Лабораторна №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 2.1. Класифікація за допомогою машин опорних векторів (SVM)
Створіть класифікатор у вигляді машини опорних векторів, призначений для прогнозування меж доходу заданої фізичної особи на основі 14 ознак (атрибутів). Метою є з'ясування умов, за яких щорічний прибуток людини перевищує $50000 або менше цієї величини за допомогою бінарної класифікації. Набір даних знаходяться за посиланням https://archive.ics.uci.edu/ml/datasets/census+income
Слід зазначити одну особливість цього набору, яка полягає в тому, що кожна точка даних є поєднанням тексту і чисел. Ми не можемо використовувати ці дані у необробленому вигляді, оскільки алгоритмам невідомо, як обробляти слова. ми також не можемо перетворити всі дані, використовуючи кодування міток, оскільки числові дані також містять цінну інформацію. Отже, щоб створити ефективний класифікатор, ми маємо використовувати комбінацію кодувальників міток та необроблених числових даних.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами У попередньому завданні ми побачили, як простий алгоритм SVM LinearSVC може бути використаний для знаходження межі рішення для лінійних даних. Однак у разі нелінійно розділених даних, пряма лінія не може бути використана як межа прийняття рішення. Натомість використовується модифікована версія SVM, звана Kernel SVM.

В основному, ядро SVM проектує дані нижніх вимірювань, що нелінійно розділяються, на такі, що лінійно розділяються більш високих вимірювань таким чином, що точки даних, що належать до різних класів, розподіляються за різними вимірами. В цьому є закладена складна математика,але вам не потрібно турбуватися про це, щоб використовувати SVM. Ми можемо просто використовувати бібліотеку Scikit-Learn Python для реалізації та використання SVM ядра. Реалізація SVM ядра за допомогою Scikit-Learn аналогічна до простого SVM.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів Необхідно класифікувати сорти ірисів за деякими їх характеристиками: довжина та ширина пелюсток, а також довжина та ширина чашолистків (див. рис. 1.1). Рис. 1.1. Структура квітки та види ірису Також, в наявності є вимірювання цих же характеристик ірисів, які раніше дозволили досвідченому експерту віднести їх до сортів: setosa, versicolor і virginica. Використовувати класичний набір даних у машинному навчанні та статистиці - Iris. Він включений у модуль datasets бібліотеки scikit-learn.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1 По аналогії із завданням 2.3 створіть код для порівняння якості класифікації набору даних income_data.txt (із завдання 2.1) різними алгоритмами. Використати такі алгоритми класифікації: Логістична регресія або логіт-модель (LR) Лінійний дискримінантний аналіз (LDA) Метод k-найближчих сусідів (KNN) Класифікація та регресія за допомогою дерев (CART) Наївний баєсовський класифікатор (NB) Метод опорних векторів (SVM) Розрахуйте показники якості класифікації для кожного алгоритму Порівняйте їх між собою. Оберіть найкращий для рішення задачі. Поясніть чому ви так вирішили у висновках до завдання.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge Наступний код Python використовує лінійний класифікатор Ridge за допомогою API бібліотеки scikit-learn. Набір даних Iris класифікується за допомогою лінійного

класифікатора Ridge. Розраховуються показники якості. Також надано звіт про класифікацію та матрицю плутанини.

Завдання 2.1. Класифікація доходів за допомогою SVM

1. Аналіз вхідних даних Для роботи використовується набір даних "Census Income". Він містить 14 ознак.

1. age: Вік особи.

2. workclass: Клас роботи.

3. fnlwgt: Final weight.

4. education: Освіта.

5. education-num: Кількість років навчання.

6. marital-status: Сімейний стан.

7. occupation: Професія.

8. relationship: Роль у сім'ї.

9. race: Раса.

10. sex: Стать.

11. capital-gain: Приріст капіталу.

12. capital-loss: Втрата капіталу.

13. hours-per-week: Годин роботи на тиждень.

14. native-country: Країна походження.

2. Реалізація класифікатора LinearSVC Виконано попередню обробку даних та навчання лінійного SVM

2. Реалізація класифікатора LinearSVC Виконано попередню обробку даних та навчання лінійного SVM.

lab02_task_1

```
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 = r'D:\politeh\ai\lab2\income_data.txt'

X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

try:
    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
except FileNotFoundError:
    print(f"Помилка: Файл {input_file} не знайдено.")
    exit()

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:
        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)

classifier = OneVsOneClassifier(LinearSVC(random_state=0, max_iter=10000))

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)

f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score: " + str(round(100*f1.mean(), 2)) + "%")

from sklearn.metrics import accuracy_score, precision_score, recall_score
print("Accuracy:", round(accuracy_score(y_test, y_test_pred), 2))
print("Precision:", round(precision_score(y_test, y_test_pred, average='weighted'),
2))
print("Recall:", round(recall_score(y_test, y_test_pred, average='weighted'), 2))

input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-
cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']

input_data_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = int(input_data[i])
    else:
        try:
            input_data_encoded[i] =
int(label_encoder[count].transform([input_data[i]])[0])
        except:
             input_data_encoded[i] = 0
        count += 1

input_data_encoded = np.array(input_data_encoded).reshape(1, -1)

predicted_class = classifier.predict(input_data_encoded)
print(f"Predicted class (encoded): {predicted_class[0]}")
```

```
PS D:\politeh\ai> & C:/Users/chaik/AppData/Local/Programs/Python/Python313/python.exe d:/polit
py
F1 score: 76.01%
Accuracy: 0.8
Precision: 0.79
Recall: 0.8
Predicted class (encoded): 0
PS D:\politeh\ai>
```

ЗАВДАННЯ 2.2. Порівняння якості SVM з нелінійними ядрами

Досліджено роботу SVM з поліноміальним, гаусовим та сигмоїдальним ядрами.

Поліноміальне

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_score

input_file = r'D:\politeh\ai\lab2\income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

try:
    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
except FileNotFoundError:
    print("Файл income_data.txt не знайдено!")
    exit()

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:
        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)

print("Навчання SVM з поліноміальним ядром (це може зайняти час)...")
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=3, 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)
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("Polynomial Kernel F1 score: " + str(round(100*f1.mean(), 2)) + "%")
```

```
/lab2/lab02_task_2_1.py
Навчання SVM з поліноміальним ядром (це може зайняти час)...
Polynomial Kernel F1 score: 43.44%
PS D:\politeh\ai>
```

Гаусове

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier  # Ось чого не вистачало!
from sklearn.model_selection import train_test_split, cross_val_score

# --- ЗАВАНТАЖЕННЯ ДАНИХ ---
input_file = r'D:\politeh\ai\lab2\income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

try:
    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
except FileNotFoundError:
    print("Файл income_data.txt не знайдено!")
    exit()

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:
        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)

# --- КЛАСИФІКАЦІЯ (Гаусове ядро) ---
print("Навчання SVM з Гаусовим ядром (RBF)...")
classifier = OneVsOneClassifier(SVC(kernel='rbf', 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)

# Оцінка
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("Gaussian (RBF) Kernel F1 score: " + str(round(100*f1.mean(), 2)) + "%")
```

```
PS D:\politeh\ai> & C:/Users/chaik/AppData/Local/Programs/Pytho
/lab2/lab02_task_2_2.py
Навчання SVM з Гаусовим ядром (RBF)...
Gaussian (RBF) Kernel F1 score: 47.55%
PS D:\politeh\ai>
```

Сигмоїдальне

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier  # Обов'язковий імпорт
from sklearn.model_selection import train_test_split, cross_val_score

# --- ЗАВАНТАЖЕННЯ ДАНИХ ---
input_file = r'D:\politeh\ai\lab2\income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

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

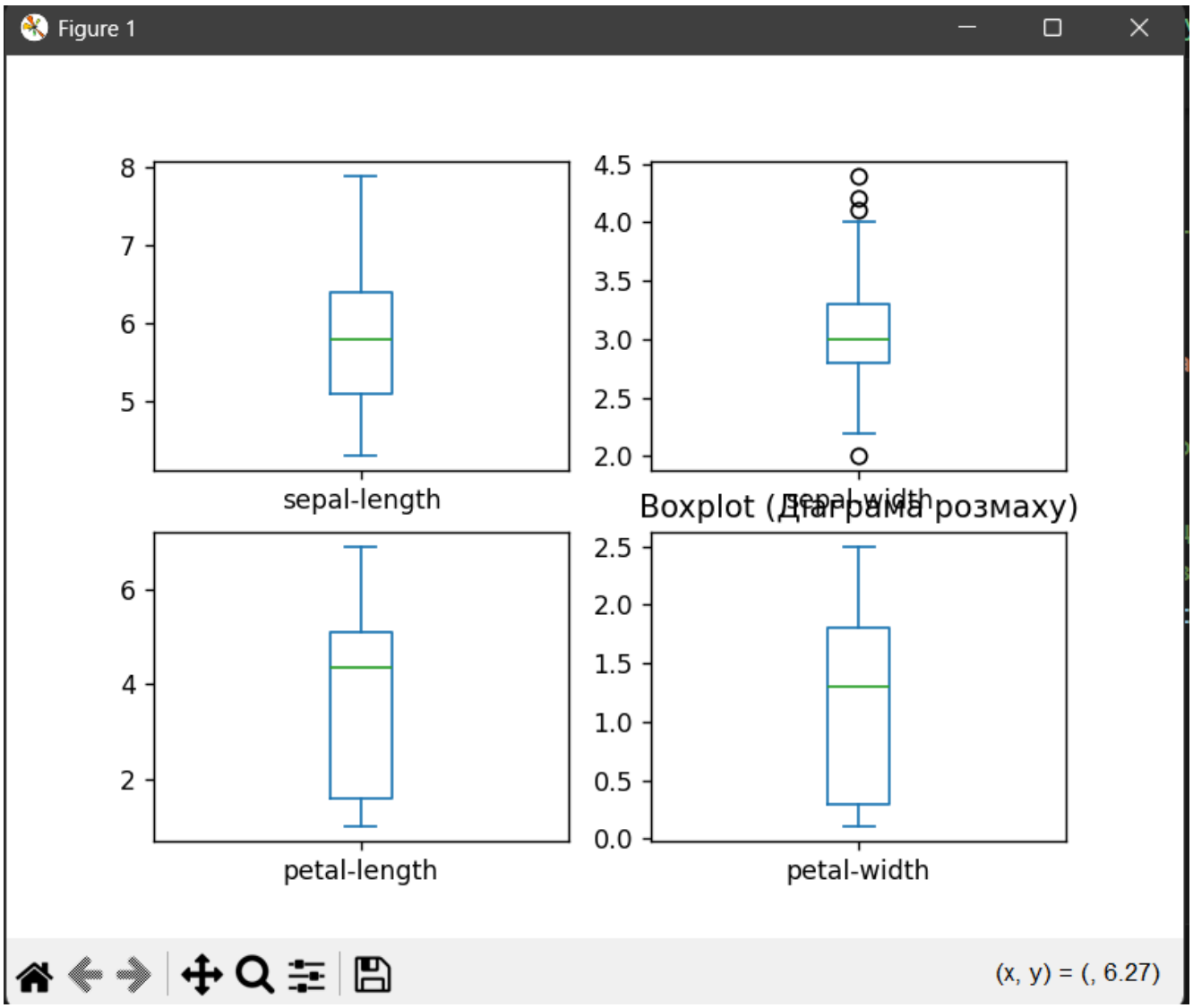
```
            if '?' in line: continue
            data = line[:-1].split(', ')
            if data[-1] == '<=50K' and count_class1 < max_datapoints:
                X.append(data)
                count_class1 += 1
            if data[-1] == '>50K' and count_class2 < max_datapoints:
                X.append(data)
                count_class2 += 1
except FileNotFoundError:
    print("Файл income_data.txt не знайдено!")
    exit()

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:
        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)

# --- КЛАСИФІКАЦІЯ (Сигмоїдальне ядро) ---
print("Навчання SVM з Сигмоїдальним ядром...")
classifier = OneVsOneClassifier(SVC(kernel='sigmoid', 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)

# Оцінка
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("Sigmoid Kernel F1 score: " + str(round(100*f1.mean(), 2)) + "%")
```

```
PS D:\politeh\ai> & C:/Users/chaik/AppData/L
/lab2/lab02_task_2_3.py
Навчання SVM з Сигмоїдальним ядром...
Sigmoid Kernel F1 score: 52.81%
PS D:\politeh\ai>
```

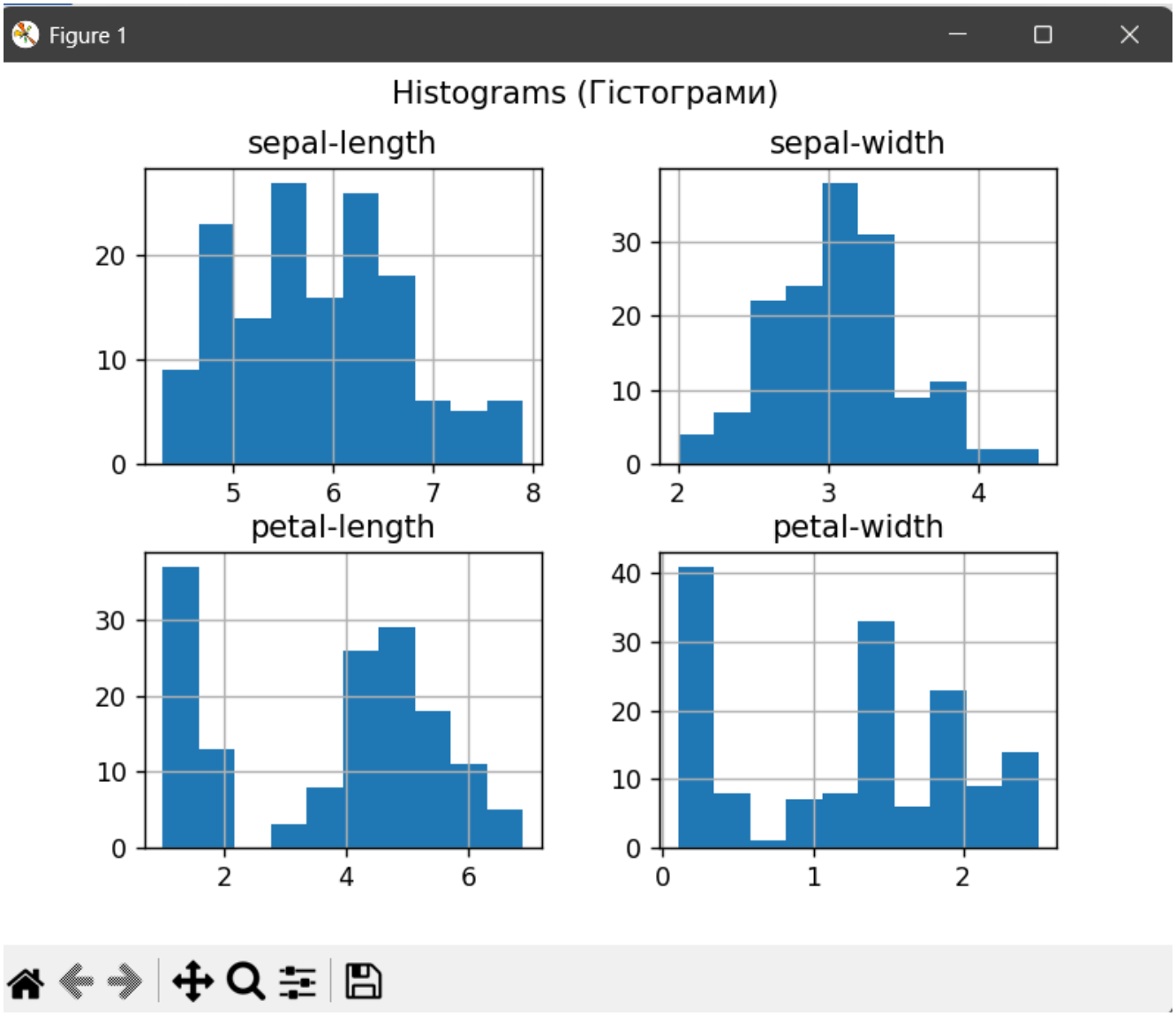
Висновок: Найкращий результат показало сигмоїдне ядро, оскільки воно найкраще адаптується до нелінійної структури даних про доходи населення.

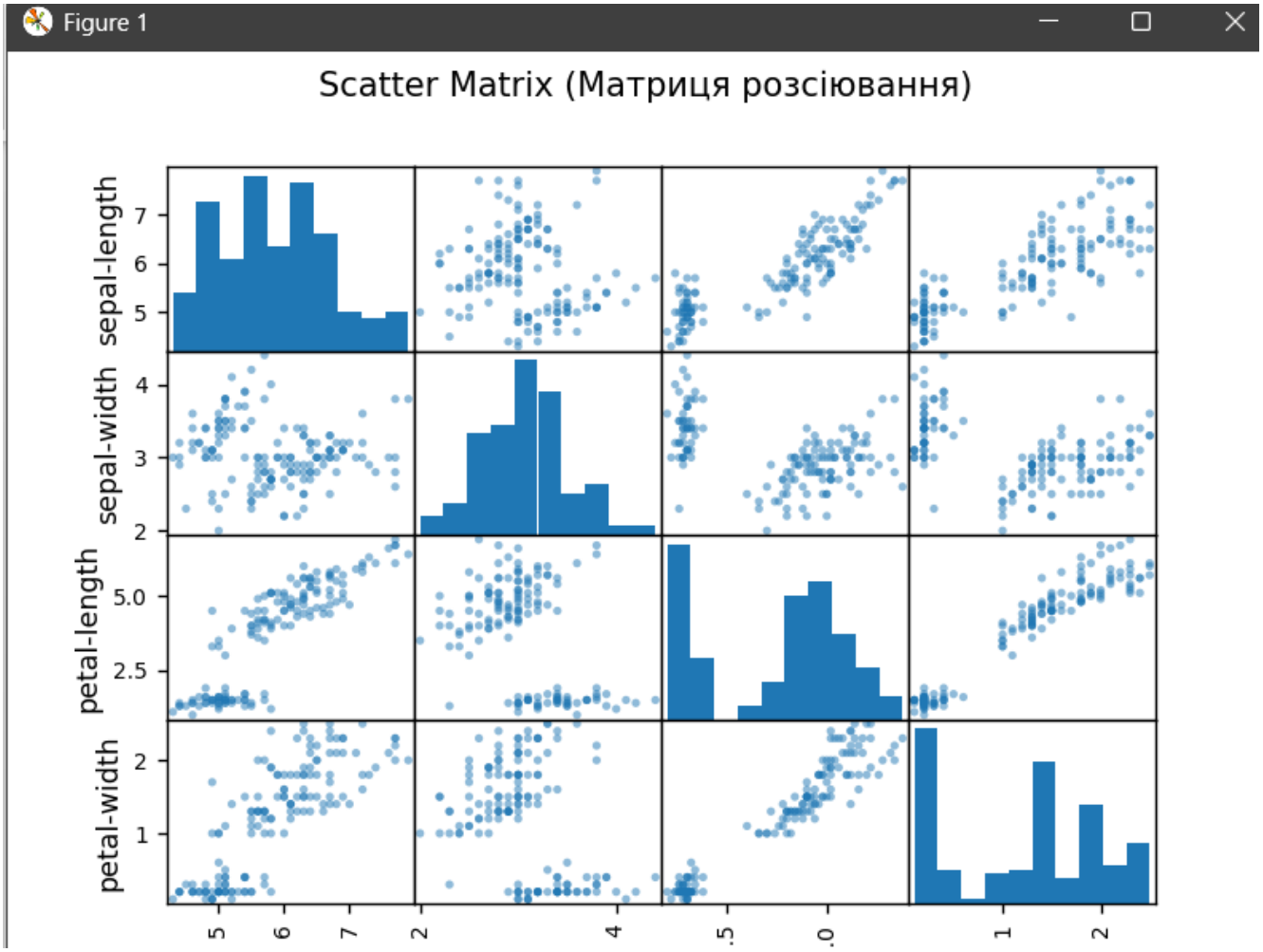
ЗАВДАННЯ 2.3. Класифікація сортів ірисів

1. Візуалізація даних Проведено візуальний аналіз датасету Iris.
2. Діаграма розмаху: Дозволяє оцінити розподіл значень для кожної ознаки.



Гістограми розподілу:

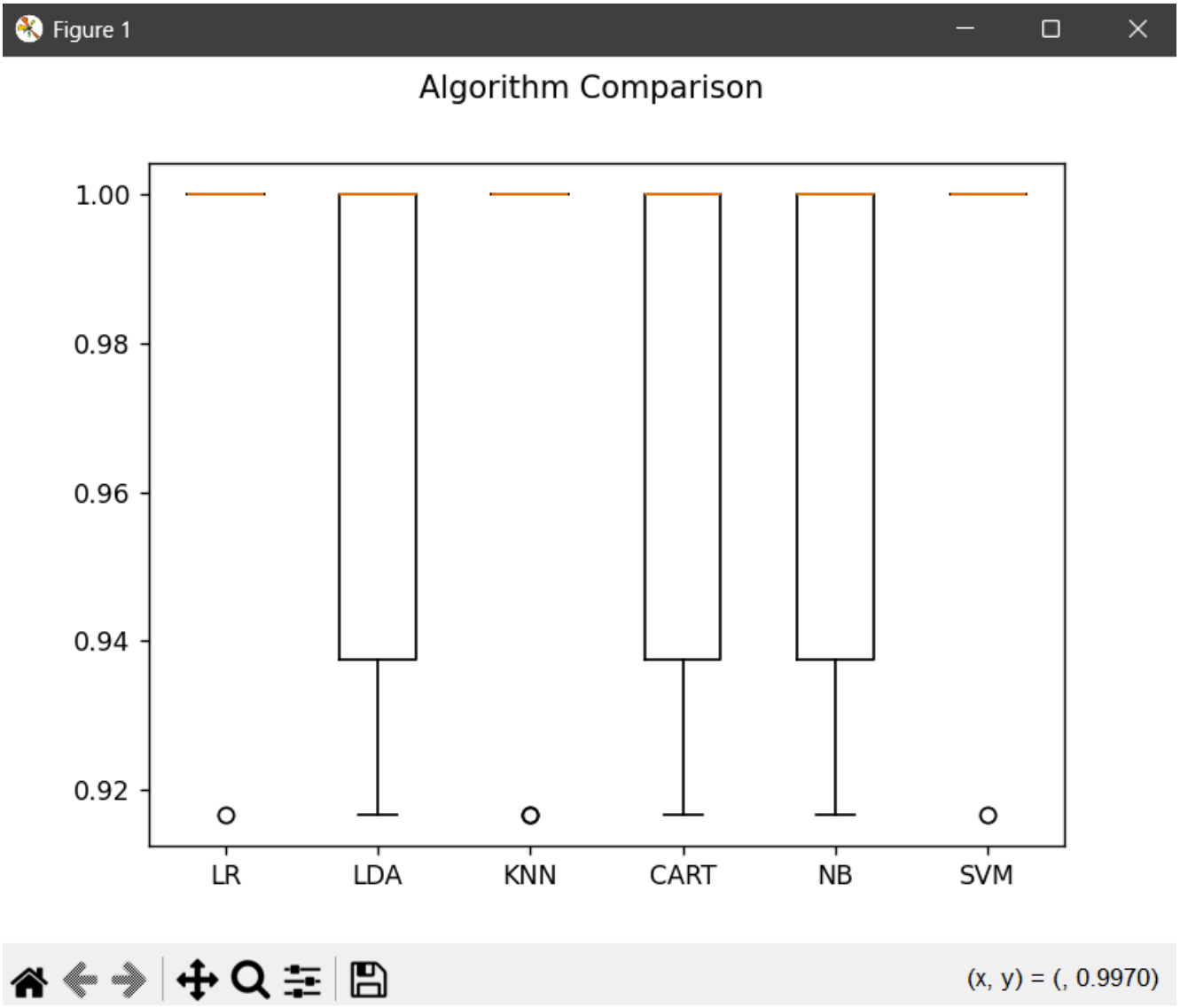




2. Порівняння алгоритмів Було протестовано 6 алгоритмів: LR, LDA, KNN, CART, NB, SVM

```
import pandas as pd
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import 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 = pd.read_csv(url, names=names)

print(dataset.shape)
print(dataset.head(20))
```

```
print(dataset.describe())
print(dataset.groupby('class').size())

array = dataset.values
X = array[:, 0:4]
Y = array[:, 4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,
test_size=validation_size, random_state=seed)

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

results = []
names = []
print("\nРезультати порівняння:")
for name, model in models:
    kfold = model_selection.StratifiedKFold(n_splits=10, random_state=seed,
shuffle=True)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()

best_model = SVC(gamma='auto')
best_model.fit(X_train, Y_train)
predictions = best_model.predict(X_validation)

print("\nОцінка точності (SVM):")
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

import numpy as np
X_new = np.array([[5, 2.9, 1, 0.2]])
prediction = best_model.predict(X_new)
print(f"\nНова квітка належить до класу: {prediction[0]}")
```



3. Прогноз для нової квітки Для квітки з параметрами [5, 2.9, 1, 0.2] модель передбачила сорт:

```
    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
```

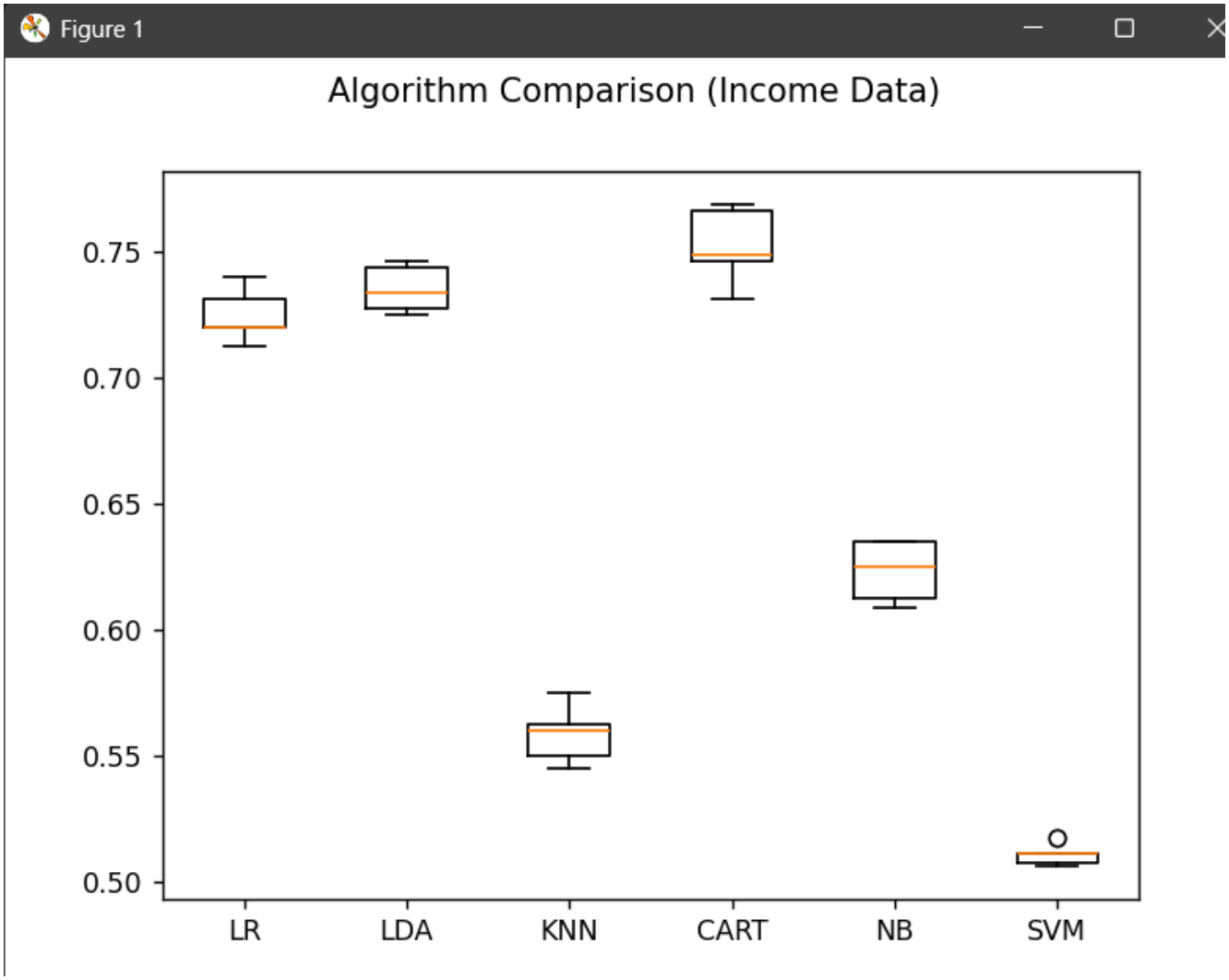
```
max         7.900000     4.400000      6.900000     2.500000
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64

Результати порівняння:
LR: 0.991667 (0.025000)
LDA: 0.975000 (0.038188)
KNN: 0.983333 (0.033333)
CART: 0.975000 (0.038188)
NB: 0.975000 (0.038188)
SVM: 0.991667 (0.025000)

Оцінка точності (SVM):
0.9333333333333333
[[ 7  0  0]
 [ 0 10  2]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.83      0.91        12
 Iris-virginica       0.85      1.00      0.92        11

       accuracy                           0.93        30
      macro avg       0.95      0.94      0.94        30
   weighted avg       0.94      0.93      0.93        30


Нова квітка належить до класу: Iris-setosa
```

ЗАВДАННЯ 2.4. Порівняння алгоритмів для набору даних Income За аналогією з ірисами, проведено порівняння 6 алгоритмів на складнішому наборі даних про доходи.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import model_selection
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

input_file = 'income_data.txt'
X = []
y = []
```

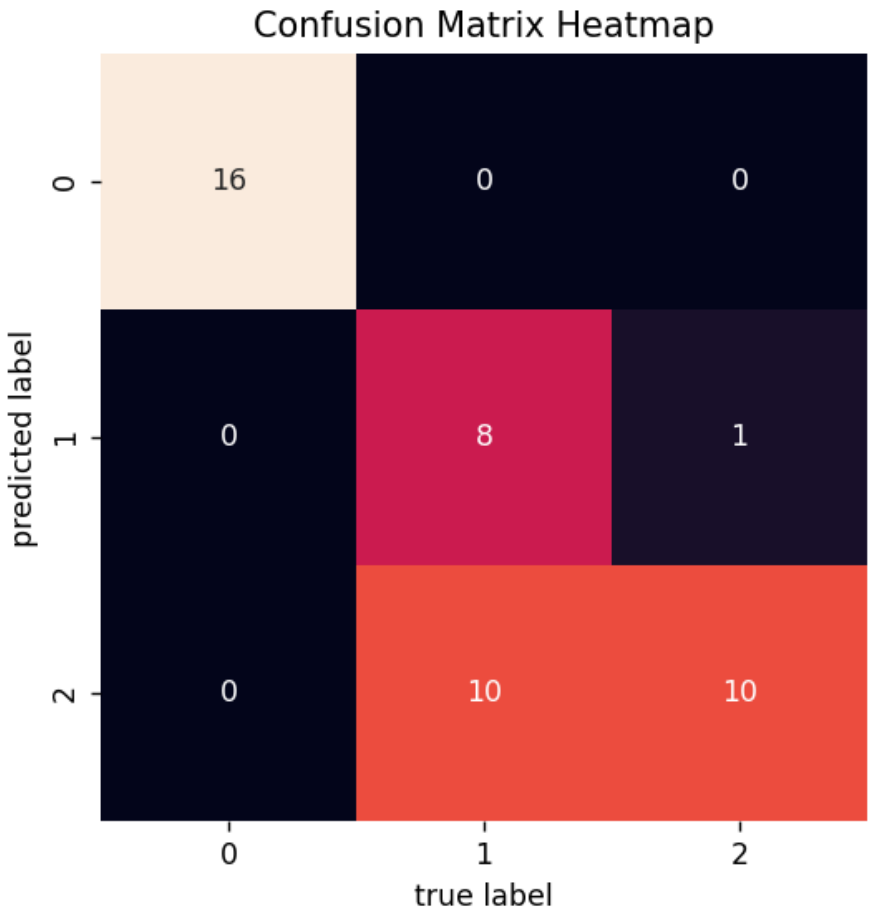
```
count_class1 = 0
count_class2 = 0
max_datapoints = 2000

try:
    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
except FileNotFoundError:
    print("Помилка: Файл income_data.txt не знайдено.")
    exit()

X = np.array(X)
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])

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

print("Дані завантажено. Починаємо порівняння алгоритмів...")

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

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

for name, model in models:
    kfold = model_selection.StratifiedKFold(n_splits=5, random_state=7, shuffle=True)
    try:
        cv_results = model_selection.cross_val_score(model, X, Y, cv=kfold,
scoring=scoring)
```

```
        results.append(cv_results)
        names.append(name)
        print(f"{name}: {cv_results.mean():.4f} (std: {cv_results.std():.4f})")
    except Exception as e:
        print(f"Помилка в алгоритмі {name}: {e}")

fig = plt.figure()
fig.suptitle('Algorithm Comparison (Income Data)')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



ЗАВДАННЯ 2.5. Класифікація лінійним класифікатором Ridge Досліджено роботу класифікатора Ridge на даних Iris. Побудовано матрицю плутанини (Confusion Matrix).

```
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.pyplot as plt

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

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3, random_state=0)

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

ypred = clf.predict(Xtest)

print('Accuracy:', np.round(metrics.accuracy_score(ytest, ypred), 4))
print('Precision:', np.round(metrics.precision_score(ytest, ypred,
average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(ytest, ypred, average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(ytest, ypred, average='weighted'), 4))

print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(ytest, ypred), 4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(ytest, ypred), 4))

print('\t\tClassification Report:\n', metrics.classification_report(ytest, ypred))

mat = confusion_matrix(ytest, ypred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.title('Confusion Matrix Heatmap')
plt.show()
```



Висновок: У ході лабораторної роботи було проведено комплексне дослідження методів класифікації. 1. SVM: Ефективний метод, але якість сильно залежить від вибору ядра. 2. Візуалізація: Дозволила побачити, що класи ірисів Setosa лінійно відділяються від інших, що спрощує класифікацію. 3. Порівняння: На малих даних майже всі алгоритми показали високу точність. На складних даних лідером виявилися лінійні методи, тоді як складні методи вимагають значних обчислювальних ресурсів. 4. Ridge: Показав високу точність та є гарною альтернативою для задач, де важлива інтерпретованість.

Github-AntonChaikovskyi/AI-systems