**ЛАБОРАТОРНА РОБОТА № 1**   
ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ   
 **Мета роботи:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити попередню обробку та класифікацію даних.   
**1. ТЕОРЕТИЧНІ ВІДОМОСТІ**

Теоретичні відомості подані на лекціях. Також доцільно вивчити матеріал поданий в літературі:

Джоши Пратик. Искусственный интеллект с примерами на Python. : Пер. с англ. - СПб. : ООО "Диалектика", 2019. - 448 с. - Парал. тит. англ. ISBN 978-5-907114-41-8 (рус.)

Можна використовувати Google Colab або Jupiter Notebook

**2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ**

**Завдання 2.1. Попередня обробка даних**

import numpy as np

from sklearn import preprocessing

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)

input\_labels = ['red', 'Ыасk', 'red', 'green', 'Ьlack', 'yellow', 'white']

encoder = preprocessing.LabelEncoder()

encoder.fit(input\_labels)

print("\nLabel mapping:")

for i, item in enumerate(encoder.classes\_) : print(item, '-->', i)

test\_labels = ['green', 'red', 'Ыасk']

encoded\_values = encoder.transform(test\_labels)

print("\nLabels =", test\_labels)

print("Encoded values =", list (encoded\_values ))

encoded\_values = [3, 0, 4, 1]

decoded\_list = encoder.inverse\_transform(encoded\_values)

print("\nEncoded values =", encoded\_values)

print("Decoded labels =", list (decoded\_list ))

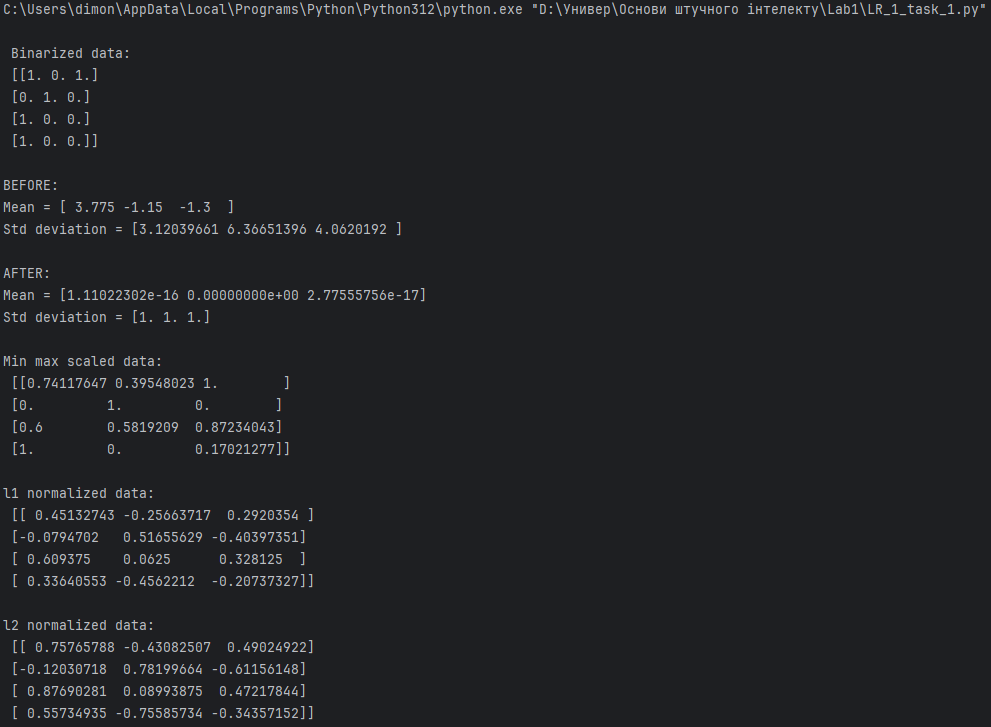


Рис.1 Результат виконання бінаризації,виведення середнього значення та стандартного відхилення, виключення середнього,масштабування, нормалізація.

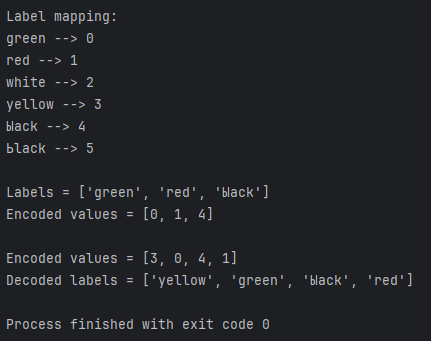


Рис.2 Результат перетворень словесних міток в числа

**Завдання 2.2. Попередня обробка нових даних**



import numpy as np  
from sklearn import preprocessing

# Варіант 2  
input\_data = np.array([[4.1, -5.9, -3.5], [-1.9, 4.6, 3.9], [-4.2, 6.8, 6.3], [3.9, 3.4, 1.2]])  
  
# Бінаризація даних  
data\_binarized = preprocessing.Binarizer(threshold=3.2).transform(input\_data)  
print("\nBinarized 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))  
  
data\_scaled = preprocessing.scale(input\_data)  
  
# Исключение среднего  
print("\nAFTER:")  
print("Mean =", data\_scaled.mean(axis=0))  
print("Std deviation =", data\_scaled.std(axis=0))  
  
# Масштабування MinMax  
data\_scaler\_minmax = preprocessing.MinMaxScaler(feature\_range=(0, 1))  
data\_scaled\_minmax = data\_scaler\_minmax.fit\_transform(input\_data)  
print("\nMin 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)

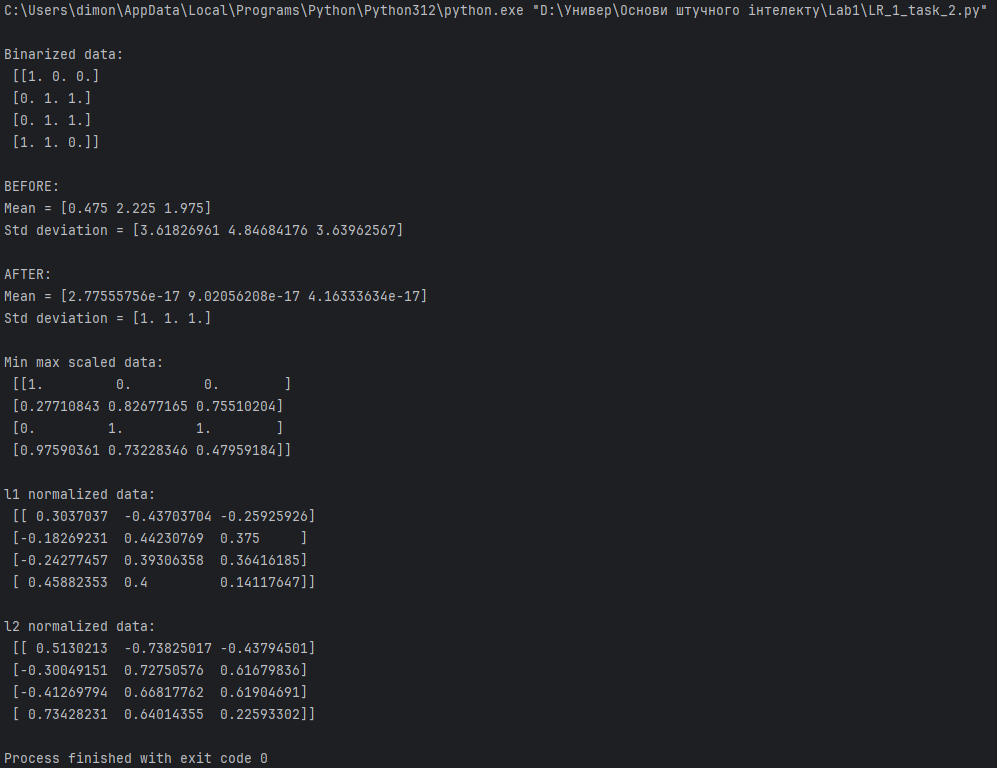


Рис.3 Результат виконання бінаризації,виведення середнього значення та стандартного відхилення, виключення середнього,масштабування, нормалізація згідно варіанту

**Завдання 2.3. Класифікація логістичною регресією або логістичний класифікатор**

import numpy as np

from sklearn import linear\_model

import matplotlib.pyplot as plt

from utilities import visualize\_classifier

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])

classifier = linear\_model.LogisticRegression(solver='liblinear',C=1)

classifier.fit(X, y)

visualize\_classifier(classifier, X, y)

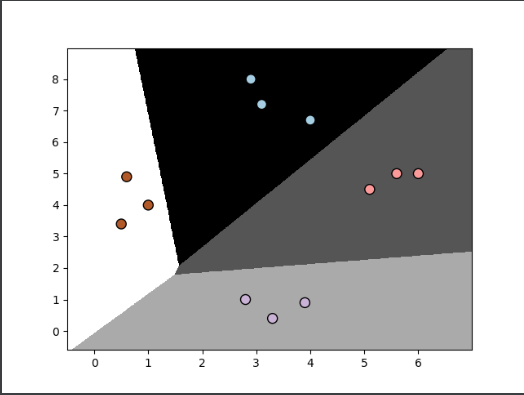


Рис.4 Результат виконання класифікації логістичної регресіїї та логістичної класифікації

**Завдання 2.4. Класифікація наївним байєсовським класифікатором**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.naive\_bayes import GaussianNB

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from utilities import visualize\_classifier

input\_file = 'data\_multivar\_nb.txt'

data = np.loadtxt(input\_file, delimiter=',')

X, y = data[:, :-1], data[:, -1]

classifier = GaussianNB()

classifier.fit(X, y)

y\_pred = classifier.predict(X)

accuracy = 100.0 \* (y == y\_pred).sum() / X.shape[0]

print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")

visualize\_classifier(classifier, X, y)

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)

accuracy = 100.0 \* (y\_test == y\_test\_pred).sum() / X\_test.shape[0]

print("Accuracy of the new classifier =", round(accuracy, 2), "%")

visualize\_classifier(classifier\_new, X\_test, y\_test)

num\_folds = 3

accuracy\_values = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=num\_folds)

print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")

precision\_values = cross\_val\_score(classifier, X, y, scoring='precision\_weighted', cv=num\_folds)

print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")

recall\_values = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=num\_folds)

print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")

f1\_values = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=num\_folds)

print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")

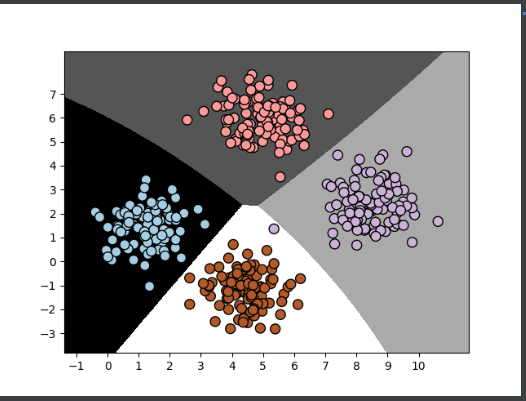


Рис.6 Результат виконання першого прогону та зображення результату класифікації

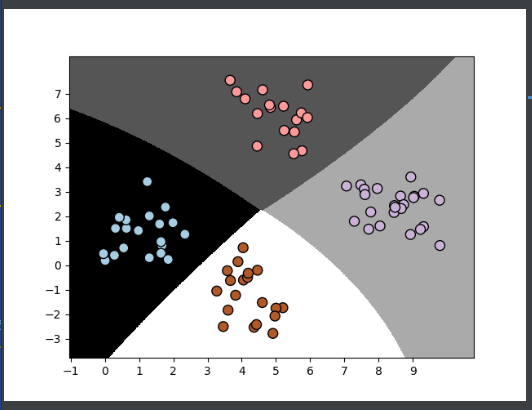
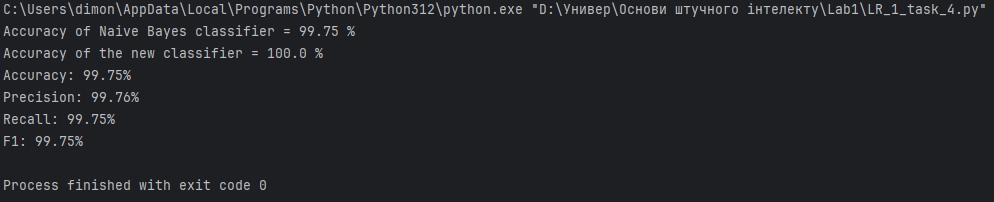


Рис.7 Результат виконання другого прогону та зображення результату класифікації

ДО та ПІСЛЯ прогонів:



Висновок: У порівнянні першого та другого прогону видно, що другий виглядає більш точним, оскільки він базується на результаті потрійної перехресної перевірки, яка враховує точність, повноту та якість.

**Завдання 2.5. Вивчити метрики якості класифікації**

import pandas as pd

df = pd.read\_csv('data\_metrics.csv')

print(df.head())

thresh = 0.5

df['predicted\_RF'] = (df.model\_RF >= 0.5).astype('int')

df['predicted\_LR'] = (df.model\_LR >= 0.5).astype('int')

df.head()

from sklearn.metrics import confusion\_matrix

def true\_positives(confusion\_matrix):

return confusion\_matrix[0, 0]

def false\_positives(confusion\_matrix):

return confusion\_matrix[1, 0]

def false\_negatives(confusion\_matrix):

return confusion\_matrix[0, 1]

def true\_negatives(confusion\_matrix):

return confusion\_matrix[1, 1]

confusion\_matrix = confusion\_matrix(df.actual\_label.values, df.predicted\_RF.values)

print("Істинно позитивні:", true\_positives(confusion\_matrix))

print("Хибні позитивні:", false\_positives(confusion\_matrix))

print("Істинно негативні:", true\_negatives(confusion\_matrix))

def kulieshov\_accuracy\_score(confusion\_matrix):

true\_positives = confusion\_matrix[0, 0]

true\_negatives = confusion\_matrix[1, 1]

return (true\_positives + true\_negatives) / (

true\_positives + true\_negatives + confusion\_matrix[0, 1] + confusion\_matrix[1, 0]

)

print("Точність:", kulieshov\_accuracy\_score(confusion\_matrix))

def kulieshov\_recall\_score(confusion\_matrix):

true\_positives = confusion\_matrix[0, 0]

possible\_positives = confusion\_matrix[0, 0] + kulieshov\_confusion\_matrix[0, 1]

return true\_positives / possible\_positives

print("Відкликання:", kulieshov\_recall\_score(confusion\_matrix))

def kulieshov\_precision\_score(confusion\_matrix):

true\_positives = confusion\_matrix[0, 0]

predicted\_positives = confusion\_matrix[0, 0] + confusion\_matrix[0, 1]

return true\_positives / predicted\_positives

print("Точність:", kulieshov\_precision\_score(confusion\_matrix))

from sklearn.metrics import roc\_curve

fpr\_RF, tpr\_RF, thresholds\_RF = roc\_curve(df.actual\_label.values, df.model\_RF.values)

fpr\_LR, tpr\_LR, thresholds\_LR = roc\_curve(df.actual\_label.values, df.model\_LR.values)

import matplotlib.pyplot as plt

plt.plot(fpr\_RF, tpr\_RF,'r-',label = 'RF')

plt.plot(fpr\_LR,tpr\_LR,'b-', label= 'LR')

plt.plot([0,1],[0,1],'k-',label='random')

plt.plot([0,0,1,1],[0,1,1,1],'g-',label='perfect')

plt.legend()

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

from sklearn.metrics import roc\_auc\_score

auc\_RF = roc\_auc\_score(df.actual\_label.values, df.model\_RF.values)

auc\_LR = roc\_auc\_score(df.actual\_label.values, df.model\_LR.values)

print('AUC RF:%.3f'% auc\_RF)

print('AUC LR:%.3f'% auc\_LR)

plt.plot(fpr\_RF, tpr\_RF,'r-',label = 'RF AUC: %.3f'%auc\_RF)

plt.plot(fpr\_LR,tpr\_LR,'b-', label= 'LR AUC: %.3f'%auc\_LR)

plt.plot([0,1],[0,1],'k-',label='random')

plt.plot([0,0,1,1],[0,1,1,1],'g-',label='perfect')

plt.legend()

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

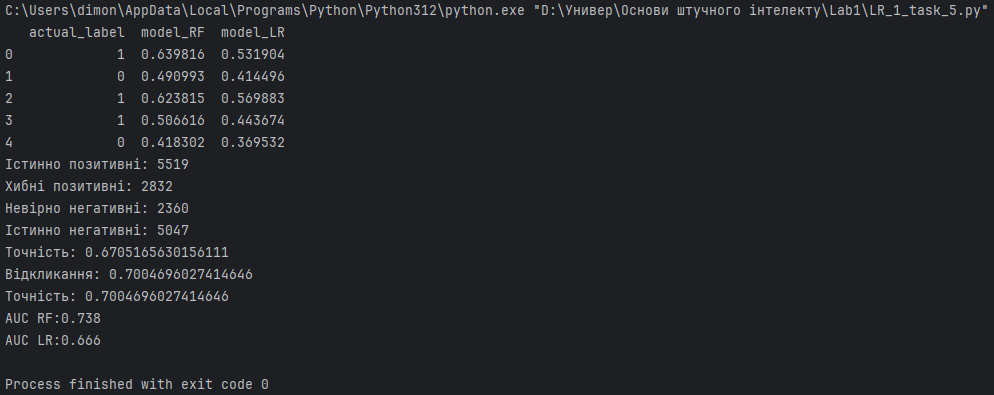


Рис.8 Результат повернення трьох списків шляхом виклику функції roc\_curve для LR та RF

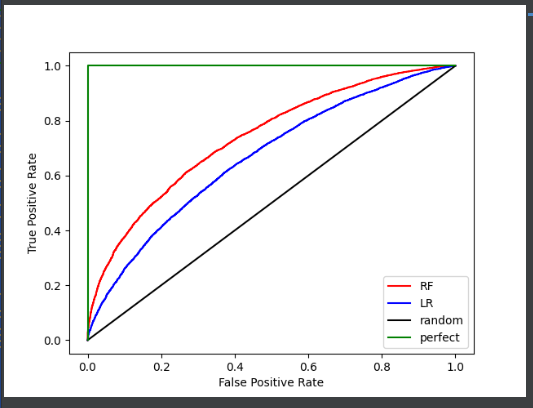


Рис.9 Побудова ROC-кривої для кожної моделі.

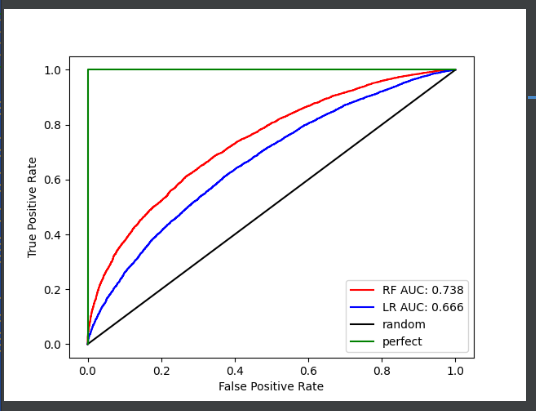


Рис.10 Знаходження метрикі плоз під кривою

Висновок: У ході аналізу виявлено, що випереджує за результатами дослідження RF (AUC = 0,738) в порівнянні з LR (AUC = 0,666). Це пояснюється тим, що крива ROC для RF охоплює більшу площу, вказуючи на вищий показник TRP (чутливість), ніж у випадку з LR.

**Завдання 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**

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

# Завантажимо дані

input\_file = 'data\_multivar\_nb.txt'

data = np.loadtxt(input\_file, delimiter=',')

# Розділимо дані на навчальний та тестовий набори

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data[:, :-1], data[:, -1], test\_size=0.2, random\_state=42)

# Підготуємо класифікатор

svm\_classifier = SVC(kernel='linear', C=1.0)

# Підіб'ємо модель

svm\_classifier.fit(X\_train, y\_train)

# Прогнозуємо класи на тестовому наборі

y\_pred\_svm = svm\_classifier.predict(X\_test)

# Розрахуємо показники якості класифікації

accuracy\_svm = accuracy\_score(y\_test, y\_pred\_svm)

precision\_svm = precision\_score(y\_test, y\_pred\_svm, average='micro')

recall\_svm = recall\_score(y\_test, y\_pred\_svm, average='micro')

f1\_score\_svm = f1\_score(y\_test, y\_pred\_svm, average='micro')

confusion\_matrix\_svm = confusion\_matrix(y\_test, y\_pred\_svm)

print("Машини опорних векторів (SVM):")

print("Accuracy:", accuracy\_svm)

print("Precision:", precision\_svm)

print("Recall:", recall\_svm)

print("F1 Score:", f1\_score\_svm)

print("Confusion Matrix:")

print(confusion\_matrix\_svm)

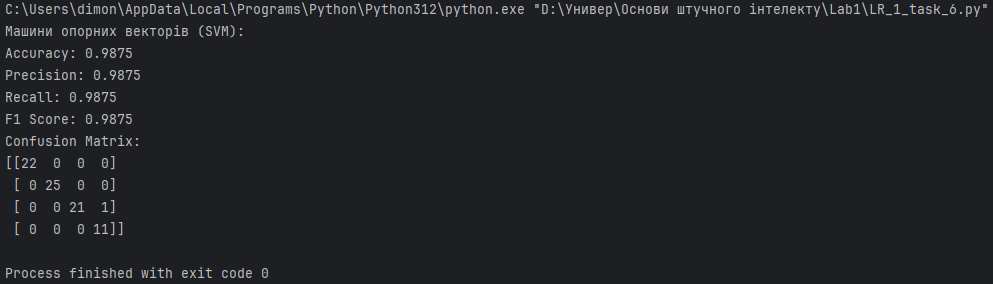


Рис.11 Результат класифікації даних.

Посилання на Git:

Висновок використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили попередню обробку та класифікацію даних.