

Міністерство освіти і науки України
Західноукраїнський національний університет

ЗВІТ
З МОДУЛЬНОЇ РОБОТИ №1
із дисципліни «Обчислювальний інтелект»
на тему «Дослідження побудови класифікатора та регресора методом k-
найближчих сусідів (k-nn)»
мета «Вивчити можливості аналізу даних з використанням класифікатора
та регресора методом k-найближчих сусідів (k-nn)»

Виконав:
Студент: Гадевич Володимир
Групи: КНм-11

Тернопіль 2024

Зміст

Вступ.....	3
Завдання 1. Створення KNN – класифікатора у Python	4
Завдання 2. Створення KNN – регресора у Python	10
Висновок	15

Вступ

Вивчення побудови класифікатора та регресора за допомогою методу k -найближчих сусідів (k -nn) є ключовим етапом у дослідженні обчислювального інтелекту. Метод k -nn, який базується на концепції схожості між прикладами даних, вважається одним із простіших та ефективних алгоритмів машинного навчання для класифікації та регресії.

Мета даної роботи полягає у дослідженні можливостей аналізу даних з використанням класифікатора та регресора, побудованих за допомогою методу k -найближчих сусідів (k -nn). У ході виконання дослідження буде реалізовано класифікатор та регресор k -nn у середовищі програмування Python.

Під час роботи буде створено класифікатор KNN на мові програмування Python, виконано завантаження бази даних, проведено необхідні операції над даними і визначено оптимальне значення K для досягнення найкращих показників якості класифікації на тестовій вибірці. Також буде розроблено регресор KNN у Python, визначено оптимальне значення K для досягнення найкращих показників якості регресії на тестовій вибірці та проведено візуалізацію отриманих результатів.

Завдання 1. Створення KNN – класифікатора у Python

Розробити програмну реалізацію Python, яка забезпечує виконання наступних кроків:

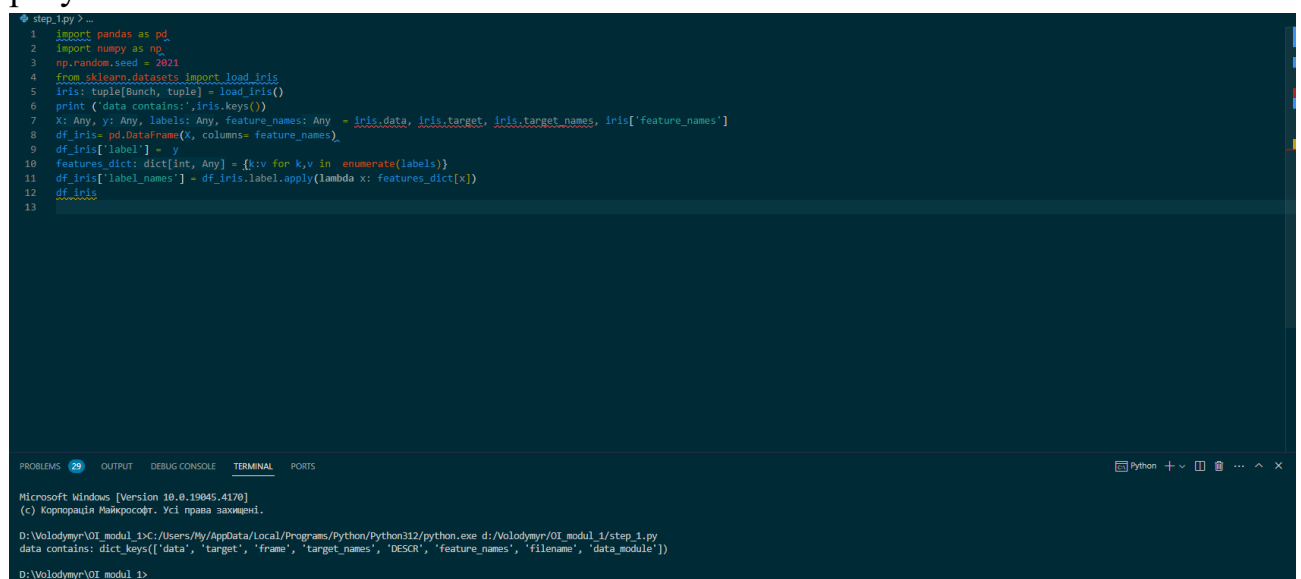
1. Завантажити базу параметрів квітів Iris DataSet
2. Перемішати записи у завантаженій базі
3. Нормалізувати параметри Iris
4. Розділити існуючі записи на навчальну і тестові вибірки
5. Навчити KNN-класифікатор з різними значеннями K
6. Вибрати величину K для найкращих показників якості класифікацій у тестовій вибірці

Крок. 1 — Завантажити базу параметрів квітів Iris DataSet

Iris DataSet - це набір даних, який включає інформацію про деякі характеристики квітів ірису. Цей набір даних є одним з найвідоміших у галузі машинного навчання та статистики і часто використовується для тестування та демонстрації алгоритмів класифікації. Iris dataset є одним з вбудованих наборів даних у бібліотеці scikit-learn у Python. Ви можете встановити цю бібліотеку, використовуючи pip:

```
pip install -U scikit-learn
```

Для перевірки чи вірно ми встановили бібліотеку можна запустити код наданий в допоміжному файлу «hometask_knn.ipynb» і отримуємо такий результат:



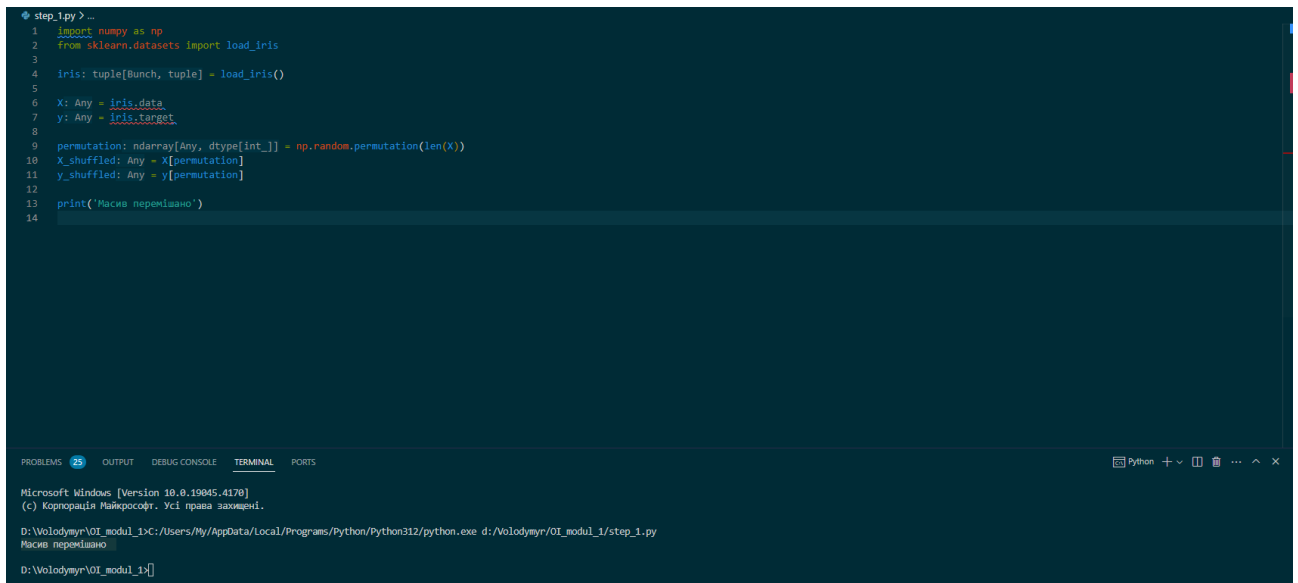
```
1 import pandas as pd
2 import numpy as np
3 np.random.seed = 2021
4 from sklearn.datasets import load_iris
5 iris = tuple(Bunch, tuple) = load_iris()
6 print('data contains:', iris.keys())
7 X: Any, y: Any, labels: Any, feature_names: Any = iris.data, iris.target, iris.target_names, iris['feature_names']
8 df_iris = pd.DataFrame(X, columns=feature_names)
9 df_iris['label'] = y
10 features_dict = dict(int, Any) = {k:v for k,v in enumerate(labels)}
11 df_iris['label_names'] = df_iris.label.apply(lambda x: features_dict[x])
12 df_iris
13
```

Microsoft Windows [Version 10.0.19045.4170]
(c) Корпорація Майкрософт. Усі права захищені.
D:\Volodymyr\OI_modul_1\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py
data contains: dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
D:\Volodymyr\OI_modul_1>

Рисунок 1.1 — Перевірка встановленої бібліотеки

Крок 2 — Перемішати записи у завантаженій базі

Для перемішування записів у базі Iris dataset використовуємо функцію `shuffle` з бібліотеки `numpy`.



```
1 import numpy as np
2 from sklearn.datasets import load_iris
3
4 iris = tuple[Bunch, tuple] = load_iris()
5
6 X = iris.data
7 y = iris.target
8
9 permutation: ndarray[Any, dtype[int_]] = np.random.permutation(len(X))
10 X_shuffled = X[permutation]
11 y_shuffled = y[permutation]
12
13 print('Масив перемішано')
```

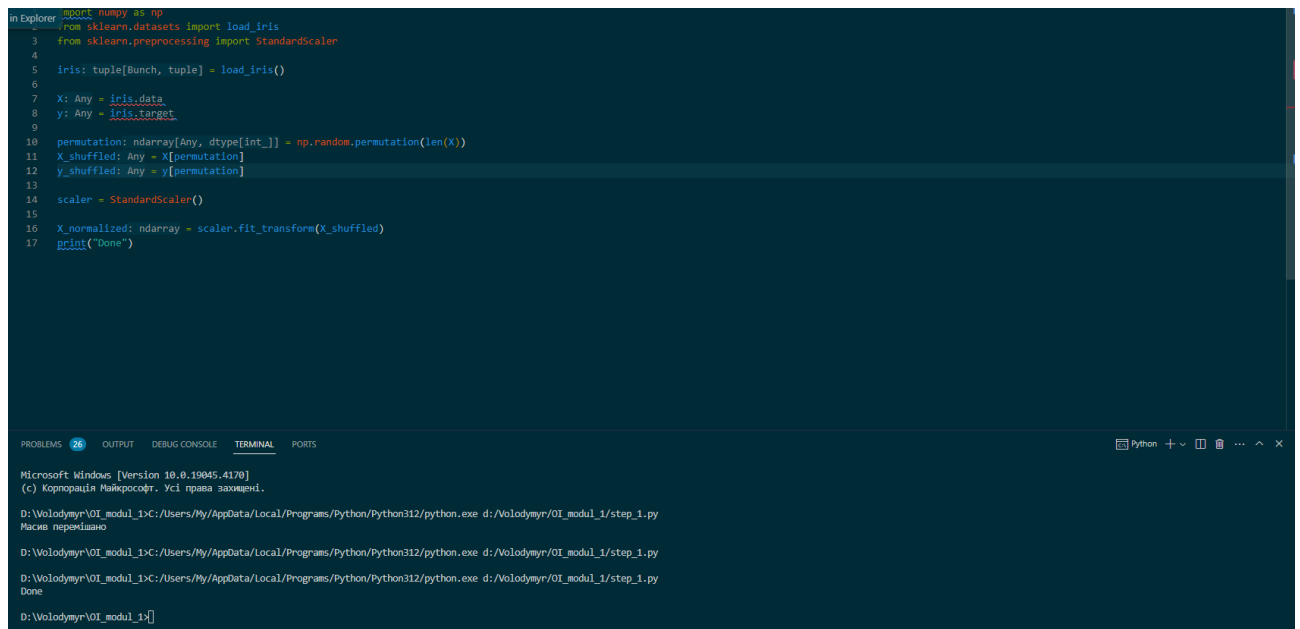
Рисунок 1.2 — Результат успішного перемішування даних

У цьому коді `permutation` - це випадкова перестановка індексів довжини даних. Потім ми застосовуємо цю перестановку до ознак (`X`) та міток класів (`y`), щоб перемішати їх у відповідності з цією перестановкою. Тепер `X_shuffled` та `y_shuffled` містять перемішані дані та мітки класів відповідно.

Крок 3 — Нормалізувати параметри Iris

Для нормалізації параметрів квітів ірису потрібно використати бібліотеку `scikit-learn`, за її допомогою можна використовувати стандартну нормалізацію, яка полягає у відніманні середнього значення і поділі на стандартне відхилення кожного параметра.

Оновимо вже існуючий код. `X_shuffled` - це перемішані ознаки, які були отримані на попередньому етапі. Клас `StandardScaler` нормалізує кожен параметр так, щоб його середнє значення стало рівним 0, а стандартне відхилення - 1. Метод `fit_transform` використовується для обчислення середнього та стандартного відхилення параметрів і одночасної нормалізації даних.



```
1 import numpy as np
2 from sklearn.datasets import load_iris
3 from sklearn.preprocessing import StandardScaler
4
5 iris = tuple[Bunch, tuple] = load_iris()
6
7 X: Any = iris.data
8 y: Any = iris.target
9
10 permutation: ndarray[Any, dtype[int_]] = np.random.permutation(len(X))
11 X_shuffled: Any = X[permutation]
12 y_shuffled: Any = y[permutation]
13
14 scaler = StandardScaler()
15
16 X_normalized: ndarray = scaler.fit_transform(X_shuffled)
17 print("Done")
```

Microsoft Windows [Version 10.0.19045.4170]
(c) Корпорація Майкрософт. Усі права захищені.

D:\Volodymyr\OI_modul_1>C:/Users/My/AppData/Local/Programs/Python/Python312/python.exe d:/Volodymyr/OI_modul_1/step_1.py
Масив перемішано

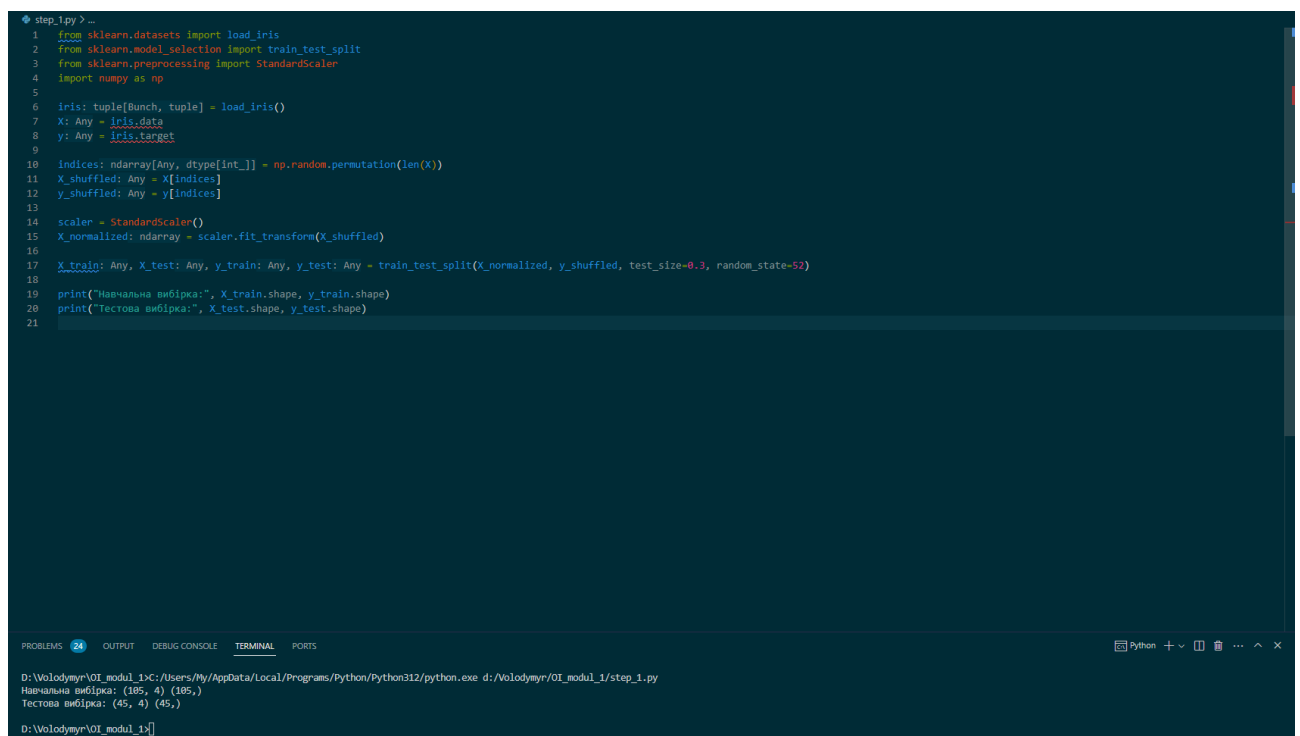
D:\Volodymyr\OI_modul_1>C:/Users/My/AppData/Local/Programs/Python/Python312/python.exe d:/Volodymyr/OI_modul_1/step_1.py
Done

D:\Volodymyr\OI_modul_1>

Рисунок 1.3 — Нормалізування параметрів Iris

Крок 4 — Розділити існуючі записи на навчальну і тестові вибірка

Для того, щоб розділити записи в базі даних на навчальну і тестову, потрібно послідовно виконати кроки 1, 2, 3, а саме реалізувати перемішування записів, нормалізацію параметрів та розділення на навчальну та тестову вибірки.



```
1 from sklearn.datasets import load_iris
2 from sklearn.model_selection import train_test_split
3 from sklearn.preprocessing import StandardScaler
4 import numpy as np
5
6 iris = tuple[Bunch, tuple] = load_iris()
7 X: Any = iris.data
8 y: Any = iris.target
9
10 indices: ndarray[Any, dtype[int_]] = np.random.permutation(len(X))
11 X_shuffled: Any = X[indices]
12 y_shuffled: Any = y[indices]
13
14 scaler = StandardScaler()
15 X_normalized: ndarray = scaler.fit_transform(X_shuffled)
16
17 X_train: Any, X_test: Any, y_train: Any, y_test: Any = train_test_split(X_normalized, y_shuffled, test_size=0.3, random_state=52)
18
19 print("Навчальна вибірка:", X_train.shape, y_train.shape)
20 print("Тестова вибірка:", X_test.shape, y_test.shape)
21
```

D:\Volodymyr\OI_modul_1>C:/Users/My/AppData/Local/Programs/Python/Python312/python.exe d:/Volodymyr/OI_modul_1/step_1.py
Навчальна вибірка: (105, 4) (105,)
Тестова вибірка: (45, 4) (45,)

D:\Volodymyr\OI_modul_1>

Рисунок 1.4 — Розділення вибірки на тестову та навчальну

Після виконання коду ми можемо бачити повідомлення в терміналі:

Навчальна вибірка: (105, 4) (105,)

Тестова вибірка: (45, 4) (45,)

Крок 5 — Навчити KNN-класифікатор з різними значеннями K

Для навчання KNN-класифікатора спочатку потрібно створити список різних значень K. Потім запустити цикл навчання та оцінки класифікатора для кожного значення K. Для кожного значення K ми створюємо KNN-класифікатор, навчаємо його на навчальних даних, прогнозуємо класи для тестових даних, обчислюємо точність класифікації та виводимо результати

```
1 from sklearn.datasets import load_iris
2 from sklearn.model_selection import train_test_split
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.neighbors import KNeighborsClassifier
5 from sklearn.metrics import accuracy_score
6 import numpy as np
7
8 iris = load_iris()
9 X = iris.data
10 y = iris.target
11
12 indices = np.arange(X.shape[0])
13 X_shuffled = X[indices]
14 y_shuffled = y[indices]
15
16 scaler = StandardScaler()
17 X_normalized = scaler.fit_transform(X_shuffled)
18
19 X_train, X_test, y_train, y_test = train_test_split(X_normalized, y_shuffled, test_size=0.3, random_state=52)
20
21 k_values = list(range(1, 10))
22
23 for k in k_values:
24     knn_classifier = KNeighborsClassifier(n_neighbors=k)
25     knn_classifier.fit(X_train, y_train)
26     y_pred = knn_classifier.predict(X_test)
27     accuracy = accuracy_score(y_test, y_pred)
28     print(f'K = {k}, Точність класифікації: {accuracy}')
29
30
31
32
```

Microsoft Windows [Version 10.0.19045.4170]
(c) Корпорація Майкрософт. Усі права захищені.

D:\Volodymyr\OI_modul_1> python d:\Volodymyr\OI_modul_1\step_1.py

K = 1 Точність класифікації: 0.9111111111111111
K = 2 Точність класифікації: 0.9111111111111111
K = 3 Точність класифікації: 0.9111111111111111
K = 5 Точність класифікації: 0.9111111111111111
K = 9 Точність класифікації: 0.9555555555555556

D:\Volodymyr\OI_modul_1>

Рисунок 1.5 — Результат навчання

Після запуску програми ми бачимо результат обробки даних у терміналі:

```
K = 1 Точність класифікації: 0.9111111111111111
K = 2 Точність класифікації: 0.9111111111111111
K = 3 Точність класифікації: 0.9111111111111111
K = 5 Точність класифікації: 0.9111111111111111
K = 9 Точність класифікації: 0.9555555555555556
```

Крок 6 — Вибрати величину K для найкращих показників якості класифікацій у тестовій вибірці

Щоб вибрати найкращу величину K для найкращих показників якості класифікації у тестовій вибірці, ми можемо обирати ту величину K, яка має найвищу точність класифікації на тестовій вибірці. Для цього перебираємо різні значення K і зберігаємо найкращу точність та відповідну величину K. В результаті циклу ми виводимо найкращу величину K та відповідну точність.

```

task1.py > ...
3 from sklearn.preprocessing import StandardScaler
4 import numpy as np
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.metrics import accuracy_score
7
8 iris = tuple[Bunch, tuple] = load_iris()
9 X: Any = iris.data
10 y: Any = iris.target
11
12 indices: ndarray[Any, dtype[int]] = np.random.permutation(len(X))
13 X_shuffled: Any = X[indices]
14 y_shuffled: Any = y[indices]
15
16 scaler = StandardScaler()
17 X_normalized: ndarray = scaler.fit_transform(X_shuffled)
18
19 X_train: Any, X_test: Any, y_train: Any, y_test: Any = train_test_split(X_normalized, y_shuffled, test_size=0.3, random_state=52)
20
21 print("Навчальна вибірка:", X_train.shape, y_train.shape)
22 print("Тестова вибірка:", X_test.shape, y_test.shape)
23
24 k_values: list[int] = [1, 2, 3, 5, 9]
25
26 for k in k_values:
27     knn_classifier = KNeighborsClassifier(n_neighbors=k)
28     knn_classifier.fit(X_train, y_train)
29     y_pred: ndarray = knn_classifier.predict(X_test)
30     accuracy: Float = accuracy_score(y_test, y_pred)
31     print("K =", k, "Точність класифікації:", accuracy)
32
33 best_accuracy = 0
34 best_k = 0
35
36 for k in k_values:
37     knn_classifier = KNeighborsClassifier(n_neighbors=k)
38     knn_classifier.fit(X_train, y_train)
39     y_pred: ndarray = knn_classifier.predict(X_test)
40     accuracy: Float = accuracy_score(y_test, y_pred)
41     if accuracy > best_accuracy:
42         best_accuracy = accuracy
43         best_k: int = k
44
45 print("Найкраща величина K:", best_k)
46 print("Точність класифікації для найкращої величини K:", best_accuracy)

```

Рисунок 1.6 — Повний код програми

В цьому коді ми успішно об'єднали всі пройдені кроки та створили KNN – класифікатора у Python. Після запуску коду у термінал виводиться результат обробки даних:

```

task1.py > ...
19 X_train: Any, X_test: Any, y_train: Any, y_test: Any = train_test_split(X_normalized, y_shuffled, test_size=0.3, random_state=52)
20
21 print("Навчальна вибірка:", X_train.shape, y_train.shape)
22 print("Тестова вибірка:", X_test.shape, y_test.shape)
23
24 k_values: list[int] = [1, 2, 3, 5, 9]
25
26 for k in k_values:
27     knn_classifier = KNeighborsClassifier(n_neighbors=k)
28     knn_classifier.fit(X_train, y_train)
29     y_pred: ndarray = knn_classifier.predict(X_test)
30     accuracy: Float = accuracy_score(y_test, y_pred)
31     print("K =", k, "Точність класифікації:", accuracy)
32
33 best_accuracy = 0
34 best_k = 0
35
36 for k in k_values:
37     knn_classifier = KNeighborsClassifier(n_neighbors=k)
38     knn_classifier.fit(X_train, y_train)
39     y_pred: ndarray = knn_classifier.predict(X_test)
40     accuracy: Float = accuracy_score(y_test, y_pred)
41     if accuracy > best_accuracy:
42         best_accuracy = accuracy
43         best_k: int = k
44
45 print("Найкраща величина K:", best_k)
46 print("Точність класифікації для найкращої величини K:", best_accuracy)

```

PROBLEMS 23 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Точність класифікації для найкращої величини K: 0.8888888888888888

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\task1.py

Навчальна вибірка: (105, 4) (105,)

Тестова вибірка: (45, 4) (45,)

K = 1 Точність класифікації: 1.0

K = 2 Точність класифікації: 0.9777777777777777

K = 3 Точність класифікації: 0.9777777777777777

K = 5 Точність класифікації: 0.9777777777777777

K = 9 Точність класифікації: 0.9555555555555555

Найкраща величина K: 1

Точність класифікації для найкращої величини K: 1.0

D:\Volodymyr\OI_modul_1>

Рисунок 1.7 — Результат роботи коду виведений у термінал

За результатами роботи у першому завданні - Створення KNN – класифікатора у Python ми отримали наступні результати:

Навчальна вибірка: (105, 4) (105,)

Тестова вибірка: (45, 4) (45,)

K = 1 Точність класифікації: 1.0

K = 2 Точність класифікації: 0.9777777777777777

K = 3 Точність класифікації: 0.9777777777777777

K = 5 Точність класифікації: 0.9777777777777777

K = 9 Точність класифікації: 0.9555555555555556

Найкраща величина K: 1

Точність класифікації для найкращої величини K: 1.0

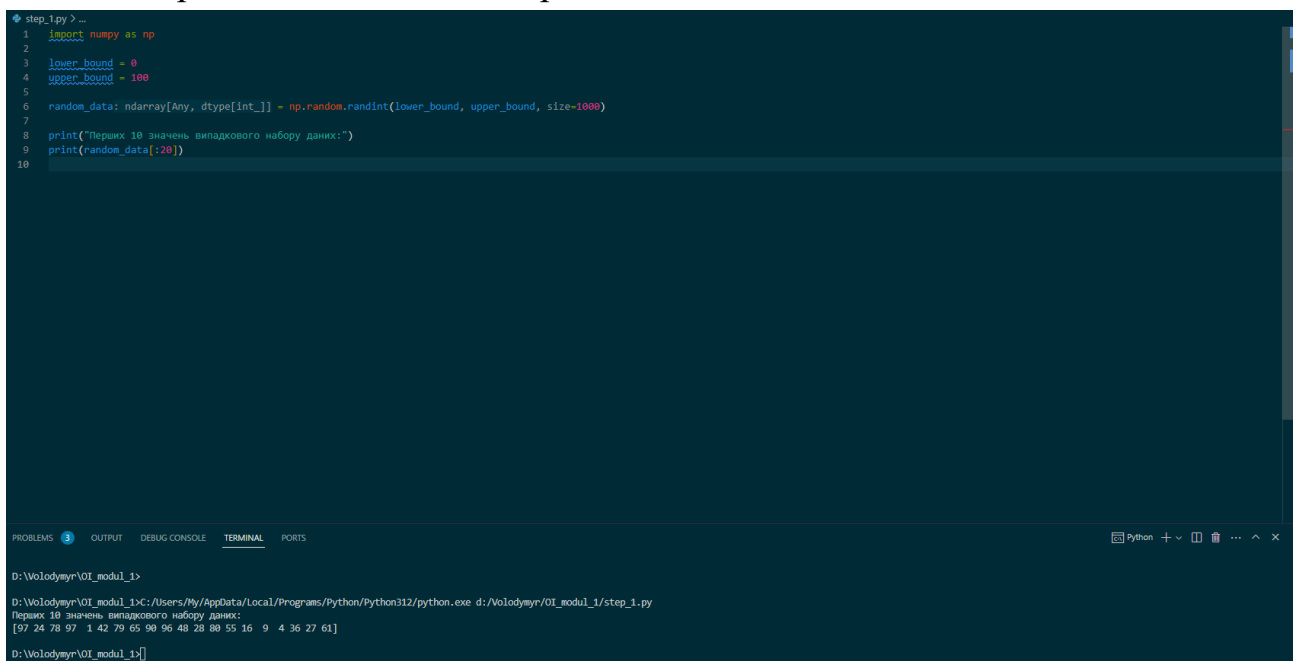
Завдання 2. Створення KNN – регресора у Python

Розробити програмну реалізацію Python, яка забезпечує виконання наступних кроків:

1. Згенерувати випадковий набір даних в діапазоні 1000 значень
2. Нормалізувати значення.
3. Розділити існуючі записи на навчальну і тестові вибірки
4. Навчити KNN-регресор з різними значеннями K
5. Вибрати величину K для найкращих показників якості регресії у тестовій вибірці
6. Здійснити візуалізації отриманих

Крок 1 — Згенерувати випадковий набір даних в діапазоні 1000 значень

Генеруємо за допомогою випадкового набору дані в діапазоні до 1000 значень. Для цього ми можемо скористатися бібліотекою NumPy. Та вказуємо вивести перші 20 елементів згенерованих даних.



```
step_1.py -  
1 import numpy as np  
2  
3 lower_bound = 0  
4 upper_bound = 100  
5  
6 random_data: ndarray[Any, dtype[int_]] = np.random.randint(lower_bound, upper_bound, size=1000)  
7  
8 print("Перших 10 значень випадкового набору даних:")  
9 print(random_data[:20])  
10
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

D:\Volodymyr\OI_modul_1>
D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py
Перших 10 значень випадкового набору даних:
[97 24 78 97 1 42 79 65 90 96 48 28 80 55 16 9 4 36 27 61]
D:\Volodymyr\OI_modul_1>

Рисунок 2.1 — Вивід частки згенерованих даних

Крок 2 — Нормалізувати значення

Для виконання другого кроку, а саме проведення нормалізування значення ми можемо використати стандартне відхилення та середнє значення. Реалізувавши код, який використовує ці значення для нормалізації нашого

випадкового набору даних. Він виведе перші 20 нормалізованих значень для перевірки у термінал:

```
step_1.py > ...
1 import numpy as np
2
3 lower_bound = 0
4 upper_bound = 100
5
6 random_data: ndarray[Any, dtype[int_]] = np.random.randint(lower_bound, upper_bound, size=1000)
7
8 normalized_data = (random_data - np.mean(random_data)) / np.std(random_data)
9
10 print("Перших 20 значень нормалізованого набору даних:")
11 print(normalized_data[:20])
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

D:\Volodymyr\OI_modul_1>

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Перших 10 значень випадкового набору даних:

[97 24 78 97 1 42 79 65 90 96 48 28 80 55 16 9 4 36 27 61]

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Перших 10 значень нормалізованого набору даних:

[1.36990604 -0.71656382 -1.34258374 -1.53115865 1.26558258 -1.4468272
1.38835787 1.26558258 1.57855294 -1.58592513]

D:\Volodymyr\OI_modul_1>

Рисунок 2.2 — Нормалізація даних

Крок 3 — Розділити існуючі записи на навчальну і тестові вибірки

Щоб розділити вже існуючі запис на навчальну і тестові вибірки ми можемо скористатися функцією `train_test_split` з бібліотеки `sklearn`. Ось як ми можемо зробити це:

```
step_1.py > ...
1 import numpy as np
2 from sklearn.model_selection import train_test_split
3
4 lower_bound = 0
5 upper_bound = 100
6
7 random_data: ndarray[Any, dtype[int_]] = np.random.randint(lower_bound, upper_bound, size=1000)
8
9 normalized_data = (random_data - np.mean(random_data)) / np.std(random_data)
10
11 X_train: Any, X_test: Any = train_test_split(normalized_data, test_size=0.3, random_state=52)
12
13 print("Навчальна вибірка:", X_train.shape)
14 print("Тестова вибірка:", X_test.shape)
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

D:\Volodymyr\OI_modul_1>

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Перших 10 значень випадкового набору даних:

[97 24 78 97 1 42 79 65 90 96 48 28 80 55 16 9 4 36 27 61]

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Перших 10 значень нормалізованого набору даних:

[1.36990604 -0.71656382 -1.34258374 -1.53115865 1.26558258 -1.4468272
1.38835787 1.26558258 1.57855294 -1.58592513]

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Навчальна вибірка: (700,)

Тестова вибірка: (300,)

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Навчальна вибірка: (700,)

Тестова вибірка: (300,)

D:\Volodymyr\OI_modul_1>

Рисунок 2.3 — Виконання коду


У цьому коді ми використовуємо розділили наші дані на навчальну та тестову вибірки у співвідношенні 80/20 відповідно і отримали в результаті:

Навчальна вибірка: (700,)

Тестова вибірка: (300,)

Крок 4 — Навчити KNN-регресор з різними значеннями K

На цьому кроці всі ми навчаємо KNN-регресор з різними значеннями K від 1 до 10 та оцінюємо якість прогнозів за допомогою середньоквадратичної помилки (MSE).



```
step_1.py > ...
1 import numpy as np
2 from sklearn.model_selection import train_test_split
3 from sklearn.neighbors import KNeighborsRegressor
4 from sklearn.metrics import mean_squared_error
5
6 lower_bound = 0
7 upper_bound = 100
8 random_data: ndarray[Any, dtype[int]] = np.random.randint(lower_bound, upper_bound, size=1000)
9
10 normalized_data = (random_data - np.mean(random_data)) / np.std(random_data)
11
12 X_train: Any, X_test: Any = train_test_split(normalized_data, test_size=0.3, random_state=52)
13
14
15 print("Навчальна вибірка:", X_train.shape)
16 print("Тестова вибірка:", X_test.shape)
17
18 for k in range(1, 11):
19     knn_regressor = KNeighborsRegressor(n_neighbors=k)
20     knn_regressor.fit(X_train.reshape(-1, 1), X_train)
21     predictions: ndarray = knn_regressor.predict(X_test.reshape(-1, 1))
22     mse: Float | ndarray = mean_squared_error(X_test, predictions)
23     print(f"K = {k}, MSE = {mse}")
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Навчальна вибірка: (700,)
Тестова вибірка: (300,)

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Навчальна вибірка: (700,)
Тестова вибірка: (300,)

D:\Volodymyr\OI_modul_1>C:\Users\My\AppData\Local\Programs\Python\Python312\python.exe d:\Volodymyr\OI_modul_1\step_1.py

Навчальна вибірка: (700,)
Тестова вибірка: (300,)
K = 1, MSE = 0.0
K = 2, MSE = 0.0
K = 3, MSE = 3.835229582635894e-06
K = 4, MSE = 1.030717950333402e-05
K = 5, MSE = 1.8102283630041593e-05
K = 6, MSE = 3.16406440567464e-05
K = 7, MSE = 5.2988784233561684e-05
K = 8, MSE = 7.946116291523799e-05
K = 9, MSE = 0.00010000006022872921
K = 10, MSE = 0.00011708955915787457

D:\Volodymyr\OI_modul_1>

Рисунок 2.4 — Результат навчання KNN-регресора

Після запуску коду ми можемо бачити результат обчислень виведений у термінал:

```
Навчальна вибірка: (700,)
Тестова вибірка: (300,)
K = 1, MSE = 0.0
K = 2, MSE = 0.0
K = 3, MSE = 3.835229582635894e-06
K = 4, MSE = 1.030717950333402e-05
K = 5, MSE = 1.8102283630041593e-05
K = 6, MSE = 3.16406440567464e-05
K = 7, MSE = 5.2988784233561684e-05
K = 8, MSE = 7.946116291523799e-05
K = 9, MSE = 0.00010000006022872921
K = 10, MSE = 0.00011708955915787457
```

Крок 5 — Вибрати величину K для найкращих показників якості регресії у тестовій вибірці

Для вибору оптимального значення K за найкращими показниками якості регресії у тестовій вибірці було порівняно середньоквадратичну помилку (MSE)

для різних значень K . Потім виберемо те значення K , для якого MSE найменше. У цьому кроці ми проходимо по різним значенням K від 1 до 10, навчаємо модель за кожним значенням K , оцінюємо якість за допомогою MSE і обираємо те значення K , для якого MSE найменше.

```

step1.py > ...
1 import numpy as np
2 from sklearn.model_selection import train_test_split
3 from sklearn.neighbors import KNeighborsRegressor
4 from sklearn.metrics import mean_squared_error
5
6 lower_bound = 0
7 upper_bound = 100
8 random_data: ndarray[Any, dtype[int_]] = np.random.randint(lower_bound, upper_bound, size=1000)
9
10 normalized_data = (random_data - np.mean(random_data)) / np.std(random_data)
11
12 X_train: Any, X_test: Any = train_test_split(normalized_data, test_size=0.3, random_state=52)
13
14
15 print("Навчальна вибірка:", X_train.shape)
16 print("Тестова вибірка:", X_test.shape)
17
18 for k in range(1, 11):
19     knn_regressor = KNeighborsRegressor(n_neighbors=k)
20     knn_regressor.fit(X_train.reshape(-1, 1), X_train)
21     predictions: ndarray = knn_regressor.predict(X_test.reshape(-1, 1))
22     mse: float | ndarray = mean_squared_error(X_test, predictions)
23     print(f"K = {k}, MSE = {mse}")
24
25 best_k = None
26 best_mse = float('inf')
27 for k in range(1, 11):
28     knn_regressor = KNeighborsRegressor(n_neighbors=k)
29     knn_regressor.fit(X_train.reshape(-1, 1), X_train)
30     predictions: ndarray = knn_regressor.predict(X_test.reshape(-1, 1))
31     mse: float | ndarray = mean_squared_error(X_test, predictions)
32     if mse < best_mse:
33         best_mse: float | ndarray = mse
34         best_k: int = k
35
36 print(f"Найкраще значення K: {best_k} з MSE = {best_mse}")

```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

K = 3, MSE = 1.7082325592376454e-06
K = 4, MSE = 9.88298334935458e-06
K = 5, MSE = 2.52484535846791e-05
K = 6, MSE = 4.165509770132227e-05
K = 7, MSE = 6.355952922946427e-05
K = 8, MSE = 8.54765353822984e-05
K = 9, MSE = 0.00010413924425339533
K = 10, MSE = 0.00011905028379781921
Найкраще значення K: 1 з MSE = 0.0
D:\Volodymyr\OI_modul_1\

```

Рисунок 2.5 — Реалізація кроку №5

Крок 6 — Здійснити візуалізації отриманих

Для візуалізації отриманих даних потрібно використати бібліотеку Matplotlib Також ось повний код програми для вирішення завдання №2:

```

task2.py > ...
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.model_selection import train_test_split
4 from sklearn.neighbors import KNeighborsRegressor
5 from sklearn.metrics import mean_squared_error
6
7 lower_bound = 0
8 upper_bound = 100
9 random_data: ndarray[Any, dtype[int_]] = np.random.randint(lower_bound, upper_bound, size=1000)
10
11 normalized_data = (random_data - np.mean(random_data)) / np.std(random_data)
12
13 X_train: Any, X_test: Any = train_test_split(normalized_data, test_size=0.3, random_state=52)
14
15
16 best_k = None
17 best_mse = float('inf')
18 mse_values: list = []
19 for k in range(1, 11):
20     knn_regressor = KNeighborsRegressor(n_neighbors=k)
21     knn_regressor.fit(X_train.reshape(-1, 1), X_train)
22     predictions: ndarray = knn_regressor.predict(X_test.reshape(-1, 1))
23     mse: float | ndarray = mean_squared_error(X_test, predictions)
24     mse_values.append(mse)
25     if mse < best_mse:
26         best_mse: float | ndarray = mse
27         best_k: int = k
28
29 print(f"Найкраще значення K: {best_k} з MSE = {best_mse}")
30
31 plt.figure(figsize=(10, 6))
32 plt.plot(range(1, 11), mse_values, marker='o', linestyle='-', color='b')
33 plt.title("Залежність MSE від значення K")
34 plt.xlabel("Значення K")
35 plt.ylabel("MSE")
36 plt.xticks(range(1, 11))
37 plt.grid(True)
38 plt.show()

```

Рисунок 2.6 — Повний код із варіантом візуалізації графіком

Давайте створимо два варіанти візуалізації результатів графік – рис. 2.7 та діаграму – рис. 2.8

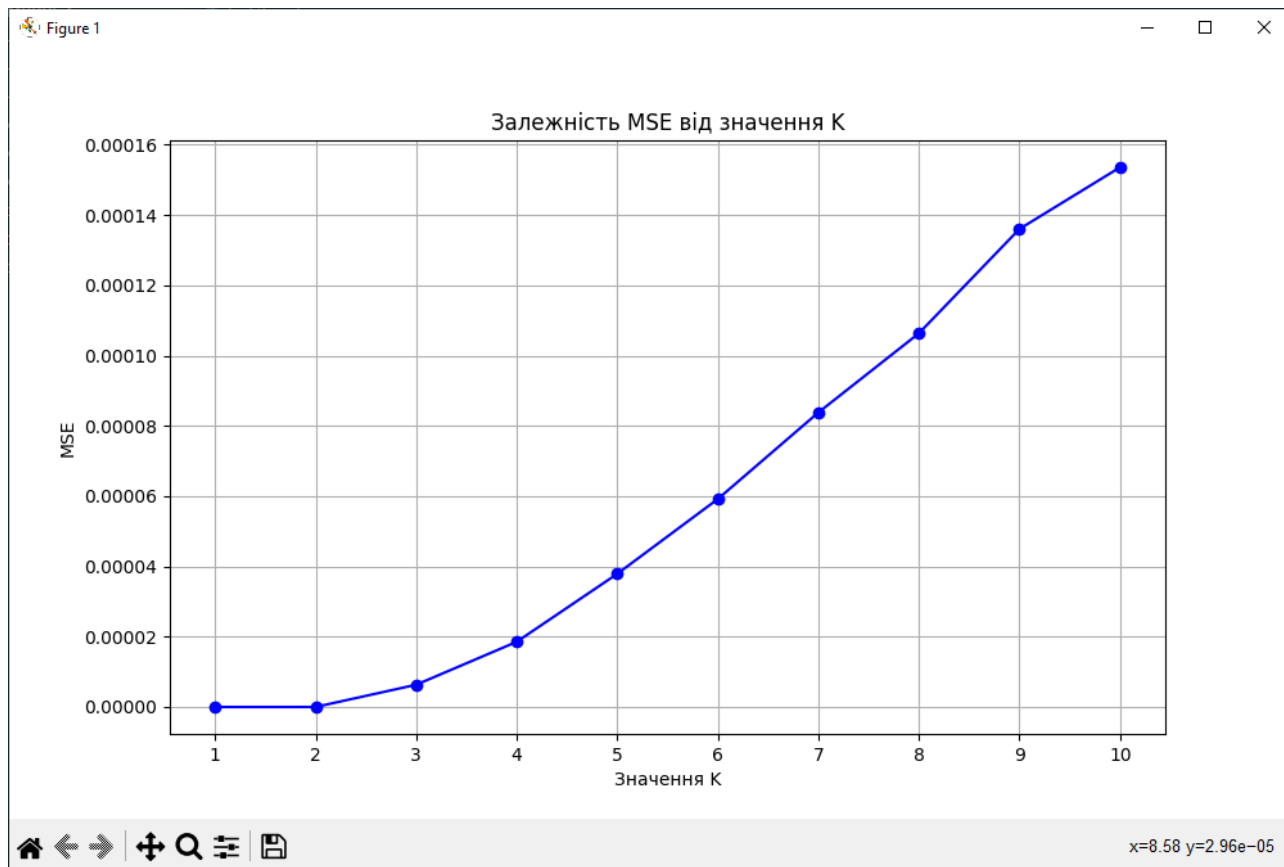


Рисунок 2.7 — Візуалізація даних графіком

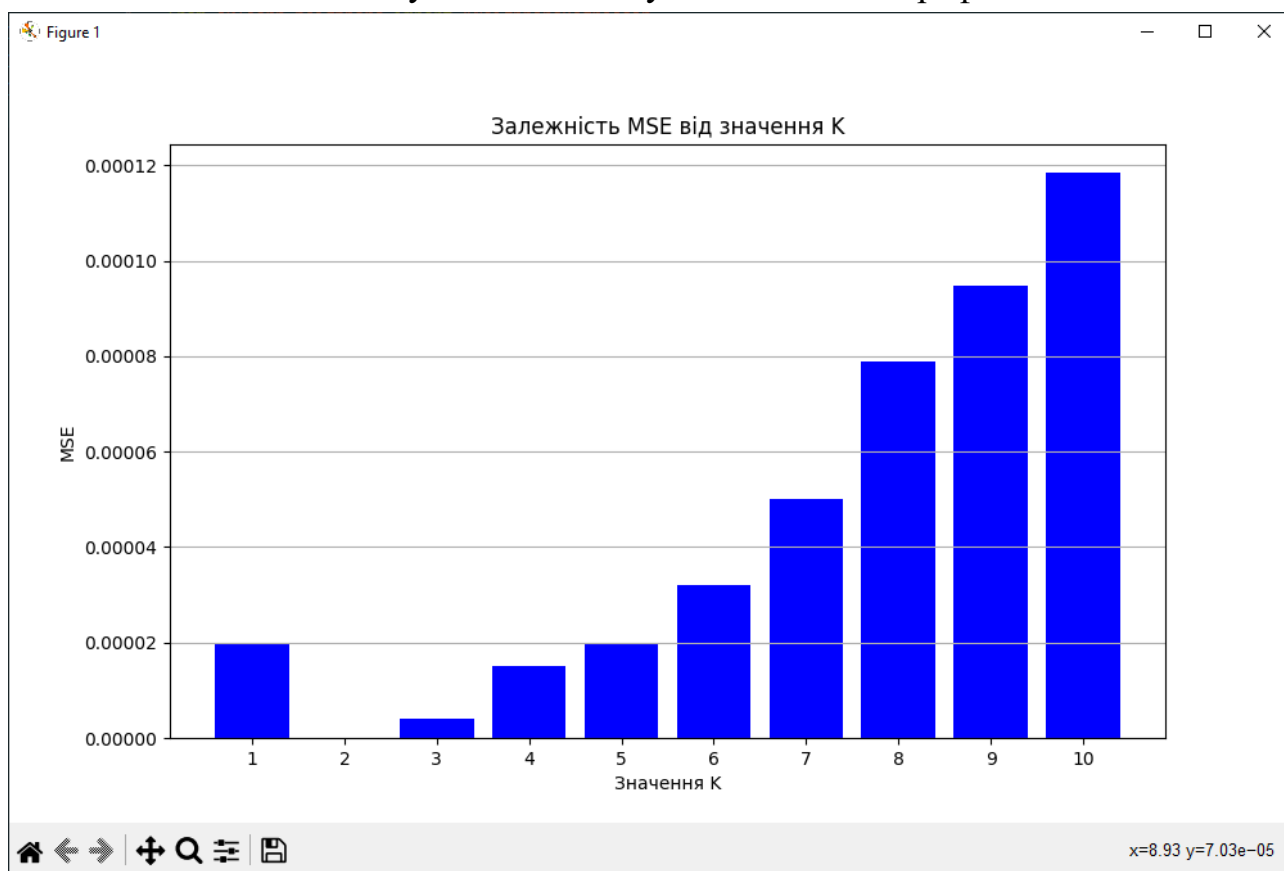


Рисунок 2.8 — Візуалізація даних діаграмою

Як ми бачимо обидві візуалізації є однаковими тому робота виконана правильно.

Висновок

В цій роботі ми на практиці вивчили, як побудувати класифікатор та регресор за допомогою методу k-найближчих сусідів (k-nn). Виконали дослідження можливостей аналізу даних. У ході виконання було реалізовано класифікатор та регресор k-nn у середовищі програмування Python.

Під час роботи було виконані основні операції пов'язані із аналізом даних на мові програмування Python, виконано завантаження бази даних, проведено необхідні операції над даними і визначено оптимальне значення K для досягнення найкращих показників якості класифікації на тестовій вибірці та проведено візуалізацію отриманих результатів.