

## ЛАБОРАТОРНА РОБОТА № 1

### ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА

### КЛАСИФІКАЦІЯ ДАНИХ

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

#### Завдання 2.1.1 Бінарізація.

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

Рис.1 Результат виконання завдання 2.1.1

#### Завдання 2.1.2 Виключення середнього.

```
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.]
```

Рис.2 Результат виконання завдання 2.1.2

#### Завдання 2.1.3 Масштабування.

```
Min max scaled data:  
[[0.74117647 0.39548023 1.      ]  
 [0.          1.          0.      ]  
 [0.6         0.5819209   0.87234043]  
 [1.          0.          0.17021277]]
```

Рис.3 Результат виконання завдання 2.1.3

					ДУ «Житомирська політехніка».22.121.04.000 – Лр1			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Демченко Я. Д.			Звіт з лабораторної роботи	Літ.	Арк.	Аркушів
Перевір.		Філіпов В. О.					1	9
Керівник						ФІКТ Гр. ІПЗ-19-2[1]		
Н. контр.								
Зав. каф.								

## Завдання 2.1.4 Нормалізація.

```
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.4 Результат виконання завдання 2.1.4

```
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]])

data_binarized = preprocessing.Binarizer(threshold=3.0).transform(input_data)
print("\n Binarized data:\n", data_binarized)
print("\nBEFORE: ")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))
data_scaled = preprocessing.scale(input_data)
print("\nAFTER: ")
print("Mean =", data_scaled.mean(axis=0))
print("Std deviation =", data_scaled.std(axis=0))
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)
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)
```

		Демченко Я. Д.			ДУ «Житомирська політехніка».22.121.04.000 - Лр1	Арк.
		Філіпов В. О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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 2.77555756e-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. 5 - 6 Результат виконання завдання 2.1.1 – 2.1.4

Як бачимо, L1-нормалізація менш чутлива до викидів.

#### Завдання 2.1.5 Кодування міток.

		Демченко Я. Д.			ДЧ «Житомирська політехніка».22.121.04.000 - Лр1	Арк.
		Філіпов В. О.				
Змн.	Арк.	№ докум.	Підпис	Дата		3

```

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

```

Run: LR\_1\_task\_1

```

"F:\for ZSTU\AI\Lab1\venv\Scripts\python.exe" "F:\for ZSTU\AI\Lab1\LR_1_task_1.py"

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']

Process finished with exit code 0

```

Рис. 7. Результат виконання завдання 2.1.5

## Завдання 2.2 Попередня обробка нових даних.

4.	-5.3	-8.9	3.0	2.9	5.1	-3.3	3.1	-2.8	-3.2	2.2	-1.4	5.1	3.0
----	------	------	-----	-----	-----	------	-----	------	------	-----	------	-----	-----

		Демченко Я. Д.			ДЧ «Житомирська політехніка».22.121.04.000 - Лр1	Арк.
		Філіпов В. О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

```

1 import numpy as np
2 from sklearn import preprocessing
3
4 input_data = np.array([[-5.3, -8.9, 3.0],
5                        [2.9, 5.1, -3.3],
6                        [3.1, -2.8, -3.2],
7                        [2.2, -1.4, 5.1]])
8
9 data_binarized = preprocessing.Binarizer(threshold=3.0).transform(input_data)
10 print("\n Binarized data:\n", data_binarized)
11 print("\nBEFORE: ")
12 print("Mean =", input_data.mean(axis=0))
13 print("Std deviation =", input_data.std(axis=0))
14 data_scaled = preprocessing.scale(input_data)
15 print("\nAFTER: ")
16 print("Mean =", data_scaled.mean(axis=0))
17 print("Std deviation =", data_scaled.std(axis=0))
18 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
19 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
20 print("\nMin max scaled data:\n", data_scaled_minmax)
21 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
22 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
23 print("\nL1 normalized data:\n", data_normalized_l1)
24 print("\nL2 normalized data:\n", data_normalized_l2)
25

```

```

Run: LR_1_task_2
"F:\for ZSTU\AI\Lab1\venv\Scripts\python.exe" "F:\for ZSTU

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

BEFORE:
Mean = [ 0.725 -2.      0.4 ]
Std deviation = [3.49454933 4.97543968 3.72491611]

AFTER:
Mean = [-2.77555756e-17 -2.42861287e-17  0.00000000e+00]
Std deviation = [1. 1. 1.]

Min max scaled data:
[[0.      0.      0.75     ]
 [0.97619048 1.      0.      ]
 [1.      0.43571429 0.01190476]
 [0.89285714 0.53571429 1.      ]]

L1 normalized data:
[[-0.30813953 -0.51744186  0.1744186 ]
 [ 0.25663717  0.45132743 -0.2920354 ]
 [ 0.34065934 -0.30769231 -0.35164835]
 [ 0.25287356 -0.16091954  0.5862069 ]]

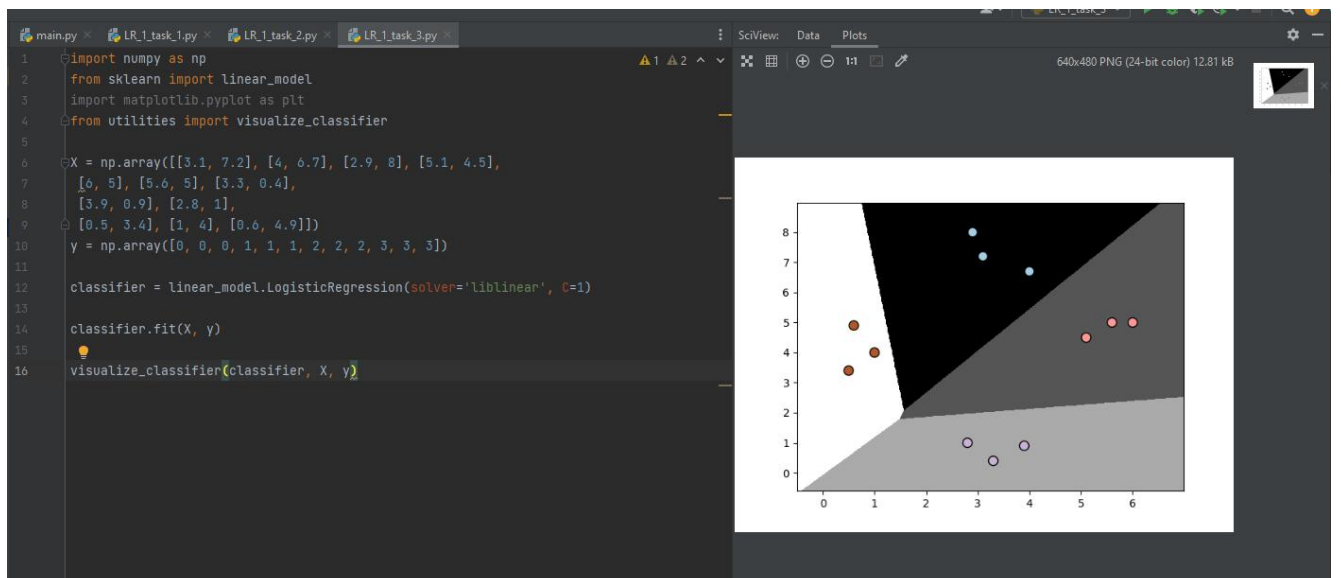
L2 normalized data:
[[-0.49145755 -0.82527777  0.27818352]
 [ 0.43082507  0.75765788 -0.49024922]
 [ 0.58911518 -0.53210404 -0.6081189 ]
 [ 0.38407812 -0.24441335  0.89036291]]

Process finished with exit code 0

```

Рис.8. Результат виконання завдання 2.2

## Завдання 2.3 Класифікація логістичною регресією або логістичний класифікатор.

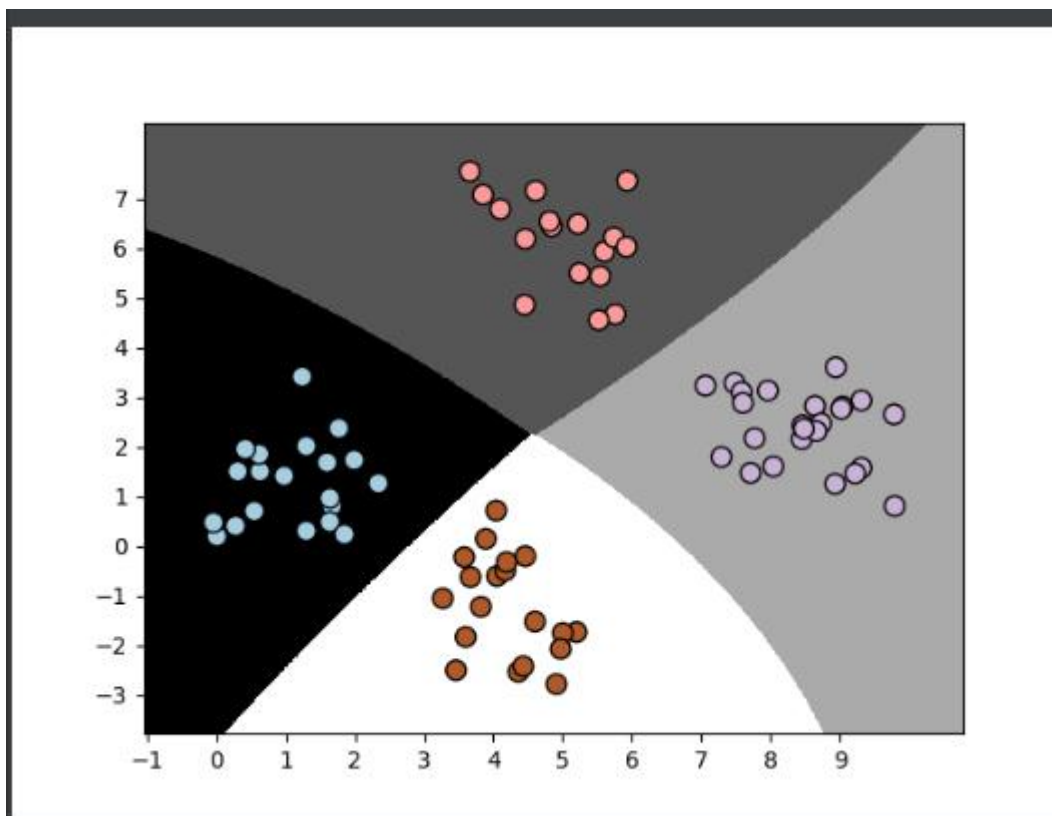


		Демченко Я. Д.			ДУ «Житомирська політехніка».22.121.04.000 – Лр1	Арк.
		Філіпов В. О.				
Змн.	Арк.	№ докум.	Підпис	Дата		5



Рис.9 Результат виконання завдання 2.3

## Завдання 2.4 Класифікація наївним байєсовським класифікатором.



```

1 import numpy as np
2 from sklearn.naive_bayes import GaussianNB
3 from sklearn.model_selection import train_test_split
4 from sklearn.model_selection import cross_val_score
5 from utilities import visualize_classifier
6
7 # Вхідний файл, який містить дані
8 input_file = 'data_multivar_nb.txt'
9
10 # Завантаження даних із вхідного файлу
11 data = np.loadtxt(input_file, delimiter=',')
12 X, y = data[:, :-1], data[:, -1]
13
14 # Створення наївного байєсовського класифікатора
15 classifier = GaussianNB()
16
17 # Тренування класифікатора
18 classifier.fit(X, y)
19
20 # Прогнозування значень для тренувальних даних
21 y_pred = classifier.predict(X)
22
23 # Обчислення якості класифікатора
24 accuracy = 100.0 * (y == y_pred).sum() / X.shape[0]
25 print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")
26
27 # Візуалізація результатів роботи класифікатора
28 visualize_classifier(classifier, X, y)
29
30 # Розбивка даних на навчальний та тестовий набори
31 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
32 classifier_new = GaussianNB()
33 classifier_new.fit(X_train, y_train)
34 y_test_pred = classifier_new.predict(X_test)
  
```

		Демченко Я. Д.			ДУ «Житомирська політехніка».22.121.04.000 - Лр1	Арк.
		Філіпов В. О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Accuracy of Naive Bayes classifier = 99.75 %
Accuracy of the new classifier = 100.0 %
Accuracy: 99.75%
Precision: 99.76%
Recall: 99.75%
F1: 99.75%

```

Рис. 10 - 12. Результат виконання завдання 2.4

### Завдання 2.5 Вивчити метрики якості класифікації.

```

[[5519 2360]
 [2832 5047]]
TP: 5047
FN: 2832
FP: 2360
TN: 5519
0.6705165630156111
Accuracy RF:0.671
0.6405635232897576
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

Scores with threshold = 0.75
Accuracy RF: 0.512
Recall RF: 0.025
Precision RF: 0.995
F1 RF: 0.049

AUC RF:0.738
AUC LR:0.666

```

Рис.13 - 15 Результат виконання завдання 2.5

З результатів бачимо, що при збільшенні порогу F1 міра зменшується.

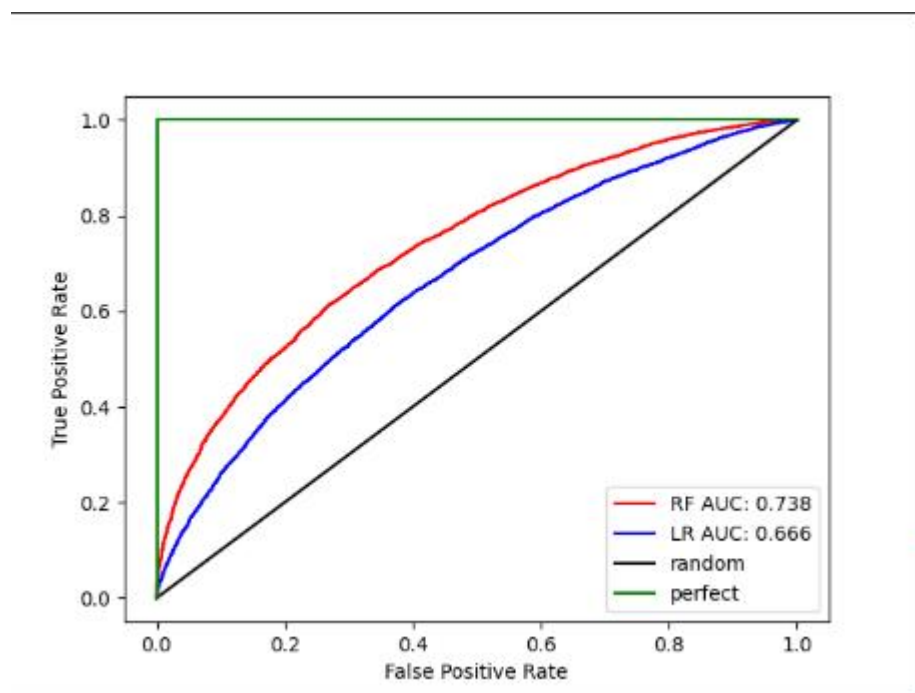


Рис.16 ROC - крива

На графіку бачимо, що RF модель виглядає краще, ніж LR модель ,  
 площа під кривою для моделі RF ( $AUC = 0,738$ ) краще, ніж LR ( $AUC = 0,666$ ).  
 Проте ефективність кожного з методів залежить від конкретної моделі.

**Завдання 2.6** Розробіть програму класифікації даних в файлі `data_multivar_nb.txt` за допомогою машини опорних векторів (Support Vector Machine - SVM). Розрахуйте показники якості класифікації. Порівняйте їх з показниками наївного байєсівського класифікатора. Зробіть висновки яку модель класифікації краще обрати і чому.

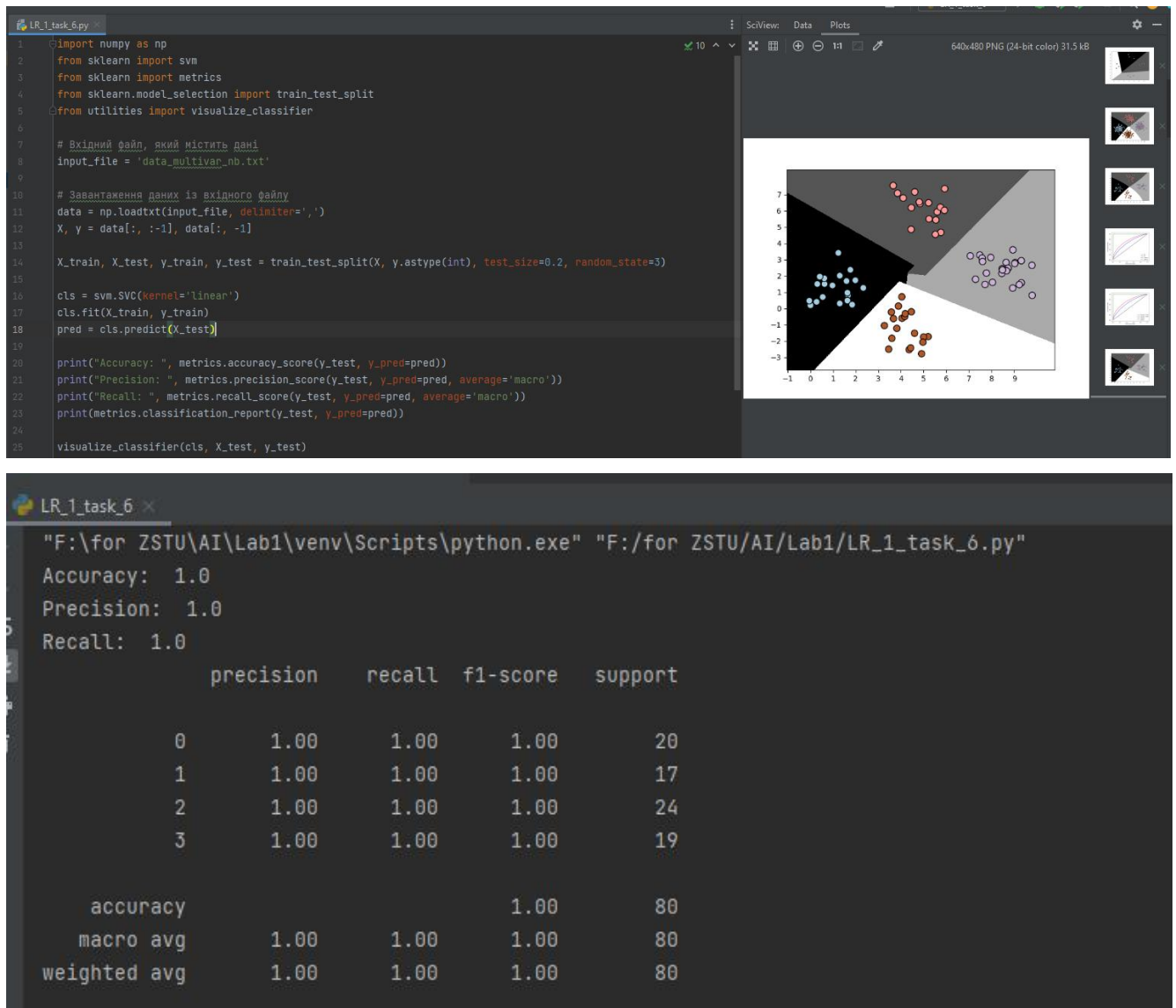


Рис.17 - 18 Результат виконання завдання 2.6



Наївний баєсівський класифікатор і метод опорних векторів (SVM) мають різні параметри, включаючи вибір функції ядра для кожного з них. Обидва алгоритми є дуже чутливими до оптимізації параметрів, тому вибір різних параметрів може суттєво вплинути на отримані результати. Для обраних параметрів NBC працює краще, ніж SVM, що видно в результатах. Проте, за інших параметрів, SVM може показати більш ефективні результати.

**Репозиторій:** <https://github.com/ipz192dyad/Artificial-intelligence-systems>

**Висновок:** в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python було досліджено попередню обробку та класифікацію даних.

		Демченко Я. Д.			ДЧ «Житомирська політехніка».22.121.04.000 - Лр1	Арк.
		Філіпов В. О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9