ЗВІТ

З лабораторної роботи №1

ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ

(навчальна дисципліна «СУЧАСНИЙ ШТУЧНИЙ ІНТЕЛЕКЕТ»)

Студента КН-20-1 навчальної групи

Кірія Даніли Олеговича варіант №6

Завдання 1.1. Попередня обробка даних

1.py

```python
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))
# Виключення середнього
data_scaled = preprocessing.scale(input_data)
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. 1.1 – Код програми

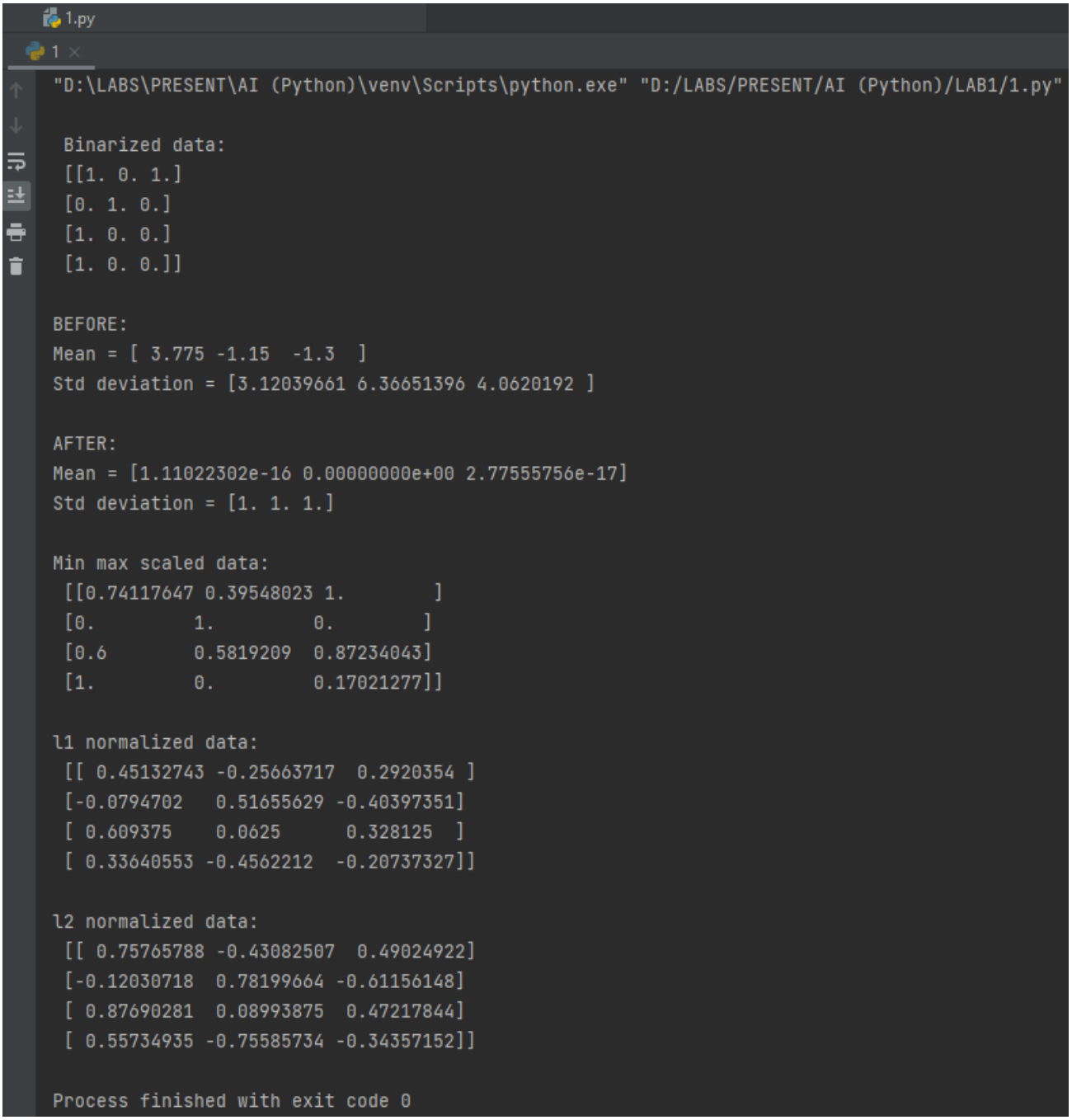


Рис. 1.2 – Результат виконання програмного коду

L1-нормалізація використовує метод найменших абсолютних відхилень (Least Absolute Deviations), **L2-нормалізація** використовує метод найменших квадратів, що забезпечує рівність 1 суми квадратів значень. *L1-нормалізації* вважається більш надійною по порівняно з *L2-нормалізацією*, оскільки вона менш чутлива до викидів.

```python
import numpy as np
from sklearn import preprocessing
# 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)
# 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. 1.3 – Код програми

```
"D:\LABS\PRESENT\AI (Python)\venv\Scripts\python.exe" "D:/LABS/PRESENT/AI (Python)/LAB1/2.py"

Label mapping:
black --> 0
black --> 1
green --> 2
red --> 3
white --> 4
yellow --> 5

Labels = ['green', 'red', 'black']
Encoded values = [2, 3, 1]

Encoded values = [3, 0, 4, 1]
Decoded labels = ['red', 'black', 'white', 'black']

Process finished with exit code 0
```

Рис. 1.4 – Результат виконання програмного коду

Завдання 2.2. Попередня обробка нових даних

```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[2.3, -1.6, 6.1],
                       [-2.4, -1.2, 4.3],
                       [3.2, 5.5, -6.1],
                       [-4.4, 1.4, -1.2]])
# 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))
# Виключення середнього
data_scaled = preprocessing.scale(input_data)
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. 2.1 – Код програми

```
"D:\LABS\PRESENT\AI (Python)\venv\Scripts\python.exe" "D:/LABS/PRESENT/AI (Python)/LAB1/LR_1_task_2.py"

 Binarized data:
 [[1. 0. 1.]
 [0. 0. 1.]
 [1. 1. 0.]
 [0. 0. 0.]]

BEFORE:
Mean = [-0.325  1.025  0.775]
Std deviation = [3.17125764 2.82875856 4.79446295]

AFTER:
Mean = [ 5.55111512e-17 -5.55111512e-17  6.93889390e-17]
Std deviation = [1. 1. 1.]

Min max scaled data:
 [[0.88157895 0.         1.        ]
 [0.26315789 0.05633803 0.85245902]
 [1.         1.         0.        ]
 [0.         0.42253521 0.40163934]]

l1 normalized data:
 [[ 0.23       -0.16        0.61      ]
 [-0.30379747 -0.15189873  0.5443038 ]
 [ 0.21621622  0.37162162 -0.41216216]
 [-0.62857143  0.2        -0.17142857]]

l2 normalized data:
 [[ 0.34263541 -0.23835507  0.90872869]
 [-0.47351004 -0.23675502  0.84837215]
 [ 0.36302745  0.62395344 -0.69202108]
 [-0.92228798  0.29345527 -0.25153308]]

Process finished with exit code 0
```

Рис. 2.2 – Результат виконання програмного коду

Завдання 2.3. Класифікація логістичною регресією або логістичний класифікатор

```python
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. 3.1 – Код програми

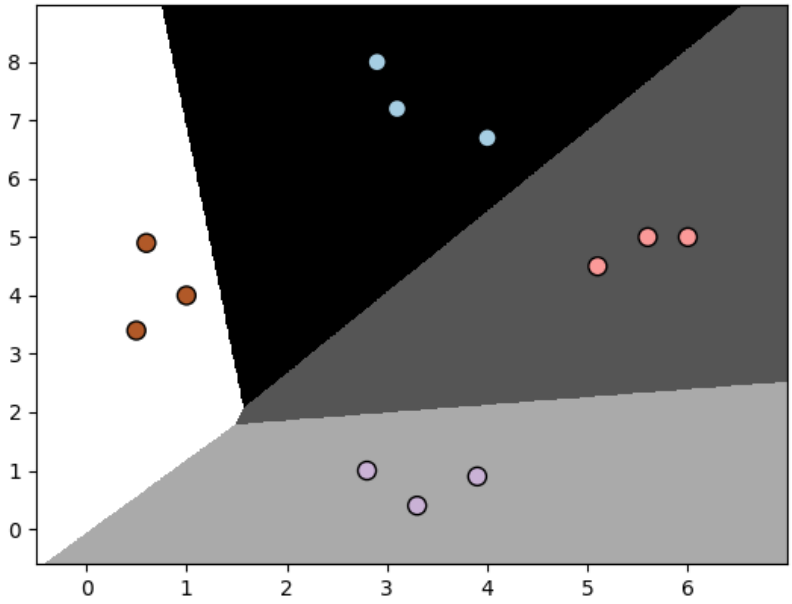


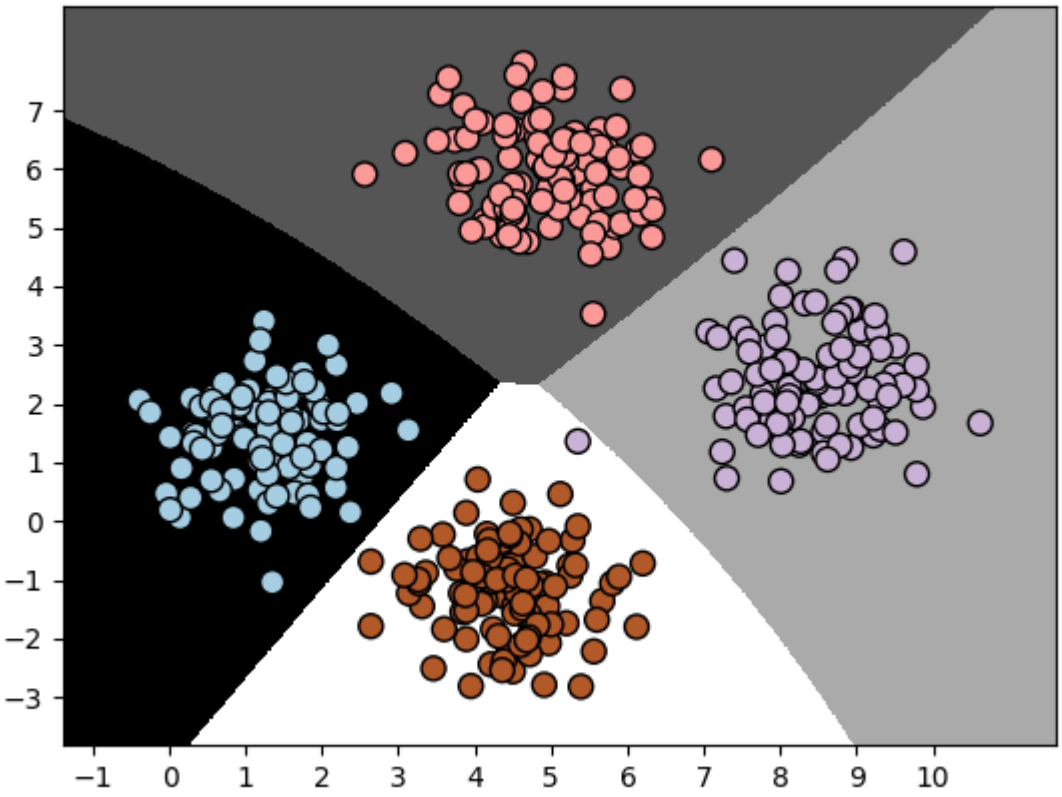
Рис. 3.2 – Результат виконання програмного коду

Завдання 2.4. Класифікація наївним байєсовським класифікаторо

```python
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. 4.1 – Код програми



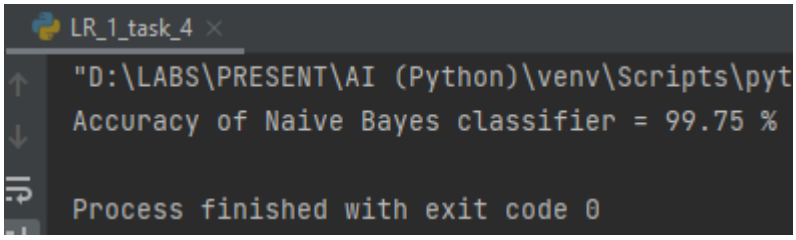


Рис. 4.2 – Результат виконання програмного коду

```
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. 4.3 – Код програми

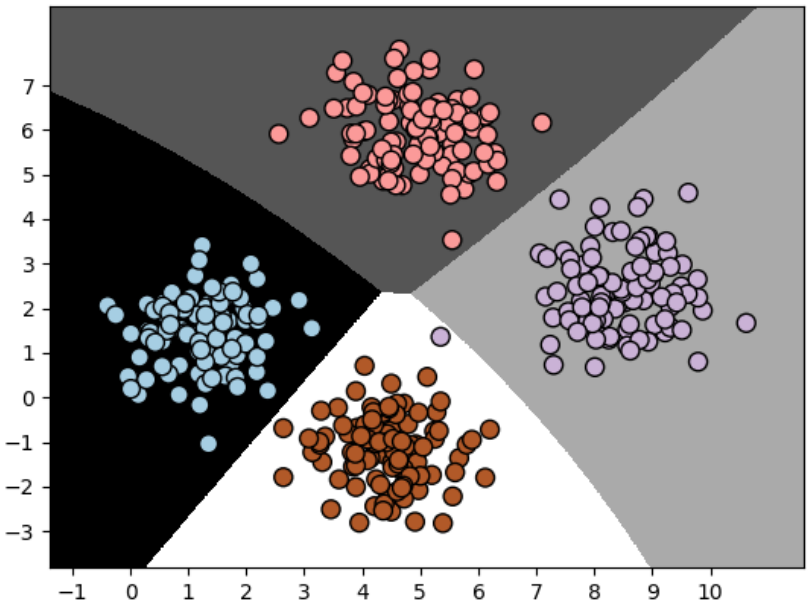
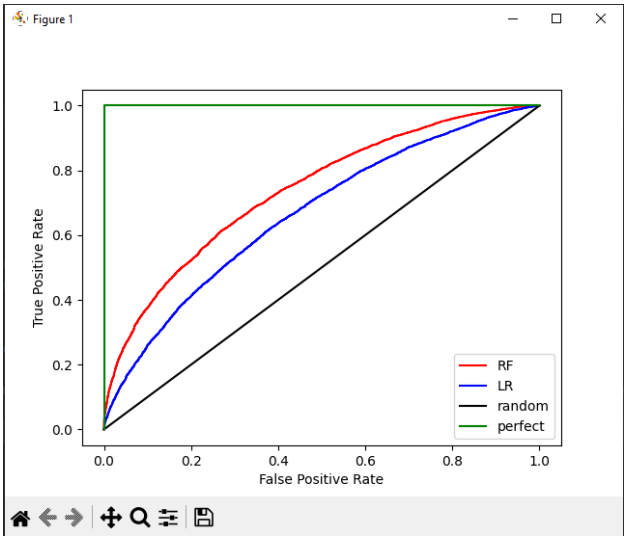
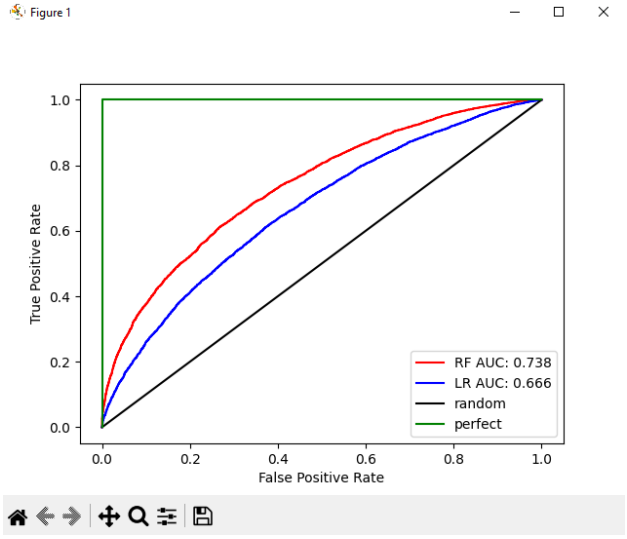


Рис. 4.4 – Результат виконання програмного коду

Ніякої різниці не побачив

Завдання 2.5. Вивчити метрики якості класифікації



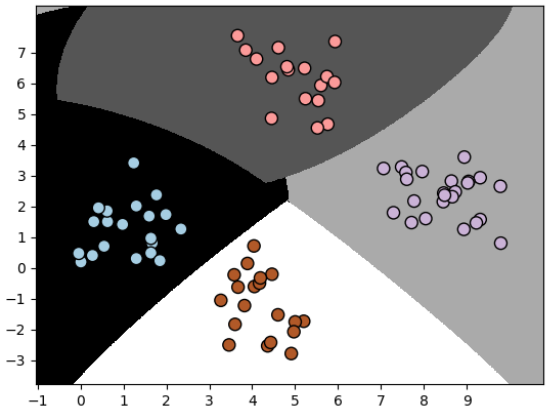



```
TP: 5047
FN: 2832
FP: 2360
TN: 5519
Accuracy RF:  0.671
Accuracy LR:  0.616
Recall RF: 0.641
Recall LR: 0.543
Precision RF:  0.681
Precision LR:  0.636
F1 RF: 0.660
F1 LR: 0.586
scores with threshold = 0.5
Accuracy RF: 0.671
Recall RF: 0.641
Precision RF: 0.681
F1 RF: 0.660
```

Рис. 5.1 – Результат виконання програмного коду

RF та LR моделі показали однаковий результат.

Завдання 2.6. Розробіть програму класифікації даних в файлі data_multivar_nb.txt за допомогою машини опорних векторів (Support Vector Machine - SVM).




```
Accuracy of SVC classifier = 100.0 %
Accuracy: 99.75%
Precision: 99.76%
Recall: 99.75%
F1: 99.75%
```

Рис. 6.1 – Результат виконання програмного коду

Висновки: в ході виконання лабораторної роботи, використовуючи спеціалізовані бібліотеки та мову програмування Python, було досліджено попередню обробку та класифікацію даних