

ЛАБОРАТОРНА РОБОТА №1

ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ

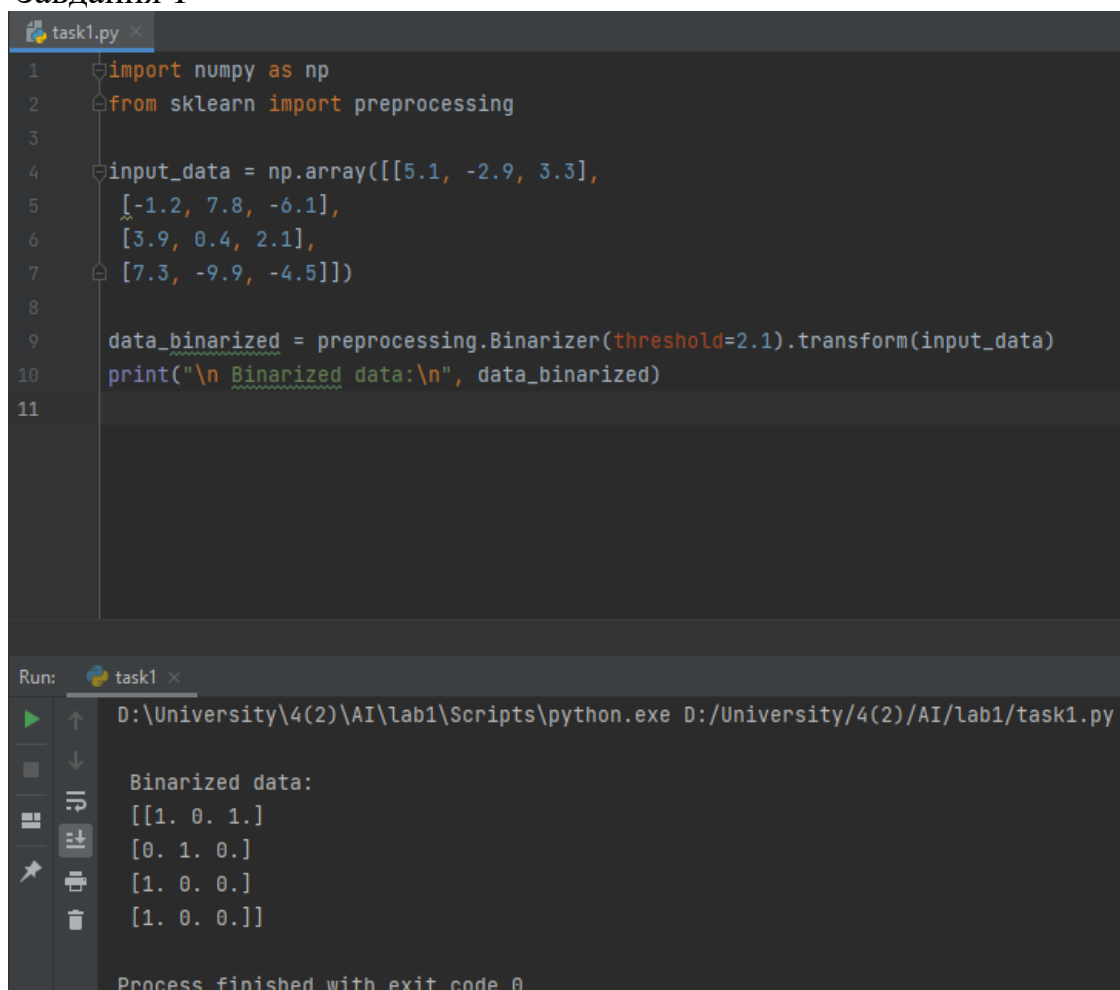
Мета заняття: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити попередню обробку та класифікацію даних.

Хід роботи

GitHub репозиторій: https://github.com/AlexanderHorielko/SAI_Horielko_PI-59

Варіант №5

Завдання 1



```
task1.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
```



```
Run: task1
D:\University\4(2)\AI\lab1\Scripts\python.exe D:/University/4(2)/AI/lab1/task1.py

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

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.05.000 – Лр1						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Горелко О. В.			Звіт з лабораторної роботи	Лім.		Арк.	Аркушів		
Перевір.		Пулеко І. В.						1	3		
Керівник						ФІКТ Гр. ПІ-59					
Н. контр.											
Зав. каф.											

```

task1.py
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))

Run: task1
D:\University\4(2)\AI\lab1\Scripts\python.exe D:/University/4(2)/AI/lab1/task1.py

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

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

AFTER:
Mean = [1.11022302e-16 0.00000000e+00 2.7755756e-17]
Std deviation = [1. 1. 1.]

Process finished with exit code 0

```

Рис. 1.1.2 Виключення середнього

```

task1.py
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))
20
21 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
22 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
23 print("\nMin max scaled data:\n", data_scaled_minmax)
24

Run: task1
D:\University\4(2)\AI\lab1\Scripts\python.exe D:/University/4(2)/AI/lab1/task1.py

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

		Горелко О. В.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.05.000 – Лр1	Арк.
		Пудеко І. В.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

25 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
26 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
27 print("\nL1 normalized data:\n", data_normalized_l1)
28 print("\nL2 normalized data:\n", data_normalized_l2)
29

```

```

Run: task1
D:\University\4(2)\AI\lab1\Scripts\python.exe D:\University\4(2)\AI\lab1\task1.py

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.

```

lab1 task2.py
1 import numpy as np
2 from sklearn import preprocessing
3
4 input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow', 'white']
5 encoder = preprocessing.LabelEncoder()
6 encoder.fit(input_labels)
7 print("\nLabel mapping:")
8 for i, item in enumerate(encoder.classes_):
9     print(item, '--->', i)
10
11 test_labels = ['green', 'red', 'black']
12 encoded_values = encoder.transform(test_labels)
13 print("\nLabels =", test_labels)
14 print("Encoded values =", list(encoded_values))
15
16 encoded_values = [3, 0, 4, 1]
17 decoded_list = encoder.inverse_transform(encoded_values)
18 print("\nDecoded values =", encoded_values)
19 print("Decoded labels =", list(decoded_list))

```

```

Run: task1
D:\University\4(2)\AI\lab1\Scripts\python.exe D:\University\4(2)\AI\lab1\task1.py

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

BEFORE:
Mean = [ 3.775 -1.15 -1.3 ]
Std deviation = [3.12039661 0.36651396 4.0620192 ]

AFTER:
Mean = [1.11822302e-16 0.00000000e+00 2.77555750e-17]
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]]

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]]

```

Рис. 1.1.5 Кодування міток

		Горелко О. В.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.05.000 – Лр1	Арк.
		Пулеко І. В.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Завдання 2: Попередня обробка нових даних

5.	-1.3	3.9	4.5	-5.3	-4.2	-1.3	5.2	-6.5	-1.1	-5.2	2.6	-2.2	3.0
----	------	-----	-----	------	------	------	-----	------	------	------	-----	------	-----

lab1 task1.py task2.py

task1.py task2.py

4 input_data = np.array([[-1.3, 3.9, 4.5],

5 [-5.3, -4.2, -1.3],

6 [5.2, -6.5, -1.1],

7 [-5.2, 2.6, -2.2]])

8

9 data_binarized = preprocessing.Binarizer(threshold=3.0).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 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))

22 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)

23 print("\nMin max scaled data:\n", data_scaled_minmax)

24

25 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')

26 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')

27 print("\nl1 normalized data:\n", data_normalized_l1)

28

Run: task1

↑ Binarized data:

↓ [[0. 1. 1.]

0. 0. 0.]

1. 0. 0.]

0. 0. 0.]]

BEFORE:

Mean = [-1.65 -1.05 -0.025]

Std deviation = [4.27112397 4.40028408 2.64516068]

AFTER:

Mean = [-2.77555756e-17 5.55111512e-17 5.55111512e-17]

Std deviation = [1. 1. 1.]

Min max scaled data:

[[0.38095238 1. 1.]

[0. 0.22115385 0.13432836]

[1. 0. 0.1641791]

[0.00952381 0.875 0.]]

l1 normalized data:

[[-0.13402062 0.40206186 0.46391753]

[-0.49074074 -0.38888889 -0.12037037]

[0.40625 -0.5078125 -0.0859375]

[-0.52 0.26 -0.22]]

l2 normalized data:

[[-0.21328678 0.63986035 0.7383004]

[-0.76965323 -0.60991388 -0.18878287]

[0.61931099 -0.77413873 -0.13100809]

[-0.83653629 0.41826814 -0.3539192]]

Рис. 1.3 Результат

Завдання 3: Класифікація логістичною регресією або логістичний класифікатор

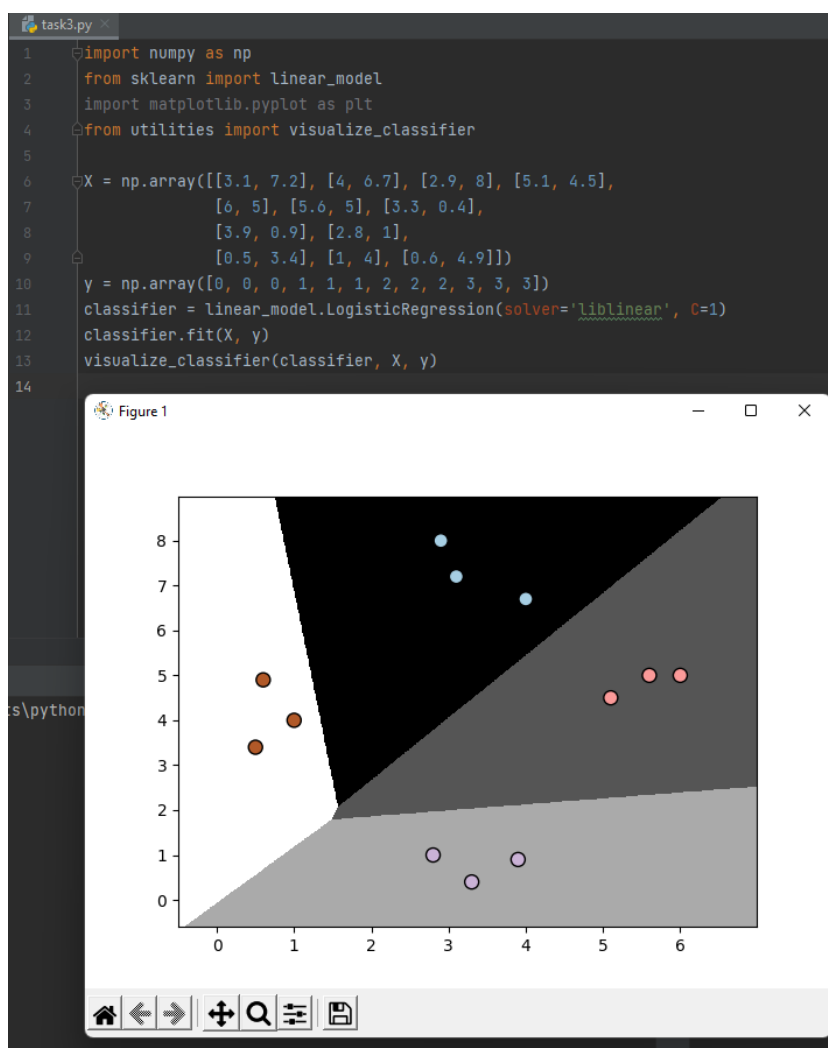


Рисунок 1.4 Візуалізація класифікації логістичною регресією

Завдання 4: Класифікація наївним байєсовським класифікатором

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

		Горелко О. В.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.05.000 – Лр1	Арк.
		Пудеко І. В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

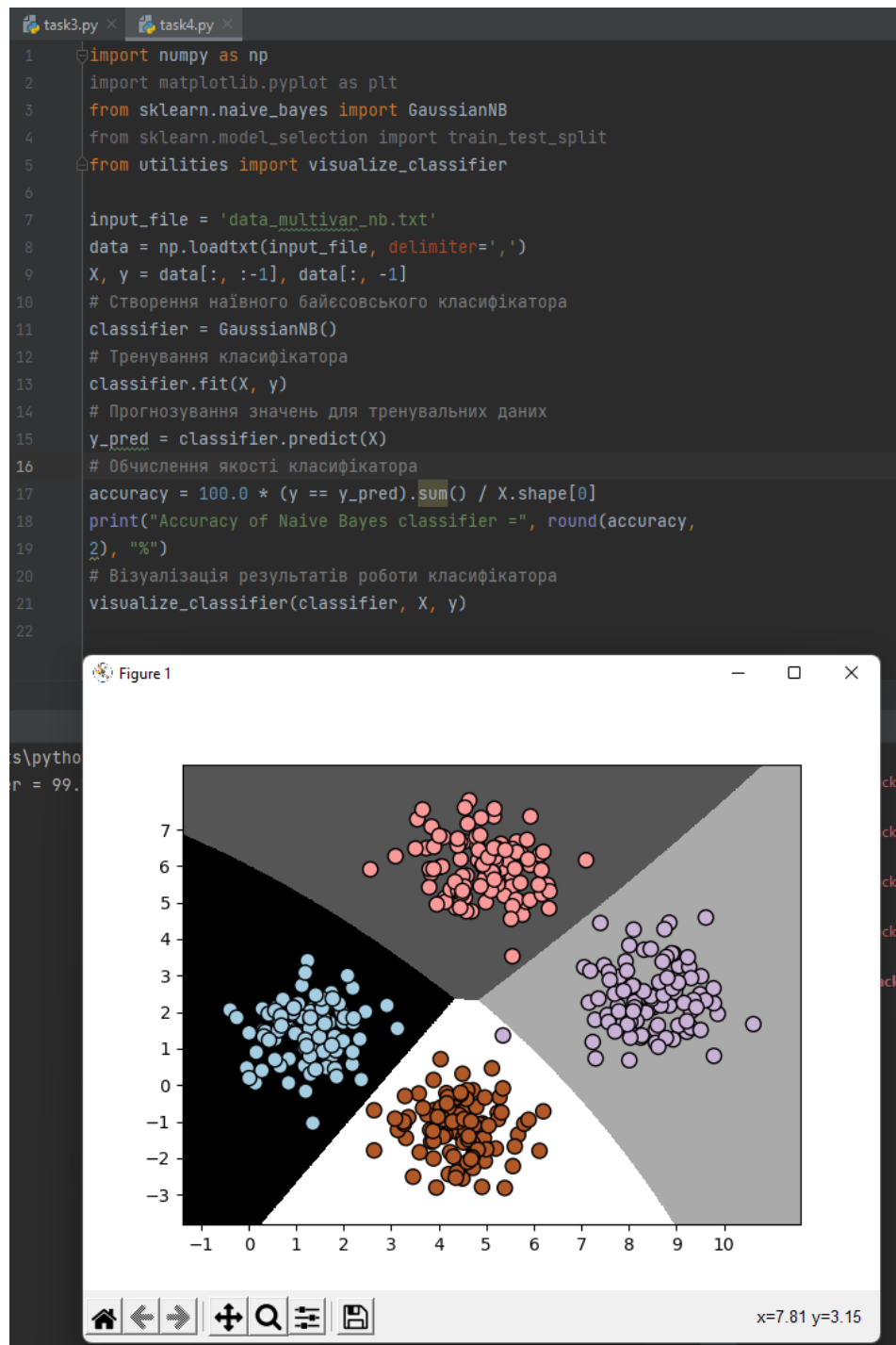


Рисунок 1.5 Класифікація наївним байєсовським класифікатором

		Горелко О. В.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.05.000 – Лр1	Арк.
		Пудеко І. В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

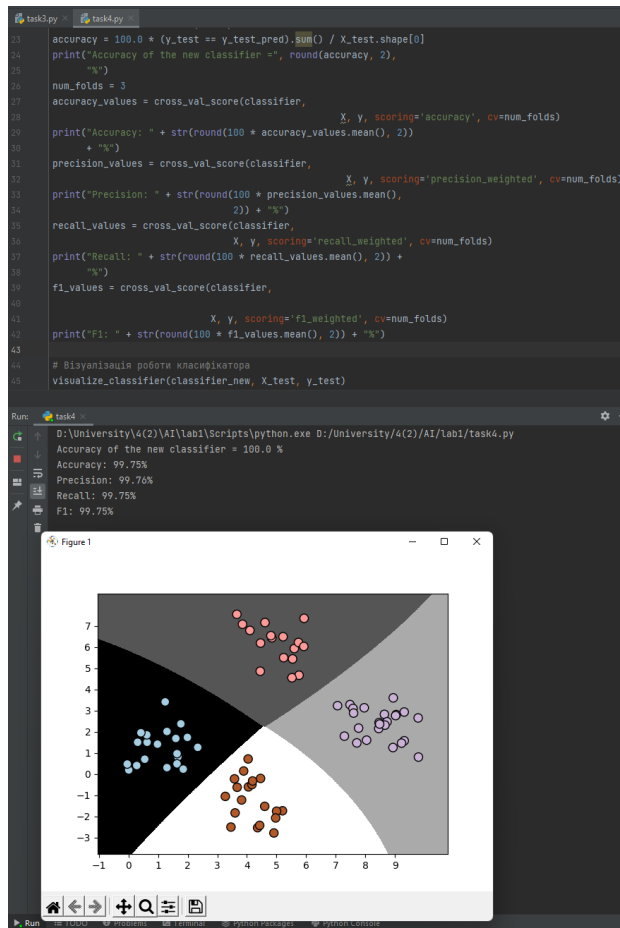


Рисунок 1.6. Класифікація найвісім байєсовським класифікатором з обчисленням якості, точності та повноти

Завдання 5: Вивчити метрики якості класифікації

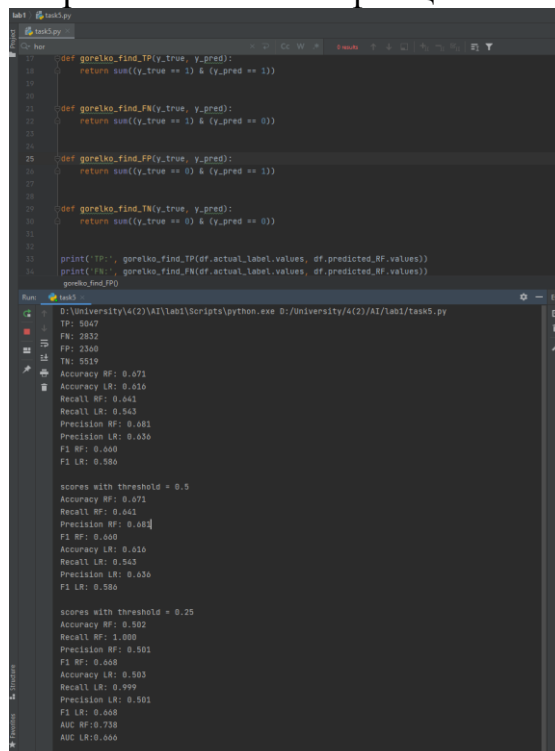


Рисунок 1.7. Порівняння моделей RF та LF на кроках 0.25 та 0.5

		Горелко О. В.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.05.000 – Лр1	Арк.
		Пудеко І. В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

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.

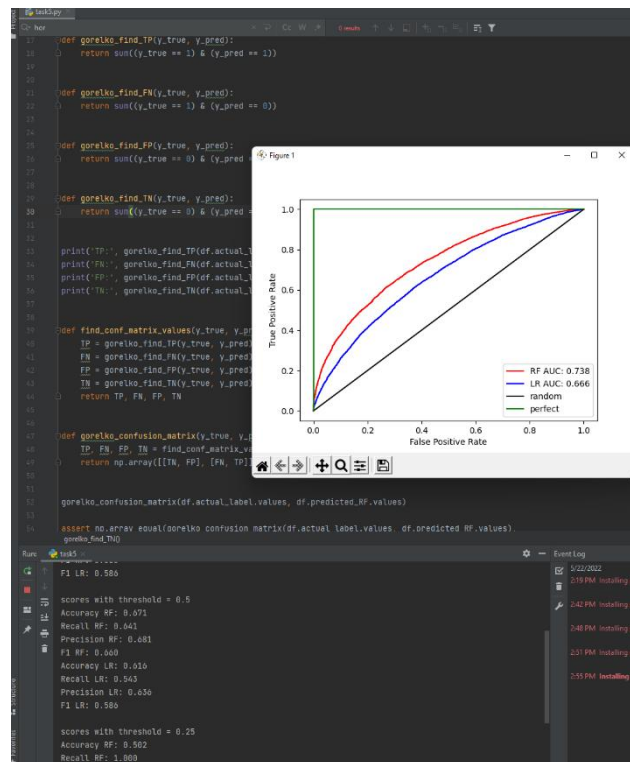


Рисунок 1.8. Порівняння моделей за допомогою кривих ROC

Завдання 6: Розробіть програму класифікації даних

```

lab1 task6.py
1 from sklearn.svm import SVC
2 from sklearn.naive_bayes import GaussianNB
3 import numpy as np
4 from sklearn.model_selection import train_test_split, cross_val_score
5 from utilities import visualize_classifier
6
7 input_file = 'data_multivar_nb.txt'
8
9 data = np.loadtxt(input_file, delimiter=',')
10 X, y = data[:, :-1], data[:, -1]
11
12 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
13
14 classifier_svm = SVC()
15 classifier_svm.fit(X_train, y_train)
16 classifier_nb = GaussianNB()
17 classifier_nb.fit(X_train, y_train)
18 y_pred_svm = classifier_svm.predict(X_test)
19 y_pred_nb = classifier_nb.predict(X_test)
20 num_folds = 3
21
22 accuracy_values = cross_val_score(classifier_svm, X, y, scoring='accuracy', cv=num_folds)
23 print('Accuracy: ' + str(round(100 * accuracy_values.mean(), 2)) + '%')
24 precision_values = cross_val_score(classifier_svm, X, y, scoring='precision_weighted', cv=num_folds)
25 print('Precision: ' + str(round(100 * precision_values.mean(), 2)) + '%')
26 recall_values = cross_val_score(classifier_svm, X, y, scoring='recall_weighted', cv=num_folds)
27 print('Recall: ' + str(round(100 * recall_values.mean(), 2)) + '%')
28 f1_values = cross_val_score(classifier_svm, X, y, scoring='f1_weighted', cv=num_folds)
29 print('F1: ' + str(round(100 * f1_values.mean(), 2)) + '%')
30
31 visualize_classifier(classifier_nb, X_test, y_test)
32 visualize_classifier(classifier_svm, X_test, y_test)
33

```

Run task6.py

D:\University\4(2)\AI\lab1\Scripts\python.exe D:\University\4(2)\AI\lab1\task6.py

Accuracy: 99.75%

Precision: 99.75%

Recall: 99.75%

F1: 99.75%

Process finished with exit code 0

		Горелко О. В.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.05.000 – Лр1	Арк.
		Пудеко І. В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

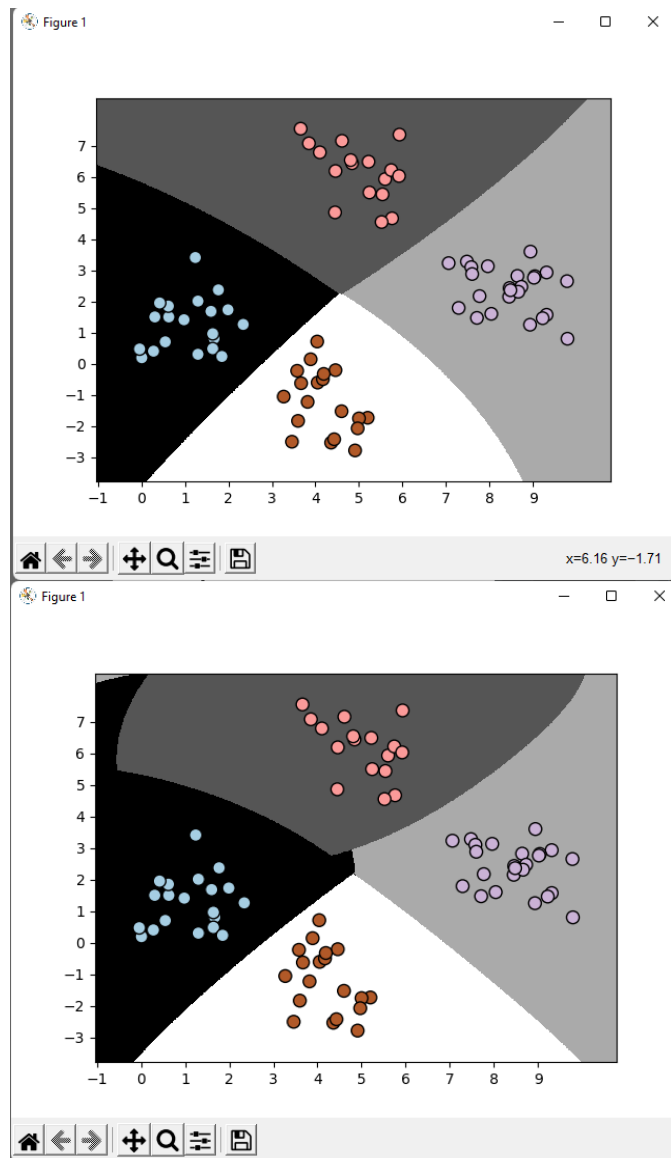


Рисунок 1.9 порівняння класифікаторів наївного байєса та SVM

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