

Лабораторна робота №1

ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ

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

Хід роботи

Завдання №1:

Попередня обробка даних:

```
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]])

# Binarize data
data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
print("\nBinarized data:\n", data_binarized)

# Print mean and standard deviation
print("\nBEFORE:")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

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

# Min max scaling
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)

# Normalize data
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)
```

Результат:

					Житомирська політехніка.24.121.14.000 – Лр1						
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи			Літ.	Арк.	Аркушів	
Розроб.		Паламарчук В.В.									
Перевір.		Голенко М.Ю.							1	16	
Керівник								ФІКТ Гр.ІПЗ-21-3[2]			
Н. контр.											
Зав. каф.											

```

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. 1. Результати різних методів обробки даних

Існування двох видів нормалізації даних - L1 та L2 - є основними. У випадку L1-нормалізації, сума абсолютних значень елементів рядка визначає одиницю, тоді як у L2-нормалізації це сума квадратів цих значень. L1-нормалізація має перевагу у більшій надійності порівняно з L2, адже вона менш схильна до впливу викидів. Навпаки, L2-нормалізація використовується там, де викиди мають значний вплив.

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

Завдання №2:

Кодування міток:

```
import numpy as np
from sklearn import preprocessing

# Sample input labels
input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow', 'white']

# Create label encoder and fit the labels
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)

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

# Encode a set of labels using the encoder
test_labels = ['green', 'red', 'black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))

# Decode a set of values using the encoder
encoded_values = [3, 0, 4, 1]
decoded_list = encoder.inverse_transform(encoded_values)
print("\nEncoded values =", encoded_values)
print("Decoded labels =", list(decoded_list))
```

Результат:

```
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']
```

Рис. 2. Результати кодування міток

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		3

Завдання №3:

Змінімо дані відповідно до варіанту(№14) та здійснимо обробку даних різними методами:

```
import numpy as np
from sklearn import preprocessing

input_data = np.array([[ -1.3,  3.9,  6.2],
                        [ -4.9,  2.2, -4.3],
                        [ -2.2,  6.5,  4.1],
                        [ -5.2, -3.4, -5.2]])

# Binarize data
data_binarized = preprocessing.Binarizer(threshold=2.2).transform(input_data)
print("\nBinarized data:\n", data_binarized)

# Print mean and standard deviation
print("\nBEFORE:")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

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

# Min max scaling
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)

# Normalize data
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)
```

Результат:

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

BEFORE:
Mean = [-3.4  2.3  0.2]
Std deviation = [1.68374582  3.62973828  5.01547605]

AFTER:
Mean = [1.66533454e-16  5.55111512e-17  0.00000000e+00]
Std deviation = [1. 1. 1.]

Min max scaled data:
[[1.          0.73737374  1.          ]
 [0.07692308  0.56565657  0.07894737]
 [0.76923077  1.          0.81578947]
 [0.          0.          0.          ]]

L1 normalized data:
[[-0.11403509  0.34210526  0.54385965]
 [-0.42982456  0.19298246 -0.37719298]
 [-0.171875    0.5078125   0.3203125 ]
 [-0.37681159 -0.24637681 -0.37681159]]

L2 normalized data:
[[-0.17475265  0.52425796  0.83343572]
 [-0.71216718  0.31974853 -0.62496303]
 [-0.2752151   0.81313551  0.51290086]
 [-0.64182859 -0.41965715 -0.64182859]]

```

Рис. 3. Результати різних методів обробки даних відповідно до варіанту

Завдання №4:

Логістична регресія:

```

import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt

from utils import visualize_classifier

# Define sample input data
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])

# Create the logistic regression classifier
classifier = linear_model.LogisticRegression(solver='liblinear', C=1)
# classifier = linear_model.LogisticRegression(solver='liblinear', C=100)

# Train the classifier
classifier.fit(X, y)

# Visualize the performance of the classifier
visualize_classifier(classifier, X, y)

```

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат:

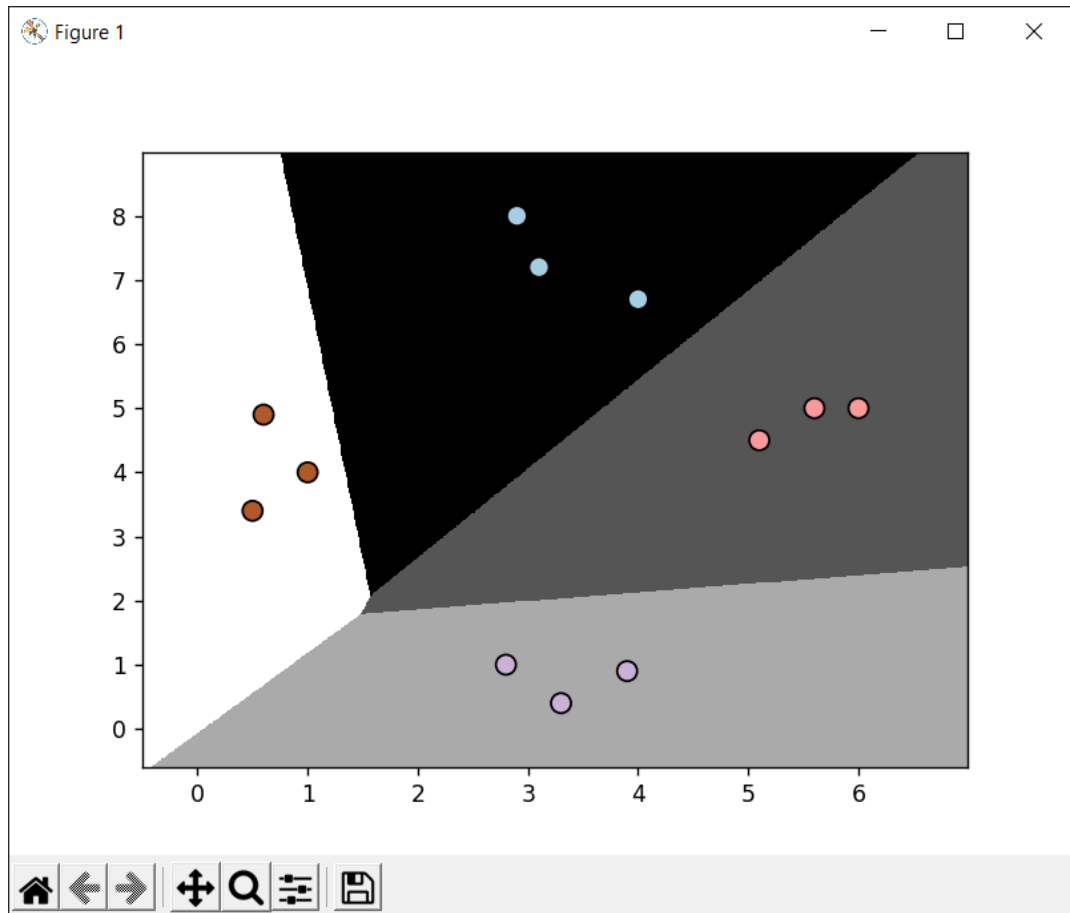


Рис. 4. Логістична регресія

Завдання №5:

Класифікація найвним байєсовським класифікатором:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB

from utils import visualize_classifier

# Input file containing data
input_file = 'data_multivar_nb.txt'

# Load data from input file
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Create Naive Bayes classifier
classifier = GaussianNB()

# Train the classifier
classifier.fit(X, y)

# Predict the values for training data
y_pred = classifier.predict(X)

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

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Visualize the performance of the classifier
visualize_classifier(classifier, X, y)

#####
# Cross validation
```

Результат:

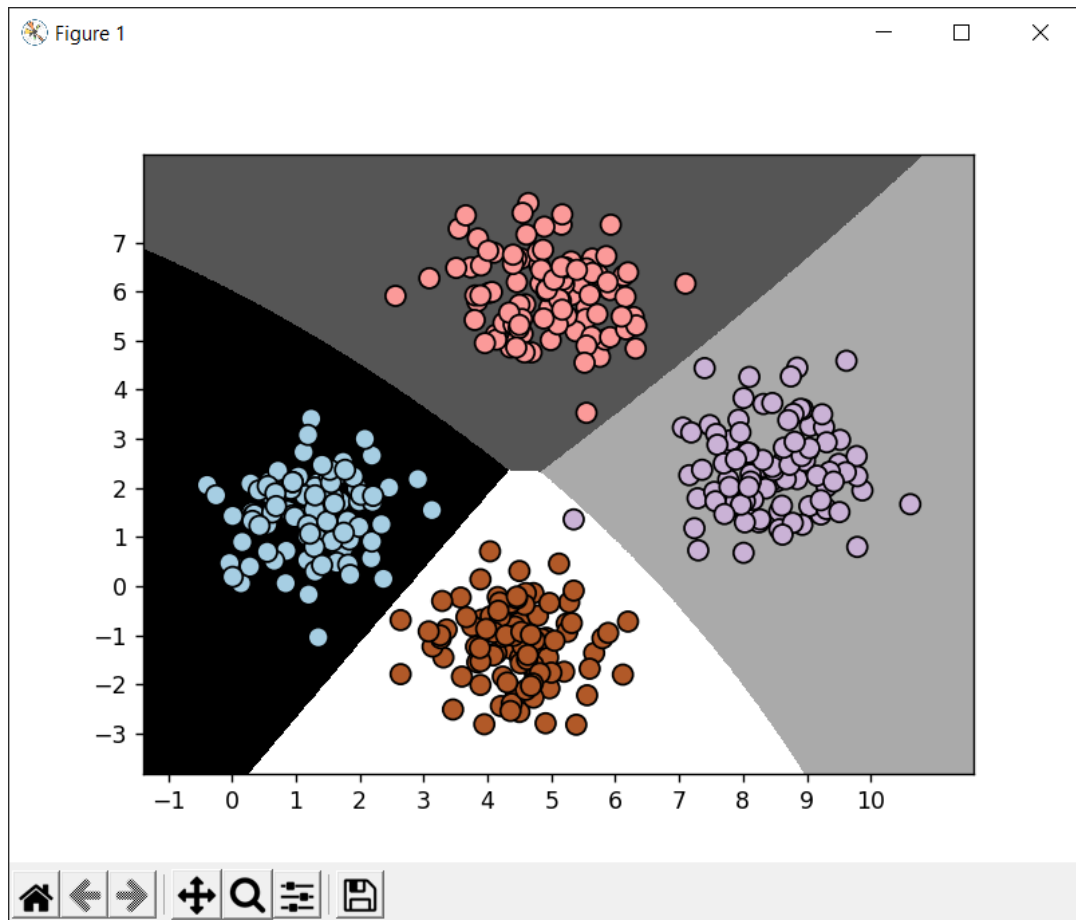


Рис. 5. Класифікація наївним байєсовським класифікатором

Точність методу 99,75%

Завдання №6:

Розбиття тестових даних:

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

from utils import visualize_classifier

# Input file containing data
input_file = 'data_multivar_nb.txt'

# Load data from input file
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Create Naive Bayes classifier
classifier = GaussianNB()

# Train the classifier
classifier.fit(X, y)
```

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# Predict the values for training data
y_pred = classifier.predict(X)

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

# Visualize the performance of the classifier
visualize_classifier(classifier, X, y)

#####
# Cross validation

# Split data into training and test data
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)

# compute accuracy of the classifier
accuracy = 100.0 * (y_test == y_test_pred).sum() / X_test.shape[0]
print("Accuracy of the new classifier =", round(accuracy, 2), "%")

# Visualize the performance of the classifier
visualize_classifier(classifier_new, X_test, y_test)

#####
# Scoring functions

num_folds = 3
accuracy_values = train_test_split.cross_val_score(classifier,
X, y, scoring='accuracy', cv=num_folds)
print("Accuracy: " + str(round(100*accuracy_values.mean(), 2)) + "%")

precision_values = train_test_split.cross_val_score(classifier,
X, y, scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100*precision_values.mean(), 2)) + "%")

recall_values = train_test_split.cross_val_score(classifier,
X, y, scoring='recall_weighted', cv=num_folds)
print("Recall: " + str(round(100*recall_values.mean(), 2)) + "%")

f1_values = train_test_split.cross_val_score(classifier,
X, y, scoring='f1_weighted', cv=num_folds)
print("F1: " + str(round(100*f1_values.mean(), 2)) + "%")

```

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат першого прогону:

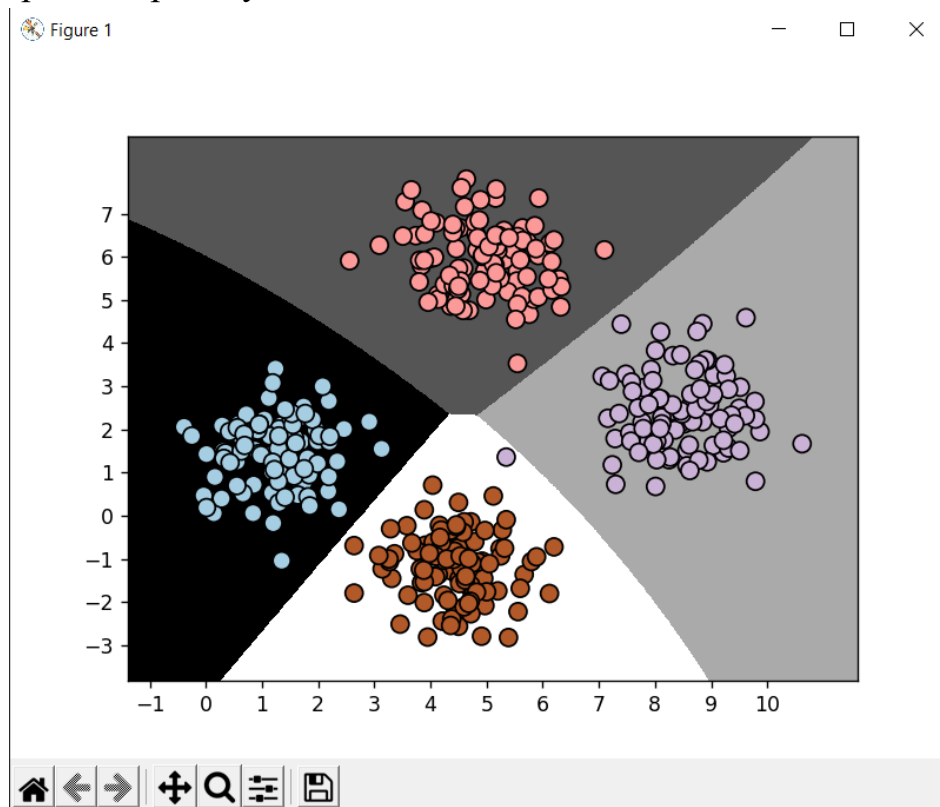


Рис. 6. Результат першого прогону класифікації наївним байєсовським класифікатором

Результат другого прогону:

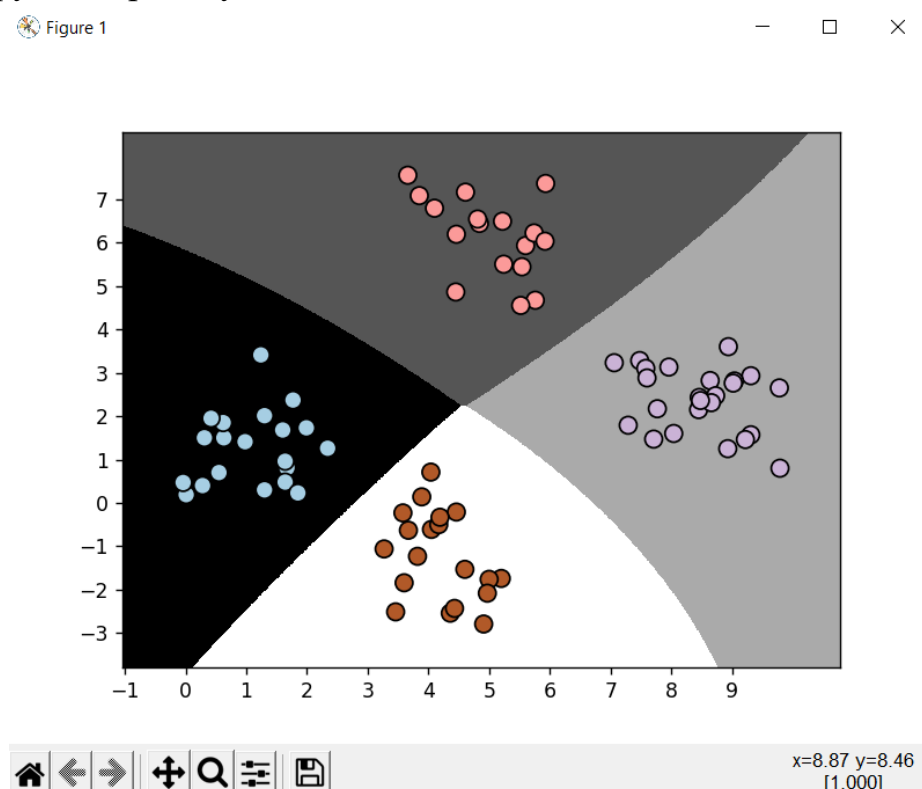


Рис. 7. Результат другого прогону класифікації наївним байєсовським класифікатором

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

На першому зображенні ми виявляємо некоректне категоризування, в той час як на другому зображенні такої помилки не спостерігається. Це підкріплюється статистичними даними: точність першого аналізу складає 99,75%, тоді як у другому випадку вона дорівнює 100%. Таким чином, результати другого аналізу виявилися більш ефективними порівняно з першим.

Завдання №6:

Власні функції для перевірки confusion_matrix:

```
import pandas as pd

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

def hordeiev_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))

def hordeiev_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))

def hordeiev_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))

def hordeiev_find_TN(y_true, y_pred):
    # counts the number of true negatives (y_true = 0, y_pred = 0)
    return sum((y_true == 0) & (y_pred == 0))

print('TP:', hordeiev_find_TP(df.actual_label.values, df.predicted_RF.values))
print('FN:', hordeiev_find_FN(df.actual_label.values, df.predicted_RF.values))
print('FP:', hordeiev_find_FP(df.actual_label.values, df.predicted_RF.values))
print('TN:', hordeiev_find_TN(df.actual_label.values, df.predicted_RF.values))
```

Власна функція, яка дублює accuracy_score:

```
def hordeiev_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)

print('Accuracy RF: %.3f' % (hordeiev_accuracy_score(df.actual_label.values,
df.predicted_RF.values)))
print('Accuracy LR: %.3f' % (hordeiev_accuracy_score(df.actual_label.values,
df.predicted_LR.values)))
```

Результати збіглись: RF - 0.67, LR - 0.62.

Власна функція, яка дублює recall_score:

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

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Результати збіглись: RF - 0.641, LF - 0.616.

Власна функція, яка дублює precision_score:

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

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

Результати збіглись: RF - 0.681, LF - 0.616.

Власна функція, яка дублює f1_score:

```
def hordeiev_f1_score(y_true, y_pred):
    # calculates the fraction of samples
    recall = hordeiev_recall_score(y_true, y_pred)
    precision = hordeiev_precision_score(y_true, y_pred)
    return 2 * (precision * recall) / (precision + recall)

print('F1 RF: %.3f'%(hordeiev_f1_score(df.actual_label.values,
df.predicted_RF.values)))
print('F1 LR: %.3f'%(hordeiev_accuracy_score(df.actual_label.values,
df.predicted_LR.values)))
```

Результати збіглись: RF - 0.66, LF - 0.616.

Результати порогів 0.5 та 0.25:

```
scores with threshold = 0.5
Accuracy RF: 0.671
Recall RF: 0.641
Precision RF: 0.681
F1 RF: 0.660

scores with threshold = 0.25
Accuracy RF: 0.502
Recall RF: 1.000
Precision RF: 0.501
F1 RF: 0.668
```

Рис. 8. Результат метрик з різними порогам

Отже, з порогом 0.5 краще такі метрики: accuracy та precision; а з порогом 0.25 – recall та f1.

Переглянемо графіки двох моделей:

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

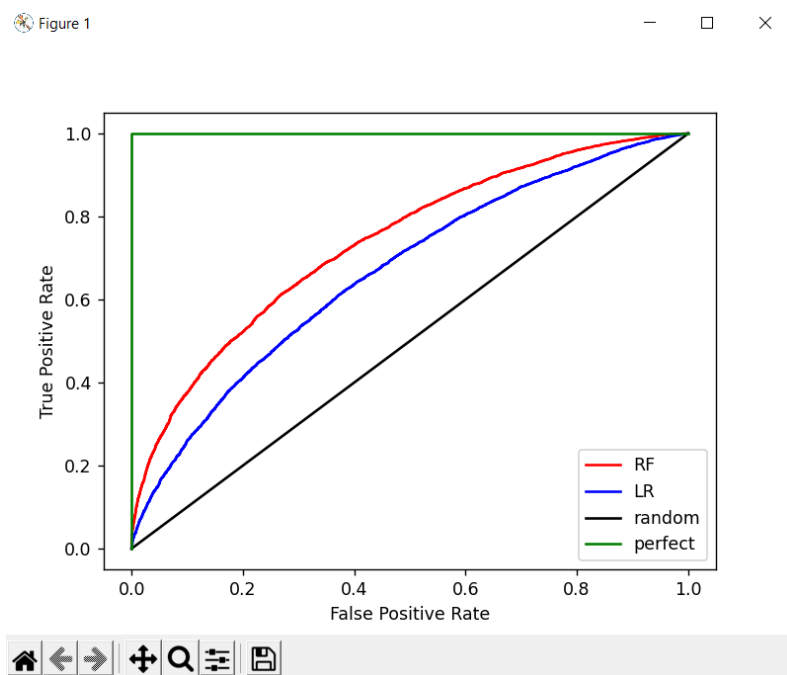


Рис. 9. ROC-криві двох моделей

На малюнку 9 спостерігаємо, що близькість до зеленої лінії означає кращі результати, тоді як наближення до чорної лінії свідчить про гірші показники. Ураховуючи, що траєкторія моделі RF ближча до зеленої лінії, можна зробити висновок, що модель RF демонструє кращі результати.

Завдання №7:

Розробимо програму класифікації даних за допомогою машини опорних векторів та наївного байєсівського класифікатора. Також порівнюємо їх показники якості:

```
Accuracy of the NB classifier = 100.0 %
Accuracy: 99.75%
Precision: 99.76%
Recall: 99.75%
F1: 99.75%

Accuracy of the SVM classifier = 100.0 %
Accuracy: 99.75%
Precision: 99.76%
Recall: 99.75%
F1: 99.75%
```

Рис. 9. Показники якості двох методів

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Бачимо, що показники якості двох методів збіглись для заданих даних. Єдине, що відрізняється – графіки. Переглянемо їх:

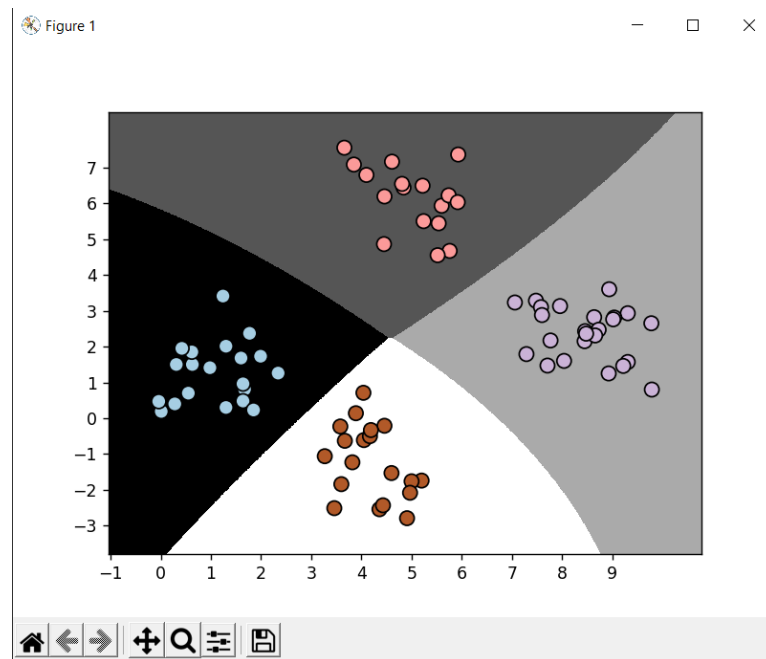


Рис. 10. Графік класифікатора найвішого байесівського класифікатора

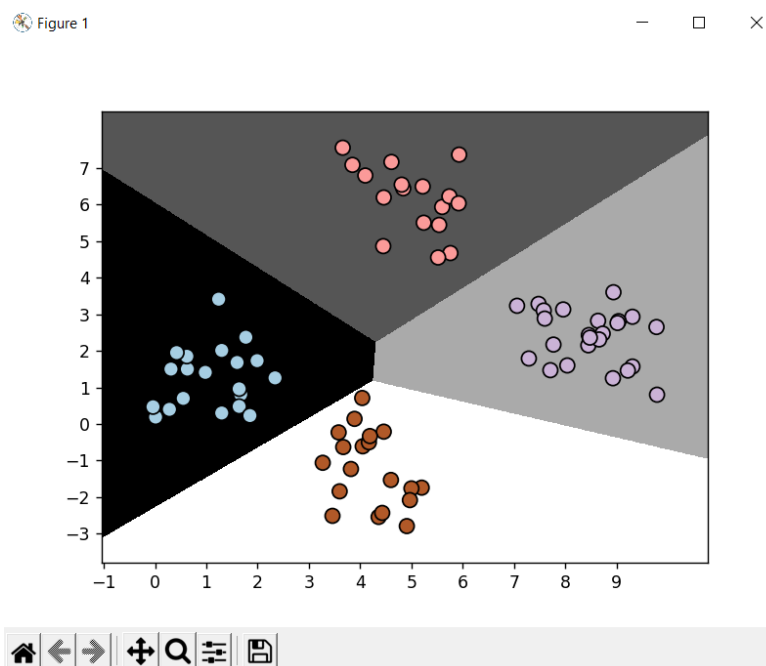


Рис. 10. Графік класифікатора за допомогою машини опорних векторів

Висновок: на цій лабораторній роботі я, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідив попередню обробку та класифікацію даних.

		Паламарчук В.В.			Житомирська політехніка.24.121.14.000 – Лр1	Арк.
		Голенко М.Ю.				13
Змн.	Арк.	№ докум.	Підпис	Дата		