Лабораторна робота №1

Тема: Попередня обробка та контрольована класифікація даних

Посилання на гітхаб:https://github.com/ViMIMercurysMight/python-ai

Завдання 1.1**.** Попередня обробка даних

L1-нормалізація відрізняється від L2-нормалізації тим, що перша використовує метод найменших абсолютних відхилень і це забезпечує рівність 1, а друга використовує метод найменших квадратів, що забезпечує рівність 1 суми квадратів значень

*Лістинг програми:*

import numpy as np  
from sklearn import preprocessing  
2  
input\_data = np.array([[5.1, -2.9, 3.3],  
 [-1.2, 7.8, -6.1],  
 [3.9, 0.4, 2.1],  
 [7.3, -9.9, -4.5]])  
  
# Бінаризація даних  
data\_binarized = preprocessing.Binarizer(threshold=2.1).transform(input\_data)  
print("\n Binarized data:\n", data\_binarized)  
  
# Виведення середнього значення та стандартного відхилення  
print("\nBEFORE: ")  
print("Mean =", input\_data.mean(axis=0))  
print("Std deviation =", input\_data.std(axis=0))  
  
# Исключение среднего  
data\_scaled = preprocessing.scale(input\_data)  
print("\nAFTER: ")  
print("Mean =", data\_scaled.mean(axis=0))  
print("Std deviation =", data\_scaled.std(axis=0))  
  
# Масштабування MinМax  
data\_scaler\_minmax = preprocessing.MinMaxScaler(feature\_range=(0, 1))  
data\_scaled\_minmax = data\_scaler\_minmax.fit\_transform(input\_data)  
print("\nМin max scaled data:\n", data\_scaled\_minmax)  
  
# Нормалізація даних  
data\_normalized\_l1 = preprocessing.normalize(input\_data, norm='l1')  
data\_normalized\_l2 = preprocessing.normalize(input\_data, norm='l2')

print("\nl1 normalized data:\n", data\_normalized\_l1)  
print("\nl2 normalized data:\n", data\_normalized\_l2)

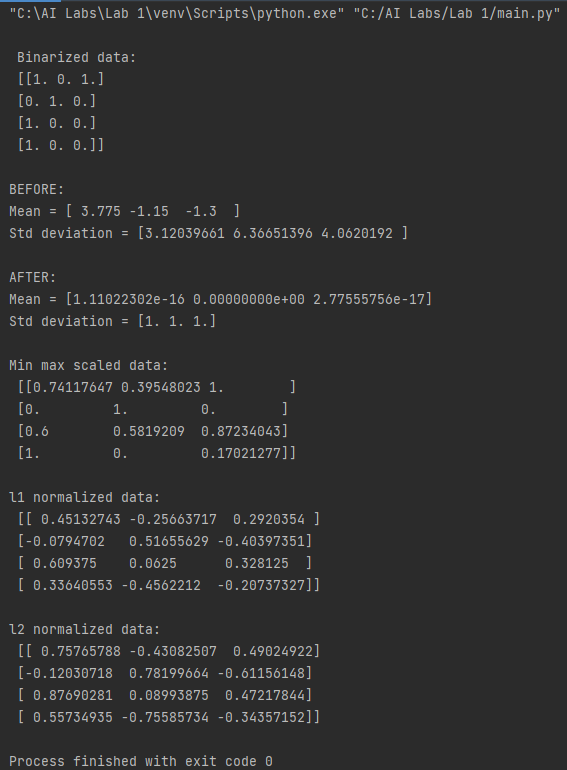


Рис. 1 Результат виконання

Завдання 2. Кодування міток

*Лістинг програми:*

import numpy as np  
from sklearn import preprocessing  
  
# Надання позначок вхідних даних  
Input\_labels = ['red', 'Ыасk', 'red', 'green', 'Ьlack', 'yellow', 'white']  
  
# Створення кодувальника та встановлення відповідності  
# між мітками та числами  
encoder = preprocessing.LabelEncoder()  
encoder.fit(Input\_labels)  
  
# Виведення відображення  
print("\nLabel mapping:")  
for i, item in enumerate(encoder.classes\_):  
 print(item, '-->', i)  
  
# перетворення міток за допомогою кодувальника  
test\_labels = ['green', 'red', 'Ыасk']  
encoded\_values = encoder.transform(test\_labels)  
print("\nLabels =", test\_labels)  
print("Encoded values =", list(encoded\_values))  
  
# Декодування набору чисел за допомогою декодера  
encoded\_values = [3, 0, 4, 1]  
decoded\_list = encoder.inverse\_transform(encoded\_values)  
print("\nEncoded values =", encoded\_values)  
print("Decoded labels =", list(decoded\_list))

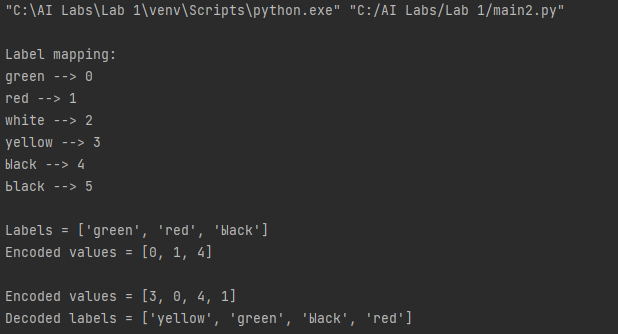


Рис.2 Результат виконання

Після процесу кодування кожної словесої мітки була отримана своя числова форма.

Завдання 2.2. Попередня обробка нових даних



*Лістинг програми:*

import numpy as np  
from sklearn import preprocessing  
  
input\_data = np.array([[-5.3, -8.9, 3.0],  
 [2.9, 5.1, -3.3],  
 [3.1, -2.8, -3.2],  
 [2.2, -1.4, 5.1]])  
  
# Бінаризація даних  
data\_binarized = preprocessing.Binarizer(threshold=2.0).transform(input\_data)  
print("\n Binarized data:\n", data\_binarized)  
  
# Виведення середнього значення та стандартного відхилення  
print("\nBEFORE: ")  
print("Mean =", input\_data.mean(axis=0))  
print("Std deviation =", input\_data.std(axis=0))  
  
# Исключение среднего  
data\_scaled = preprocessing.scale(input\_data)  
print("\nAFTER: ")  
print("Mean =", data\_scaled.mean(axis=0))  
print("Std deviation =", data\_scaled.std(axis=0))  
  
# Масштабування MinМax  
data\_scaler\_minmax = preprocessing.MinMaxScaler(feature\_range=(0, 1))  
data\_scaled\_minmax = data\_scaler\_minmax.fit\_transform(input\_data)  
print("\nМin max scaled data:\n", data\_scaled\_minmax)  
  
# Нормалізація даних  
data\_normalized\_l1 = preprocessing.normalize(input\_data, norm='l1')  
data\_normalized\_l2 = preprocessing.normalize(input\_data, norm='l2')  
print("\nl1 normalized data:\n", data\_normalized\_l1)  
print("\nl2 normalized data:\n", data\_normalized\_l2)

Text

Description automatically generated

Рисунок 3. Результат виконання

Завдання 2.3. Класифікація логістичною регресією або логістичний класифікатор

*Лістинг програми:*

import numpy as np  
from sklearn import linear\_model  
import matplotlib.pyplot as plt  
from utilities import visualize\_classifier  
  
# Визначення зразка вхідних даних  
X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5],  
 [6, 5], [5.6, 5], [3.3, 0.4],  
 [3.9, 0.9], [2.8, 1],  
 [0.5, 3.4], [1, 4], [0.6, 4.9]])  
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])  
  
# Створення логістичного класифікатора  
classifier = linear\_model.LogisticRegression(solver='liblinear', C=1)  
  
# Тренування класифікатора  
classifier.fit(X, y)  
visualize\_classifier(classifier, X, y)

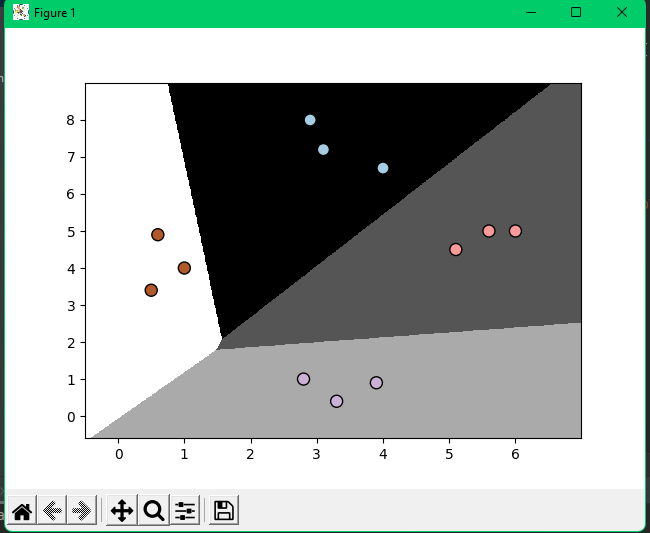


Рис. 4 Результат виконання

Завдання 2.4. Класифікація наївним байєсовським класифікатором

*Лістинг програми:*

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.model\_selection import train\_test\_split  
from utilities import visualize\_classifier  
  
# Вхідний файл, який містить дані  
input\_file = 'data\_multivar\_nb.txt'  
  
# Завантаження даних із вхідного файлу  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Створення наївного байєсовського класифікатора  
classifier = GaussianNB()  
  
# Тренування класифікатора  
classifier.fit(X, y)  
  
# Прогнозування значень для тренувальних даних  
y\_pred = classifier.predict(X)  
  
# Обчислення якості класифікатора  
accuracy = 100.0 \* (y == y\_pred).sum() / X.shape[0]  
print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")  
  
# Візуалізація результатів роботи класифікатора  
visualize\_classifier(classifier, X, y)

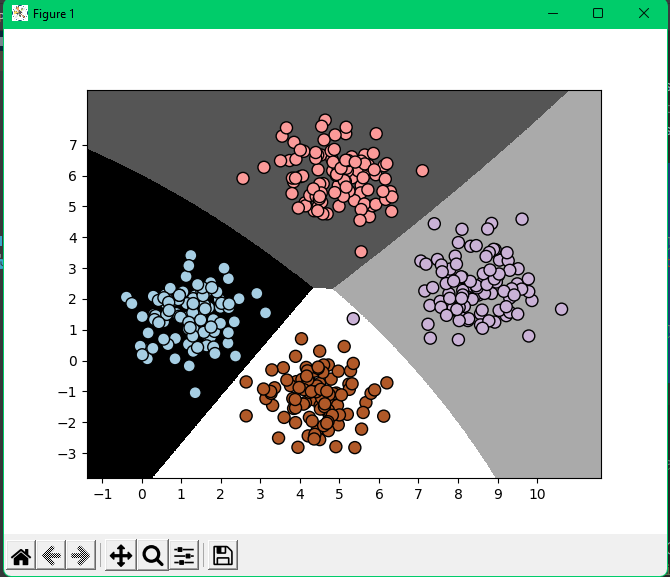


Рис. 5 Результат виконання

*Лістинг програми:*

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.model\_selection import train\_test\_split  
from utilities import visualize\_classifier  
  
# Вхідний файл, який містить дані  
input\_file = 'data\_multivar\_nb.txt'  
  
# Завантаження даних із вхідного файлу  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Створення наївного байєсовського класифікатора  
classifier = GaussianNB()  
  
# Тренування класифікатора  
classifier.fit(X, y)  
  
# Прогнозування значень для тренувальних даних  
y\_pred = classifier.predict(X)  
  
# Обчислення якості класифікатора  
accuracy = 100.0 \* (y == y\_pred).sum() / X.shape[0]  
print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")  
  
# Візуалізація результатів роботи класифікатора  
visualize\_classifier(classifier, X, y)  
  
# Розбивка даних на навчальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=3)  
classifier\_new = GaussianNB()  
classifier\_new.fit(X\_train, y\_train)  
y\_test\_pred = classifier\_new.predict(X\_test)  
  
# Обчислення якості класифікатора  
accuracy = 100.0 \* (y\_test == y\_test\_pred).sum() / X\_test.shape[0]  
print("Accuracy of the new classifier =", round(accuracy, 2), "%")  
  
# Візуалізація роботи класифікатора  
visualize\_classifier(classifier\_new, X\_test, y\_test)  
  
num\_folds = 3  
accuracy\_values = train\_test\_split.cross\_val\_score(classifier, X, y, scoring='accuracy', cv=num\_folds)  
print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
precision\_values = train\_test\_split.cross\_val\_score(classifier, X, y, scoring='precision\_weighted', cv=num\_folds)  
print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
recall\_values = train\_test\_split.cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=num\_folds)  
print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")  
f1\_values = train\_test\_split.cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=num\_folds)  
print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")

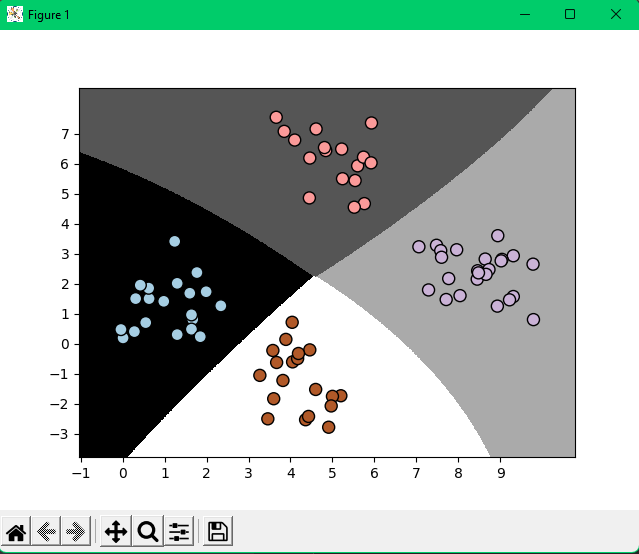


Рис. 6 Результат виконання

Завдання 2.5. Вивчити метрики якості класифікації

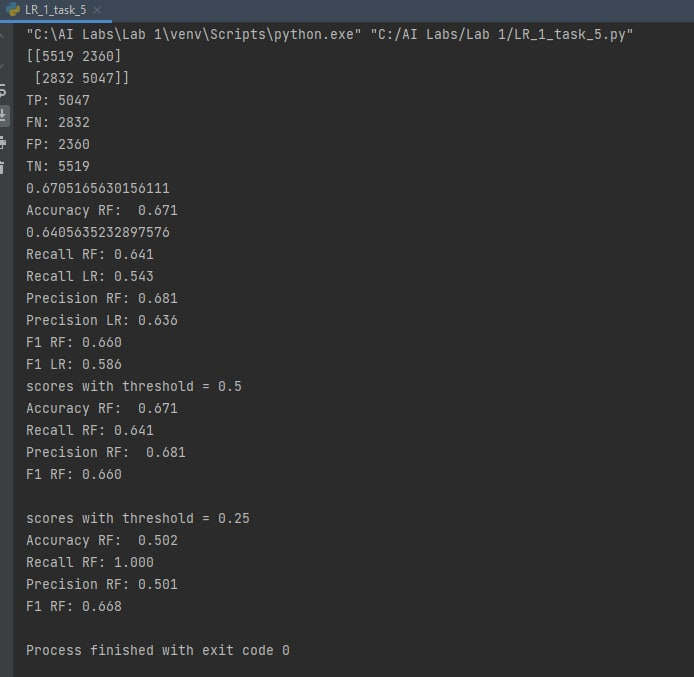


Рис. 7Результат для порогу 0,25

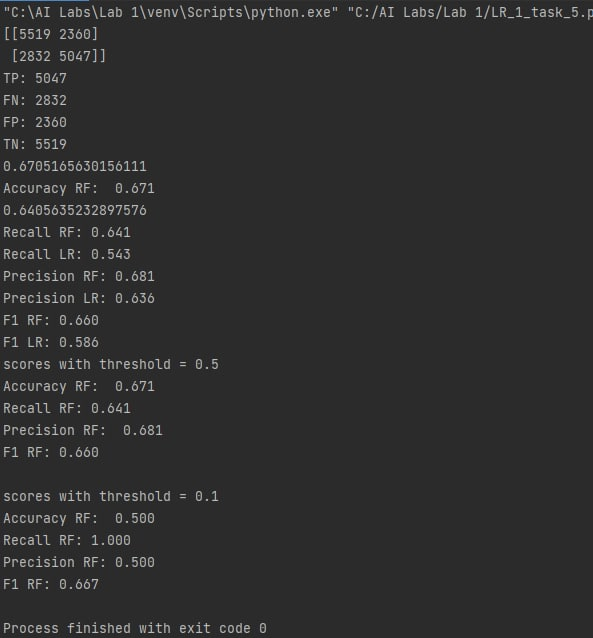


Рис. 8 Результат для порогу 0.10

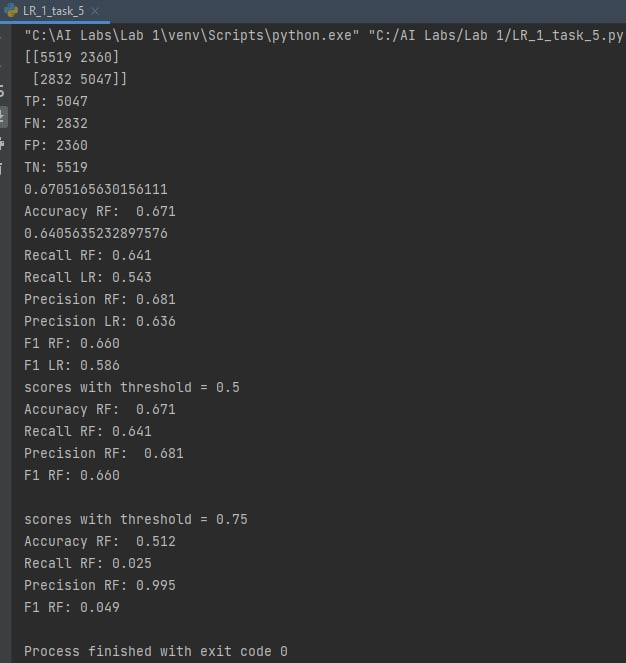


Рис. 9 Результат для порогу 0.75

***Висновки:*** в результаті збільшення порогу, F1 міра зменшується.

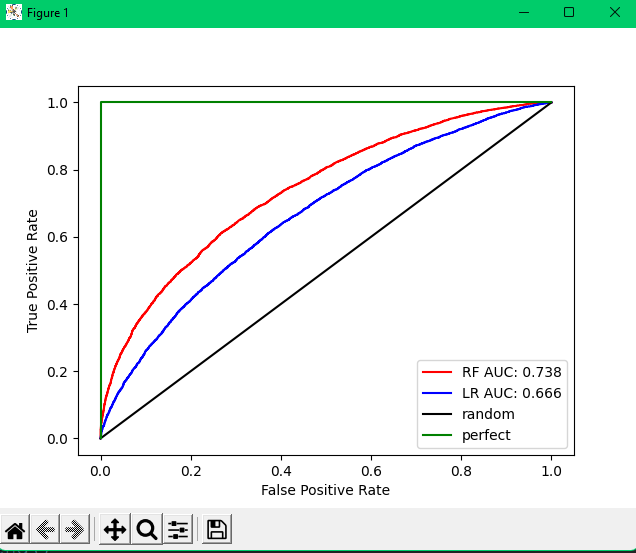
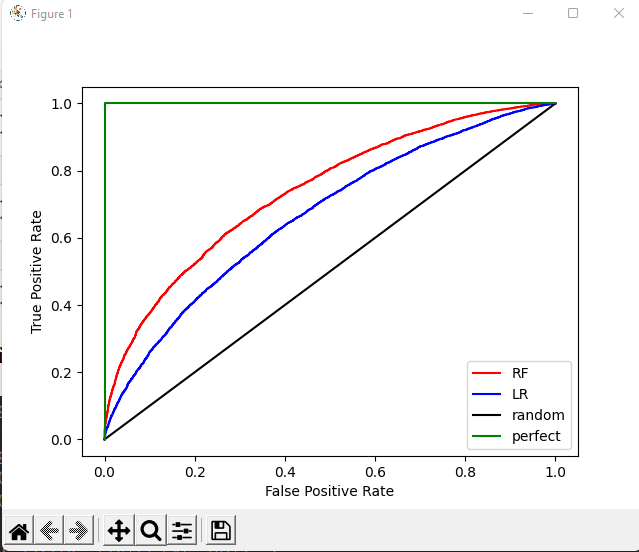


Рис. 10 ROC-крива

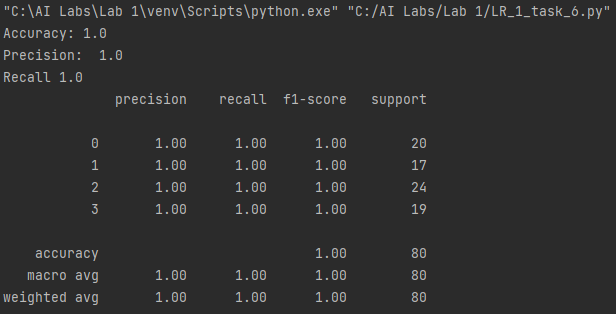
Як бачимо з графіку RF модель має більшу точність, аніж LR модель. Звісно, що можуть бути ситуації, коли LR має переваги перед RF, але я думаю, що важливіше, що слід враховувати, — це складність моделі. У лінійних моделей дуже мало параметрів, у RF набагато більше. Це означає, що випадкові ліси будуть легше переміщуватися, ніж LR.

Завдання 2.6. Розробіть програму класифікації даних в файлі data\_multivar\_nb.txt за допомогою машини опорних векторів (Support Vector Machine - SVМ). Розрахуйте показники якості класифікації. Порівняйте їх з показниками наївного байєсівського класифікатора. Зробіть висновки яку модель класифікації краще обрати і чому

Так як в методичних рекомендаціях до першої роботи не було наведено як формувати SVM інформацію було узято з відкритих джерел.

Лістинг програми**:**

import numpy as np  
from sklearn import datasets  
from sklearn.model\_selection import train\_test\_split  
from sklearn import svm  
from sklearn import metrics  
  
# Вхідний файл, який містить дані  
from utilities import visualize\_classifier  
  
input\_file = 'data\_multivar\_nb.txt'  
  
# Завантаження даних із вхідного файлу  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y.astype(int), test\_size=0.2, random\_state=3)  
  
cls = svm.SVC(kernel='linear')  
  
cls.fit(X\_train, y\_train)  
  
pred = cls.predict(X\_test)  
  
print("Accuracy:", metrics.accuracy\_score(y\_test, y\_pred=pred))  
  
print("Precision: ", metrics.precision\_score(y\_test, y\_pred=pred, average='macro'))  
  
print("Recall", metrics.recall\_score(y\_test, y\_pred=pred, average='macro'))  
print(metrics.classification\_report(y\_test, y\_pred=pred))  
  
visualize\_classifier(cls, X\_test, y\_test)



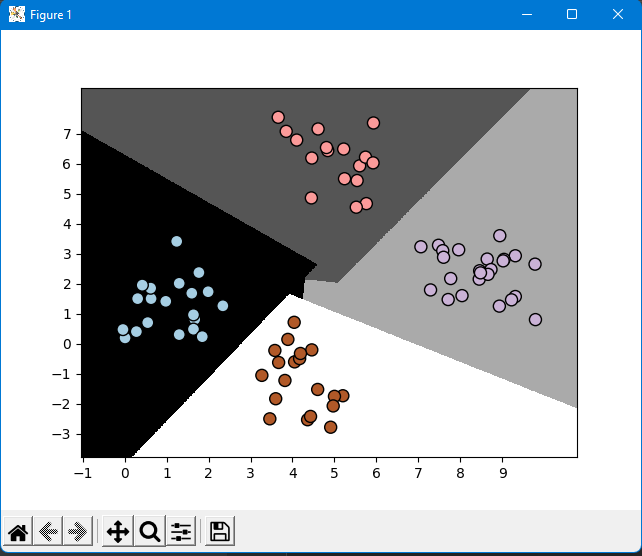


Рис. 13 Графічне відображення

**Результат порівняння:** Наївний класифікатор байесовського і метод опорних векторів (SVM) мають різні параметри, включаючи вибір функції ядра для кожного з них (показано в лабораторній 2). Обидва алгоритми є дуже чутливими до оптимізації параметрів, тобто вибір різних параметрів може суттєво змінити їхній вихід. Результат наразі показує, що NBC працює краще, ніж SVM, однак це вірно тільки для вибраних поточних налаштувань системи користувачем і може змінитися при зміні вхідного набору даних.

**Висновок:** при виконанні лабораторної роботи було досліджено попередню обробку та класифікацію даних з використанням спеціалізованих бібліотек та мови програмування Python.