

ЛАБОРАТОРНА РОБОТА № 1

ПОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ

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

Хід роботи:

Завдання 1: Попередня обробка даних

Лістинг програми:

```
# Diachenko Viktor, ZPI-18, Lab1, Task 1

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))

# Вилучення середнього
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)
```

					ДУ «Житомирська політехніка».20.121.03.000 – Лр1			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Дяченко В. В.			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Пулеко І. В.						Аркушів
Керівник								1
Н. контр.								07
Зав. каф.							ФІКТ Гр. ЗПІ-18	

```
# 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)
```

Результат виконання програми:

```
albedych@MiWiFi-R4A-srv:~/Документи/AI/l1
python LR_1_task_1.py

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 попередньої обробки даних

Нормалізація типу L1 дозволяє запобігти сильному впливу викидів. Вона є більш надійною, у порівнянні з нормалізацією типу L2.

Завдання 2. Попередня обробка нових даних

Варіант №3:

3.	1.3	-3.9	6.5	-4.9	-2.2	1.3	2.2	6.5	-6.1	-5.4	-1.4	2.2	1.1
----	-----	------	-----	------	------	-----	-----	-----	------	------	------	-----	-----

Лістинг програми:

```
# Diachenko Viktor, ZPI-18, Lab1, Task 2

import numpy as np
from sklearn import preprocessing

input_data = np.array([[2.3, -1.6, 6.1], [-1.2, 4.3, 3.2], [5.5, -6.1, -4.4], [1.4, -1.2, 2.1]]) # варіант 3

# 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))

# Вилучення середнього
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)
```

Результат виконання програми:

		Дяченко В. В.			ДУ «Житомирська політехніка».20.121.03.000 – Лр1	Арк.
		Пулеко І. В.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

albedych@MiWiFi-R4A-srv:~/Документи/AI/l1
python LR_1_task_2.py

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

BEFORE:
Mean = [ 2.  -1.15  1.75]
Std deviation = [2.394786  3.68815672  3.83959633]

AFTER:
Mean = [-4.16333634e-17 -3.12250226e-17 -2.77555756e-17]
Std deviation = [1. 1. 1.]

Min max scaled data:
[[0.52238806 0.43269231 1.         ]
 [0.         1.         0.72380952]
 [1.         0.         0.         ]
 [0.3880597  0.47115385 0.61904762]]

l1 normalized data:
[[ 0.23      -0.16      0.61      ]
 [-0.13793103  0.49425287  0.36781609]
 [ 0.34375   -0.38125   -0.275    ]
 [ 0.29787234 -0.25531915  0.44680851]]

l2 normalized data:
[[ 0.34263541 -0.23835507  0.90872869]
 [-0.2184709  0.78285405  0.58258906]
 [ 0.59027284 -0.65466624 -0.47221827]
 [ 0.50095939 -0.42939376  0.75143908]]

```

Рис. 2. Результат попередньої обробки нових даних

Завдання 3: Класифікація логістичною регресією або логістичний класифікатор

Лістинг програми:

```

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, 2, 3, 3, 3])

# Створення логістичного класифікатора
classifier = linear_model.LogisticRegression(solver='liblinear', C=1)

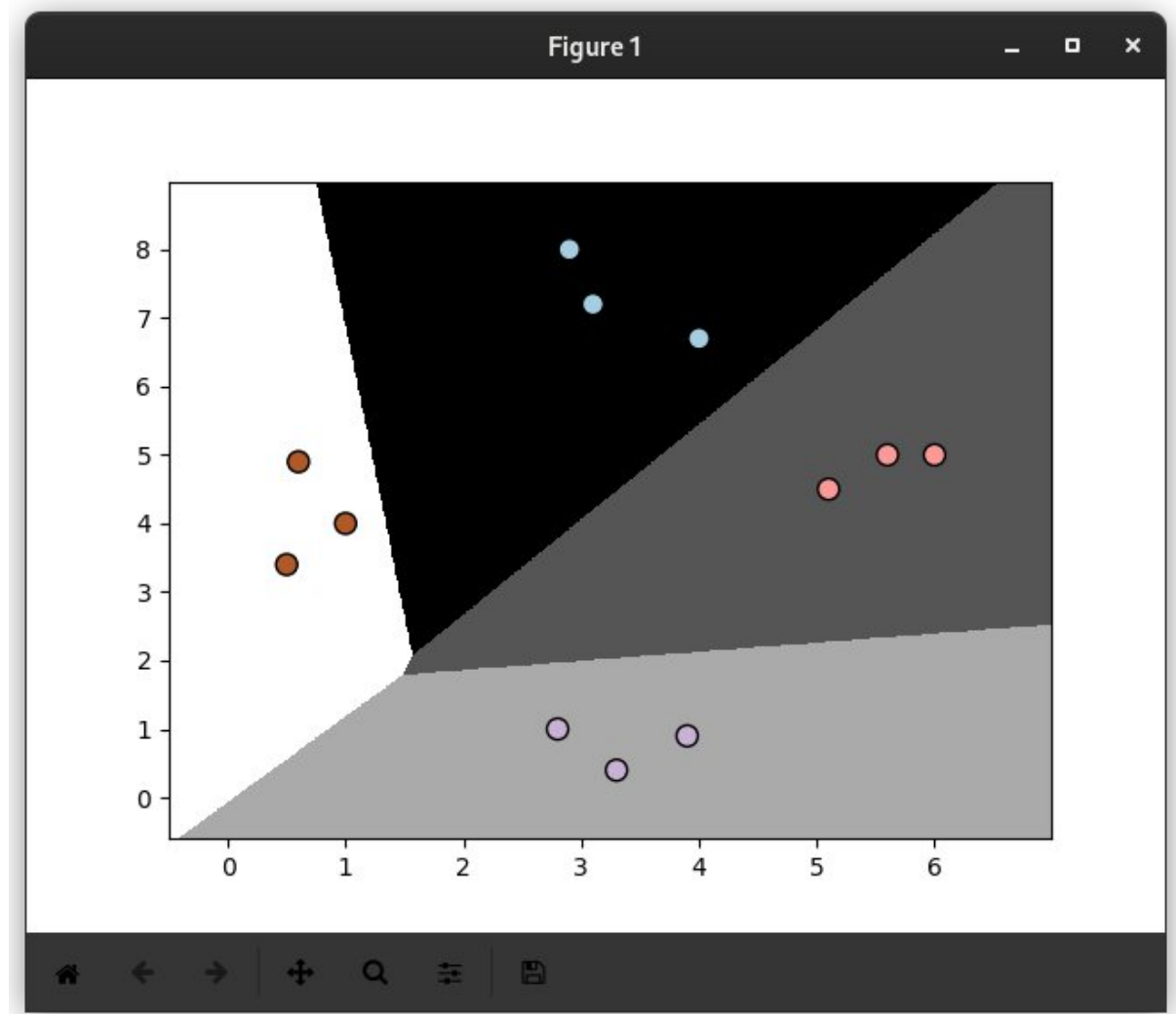
```

		Дяченко В. В.			ДУ «Житомирська політехніка». 20.121.03.000 – Лр1	Арк.
		Пулеко І. В.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

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

visualize_classifier(classifier, X, y)
```

Результат виконання програми:



Завдання 4: Класифікація наївним байєсовським класифікатором

Лістинг програми:

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

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

# Завантаження даних із вхідного файлу
```

		Дяченко В. В.			ДУ «Житомирська політехніка».20.121.03.000 – Лр1	Арк.
		Пулеко І. В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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.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 = 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)) + "%")

```

		Дяченко В. В.			ДУ «Житомирська політехніка».20.121.03.000 – Лр1	Арк.
		Пулеко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

Результат виконання програми:

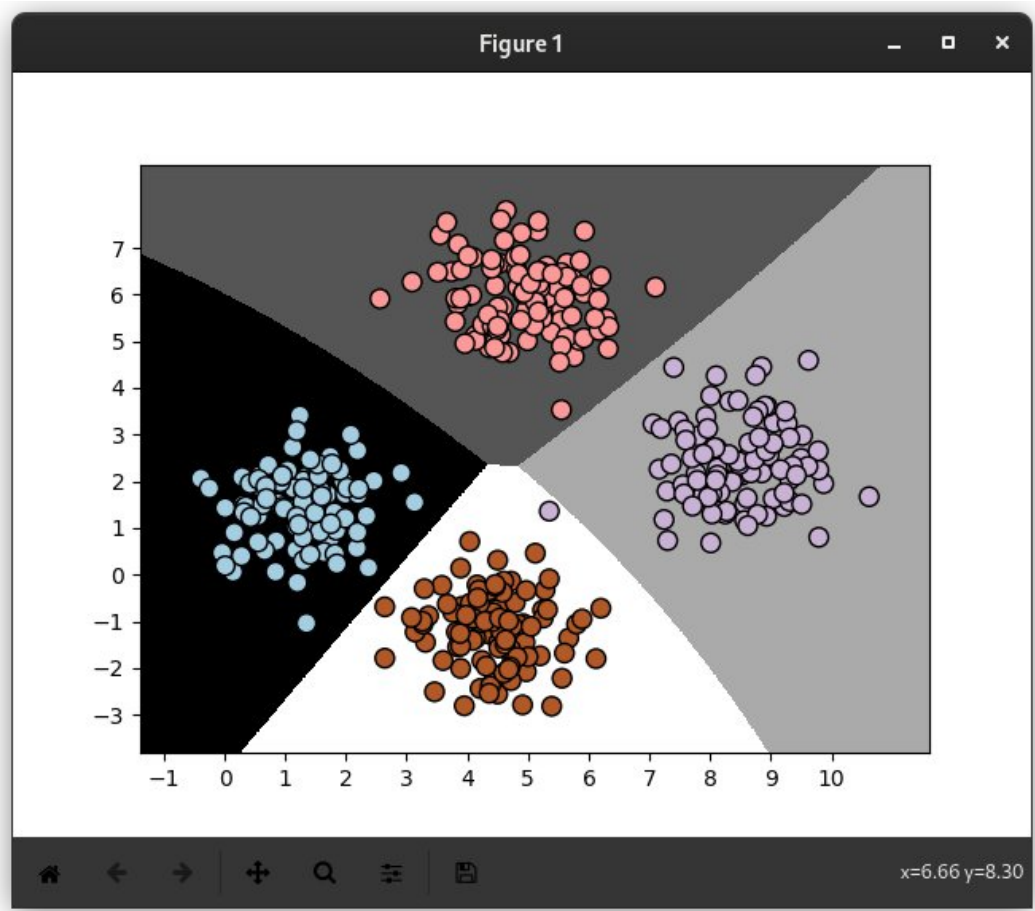


Рис. 4 Візуалізація результату класифікації наївним баєсовським класифікатором

Посилання на GitHub: https://github.com/diachenkovv/AI_python

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

		Дяченко В. В.			ДУ «Житомирська політехніка».20.121.03.000 – Лр1	Арк.
		Пулеко І. В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		