Лабораторне заняття 1

Попередня обробка та контрольована класифікація даних

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

Хід роботи

1. Завдання 2.1.1

Код програми

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]])

# 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 MinМax

data\_scaler\_minmax = preprocessing.MinMaxScaler(feature\_range=(0, 1))

data\_scaled\_minmax = data\_scaler\_minmax.fit\_transform(input\_data)

print("\nМin 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)

data\_binarized = preprocessing.Binarizer(threshold=2.1).transform(input\_data)

print("\n Binarized data:\n", data\_binarized)

Результат виконання:

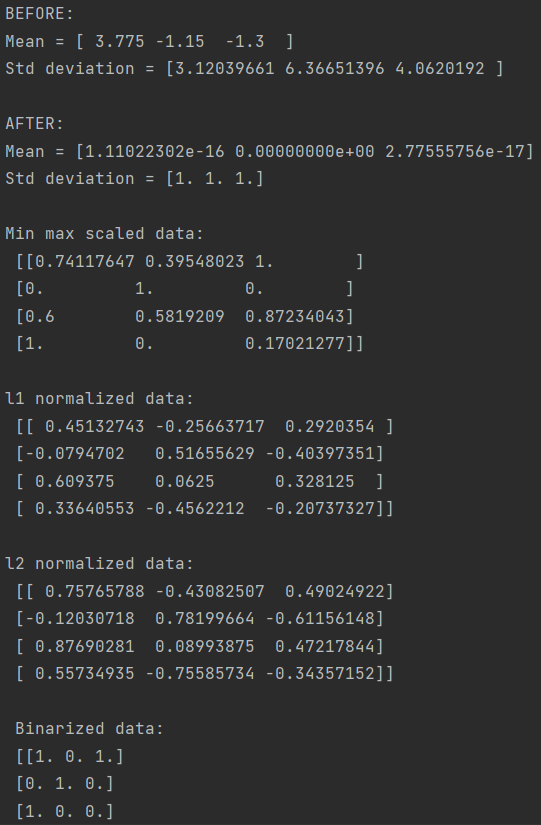


Рис. 1.1 – Результат виконання програми

Зробіть висновок чим відрізняються L1-нормалізація від L2-нормалізацієї.

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 значень.

1. Завдання 2.1.2

Код програми

import numpy as np

from sklearn import preprocessing

# Надання позначок вхідних даних

input\_labels = ['red', 'black', 'red', 'green',

'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))

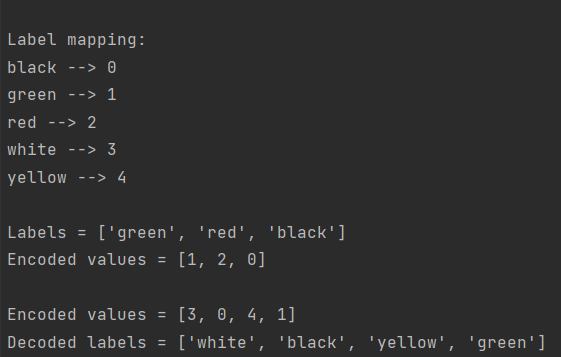


Рис. 1.2 – Результат виконання програми

1. Завдання 2.2. Попередня обробка нових даних

Код програми

import numpy as np

from sklearn import preprocessing

input\_data = np.array([[4.6, 9.9, -3.5],

[-2.9, 4.1, 3.3],

[-2.2, 8.8, -6.1],

[3.9, 1.4, 2.2]])

# 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 MinМax

data\_scaler\_minmax = preprocessing.MinMaxScaler(feature\_range=(0, 1))

data\_scaled\_minmax = data\_scaler\_minmax.fit\_transform(input\_data)

print("\nМin 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)

data\_binarized = preprocessing.Binarizer(threshold=2.2).transform(input\_data)

print("\n Binarized data:\n", data\_binarized)

Результат виконання:

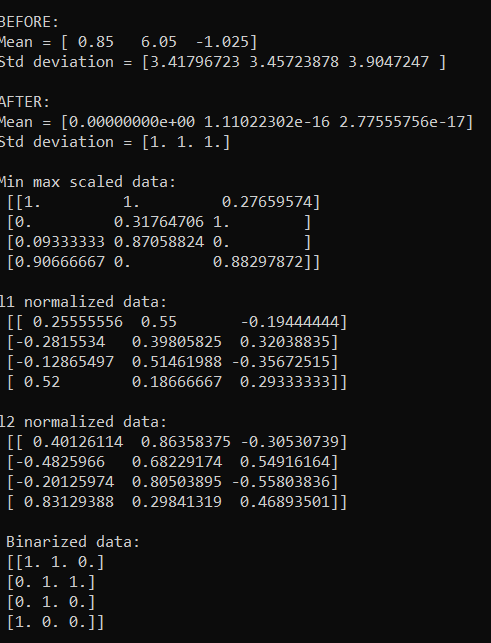


Рис. 1.3 – Результат виконання програми

1. Завдання 2.3. Класифікація логістичною регресією або логістичний класифікатор

Код 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, 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)

Результат виконання:

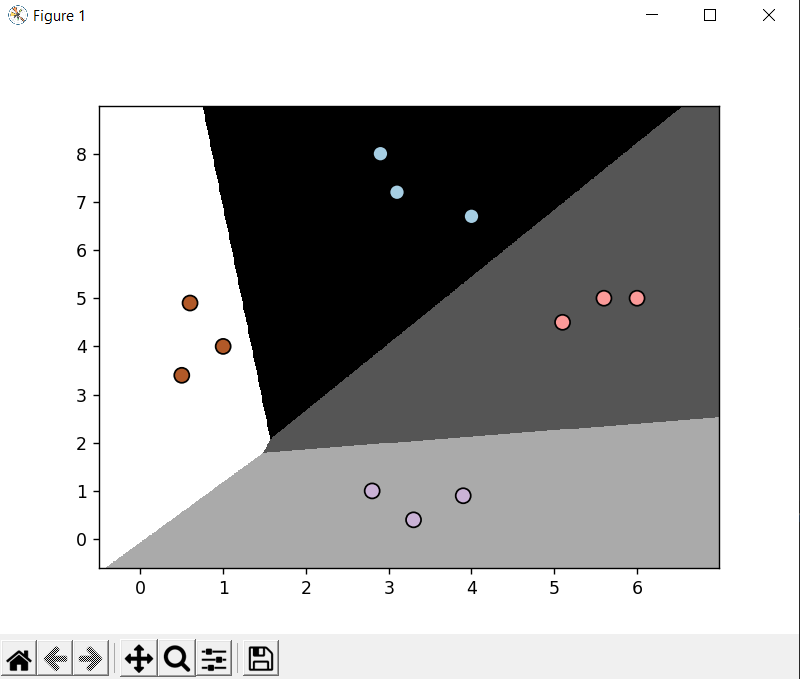


Рис. 1.4 – Результат виконання програми

1. Завдання 2.4. Класифікація наївним байєсовським класифікатором

Код 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

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)

Результат виконання:

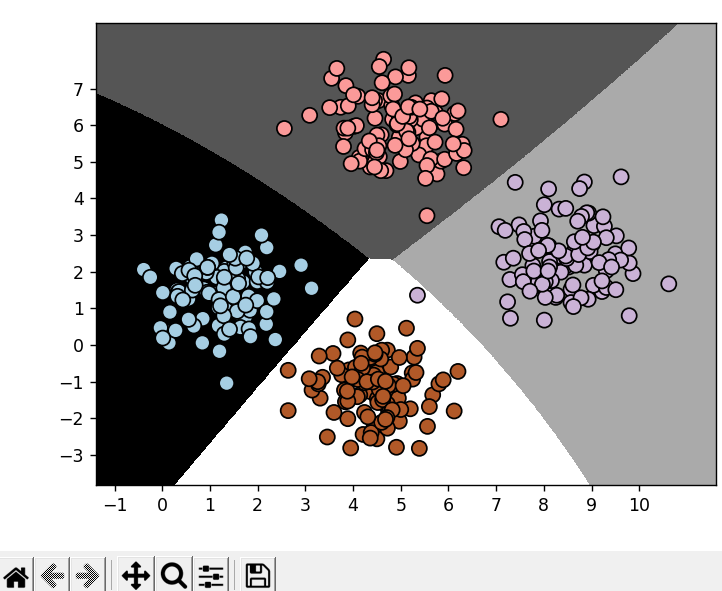


Рис. 1.5 – Результат виконання програми

1. Завдання 2.4.1

Код програми

import numpy as np

import matplotlib.pyplot as plt

import sklearn

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

data = np.loadtxt(input\_file, delimiter=',')

X, y = data[:, :-1], data[:, -1]

# 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), "%")

num\_folds = 3

accuracy\_values = sklearn.model\_selection.cross\_val\_score(classifier\_new, X, y, scoring='accuracy', cv=num\_folds)

print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2))

+ "%")

precision\_values = sklearn.model\_selection.cross\_val\_score(classifier\_new, X, y, scoring='precision\_weighted',

cv=num\_folds)

print("Precision: " + str(round(100 \* precision\_values.mean(),

2)) + "%")

recall\_values = sklearn.model\_selection.cross\_val\_score(classifier\_new, X, y, scoring='recall\_weighted', cv=num\_folds)

print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) +

"%")

f1\_values = sklearn.model\_selection.cross\_val\_score(classifier\_new, X, y, scoring='f1\_weighted', cv=num\_folds)

print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")

# Візуалізація роботи класифікатора

visualize\_classifier(classifier\_new, X\_test, y\_test)

Результат виконання:

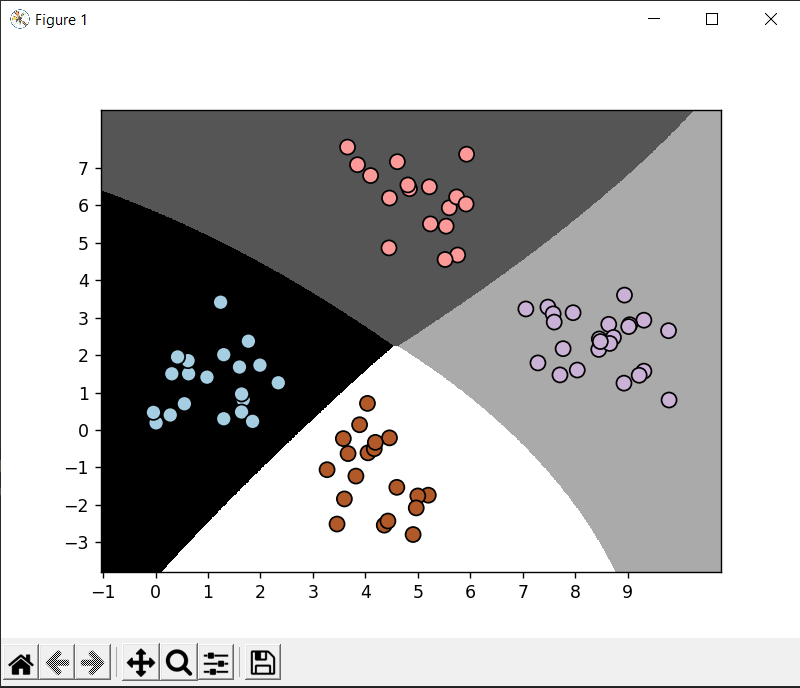


Рис. 1.6 – Результат виконання програми

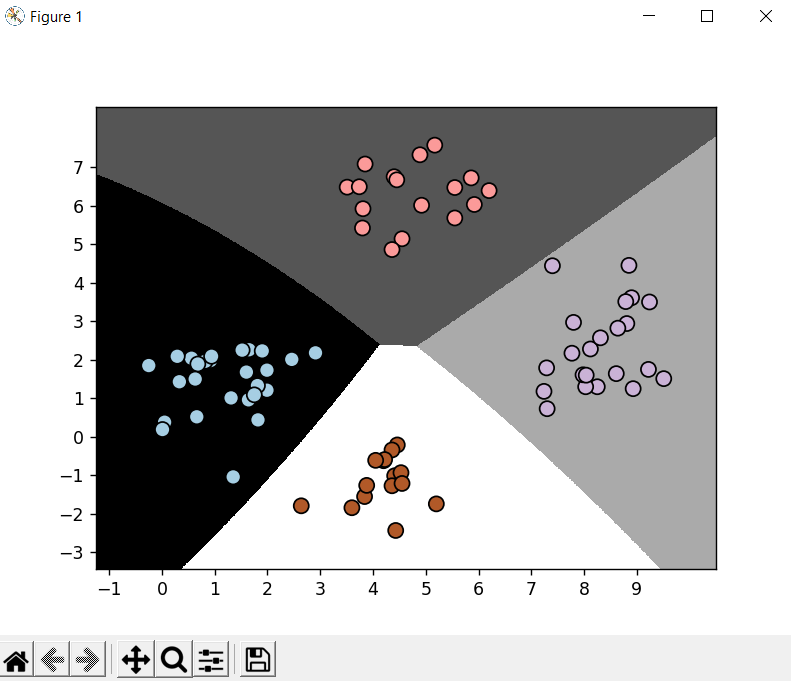


Рис. 1.7 – Результат після другого прогону

Як показано на рисунках, результати відрізняються, так як для другого прогону були обрані інші дані.

1. Завдання 2.5

Код програми збережено під назвою LR\_1\_task\_5.py

Результат виконання:

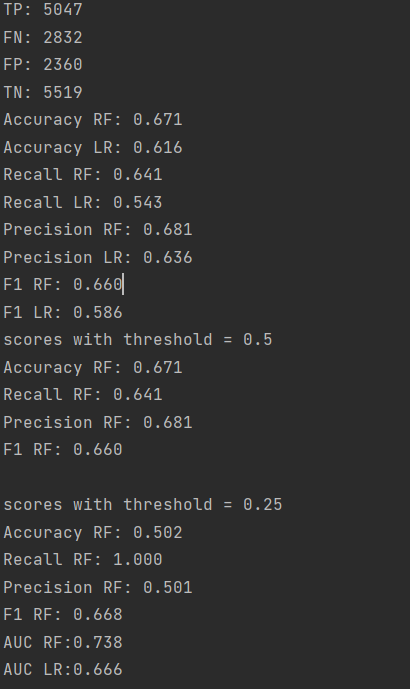


Рис. 1.8 – Результат виконання програми

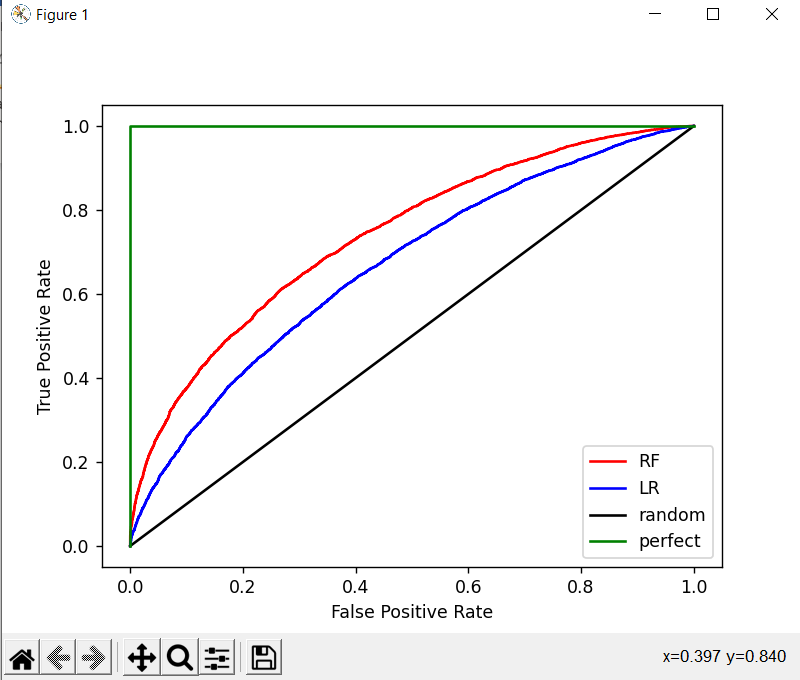


Рис. 1.9 – Результат виконання програми

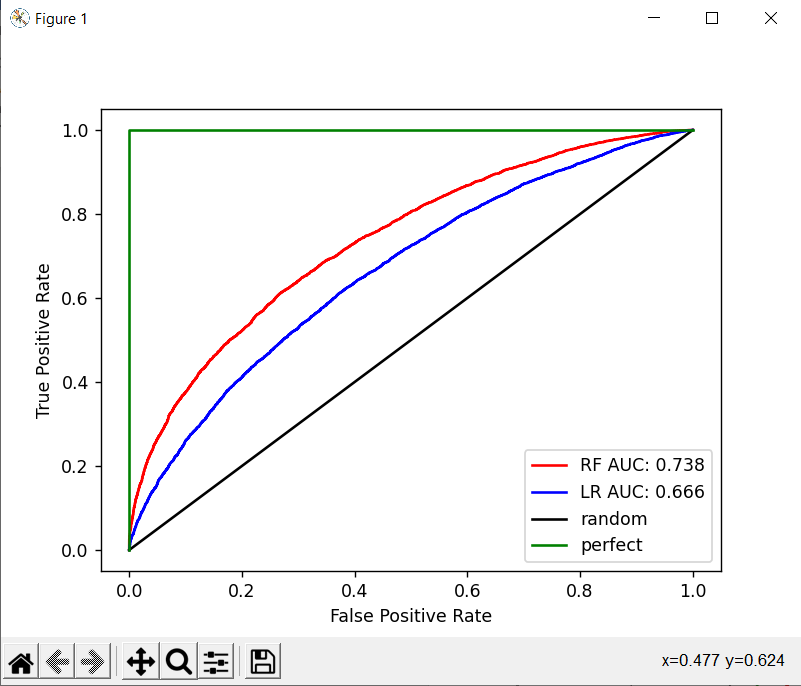


Рис. 1.10 – Результат виконання програми

**Висновок**: Отже я ознайомився з принципами роботи переривань та отримав практичні навички по організації переривань та структури програми.