

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 – 2.1.4

```
1 import numpy as np
2 from sklearn import preprocessing
3
4 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]])
5
6 # Бінаризація даних
7 data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
8 print("\n Binarized data:\n", data_binarized)
9
10 # Виведення середнього значення та стандартного відхилення
11 print("\nBEFORE: ")
12 print("Mean =", input_data.mean(axis=0))
13 print("Std deviation =", input_data.std(axis=0))
14
15 # Исклучение среднего
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)
```

Рис 2.1 - Файл main.py

					ДУ «Житомирська політехніка».20.121.18			
Змн.	Арк.	№ докум.	Підпис	Дата	<div>Лім. Арк. Аркушів</div> <div>1 14</div> <div>ФІКТ Гр. ІПЗк-20-1</div>			
Розроб.		Соболевський Д.А.						
Перевір.		Філіпов В.О.						
Керівник								
Н. контр.								
Зав. каф.								

Результат:

```
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]]
PS C:\ztu\штучний інтелект>
```

Рис 2.2 – Результат виконання коду фалу main.py

Висновок: **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. Взагалі, техніка *L1-нормалізації* вважається більш надійною по порівняно з *L2-нормалізацією*, оскільки вона менш чутлива до викидів

Завдання 2.1.5

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

```

1 import numpy as np
2 from sklearn import preprocessing
3
4 # Надання позначок вхідних даних
5 input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow', 'white']
6
7 # Створення кодувальника та встановлення відповідності # між мітками та числами
8 encoder = preprocessing.LabelEncoder()
9 encoder.fit(input_labels)
10
11 # Виведення відображення
12 print("\nLabel mapping:")
13 for i, item in enumerate(encoder.classes_):
14     print(item, '-->', i)
15
16 # перетворення міток за допомогою кодувальника
17 test_labels = ['green', 'red', 'black']
18 encoded_values = encoder.transform(test_labels)
19
20 print("\nLabels =", test_labels )
21 print("Encoded values =", list(encoded_values))
22
23 # Декодування набору чисел за допомогою декодера
24
25 encoded_values = [3, 0, 4, 1]
26 decoded_list = encoder.inverse_transform(encoded_values)
27
28 print("\nEncoded values =", encoded_values)
29 print("Decoded labels =", list(decoded_list))

```

Рис 2.3 Код файлу LR_1_task1.py

Результат:

```

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

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

Encoded values = [3, 0, 4, 1]
Decoded labels = ['white', 'black', 'yellow', 'green']
PS C:\ztu\штучний інтелект>

```

Рис 2.4 Результат файлу LR_1_task1.py

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2.2 Попередня обробка нових даних

18.	4.6	3.9	-3.5	-2.9	4.1	3.3	2.2	8.8	-6.1	3.9	1.4	2.2	2.2
-----	-----	-----	------	------	-----	-----	-----	-----	------	-----	-----	-----	-----

```

1 import numpy as np
2 from sklearn import preprocessing
3
4 input_data = np.array([[4.6, 3.9, -3.5], [-2.9, 4.1, 3.3], [2.2, 8.8, -6.1], [3.9, 1.4, 2.2]])
5 threshold_limit = 2.2
6
7 # Бінаризація даних
8 data_binarized = preprocessing.Binarizer(threshold = threshold_limit).transform(input_data)
9 print("\n Binarized data:\n", data_binarized)
10
11 # Виведення середнього значення та стандартного відхилення
12 print("\nBEFORE: ")
13 print("Mean =", input_data.mean(axis = 0))
14 print("Std deviation =", input_data.std(axis = 0))
15
16 # Исключение среднего
17 data_scaled = preprocessing.scale(input_data)
18 print("\nAFTER: ")
19 print("Mean =", data_scaled.mean(axis = 0))
20 print("Std deviation =", data_scaled.std(axis = 0))
21
22 # Масштабування MinMax
23 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range = (0, 1))
24 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
25 print("\nMin max scaled data:\n", data_scaled_minmax)
26
27 # Нормалізація даних
28 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
29 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
30 print("\nl1 normalized data:\n", data_normalized_l1)
31 print("\nl2 normalized data:\n", data_normalized_l2)

```

Рис 2.5. Код файлу LR_1_task2.py

Результат:

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

BEFORE:
Mean = [ 1.95  4.55 -1.025]
Std deviation = [2.93300188 2.67441582 3.9047247 ]

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

Min max scaled data:
[[1.          0.33783784 0.27659574]
 [0.          0.36486486 1.          ]
 [0.68         1.          0.          ]
 [0.90666667 0.          0.88297872]]

l1 normalized data:
[[ 0.38333333  0.325      -0.29166667]
 [-0.2815534   0.39805825  0.32038835]
 [ 0.12865497  0.51461988 -0.35672515]
 [ 0.52         0.18666667  0.29333333]]

l2 normalized data:
[[ 0.65970588  0.55931585 -0.50195013]
 [-0.4825966   0.68229174  0.54916164]
 [ 0.20125974  0.80503895 -0.55803836]
 [ 0.83129388  0.29841319  0.46893501]]
PS C:\ztu\штучний інтелект>

```

Рис 2.6 Результат файлу LR_1_task2.py

Завдання 2.3. Класифікація логістичною регресією або логістичний класифікатор

```

1 import numpy as np
2 from sklearn import linear_model
3 import matplotlib.pyplot as plt
4 from utilities import visualize_classifier
5
6 # Визначення зразка вхідних даних
7 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]])
8 y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])
9
10 # Створення логістичного класифікатора
11 classifier = linear_model.LogisticRegression(solver = 'liblinear', C = 1)
12 # Тренування класифікатора
13 classifier.fit(X, y)
14
15 visualize_classifier(classifier, X, y)

```

Рис 2.7 Код файлу LR_1_task3.py

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат:

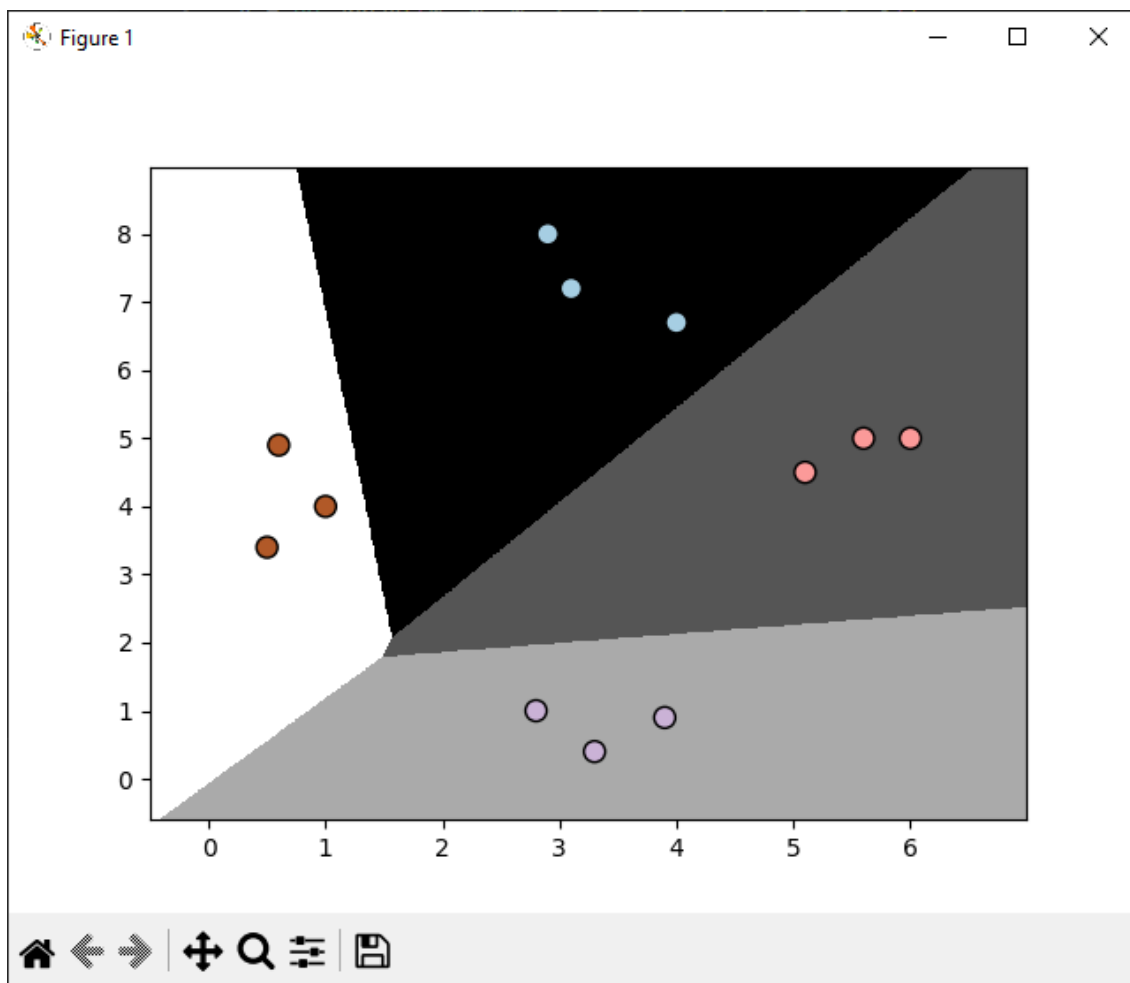


Рис 2.8 Результат файлу LR_1_task3.py

Завдання 2.4

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.naive_bayes import GaussianNB
4 from sklearn.model_selection import train_test_split
5 from sklearn.model_selection import cross_val_score
6 from utilities import visualize_classifier
7
8 # Вхідний файл, який містить дані
9 input_file = 'data_multivar_nb.txt'
10
11 # Завантаження даних із вхідного файлу
12 data = np.loadtxt(input_file, delimiter=',')
13 X, y = data[:, :-1], data[:, -1]
14
15 # Створення наївного байєсовського класифікатора
16 classifier = GaussianNB()
17
18 # Тренування класифікатора
19 classifier.fit(X, y)
20
21 # Прогнозування значень для тренувальних даних
22 y_pred = classifier.predict(X)
23
24 # Обчислення якості класифікатора
25 accuracy = 100.0 * (y == y_pred).sum() / X.shape[0]
26
27 print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")
28
29 # Візуалізація результатів роботи класифікатора
30 visualize_classifier(classifier, X, y)
31
32 # Розбивка даних на навчальний та тестовий набори
33 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 3)
34 classifier_new = GaussianNB()
35
36 classifier_new.fit(X_train, y_train)
37
38 y_test_pred = classifier_new.predict(X_test)
39
40 # Обчислення якості класифікатора
41 accuracy = 100.0 * (y_test == y_test_pred).sum() / X_test.shape[0]
42
43 print("Accuracy of the new classifier =", round(accuracy, 2), "%")
44 # Візуалізація роботи класифікатора
45 visualize_classifier(classifier_new, X_test, y_test)
46
47 num_folds = 3
48 accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=num_folds)
49
50 print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
51
52 precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=num_folds)
53
54 print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
55
56 recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=num_folds)
57
58 print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
59
60 f1_values = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=num_folds)
61
62 print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")

```

Рис 2.9 Код файлу LR_1_task4.py

Результат:

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

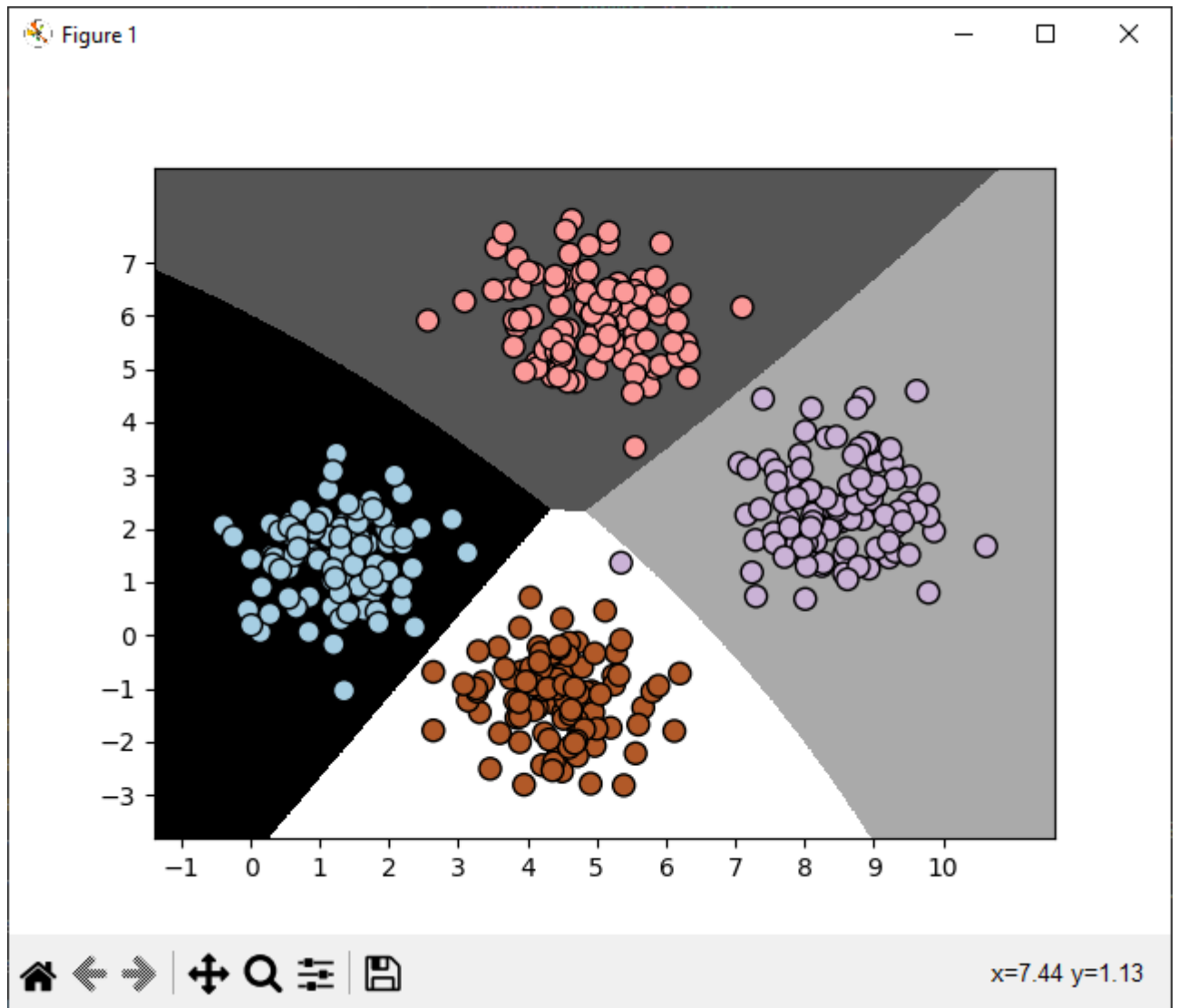


Рис 2.10 Результат файлу LR_1_task4.py

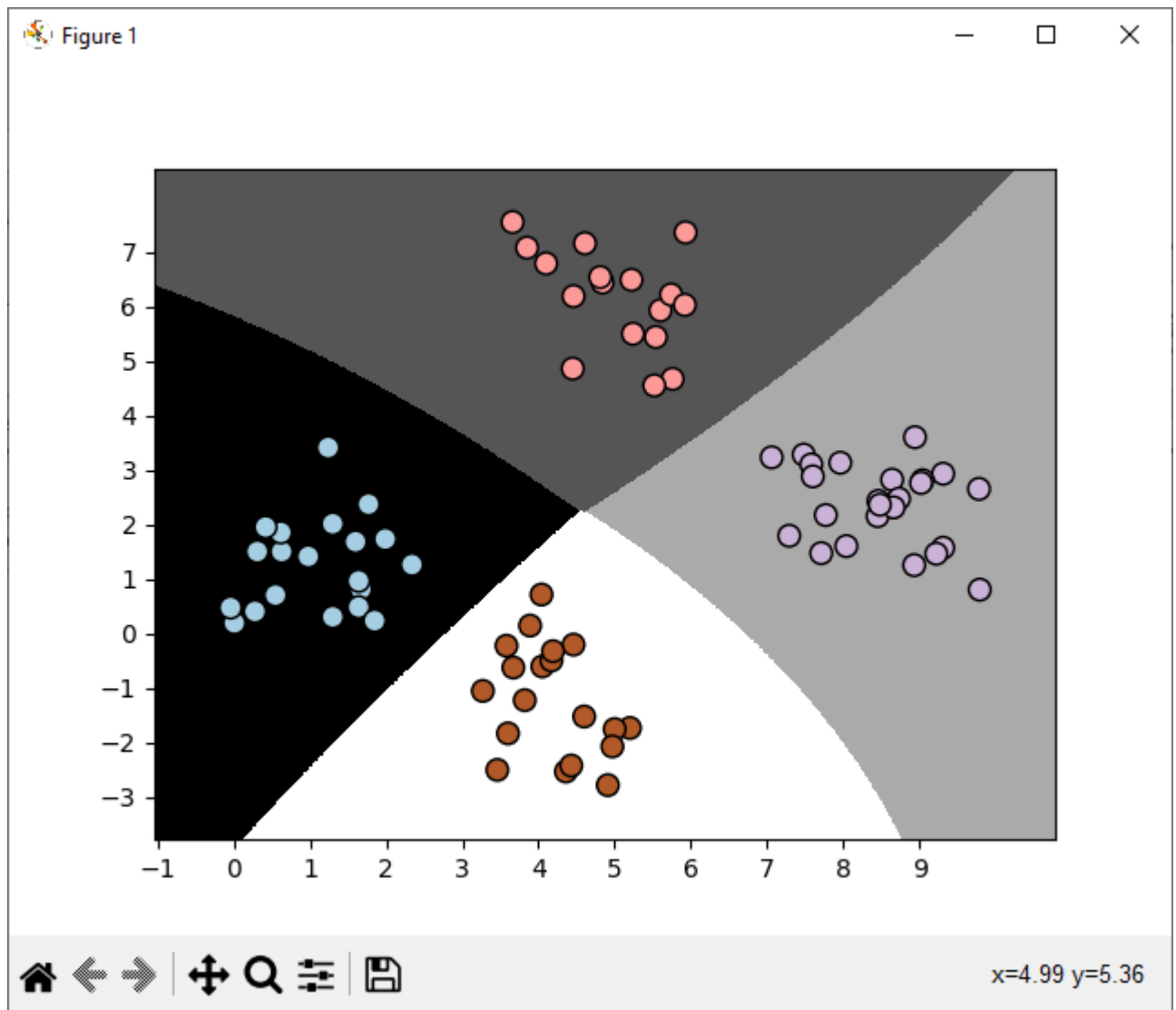


Рис 2.11 Результат файлу LR_1_task4.py

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Accuracy of Naive Bayes classifier = 99.75 %
Accuracy of the new classifier = 100.0 %

```

Рис 2.12 Результат файлу LR_1_task4.py

Завдання 2.5. Вивчити метрики якості класифікації

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.metrics import confusion_matrix
5 from sklearn.metrics import accuracy_score
6 from sklearn.metrics import recall_score
7 from sklearn.metrics import precision_score
8 from sklearn.metrics import f1_score
9 from sklearn.metrics import roc_curve
10 from sklearn.metrics import roc_auc_score
11
12 df = pd.read_csv('data_metrics.csv')
13 thresh = 0.5
14
15 df.head()
16
17 df['predicted_RF'] = (df.model_RF >= 0.5).astype('int')
18 df['predicted_LR'] = (df.model_LR >= 0.5).astype('int')
19
20 df.head()
21 print(confusion_matrix(df.actual_label.values, df.predicted_RF.values))
22
23
24 def find_TP(y_true, y_pred):
25     # counts the number of true positives (y_true = 1, y_pred = 1)
26     return sum((y_true == 1) & (y_pred == 1))
27
28 def find_FN(y_true, y_pred):
29     # counts the number of false negatives (y_true = 1, y_pred = 0)
30     return sum((y_true == 1) & (y_pred == 0))
31
32 def find_FP(y_true, y_pred):
33     # counts the number of false positives (y_true = 0, y_pred = 1)
34     return sum((y_true == 0) & (y_pred == 1))
35
36 def find_TN(y_true, y_pred):
37     # counts the number of true negatives (y_true = 0, y_pred = 0)
38     return sum((y_true == 0) & (y_pred == 0))
39
40 print('TP:', find_TP(df.actual_label.values, df.predicted_RF.values))
41 print('FN:', find_FN(df.actual_label.values, df.predicted_RF.values))
42 print('FP:', find_FP(df.actual_label.values, df.predicted_RF.values))
43 print('TN:', find_TN(df.actual_label.values, df.predicted_RF.values))
```

```

1 def find_conf_matrix_values(y_true, y_pred):
2     # calculate TP, FN, FP, TN
3     TP = find_TP(y_true, y_pred)
4     FN = find_FN(y_true, y_pred)
5     FP = find_FP(y_true, y_pred)
6     TN = find_TN(y_true, y_pred)
7     return TP, FN, FP, TN
8
9
10 def sobolevskiy_confusion_matrix(y_true, y_pred):
11     TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
12
13     return np.array([[TN, FP], [FN, TP]])
14
15
16 sobolevskiy_confusion_matrix(df.actual_label.values, df.predicted_RF.values)
17
18 assert np.array_equal(
19     sobolevskiy_confusion_matrix(df.actual_label.values, df.predicted_RF.values),
20     confusion_matrix(df.actual_label.values, df.predicted_RF.values)
21 ), 'my_confusion_matrix() is not correct for RF'
22 assert np.array_equal(
23     sobolevskiy_confusion_matrix(df.actual_label.values, df.predicted_LR.values),
24     confusion_matrix(df.actual_label.values, df.predicted_LR.values)
25 ), 'my_confusion_matrix() is not correct for LR'
26
27 print(accuracy_score(df.actual_label.values, df.predicted_RF.values))
28
29 def sobolevskiy_accuracy_score(y_true, y_pred): # calculates the fraction of samples
30     TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
31     return (TP + TN) / (TP + TN + FP + FN)
32
33 assert sobolevskiy_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'
34 assert sobolevskiy_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'
35
36 print('Accuracy RF: %.3f' % (sobolevskiy_accuracy_score(df.actual_label.values, df.predicted_RF.values)))
37 print(recall_score(df.actual_label.values, df.predicted_RF.values))

```

```

1 def sobolevskiy_recall_score(y_true, y_pred):
2     # calculates the fraction of positive samples predicted correctly
3     TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
4     return TP / (TP + FN)
5
6
7 assert sobolevskiy_recall_score(df.actual_label.values, df.predicted_RF.values) == recall_score(df.actual_label.values, df.predicted_RF.values), 'sobolevskiy_accuracy_score failed on RF'
8 assert sobolevskiy_recall_score(df.actual_label.values, df.predicted_LR.values) == recall_score(df.actual_label.values, df.predicted_LR.values), 'sobolevskiy_accuracy_score failed on LR'
9
10 print('Recall RF: %.3f' % (sobolevskiy_recall_score(df.actual_label.values, df.predicted_RF.values)))
11 print('Recall LR: %.3f' % (sobolevskiy_recall_score(df.actual_label.values, df.predicted_LR.values)))
12 precision_score(df.actual_label.values, df.predicted_RF.values)
13
14 def sobolevskiy_precision_score(y_true, y_pred):
15     # calculates the fraction of predicted positives samples that are actually positive
16     TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
17     return TP / (TP + FP)
18
19 assert sobolevskiy_precision_score(df.actual_label.values, df.predicted_RF.values) == precision_score(df.actual_label.values, df.predicted_RF.values), 'my_accuracy_score failed on RF'
20 assert sobolevskiy_precision_score(df.actual_label.values, df.predicted_LR.values) == precision_score(df.actual_label.values, df.predicted_LR.values), 'my_accuracy_score failed on LR'
21
22 print('Precision RF: %.3f' % (sobolevskiy_precision_score(df.actual_label.values, df.predicted_RF.values)))
23 print('Precision LR: %.3f' % (sobolevskiy_precision_score(df.actual_label.values, df.predicted_LR.values)))
24 f1_score(df.actual_label.values, df.predicted_RF.values)
25
26 def sobolevskiy_f1_score(y_true, y_pred): # calculates the F1 score
27     recall = sobolevskiy_recall_score(y_true, y_pred)
28     precision = sobolevskiy_precision_score(y_true, y_pred)
29     return 2 * (precision * recall) / (precision + recall)
30
31 assert sobolevskiy_f1_score(df.actual_label.values, df.predicted_RF.values) == f1_score(df.actual_label.values, df.predicted_RF.values), 'my_accuracy_score failed on RF'
32 assert sobolevskiy_f1_score(df.actual_label.values, df.predicted_LR.values) == f1_score(df.actual_label.values, df.predicted_LR.values), 'my_accuracy_score failed on LR'
33

```

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1 print('F1 RF: %.3f' % (sobolevskiy_f1_score(df.actual_label.values, df.predicted_RF.values)))
2 print('F1 LR: %.3f' % (sobolevskiy_f1_score(df.actual_label.values, df.predicted_LR.values)))
3 print('scores with threshold = 0.5')
4 print('Accuracy RF: %.3f' % (sobolevskiy_accuracy_score(df.actual_label.values, df.predicted_RF.values)))
5 print('Recall RF: %.3f' % (sobolevskiy_recall_score(df.actual_label.values, df.predicted_RF.values)))
6 print('Precision RF: %.3f' % (sobolevskiy_precision_score(df.actual_label.values, df.predicted_RF.values)))
7 print('F1 RF: %.3f' % (sobolevskiy_f1_score(df.actual_label.values, df.predicted_RF.values)))
8 print('')
9
10 threshold = 0.75
11
12 print(f'Scores with threshold = {threshold}')
13 print('Accuracy RF: %.3f' % (sobolevskiy_accuracy_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
14 print('Recall RF: %.3f' % (sobolevskiy_recall_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
15 print('Precision RF: %.3f' % (sobolevskiy_precision_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
16 print('F1 RF: %.3f' % (sobolevskiy_f1_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
17
18 fpr_RF, tpr_RF, thresholds_RF = roc_curve(df.actual_label.values, df.model_RF.values)
19 fpr_LR, tpr_LR, thresholds_LR = roc_curve(df.actual_label.values, df.model_LR.values)
20
21 plt.plot(fpr_RF, tpr_RF, 'r-', label='RF')
22 plt.plot(fpr_LR, tpr_LR, 'b-', label='LR')
23 plt.plot([0, 1], [0, 1], 'k-', label='random')
24 plt.plot([0, 0, 1, 1], [0, 1, 1, 1], 'g-', label='perfect')
25 plt.legend()
26 plt.xlabel('False Positive Rate')
27 plt.ylabel('True Positive Rate')
28 plt.show()
29
30 auc_RF = roc_auc_score(df.actual_label.values, df.model_RF.values)
31 auc_LR = roc_auc_score(df.actual_label.values, df.model_LR.values)
32 print('AUC RF: %.3f' % auc_RF)
33 print('AUC LR: %.3f' % auc_LR)
34
35 plt.plot(fpr_RF, tpr_RF, 'r-', label='RF AUC: %.3f' % auc_RF)
36 plt.plot(fpr_LR, tpr_LR, 'b-', label='LR AUC: %.3f' % auc_LR)
37 plt.plot([0, 1], [0, 1], 'k-', label='random')
38 plt.plot([0, 0, 1, 1], [0, 1, 1, 1], 'g-', label='perfect')
39 plt.legend()
40 plt.xlabel('False Positive Rate')
41 plt.ylabel('True Positive Rate')
42 plt.show()

```

Рис. 2.13 Код файлу LR_1_task5.py

Результат:

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```

keyBoardInterface
PS C:\ztu\штучний інтелект\lab1> python LR_1_task_5.py
[[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

```

Рис 2.14 Результат файлу LR_1_task5.py

F1 міра зменшується в результаті збільшення порогу.

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

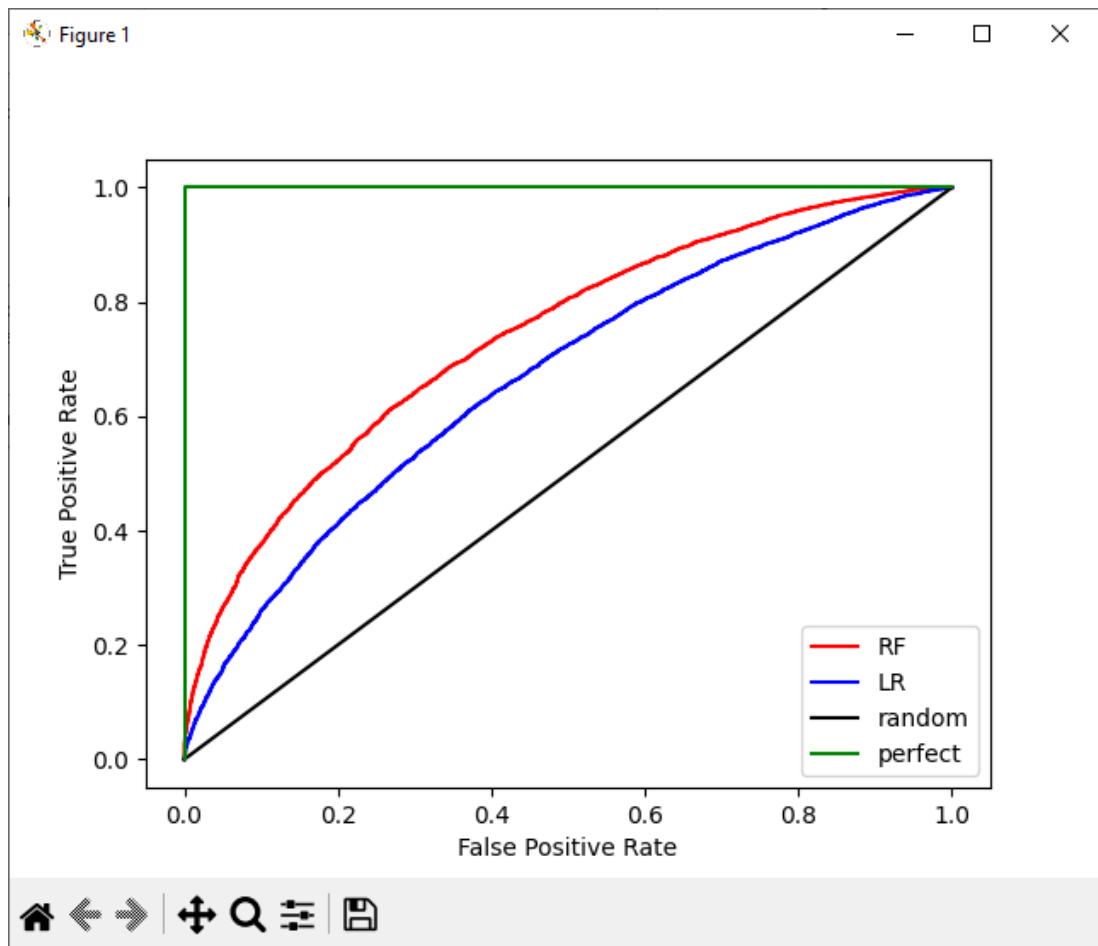


Рис 2.15. ROC – крива.

Подивившись на модель, бачимо що RF модель має більшу зрозумілість, аніж LR модель. Але залежить також і від складності моделі. Тому це не завжди є основним показником.

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2.6

```

1 import numpy as np
2 from sklearn import datasets
3 from sklearn.model_selection import train_test_split
4 from sklearn import svm
5 from sklearn import metrics
6
7 from utilities import visualize_classifier
8 input_file = 'data_multivar_nb.txt'
9
10 data = np.loadtxt(input_file, delimiter=',')
11 X, y = data[:, :-1], data[:, -1]
12 X_train, X_test, y_train, y_test = train_test_split(X, y.astype(int), test_size=0.2, random_state=3)
13 cls = svm.SVC(kernel='linear')
14 cls.fit(X_train, y_train)
15 pred = cls.predict(X_test)
16 print("Accuracy:", metrics.accuracy_score(y_test, y_pred=pred))
17 print("Precision: ", metrics.precision_score(y_test, y_pred=pred, average='macro'))
18 print("Recall", metrics.recall_score(y_test, y_pred=pred, average='macro'))
19 print(metrics.classification_report(y_test, y_pred=pred))
20 visualize_classifier(cls, X_test, y_test)

```

Рис. 2.16 Код файлу LR_1_task6.py

```

PS C:\ztu\штучний інтелект\lab1> python LR_1_task_6.py
Accuracy: 1.0
Precision: 1.0
Recall 1.0

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	1.00	1.00	1.00	17
2	1.00	1.00	1.00	24
3	1.00	1.00	1.00	19
accuracy			1.00	80
macro avg	1.00	1.00	1.00	80
weighted avg	1.00	1.00	1.00	80

Рис 2.17 Результат файлу LR_1_task6.py

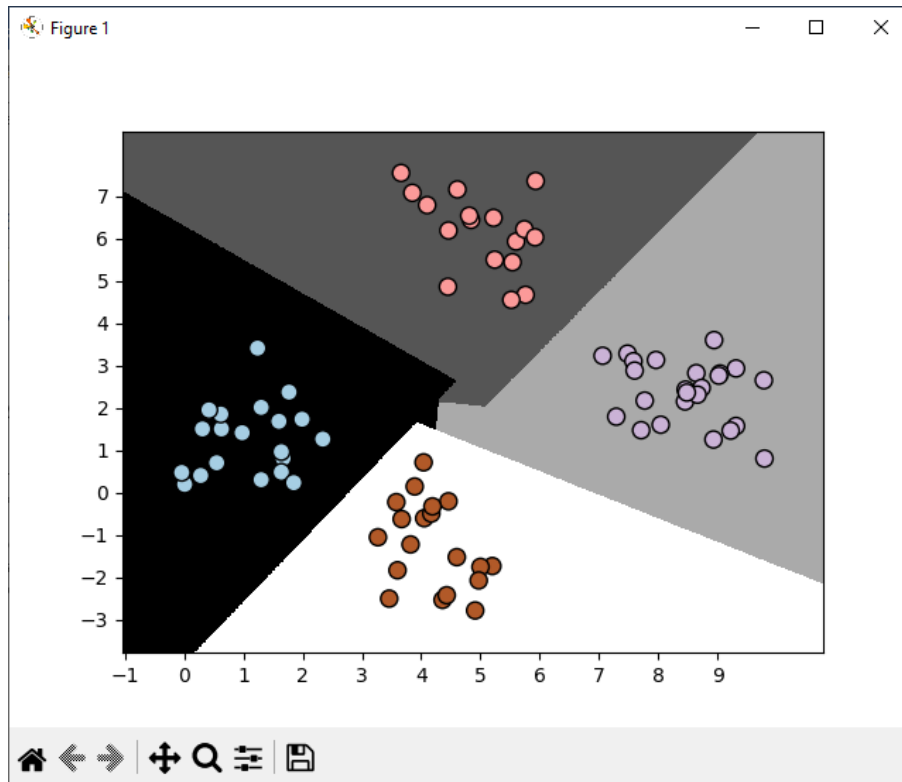


Рис 2.18 Результат файлу LR_1_task6.py

Висновок: після виконання лабораторної роботи навчився використовувати спеціалізовані бібліотеки та мову програмування Python, дослідити попередню обробку та класифікацію даних.

		Соболевський Д.А.			ДУ «Житомирська політехніка».20.121.18 – Лр1	Арк.
		Філіпов В.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		