ЛАБОРАТОРНА РОБОТА № 2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати

Хід роботи

Завдання 1. Класифікація за допомогою машин опорних векторів

Серед вхідних ознак можливо точно охарактеризувати наступні ознаки:

- Вік – чисельна
- Форма працевлаштування (державне, приватне тощо) – категоріальна
- Рівень освіти – категоріальна
- Досвід роботи – чисельна
- Сімейний стан – категоріальна
- Тип працевлаштування (тип роботи) – категоріальна
- Поточна роль у сім'ї – категоріальна
- Колір шкіри – категоріальна
- Стать – категоріальна
- Країна проживання - категоріальна

Решта ознак є чисельними, охарактеризувати їх роль та назвати неможливо через відсутність назв колонок у текстовому файлі з даними та можливих підказок у самих даних.

Лістинг коду файлу Task_1.py:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model selection import train test split, cross val score
```

```
input_file = 'income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

with open(input file, 'r') as f:
    for line in f.readlines():
        if count class1 >= max datapoints and count class2 >= max datapoints:
            break
        if '?' in line:
            continue

        data = line[:-1].split(', ')
        if data[-1] == "<=50K" and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1

        if data[-1] == ">50K" and count_class2 < max_datapoints:
            X.append(data)
            count class2 += 1

X = np.array(X)

label encoder = []
X encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X encoded[:, i] = label encoder[-1].fit transform(X[:, i])

X = X encoded[:, :-1].astype(int)
y = X encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(LinearSVC(random state=0))
X train, X test, y train, y test = train test split(X, y, test size=0.2, random state=5)
classifier.fit(X train, y train)
y_test_pred = classifier.predict(X_test)

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision = cross val score(classifier, X, y, scoring='precision weighted', cv=3)
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
f1 = cross val score(classifier, X, y, scoring='f1 weighted', cv=3)
print("Accuracy: " + str(round(100*accuracy.mean(), 2)) + "%")
print("Precision: " + str(round(100*precision.mean(), 2)) + "%")
print("Recall: " + str(round(100*recall.mean(), 2)) + "%")
print("F1 score: " + str(round(100*f1.mean(), 2)) + "%")


def predict(input data):
    input data encoded = [-1] * len(input data)
    count = 0
    for index, item in enumerate(input data):
        if item.isdigit():
            input data encoded[index] = int(input data[index])
        else:
            input data encoded[index] =
label_encoder[count].transform([input_data[index]])[0]
            count += 1

    input_data_encoded = np.array(input_data_encoded)
    predicted class = classifier.predict([input data encoded])
    print("Input data:", input data)
    print("Predicted class:", label encoder[-1].inverse transform(predicted class)[0])


input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-
```

```
cleaners', 'Not-in-family', 'White',
              'Male', '0', '0', '40', 'United-States']
predict(input_data)
```

```
Accuracy: 62.84%
Precision: 75.88%
Recall: 62.64%
F1 score: 56.15%
Input data: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Predicted class: <=50K
```

Рис.2.1 – Проаналізовані якість, точність та повнота алгоритму, прогнозований клас вхідних даних

Завдання 2. Порівняння якості класифікаторів SVM з нелінійними ядрами

До порівняння буде взято поліноміальне, гаусове, сигмоїдальне ядра, для цього буде скопійовано попередній код та змінено рядок створення класифікатора, а саме змінено спосіб ініціалізації ядра.

Через велику кількість даних, час навчання для поліноміального ядра займає дуже значну кількість часу, а саме має квадратичну залежність від кількості даних. Для можливого отримання швидкого результату під час виконання роботи, було встановлено максимальний ліміт на кількість даних у 100 рядків. Для інших ядер було встановлено ліміт в 15000 рядків.

Лістинг змін для поліноміального ядра, файл Task_2_1.py:

```
max_datapoints = 100

with open(input_file, 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints or len(X) >=
max_datapoints:
            break

* code skipped *

classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8, random_state=0))
```

```
Accuracy: 76.02%
Precision: 64.96%
Recall: 76.02%
F1 score: 66.49%
Input data: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Predicted class: <=50K
```

Рис.2.2 – Результат роботи поліноміального ядра

Лістинг змін для гаусового ядра, файл Task_2_2.py:

```
classifier = OneVsOneClassifier(SVC(kernel='rbf'))
```

Рис.2.3 – Результат роботи гаусового ядра

Лістинг змін для сигмоїдального ядра, файл Task_2_3.py:

```
classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))
```

Рис.2.4 – Результат роботи сигмоїдального ядра

За даних умов нерівномірності використання даних, найкращий результат надає гаусове ядро. За використання всіх наявних даних, результати можуть бути іншими, проте для їх отримання необхідно надати дуже багато часу на навчання поліноміальному ядру.

Завдання 3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Для аналізу вхідних даних було створено окремий файл Task_3_data.py, лістинг файлу:

```
from sklearn.datasets import load_iris
iris_dataset = load_iris()

print(f"Ключі iris_dataset: \n{iris_dataset.keys()}")
print(iris_dataset['DESCR'][:193] + "\n...")

print(f"Назви відповідей: {iris_dataset['target_names']}")
print(f"Назви ознак: {iris_dataset['feature_names']}")
print(f"Тип даних: {type(iris_dataset['data'])}")
print(f"Розмір даних: {iris_dataset['data'].shape}")
print(f"Перші п'ять рядків даних:\n{iris_dataset['data'][:5]}")
print(f"Тип масиву відповідей: {type(iris_dataset['target'])}")
print(f"Розмір масиву відповідей: {iris_dataset['target'].shape}")
print(f"Відповіді:\n{iris_dataset['target']}")
```

```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
```

Рис.2.5 – Ключі словника вхідних даних та опис

```
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назви ознак: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип даних: <class 'numpy.ndarray'>
Розмір даних: (150, 4)
Перші п'ять рядків даних:
[[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]]
Тип масиву відповідей: <class 'numpy.ndarray'>
Розмір масиву відповідей: (150,)
Відповіді:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Рис.2.6 – Детальна інформація про відповіді, ознаки та дані

Для класифікації даних було створено файл Task_3_classify.py, лістинг файлу:

```
from pandas import read csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.metrics import classification report, confusion matrix, accuracy score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.svm import SVC

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)

print(dataset.shape)
print(dataset.head(20))
print(dataset.describe())
print(dataset.groupby('class').size())

# Діаграма розмаху
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

# Гістограми
dataset.hist()
plt.show()

# Матриця розсіювання
scatter matrix(dataset)
plt.show()

# Розділення набору даних на навчальний та тестовий
array = dataset.values
X = array[:, 0:4]
Y = array[:, 4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=0.20,
random state=1)

# Перевірка алгоритмів
models = [
    ('LR', LogisticRegression(solver='liblinear', multi class='ovr')),
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', KNeighborsClassifier()),
    ('CART', DecisionTreeClassifier()),
    ('NB', GaussianNB()),
    ('SVM', SVC(gamma='auto'))
]

# Оцінка кожного алгоритму
results = []
names = []

for name, model in models:
    kfold = StratifiedKFold(n splits=10, random state=1, shuffle=True)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
    results.append(cv results)
    names.append(name)
    print(f'{name}: {cv results.mean()} ({cv results.std()})')

# Порівняння алгоритмів
plt.boxplot(results, labels=names)
plt.title('Алгоритми порівняння')
plt.show()

# Побудова моделі на основі алгоритму SVM
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X validation)

# Оцінка моделі
print(accuracy score(Y validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification report(Y validation, predictions))

# Вивід результатів по власним даним
X new = [[5.0, 3.6, 1.3, 0.25], [5.9, 3.0, 5.1, 1.8], [6.3, 3.3, 6.0, 2.5], [5.8, 2.7, 5.1,
1.9], [5.1, 3.5, 1.4, 0.2]]
```

```
predictions = model.predict(X_new)
print(f"X_new: {X_new}\nPredictions: {predictions}")
```

```
(150, 5)
    sepal-length  sepal-width  petal-length  petal-width        class
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
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
```

Рис.2.7 – Розмір масиву даних та перші 20 записів

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Рис.2.8 – Характеристики даних, кількість по класам та тип даних

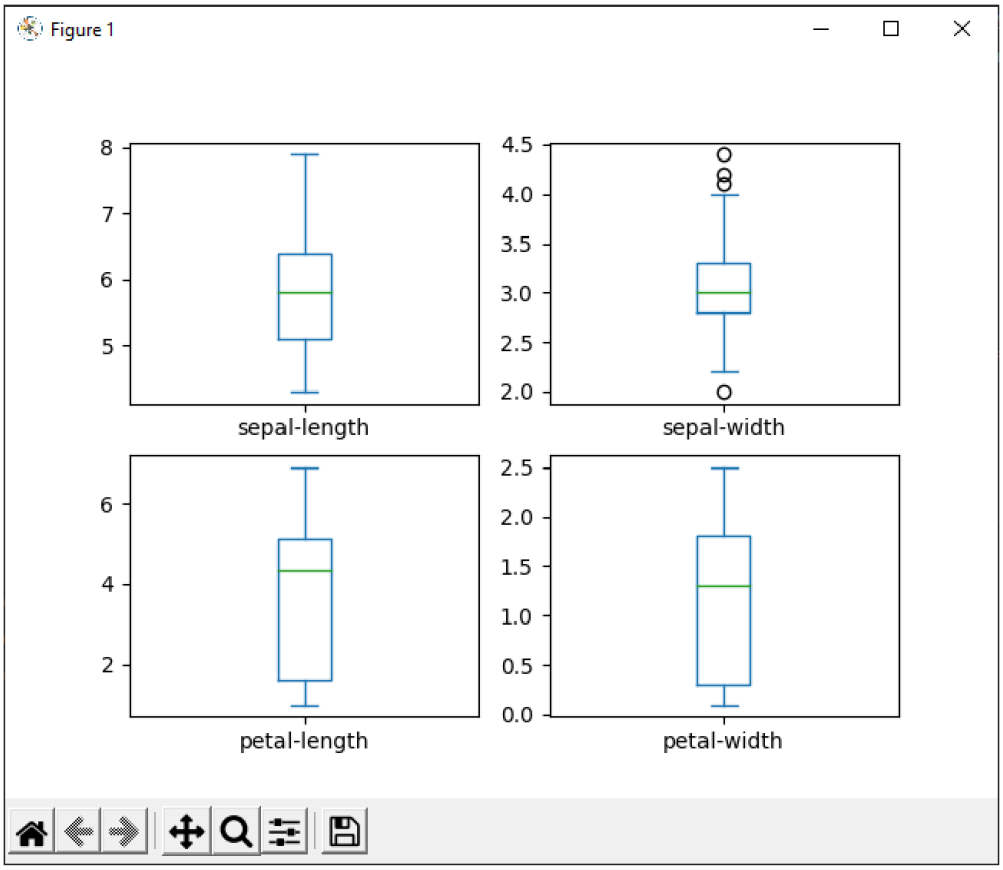

Рис.2.9 – Діаграма розмаху

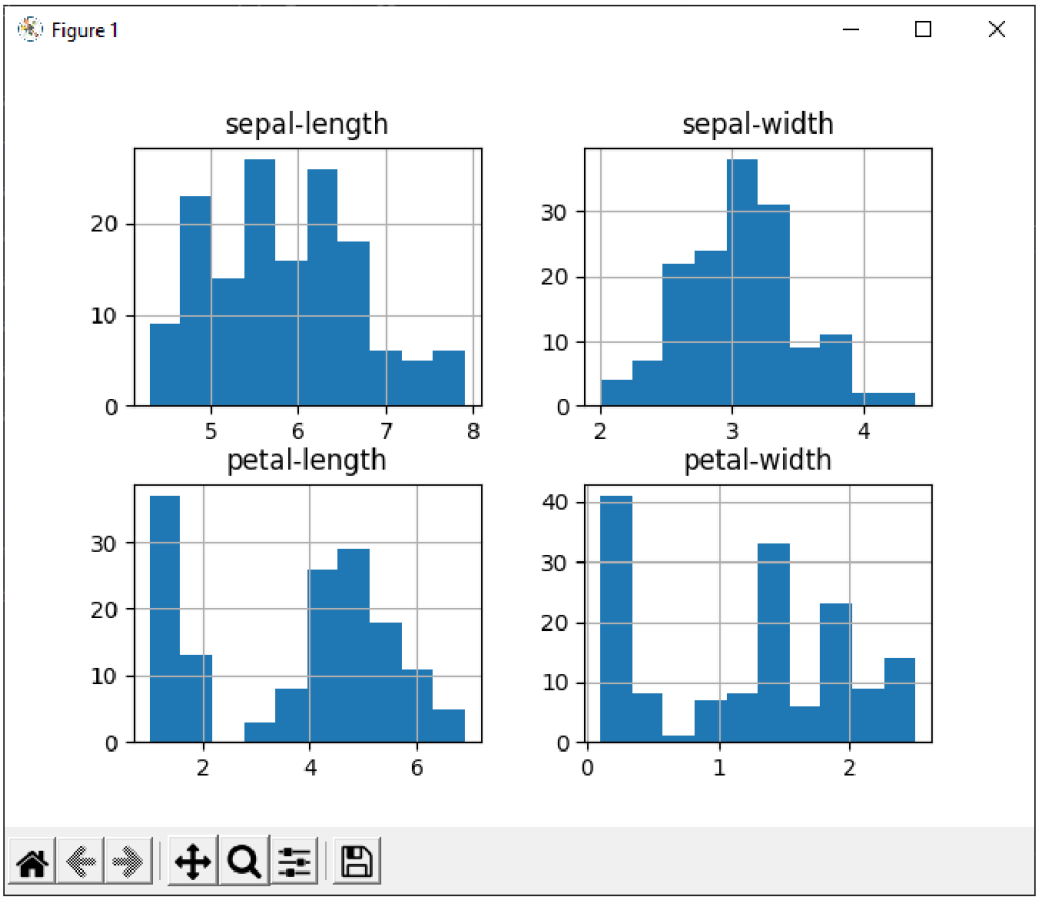

Рис.2.10 – Гістограма даних

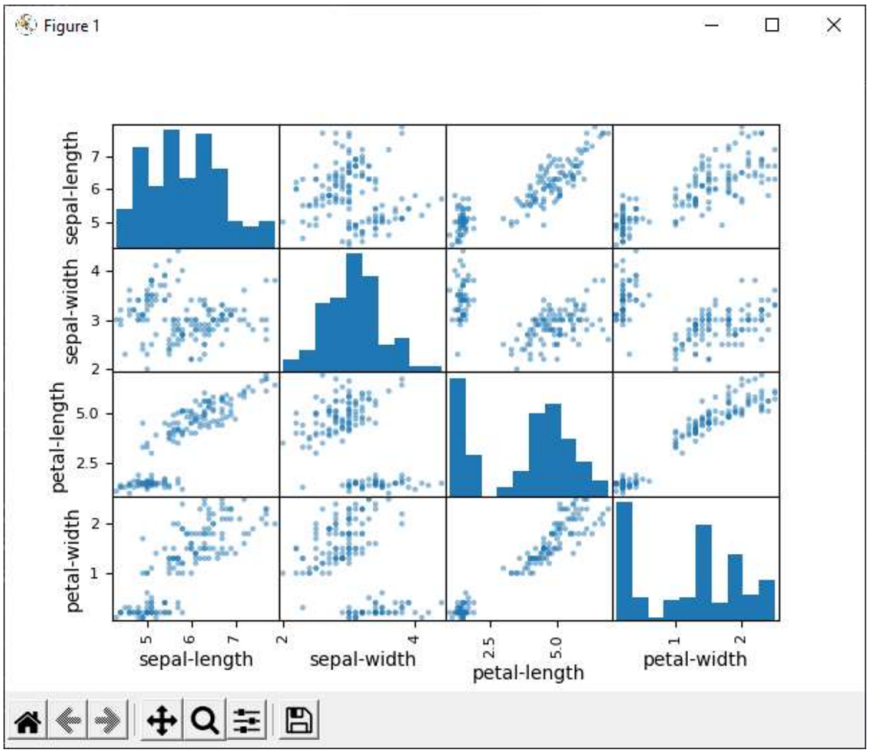

Рис.2.11 – Багатовимірна діаграма розсіювання

```
LR: 0.9416666666666667 (0.06508541396588878)
LDA: 0.975 (0.03818813079129868)
KNN: 0.9583333333333333 (0.04166666666666669)
CART: 0.9499999999999998 (0.04082482904638632)
NB: 0.95 (0.05527707983925667)
SVM: 0.9833333333333332 (0.03333333333333335)
```

Рис.2.12 – Отримані результати навчання моделей (лише accuracy)

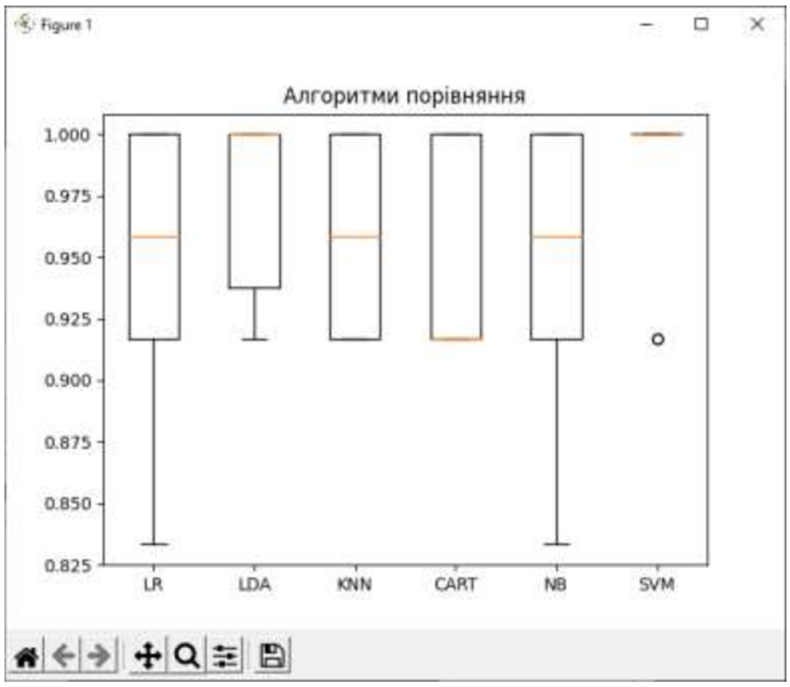

Рис.2.13 – Діаграма розмаху атрибутів вхідних даних

Найкращим за власною думкою є SVM через найкращі результати класифікації та якості роботи.

```
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

Рис.2.14 – Якість, матриця помилок та звіт по класифікації даних через SVC

```
X_new: [[5.0, 3.6, 1.3, 0.25], [5.9, 3.0, 5.1, 1.8], [6.3, 3.3, 6.0, 2.5], [5.8, 2.7, 5.1, 1.9], [5.1, 3.5, 1.4, 0.2]]
Predictions: ['Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa']
```

Рис.2.15 – Прогнозування класів власних даних

Висновок: під час виконання завдань лабораторної із використанням спеціалізованих бібліотек та мови програмування Python було досліджено різні методи класифікації даних та отримано навички їх порівнювати.

Протягом роботи було створено кілька файлів формату .py для кожного завдання для окремого порівняння роботи різних класифікаторів між собою та окремого відображення даних та їх обробки.

Для виконання робіт було використано функції бібліотек pandas, matplotlib, scikit-learn та відображено результати класифікації з характеристиками різних способів класифікації.