

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».20.121.12.			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Надворний М.Ю.			ФІКТ Гр. ІПЗк-20-1			
Перевір.		Філіпов В.О.						
Керівник								
Н. контр.								
Зав. каф.								
					Літ.	Арк.	Аркушів	
						1	14	

Результат:

```
[[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 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

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2.1.5

```

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_): print(item, '-->', i)
14
15 # перетворення міток за допомогою кодувальника
16 test_labels = ['green', 'red', 'black']
17 encoded_values = encoder.transform(test_labels)
18 print("\nLabels =", test_labels)
19 print("Encoded values =", list(encoded_values))
20
21 # Декодування набору чисел за допомогою декодера
22
23 encoded_values = [3, 0, 4, 1]
24 decoded_list = encoder.inverse_transform(encoded_values)
25 print("\nEncoded values =", encoded_values)
26 print("Decoded labels =", list(decoded_list))

```

Рис 2.3 Код файлу LR_1_task1.py

Результат:

```

"D:\Штучний інтелект\Lab1\lab1\venv\Scripts\python.exe" "D:\
...
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']

Process finished with exit code 0

```

Рис 2.4 Результат файлу LR_1_task1.py

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2.2 Попередня обробка нових даних

12.	-1.3	3.9	4.5	-5.3	-4.2	3.3	-5.2	-6.5	-1.1	-5.2	2.6	-2.2	1.8
-----	------	-----	-----	------	------	-----	------	------	------	------	-----	------	-----

```

1  import numpy as np
2  from sklearn import preprocessing
3
4  input_data = np.array([[ -1.3, 3.9, 4.5], [-5.3, -4.2, 3.3], [-5.2, -6.5, -1.1], [-5.2, 2.6, -2.2]])
5
6  # Бінаризація даних
7  data_binarized = preprocessing.Binarizer(threshold=1.8).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 2.5. Код файлу LR_1_task2.py

Результат:

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_1_task_2 x
[[0. 1. 1.]
 [0. 0. 1.]
 [0. 0. 0.]
 [0. 1. 0.]]

BEFORE:
Mean = [-4.25 -1.05  1.125]
Std deviation = [1.7036725  4.40028408 2.83405628]

AFTER:
Mean = [0.00000000e+00 5.55111512e-17 0.00000000e+00]
Std deviation = [1. 1. 1.]

Min max scaled data:
[[1.      1.      1.      ]
 [0.      0.22115385 0.82089552]
 [0.025   0.      0.1641791 ]
 [0.025   0.875   0.      ]]

l1 normalized data:
[[-0.13402062  0.40206186  0.46391753]
 [-0.4140625  -0.328125    0.2578125 ]
 [-0.40625    -0.5078125  -0.0859375 ]
 [-0.52        0.26       -0.22        ]]

l2 normalized data:
[[-0.21328678  0.63986035  0.7383004 ]
 [-0.70435392 -0.55816726  0.43855999]
 [-0.61931099 -0.77413873 -0.13100809]
 [-0.83653629  0.41826814 -0.3539192 ]]

```

Рис 2.6 Результат файлу LR_1_task2.py

Завдання 2.3. Класифікація логістичною регресією або логістичний класифікатор

```

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, 1], [0.7, 1], [0.8, 1], [0.9, 1], [1, 1], [1, 2], [2, 2, 3, 3, 3]])
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 2.7 Код файлу LR_1_task3.py

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат:

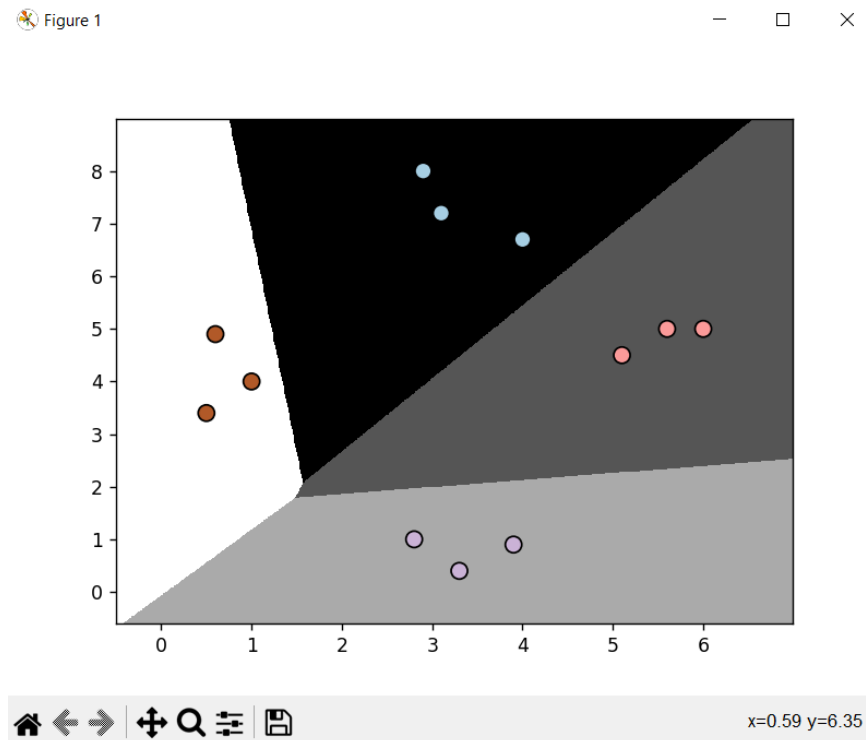


Рис 2.8 Результат файлу LR_1_task3.py

Завдання 2.4

```

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
7 from utilities import visualize_classifier
8
9 # Вхідний файл, який містить дані
10 input_file = 'data_multivar_nb.txt'
11
12 # Завантаження даних із вхідного файлу
13 data = np.loadtxt(input_file, delimiter=',')
14 X, y = data[:, :-1], data[:, -1]
15
16 # Створення наївного байєсовського класифікатора
17 classifier = GaussianNB()
18
19 # Тренування класифікатора
20 classifier.fit(X, y)
21
22 # Прогнозування значень для тренувальних даних
23 y_pred = classifier.predict(X)
24
25 # Обчислення якості класифікатора
26 accuracy = 100.0 * (y == y_pred).sum() / X.shape[0]
27 print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")
28
29 # Візуалізація результатів роботи класифікатора
30 visualize_classifier(classifier, X, y)

```

Рис 2.9 Код файлу LR_1_task4.py

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат:

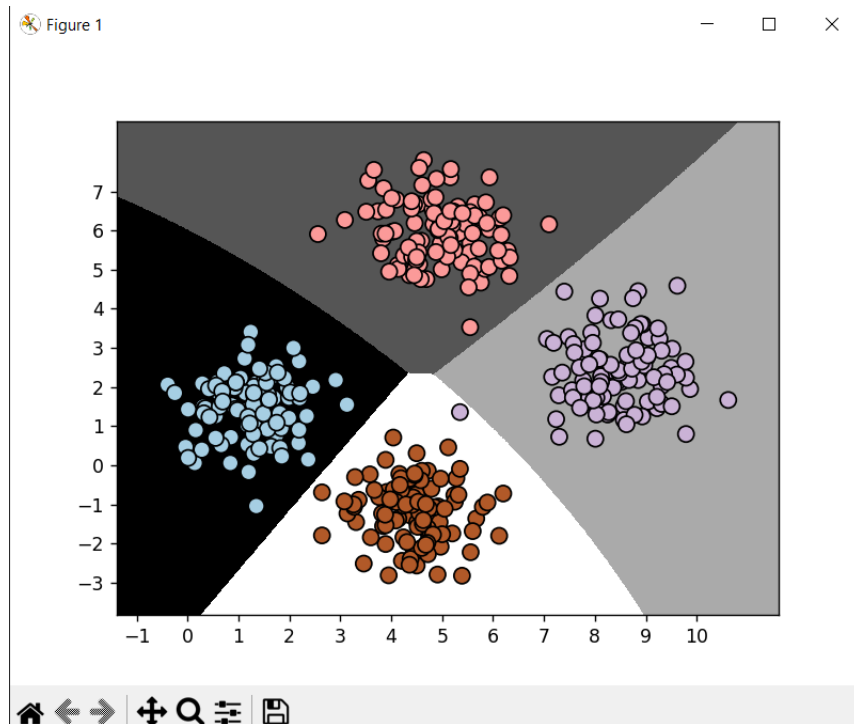


Рис 2.10 Результат файлу LR_1_task4.py

```

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 classifier_new.fit(X_train, y_train)
36 y_test_pred = classifier_new.predict(X_test)
37
38 # Обчислення якості класифікатора
39 accuracy = 100.0 * (y_test == y_test_pred).sum() / X_test.shape[0]
40 print("Accuracy of the new classifier =", round(accuracy, 2), "%")
41 # Візуалізація роботи класифікатора
42 visualize_classifier(classifier_new, X_test, y_test)
43
44 num_folds = 3
45 accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=num_folds)
46 print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
47 precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=num_folds)
48 print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
49 recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=num_folds)
50 print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
51 f1_values = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=num_folds)
52 print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")

```

Рис 2.11 Код файлу LR_1_task4.py

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

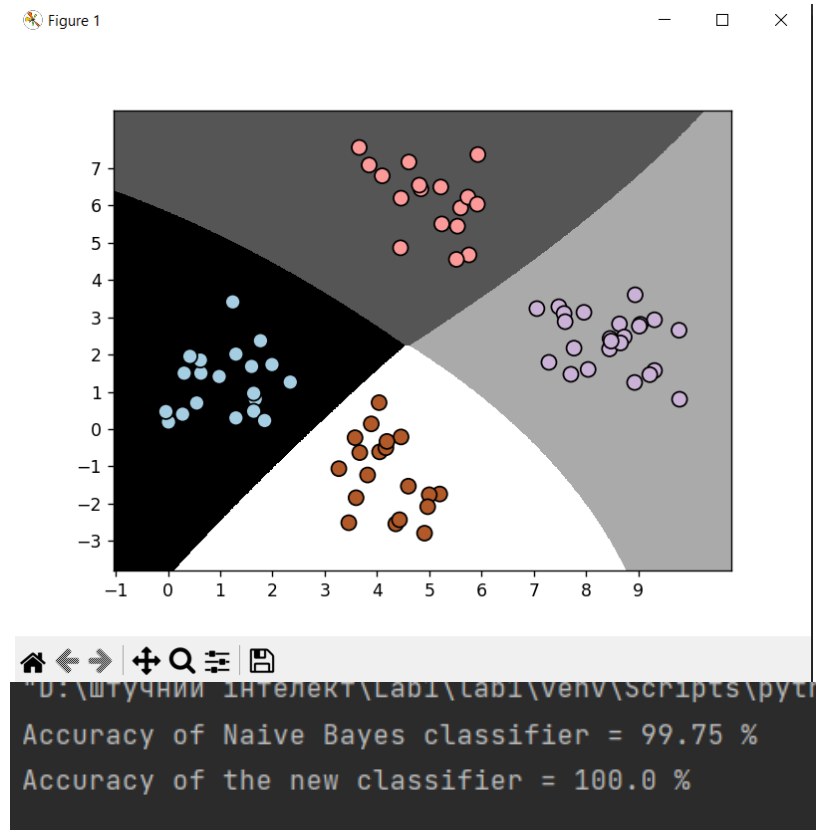


Рис 2.12 Результат файлу LR_1_task4.py

Завдання 2.5. Вивчити метрики якості класифікації

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		


```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 from sklearn.metrics import confusion_matrix
6 from sklearn.metrics import accuracy_score
7 from sklearn.metrics import recall_score
8 from sklearn.metrics import precision_score
9 from sklearn.metrics import f1_score
10 from sklearn.metrics import roc_curve
11 from sklearn.metrics import roc_auc_score
12
13 df = pd.read_csv('data_metrics.csv')
14 df.head()
15 thresh = 0.5
16 df['predicted_RF'] = (df.model_RF >= 0.5).astype('int')
17 df['predicted_LR'] = (df.model_LR >= 0.5).astype('int')
18 df.head()
19
20 print(confusion_matrix(df.actual_label.values, df.predicted_RF.values))
21
22 new *
23 def find_TP(y_true, y_pred):
24     # counts the number of true positives (y_true = 1, y_pred = 1)
25     return sum((y_true == 1) & (y_pred == 1))
26
27 def find_FN(y_true, y_pred):
28     # counts the number of false negatives (y_true = 1, y_pred = 0)
29     return sum((y_true == 1) & (y_pred == 0))
30
31 new *
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 new *
37 def find_TN(y_true, y_pred):
38     # counts the number of true negatives (y_true = 0, y_pred = 0)
39     return sum((y_true == 0) & (y_pred == 0))
40
41 print('TP:', find_TP(df.actual_label.values, df.predicted_RF.values))
42 print('FN:', find_FN(df.actual_label.values, df.predicted_RF.values))
43 print('FP:', find_FP(df.actual_label.values, df.predicted_RF.values))
44 print('TN:', find_TN(df.actual_label.values, df.predicted_RF.values))
45
46 new *
47 def find_conf_matrix_values(y_true, y_pred):
48     # calculate TP, FN, FP, TN
49     TP = find_TP(y_true, y_pred)
50     FN = find_FN(y_true, y_pred)
51     FP = find_FP(y_true, y_pred)

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

55         return TP, FN, FP, TN
56
57     new *
58     def nadvorny_confusion_matrix(y_true, y_pred):
59         TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
60         return np.array([[TN, FP], [FN, TP]])
61
62     nadvorny_confusion_matrix(df.actual_label.values, df.predicted_RF.values)
63
64     assert np.array_equal(nadvorny_confusion_matrix(df.actual_label.values, df.predicted_RF.values),
65                           confusion_matrix(df.actual_label.values,
66                                           df.predicted_RF.values)), 'my_confusion_matrix() is not correct for RF'
67
68     assert np.array_equal(nadvorny_confusion_matrix(df.actual_label.values, df.predicted_LR.values),
69                           confusion_matrix(df.actual_label.values,
70                                           df.predicted_LR.values)), 'my_confusion_matrix() is not correct for LR'
71
72     print(accuracy_score(df.actual_label.values, df.predicted_RF.values))
73
74     new *
75     def nadvorny_accuracy_score(y_true, y_pred): # calculates the fraction of samples
76         TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
77         return (TP + TN) / (TP + TN + FP + FN)
78
79
80     assert nadvorny_accuracy_score(df.actual_label.values, df.predicted_RF.values) == accuracy_score(
81         df.actual_label.values, df.predicted_RF.values), 'my_accuracy_score failed on RF'
82     assert nadvorny_accuracy_score(df.actual_label.values, df.predicted_LR.values) == accuracy_score(
83         df.actual_label.values, df.predicted_LR.values), 'my_accuracy_score failed on LR'
84     print('Accuracy RF: %.3f' % (nadvorny_accuracy_score(df.actual_label.values, df.predicted_RF.values)))
85
86     print(recall_score(df.actual_label.values, df.predicted_RF.values))
87
88     new *
89     def nadvorny_recall_score(y_true, y_pred):
90         # calculates the fraction of positive samples predicted correctly
91         TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
92         return TP / (TP + FN)
93
94
95     assert nadvorny_recall_score(df.actual_label.values, df.predicted_RF.values) == recall_score(df.actual_label.values,
96                                                                                               df.predicted_RF.values), 'nadvorny_recall_score failed on RF'
97     assert nadvorny_recall_score(df.actual_label.values, df.predicted_LR.values) == recall_score(df.actual_label.values,
98                                                                                               df.predicted_LR.values), 'nadvorny_recall_score failed on LR'
99
100    print('Recall RF: %.3f' % (nadvorny_recall_score(df.actual_label.values, df.predicted_RF.values)))
101    print('Recall LR: %.3f' % (nadvorny_recall_score(df.actual_label.values, df.predicted_LR.values)))
102
103    precision_score(df.actual_label.values, df.predicted_RF.values)
104
105    new *
106    def nadvorny_precision_score(y_true, y_pred):
107        # calculates the fraction of predicted positives samples that are actually positive
108        TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
109        return TP / (TP + FP)
110
111
112    assert nadvorny_precision_score(df.actual_label.values, df.predicted_RF.values) == precision_score(
113        df.actual_label.values, df.predicted_RF.values), 'my_precision_score failed on RF'
114    assert nadvorny_precision_score(df.actual_label.values, df.predicted_LR.values) == precision_score(
115        df.actual_label.values, df.predicted_LR.values), 'my_precision_score failed on LR'
116
117    print('Precision RF: %.3f' % (nadvorny_precision_score(df.actual_label.values, df.predicted_RF.values)))
118    print('Precision LR: %.3f' % (nadvorny_precision_score(df.actual_label.values, df.predicted_LR.values)))
119
120    f1_score(df.actual_label.values, df.predicted_RF.values)
121
122    new *
123    def nadvorny_f1_score(y_true, y_pred): # calculates the F1 score
124        recall = nadvorny_recall_score(y_true, y_pred)
125        precision = nadvorny_precision_score(y_true, y_pred)
126        return 2 * (precision * recall) / (precision + recall)
127
128
129    assert nadvorny_f1_score(df.actual_label.values, df.predicted_RF.values) == f1_score(df.actual_label.values,
130                                                                                          df.predicted_RF.values), 'my_f1_score failed on RF'
131    assert nadvorny_f1_score(df.actual_label.values, df.predicted_LR.values) == f1_score(df.actual_label.values,
132                                                                                          df.predicted_LR.values), 'my_f1_score failed on LR'

```

		Надворний М.Ю		
		Філіпов В.О.		
Змн.	Арк.	№ докум.	Підпис	Дата

ДУ «Житомирська політехніка».20.121.12 – Лр1

Арк.

10

```

print('F1 LR: %.3f' % (nadvornyi_f1_score(df.actual_label.values, df.predicted_LR.values)))
print('scores with threshold = 0.5')

print('Accuracy RF: %.3f' % (nadvornyi_accuracy_score(df.actual_label.values, df.predicted_RF.values)))
print('Recall RF: %.3f' % (nadvornyi_recall_score(df.actual_label.values, df.predicted_RF.values)))
print('Precision RF: %.3f' % (nadvornyi_precision_score(df.actual_label.values, df.predicted_RF.values)))
print('F1 RF: %.3f' % (nadvornyi_f1_score(df.actual_label.values, df.predicted_RF.values)))
print('')

threshold = 0.75

print('Scores with threshold = {threshold}')
print('Accuracy RF: %.3f' % (nadvornyi_accuracy_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
print('Recall RF: %.3f' % (nadvornyi_recall_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
print('Precision RF: %.3f' % (nadvornyi_precision_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
print('F1 RF: %.3f' % (nadvornyi_f1_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))

fpr_RF, tpr_RF, thresholds_RF = roc_curve(df.actual_label.values, df.model_RF.values)
fpr_LR, tpr_LR, thresholds_LR = roc_curve(df.actual_label.values, df.model_LR.values)

plt.plot(fpr_RF, tpr_RF, 'r-', label='RF')
plt.plot(fpr_LR, tpr_LR, 'b-', label='LR')
plt.plot([0, 1], [0, 1], 'k-', label='random')
plt.plot([0, 0, 1, 1], [0, 1, 1, 1], 'g-', label='perfect')
plt.legend()
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

```

Рис. 2.13 Код файлу LR_1_task5.py

Результат:

```

"D:\Штучний інтелект\Lab1\Lab1\
[[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

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Scores with threshold = 0.25
Accuracy RF: 0.502
Recall RF: 1.000
Precision RF: 0.501
F1 RF: 0.668
AUC RF:0.738
AUC LR:0.666
```

```
Scores with threshold = 0.1
Accuracy RF: 0.500
Recall RF: 1.000
Precision RF: 0.500
F1 RF: 0.667
AUC RF:0.738
AUC LR:0.666
```

Рис 2.14 Результат файлу LR_1_task5.py

F1 міра зменшується в результаті збільшення порогу.

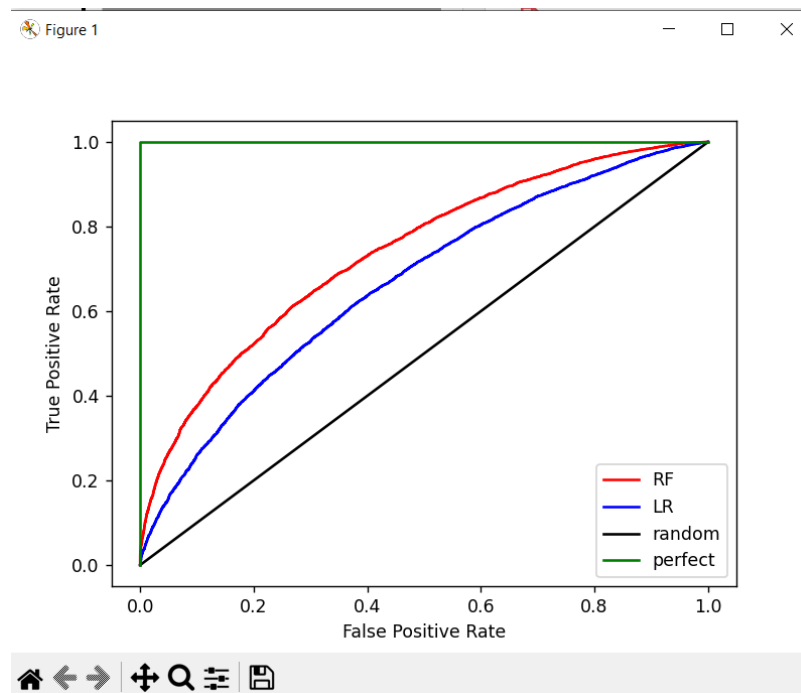


Рис 2.15. ROC – крива.

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

Завдання 2.6

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics

from utilities import visualize_classifier
input_file = 'data_multivar_nb.txt'

data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y.astype(int), test_size=0.2, random_state=3)
cls = svm.SVC(kernel='linear')
cls.fit(X_train, y_train)
pred = cls.predict(X_test)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred=pred))
print("Precision: ", metrics.precision_score(y_test, y_pred=pred, average='macro'))
print("Recall", metrics.recall_score(y_test, y_pred=pred, average='macro'))
print(metrics.classification_report(y_test, y_pred=pred))
visualize_classifier(cls, X_test, y_test)
```

Рис. 2.16 Код файлу LR_1_task6.py

```
0. (Штучний інтелект) Lab1 (Lab1\venv\Scripts\python.exe - 0. (Ш
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

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

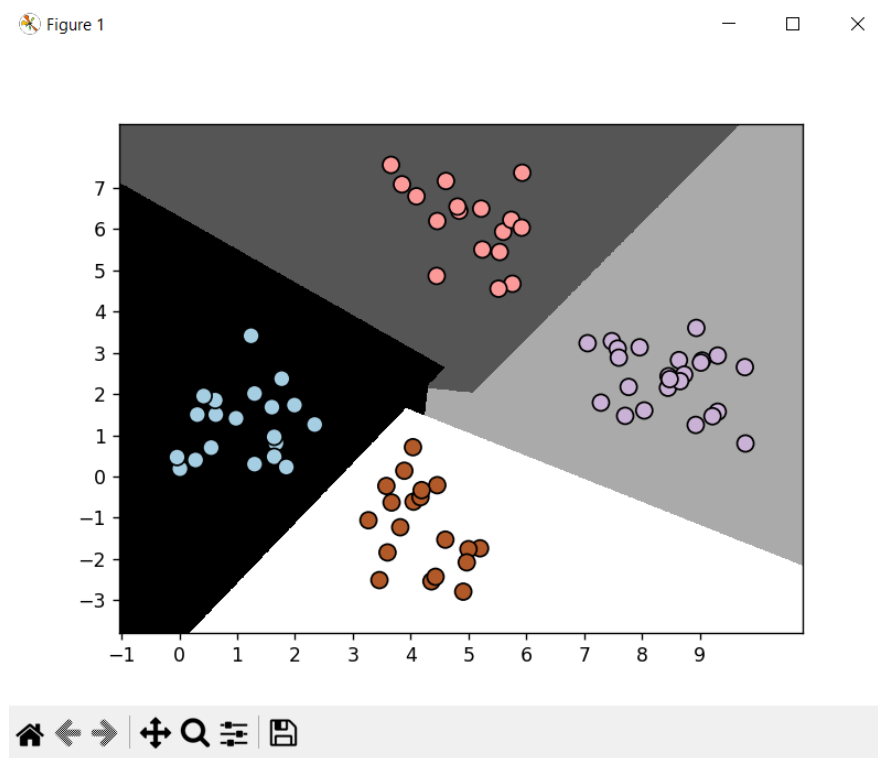


Рис 2.18 Результат файлу LR_1_task6.py

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

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр1	Арк.
		Філіпов В.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		