

ЛАБОРАТОРНА РОБОТА №1

ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ

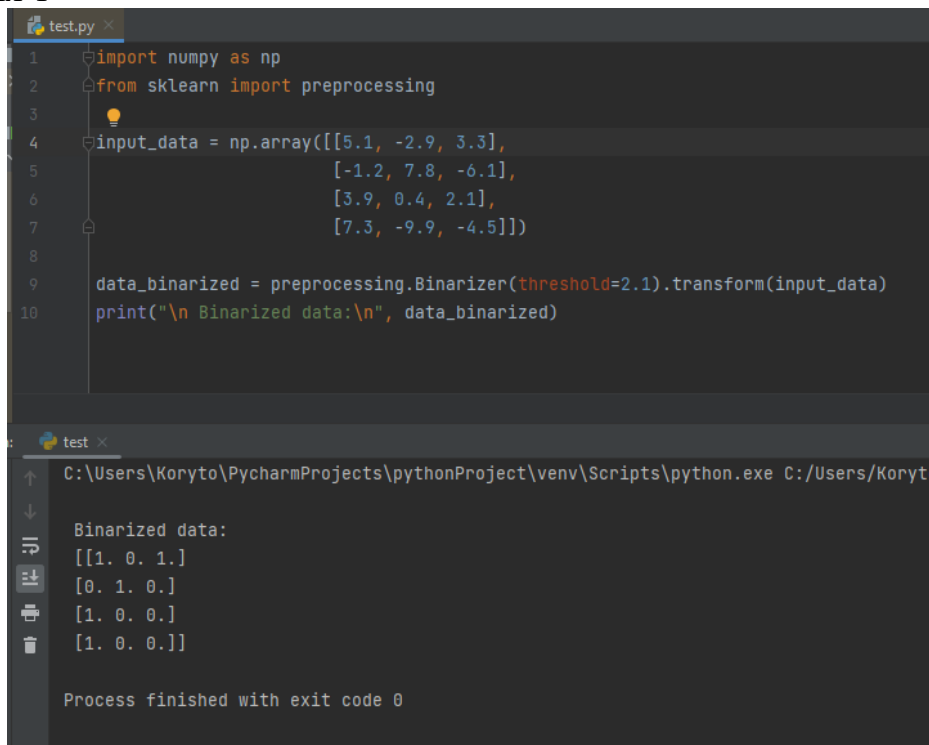
Мета заняття: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити попередню обробку та класифікацію даних.

Хід роботи

Варіант №3

GitHub репозиторій: https://github.com/dimonrok/SAI_Horbenko_PI-59/tree/main/lab1

Завдання 1



```
1 import numpy as np
2 from sklearn import preprocessing
3
4 input_data = np.array([[5.1, -2.9, 3.3],
5                        [-1.2, 7.8, -6.1],
6                        [3.9, 0.4, 2.1],
7                        [7.3, -9.9, -4.5]])
8
9 data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
10 print("\n Binarized data:\n", data_binarized)
```

test

C:\Users\Koryto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Koryto

Binarized data:

```
[[1. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
```

Process finished with exit code 0

Рис. 1.1.1 Бінарізація

Змін.	Арк.	№ документа	Підпис	Дата				
Розроб.		Горбенко Д.С.			Звіт з лабораторної роботи №1	Лім.	Арк.	Аркушів
Перевір.		Пулеко І. В.					1	7
Керівник						ФІКТ, гр.ПІ-59		
Н.контр.								
Зав. каф.								

```

test.py
1 import numpy as np
2 from sklearn import preprocessing
3
4 input_data = np.array([[5.1, -2.9, 3.3],
5                        [-1.2, 7.8, -6.1],
6                        [3.9, 0.4, 2.1],
7                        [7.3, -9.9, -4.5]])
8
9 data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
10 print("\n Binarized data:\n", data_binarized)
11
12 print("\nBEFORE: ")
13 print("Mean =", input_data.mean(axis=0))
14 print("Std deviation =", input_data.std(axis=0))
15
16 data_scaled = preprocessing.scale(input_data)
17 print("\nAFTER: ")
18 print("Mean = ", data_scaled.mean(axis=0))
19 print("Std deviation =", data_scaled.std(axis=0))

```

```

test
↑ BEFORE:
↓ Mean = [ 3.775 -1.15 -1.3 ]
Std deviation = [3.12039661 6.36651396 4.0620192 ]

AFTER:
Mean = [1.11022302e-16 0.00000000e+00 2.77555756e-17]
Std deviation = [1. 1. 1.]

Process finished with exit code 0

```

Рис. 1.1.2 Виключення середнього

```

20
21 #Масштабування MinMax
22 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
23 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
24 print("\nMin max scaled data: \n", data_scaled_minmax)

```

```

Run: test
↑ Std deviation = [1. 1. 1.]
↓
Min max scaled data:
[[0.74117647 0.39548023 1.         ]
 [0.         1.         0.         ]
 [0.6        0.5819209  0.87234043]
 [1.         0.         0.17021277]]

Process finished with exit code 0

```

Рис. 1.1.3 Масштабування

```
25
26 # нормалізація даних
27 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
28 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
29 print ("\nl1 normalized data:\n", data_normalized_l1)
30 print ("\nl2 normalized data:\n", data_normalized_l2)]
```



```
l1 normalized data:
[[ 0.45132743 -0.25663717  0.2920354 ]
 [-0.0794702  0.51655629 -0.40397351]
 [ 0.609375   0.0625     0.328125  ]
 [ 0.33640553 -0.4562212  -0.20737327]]

l2 normalized data:
[[ 0.75765788 -0.43082507  0.49024922]
 [-0.12030718  0.78199664 -0.61156148]
 [ 0.87690281  0.08993875  0.47217844]
 [ 0.55734935 -0.75585734 -0.34357152]]

Process finished with exit code 0
```

Рис. 1.1.4 Нормалізація

L1-нормалізація використовує метод найменших абсолютних відхилень (Least Absolute Deviations), що забезпечує рівність 1 суми абсолютних значень в кожному ряду. L2-нормалізація використовує метод найменших квадратів, що забезпечує рівність 1 суми квадратів 4 значень. Тому можна зробити висновки, що L1 нормалізація є більш надійною у порівнянні з L2.

```

1 import numpy as np
2 from sklearn import preprocessing
3
4 Input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow', 'white']
5
6 encoder = preprocessing.LabelEncoder()
7 encoder.fit(Input_labels)
8
9 print("\nLabel mapping:")
10 for i, item in enumerate(encoder.classes_):
11     print(item, '-->', i)
12
13 #перетворення міток за допомогою кодувальника
14
15 test_labels = ['green', 'red', 'black']
16 encoded_values = encoder.transform(test_labels)
17 print("\nLabels =", test_labels)
18 print("Encoded values =", list(encoded_values))
19
20 # Декодування набору чисел за допомогою декодера
21
22 encoded_values = [3, 0, 4, 1]
23 decoded_list = encoder.inverse_transform(encoded_values)
24 print("\nEncoded values =", encoded_values)
25 print("Decoded labels =", list(decoded_list))

```

Run: nextExercise x

```

C:\Users\Koryto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Koryto
Label mapping:
black --> 0
green --> 1
red --> 2
white --> 3
yellow --> 4

Labels = ['green', 'red', 'black']
Encoded values = [1, 2, 0]

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

```

Рис. 1.2 Кодування міток

Масив Input_labels був пересортований за алфавітним порядком, та був проіндексований від 0 до 4. Наступна частина коду демонструє роботу кодувальника, (слова замінюються числами). Третя частина коду демонструє зворотню процедуру.

Завдання 2: Попередня обробка нових даних

3.	1.3	-3.9	6.5	-4.9	-2.2	1.3	2.2	6.5	-6.1	-5.4	-1.4	2.2	1.1
----	-----	------	-----	------	------	-----	-----	-----	------	------	------	-----	-----

```

1 import numpy as np
2 from sklearn import preprocessing
3 # Horbenko variant3
4 input_data = np.array([[1.3, -3.9, 6.5],
5                        [-4.9, -2.2, 1.3],
6                        [2.2, 6.5, -6.1],
7                        [-5.4, -1.4, 2.2]])
8
9 data_binarized = preprocessing.Binarizer(threshold=1.1).transform(input_data)
10 print("\n Binarized data:\n", data_binarized)
11
12 print("\nBEFORE: ")
13 print("Mean =", input_data.mean(axis=0))
14 print("Std deviation =", input_data.std(axis=0))
15
16 data_scaled = preprocessing.scale(input_data)
17 print("\nAFTER: ")
18 print("Mean = ", data_scaled.mean(axis=0))
19 print("Std deviation =", data_scaled.std(axis=0))
20
21 #Масштабування MinMax
22 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
23 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
24 print("\nMin max scaled data: \n", data_scaled_minmax)
25
26 # нормалізація даних
27 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
28 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
29 print("\nl1 normalized data:\n", data_normalized_l1)
30 print("\nl2 normalized data:\n", data_normalized_l2)

```

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

BEFORE:
 Mean = [-1.7 -0.25 0.975]
 Std deviation = [3.46914975 4.00031249 4.53286609]

AFTER:
 Mean = [0.00000000e+00 -2.77555756e-17 8.32667268e-17]
 Std deviation = [1. 1. 1.]

Min max scaled data:
 [[0.88157895 0. 1.]
 [0.06578947 0.16346154 0.58730159]
 [1. 1. 0.]
 [0. 0.24038462 0.65873016]]

l1 normalized data:
 [[0.11111111 -0.33333333 0.55555556]
 [-0.58333333 -0.26190476 0.1547619]
 [0.14864865 0.43918919 -0.41216216]
 [-0.6 -0.15555556 0.24444444]]

l2 normalized data:
 [[0.16903085 -0.50709255 0.84515425]
 [-0.88666908 -0.39809632 0.23523874]
 [0.23961218 0.70794508 -0.66437923]
 [-0.90050042 -0.23346307 0.36687054]]

Process finished with exit code 0

Рис. 1.3 Результат виконання

						Арк
						6
Змін.	Арк.	№ документа	Підпис	Дата		

Завдання 3: Класифікація логістичною регресією або логістичний класифікатор

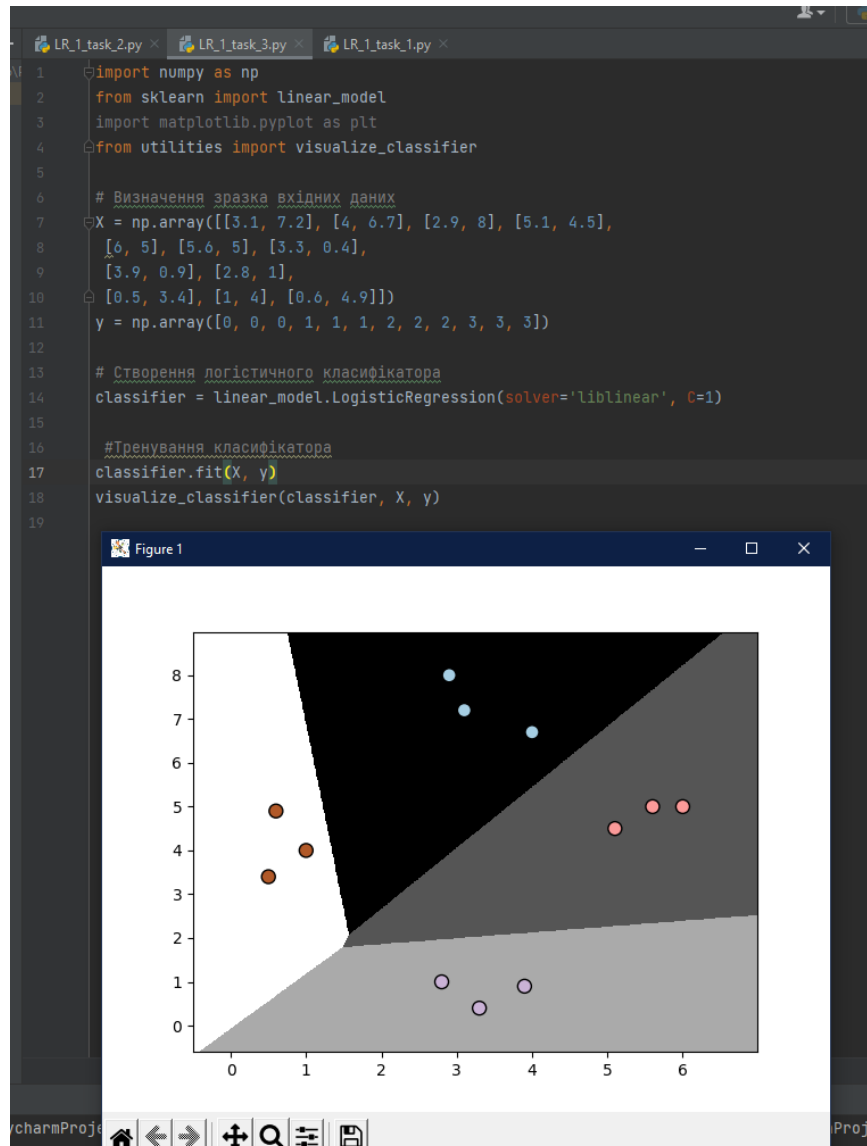


Рис. 1.4 Візуалізація класифікації логістичною регресією

Завдання 4: Класифікація наївним байєсовським класифікатором

Обидва прогони дали ідентичний результат, оскільки генерувались однакові набори даних для навчання й тестування.

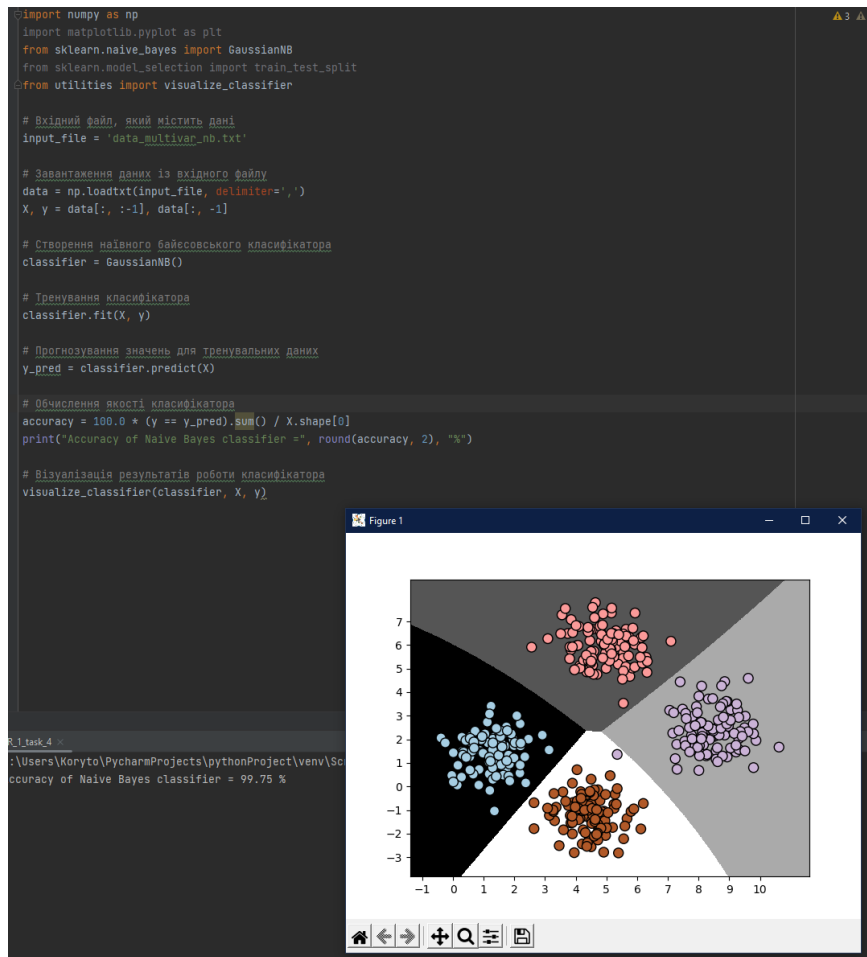


Рис. 1.5 Класифікація найпростішим байєсовським класифікатором

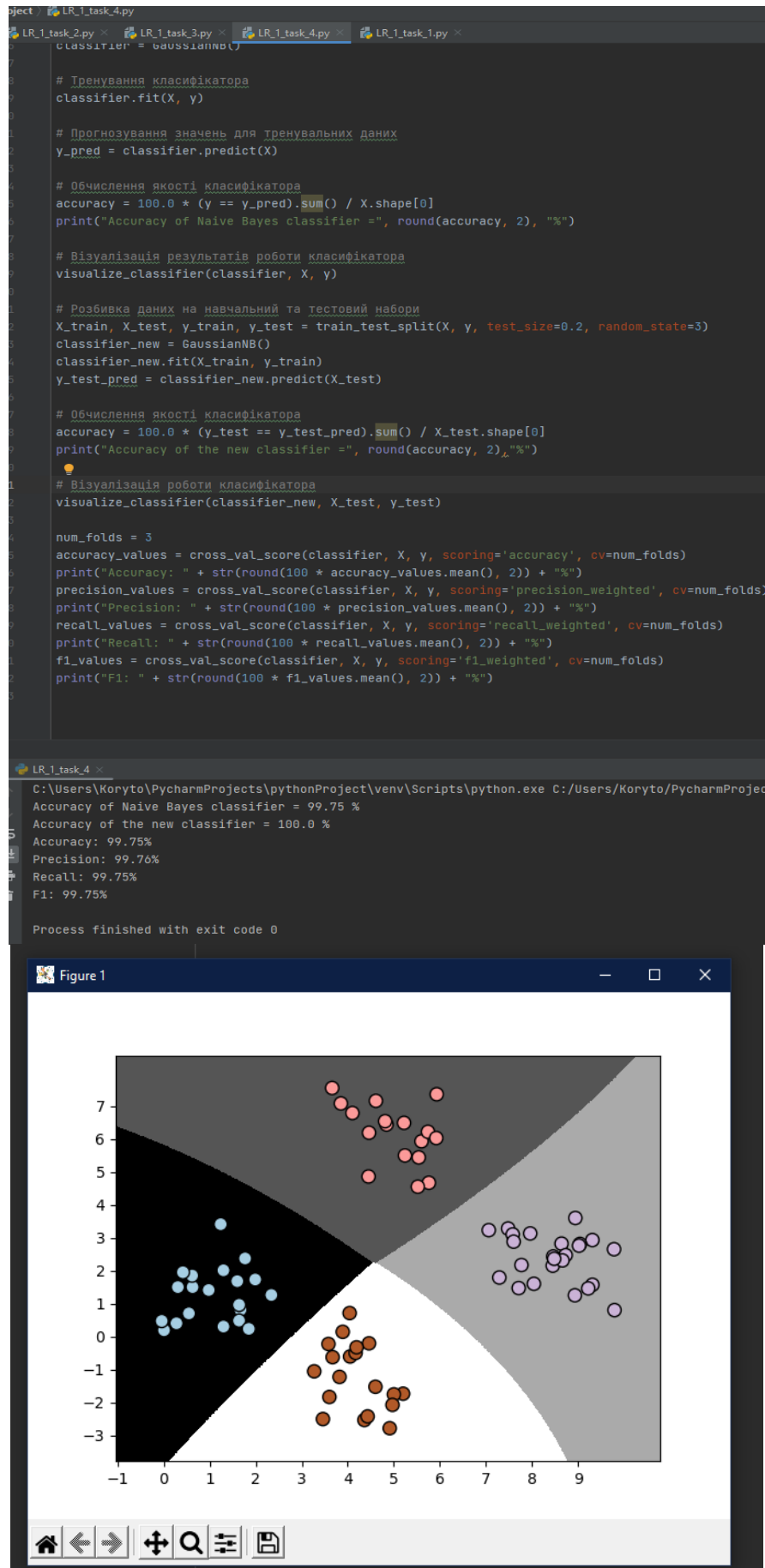


Рис. 1.6. Класифікація найвісім байесовським класифікатором з обчисленням якості, точності та повноти

Змін.	Арк.	№ документа	Підпис	Дата

Завдання 5

```

30 print('TP:', hor_find_TP(df.actual_label.values, df.predicted_RF.values))
31 print('FN:', hor_find_FN(df.actual_label.values, df.predicted_RF.values))
32 print('FP:', hor_find_FP(df.actual_label.values, df.predicted_RF.values))
33 print('TN:', hor_find_TN(df.actual_label.values, df.predicted_RF.values))
34
35 def find_conf_matrix_values(y_true, y_pred):
36     TP = hor_find_TP(y_true, y_pred)
37     FN = hor_find_FN(y_true, y_pred)
38     FP = hor_find_FP(y_true, y_pred)
39     TN = hor_find_TN(y_true, y_pred)
40     return TP, FN, FP, TN
41
42 def hor_confusion_matrix(y_true, y_pred):
43     TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
44     return np.array([[TN, FP], [FN, TP]])
45
46 hor_confusion_matrix(df.actual_label.values, df.predicted_RF.values)
47
48 assert np.array_equal(hor_confusion_matrix(df.actual_label.values, df.predicted_RF.
49

```

```

C:\Users\Koryto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Koryto/P
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
Accuracy LR: 0.616
Recall LR: 0.543
Precision LR: 0.636
F1 LR: 0.586

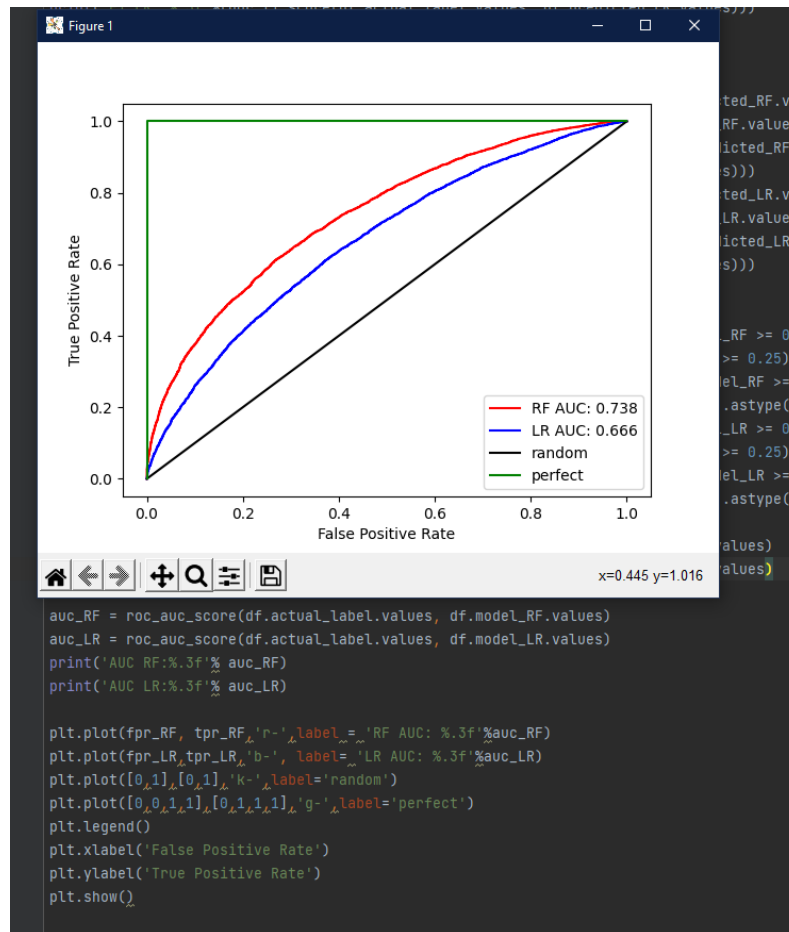
scores with threshold = 0.25
Accuracy RF: 0.502
Recall RF: 1.000
Precision RF: 0.501
F1 RF: 0.668
Accuracy LR: 0.503
Recall LR: 0.999
Precision LR: 0.501
F1 LR: 0.668
AUC RF: 0.738
AUC LR: 0.666

```

Рис. 1.7. Порівняння моделей RF та LR на кроках 0.25 та 0.5

При порозі 0.5 якість та точність значно вищі, у разі використання моделі RF, тому, як на мене вона є більш оптимальною, але при порозі 0.25 LR модель справляється краще, тому остаточний вибір варто робити виходячи з вхідних даних.

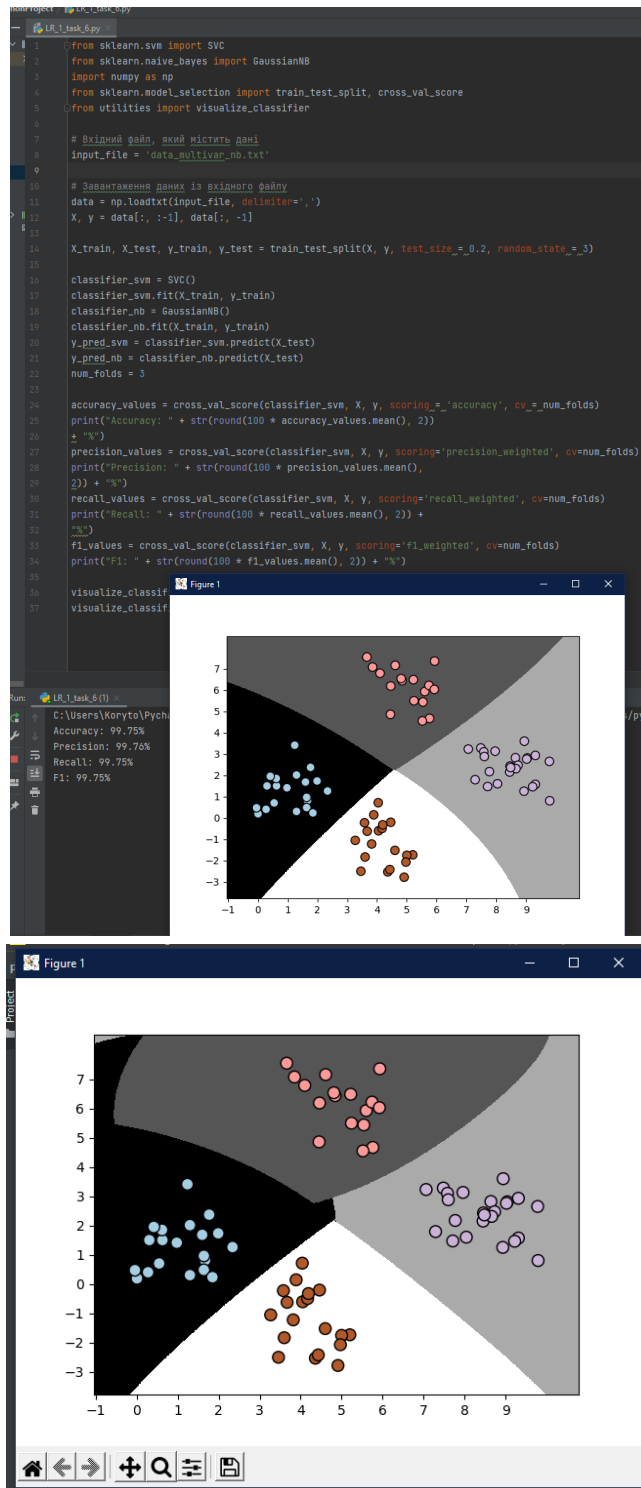
					Арк
					10
Змін.	Арк.	№ документа	Підпис	Дата	



1.8. Порівняння моделей за допомогою кривих ROC

Завдання 6: Розробіть програму класифікації даних

Змін.	Арк.	№ документа	Підпис	Дата



1.9 порівняння класифікаторів наївного байєса та SVM

Як на мене краще обрати класифікатор SVM, оскільки він є більш гнучким, в плані того що припускає можливість поєднання декількох характеристик, навідміну від наївного класифікатора байєса.

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

Змін.	Арк.	№ документа	Підпис	Дата

						Арк
						2
Змін.	Арк.	№ документа	Підпис	Дата		