**ЛАБОРАТОРНА РОБОТА № 1**

**ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ**

***Мета:*** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити попередню обробку та класифікацію даних

**Хід роботи**

**Посилання на GitHub:** <https://github.com/Max2002/AI_IPZ-19-3_LMV>

**Завдання 1.** Нормалізація даних. Кодування міток.

Лістинг файлу LR\_1\_task\_1.py:

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]])  
  
# 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)  
  
# Надання позначок вхідних даних  
input\_labels = ['red', 'black', 'red', 'green', 'black', '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', 'black']  
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))

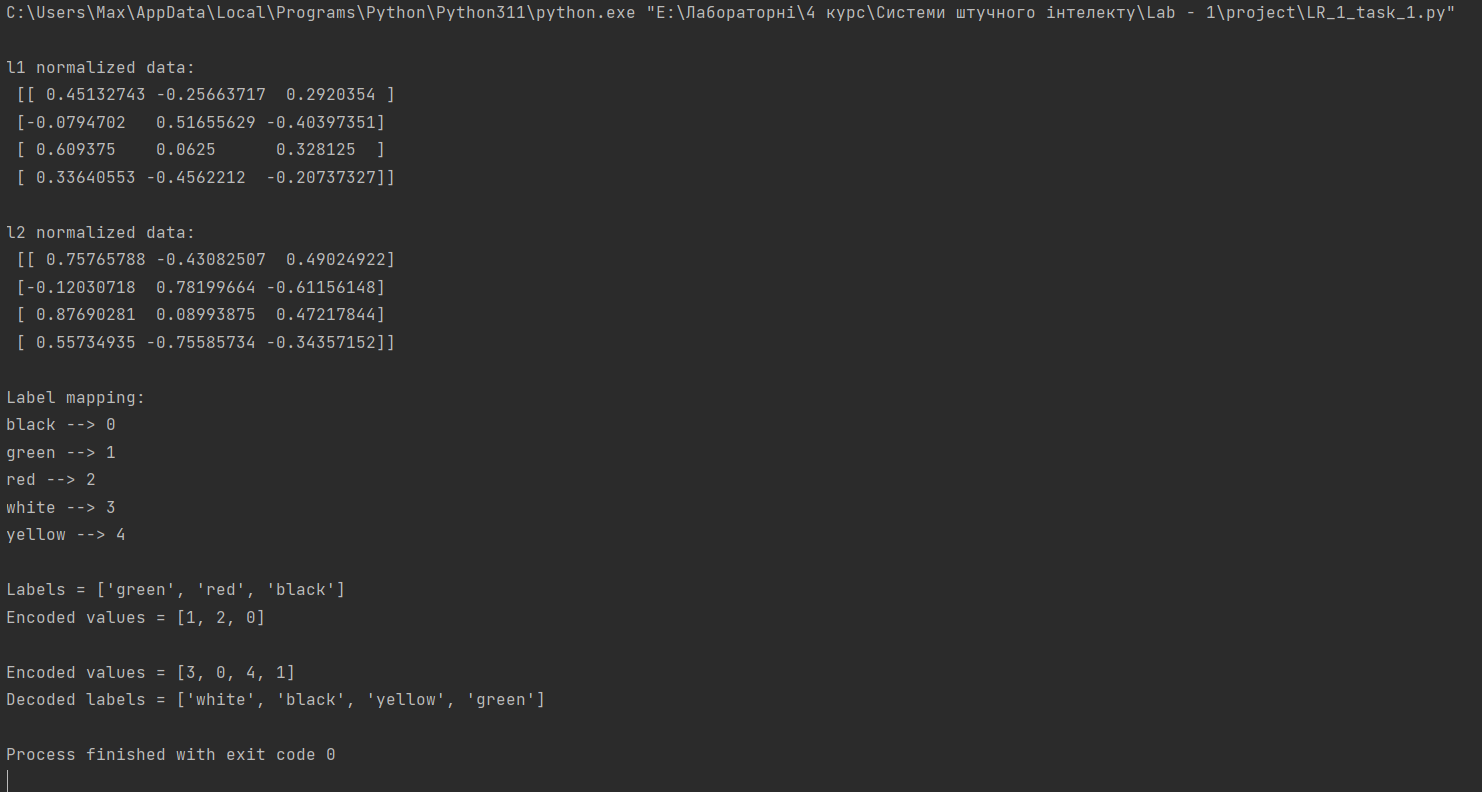


Рис. 1. Кодування міток.

Порівняння нормалізації L1 та L2: Нормалізація L2 має результат в цілих числах, тому вона не така точна в порівняні з нормалізацією L1, але L1 не дозволяє вирішувати завдання, де необхідно простежувати неточність вхідних даних (викиди).

**Завдання 2.** Попередня обробка нових даних.

Лістинг файлу LR\_1\_task\_2.py:

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=3.0).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)

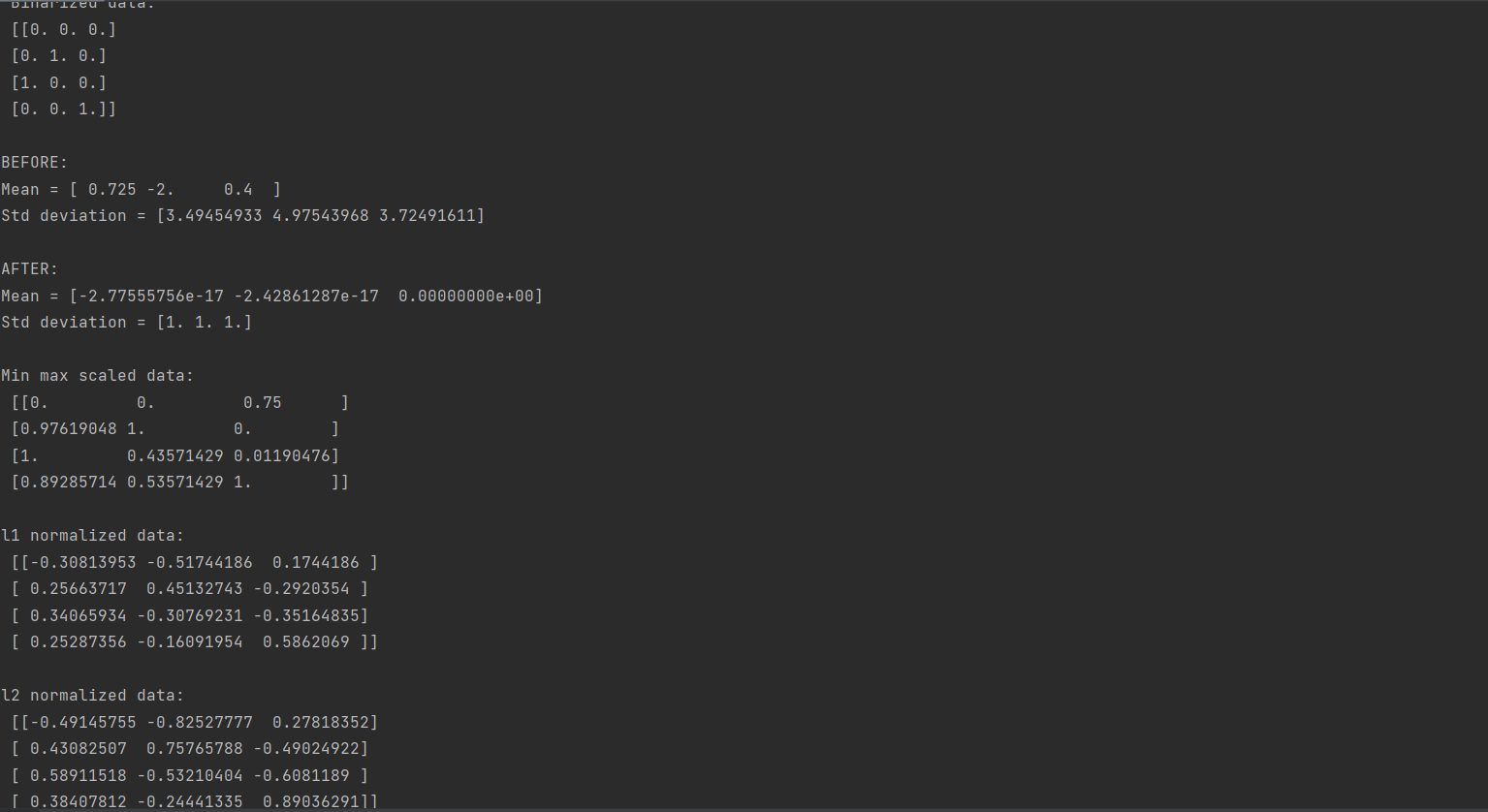


Рис. 2. Результат виконання завдання.

**Завдання 3.** Класифікація логістичною регресією або логістичний класифікатор.

Лістинг файлу LR\_1\_task\_3.py:

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)

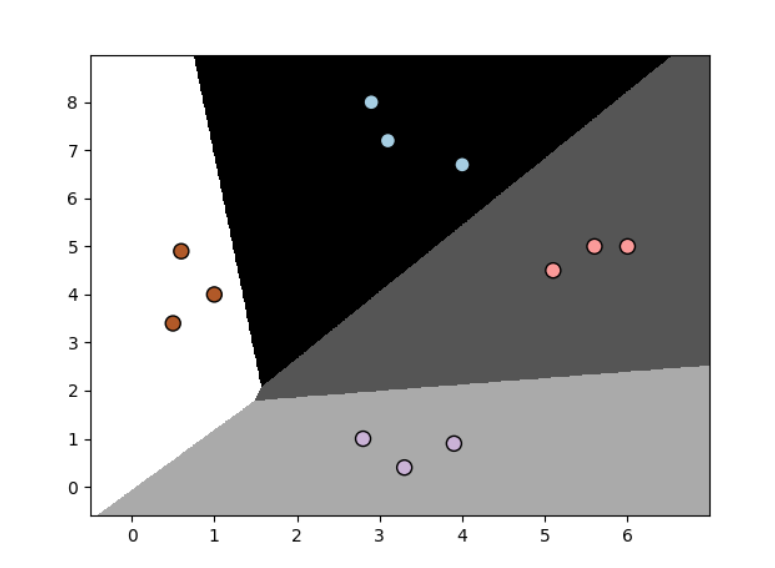


Рис. 3. Результат класифікації лінійною регресією.

**Завдання 4.** Класифікація наївним байєсовським класифікатором.

Лістинг файлу LR\_1\_task\_4.py:

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  
  
# 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 = 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)) + "%")

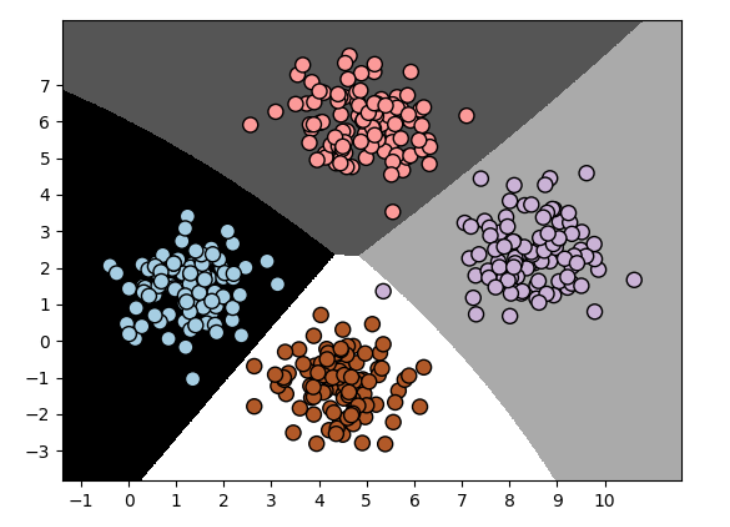


Рис. 4. Відображення результату класифікації

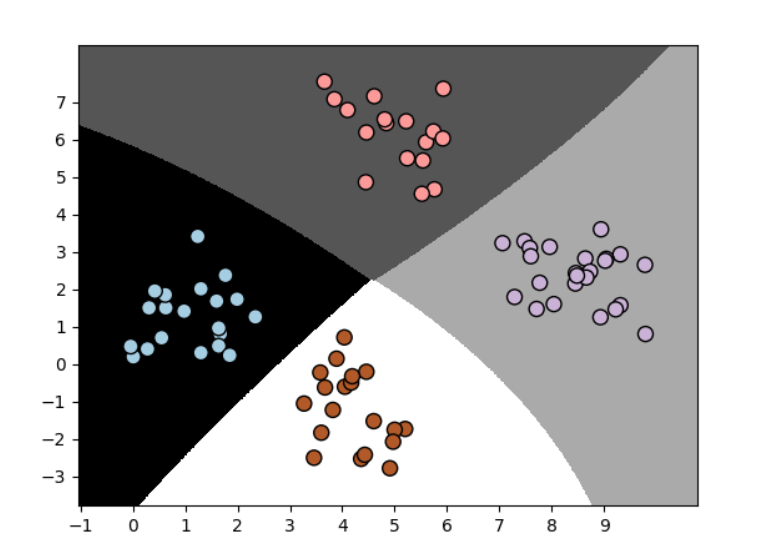


Рис. 5. Зображення результату класифікації тестових даних

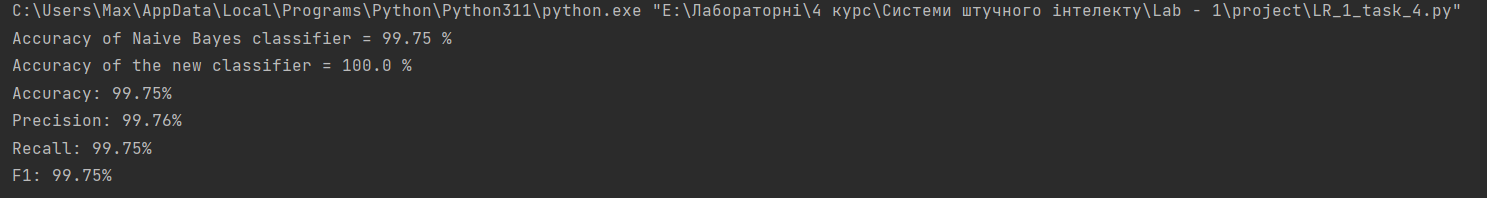


Рис. 6. Дані про якість класифікатора

**Завдання 5.** Вивчити метрики якості класифікації

Лістинг файлу LR\_1\_task\_5.py:

import pandas as pd  
import numpy as np  
from sklearn.metrics import confusion\_matrix, accuracy\_score, recall\_score, precision\_score, f1\_score, \  
 roc\_curve, roc\_auc\_score  
import matplotlib.pyplot as plt  
  
df = pd.read\_csv('data\_metrics.csv')  
df.head()  
  
thresh = 0.5  
df['predicted\_RF'] = (df.model\_RF >= thresh).astype('int')  
df['predicted\_LR'] = (df.model\_LR >= thresh).astype('int')  
df.head()  
  
# confusion\_matix  
  
print(confusion\_matrix(df.actual\_label.values, df.predicted\_RF.values))  
  
  
def find\_TP(y\_true, y\_pred):  
 return sum((y\_true == 1) & (y\_pred == 1))  
  
  
def find\_FN(y\_true, y\_pred):  
 return sum((y\_true == 1) & (y\_pred == 0))  
  
  
def find\_FP(y\_true, y\_pred):  
 return sum((y\_true == 0) & (y\_pred == 1))  
  
  
def find\_TN(y\_true, y\_pred):  
 return sum((y\_true == 0) & (y\_pred == 0))  
  
  
print('TP:', find\_TP(df.actual\_label.values, df.predicted\_RF.values))  
print('FN:', find\_FN(df.actual\_label.values, df.predicted\_RF.values))  
print('FP:', find\_FP(df.actual\_label.values, df.predicted\_RF.values))  
print('TN:', find\_TN(df.actual\_label.values, df.predicted\_RF.values))  
  
  
def find\_conf\_matrix\_values(y\_true, y\_pred):  
 TP = find\_TP(y\_true, y\_pred)  
 FN = find\_FN(y\_true, y\_pred)  
 FP = find\_FP(y\_true, y\_pred)  
 TN = find\_TN(y\_true, y\_pred)  
 return TP, FN, FP, TN  
  
  
def liashuk\_confusion\_matrix(y\_true, y\_pred):  
 TP, FN, FP, TN = find\_conf\_matrix\_values(y\_true, y\_pred)  
 return np.array([[TN, FP], [FN, TP]])  
  
  
print(liashuk\_confusion\_matrix(df.actual\_label.values, df.predicted\_RF.values))  
  
assert np.array\_equal(liashuk\_confusion\_matrix(df.actual\_label.values, df.predicted\_RF.values),  
 confusion\_matrix(df.actual\_label.values,  
 df.predicted\_RF.values)), 'my\_confusion\_matrix() is not correct for RF'  
assert np.array\_equal(liashuk\_confusion\_matrix(df.actual\_label.values, df.predicted\_LR.values),  
 confusion\_matrix(df.actual\_label.values,  
 df.predicted\_LR.values)), 'my\_confusion\_matrix() is not correct for LR'  
  
  
# accuracy  
score = accuracy\_score(df.actual\_label.values, df.predicted\_RF.values)  
print("Accuracy score on RF:", score)  
  
  
def liashuk\_accuracy\_score(y\_true, y\_pred):  
 TP, FN, FP, TN = find\_conf\_matrix\_values(y\_true, y\_pred)  
 return (TP + TN) / (TP + FN + FP + TN)  
  
  
assert liashuk\_accuracy\_score(df.actual\_label.values, df.predicted\_RF.values) == accuracy\_score(df.actual\_label.values, df.predicted\_RF.values), \  
 'my accuracy\_score failed on RF'  
  
assert liashuk\_accuracy\_score(df.actual\_label.values, df.predicted\_LR.values) == accuracy\_score(df.actual\_label.values, df.predicted\_LR.values), \  
 'my accuracy\_score failed on LR'  
  
print("my accuracy score on RF:", liashuk\_accuracy\_score(df.actual\_label.values, df.predicted\_RF.values))  
print("my accuracy score on LR:", liashuk\_accuracy\_score(df.actual\_label.values, df.predicted\_LR.values))  
  
# Recall  
print('Recall score on RF:', recall\_score(df.actual\_label.values, df.predicted\_RF.values))  
  
  
def liashuk\_recal\_score(y\_true, y\_pred):  
 TP, FN, FP, TN = find\_conf\_matrix\_values(y\_true, y\_pred)  
 return TP / (TP + FN)  
  
  
assert liashuk\_recal\_score(df.actual\_label.values, df.predicted\_RF.values) == recall\_score(df.actual\_label.values, df.predicted\_RF.values),\  
 'my recal\_score fails on RF'  
  
assert liashuk\_recal\_score(df.actual\_label.values, df.predicted\_LR.values) == recall\_score(df.actual\_label.values, df.predicted\_LR.values),\  
 'my recal\_score fails on LR'  
  
print("My recall score on RF:", liashuk\_recal\_score(df.actual\_label.values, df.predicted\_RF.values))  
print("My recall score on LR:", liashuk\_recal\_score(df.actual\_label.values, df.predicted\_LR.values))  
  
# precision\_score  
  
print("Precision score on RF:", precision\_score(df.actual\_label.values, df.predicted\_RF.values))  
  
def liashuk\_precision\_score(y\_true, y\_pred):  
 TP, FN, FP, TN = find\_conf\_matrix\_values(y\_true, y\_pred)  
 return TP / (TP + FP)  
  
  
assert liashuk\_precision\_score(df.actual\_label.values, df.predicted\_RF.values) == precision\_score(df.actual\_label.values, df.predicted\_RF.values),\  
 'my precision\_score fails on RF'  
  
assert liashuk\_precision\_score(df.actual\_label.values, df.predicted\_LR.values) == precision\_score(df.actual\_label.values, df.predicted\_LR.values),\  
 'my precision\_score fails on LR'  
  
print("my precision score on RF:", liashuk\_precision\_score(df.actual\_label.values, df.predicted\_RF.values))  
print("my precision score on LR:", liashuk\_precision\_score(df.actual\_label.values, df.predicted\_LR.values))  
  
# F1 score  
print("F1 score on RF", f1\_score(df.actual\_label.values, df.predicted\_RF.values))  
def liashuk\_f1\_score(y\_true, y\_pred):  
 precision = liashuk\_precision\_score(y\_true, y\_pred)  
 recall = liashuk\_recal\_score(y\_true, y\_pred)  
 return (2 \* (precision \* recall)) / (precision + recall)  
  
  
assert liashuk\_f1\_score(df.actual\_label.values, df.predicted\_RF.values) == f1\_score(df.actual\_label.values, df.predicted\_RF.values),\  
 'my f1\_score fails on RF'  
  
assert liashuk\_f1\_score(df.actual\_label.values, df.predicted\_LR.values) == f1\_score(df.actual\_label.values, df.predicted\_LR.values),\  
 'my f1\_score fails on LR'  
  
print("My F1 score score on RF:", liashuk\_f1\_score(df.actual\_label.values, df.predicted\_RF.values))  
print("My F1 score score on LR:", liashuk\_f1\_score(df.actual\_label.values, df.predicted\_LR.values))  
print()  
  
  
def test\_thresholds(threshold: float):  
 print(f"Scores with threshold = {threshold}")  
 predicted = (df.model\_RF >= threshold).astype('int')  
  
 print("Accuracy RF:", liashuk\_accuracy\_score(df.actual\_label.values, predicted))  
 print("Precision RF:", liashuk\_precision\_score(df.actual\_label.values, predicted))  
 print("Recall RF:", liashuk\_recal\_score(df.actual\_label.values, predicted))  
 print("F1 RF:", liashuk\_f1\_score(df.actual\_label.values, predicted))  
 print()  
  
  
test\_thresholds(thresh)  
test\_thresholds(.25)  
test\_thresholds(.75)  
test\_thresholds(.15)  
  
# 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)  
  
# 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:", auc\_RF)  
print("AUC LR:", auc\_LR)  
  
plt.plot(fpr\_RF, tpr\_RF, 'r-', label=f'AUC RF: {auc\_RF}')  
plt.plot(fpr\_LR, tpr\_LR, 'b-', label=f'AUC LR: {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()

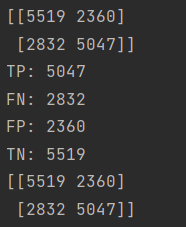


Рис. 7. Результат роботи функції по обчисленю матриці помилок(результат збігається)

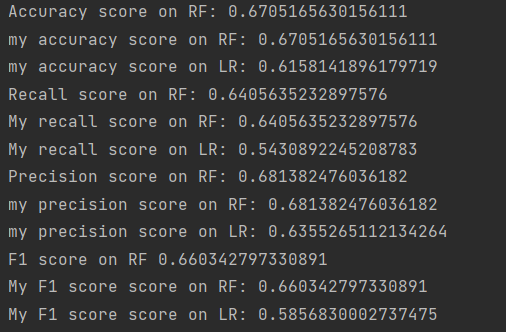


Рис. 8. Метрика моделей. Перевірка результатів власних функцій

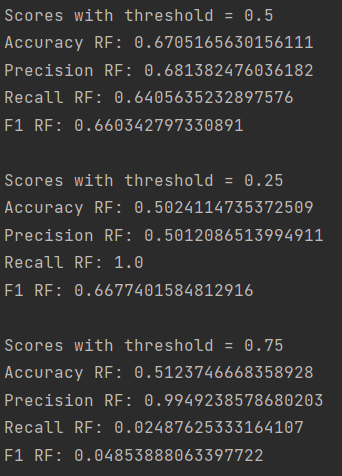


Рис. 9. Метрика моделі RF за різних порогів

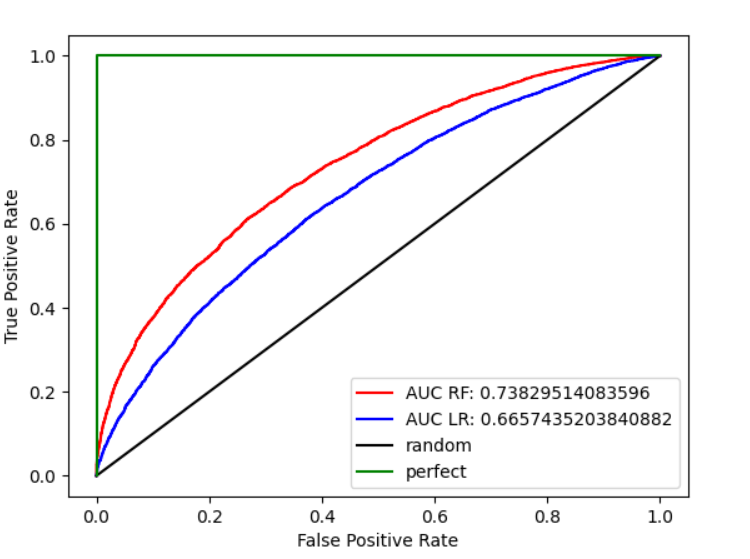


Рис. 10. Графік отриманих значень ROC

**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..

Лістинг коду файлу LR\_1\_task\_6.py:

import numpy as np  
from sklearn import svm  
from sklearn.model\_selection import train\_test\_split, 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]  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=3)  
classifier = svm.SVC(decision\_function\_shape='ovr')  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
visualize\_classifier(classifier, X\_test, y\_test)  
  
num\_folds = 3  
accuracy\_values = cross\_val\_score(classifier, X\_test, y\_test, scoring='accuracy', cv=num\_folds)  
print(f"Accuracy: {round(100 \* accuracy\_values.mean(), 2)}%")  
  
precision\_values = cross\_val\_score(classifier, X\_test, y\_test, scoring='precision\_weighted', cv=num\_folds)  
print(f"Precision: {round(100 \* precision\_values.mean(), 2)}%")  
  
recall\_values = cross\_val\_score(classifier, X\_test, y\_test, scoring='recall\_weighted', cv=num\_folds)  
print(f"Recall: {round(100 \* recall\_values.mean(), 2)}%")  
  
f1\_values = cross\_val\_score(classifier, X\_test, y\_test, scoring='f1\_weighted', cv=num\_folds)  
print(f"F1: {round(100 \* f1\_values.mean(), 2)}%")

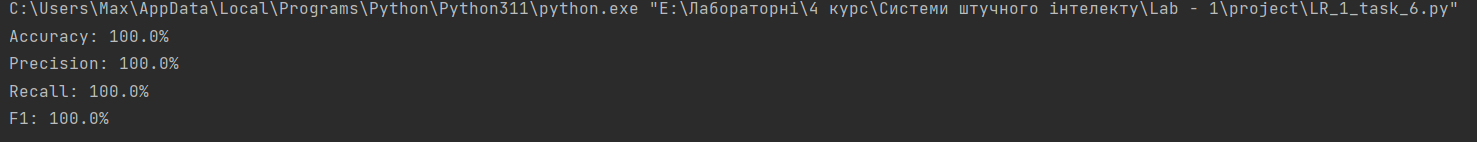


Рис. 11. Показники якості класифікатора

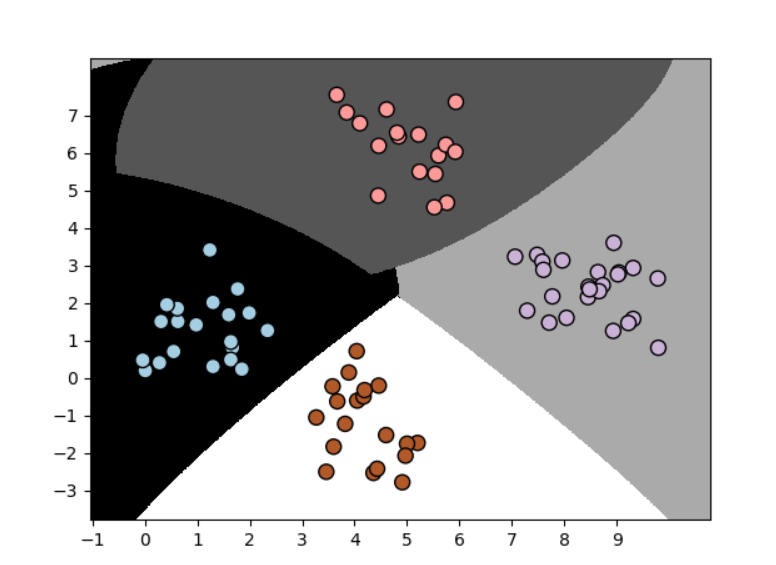


Рис. 12. Результат класифікації тестових даних за допомоги SVM

Висновки по використанню SVM класифікатора в порівняні з байєсівським класифікатором: SVM класифікатор є швидшим та простішим, але для використання у багатокласовій класифікації не пристосований. Окрім цього, кількість даних може бути недостатньою через однакові показники.

***Висновок:*** було досліджено попередню обробку та класифікацію даних, використовуючи спеціальні бібліотеки та мову програмування Python.