**ЛАБОРАТОРНА РОБОТА №2**

**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ**

**Мета заняття:** 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)

|  |  |  |
| --- | --- | --- |
| **Назва** | **Опис** | **Тип значень** |
| age | Вік | Числове |
| workclass | Вид працевлаштування | Категоріальне |
| fnlwgt | Кількість осіб з такими ж ознаками | Числове |
| education | Навчання | Категоріальне |
| education-num | Років навчання | Числове |
| marital-status | Сімейне положення | Категоріальне |
| occupation | Професія | Категоріальне |
| relationship | Відносини | Категоріальне |
| race | Раса | Категоріальне |
| sex | Стать | Категоріальне |
| capital-gain | Приріст капіталу | Числове |
| capital-loss | Втрата капіталу | Числове |
| hours-per-week | Кількість робочих годин на тиждень | Числове |
| native-country | Країна походження | Категоріальне |

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  
  
# Вхідний файл, який містить дані  
input\_file = 'income\_data.txt'  
  
# Читання даних  
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  
  
# Перетворення на масив numpy  
X = np.array(X)  
  
# Перетворення рядкових даних на числові  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 current\_label\_encoder = preprocessing.LabelEncoder()  
 label\_encoder.append(current\_label\_encoder)  
 X\_encoded[:, i] = current\_label\_encoder.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)  
print("Accuracy score: " + str(round(100 \* accuracy.mean(), 2)) + "%")  
  
precision = cross\_val\_score(classifier, X, y, scoring='precision', cv=3)  
print("Precision score: " + str(round(100 \* precision.mean(), 2)) + "%")  
  
recall = cross\_val\_score(classifier, X, y, scoring='recall', cv=3)  
print("Recall score: " + str(round(100 \* recall.mean(), 2)) + "%")  
  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")  
  
# Передбачення результату для тестової точки даних  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',  
 'Male','0', '0', '40', 'United-States']  
  
# Кодування тестової точки даних  
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:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([(input\_data[i])])[-1])  
 count += 1  
  
input\_data\_encoded = np.array(input\_data\_encoded)  
  
# Використання класифікатора для кодованої точки даних  
# та виведення результату  
predicted\_class = classifier.predict([input\_data\_encoded])  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Рисунок 1. Код програми

Accuracy score: 62.64%

Precision score: 69.18%

Recall score: 38.24%

F1 score: 56.15%

Тестова точка - <=50K. Отже тестова точка має дохід менше 50 тисяч в рік.

**Завдання 2.2.** Порівняння якості класифікаторів SVM з нелінійними ядрами

**Поліноміальне ядро**

Accuracy score: 58.41%

Precision score: 41.6%

Recall score: 33.05%

F1 score: 46.5%

>50K

**Гаусове ядро**

Accuracy score: 78.61%

Precision score: 98.72%

Recall score: 14.26%

F1 score: 71.95%

>50K

**Сигмоїдальне ядро**

Accuracy score: 63.89%

Precision score: 27.01%

Recall score: 26.48%

F1 score: 63.77%

<=50K

Найбільш точним виявився SVM класифікатор з гаусовим ядром.

**Завдання 2.3.** Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

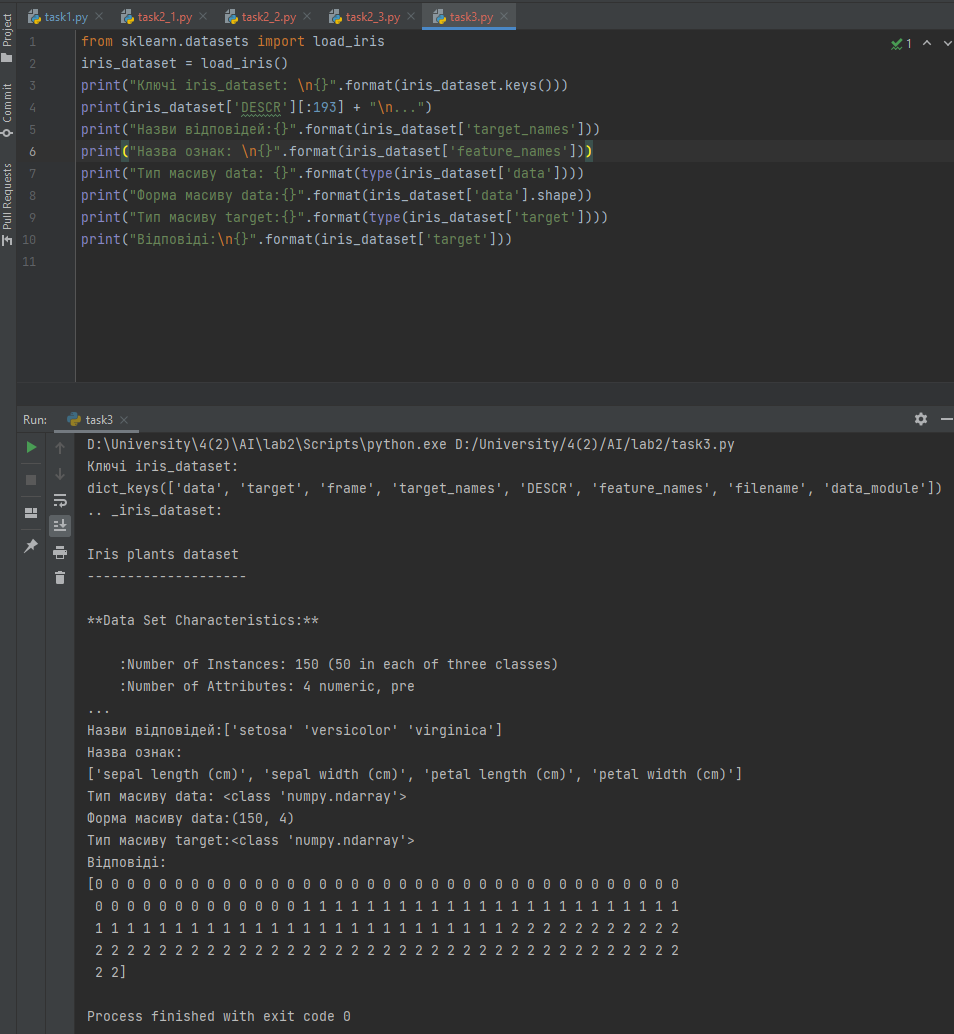


Рисунок 1. Код структури даних

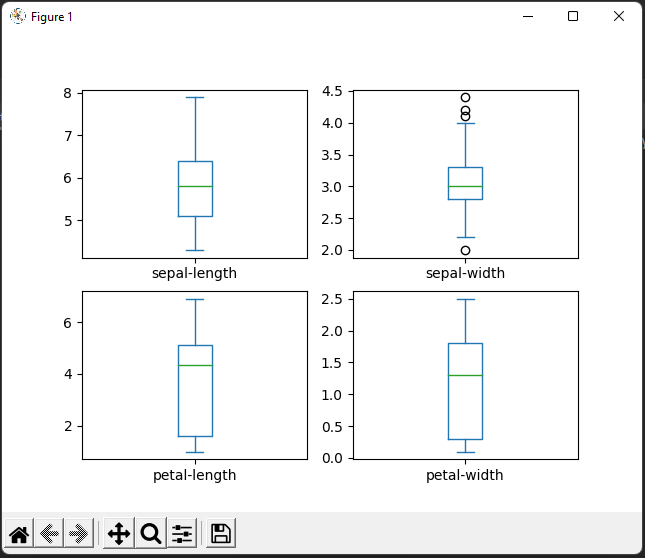


Рисунок 2. Одновимірні графіки

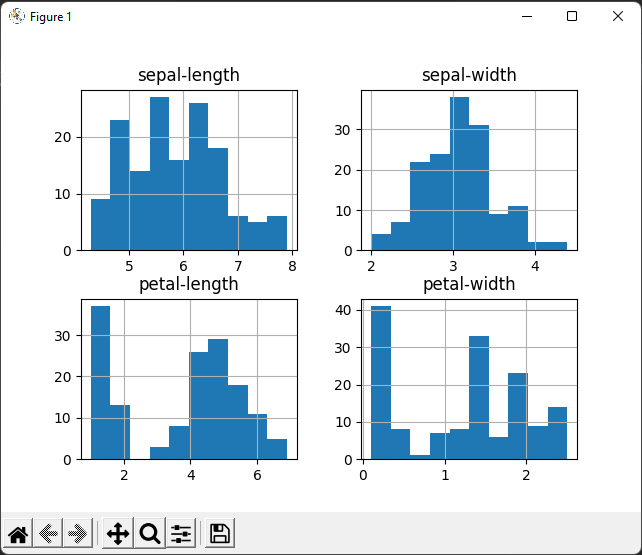


Рисунок 3. Діаграма розмаху атрибутів вхідних даних

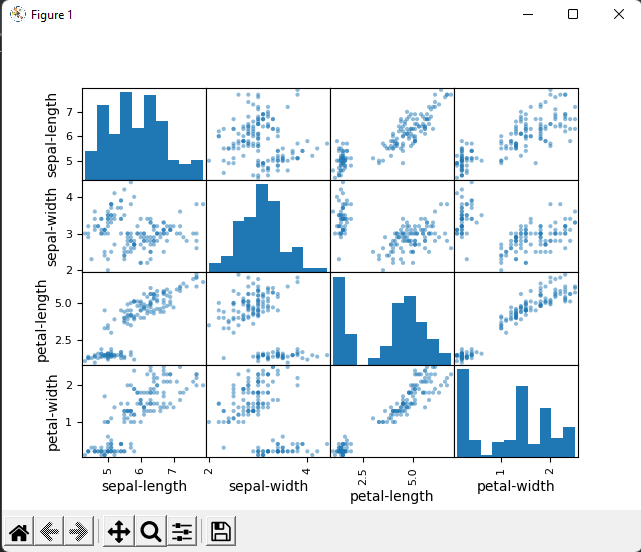


Рисунок 4. Матриця діаграм розсіювання

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)  
  
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)  
pyplot.show()  
  
dataset.hist()  
pyplot.show()  
  
scatter\_matrix(dataset)  
pyplot.show()

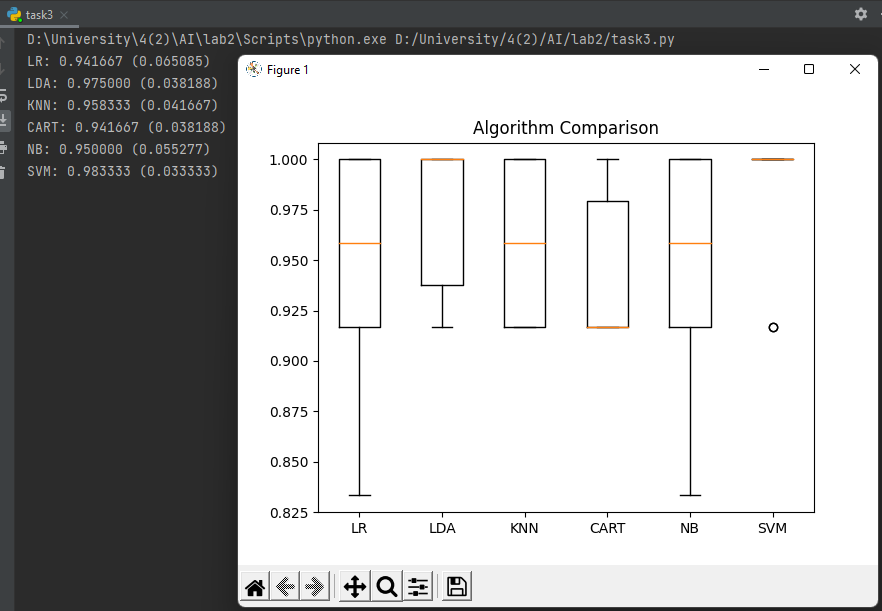


Рисунок 5. Порівняння алгоритмів

Проаналізувавши ортиманий графік, я обрав метод класифікації SVM, тому що він показав найвищу якість.

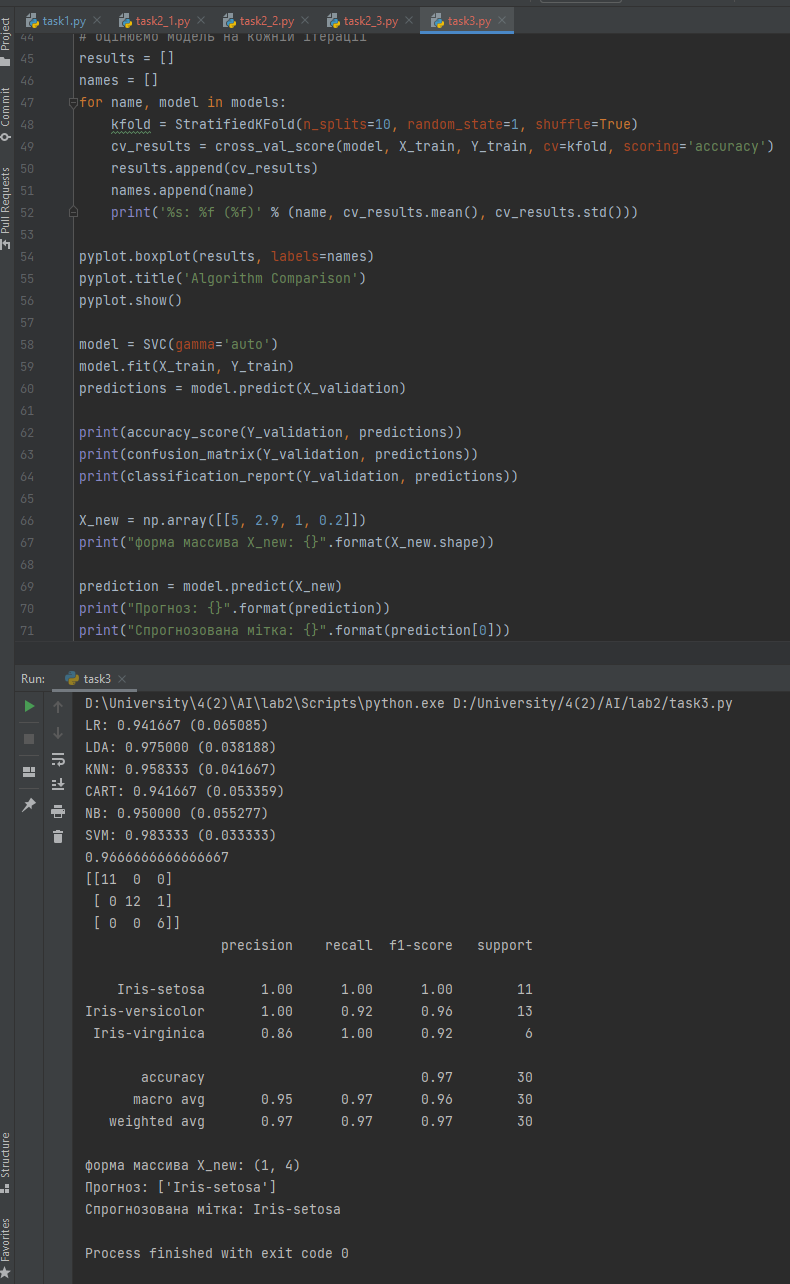


Рисунок 6. Результат

**Завдання 2.4.** Порівняння якості класифікаторів для набору даних завдання 2.1

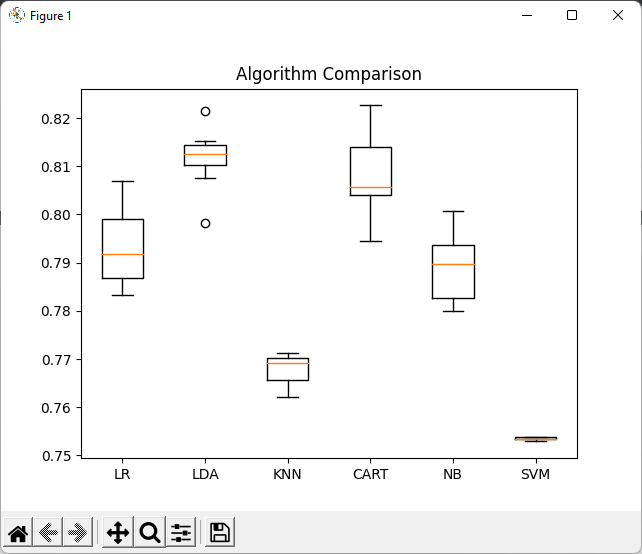


Рисунок 7. Порівняння алгоритмів

**Завдання 2.5.** Класифікація даних лінійним класифікатором Ridge

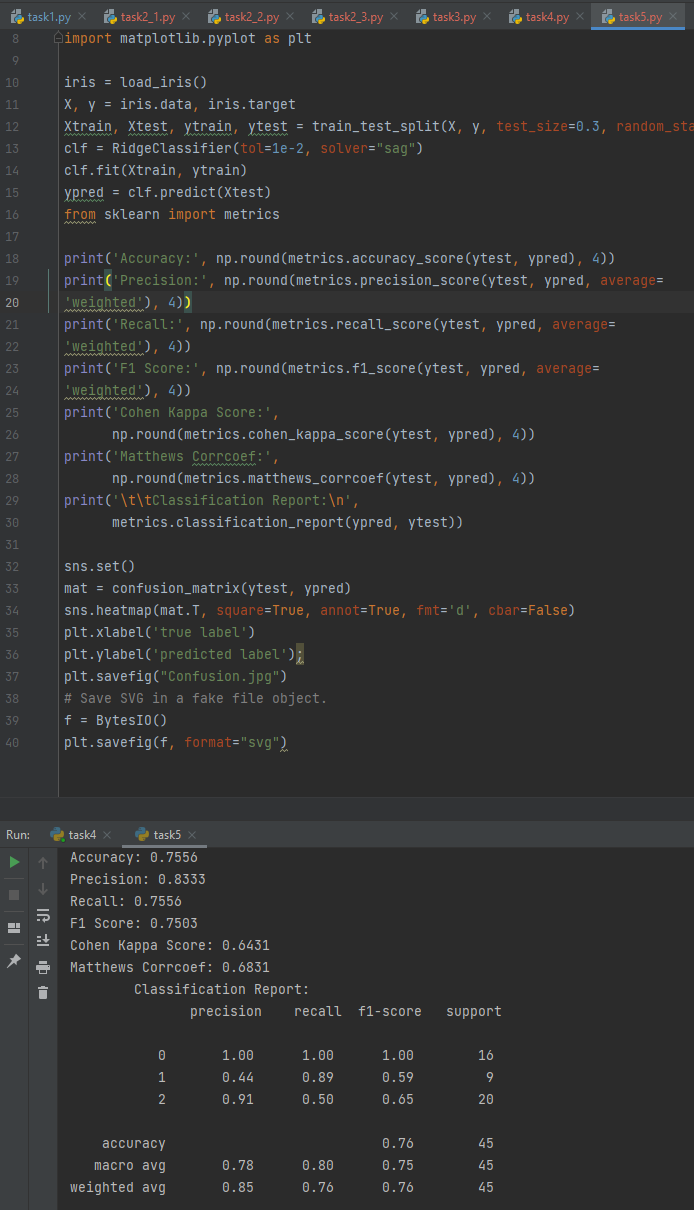


Рисунок 8. Класифікатор Ridge

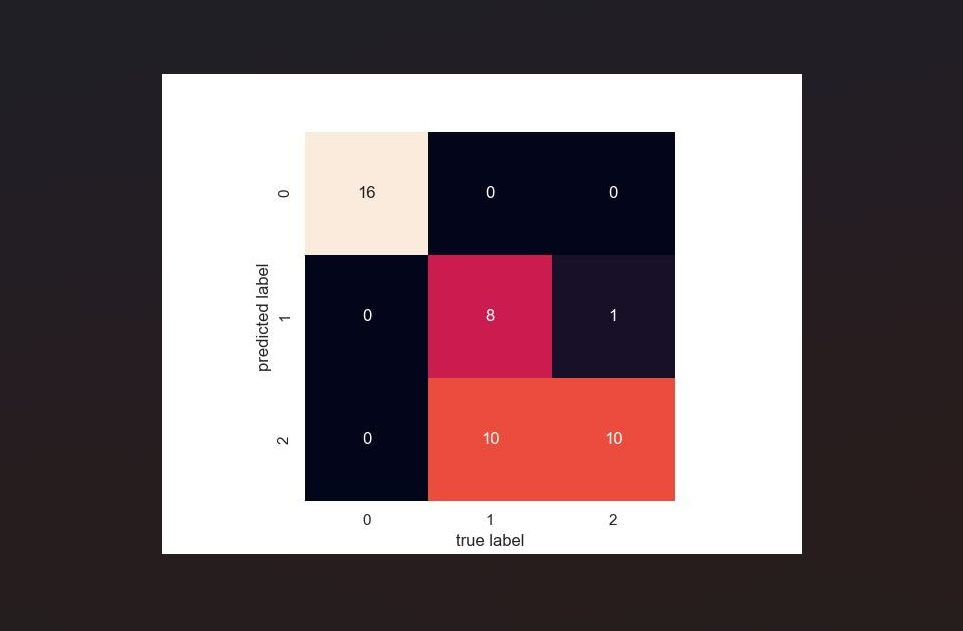


Рисунок 9. Confusion.jpg

Класифікатор має наступні параметри:

• tol – точність класифікації

• solver – алгоритм, який виконує класифікацію

На зображені Confusion.jpg наведені результати класифікації. На вертикальній шкалі відкладені наявні класи ірису в числовій репрезентації, а на горизонтальній передбаченя класи ірису. Цифра на перетині – кількість результатів

системи при справжньому і передбаченому класі.

Коефіцієнт кореляції Метьюза – коефіцієнт, який на основі матриці помилок вираховує коефіцієнт від -1 до 1, де 1 – є результатом ідеальної класифікації, а 0 – рівень випадкового вибору.

Коефіцієнт Коена Каппа – коефіцієнт, якй також за основу бере матрицю

помилок, але замість загальної якості, звертає увагу на нерівноцінне

розподілення класів.

**Висновок:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.