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

```python
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]])

# Бінаризація даних
data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
print("\n Binarized data:\n", data_binarized)

# Виведення середнього значення та стандартного відхилення
print("\nBEFORE: ")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

# Исключение среднего
data_scaled = preprocessing.scale(input_data)
print("\nAFTER: ")
print("Mean =", data_scaled.mean(axis=0))
print("Std deviation =", data_scaled.std(axis=0))

#Масштабування MinMax
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)

# Нормалізація даних
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)
```

Рис 2.1 - Файл main.py

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]]
```

Рис 2.2 – Результат виконання коду фалу main.py

Завдання 2.1.5

```python
import numpy as np
from sklearn import preprocessing

# Надання позначок вхідних даних
input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow', 'white']

# Створення кодувальника та встановлення відповідності # між мітками та числами
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)

# Виведення відображення
print("\nLabel mapping:")
for i, item in enumerate(encoder.classes_) : print(item, '-->', i)

# перетворення міток за допомогою кодувальника
test_labels = ['green', 'red', 'black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))

# Декодування набору чисел за допомогою декодера

encoded_values = [3, 0, 4, 1]
decoded_list = encoder.inverse_transform(encoded_values)
print("\nEncoded values =", encoded_values)
print("Decoded labels =", list(decoded_list))
```

Рис 2.3 Код файлу LR_1_task_1.py

Результат:

```
sys.path.extend(['C:\\Users\\ankud\\Desktop\\myAi\\lab1'])

Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)]

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']
```

Рис 2.4 Результат файлу LR_1_task_1.py

Завдання 2.2 Попередня обробка нових даних

1.	4.3	-9.9	-3.5	-2.9	4.1	3.3	-2.2	8.8	-6.1	3.9	1.4	2.2	2.2

```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[4.3, -9.9, -3.5], [-2.9, 4.1, 3.3], [-2.2, 8.8, -6.1], [3.9, 1.4, 2.2]])

# Бінаризація даних
data_binarized = preprocessing.Binarizer(threshold=1.8).transform(input_data)
print("\n Binarized data:\n", data_binarized)

# Виведення середнього значення та стандартного відхилення
print("\nBEFORE: ")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

# Исключение среднего
data_scaled = preprocessing.scale(input_data)
print("\nAFTER: ")
print("Mean =", data_scaled.mean(axis=0))
print("Std deviation =", data_scaled.std(axis=0))

#Масштабування MinMax
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)

# Нормалізація даних
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)
```

Рис 2.5. Код файлу LR_1_task_2.py

Результат:

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

BEFORE:
Mean = [ 0.775  1.1   -1.025]
Std deviation = [3.33719568 6.88077031 3.9047247 ]

AFTER:
Mean = [-2.77555756e-17  4.85722573e-17  2.77555756e-17]
Std deviation = [1. 1. 1.]

Min max scaled data:
 [[1.         0.         0.27659574]
 [0.         0.7486631  1.        ]
 [0.09722222 1.         0.        ]
 [0.94444444 0.60427807 0.88297872]]

l1 normalized data:
 [[ 0.24293785 -0.55932203 -0.19774011]
 [-0.2815534   0.39805825  0.32038835]
 [-0.12865497  0.51461988 -0.35672515]
 [ 0.52        0.18666667  0.29333333]]

l2 normalized data:
 [[ 0.37896128 -0.87249225 -0.30845685]
 [-0.4825966   0.68229174  0.54916164]
 [-0.20125974  0.80503895 -0.55803836]
 [ 0.83129388  0.29841319  0.46893501]]
```

Рис 2.6 Результат файлу LR_1_task_2.py

Завдання 2.3. Класифікація логістичною регресією або логістичний класифікатор

```python
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, 4.9]])
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_task_3.py

Результат:

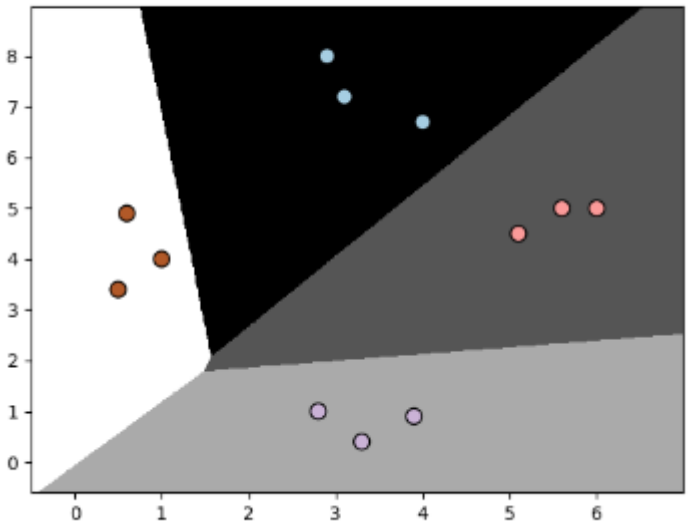

Рис 2.8 Результат файлу LR_1_task_3.py

Завдання 2.4

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from utilities import visualize_classifier

# Вхідний файл, який містить дані
input_file = 'data_multivar_nb.txt'

# Завантаження даних із вхідного файлу
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Створення наївного байєсовського класифікатора
classifier = GaussianNB()

# Тренування класифікатора
classifier.fit(X, y)

# Прогнозування значень для тренувальних даних
y_pred = classifier.predict(X)

# Обчислення якості класифікатора
accuracy = 100.0 * (y == y_pred).sum() / X.shape[0]
print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")

# Візуалізація результатів роботи класифікатора
visualize_classifier(classifier, X, y)

# Розбивка даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=3)
classifier_new = GaussianNB()
classifier_new.fit(X_train, y_train)
y_test_pred = classifier_new.predict(X_test)
```

```python
# Тренування класифікатора
classifier.fit(X, y)

# Прогнозування значень для тренувальних даних
y_pred = classifier.predict(X)

# Обчислення якості класифікатора
accuracy = 100.0 * (y == y_pred).sum() / X.shape[0]
print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")

# Візуалізація результатів роботи класифікатора
visualize_classifier(classifier, X, y)

# Розбивка
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=3)
classifier_new = GaussianNB()
classifier_new.fit(X_train, y_train)
y_test_pred = classifier_new.predict(X_test)

# Обчислення якості класифікатора
accuracy = 100.0 * (y_test == y_test_pred).sum() / X_test.shape[0]
print("Accuracy of the new classifier =", round(accuracy, 2), "%")
# Візуалізація роботи класифікатора
visualize_classifier(classifier_new, X_test, y_test)

num_folds = 3
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier,X, y, scoring='f1_weighted', cv=num_folds)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
```

Fetching Documentation...

Рис 2.9 Код файлу LR_1_task_4.py

Результат:

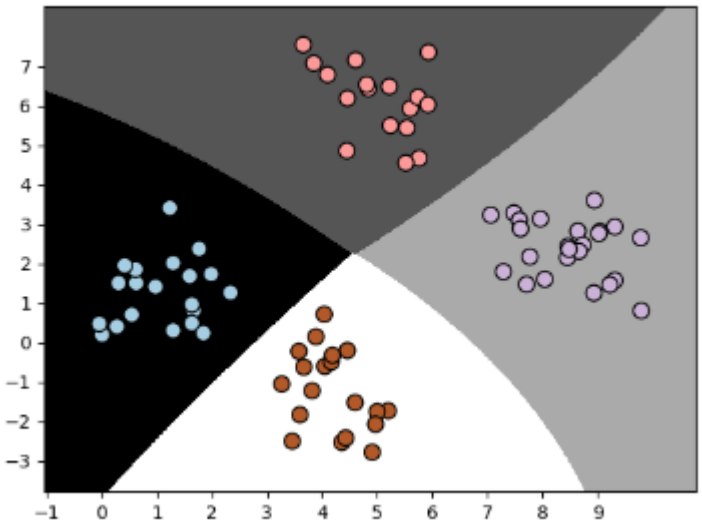

Рис 2.10 Результат файлу LR_1_task4.py

```python
# Розбивка даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
classifier_new = GaussianNB()
classifier_new.fit(X_train, y_train)
y_test_pred = classifier_new.predict(X_test)

# Обчислення якості класифікатора
accuracy = 100.0 * (y_test == y_test_pred).sum() / X_test.shape[0]
print("Accuracy of the new classifier =", round(accuracy, 2), "%")
# Візуалізація роботи класифікатора
visualize_classifier(classifier_new, X_test, y_test)

num_folds = 3
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=num_folds)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
```

Рис 2.11 Код файлу LR_1_task4.py

```
sys.path.extend(['C:\\Users\\ankud\\Desktop\\myAi\\lab1'])

Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)]
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 2.12 Результат файлу LR_1_task4.py

Завдання 2.5.Вивчити метрики якості класифікації

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

df = pd.read_csv('data_metrics.csv')
df.head()
thresh = 0.5
df['predicted_RF'] = (df.model_RF >= 0.5).astype('int')
df['predicted_LR'] = (df.model_LR >= 0.5).astype('int')
df.head()

print(confusion_matrix(df.actual_label.values, df.predicted_RF.values))


2 usages
def find_TP(y_true, y_pred):
    # counts the number of true positives (y_true = 1, y_pred = 1)
    return sum((y_true == 1) & (y_pred == 1))


2 usages
def find_FN(y_true, y_pred):
    # counts the number of false negatives (y_true = 1, y_pred = 0)
    return sum((y_true == 1) & (y_pred == 0))


2 usages
def find_FP(y_true, y_pred):
    # counts the number of false positives (y_true = 0, y_pred = 1)
    return sum((y_true == 0) & (y_pred == 1))

```

```
3 usages
def ankudevich_confusion_matrix(y_true, y_pred):
    TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
    return np.array([[TN, FP], [FN, TP]])


ankudevich_confusion_matrix(df.actual_label.values, df.predicted_RF.values)

assert np.array_equal(ankudevich_confusion_matrix(df.actual_label.values, df.predicted_RF.values),
                      confusion_matrix(df.actual_label.values,
                                       df.predicted_RF.values)), 'my_confusion_matrix() is not correct for RF'
assert np.array_equal(ankudevich_confusion_matrix(df.actual_label.values, df.predicted_LR.values),
                      confusion_matrix(df.actual_label.values,
                                       df.predicted_LR.values)), 'my_confusion_matrix() is not correct for LR'

print(accuracy_score(df.actual_label.values, df.predicted_RF.values))
```

```
5 usages
def ankudevich_accuracy_score(y_true, y_pred):  # calculates the fraction of samples
    TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
    return (TP + TN) / (TP + TN + FP + FN)


assert ankudevich_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'
assert ankudevich_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'
print('Accuracy RF:%.3f' % (ankudevich_accuracy_score(df.actual_label.values, df.predicted_RF.values)))

print(recall_score(df.actual_label.values, df.predicted_RF.values))
```

```
7 usages
def ankudevich_recall_score(y_true, y_pred):
    # calculates the fraction of positive samples predicted correctly
    TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
    return TP / (TP + FN)


assert ankudevich_recall_score(df.actual_label.values, df.predicted_RF.values) == recall_score(df.actual_label.values,
                                                                                               df.predicted_RF.values), 'ankudevich_accuracy_score failed on RF'
assert ankudevich_recall_score(df.actual_label.values, df.predicted_LR.values) == recall_score(df.actual_label.values,
                                                                                               df.predicted_LR.values), 'ankudevich_accuracy_score failed on LR'

print('Recall RF: %.3f' % (ankudevich_recall_score(df.actual_label.values, df.predicted_RF.values)))
print('Recall LR: %.3f' % (ankudevich_recall_score(df.actual_label.values, df.predicted_LR.values)))

precision_score(df.actual_label.values, df.predicted_RF.values)


7 usages
def ankudevich_precision_score(y_true, y_pred):
    # calculates the fraction of predicted positives samples that are actually positive
    TP, FN, FP, TN = find_conf_matrix_values(y_true, y_pred)
    return TP / (TP + FP)


assert ankudevich_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'
assert ankudevich_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'

print('Precision RF: %.3f' % (ankudevich_precision_score(df.actual_label.values, df.predicted_RF.values)))
print('Precision LR: %.3f' % (ankudevich_precision_score(df.actual_label.values, df.predicted_LR.values)))

f1_score(df.actual_label.values, df.predicted_RF.values)
```

```python
def ankudevich_f1_score(y_true, y_pred):  # calculates the F1 score
    recall = ankudevich_recall_score(y_true, y_pred)
    precision = ankudevich_precision_score(y_true, y_pred)
    return 2 * (precision * recall) / (precision + recall)


assert ankudevich_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'
assert ankudevich_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'

print('F1 RF: %.3f' % (ankudevich_f1_score(df.actual_label.values, df.predicted_RF.values)))
print('F1 LR: %.3f' % (ankudevich_f1_score(df.actual_label.values, df.predicted_LR.values)))
print('scores with threshold = 0.5')

print('Accuracy RF: % .3f' % (ankudevich_accuracy_score(df.actual_label.values, df.predicted_RF.values)))
print('Recall RF: %.3f' % (ankudevich_recall_score(df.actual_label.values, df.predicted_RF.values)))
print('Precision RF: % .3f' % (ankudevich_precision_score(df.actual_label.values, df.predicted_RF.values)))
print('F1 RF: %.3f' % (ankudevich_f1_score(df.actual_label.values, df.predicted_RF.values)))
print('')

threshold = 0.75

print(f'Scores with threshold = {threshold}')
print('Accuracy RF: % .3f' % (ankudevich_accuracy_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
print('Recall RF: %.3f' % (ankudevich_recall_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
print('Precision RF: %.3f' % (ankudevich_precision_score(df.actual_label.values, (df.model_RF >= threshold).astype('int').values)))
print('F1 RF: %.3f' % (ankudevich_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()
```

```python
auc_RF = roc_auc_score(df.actual_label.values, df.model_RF.values)
auc_LR = roc_auc_score(df.actual_label.values, df.model_LR.values)
print('AUC RF:%.3f' % auc_RF)
print('AUC LR:%.3f' % auc_LR)

plt.plot(fpr_RF, tpr_RF, 'r-', label='RF AUC: %.3f' % auc_RF)
plt.plot(fpr_LR, tpr_LR, 'b-', label='LR AUC: %.3f' % auc_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_task_5.py

Результат:

```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)]
[[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 2.14 Результат файлу LR_1_task5.py

F1 міра зменшується в результаті збільшення порогу.

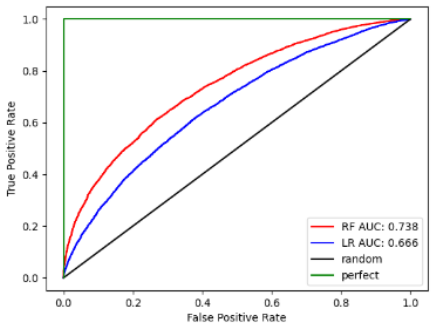

Рис 2.15. ROC – крива.

Завдання 2.6

```python
import numpy as np
from sklearn import svm
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix

# Завантаження даних з файлу
data = np.loadtxt('data_multivar_nb.txt', delimiter=',')

# Розділення вхідних ознак та міток класів
X = data[:, :-1]
y = data[:, -1]

# Розділення даних на тренувальний та тестувальний набори
# Ви можете використовувати свою власну стратегію розділення, наприклад, train_test_split з scikit-learn
X_train = X[:80]
y_train = y[:80]
X_test = X[80:]
y_test = y[80:]

# Навчання SVM
svm_model = svm.SVC()
svm_model.fit(X_train, y_train)

# Навчання наївного байєсівського класифікатора
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)

# Класифікація за допомогою SVM
svm_predictions = svm_model.predict(X_test)

# Класифікація за допомогою наївного байєсівського класифікатора
nb_predictions = nb_model.predict(X_test)

# Оцінка якості класифікації для SVM
svm_report = classification_report(y_test, svm_predictions)
svm_confusion_matrix = confusion_matrix(y_test, svm_predictions)
```

```python
svm_report = classification_report(y_test, svm_predictions)
svm_confusion_matrix = confusion_matrix(y_test, svm_predictions)

# Оцінка якості класифікації для наївного байєсівського класифікатора
nb_report = classification_report(y_test, nb_predictions)
nb_confusion_matrix = confusion_matrix(y_test, nb_predictions)

# Виведення результатів
print("SVM Classification Report:")
print(svm_report)
print("SVM Confusion Matrix:")
print(svm_confusion_matrix)

print("Naive Bayes Classification Report:")
print(nb_report)
print("Naive Bayes Confusion Matrix:")
print(nb_confusion_matrix)
```

Рис. 2.16 Код файлу LR_1_task6.py

```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)]
SVM Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        80
         1.0       1.00      1.00      1.00        80
         2.0       1.00      1.00      1.00        80
         3.0       1.00      1.00      1.00        80

    accuracy                           1.00       320
   macro avg       1.00      1.00      1.00       320
weighted avg       1.00      1.00      1.00       320

SVM Confusion Matrix:
[[80  0  0  0]
 [ 0 80  0  0]
 [ 0  0 80  0]
 [ 0  0  0 80]]
Naive Bayes Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        80
         1.0       1.00      0.99      0.99        80
         2.0       0.99      1.00      0.99        80
         3.0       1.00      1.00      1.00        80

    accuracy                           1.00       320
   macro avg       1.00      1.00      1.00       320
weighted avg       1.00      1.00      1.00       320

Naive Bayes Confusion Matrix:
[[80  0  0  0]
 [ 0 79  1  0]
 [ 0  0 80  0]
 [ 0  0  0 80]]
```

Рис 2.17 Результат файлу LR_1_task_6.py

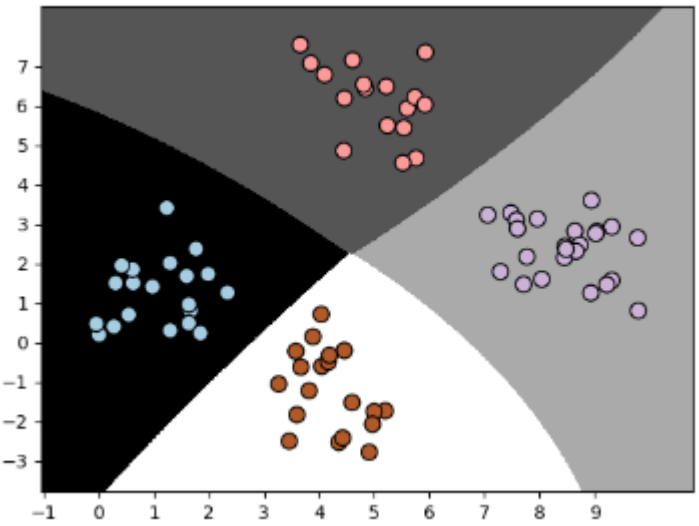

Рис 2.18 Результат файлу LR_1_task6.py

Висновок: Під час виконання лабораторної роботи я отримав досвід використання спеціалізованих бібліотек та мови програмування Python для обробки та класифікації даних. Я ознайомився з бібліотеками scikit-learn та numpy, які надають потужні інструменти для роботи з машинним навчанням.