание.

### 3.2 Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.
2. Постройте модель многомерной регрессии с использованием стратегии backward elimination.

### **4. Содержание отчета и его форма**

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.
4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

### **5. Контрольные вопросы**

1. Почему при реализации многомерной линейной регрессии необходимо добавить фиктивный признак с единственным значением 1.0?.
2. Что такое фиктивная переменная? Поясните причину удаления одной фиктивной переменной, возникающей при перекодировке категориального признака.
3. На основе какого критерия можно выбирать удаляемый признак в алгоритме back elimination.
4. В чем заключается алгоритм all-in regression?
5. В чем заключается алгоритм forward selection regression?

6. В чем заключается алгоритм Bidirectional Elimination?
7. Стратегия Backward Elimination предполагает удаление признаков на основе анализа р-критерия. Как реализовать удаление признаков в автоматическом режиме?

## **9. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

## ЛАБОРАТОРНАЯ РАБОТА 8. ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ

### 1. Цели и задачи

Цель лабораторной работы: научиться применять разработанный пайплайн для тиражирования кода с целью решения задачи полиномиальной регрессии.

Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- изучение поведения модели полиномиальной регрессии при изменении степени полинома;
- исследование свойств набора данных в рамках задачи полиномиальной регрессии.

### 2. Теоретическое обоснование

Линейная и параболическая модели являются частными случаями более сложной модели – полиномиальной. Построить модель регрессии – это значит найти параметры той функции, которая будет в ней фигурировать. Для линейной регрессии – два параметра: коэффициент и свободный член.

Полиномиальная регрессия может применяться в математической статистике при моделировании трендовых составляющих временных рядов. Временной ряд – это, по сути, ряд чисел, которые зависят от времени. Например, средние значения температуры воздуха по дням за прошедший год, или доход предприятия по месяцам. Порядок моделируемого полинома оценивается специальными методами, например, критерием серий. Цель построения модели полиномиальной регрессии в области временных рядов всё та же – прогнозирование.

### 3. Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

#### 3.1 Учебная задача

**Задание.** На основе разработанного пайплайна для линейной одномерной регрессии разработать полиномиальную модель регрессии.

**Решение.** Для разработки модели необходимо реализовать следующий код:

##### 1.1 Подключение библиотек

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

executed in 510ms, finished 22:35:43 2019-10-11

##### 1.2 Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
dataset = pd.read_csv('Position_Salaries.csv')
dataset.head()
```

executed in 23ms, finished 22:35:54 2019-10-11

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

##### 1.6 Обучение модели

###### 1.6.1 Обучение линейной модели

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

executed in 884ms, finished 22:38:01 2019-10-11

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

###### 1.6.2 Обучение полиномиальной модели

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 10)
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

executed in 45ms, finished 22:59:17 2019-10-11

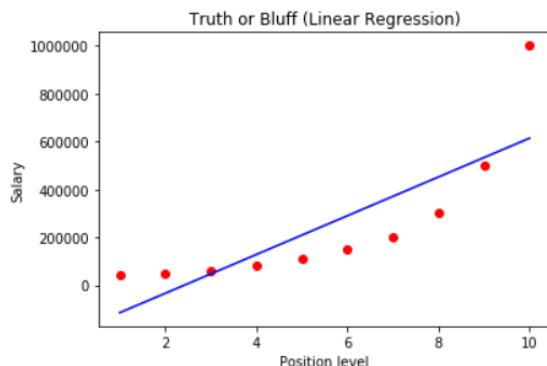
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

## 1.7 Предсказание, обработка и визуализация результатов

```
y_pred_lin = lin_reg.predict([[6.5]])
y_pred_poly = lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
print(y_pred_lin, y_pred_poly)
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

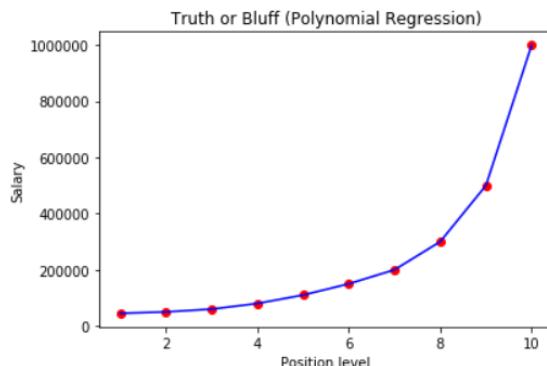
executed in 137ms, finished 22:53:36 2019-10-11

[330378.78787879] [176235.71632004]



```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

executed in 134ms, finished 22:59:21 2019-10-11



```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

executed in 135ms, finished 22:59:24 2019-10-11

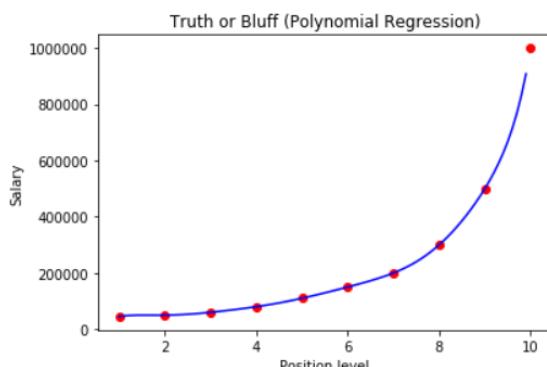


Рисунок 8.1 – Код Python для построения модели полиномиальной регрессии

Рассмотрим как изменяется модель при изменении степени аппроксимирующего полинома (рисунок 8.2).

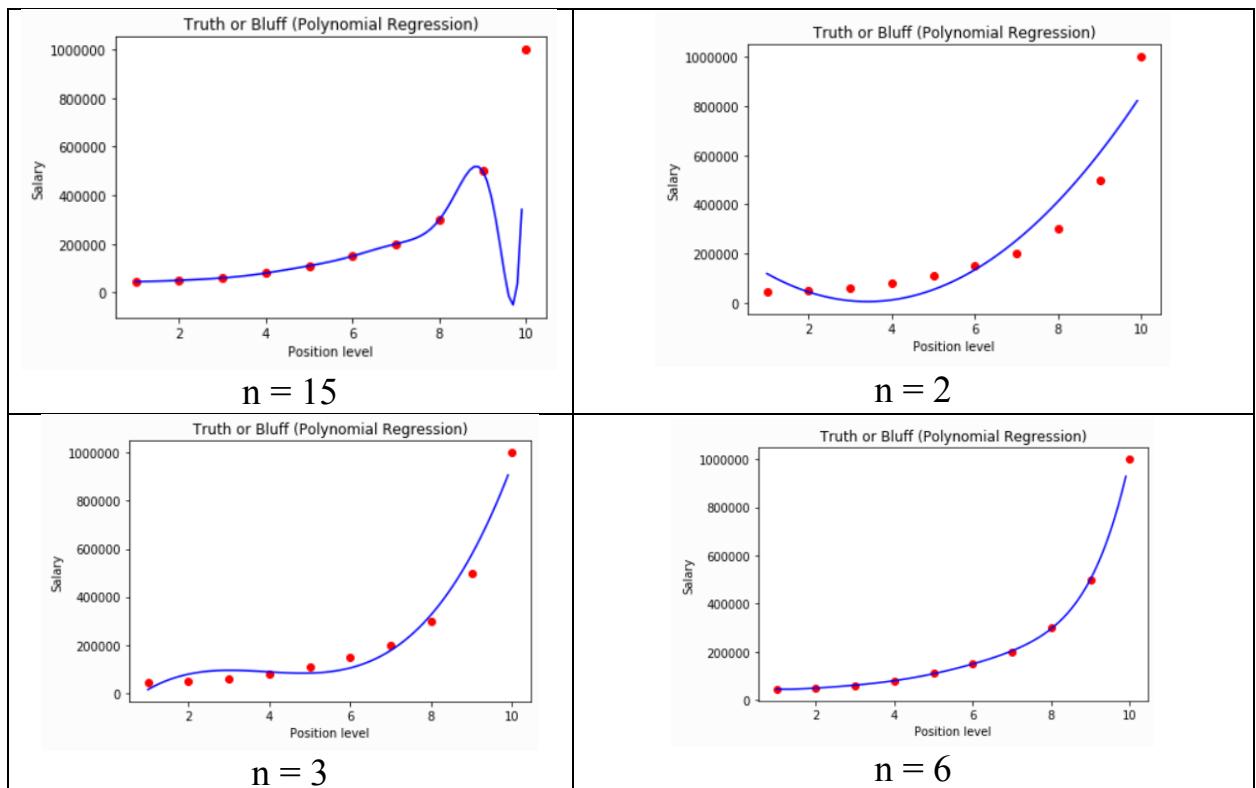


Рисунок 8.2 – Анализ модели полиномиальной регрессии

Очевидно, что при реализации полиномиальной регрессии нет необходимости в немотивированном увеличении степени аппроксимации. Выполните индивидуальное задание.

### 3.2 Индивидуальное задание

- Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.
- Постройте модель полиномиальной регрессии с использованием. Проанализируйте кривые аппроксимации при различных степенях полинома.

#### **4. Содержание отчета и его форма**

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.
4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

#### **5. Контрольные вопросы**

1. Почему при реализации многомерной линейной регрессии необходимо добавить фиктивный признак с единственным значением 1.0?
2. Что такое фиктивная переменная? Поясните причину удаления одной фиктивной переменной, возникающей при перекодировке категориального признака.
3. С использованием какого класса создается модель полиномиальной регрессии?
4. Поясните принцип преобразования признаков при построении полиномиальной регрессии.
5. Возможно ли применение технологий масштабирования признаков при реализации полиномиальной регрессии?

## **6. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

## ЛАБОРАТОРНАЯ РАБОТА 9. КЛАСТЕРИЗАЦИЯ

### 1. Цели и задачи

Цель лабораторной работы: научится производить кластерный анализ данных на основе метода К-средних.

Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- изучение принципов определения оптимального количества кластеров в методах кластерного анализа;
- изучение возможностей языка Python для реализации кластероного анализа.

### 2. Теоретическое обоснование

Кластеризация – это разбиение множества объектов на подмножества (кластеры) по заданному критерию. Каждый кластер включает максимально схожие между собой объекты. Представим переезд: нужно разложить по коробкам вещи по категориям (кластерам) – например одежда, посуда, декор, канцелярия, книги. Так удобнее перевозить и раскладывать предметы в новом жилье. Процесс сбора вещей по коробкам и будет кластеризацией. Критерии кластеризации определяет человек, а не алгоритм, – этим она отличается от классификации. Этот метод машинного обучения часто применяют в различных неструктурированных данных – например если нужно автоматически разбить коллекцию изображений на мини-группы по цветам.

Кластерный анализ применяют в разных сферах:

- в маркетинге – для сегментирования клиентов, конкурентов, исследования рынка;
- медицине – для кластеризации симптомов, заболеваний, препаратов;

- биологии – для классификации животных и растений;
- социологии – для разбиения респондентов на однородные группы;
- компьютерных наук – для группировки результатов при поиске сайтов, файлов и других объектов.

### 3. Методика и порядок выполнения работы

#### 3.1 Учебная задача

Задание. На основе предоставленного набора данных `Mall_customers.csv` реализуйте модель кластеризации методом К-средних.

Подключаем библиотеки и загружаем имеющийся набор данных:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
executed in 509ms, finished 17:57:47 2022-09-01

1 ## Загрузка данных
2 dataset = pd.read_csv('../datasets/LabWork9/Mall_Customers.csv')
3 dataset.head()
executed in 16ms, finished 18:06:44 2022-09-01

   CustomerID  Genre  Age  Annual Income (k$)  Spending Score (1-100)
0            1    Male    19                 15                  39
1            2    Male    21                 15                  81
2            3  Female    20                 16                   6
3            4  Female    23                 16                  77
4            5  Female    31                 17                  40

```

Рисунок 9.1 – Набор данных для задачи кластеризации

Исходный набор данных содержит сведения о посетителях торгового центра. В наборе присутствуют признаки, представленные в таблице 9.1.

Признак набора данных	Описание
CustomerID	Идентификатор клиента
Genre	Пол
Age	Возраст
Annual Income	Годовой доход
Spending Score	Баллы, присвеваемые клиенту специалистами по анализу данных торгового центра (от 1 до 100).

	Чем больше клиент тратит – тем больше баллов ему присваивается.
--	---

**Решение.** Для разработки модели необходимо реализовать следующий код:

```

1 X = dataset.iloc[:, [3, 4]].values
2 X
executed in 10ms, finished 18:06:58 2022-09-01
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
       [ 19,  14],
       [ 19,  99],
       [ 20,  15],
       [ 20,  77],
       [ 20,  13],
       [ 20,  79]]).

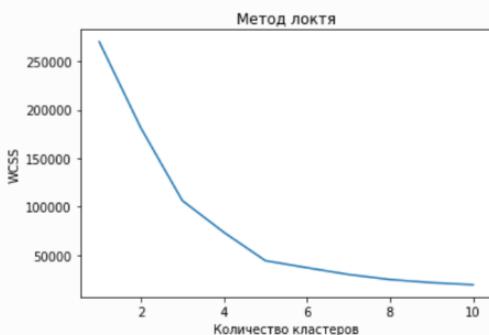
```

## 1.2 Определение оптимального количества кластеров

```

1 # Использование elbow method для поиска оптимального количества кластеров
2 from sklearn.cluster import KMeans
3 # Within Clusters Sum of Squares (WCSS)
4 wcss = []
5 for i in range(1, 11):
6     kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
7     kmeans.fit(X)
8     wcss.append(kmeans.inertia_)
9 plt.plot(range(1, 11), wcss)
10 plt.title('Метод локтя')
11 plt.xlabel('Количество кластеров')
12 plt.ylabel('WCSS')
13 plt.show()
executed in 467ms, finished 18:30:58 2022-09-01

```



## 1.3 Обучение модели кластеризации для оптимального количества кластеров

```

1 # Training the K-Means model on the dataset
2 kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
3 y_kmeans = kmeans.fit_predict(X)
executed in 33ms, finished 18:00:59 2022-09-01

```

## 1.4 Визуализация результатов

```

1 plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
2 plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
3 plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
4 plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
5 plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
6 plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
7 plt.title('Кластеры потребителей')
8 plt.xlabel('Ежегодный доход (k$)')
9 plt.ylabel('Баллы (1-100)')
10 plt.legend()
11 plt.show()
executed in 207ms, finished 18:33:24 2022-09-01

```

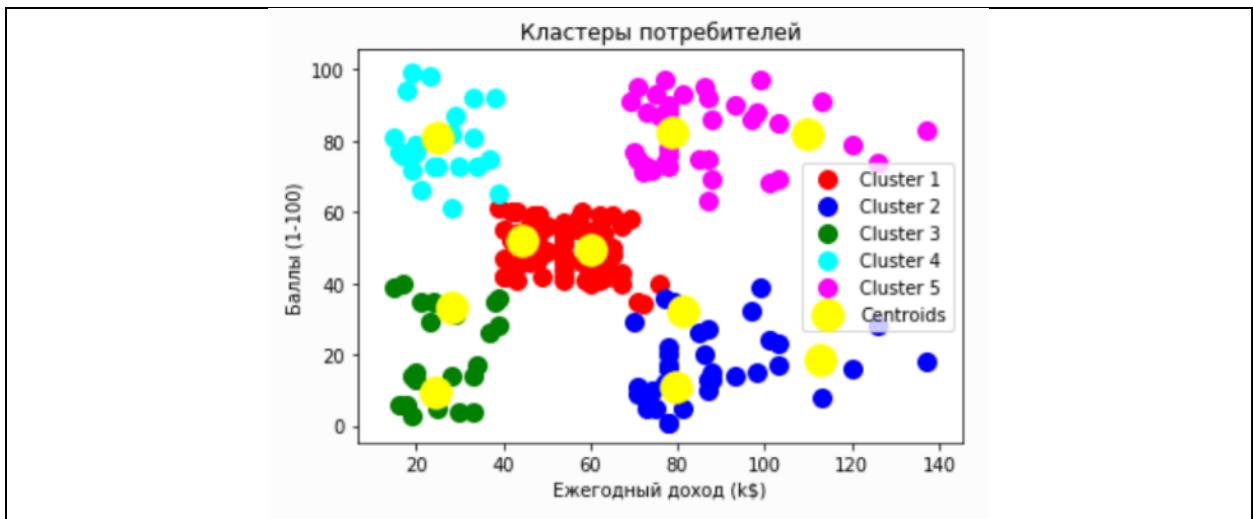


Рисунок 9.2 – Реализация метода кластеризации методом К-средних

### 3.2 Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.
2. Постройте модель кластеризации (K-Means) с использованием. Проанализируйте кривые аппроксимации при различных степенях полинома.

### 4. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.
4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

**5. Контрольные вопросы**

1. Что такое кластерный анализ?
2. Перечислите известные методы кластерного анализа.
3. Перечислите классы и функции Python, которые задействованы при реализации кластерного анализа.
4. Опишите принцип определения оптимального количества кластеров.
5. Опишите принципиальные отличия методов регрессии, кластеризации и классификации.

**6. Список литературы**

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

## ЗАКЛЮЧЕНИЕ

Учебное пособие (лабораторный практикум) по дисциплине «Методы машинного обучения» для студентов направления 09.03.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов машинного обучения, а также предлагает студентам практические рекомендации по разработке интеллектуальных систем. Основное внимание уделяется теории обучения машин (машинальное обучение, machine learning).

В пособии рассмотрены практические аспекты проектирования и разработки информационных систем для решения различных задач с использованием следующих подходов: линейных методов классификации, алгоритмов восстановления регрессии, алгоритмов логической классификации, кластерного анализа.

Многие задачи, возникающие в практических приложениях, не могут быть решены заранее известными методами или алгоритмами. Это происходит по той причине, что нам заранее не известны механизмы порождения исходных данных или же известная нам информация недостаточна для построения модели источника, генерирующего поступающие к нам данные. Машинное обучение – чрезвычайно широкая и динамически развивающаяся область исследований, использующая огромное число теоретических и практических методов.

## СПИСОК ЛИТЕРАТУРЫ

### Список основной литературы

1. Уэс, Маккинли. Python и анализ данных Электронный ресурс / Маккинли Уэс ; пер. А. А. Слинкин. - Python и анализ данных, 2022-04-19. - Саратов : Профобразование, 2017. - 482 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0046-7, экземпляров неограниченно.
2. Сузи, Р.А. Язык программирования Python Электронный ресурс : учебное пособие / Р.А. Сузи. - Язык программирования Python, 2020-07-28. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 350 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-9556-0058-2, экземпляров неограничено

### Список дополнительной литературы

3. Стенли, Липпман. Язык программирования C++ Электронный ресурс : Полное руководство / Липпман Стенли, Лажойе Жози ; пер. А. Слинкин. - Язык программирования C++, 2022-04-19. - Саратов : Профобразование, 2017. - 1104 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0136-5, экземпляров неограничено
4. <https://github.com/enikolaev/MMO> – Репозиторий с примерами кода из лабораторных работ.
5. <https://archive.ics.uci.edu/ml/index.html> – Репозиторий наборов данных для машинного обучения (Центр машинного обучения и интеллектуальных систем).
6. <https://www.kaggle.com> – Портал и система проведения соревнований по проблемам анализа данных.
7. <https://www.mockaroo.com> – Сайт для генерации наборов данных.

# **Методы машинного обучения**

УЧЕБНОЕ ПОСОБИЕ (ЛАБОРАТОРНЫЙ ПРАКТИКУМ)

Автор

Николаев Евгений Иванович

