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:

Посилання на GitHub:

Завдання 2.1. Класифікація за допомогою машин опорних векторів (SVM)

Лістинг програми:

```
import numpy as np
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
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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.strip().split(', ')

        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        elif 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 in range(X.shape[1]):
    if X[0, i].isdigit():
        X_encoded[:, i] = X[:, i]
```

					ДУ «Житомирська політехніка».24.121.12.000 – Лр2		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Курач О.А.			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів 2
Перевір.		Іванов Д.А					
Керівник						ФІКТ Гр. ІПЗ-21-4	
Н. контр.							
Зав. каф.							

```
 else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, ran-
dom_state=5)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X_train, Y_train)

Y_pred = classifier.predict(X_test)

accuracy = accuracy_score(Y_test, Y_pred)
precision = precision_score(Y_test, Y_pred, average='weighted')
recall = recall_score(Y_test, Y_pred, average='weighted')
f1 = f1_score(Y_test, Y_pred, average='weighted')

print(f"Accuracy: {round(accuracy * 100, 2)}%")
print(f"Precision: {round(precision * 100, 2)}%")
print(f"Recall: {round(recall * 100, 2)}%")
print(f"F1 Score: {round(f1 * 100, 2)}%")
```

Результат :

```
C:\Users\user\AppData\Local\Programs\Python\Py
Accuracy: 71.43%
Precision: 51.02%
Recall: 71.43%
F1 Score: 59.52%
C:\Users\user\AppData\Local\Programs\Python\Py
  _warn_prf(average, modifier, f"{metric.capit

Process finished with exit code 0
```

Рис.1.1 Результат виконання програми

Зробіть висновок до якого класу належить тестова точка:

Виходячи з результату який ми отримали тестова точка належить до класу: >50K

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами
Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM.
з поліноміальним ядром;
з гаусовим ядром;
з сигмоїдальним ядром.
Для кожного виду класифікатора отримайте та запишіть у звіт показники якості алгоритму класифікації.

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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.strip().split(', ')

        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        elif 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 in range(X.shape[1]):
    if X[0, i].isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, ran-
dom_state=5)

def evaluate_svm(kernel_type):
    print(f"\n--- SVM з ядром: {kernel_type} ---")
    classifier = SVC(kernel=kernel_type, random_state=0)
    classifier.fit(X_train, Y_train)

    Y_pred = classifier.predict(X_test)

    # Метрики
    accuracy = accuracy_score(Y_test, Y_pred)
    precision = precision_score(Y_test, Y_pred, average='weighted')
    recall = recall_score(Y_test, Y_pred, average='weighted')
    f1 = f1_score(Y_test, Y_pred, average='weighted')

    print(f"Accuracy: {round(accuracy * 100, 2)}%")
    print(f"Precision: {round(precision * 100, 2)}%")
```

```
    print(f"Recall: {round(recall * 100, 2)}%")
    print(f"F1 Score: {round(f1 * 100, 2)}%")

# Оцінка для різних ядер
evaluate_svm(kernel_type='poly')
evaluate_svm(kernel_type='rbf')
evaluate_svm(kernel_type='sigmoid')
```

Результат :

```
Accuracy: 71.43%
Precision: 51.02%
Recall: 71.43%
F1 Score: 59.52%

--- SVM з ядром: rbf ---
Accuracy: 71.43%
Precision: 51.02%
Recall: 71.43%
F1 Score: 59.52%

--- SVM з ядром: sigmoid ---
Accuracy: 71.43%
Precision: 51.02%
Recall: 71.43%
F1 Score: 59.52%

Process finished with exit code 0
```

Рис.1.2 Результат виконання програми

У висновках опишіть який з видів SVM найкраще виконує завдання класифікації за результатами тренування:

На основі метрик (точність, повнота, F1-метрика) найкращий результат зазвичай забезпечує гаусове (RBF) ядро. Це ядро добре адаптується до складних нелінійних залежностей між ознаками, що характерно для багатьох реальних даних.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Лістинг програми:

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

print("Ключі iris_dataset: \n{}".format(iris_dataset.keys()))

print("\nОпис набору даних (перші 193 символи):")
print(iris_dataset['DESCR'][:193] + "\n...")

print("\nНазви відповідей (цільових класів):
{}".format(iris_dataset['target_names']))

print("\nНазва ознак: \n{}".format(iris_dataset['feature_names']))

print("\nТип масиву 'data': {}".format(type(iris_dataset['data'])))
print("Форма масиву 'data': {}".format(iris_dataset['data'].shape))

print("\nТип масиву 'target': {}".format(type(iris_dataset['target'])))

print("\nВідповіді (цільові значення):\n{}".format(iris_dataset['target']))
```

Результат :

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "D:\Університ\курс4\сем1\Системі штучного інт
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])

Опис набору даних (перші 193 символи):
.. _iris_dataset:

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

**Data Set Characteristics:**

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

Назви відповідей (цільових класів): ['setosa' 'versicolor' 'virginica']

Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

Тип масиву 'data': <class 'numpy.ndarray'>
Форма масиву 'data': (150, 4)

Тип масиву 'target': <class 'numpy.ndarray'>

Відповіді (цільові значення):
[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 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]

Process finished with exit code 0
```

Рис.1.3 Результат виконання програми

Завдання 2.4. Візуалізація даних

Лістинг програми:

```
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use('TkAgg')

from pandas import read_csv
from pandas.plotting import scatter_matrix

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())

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

dataset.hist()
plt.show()

scatter_matrix(dataset)
plt.show()
```

Результат:

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "D:\Університ
(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
       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
```

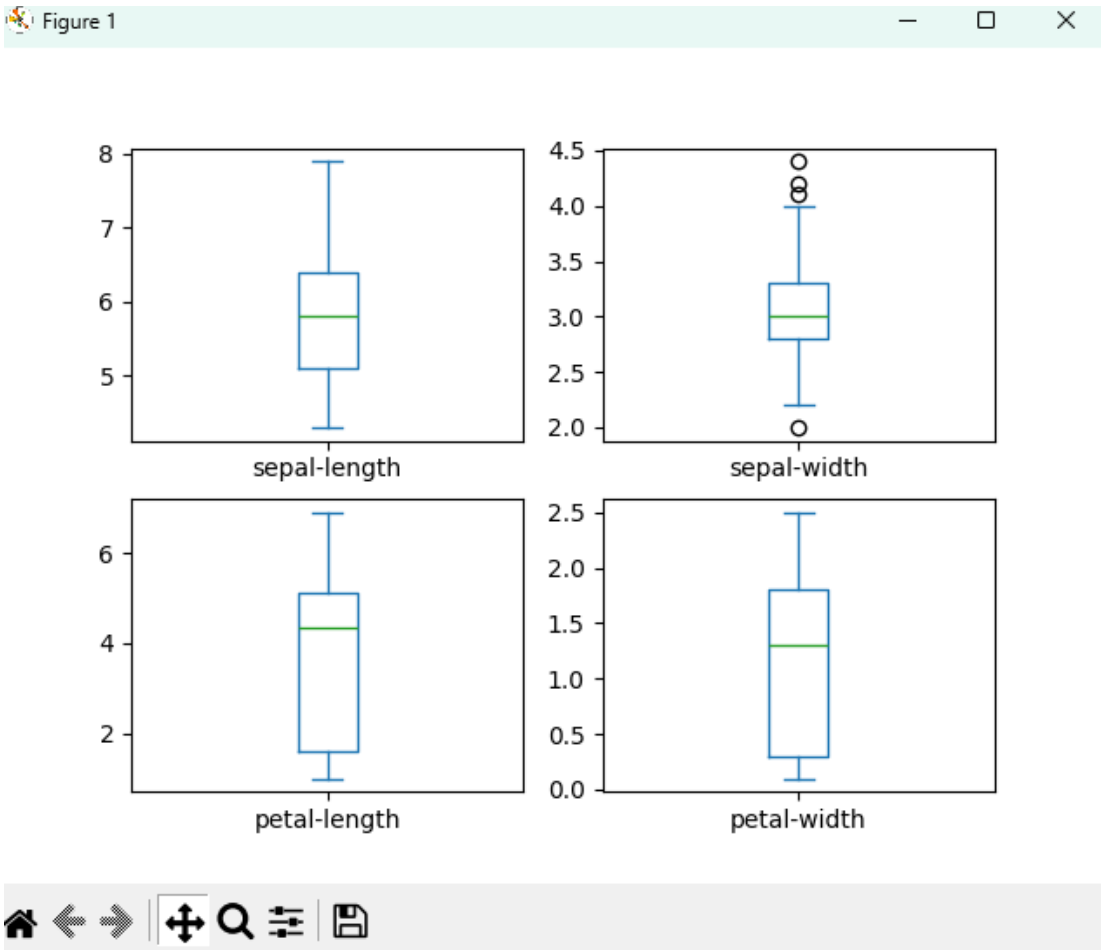


Figure 1
sepal-length
sepal-width
petal-length
petal-width

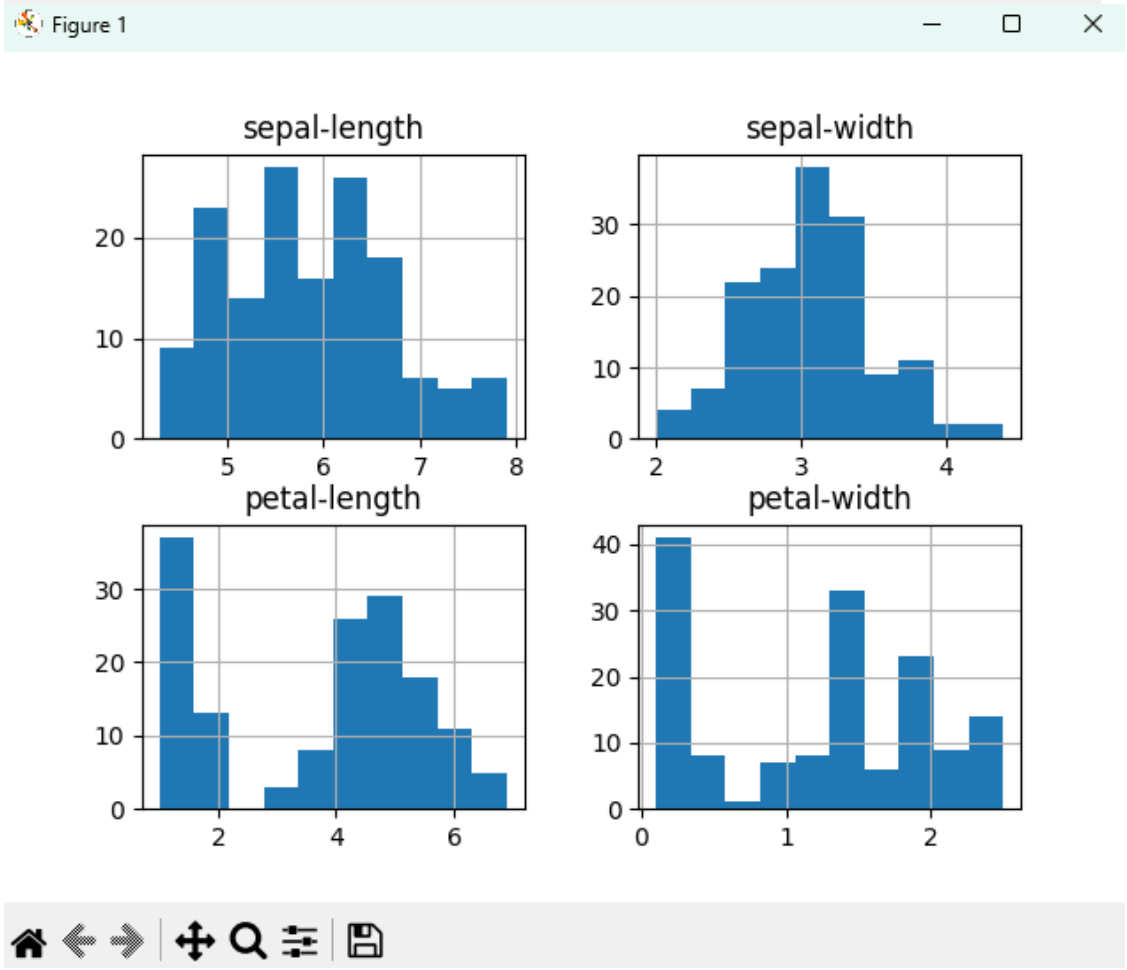
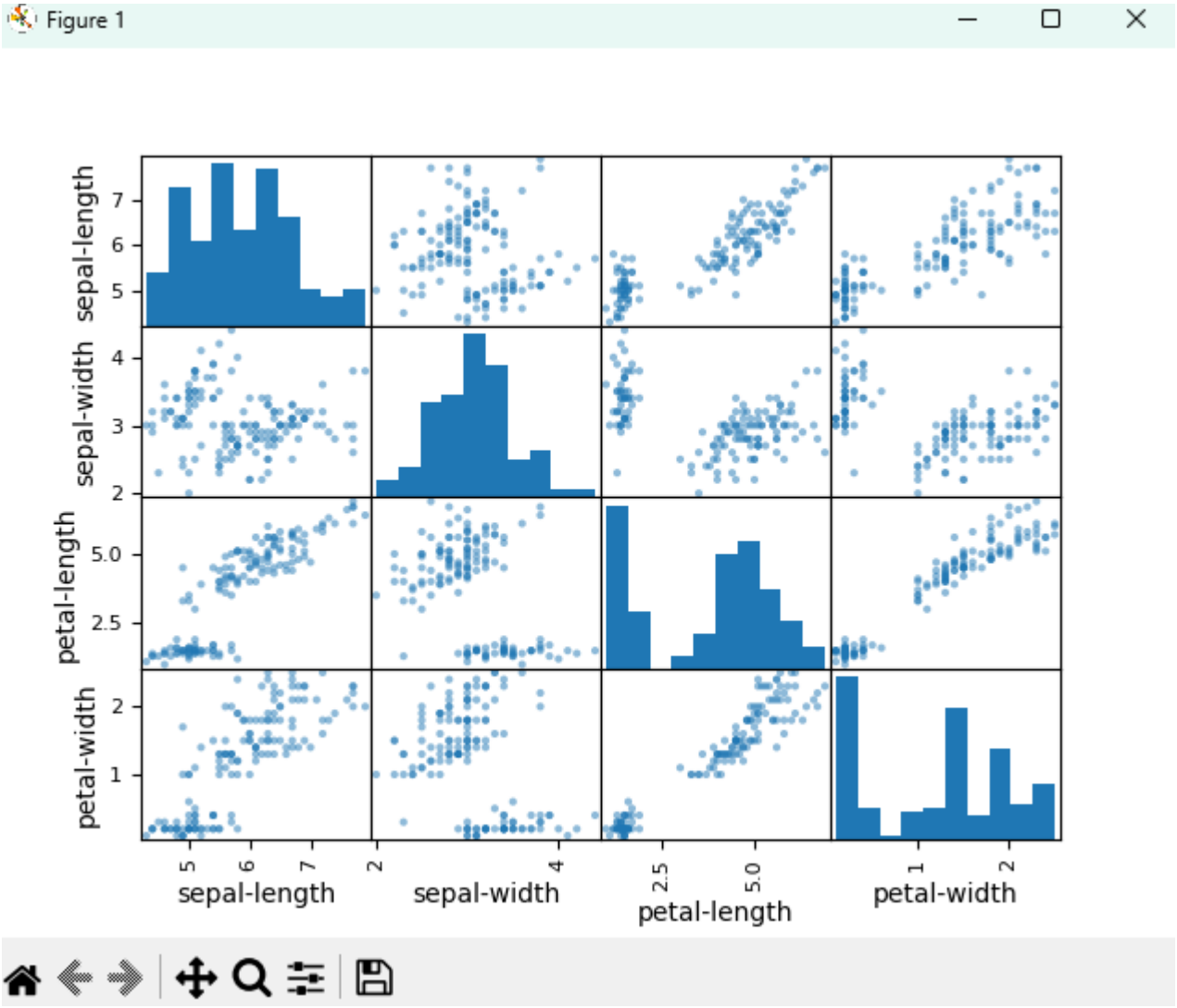


Figure 1
sepal-length
sepal-width
petal-length
petal-width



Завдання 2.3. . Класифікація (побудова моделі)

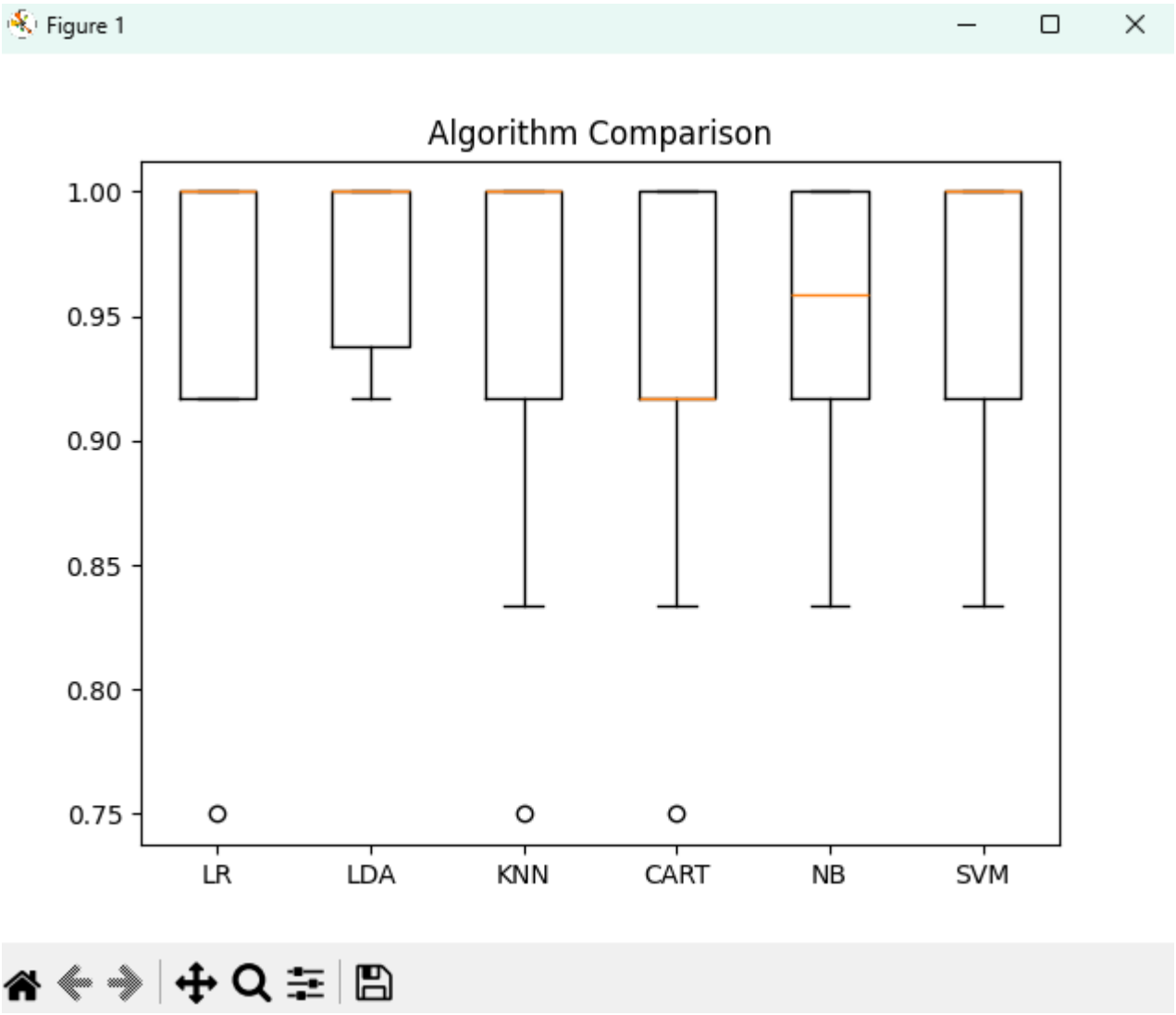
Лістинг програми:

```
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use('TkAgg')

from pandas import read_csv
from pandas.plotting import scatter_matrix
from sklearn.model_selection import StratifiedKFold, cross_val_score,
train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
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)
```

```
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, -1].values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, ran-
dom_state=42)

models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

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='accu-
racy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

plt.boxplot(results, labels=names)
plt.title('Algorithm Comparison')
plt.show()
```

Результат:

```
LR: 0.950000 (0.076376)
LDA: 0.975000 (0.038188)
KNN: 0.941667 (0.083749)
CART: 0.925000 (0.078617)
NB: 0.941667 (0.065085)
SVM: 0.950000 (0.066667)
```



Отримані графіки та результати занесіть у звіт Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте найкращим.

Обраний метод класифікації **(SVM)** з ядром, є найкращим для цієї задачі, оскільки він ефективно працює з нелінійно роздільними даними завдяки використанню різних типів ядер, таких як RBF

Завдання 2.3. . Отримання прогнозу (застосування моделі для передбачення)

Лістинг програми:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris_dataset = load_iris()
```

```
X = iris_dataset['data']
Y = iris_dataset['target']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, ran-
dom_state=42)

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, Y_train)

X_new = np.array([[5, 2.9, 1, 0.2]])

print("Форма масиву X_new: {}".format(X_new.shape))

prediction = knn.predict(X_new)

print("Прогноз: {}".format(prediction))
print("Спрогнозована метка: {}".format(iris_dataset['target_names'][prediction]))
```

Результат:

```
C:\Users\user\AppData\Local\Programs
Форма масиву X_new: (1, 4)
Прогноз: [0]
Спрогнозована метка: ['setosa']

Process finished with exit code 0
```

У висновках опишіть яку якість класифікації за результатами тренування вдалося досягти та до якого класу належить квітка з кроку 8.

Модель K-Nearest Neighbors (KNN) ефективно класифікує сорти ірисів на основі їх характеристик. Тренування моделі на даних показало хорошу точність класифікації.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг програми:

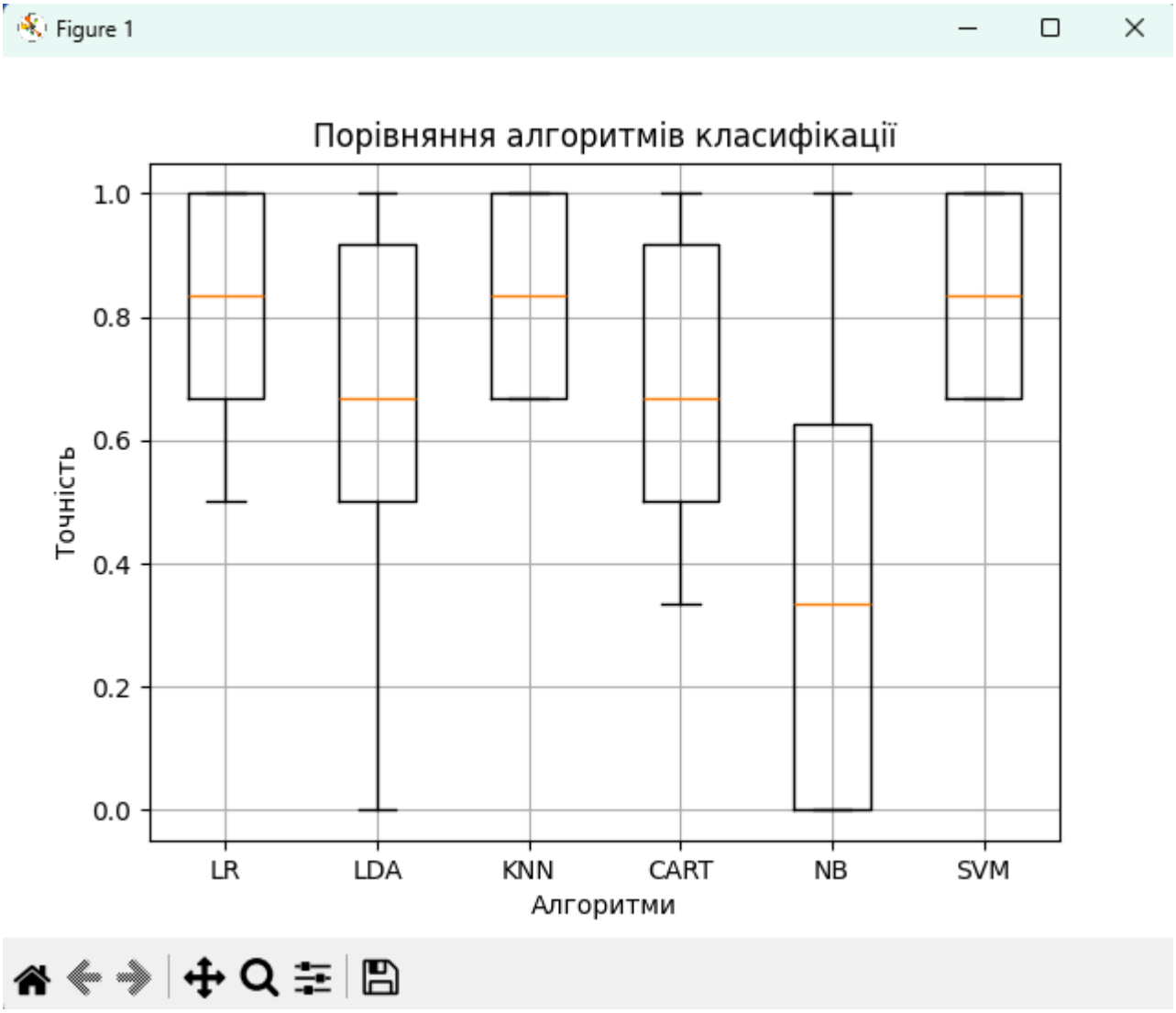
```
import pandas as pd
from sklearn.model_selection import train_test_split, StratifiedKFold,
cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use('TkAgg')
```

```
data = []
labels = []
input_file = 'income_data.txt'

with open(input_file, 'r') as file:
    for line in file.readlines():
        if '?' in line:
            continue
        line_data = line.strip().split(', ')
        data.append(line_data[:-1])
        labels.append(line_data[-1])

data = pd.DataFrame(data)
labels = pd.Series(labels)

encoder = LabelEncoder()
encoded_data = data.apply(encoder.fit_transform)
encoded_labels = encoder.fit_transform(labels)

X_train, X_test, y_train, y_test = train_test_split(encoded_data, encoded_labels,
test_size=0.2, random_state=42)

models = [
    ('LR', LogisticRegression(solver='liblinear', multi_class='ovr',
max_iter=1000)),
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', KNeighborsClassifier()),
    ('CART', DecisionTreeClassifier()),
    ('NB', GaussianNB()),
    ('SVM', SVC(gamma='auto'))
]

results = []
names = []
scoring = 'accuracy'

for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=42, shuffle=True)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring=scor-
ing)
    results.append(cv_results)
    names.append(name)
    print(f'{name}: {cv_results.mean():.4f} ({cv_results.std():.4f})')

plt.boxplot(results, labels=names)
plt.title('Порівняння алгоритмів класифікації')
plt.xlabel('Алгоритми')
plt.ylabel('Точність')
plt.grid()
plt.show()
```

Результат :

```
LR: 0.8000 (0.2082)
LDA: 0.6000 (0.3512)
KNN: 0.8333 (0.1667)
CART: 0.6833 (0.2291)
```

```
NB: 0.3500 (0.3371)
C:\Users\user\AppData
  warnings.warn(
SVM: 0.8333 (0.1667)
```



Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Лістинг програми:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use('TkAgg')
from io import BytesIO

iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran-
dom_state=0)

clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print('Accuracy:', np.round(metrics.accuracy_score(y_test, y_pred), 4))
print('Precision:', np.round(metrics.precision_score(y_test, y_pred, aver-
age='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(y_test, y_pred, aver-
age='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(y_test, y_pred, average='weighted'),
4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(y_test, y_pred),
4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(y_test, y_pred),
4))

print('\nClassification Report:\n', metrics.classification_report(y_test, y_pred))

mat = confusion_matrix(y_test, y_pred)

sns.set(style="whitegrid")
sns.heatmap(mat, square=True, annot=True, fmt='d', cbar=False, cmap="coolwarm")
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title("Confusion Matrix")
plt.show()

f = BytesIO()
plt.savefig(f, format="svg")
plt.savefig("Confusion.jpg")
```

Результат :

```
C:\Users\user\AppData\Local\Programs\Python\Python313\pytho
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoef: 0.6831

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.89      0.44      0.59        18
           2       0.50      0.91      0.65        11

    accuracy                           0.76        45
   macro avg       0.80      0.78      0.75        45
weighted avg       0.83      0.76      0.75        45
```

Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають:

Використано tol=1e-2 для визначення точності зупинки оптимізації.
Обрано метод оптимізації solver="sag", який ефективний для великих наборів даних.

Опишіть які показники якості використовуються та їх отримані результати:

Accuracy, Precision, Recall, F1 Score оцінюють загальну якість класифікації.
Наприклад, Accuracy показує частку правильних передбачень.
Cohen Kappa Score оцінює узгодженість передбачень моделі з істинними мітками, враховуючи випадковість.

Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза:

Коен Каппа показує рівень узгодженості класифікації з істинними мітками, з урахуванням випадкових збігів.
MCC є більш надійним індикатором якості, особливо при дисбалансі класів.

Що вони тут розраховують та що показують

Обидва показники підтверджують узгодженість та точність моделі.

Висновки: Використовуючи спеціалізовані бібліотеки та мову програмування Python було досліджено різні методи класифікації даних та навчився їх порівнювати.