МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе № 1

по дисциплине «Основы машинного обучения»

Тема: изучение и предобработка данных

Вариант 1

Студентка гр. 1304 _____ Хорошкова А.С.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2024

Задание.

Изучить набра данных iris.csv с использованием Pandas и Seaborn.

Изучить набор данных iris.csv с использованием NumPy.

Изучить набор данных первого варианта (lab1_var1.csv).

Выполнить преобразование данных.

Понизить размерность данных.

Выполнение работы.

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

   Были загружены данные из файла iris.csv как Pandas DataFrame с помощью функции upload_df, представленной на Листинг 1.1.

Листинг 1.1

```
def upload_df(file_name, delete_first=True):
    df = pd.read_csv(file_name, delimiter=',')
    if delete_first:
        return df.drop(df.columns[[0]], axis=1)
    else:
        return df
```

С помощью метода print_df_data (Листинг 1.2) был вызван у датафрейма метод head, выводящий первые 5 строк датафрейма, и метод describe, выводящий характеристики датафрейма.

Листинг 1.2

```
def print_df_data(df):
    print("\n============== head ==============")
    print(df.head())
    print("\n============= describe ===============")
    print(df.describe())
```

Результат вывода представлен на Рисунок 1.

```
============== head ==============
   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

============= describe ===============
       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
```

Рисунок 1 - вывод первых пяти строк датафрейма и описания датафрейма

С помощью функции replace_df_column, представленной на Листинг 1.4, был видоизменён полученный датафрейм таким образом, чтобы метки классов были следующими: 0 - Iris-setosa, 1 - Iris-versicolor, 2 - Iris-virginica.

Листинг 1.3

```
def replace_df_column(df, column, class_mapping):
    df_copy = df.copy(deep=True)
    df_copy[column] = df_copy['target'].replace(class_mapping)
    print("\n============= replaced  ===============")
    print(df_copy.head())
    return df_copy
```

Первые пять строк изменённого датафрейма представлены на Рисунок 2.

```
============= replaced  ===============
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)      target
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 2 - датафрейм с заменёнными значениями столбца target

Полученный датафрейм был сохранён в отдельный файл формата csv с помощью функции safe_df, представленной на Листинг 1.4.

Листинг 1.4

```
def save_df(df, file_name):
    df.to_csv(file_name, index=False)
```

С помощью функции show_pair_grid, представленной на Листинг 1.5, были построены графики ядерной оценки плотности каждого признака (кроме признака класса), диаграмма рассеяния и двумерная ядерная оценка плотности для каждых признаков. Полученные графики представлены на Рисунок 3.

Листинг 1.5

```
def show_pair_grid(df, hue):
    sns.set_theme(style="white")
    sns.set_palette("Set1")
    g = sns.PairGrid(df, diag_sharey=False, hue=hue)
    g.map_upper(sns.scatterplot, s=15)
    g.map_lower(sns.kdeplot)
    g.map_diag(sns.kdeplot)
    g.add_legend()
    g.fig.show()
    plt.show()
```

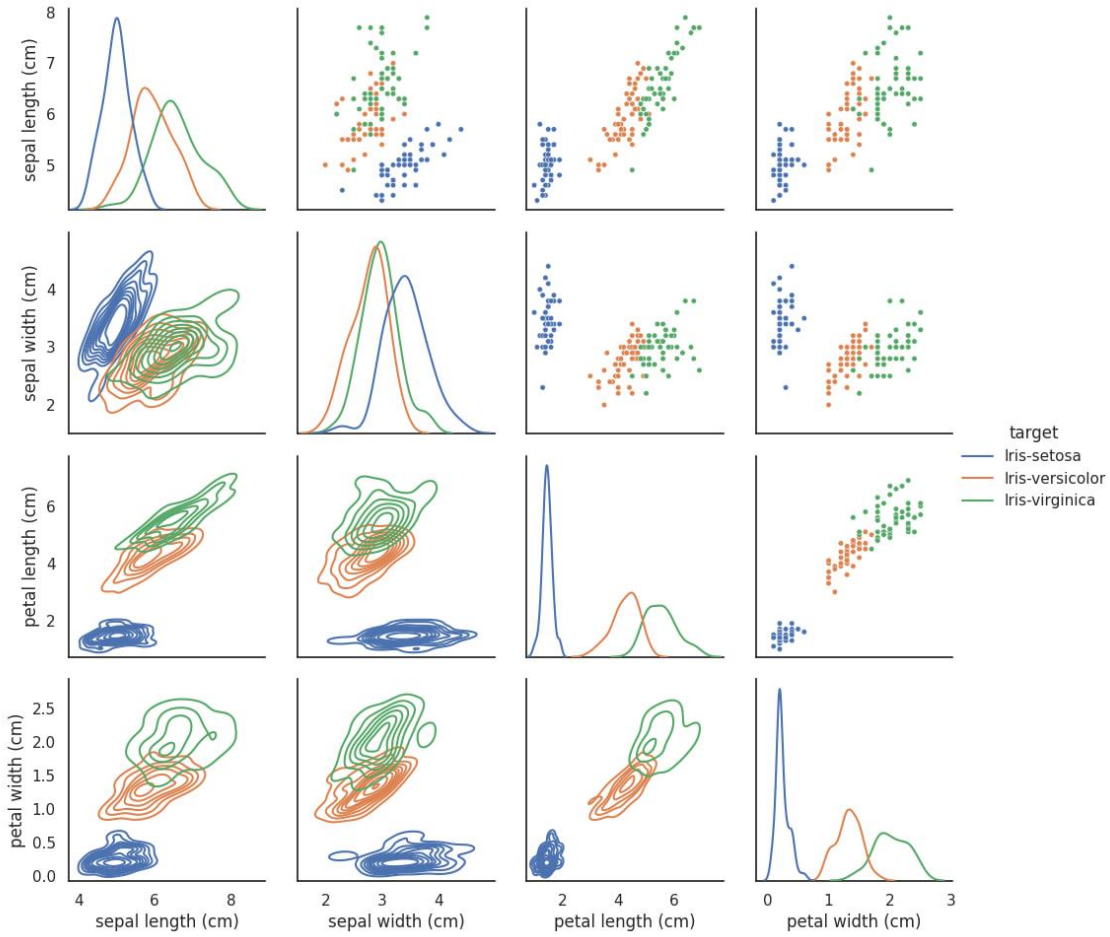

Рисунок 3 - графики распределения характеристик набора данных по классам

Рассмотрим, какие выводы можно сделать по полученным графикам.

В Таблица 1 представлены примерные оценки значений признаков, которые можно получить по графикам ядерной оценки плотности признаков. В строке разброс представлены примерные границы для конкретного признака конкретного сорта. В строке пик представлены пиковые значения ядерной оценки для конкретного признака конкретного сорта.

Таблица 1

		Iris-setosa	Iris-versicolor	Iris-virginica
Sepal length (cm)	разброс	4-6	4.2-6.8	4.3-8
	пик	4.5	4.8	6.1
Sepal width (cm)	разброс	2-4.5	1.8-3.5	1.8-4
	пик	3.4	2.8	2.9
Petal length (cm)	разброс	0.5-2	2-4.9	4-7
	пик	1.5	4.1	5
Petal width (cm)	разброс	0-0.9	0.9-2	1-3
	пик	0.2	1.4	2

По графикам можно сказать, что Iris-versicolor в среднем имеет более мелкие характеристики, чем и Iris-virginica, но их допустимые значения имеют большое пересечение, а пики на графиках находятся вблизи друг от друга. Поэтому только по заданным характеристикам может быть сложно однозначно различить эти два сорта.

Iris-setosa имеет наименьшее из всех значений для petal (length и width), при этом не имеет пересечений в полученных значениях по этим параметрам с Iris-versicolor и Iris-virginica. Sepal length также имеет в среднем меньшее значение, однако имеется пересечение с заданными характеристиками других сортов. Sepal width имеет в среднем чуть больше значение, чем у остальных сортов, однако пересечение допустимых значений значительное.

По диаграмме рассеяния мы можем наблюдать оценки разброса значений, которые были ранее получены по графикам ядерной оценки, и тенденции к зависимости одних признаков от других. Оценки разброса подтверждают данные, отражённые в Таблица 1.

Рассмотрим более подробно попарную зависимость признаков. Например, для Iris-versicolor и Iris-virginica заметна корреляция между petal width и petal length и между petal length и sepal length. С возрастанием одного возрастает другое. Для Iris-setosa явной зависимости для этих признаков не наблюдается. Возможно, из-за того, что и petal width, и petal length у этих цветков имеют небольшой диапазон, в котором сложно оценить зависимость.

Также прямая зависимость видна между sepal width, и sepal length у Iris-setosa. У Iris-versicolor и Iris-virginica зависимость также есть, но менее явная.

Далее рассмотрим график двумерной оценки плотности для каждых признаков. По этим графикам можно оценить разброс значений каждого из признаков, оценить зависимость признаков друг от друга и оценить плотность этих зависимостей.

В целом эти графики подтверждают данные графиков, рассмотренных выше, как по разбросу, так и по зависимости. Особенно чётко видна для Iris-versicolor между petal width и petal length и между petal length и sepal length. Зависимость у Iris-virginica между этими признаками более слабая. Также явно видна зависимость между sepal width, и sepal length у Iris-setosa. Зависимость между petal width и petal length у Iris-setosa не видна.

Были построены гистограммы распределения для каждого признака с различным количеством столбцов: 5 (Рисунок 4), 10 (Рисунок 5), 15 (Рисунок 6), 20 (Рисунок 7), 30 (Рисунок 8).

Количество столбцов менялось с помощью параметра bins функции sns.histplot.

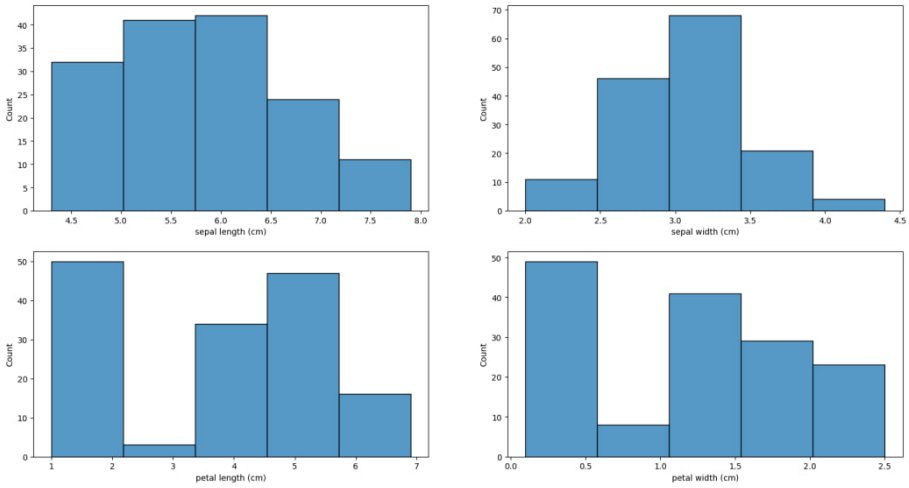

Рисунок 4 - гистограммы распределения для каждого признака с 5-ю столбцами

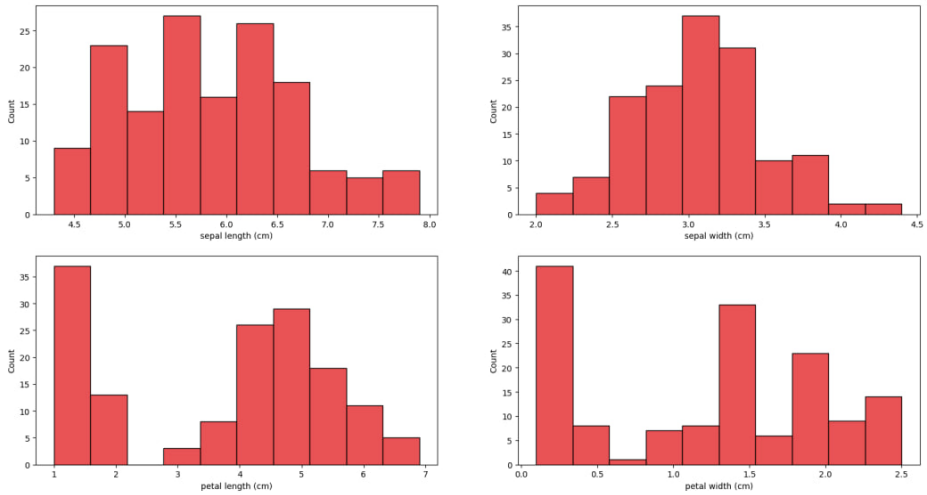

Рисунок 5 - гистограммы распределения для каждого признака с 10-ю столбцами

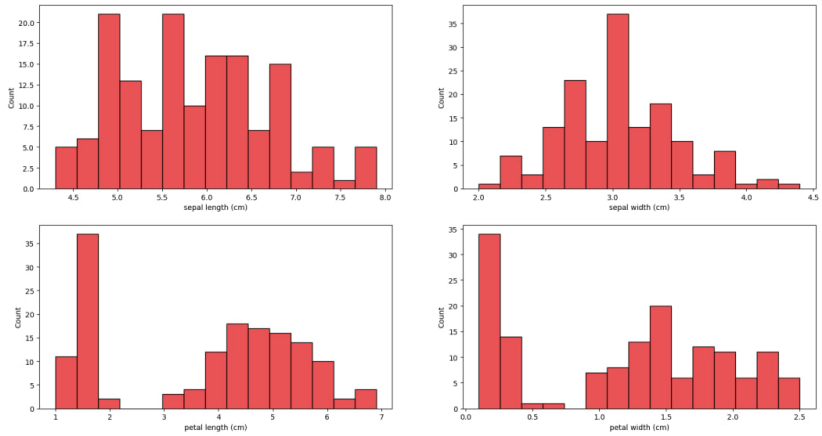

Рисунок 6 - гистограммы распределения для каждого признака с 15-ю столбцами

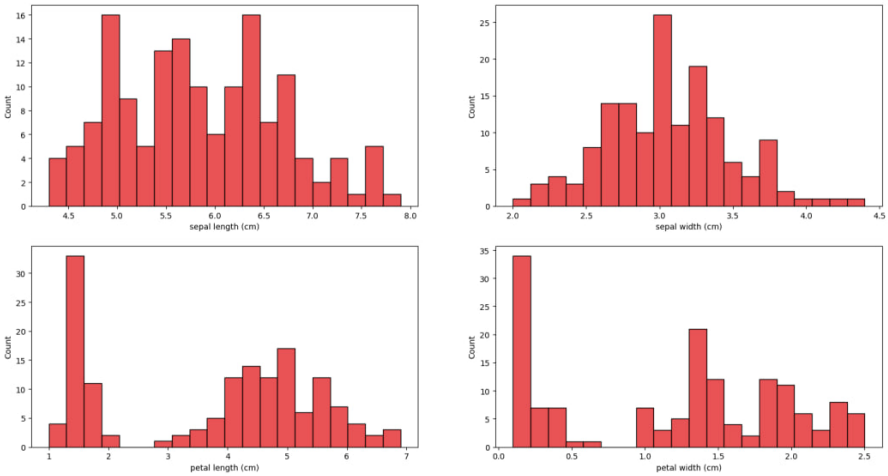

Рисунок 7 - гистограммы распределения для каждого признака с 20-ю столбцами

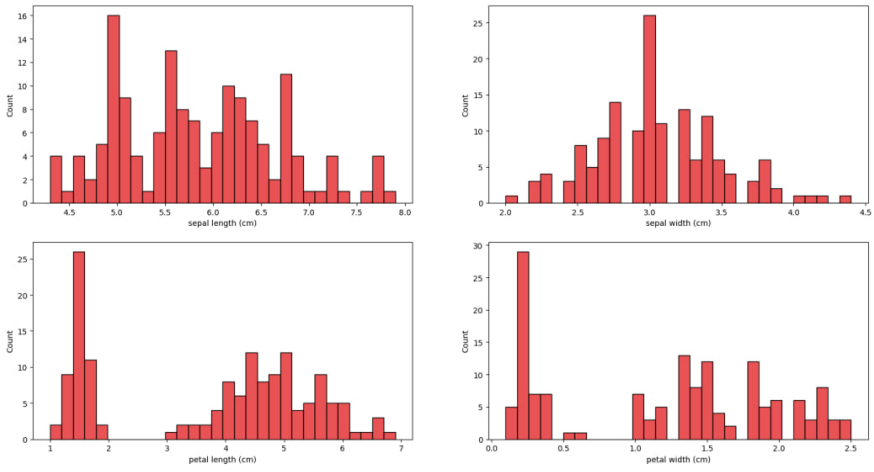

Рисунок 8 - гистограммы распределения для каждого признака с 30-ю столбцами

На мой взгляд, график с 5-ю столбцами недостаточно описывает форму распределения, так как распределение слишком большое. Распределение с 10-ю столбцами описывает форму намного лучше, однако недостаточно подробно видно распределение значений в начале гистограмм на нижних двух графиках. График с 15-ю столбцами мне кажется оптимальным, так как он показывает основные тенденции распределения. Графики с 20-ю и 30-ю столбцами содержат большое количество перепадов в высоте столбцов из-за слишком мелкого разбиения для данного размера выборки. На Рисунок 12 видно, что во многих местах резкого перепада высоты столбцов на графиках с 20-ю и 30-ю столбцами на самом деле нет такой резкой просадки количества значений, попавших в данную область, значения изменяются плавно. В дальнейшем для количества столбцов в гистограммах будет зафиксировано значение 15.

С помощью параметра hue функции sns.histplot на каждой гистограмме было сделано разделение по цвету согласно классу.

На Рисунок 9 представлен график с режимом, когда гистограммы суммируются (с помощью параметра multiple='layer' функции sns.histplot). На Рисунок 10 представлен режим, когда гистограммы пересекаются (С помощью параметра multiple='stack функции sns.histplot). Далее на графиках будет использован режим с пересечением.

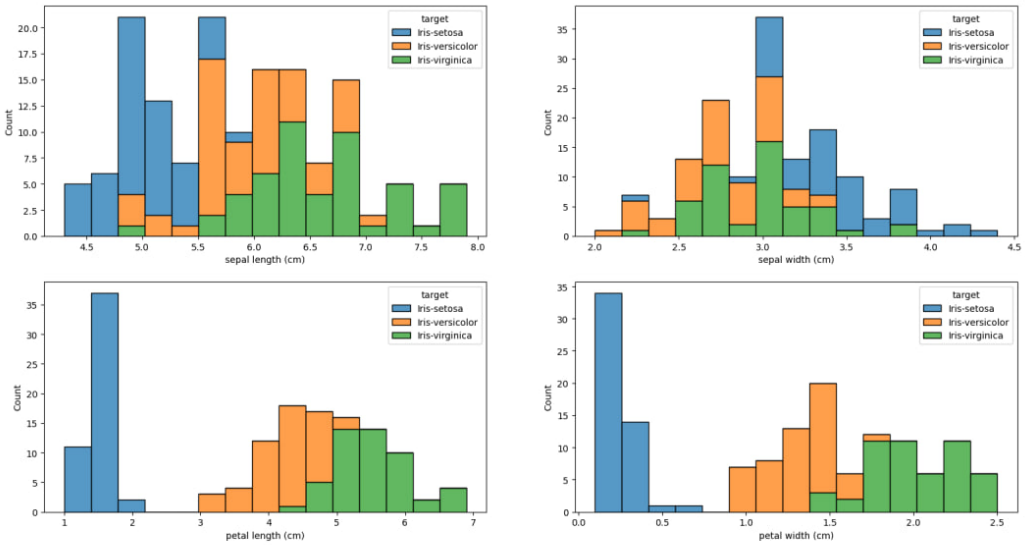

Рисунок 9 – разделение по цвету с режимом суммирования

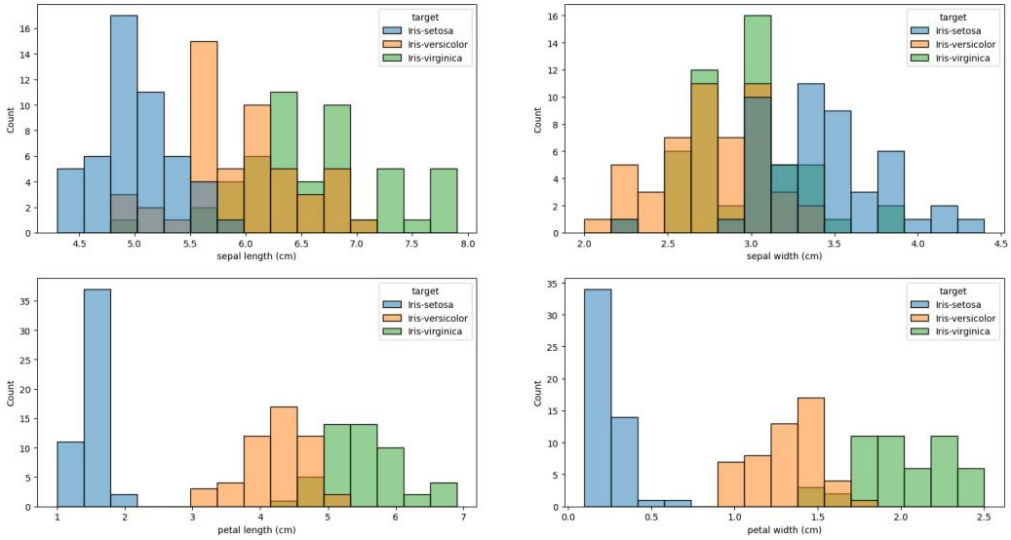

Рисунок 10 – разделение по цвету с режимом пересечения

С помощью заданного параметра element='step' функции sns.histplot на гистограммах столбцы были заменены на ступеньки (Рисунок 11).

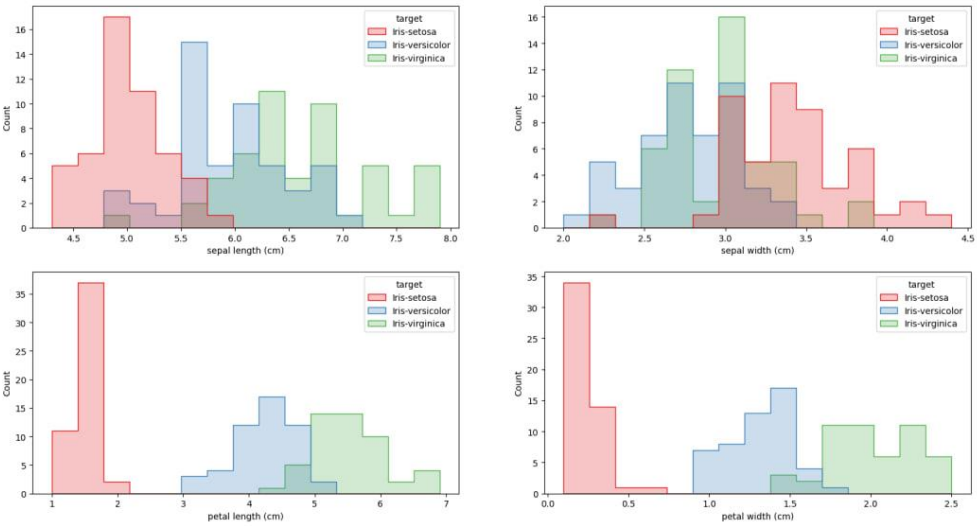

Рисунок 11 – ступенчатые гистограммы

С помощью заданного параметра kde=True функции sns.histplot на гистограммах был добавлен график ядерной оценки плотности (Рисунок 12).

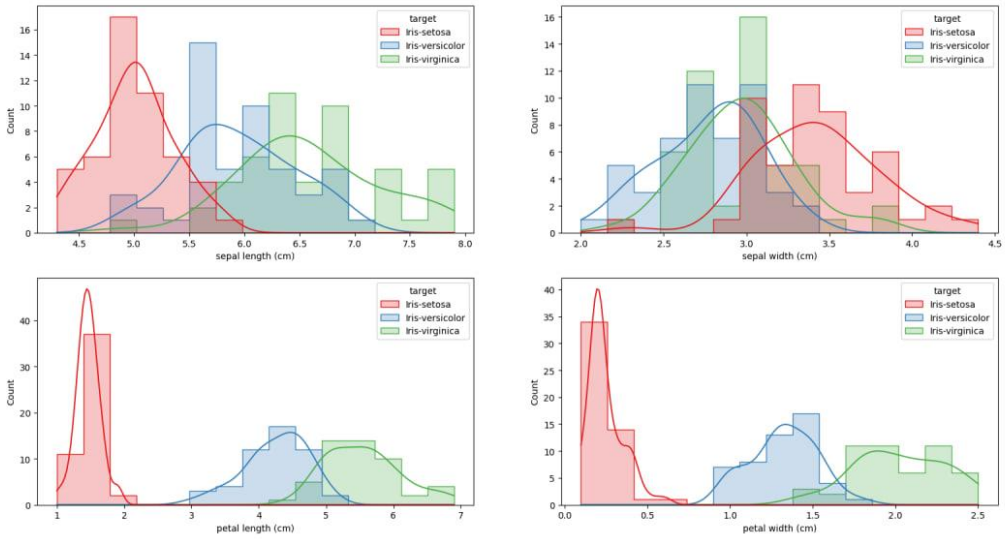

Рисунок 12 – гистограммы с графиком ядерной оценки плотности

Итоговая функция show_histplot для отрисовки гистограмм представлена на Листинг 1..

Листинг 1.6

```
def show_histplot(df, hue, bins_in_histogram=15, row=2, column=2,
multiple='layer', kde=False):
    column_names = df.columns.values
    fig, axes = plt.subplots(row, column, figsize=(18, 10))
    for i in range(0, min(column_names.size, row * column)):
        sns.histplot(df,
                     ax=axes[i // column, i % column],
                     x=df[column_names[i]],
                     element='step',
                     hue=hue,
                     kde=kde,
                     bins=bins_in_histogram,
                     multiple=multiple)
    sns.set_palette("Set1")
    plt.show()
```

2. Изучение набора данных iris.csv с использованием NumPy

С помощью функции get_a (Листинг 2.1) были загружены данные из файла как массив NumPy.

Листинг 2.1

```
def get_a(file_name, delete_first=True, delete_last=False):
    a = np.loadtxt(open(file_name, "rb"), delimiter=",", skiprows=1)
    if delete_first:
        a = np.delete(a, 0, 1)
    if delete_last:
        a = np.delete(a, a.shape[1] - 1, 1)

    return a
```

С помощью функции print_a_head (Листинг 2.2) были выведены первые 10 наблюдений набора данных.

Листинг 2.2

```
def print_a_head(a, n=10):
    print("\n============== head ==============")
    print(iris_a[:n])
```

С помощью функции print_a_description (Листинг 2.3) были рассчитаны характеристики, полученные методом describe в п. 1.3, с использованием методов NumPy.

Листинг 2.3

```
def print_a_description(a):
    print("\n============= describe ===============")

    print('count: ', np.full(a.shape[1], a.shape[0]))
    print('mean: ', a.mean(axis=0))
    print('std: ', a.std(axis=0))
    print('min: ', a.min(axis=0))
    print('25%: ', np.percentile(a, 25, axis=0))
    print('50%: ', np.percentile(a, 50, axis=0))
    print('75%: ', np.percentile(a, 75, axis=0))
    print('max: ', a.max(axis=0))
```

Вывод первых 10 наблюдений набора данных функцией print_a_head и вывод рассчитанных характеристик функцией print_a_description представлен на Рисунок 13.

```
============== head ==============
[[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. ]]

============= describe ===============
count:  [150 150 150 150 150]
mean:  [5.84333333 3.05733333 3.758      1.19933333 1.        ]
std:  [0.82530129 0.43441097 1.75940407 0.75969263 0.81649658]
min:  [4.3 2.  1.  0.1 0. ]
25%:  [5.1 2.8 1.6 0.3 0. ]
50%:  [5.8  3.   4.35 1.3  1.  ]
75%:  [6.4 3.3 5.1 1.8 2. ]
max:  [7.9 4.4 6.9 2.5 2. ]
```

Рисунок 13 - вывод первых 10 наблюдений набора данных и характеристик

3. Изучение набора данных вашего варианта

С помощью методов из пункта 1 были рассмотрены данных варианта 1.

На Рисунок 14 показан вывод первых 5 наблюдений набора данных функцией head и вывод рассчитанных характеристик функцией describe.

```
============== head ==============
      first    second     third    fourth  label
0  0.342494 -4.170293 -0.427134  5.285126      0
1  2.506861  1.536588  3.311512  3.455109      1
2 -0.723178 -2.866860 -1.778395  3.604657      0
3  2.413882  3.921118  2.923352  2.768869      1
4 -0.377760 -2.471150 -2.796839  3.968686      0

============= describe ===============
            first      second       third      fourth       label
count  200.000000  200.000000  200.000000  200.000000  200.000000
mean    -0.079295    0.056456   -0.078393    3.019419    0.500000
std      3.199506    3.026384    3.285926    1.055298    0.501255
min     -6.200980   -4.808300   -6.728692    0.523930    0.000000
25%     -3.150403   -2.655759   -3.176466    2.350288    0.000000
50%      0.708355   -0.549150   -0.120588    2.986238    0.500000
75%      2.775412    2.894577    2.872863    3.763799    1.000000
max      5.443596    5.361941    6.107679    6.353969    1.000000
```

Рисунок 14 - вывод первых 10 наблюдений набора данных и характеристик

На Рисунок 15 показаны графики ядерной оценки плотности каждого признака (кроме признака класса), диаграмма рассеяния и двумерная ядерная оценка плотности для каждых признаков.

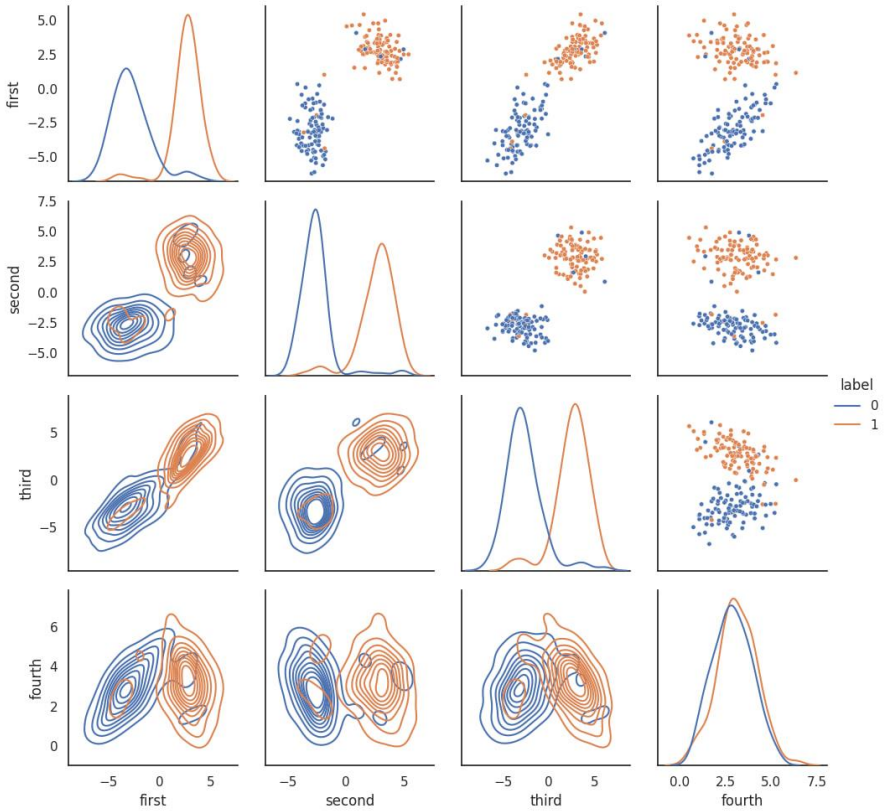

Рисунок 15 - графики распределения характеристик набора данных по классам

Рассмотрим, какие выводы можно сделать по полученным графикам.

В Таблица 2 представлены примерные оценки значений признаков, которые можно получить по графикам ядерной оценки плотности признаков. В строке разброс представлены примерные границы для конкретного признака конкретного сорта. В строке пик представлены пиковые значения ядерной оценки для конкретного признака конкретного сорта.

Таблица 2

		0	1
first	разброс	-7 - 5	-5 - 7
	пик	-4	4
second	разброс	-6 - 6	-5 - 6
	пик	-3	3
third	разброс	-10 - 10	-5 - 7
	пик	-4	3
fouth	разброс	0-7.5	0-7.5
	пик	3	3

По графикам можно сказать, что значения класса 0 в среднем имеют меньшие характеристики, чем значения класса 1. По первому, второму и третьему признаку область пересечения довольно маленькая, и в большинстве случаев по этим признакам можно определить, к какому из классов объект относится с большей вероятностью. Однако и в первом, в во втором классе имеются данные, образующие небольшой выброс в области пересечения с другим классом.

Распределение значения четвёртого признака очень схоже у двух классов, поэтому различить по этим признакам объекты классов не получится.

По диаграмме рассеяния мы можем также явно наблюдать скопления точек каждого класса в определённой области, но эти скопления содержат небольшое количество включений точек другого класса.

Рассмотрим более подробно попарную зависимость признаков. Можно предположить, что четвёртый и третий признаки растут совместно с первым у 0 класса, а у класса четвёртый признак уменьшается с ростом третьего.

Графики двумерной оценки плотности в целом подтверждают данные графиков, рассмотренных выше. На них также присутствуют обособленные от основного скопления точек группы, включённые в противоположный класс.

На Рисунок 16 показана гистограмма с ядерной оценкой плотности признаков. Графики соответствуют графикам ядерной оценки плотности признаков, полученных выше. По гистограммам можно сделать выводы о количестве данных в том или ином промежутке значений по каждому признаку.

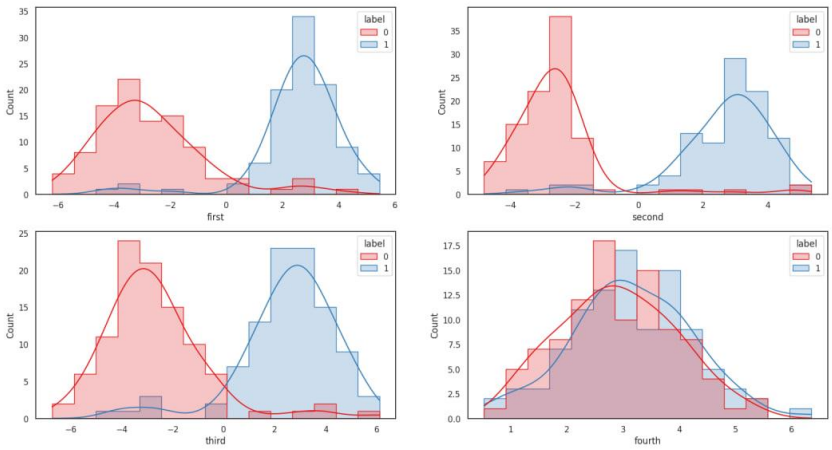

Рисунок 16 – гистограммы с графиком ядерной оценки плотности

4. Преобразование данных

С помощью функции print_encoders (Листинг 4.1) были выведены результаты различных способов кодирования меток класса: LabelEncoder и OneHotEncoder.

Листинг 4.1

```
def print_encoders(df_labels):
    le = LabelEncoder()
    le.fit(df_labels)
    print("LabelEncoder transformation: ", le.classes_, " -> ",
le.transform(le.classes_))

    ohe = OneHotEncoder(sparse_output=False)
    ohe.fit(np.array(df_labels).reshape(-1, 1))
    print("OneHotEncoder transformation: ", ohe.categories_, " ->\n",
          ohe.transform(np.array(ohe.categories_[0].reshape(-1, 1))))
```

На Рисунок 17 показан вывод с демонстрацией различных способов кодирования метод класса.

```
LabelEncoder transformation:  ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']  ->  [0 1 2]
OneHotEncoder transformation:  [array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)]  ->
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Рисунок 17 – кодирование с помощью LabelEncoder и OneHotEncoder

С помощью функций, показанных на Листинг 4.2, применены StandardScaler, MinMaxScaler, MaxAbsScaler и RobustScaler. Гистограммы с результатами по каждому признаку без разделения по классам представлены на Рисунок 18. По результату видно, что различиями между преобразованиями данных является коэффициент нормировки. Высота столбцов не меняется.

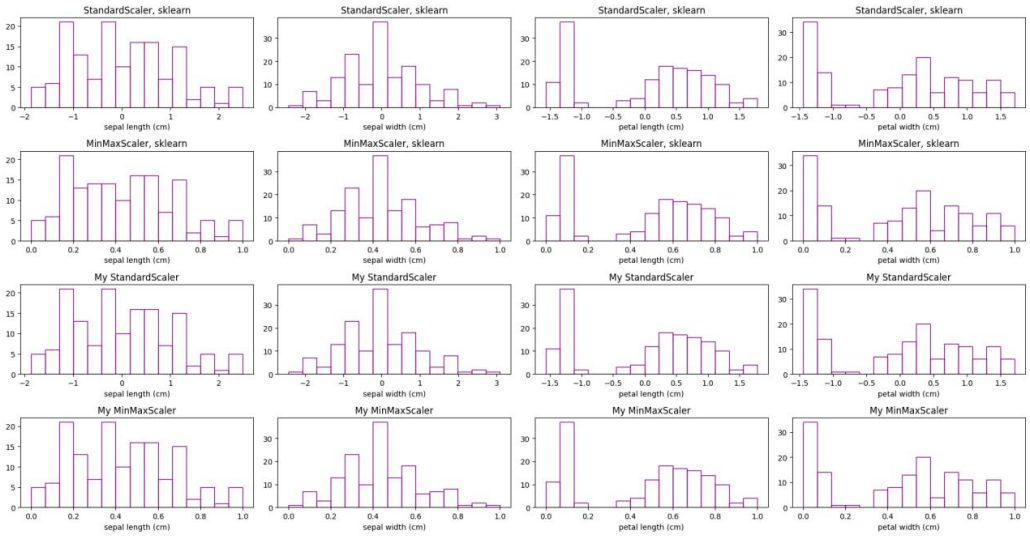

Рисунок 18 – гистограммы с StandardScaler, MinMaxScaler, MaxAbsScaler и RobustScaler.

Листинг 4.2

```
def show_hist(a, name, axs, n, columns_names):
    for i in range(4):
        axs[n][i].hist(a[:, i], bins=15, fill=False, ec='purple')
        axs[n][i].set_title(name)
        axs[n][i].set_xlabel(columns_names[i])


def show_scaler_sklearn(scaler, scaler_name, a_features, axs, n,
columns_names):
    scaler.fit(a_features)
    a = scaler.transform(a_features)
    show_hist(a, scaler_name, axs, n, columns_names)


def show_scalers_sklearn(a_features, columns_names):
    ig, axs = plt.subplots(4, 4, figsize=(18, 10))

    show_scaler_sklearn(StandardScaler(), "StandardScaler", a_features,
axs, 0, columns_names)
    show_scaler_sklearn(MinMaxScaler(), "MinMaxScaler", a_features, axs, 1,
columns_names)
    show_scaler_sklearn(MaxAbsScaler(), "MaxAbsScaler", a_features, axs, 2,
columns_names)
    show_scaler_sklearn(RobustScaler(), "RobustScaler", a_features, axs, 3,
columns_names)
    plt.tight_layout()
    plt.show()
```

Были реализованы StandardScaler и MinMaxScaler с использованием библиотеки NumPy. Было проведено сравнение результатов между моей реализацией и реализацией из Sklearn (Рисунок 19). Было рассчитано минимальное, максимальное, среднее значение и дисперсии после преобразования (Рисунок 20). На По результатам вывода на Рисунок 19 и Рисунок 20 видно, что с помощью самостоятельной реализации удалось построить те же графики и получить те же значения максимального, минимального, среднего значения и дисперсии, что и с помощью реализации Sklearn.

Листинг 4.3 представлен код, запускающий сравнение результатов.

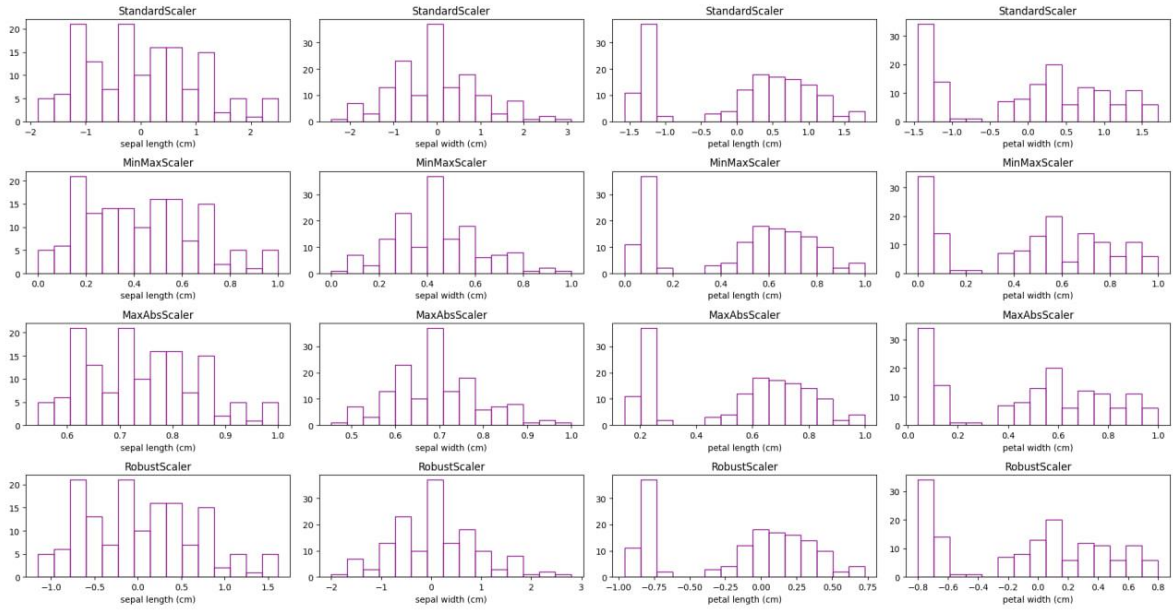

Рисунок 19 - сравнение результатов собственной реализацией и реализацией из Sklearn

```
StandardScaler, sklearn  min:  [-1.87002413 -2.43394714 -1.56757623 -1.44707648]
StandardScaler, sklearn  max:  [2.4920192  3.09077525 1.78583195 1.71209594]
StandardScaler, sklearn mean:  [-4.73695157e-16 -7.81597009e-16 -4.26325641e-16 -4.73695157e-16]
StandardScaler, sklearn std:  [1. 1. 1. 1.]

MinMaxScaler, sklearn  min:  [0. 0. 0. 0.]
MinMaxScaler, sklearn  max:  [1. 1. 1. 1.]
MinMaxScaler, sklearn mean:  [0.4287037  0.44055556 0.46745763 0.45805556]
MinMaxScaler, sklearn std:  [0.22925036 0.18100457 0.29820408 0.31653859]

My StandardScaler  min:  [-1.87002413 -2.43394714 -1.56757623 -1.44707648]
My StandardScaler  max:  [2.4920192  3.09077525 1.78583195 1.71209594]
My StandardScaler mean:  [-4.73695157e-16 -7.81597009e-16 -4.26325641e-16 -4.73695157e-16]
My StandardScaler std:  [1. 1. 1. 1.]

My MinMaxScaler  min:  [0. 0. 0. 0.]
My MinMaxScaler  max:  [1. 1. 1. 1.]
My MinMaxScaler mean:  [0.4287037  0.44055556 0.46745763 0.45805556]
My MinMaxScaler std:  [0.22925036 0.18100457 0.29820408 0.31653859]
```

Рисунок 20 -минимальное, максимальное, среднее значение и дисперсии после преобразования данных

По результатам вывода на Рисунок 19 и Рисунок 20 видно, что с помощью самостоятельной реализации удалось построить те же графики и получить те же значения максимального, минимального, среднего значения и дисперсии, что и с помощью реализации Sklearn.

Листинг 4.3

```
def show_my_scalers(a_features, columns_names):
    ig, axs = plt.subplots(4, 4, figsize=(18, 10))
```

```
a = StandardScaler().fit(a_features).transform(a_features)
print_metrics("StandardScaler, sklearn", a)
show_hist(a, "StandardScaler, sklearn", axs, 0, columns_names)

a = MinMaxScaler().fit(a_features).transform(a_features)
print_metrics("MinMaxScaler, sklearn", a)
show_hist(a, "MinMaxScaler, sklearn", axs, 1, columns_names)

a = get_my_std_scaled_a(a_features)
print_metrics("My StandardScaler", a)
show_hist(a, "My StandardScaler", axs, 2, columns_names)

a = get_my_min_max_scaled_a(a_features)
print_metrics("My MinMaxScaler", a)
show_hist(a, "My MinMaxScaler", axs, 3, columns_names)
plt.tight_layout()
plt.show()
```

С помощью функции get_filtered_df (Листинг 4.4) был получен датафрейм, 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%. Результат вывода представлен на Рисунок 21.

	sepal length (cm)	petal length (cm)	target
50	7.0	4.7	Iris-versicolor
51	6.4	4.5	Iris-versicolor
52	6.9	4.9	Iris-versicolor
54	6.5	4.6	Iris-versicolor
55	5.7	4.5	Iris-versicolor
..	...	...	...
143	6.8	5.9	Iris-virginica
144	6.7	5.7	Iris-virginica
145	6.7	5.2	Iris-virginica
147	6.5	5.2	Iris-virginica
149	5.9	5.1	Iris-virginica

Рисунок 21 – вывод отфильтрованных данных

Листинг 4.4

```
def get_filtered_df(df):
    p_25 = df['sepal width (cm)'].quantile(0.25)
    p_75 = df['sepal width (cm)'].quantile(0.75)
    print(p_25, p_75)
    df = df[(df['target'].isin(['Iris-versicolor', 'Iris-virginica'])) &
(p_25 <= df['sepal width (cm)']) & (
                df['sepal width (cm)'] <= p_75)]
    return df[['sepal length (cm)', 'petal length (cm)', 'target']]
```

5. Понижение размерности

Для набора данных iris.csv было применено понижение размерности до 2, используя PCA и TSNE из Sklearn с помощью функций, представленных на Листинг 5.1. Для каждого из результатов была построена диаграмма рассеяния с выделением разным цветом наблюдений разных классов (Рисунок 22 и Рисунок 23).

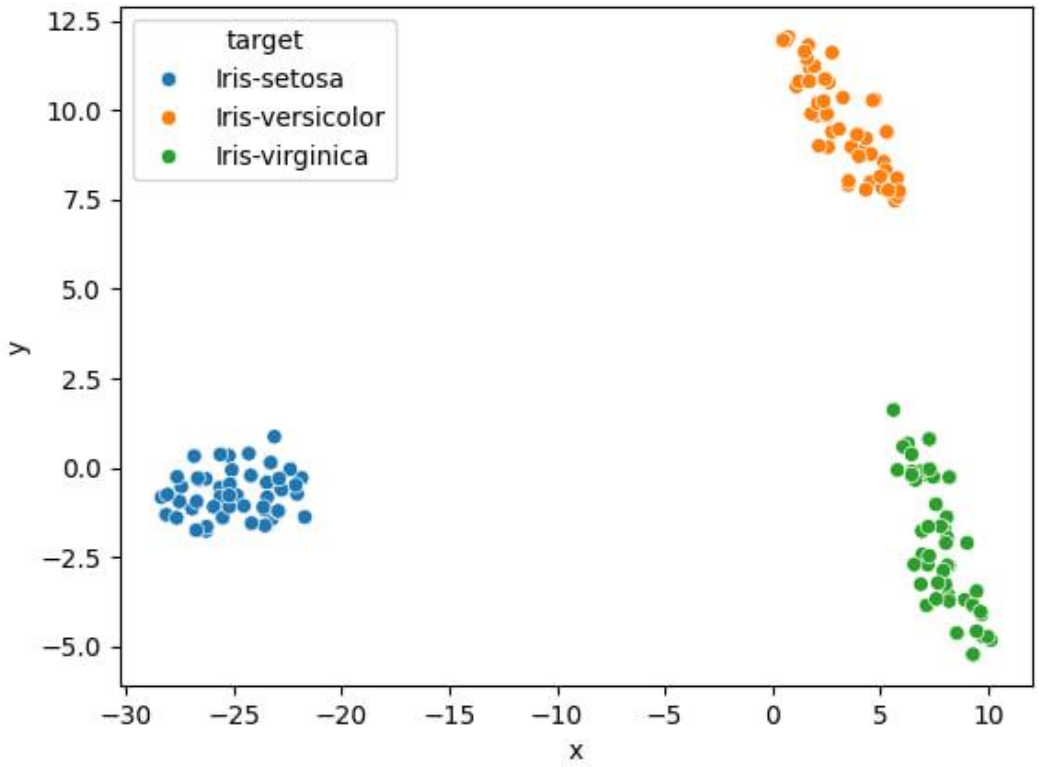

Рисунок 22 - понижение размерности до 2, используя PCA

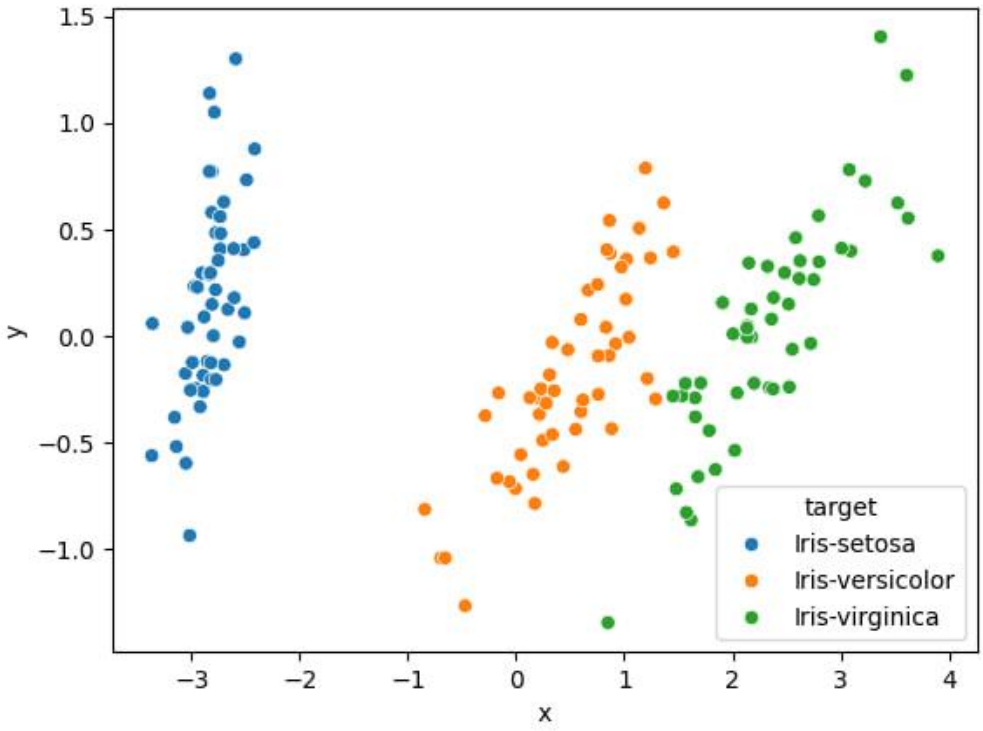

Рисунок 23 - понижение размерности до 2, используя TSNE

По графикам видно, что с помощью PCA удалось более чётко разделить классы на обособленные группы, в то время как с помощью TSNE получившиеся группы более разреженные и находятся на меньшем расстоянии друг от друга.

С другой стороны, с помощью TSNE был построен график, похожий на графики рассеяния, построенные на данных без понижения размерности (Рисунок 3). График PCA не сохранил структуру данных.

Листинг 5.1

```
def get_reduced_dimensionality_df(df, transformer):
    a_dim_2 = transformer.fit_transform(df)
    return pd.DataFrame(
        data=a_dim_2,
        columns=['x', 'y'])


def show_reduced_dimensionality(df, transformer):
    df_dim_2 = get_reduced_dimensionality_df(df, transformer)

    class_mapping = {
        0: 'Iris-setosa',
        1: 'Iris-versicolor',
        2: 'Iris-virginica'
    }
    x = replace_df_column(df, 'target', class_mapping)
    df_dim_2['target'] = x['target']
```

```
    sns.scatterplot(
        x='x',
        y='y',
        data=df_dim_2,
        hue='target',
        legend=True
    )
    plt.show()


def show_PCA(df):
    show_reduced_dimensionality(df, PCA(n_components=2))


def show_TSNE(df):
    show_reduced_dimensionality(df, TSNE(n_components=2))
```

Заключение.

В ходе лабораторной работы были изучены наборы данных с использованием библиотек Pandas, NumPy. Были построены графики с помощью библиотек Seaborn и Matplotlib.

Были построены графики ядерной оценки плотности каждого признака, диаграммы рассеяния и двумерные ядерные оценки плотности для каждых признаков.

Были построены гистограммы распределения для каждого признака. Данные с графиков оценки плотности и гистограмм показали похожие результаты.

Было выполнено преобразование данных с помощью реализации Sklearn и с помощью самостоятельной реализации. Результаты преобразований с помощью двух реализаций совпали.

Было произведено понижение размерности данных с помощью методов PCA и TSNE. Метод PCA показал более чёткое разделение данных, а метод TSNE сохранил структуру данных, схожую с диаграммами рассеяния, построенными ранее.