МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
"ЛЭТИ" ИМ. В.И.УЛЬЯНОВА(ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе
по дисциплине «Основы машинного обучения»
Тема: «Изучение и предобработка данных».

Студент гр.1304 _____ Поршнев Р.А.
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2024

Задание

1. Изучение набора данных iris.csv с использованием Pandas и Seaborn:

1.1. Загрузить данные из файла как Pandas DataFrame.

1.2. Вызвав у датафрейма метод head, проверить корректность загруженных данных.

1.3. Вызвав у датафрейма метод describe, получить характеристики. Опишите полученный результат.

1.4. Видоизмените полученный датафрейм таким образом, чтобы метка классов были следующими: 0 - Iris-setosa, 1 - Iris-versicolor, 2 - Iris-virginica. Сохраните полученный датафрейм в отдельный файл формата csv.

1.5. Визуально оцените набор данных, построив изображение, содержащее графики ядерной оценки плотности каждого признака, диаграмму рассеяния и двумерную ядерную оценку плотности для каждых признаков. Наблюдения разных классов должны быть выделены отдельным цветом (рекомендуемая палитра ‘tab10’ или ‘Set1’). Опишите полученный график, что на нем изображено, какие выводы о данных можно сделать.

1.6. На одном изображении постройте гистограммы распределения для каждого признака. Затем:

1.6.1. Постройте гистограммы для разного количества столбцов: 5,10,15,20, 30. Выберите на ваш взгляд такое количество столбцов, который лучшие образом описывает форму распределения признаков.

1.6.2. Сделайте на каждой гистограмме разделение по цвету согласно классу. Проведите это в двух режимах, когда гистограммы накладываются/суммируются и когда пересекаются.

1.6.3. Постройте гистограммы, чтобы вместо столбцов изображались ступеньки.

1.6.4. Добавьте на гистограммы график ядерной оценки плотности.

2. Изучение набора данных iris.csv с использованием NumPy:

2.1. Загрузите данные из файла как массив NumPy.

2.2. Выведите первые 10 наблюдений набора данных.

2.3. Рассчитайте характеристики полученные методом describe в п. 1.3 с использованием методов NumPy.

3. Изучение набора данных вашего варианта:

3.1. Оцените и опишите набор данных вашего варианта с использованием методов в п. 1

4. Преобразование данных:

4.1. Получите из датафрейма из п. 1.4 столбец с названием классов. Используя LabelEncoder и OneHotEncoder получите различные способы кодирования меток класса. В чем различия полученных кодировок?

4.2. Для датафрейма из п. 1.4, получите все столбцы признаков (столбцы не содержащие метки классов). Преобразуйте полученные столбцы в массив NumPy.

4.3. Для массива NumPy из п. 4.2 примените StandardScaler, MinMaxScaler, MaxAbsScaler и RobustScaler. Для каждого из результатов постройте гистограммы по каждому признаку без разделения по классам. В чем различия между такими преобразованиями данных?

4.4. Согласно варианту, самостоятельно реализуйте StandardScaler или MinMaxScaler с использованием NumPy. Проверьте корректность работы на вашем наборе данных, сравните результаты между вашей реализацией и реализацией из Sklearn, а также рассчитав минимальное, максимальное, среднее значение и дисперсию, после преобразования.

4.5. Для датафрейма из п. 1.4 получите новый, который содержит только классы Iris-versicolor и Iris-virginica, признаки “sepal length (cm)” и “petal length (cm)”, и наблюдения, для которых значения признака “sepal width (cm)” лежат между квантилями 25% и 75%.

5. Понижение размерности:

5.1. Для набора данных iris.csv примените понижение размерности до 2, используя PCA и TSNE из Sklearn. Для каждого из результатов постройте диаграмму рассеяния с выделением разным цветом наблюдений разных классов.

Выполнение работы

Вариант 2

1. Изучение набора данных iris.csv с использованием Pandas и Seaborn:

1.1. Загрузить данные из файла как Pandas DataFrame

Для получения возможности использовать функционал для работы с датафреймами нужно импортировать библиотеку pandas. Для загрузки данных из файла как Pandas DataFrame нужно вызвать соответствующий метод библиотеки pandas с указанием наименования файла с данными. Реализация данного функционала представлена в Листинге 1.1.1

Листинг 1.1.1 – Загрузка данных из файла как Pandas DataFrame

```
import pandas as pd

iris_data = pd.read_csv('sample_data/iris.csv')
```

1.2. Вызвав у датафрейма метод head, проверить корректность загруженных данных

Для проверки корректности загруженных данных достаточно вызвать метод head без параметров, который выведет первые 5 записей датафрейма, и сравнить их с исходными данными, которые хранились в файле. Реализация представлена в Листинге 1.2.1, результат вывода первых пяти записей датафрейма представлен в Листинге 1.2.2.

Листинг 1.2.1 – Реализация вывода первых пяти записей датафрейма

```
iris_data.head()
```

Листинг 1.2.2 – Результат вывода первых пяти записей датафрейма

	Unnamed: 0	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	0	5.1	3.5	1.4	0.2	0
1	1	4.9	3.0	1.4	0.2	0
2	2	4.7	3.2	1.3	0.2	0
3	3	4.6	3.1	1.5	0.2	0
4	4	5.0	3.6	1.4	0.2	0

1.3. Вызвав у датафрейма метод describe, получить характеристики. Опишите полученный результат.

Данный датафрейм имеет избыточный первый столбец, который следует удалить. Реализация операции удаления и вывод первых пяти записей датафрейма представлены в Листинге 1.3.1, результат выполненных операций – в Листинге 1.3.2.

Листинг 1.3.1 – Реализация удаления первого столбца датафрейма и вывод первых пяти его записей

```
del iris_data[iris_data.columns[0]]
iris_data.head()
```

Листинг 1.3.2 – Результат удаления первого столбца датафрейма и вывод первых пяти его записей

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

После выполнения данных манипуляций с данными можно приступить к описательной статистике датасета. Для этого существует метод describe в библиотеке pandas. Реализация вызова данного метода представлена в Листинге 1.3.3, результат – в Листинге 1.3.4.

Листинг 1.3.3 – Реализация вызова метода для анализа датасета

```
iris_data.describe()
```

Листинг 1.3.4 – Результат вызова метода для анализа датасета

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

1.4. Видоизмените полученный датафрейм таким образом, чтобы метка классов были следующими: 0 - Iris-setosa, 1 - Iris-versicolor, 2 - Iris-virginica. Сохраните полученный датафрейм в отдельный файл формата csv.

Для выполнения данной задачи нужно для всего столбца 'target' сделать замену в соответствии с условием, что делается с помощью метода replace. Данный метод принимает два списка: какие значения надо заменить и на какие значения надо заменить соответственно. Также данный метод принимает параметр inplace, который по умолчанию равен False. Он отвечает за то, нужно ли произвести замену непосредственно в том датафрейме, относительно которого вызывается метод replace, или нужно вернуть видоизменённую копию. Для сохранения видоизменённого датафрейма в формате csv нужно вызвать метод to_csv с параметром пути. Для проверки замены был использован метод iloc, которому был передан список индексов, значения записей по котором требуется узнать. С помощью данного механизма можно узнать правильно ли была выполнена замена в полной мере. Реализация данных команд представлена в Листинге 1.4.1, результат выполнения – в Листинге 1.4.2.

Листинг 1.4.1 – Реализация замены данных в датафрейме, её проверка и сохранение датафрейма в файл

```
iris_data['target'].replace([0, 1, 2], ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
↪   inplace=True)
iris_data.to_csv('sample_data/iris_modify.csv', index=False)
iris_data.iloc[[0, 50, 100]]
```

Листинг 1.4.2 – Результат замены данных в датафрейме

	sepal length (cm)	**sepal width (cm)**	**petal length (cm)**	**petal width (cm)**	**target**
0	5.1	3.5	1.4	0.2	Iris-setosa
50	7.0	3.2	4.7	1.4	Iris-versicolor
100	6.3	3.3	6.0	2.5	Iris-virginica

1.5. Визуально оцените набор данных, построив изображение, содержащее графики ядерной оценки плотности каждого признака, диаграмму рассеяния и двумерную ядерную оценку плотности для каждых признаков. Наблюдения разных классов должны быть выделены отдельным цветом (рекомендуемая палитра ‘tab10’ или ‘Set1’). Опишите полученный график, что на нем изображено, какие выводы о данных можно сделать.

Для выполнения данного задания в целях удобства последующего использования (п.3) была реализована функция pair_grid и подключены библиотеки для визуалцизации данных и установлена тема для графиков seaborn. Функция pair_grid состоит из следующей последовательности команд:

• создание экземпляра PairGrid, который инициализируется полученным датасетом, переменной, по которой делятся данные и цветовая палитра;

• выше диагонали будут располагаться графики рассеяния, где по первой оси отложены значения первого признака, а по второй – другого;

• ниже диагонали будут располагаться графики двумерной оценки ядерной плотности, где по первой оси отложены значения первого признака, а по второй – другого;

• на диагоноли будут располагаться графики одномерной оценки ядерной плотности, где по первой оси отложены значения признака, а по второй – плотность

вероятности;

• создание объектов графики, которые будут применяться в легенде полученного графика;

• добавление легенды на график.

Реализация вышеописанного функционала на примере датасета iris.csv представлена в Листинге 1.5.1, полученные графики изображены на Рисунке 1.5.1.

Листинг 1.5.1 – Реализация построения графиков оценки ядерной плотности каждого признака, диаграмм рассеивания и двумерной оценки ядерной плотности

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style='darkgrid')

def pair_grid(dataset, target):
    g = sns.PairGrid(dataset, diag_sharey=False, hue=target, palette='tab10')
    g.map_upper(sns.scatterplot, s=15)
    g.map_lower(sns.kdeplot)
    g.map_diag(sns.kdeplot)
    color_labels = []
    for curr_color in g.palette:
        color_labels.append(plt.Circle((0, 0), 1, color=curr_color))
    g.add_legend({g.hue_names[i]: color_labels[i] for i in range(len(g.palette))})
    g.fig.suptitle("Scatter plots and KDE's", y=1.02)
pair_grid(dataset=iris_data, target='target')
```

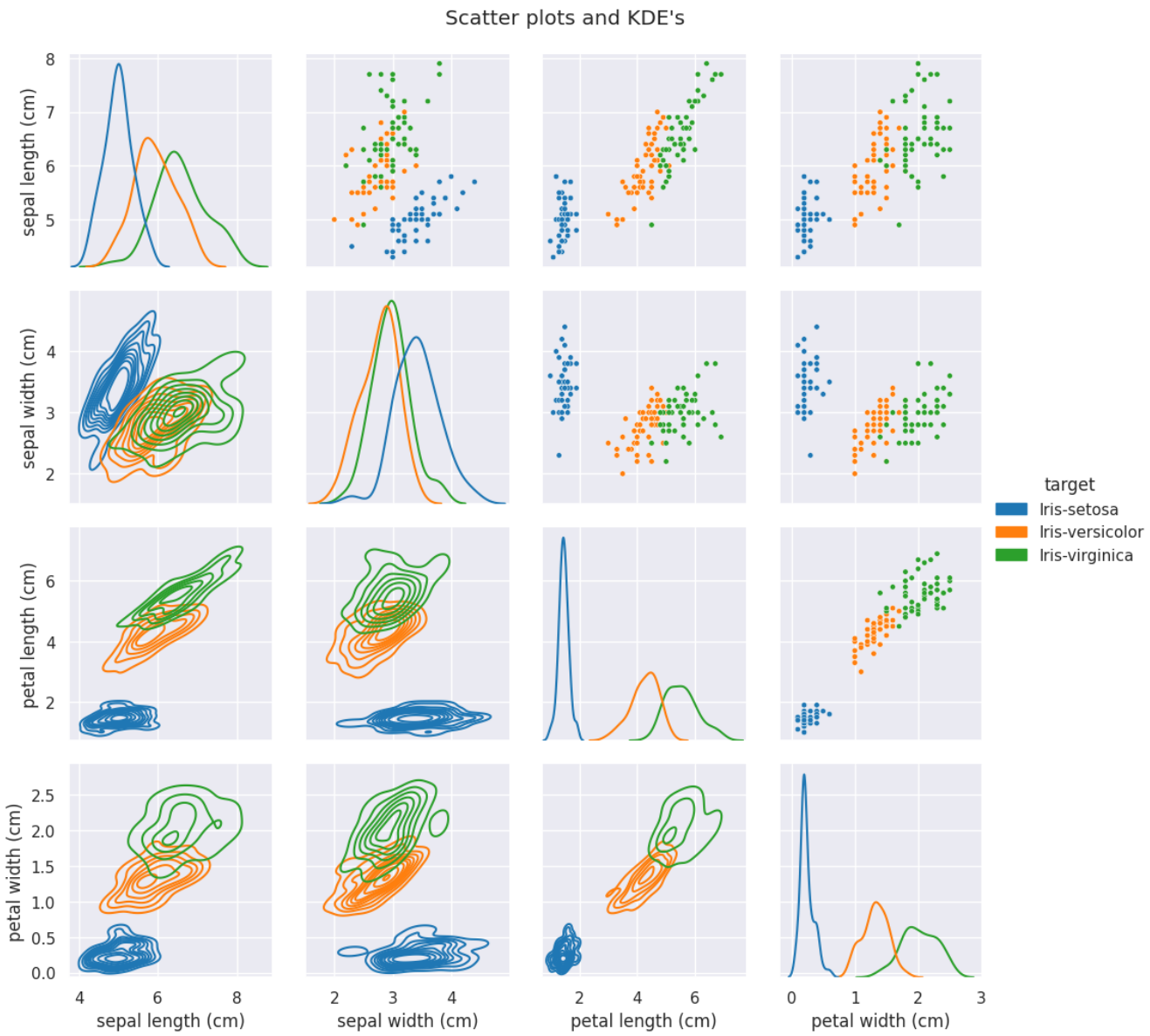

Рисунок 1.5.1 – Графики оценки ядерной плотности каждого признака, диаграмм рассеивания и двумерной оценки ядерной плотности

Исходя из графиков, можно сделать следующие выводы:

• «Iris-setosa» можно отделить от остальных классов по всем парам признаков;

• «Iris-setosa» чаще всего имеет «sepal length (cm)» около 5 см, «Iris-versicolor» – около 5.75 см, «Iris-virginica» – около 6 см;

• «Iris-setosa» чаще всего имеет «sepal width (cm)» около 3.4 см, «Iris-versicolor» – около 2.8 см, «Iris-virginica» – около 3 см;

• «Iris-setosa» чаще всего имеет «petal length (cm)» около 1.5 см, «Iris-versicolor» – около 4.5 см, «Iris-virginica» – между 5 см и 5.75 см;

• «Iris-setosa» чаще всего имеет «petal width (cm)» около 0.25 см, «Iris-versicolor» – около 1.4 см, «Iris-virginica» – около 1.8 см.

1.6. На одном изображении постройте гистограммы распределения для каждого признака.

Для выполнения данного задания и его подпунктов была реализована функция histograms_plot, которая выполняет следующие действия:

• создаёт заданное количество пустых графиков;

• в зависимости от наличия параметра hue задаёт палитру;

• рисует гистограммы в соответствии с переданными параметрами и задаёт им названия;

• задаёт расстояние между графиками.

Реализация данной функции и её вызова представлены в Листинге 1.6.1, графики гистограмм распределения каждого признака представлены на Рисунке 1.6.1.

Листинг 1.6.1 – Реализация функции для рисования гистограмм в соответствии с переданными параметрами

```
def histograms_plot(dataset, n_cols, n_bins=15, hue_val=None,
    kde_val=False, multiple_mode='layer', element_val='bars',
    title_name=''):

    fig, axes = plt.subplots(nrows=1, ncols=n_cols, figsize=(n_cols**2, n_cols))
    palette_val = None
    if hue_val is not None:
        palette_val = 'tab10'
    for i in range(n_cols):
        sns.histplot(data=dataset, x=dataset.columns[i],
                     hue=hue_val, kde=kde_val, ax=axes[i],
                     palette=palette_val, bins=n_bins,
                     multiple=multiple_mode, element=element_val)
    fig.suptitle(title_name, y=1.02)
    plt.tight_layout(pad=0.5)
    plt.show()

histograms_plot(dataset=iris_data, n_cols=len(iris_data.columns)-1)
```

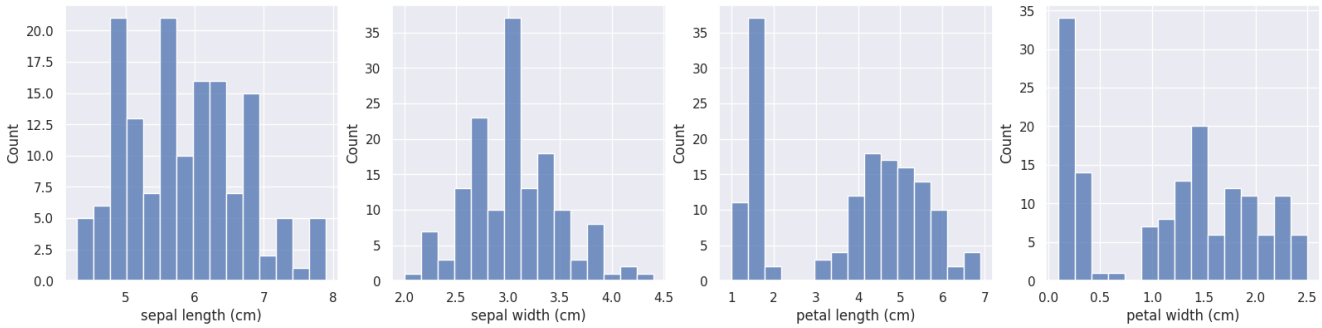

Рисунок 1.6.1 – Распределения признаков

1.6.1. Постройте гистограммы для разного количества столбцов: 5,10,15,20,30. Выберите на ваш взгляд такое количество столбцов, который лучшие образом описывает форму распределения признаков.

Для выполнения данного задания достаточно вызвать функцию histogram_plot с разными параметрами n_bins и оценить результат. Реализация вызовов представлена в Листинге 1.6.1.1, результат вызовов представлен на Рисунке 1.6.1.1, Рисунке 1.6.1.2, Рисунке 1.6.1.3, Рисунке 1.6.1.4, Рисунке 1.6.1.5 для количества столбцов 5, 10, 15, 20, 30 соответственно.

Листинг 1.6.1.1 – Реализация вызовов функции с разным количеством столбцов гистограммы

```
histograms_plot(dataset=iris_data, n_bins=5, n_cols=len(iris_data.columns)-1)
histograms_plot(dataset=iris_data, n_bins=10, n_cols=len(iris_data.columns)-1)
histograms_plot(dataset=iris_data, n_bins=15, n_cols=len(iris_data.columns)-1)
histograms_plot(dataset=iris_data, n_bins=20, n_cols=len(iris_data.columns)-1)
histograms_plot(dataset=iris_data, n_bins=30, n_cols=len(iris_data.columns)-1)
```

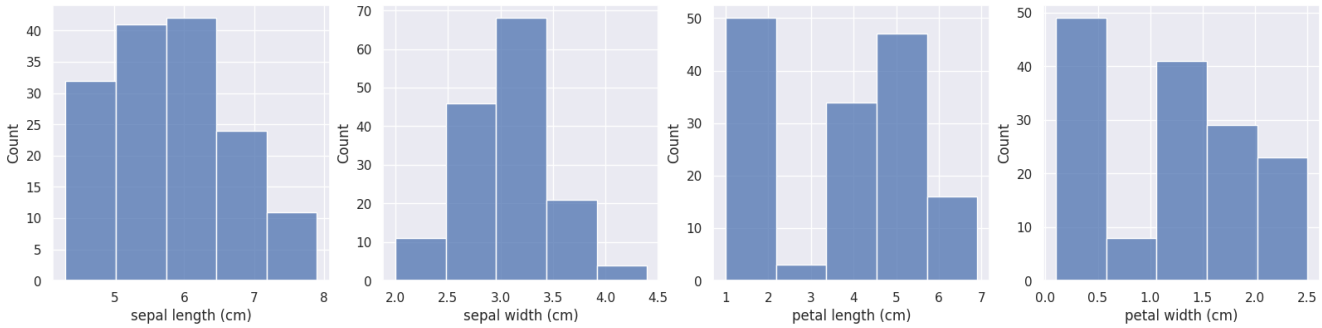

Рисунок 1.6.1.1 – Распределения признаков, когда количество столбцов равно 5

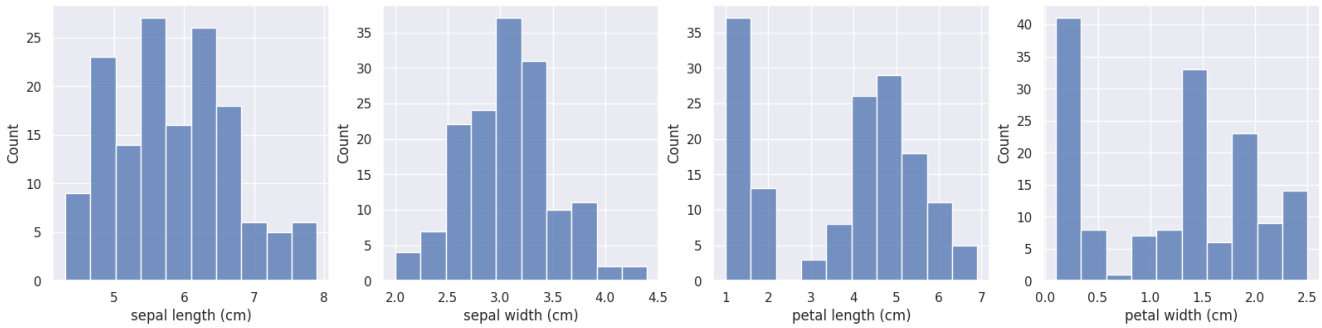

Рисунок 1.6.1.2 – Распределения признаков, когда количество столбцов равно 10

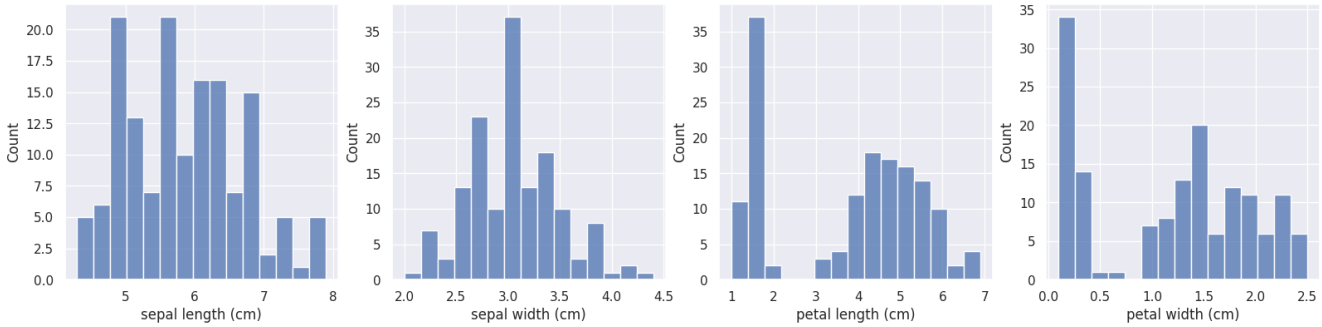

Рисунок 1.6.1.3 – Распределения признаков, когда количество столбцов равно 15

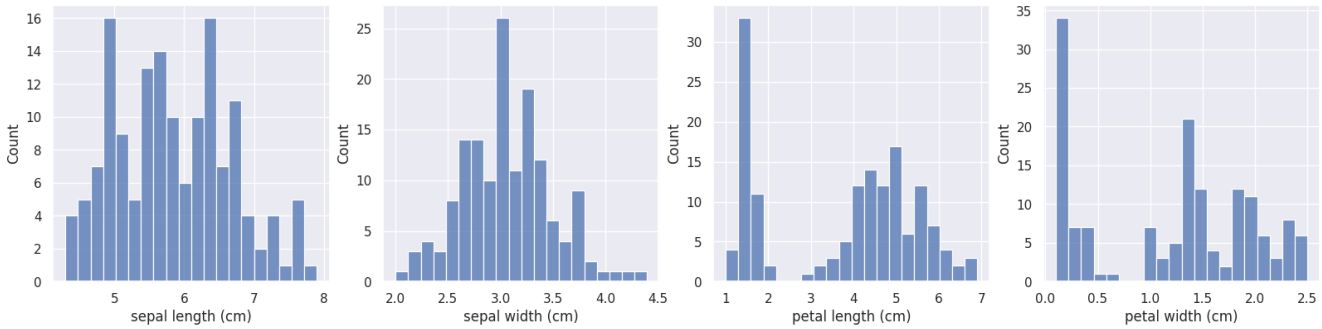

Рисунок 1.6.1.4 – Распределения признаков, когда количество столбцов равно 20

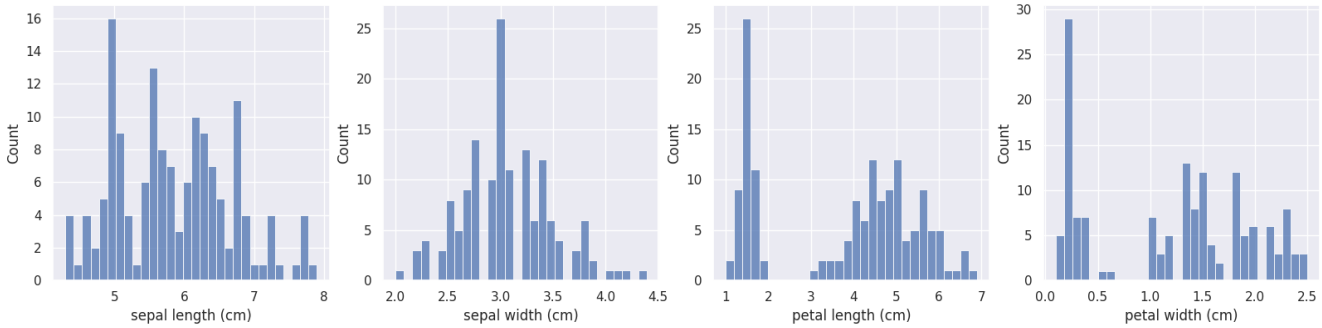

Рисунок 1.6.1.5 – Распределения признаков, когда количество столбцов равно 30

По-моему, оптимальное количество столбцов равно 10, так как при таком дроблении гистограмма не выглядит перегруженной, но при этом не кажется слишком обобщённой.

1.6.2. Сделайте на каждой гистограмме разделение по цвету согласно классу. Проведите это в двух режимах, когда гистограммы накладываются/суммируются и когда пересекаются.

Для разделения датасета на гистограмме по цвету класса достаточно вызвать функцию. histogram_plot с параметром hue_val равным 'target'. За наложение и суммирование отвечает параметр multiple_mode, по умолчанию равный 'layer'. Реализация данного шага при наложении и суммировании представлена в Листинге 1.6.2.1, результат при наложении представлен на Рисунке 1.6.2.1, при суммировании – а Рисунке 1.6.2.2.

Листинг 1.6.2.1 – Реализация вызовов функции в режиме наложения и суммирования

```
histograms_plot(dataset=iris_data, n_bins=10, hue_val='target',
↪    n_cols=len(iris_data.columns)-1)
histograms_plot(dataset=iris_data, n_bins=10, hue_val='target',multiple_mode='stack',
↪    n_cols=len(iris_data.columns)-1)
```

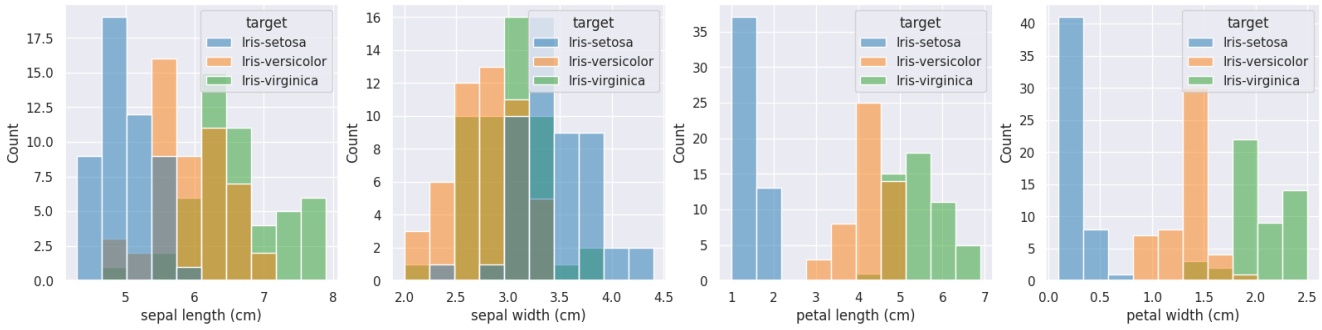

Рисунок 1.6.2.1 – Распределения признаков при делении на классы в режиме наложения

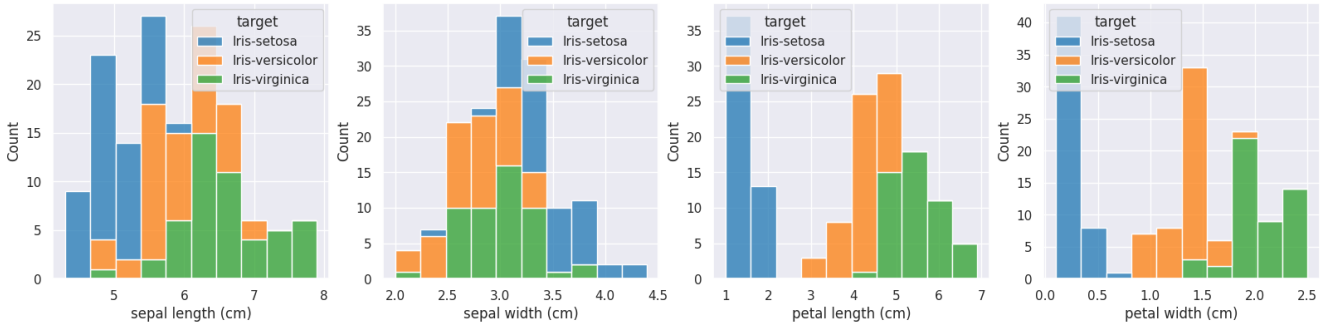

Рисунок 1.6.2.2 – Распределения признаков при делении на классы в режиме суммирования

1.6.3. Постройте гистограммы, чтобы вместо столбцов изображались ступеньки.

Для реализации данного задания следует передать значение 'step' в параметр element_val функции histograms_plot. Реализация вызова представлена в Листинге 1.6.3.1, построенные гистограммы представлены на Рисунке 1.6.3.1.

Листинг 1.6.3.1 – Реализация вызова функции для рисовании гистограмм в виде ступенек

```
histograms_plot(dataset=iris_data, n_bins=10, hue_val='target', multiple_mode='stack',
↪   element_val='step', n_cols=len(iris_data.columns)-1)
```

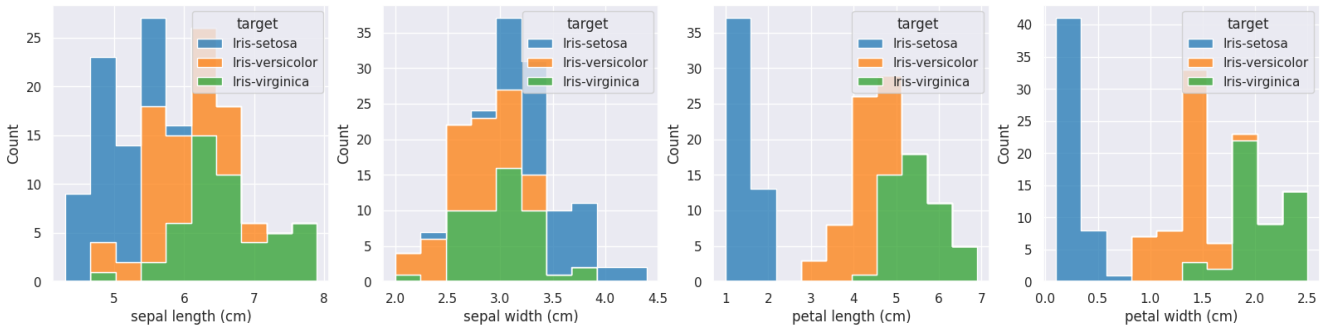

Рисунок 1.6.3.1 – Распределения признаков при отрисовке гистограмм в виде ступенек

1.6.4. Добавьте на гистограммы график ядерной оценки плотности

Для реализации данного задания следует передать значение 'True' в параметр kde_val функции histograms_plot. Реализация вызова представлена в Листинге 1.6.4.1, построенные гистограммы представлены на Рисунке 1.6.4.1.

Листинг 1.6.4.1 – Реализация вызова функции для рисовании гистограмм и ядерной оценки плотности

```
histograms_plot(dataset=iris_data, n_bins=10, hue_val='target', kde_val=True,
↪   element_val='step', n_cols=len(iris_data.columns)-1)
```

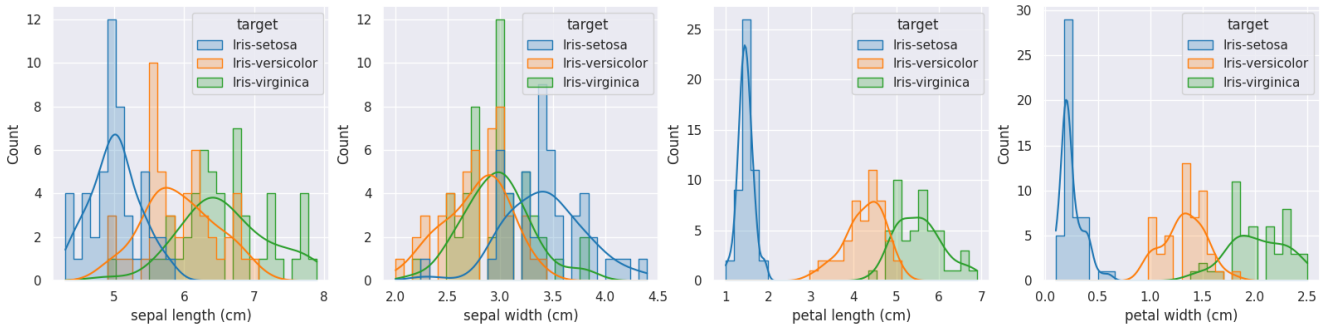

Рисунок 1.6.4.1 – Распределения признаков с отрисовкой ядерной оценки плотности

2. Изучение набора данных iris.csv с использованием NumPy

2.1. Загрузите данные из файла как массив NumPy

Для выполнения поставленных задач необходимо импортировать библиотеку numpy, вернуть данные в исходную форму, а именно, текстовые метки классы заменить на числовые, и, наконец, преобразовать данный в numpy-массив. Реализация вышеописанных операций представлена в Листинге 2.1.1.

Листинг 2.1.1 – Реализация преобразования данных в numpy-массив

```
import numpy as np
iris_data['target'].replace(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], [0, 1, 2],
↪   inplace=True)
iris_data_np = iris_data.to_numpy()
iris_data_np[:10]
```

2.2. Выведите первые 10 наблюдений набора данных

Для реализации данной задачи достаточно воспользоваться операцией вырезки (см. Листинг 2.2.1).

Листинг 2.2.1 – Вывод первых десяти записей датасета

```
array([[5.1, 3.5, 1.4, 0.2, 0. ],
   [4.9, 3. , 1.4, 0.2, 0. ],
   [4.7, 3.2, 1.3, 0.2, 0. ],
   [4.6, 3.1, 1.5, 0.2, 0. ],
   [5. , 3.6, 1.4, 0.2, 0. ],
   [5.4, 3.9, 1.7, 0.4, 0. ],
   [4.6, 3.4, 1.4, 0.3, 0. ],
   [5. , 3.4, 1.5, 0.2, 0. ],
   [4.4, 2.9, 1.4, 0.2, 0. ],
   [4.9, 3.1, 1.5, 0.1, 0. ]])
```

2.3. Рассчитайте характеристики полученные методом describe в п. 1.3 с использованием методов NumPy.

Для выполнения данного задания был создан словарь describe, который имел такие ключи, как 'count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max'. Для каждого ключа вызывается соответствующий названию ключа метод библиотеки numpy, возвращающий числовое значение. Почти у всех методов можно заметить параметр axis, равный 0. Значение данного параметра говорит о том, что вычисления будут производиться относительно столбцов датасета. Для расчёта стандартного отклонения использовалась несмещённая оценка, что отражено в значении параметра ddof, ведь pandas по умолчанию использует именно эту оценку. Затем данный словарь использовался для создания датафрейма и последующего его сравнения со значением метода describe для датасета из iris.csv. Реализация расчёта характеристик представлена в Листинге 2.3.1, результат сравнения – в Листинге 2.3.2.

Листинг 2.3.1 – Реализация расчёта и сравнения характеристик датасета

```
description = {'count': np.array([len(iris_data_np[:,i] != np.nan) for i in range(5)]),
'mean': np.mean(iris_data_np, axis=0, ddof=1),
'std': np.std(iris_data_np, axis=0),
'min': np.min(iris_data_np, axis=0),
'25%': np.quantile(iris_data_np, q=0.25, axis=0),
'50%': np.quantile(iris_data_np, q=0.5, axis=0),
'75%': np.quantile(iris_data_np, q=0.75, axis=0),
'max': np.max(iris_data_np, axis=0)}
description_pd = pd.DataFrame(description, index=['sepal length (cm)', 'sepal width (cm)',
↪   'petal length (cm)', 'petal width (cm)', 'target']).T
description_pd == iris_data.describe()
```

Листинг 2.3.2 – Результат сравнения собственных расчётов с помощью numpy и метода describe из pandas

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	True	True	True	True	True
mean	True	True	True	True	True
std	True	True	True	True	True
min	True	True	True	True	True
25%	True	True	True	True	True
50%	True	True	True	True	True
75%	True	True	True	True	True
max	True	True	True	True	True

3. Изучение набора данных вашего варианта

3.1. Оцените и опишите набор данных вашего варианта с использованием методов в п.1

Реализация загрузки данных из файла и вывода первых пяти записей датафрейма представлены в Листинге 3.1.1, результат вывода записей – Листинге 3.1.2.

Листинг 3.1.1 – Реализация загрузки данных из файла и вывода первых пяти записей датафрейма

```
var2_data = pd.read_csv('sample_data/lab1_var2.csv')
var2_data.head()
```

Листинг 3.1.2 – Результат вывода первых пяти записей датафрейма

	Unnamed: 0	A	B	C	D	E	label
0	0	3.820369	0.944005	-2.515361	2.691548	4.587330	1
1	1	2.317692	-4.031270	1.096187	-4.324973	4.395063	1
2	2	3.031215	2.313363	-3.358096	2.744431	3.922414	0
3	3	2.880855	-2.467154	2.550000	-2.215556	0.858642	1
4	4	3.674317	-5.525880	5.199131	1.230003	2.516113	1

В Листинге 3.1.3 представлено две команды: первая удаляет лишний первый столбец, вторая – выводит первые пять записей видоизменённого датафрейма.

Листинг 3.1.3 – Реализация загрузки данных из файла и вывода первых пяти записей датафрейма

```
del var2_data[var2_data.columns[0]]
var2_data.head()
```

Результат выполнения метода describe для данного датасета представлен в Листинге 3.1.4.

Листинг 3.1.4 – Результат вывода первых пяти записей датафрейма

	A	B	C	D	E	label
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	3.059112	-0.050611	0.043850	-0.089783	3.123704	0.565000
std	1.167444	3.336619	3.126977	3.241638	1.337344	0.497001
min	0.153019	-6.179742	-5.131545	-7.268231	-0.458386	0.000000
25%	2.337088	-2.951520	-3.030783	-3.032036	2.279943	0.000000
50%	3.037541	0.129381	0.374458	1.189169	3.103441	1.000000
75%	3.804794	2.896585	2.984745	2.992634	3.977151	1.000000
max	6.560331	6.426588	5.199131	4.403511	7.304125	1.000000

Реализация замены значений 'label' и вывода первых трёх записей датафрейма для проверки замены представлены в Листинге 3.1.5, результат вывода – в Листинге 3.1.6.

Листинг 3.1.5 – Реализация замены значений 'label' и вывода первых трёх записей датафрейма

```
var2_data['label'].replace([0, 1],
['Iris-setosa', 'Iris-versicolor'], inplace=True)
var2_data.head(3)
```

Листинг 3.1.6 – Результат вывода первых трёх записей датафрейма

	A	B	C	D	E	label
0	3.820369	0.944005	-2.515361	2.691548	4.587330	Iris-versicolor
1	2.317692	-4.031270	1.096187	-4.324973	4.395063	Iris-versicolor
2	3.031215	2.313363	-3.358096	2.744431	3.922414	Iris-setosa

Для построения графиков рассеяния, одномерной и двумерной оценки ядерной плотности следует воспользоваться функцией pair_grid (см. Листинге 3.1.7). Набор графиков изображён на Рисунке 3.1.1.

Листинг 3.1.7 – Реализация замены значений ’label’ и вывода первых трёх записей датафрейма

```
pair_grid(dataset=var2_data, target='label')
```

Листинг 3.1.8 – Реализация замены значений ’label’ и вывода первых трёх записей датафрейма

```
pair_grid(dataset=var2_data, target='label')
```

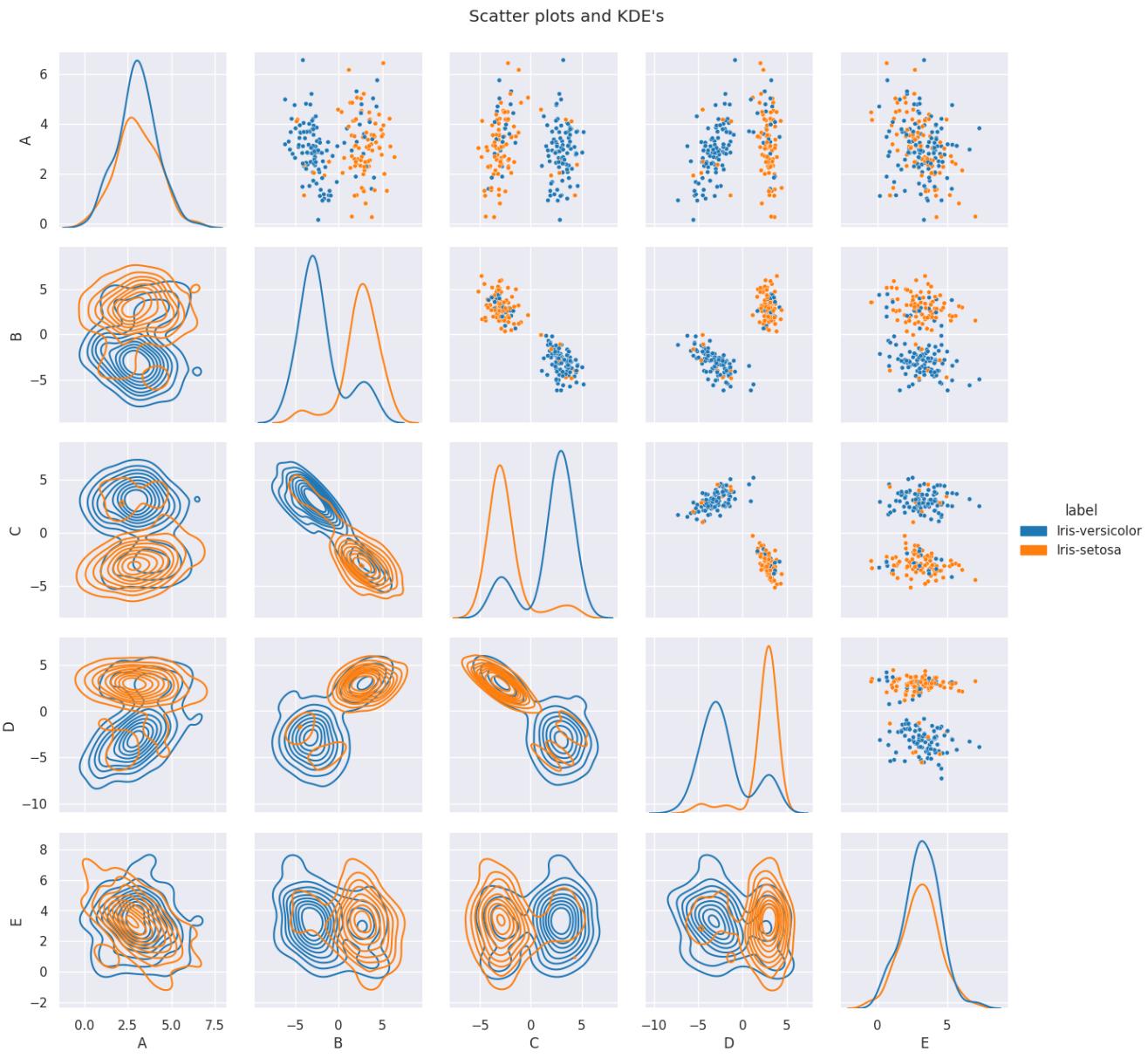

Рисунок 3.1.1 – Графики оценки ядерной плотности каждого признака, диаграмм рассеивания и двумерной оценки ядерной плотности

Исходя из графиков, можно сделать следующие выводы:

• классы не разделимы;

• «Iris-setosa» чаще всего имеет значение признака «A» около 2.5, «Iris-versicolor» – около 2.75;

• «Iris-setosa» чаще всего имеет значение признака «B» около 2.5, «Iris-versicolor» – около -2.5;

• «Iris-setosa» чаще всего имеет значение признака «C» около -2.5, «Iris-versicolor» – около 2.5;

• «Iris-setosa» чаще всего имеет «D» около 3, «Iris-versicolor» – около -3;

• «Iris-setosa» и «Iris-versicolor» чаще всего имеет «E» около 3.5, но у «Iris-versicolor» эта вероятность выше.

Построение гистограмм распределения для каждого признака реализовано с помощью функции histograms_plot, которая получает на вход следующие данные:

• dataset=var2_data – искомый датасет;

• n_bins=15 – явно не указывается, значит по умолчанию количество столбцов будет равно 15 (для данного датасета оптимальное количество);

• hue_val='label' – данные разделятся по данному значению;

• multiple_mode='layer' – используется наложение, так как имеется всего лишь два класса, гистограмма не будет перегружена разными цветами;

• kde_val=True – на графике нужно отобразить ядерную оценку плотности;

• n_cols=len(var2.columns)-1 – передать столбцы, кроме последнего, ведь он является классом.

Результат построения гистограмм представлен на Рисунке 3.1.2.

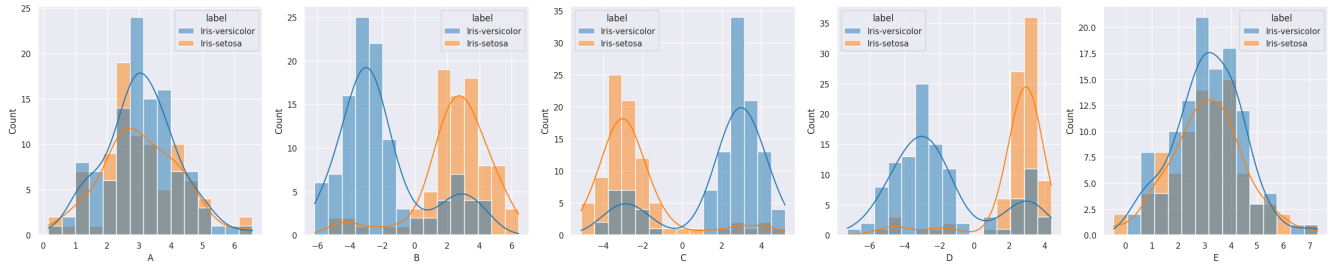

Рисунок 3.1.2 – Распределения признаков

4. Преобразование данных:

4.1. Получите из датафрейма из п. 1.4 столбец с названием классов. Используя LabelEncoder и OneHotEncoder получите различные способы кодирования меток класса. В чем различия полученных кодировок?

Реализация кодирования с помощью LabelEncoder представлена в Листинге 4.1.1. Для кодирования достаточно получить экземпляр класса LabelEncoder и вызвать у него метод fit_transform с параметром, равным целевому столбцу датасета. Результат кодирования записей с индексами 0, 50, 100 представлен в Листинге 4.1.2.

Листинг 4.1.1 – Реализация кодирования с помощью LabelEncoder

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

label_encoder = LabelEncoder()
label_encoder.fit_transform(iris_data['target'])[[0, 50, 100]]
```

Листинг 4.1.2 – Реализация кодирования с помощью LabelEncoder

```
array([0, 1, 2])
```

Реализация кодирования с помощью OneHotEncoder представлена в Листинге 4.1.3. Для кодирования достаточно получить экземпляр класса OneHotEncoder и вызвать у него метод fit_transform с параметром, равным целевому столбцу датасета. Результат кодирования записей с индексами 0, 50, 100 представлен в Листинге 4.1.4.

Листинг 4.1.3 – Реализация кодирования с помощью OneHotEncoder

```
one_hot_encoder = OneHotEncoder()
one_hot_encoder.fit_transform(iris_data[['target']]).toarray()[[0, 50, 100]]
```

Листинг 4.1.4 – Реализация кодирования с помощью OneHotEncoder

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Отличие данных способов кодирования заключается в том, что LabelEncoder сопоставляет значениям категориальных признаков числа из множества {0, 1, ..., n_classes - 1}, а OneHotEncoder каждому значению категориального признака сопоставляет маску, длина которой равна количеству категорий. Единица устанавливается в том индексе маски, которому соответствует данная категория.

4.2. Для датафрейма из п. 1.4, получите все столбцы признаков (столбцы не содержащие метки классов). Преобразуйте полученные столбцы в массив NumPy.

Для получения столбцов, не содержащих метки классов, достаточно удалить столбец классов с помощью метода drop с признаком удаления 'target' вдоль столбцов. Полученный результат следует преобразовать в numpy-массив с помощью метода to_array(). Реализация вышеописанных шагов и вызов вывода результата представлены в Листинге 4.2.1, результат операций – в Листинге 4.2.2.

Листинг 4.2.1 – Реализация удаления столбца 'target', преобразования в numpy-массив и вывода первых пяти записей датасета

```
iris_data_np = iris_data.drop('target', axis=1).to_numpy()
iris_data_np[:5]
```

Листинг 4.2.2 – Результат операций удаления, преобразования в numpy-массив и вывода первых пяти записей датасета

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

4.3. Для массива NumPy из п. 4.2 примените StandardScaler, MinMaxScaler, MaxAbsScaler и RobustScaler. Для каждого из результатов постройте гистограммы по каждому признаку без разделения по классам. В чем различия между такими преобразованиями данных?

Для реализации данной задачи необходимо импортировать из библиотеки sklearn.preprocessing препроцессоры StandardScaler, MinMaxScaler, MaxAbsScaler и RobustScaler и создать массив экземпляров данных классов. Затем с помощью

функции histograms_plot сделать отрисовку данных без использования препроцессоров и с использованием. Для отрисовки предобработанных данных производилось преобразование в датафреймы pandas для получения возможности использовать функцию histograms_plot. Реализация данных шагов представлена в Листинге 4.3.1, распределния признаков без предобработки – Рисунке 4.3.1, распределения признаков после предобработки с помощью StandardScaler – на Рисунке 4.3.2, MinMaxScaler – Рисунке 4.3.3, MaxAbsScaler – Рисунке 4.3.4, RobustScaler – Рисунке 4.3.5.

Листинг 4.3.1 – Реализация отрисовки данных после предобработки с помощью StandardScaler, MinMaxScaler, MaxAbsScaler и RobustScaler

```
from sklearn.preprocessing import StandardScaler, \
MinMaxScaler, MaxAbsScaler, RobustScaler

preprocessors = [StandardScaler(), MinMaxScaler(), MaxAbsScaler(), RobustScaler()]

histograms_plot(dataset=iris_data, n_cols=len(iris_data.columns)-1, title_name='Without
↪  any scaler')

for preproc in preprocessors:
    iris_data_preproc = pd.DataFrame(preproc.fit_transform(iris_data_np).T,
    ↪  index=iris_data.columns[:-1]).T
    histograms_plot(dataset=iris_data_preproc, n_cols=len(iris_data_preproc.columns),
    ↪  title_name=str(preproc)[:-2])
```

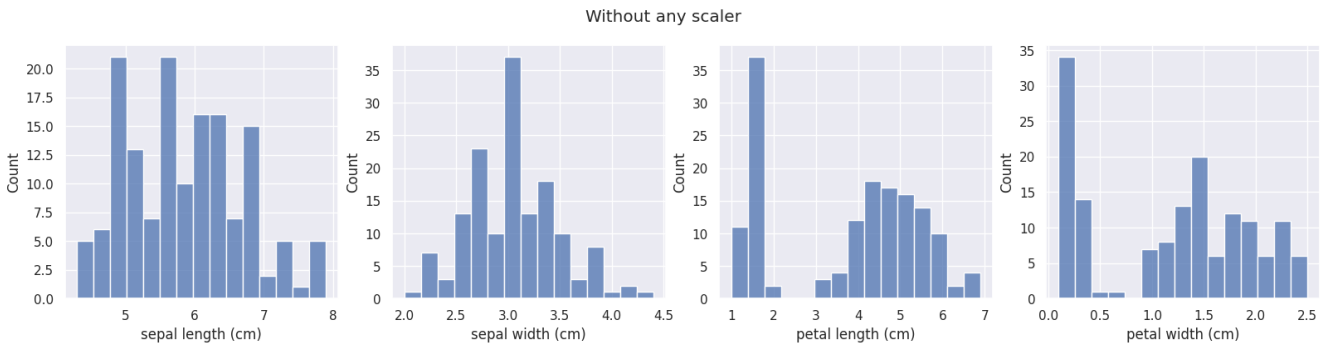

Рисунок 4.3.1 – Распределения признаков без предобработки

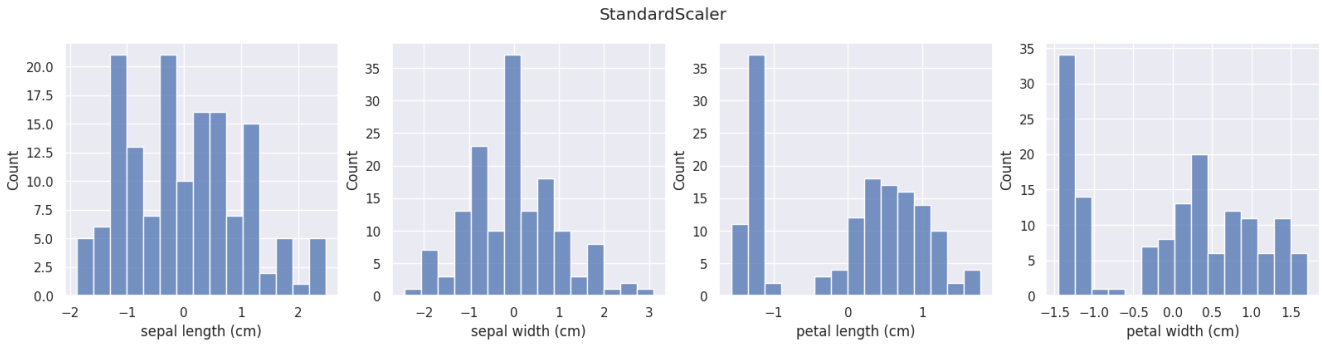

Рисунок 4.3.2 – Распределения признаков после предобработки StandardScaler

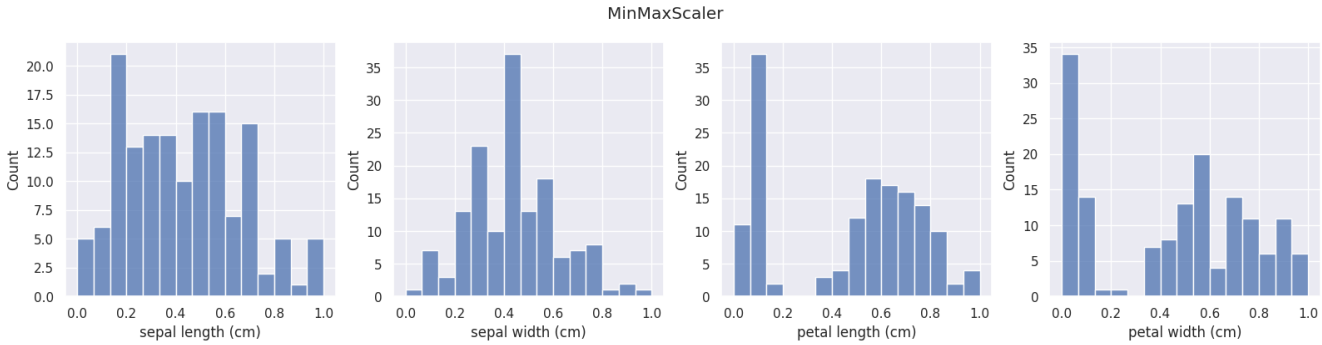

Рисунок 4.3.3 – Распределения признаков после предобработки MinMaxScaler

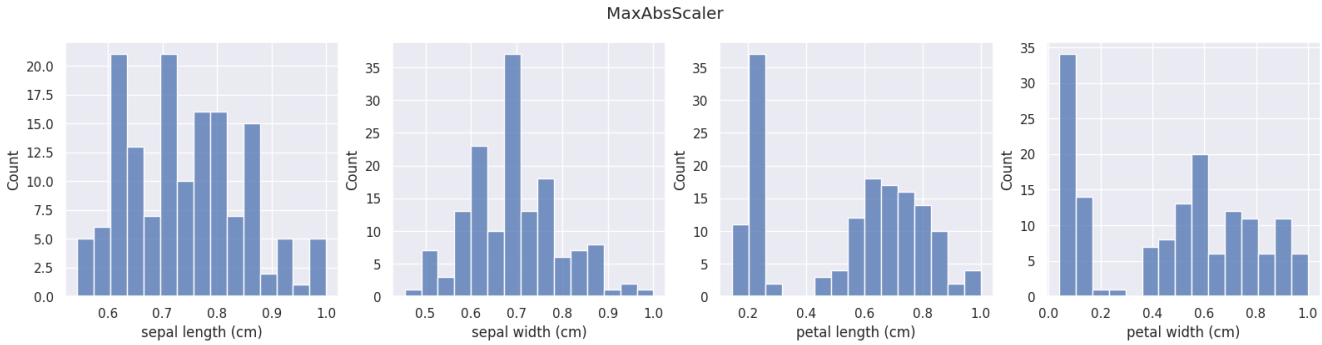

Рисунок 4.3.4 – Распределения признаков после предобработки MaxAbsScaler

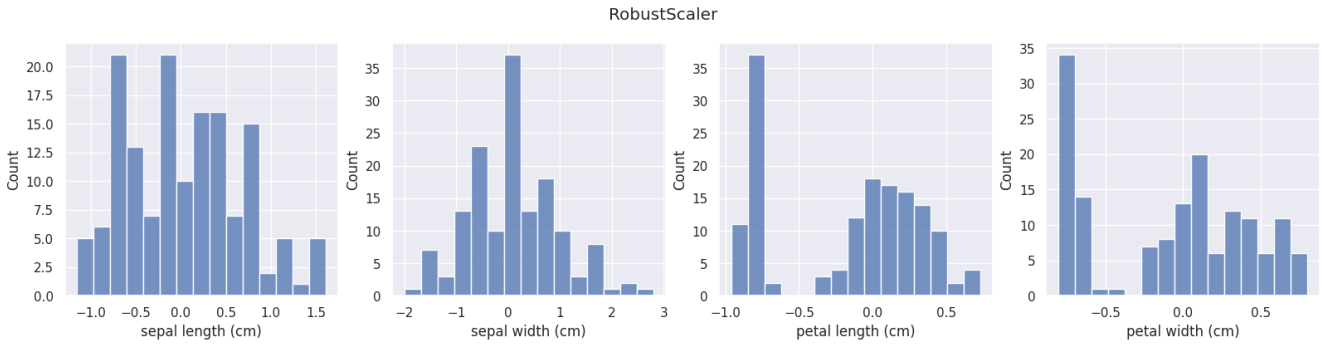

Рисунок 4.3.5 – Распределения признаков после предобработки RobustScaler

Описание StandardScaler:

• формула преобразования:

$$\hat{\mathbf{X}} = \frac{\mathbf{X} - \mathbb{E}[\mathbf{X}]}{\sigma[\mathbf{X}]} \tag{1}$$

• данные центрируются вокруг среднего значения с единичным стандартным отклонением (дисперсией);

• следует применять, если данные подчиняются нормальному распределению;

• стандартизация менее чувствительна к выбросам по сравнению с MinMaxScaler и MaxAbsScaler.

Описание MinMaxScaler:

• формула преобразования:

$$\hat{\mathbf{X}} = \frac{\mathbf{X} - \min[\mathbf{X}]}{\max[\mathbf{X}] - \min[\mathbf{X}]} \tag{2}$$

• проецирует данные в [0, 1];

• следует применять, если данные не подчиняются нормальному распределению;

• нормализация чувствительна к выбросам.

Описание MaxAbsScaler:

• формула преобразования:

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{\max[|\mathbf{X}|]} \tag{3}$$

• проецирует данные в [-1, 1], таким образом, сохраняется знак значения признака;

• остальные свойства такие же, как и у MinMaxScaler.

Описание RobustScaler:

• формула преобразования:

$$\hat{\mathbf{X}} = \frac{\mathbf{X} - \tilde{\mathbf{X}}}{\mathbf{Q_3} - \mathbf{Q_1}} \tag{4}$$

• данный метод предназначен для решения проблем с выбросами;

• данные центрируются относительно медианы и делятся на разницу между 1 и 3 квартильями.

4.4. Согласно варианту, самостоятельно реализуйте StandardScaler или MinMaxScaler с использованием NumPy. Проверьте корректность работы на вашем наборе данных, сравните результаты между вашей реализацией и реализацией из Sklearn, а также рассчитав минимальное, максимальное, среднее значение и дисперсию, после преобразования.

Для реализации данного задания нужно вычислить среднее для каждого признака датасета, стандартное отклонение, разделить каждую компоненту вектора среднего на каждую компоненту вектора стандартного отклонения и сформировать матрицу, состоящую из полученного вектора. Таким образом, строки матрицы будут одинаковыми и их количество равно мощности датасета. Затем, каждую строку искомого датасета нужно разделить поэлементно на вектор стандартного отклонения и отнять матрицу, сформированную на предыдущем шаге. Для проверки преобразования следует найти модуль разности полученного результата и матрицы, сформированной с помощью встроенного StandardScaler. Реализация данных шагов представлена в Листинге 4.4.1, результат сравнения – в Листинге 4.4.2.

Листинг 4.4.1 – Самостоятельная реализация StandardScaler

```
mean = np.mean(iris_data_np, axis=0)
std = np.std(iris_data_np, axis=0)
vec = mean/std
matrix = np.array([vec for i in range(len(iris_data_np))])
iris_data_scaled_np = iris_data_np/std - matrix
np.abs(iris_data_scaled_np -
       StandardScaler().fit_transform(iris_data_np))[0:10]
```

Листинг 4.4.2 – Сравнение самостоятельной реализации StandardScaler с уже существующей

```
array([[4.44089210e-16, 2.22044605e-16, 2.22044605e-16, 2.22044605e-16],
   [2.22044605e-16, 2.22044605e-16, 2.22044605e-16, 2.22044605e-16],
   [2.22044605e-16, 3.88578059e-16, 2.22044605e-16, 2.22044605e-16],
   [0.00000000e+00, 1.38777878e-16, 0.00000000e+00, 2.22044605e-16],
   [0.00000000e+00, 8.88178420e-16, 2.22044605e-16, 2.22044605e-16],
   [1.11022302e-16, 2.22044605e-16, 2.22044605e-16, 2.22044605e-16],
   [0.00000000e+00, 2.22044605e-16, 2.22044605e-16, 0.00000000e+00],
   [0.00000000e+00, 2.22044605e-16, 0.00000000e+00, 2.22044605e-16],
   [2.22044605e-16, 2.77555756e-16, 2.22044605e-16, 2.22044605e-16],
   [2.22044605e-16, 1.38777878e-16, 0.00000000e+00, 2.22044605e-16]])
```

Исходя из данной матрицы можно сделать вывод, что собственная реализация незначительно отличается от библиотечной, что может быть связано с разными подходами при вычислении.

Реализация расчёта минимального, максимального, среднего значения и дисперсии после преобразования представлена в Листинге 4.4.3, демонстрация результатов расчёта – в Листинге 4.4.4.

Листинг 4.4.3 – Реализация расчёта статистических параметров

```
mean = np.mean(iris_data_scaled_np, axis=0)
var = np.var(iris_data_scaled_np, axis=0)
min = np.min(iris_data_scaled_np, axis=0)
max = np.max(iris_data_scaled_np, axis=0)
print(f"mean = {mean}")
print(f"var = {var}")
print(f"min = {min}")
print(f"max = {max}")
```

Листинг 4.4.4 – Статистические данные после предобработки с помощью StandardScaler

```
mean = [-6.57252031e-16 -6.92779167e-16 -9.62193288e-16 -2.75335310e-16]
var = [1. 1. 1. 1.]
min = [-1.87002413 -2.43394714 -1.56757623 -1.44707648]
max = [2.4920192  3.09077525 1.78583195 1.71209594]
```

Исходя из полученных результатов, можно сделать вывод, что преобразование выполнено правильно, так как среднее значение по каждому признаку равно 0, а дисперсия равна 1.

4.5. Для датафрейма из п. 1.4 получите новый, который содержит только классы Iris-versicolor и Iris-virginica, признаки “sepal length (cm)” и “petal length (cm)”, и наблюдения, для которых значения признака “sepal width (cm)” лежат между квантилями 25% и 75%.

Для выполнения данного задания нужно вернуть датасет в ту форму, в которой он был на момент выполнения п.1.4, что реализовывается с помощью метода replace, который заменяет числовые метки на категориальные. Для извлечения объектов с метками ’Iris-versicolor’ и ’Iris-virginica’ следует воспользоваться методом isin. Для расчёта квантилий достаточно вызвать методы библиотеки numpy. Для завершения отбора записей по значениям признака “sepal width (cm)” достаточно воспользоваться механизмом селекции в pandas. Для удаления признаков “sepal length (cm)” и “petal length (cm)” достаточно воспользоваться методом drop. Реализация данных команд представлена в Листинге 4.5.1, первые пяти записей полученного датафрейма представлены в Листинге 4.5.2.

Листинг 4.5.1 – Реализация селекции в датафрейме

```
iris_data['target'].replace([0, 1, 2],
 ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], inplace=True)

new_iris_data = iris_data[iris_data['target'].isin(['Iris-versicolor', 'Iris-virginica'])]
low_lim = np.quantile(new_iris_data.to_numpy()[:, 1], q=0.25)
high_lim = np.quantile(new_iris_data.to_numpy()[:, 1], q=0.75)
new_iris_data = new_iris_data[(new_iris_data['sepal width (cm)'] > low_lim) &
↪   (new_iris_data['sepal width (cm)'] < high_lim)]

new_iris_data.drop(columns=['sepal width (cm)', 'petal width (cm)'], inplace=True)
new_iris_data.head()
```

Листинг 4.5.2 – Результат селекции

	sepal length (cm)	petal length (cm)	target
54	6.5	4.6	Iris-versicolor
55	5.7	4.5	Iris-versicolor
58	6.6	4.6	Iris-versicolor
61	5.9	4.2	Iris-versicolor
63	6.1	4.7	Iris-versicolor

5. Понижение размерности:

5.1. Для набора данных iris.csv примените понижение размерности до 2, используя PCA и TSNE из Sklearn. Для каждого из результатов постройте диаграмму рассеяния с выделением разным цветом наблюдений разных классов.

Для решения поставленной задачи из sklearn.decomposition был импортирован PCA, а из sklearn.manifold – TSNE. Для понижения размерности с помощью PCA был создан экземпляр класса PCA с двумя осями проецирования, получены координаты в новой системе координат с помощью метода fit_transform, который получает на вход стандартизированный датасет. Затем к полученному датасету добавляется столбец меток для более информативного графического представления и, наконец, происходит отрисовка спроецированных данных. Аналогичные действия производятся и для понижения размерности с помощью TSNE, который по умолчанию проецирует данные на две оси. Исходный код представлен в Листинге 5.1.1, результат понижения размерности – на Рисунке 5.1.1.

Листинг 5.1.1 – Реализация понижения размерности с помощью PCA и t-SNE

```
iris_data['target'].replace([0, 1, 2],
 ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], inplace=True)

new_iris_data = iris_data[iris_data['target'].isin(['Iris-versicolor', 'Iris-virginica'])]
low_lim = np.quantile(new_iris_data.to_numpy()[:, 1], q=0.25)
high_lim = np.quantile(new_iris_data.to_numpy()[:, 1], q=0.75)
new_iris_data = new_iris_data[(new_iris_data['sepal width (cm)'] > low_lim) &
↪   (new_iris_data['sepal width (cm)'] < high_lim)]

new_iris_data.drop(columns=['sepal width (cm)', 'petal width (cm)'], inplace=True)
new_iris_data.head()
```

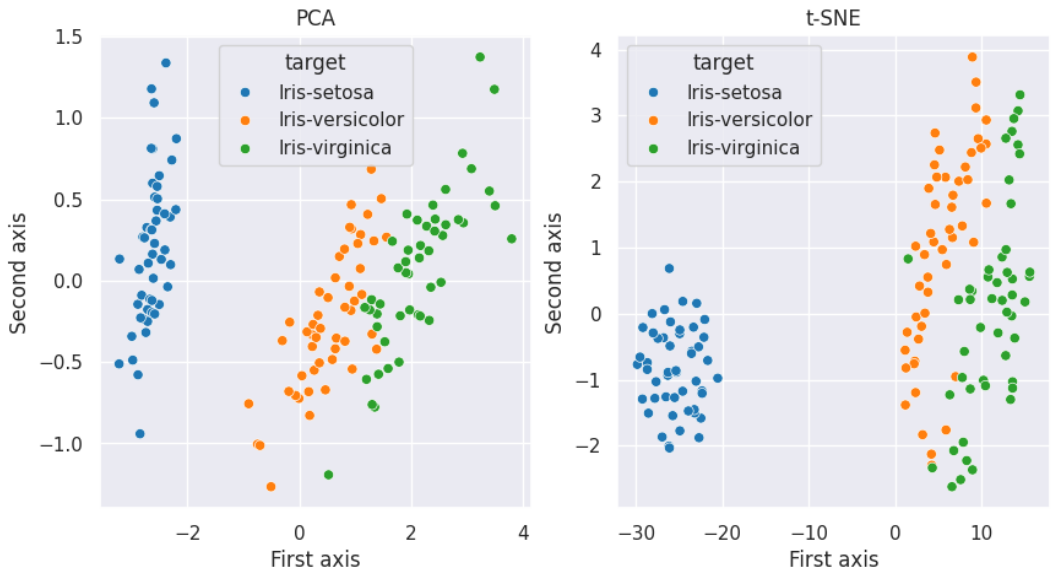

Рисунок 5.1.1 – Результат понижения размерности с помощью PCA и t-SNE

Исходя из полученных графиков можно ещё раз убедиться, что «Iris-setosa» меньше похож на остальные классы, чем они между собой.

Вывод.

В ходе выполнения работы было проанализировано два датасета. Для каждого датасета была произведена загрузка в программу, выполнена описательная статистика, изучены методы таких библиотек, как pandas, numpy, sklearn, построены гистограммы с разными параметрами, графики рассеяния и ядерной оценки плотности и сделаны соответствующие выводы. Были изучены методы кодирова-

ния категориальных признаков, предобработки данных, реализована собственная стандартизация данных и изучено два способа понижения размерности данных.