Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Звіт

Про виконання лабораторної роботи №2
Дослідницький аналіз даних у Python.
з дисципліни « Методи та системи штучного інтелекту»

Виконав:
Студент групи КН-33
Цьома І.С.

Тернопіль 2024р.

ЗВІТ ЛАБОРАТОРНОЇ РОБОТИ №2: Побудова регресора методом k-найближчих сусідів (k-nn)

Мета роботи: отримати навики з аналізу даних з використанням регресора за методом k-найблищих сусідів (k-nn).

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

Розробити програмну реалізацію Matlab/Prolog, яка забезпечує виконання наступних кроків:

- Згенерувати вибадковий набір даних в діапазоні 1000 значень.
- Нормалізувати значення.
- Розділити існуючі записи на навчальну і тестові вибірки.
- Навчити KNN-регресор з різними значеннями К.
- Вибрати величину К для найкращих показників якості регресії у тестовій вибірці.
- Здійснити візуалізації отриманих рішень.

Хід роботи

Крок 1: Генерація випадкового набору даних

Ми згенеруємо 1000 випадкових чисел в діапазоні [0, 1000].

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Крок 1: Генерація випадкових даних
data = np.random.rand(1000) * 1000
print("Згенеровані дані:", data[:10])  # Виведемо перші 10 елементів для перевірки
```

Крок 2: Нормалізація значень

Нормалізуємо дані за допомогою мінімаксної нормалізації, щоб привести значення у діапазон [0, 1].

```
# Крок 2: Нормалізація даних
scaler = MinMaxScaler()
data_normalized = scaler.fit_transform(data.reshape(-1, 1))
print("Нормалізовані дані:", data_normalized[:10])  # Перевірка перших 10 елементів
```

Крок 3: Розділення на навчальну та тестову вибірки

Ми розділимо дані на навчальну (80%) та тестову (20%) вибірки.

```
# Крок 3: Розділення на навчальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(data_normalized, data_normalized, test_size=0.2, random_state=42)
print(f"Розмір навчальної вибірки: {len(X_train)}")
print(f"Розмір тестової вибірки: {len(X_test)}")
```

Крок 4: Навчання KNN-регресора з різними значеннями K

Ми будемо використовувати алгоритм KNN-регресії для різних значень K і визначимо найкраще значення.

```
# Крок 4: Навчання KNN-регресора з різними значеннями K
def evaluate_knn(X_train, X_test, y_train, y_test, k_values):
    best_k = None
    best_mse = float('inf')
    results = {}

    for k in k_values:
        knn = KNeighborsRegressor(n_neighbors=k)
        knn.fit(X_train, y_train)

        y_pred = knn.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)

        results[k] = mse

        if mse < best_mse:
            best_mse = mse
            best_k = k

    return best_k, results

k_values = range(1, 20)
best_k, results = evaluate_knn(X_train, X_test, y_train, y_test, k_values)
print(f"Найкраще значення K: {best_k}")
print(f"Результати для кожного K: {results}")
```

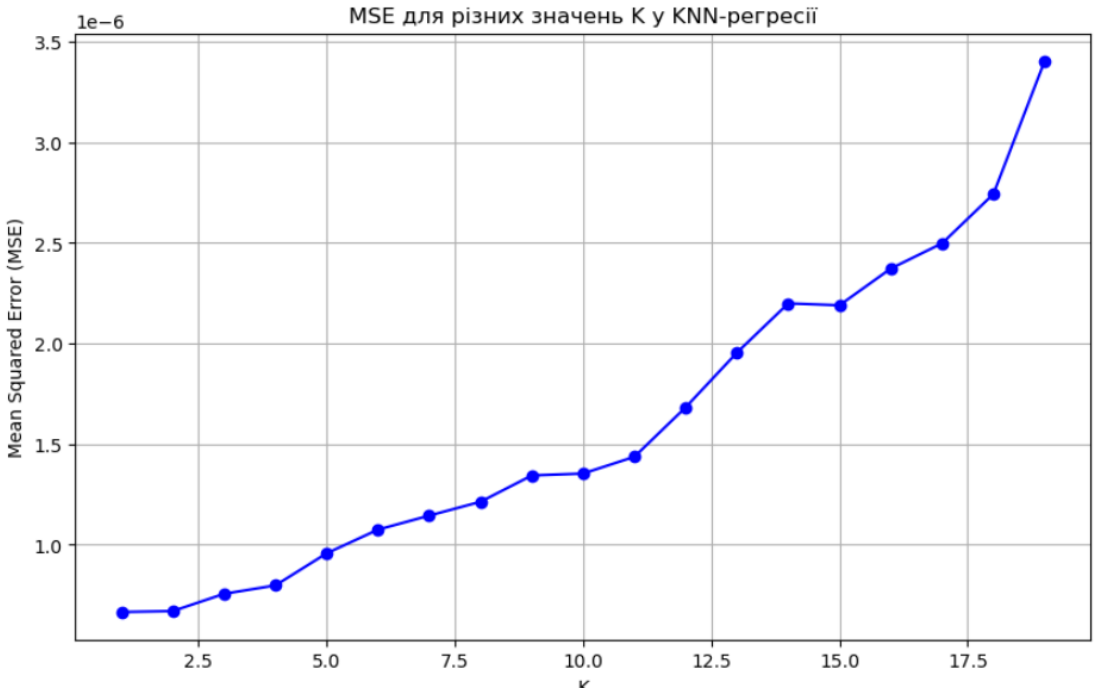
Крок 5: Візуалізація результатів

Побудуємо графік помилки (MSE) для різних значень K, щоб візуально оцінити якість моделі.

```
# Крок 5: Візуалізація результатів
plt.figure(figsize=(10, 6))
plt.plot(list(results.keys()), list(results.values()), marker='o', color='b')
plt.title('MSE для різних значень K у KNN-регресії')
plt.xlabel('K')
plt.ylabel('Mean Squared Error (MSE)')
plt.grid(True)
plt.show()
```

Після запуску нашого коду ми отримуємо наступний результат:

```
Згенеровані дані: [567.51688384 211.21529768 714.48898361 574.97267908 734.70386453
 187.19850117 588.72238562 667.53542215 575.21501683  64.84162643]
Нормалізовані дані: [[0.56747037]
 [0.21065163]
 [0.71465579]
 [0.57493698]
 [0.73490001]
 [0.18659998]
 [0.58870665]
 [0.66763408]
 [0.57517967]
 [0.06406551]]
Розмір навчальної вибірки: 800
Розмір тестової вибірки: 200
Найкраще значення K: 1
Результати для кожного K: {1: 6.652303740982771e-07, 2: 6.699009699004706e-07, 3: 7.556479134795837e-07, 4: 7.974114168449872e-07, 5: 9.571191136255913e-
07, 6: 1.0747634685319368e-06, 7: 1.14454101359057e-06, 8: 1.2133171399349794e-06, 9: 1.3441641779587656e-06, 10: 1.3535669422090282e-06, 11: 1.436960719
7583688e-06, 12: 1.6830600497753708e-06, 13: 1.955019095674708e-06, 14: 2.1995559621132655e-06, 15: 2.190294317567268e-06, 16: 2.372888203348804e-06, 17:
2.498267396049074e-06, 18: 2.7431221946971863e-06, 19: 3.4048816245102463e-06}
```



Генерація випадкових даних

Було згенеровано 1000 випадкових значень у діапазоні [0, 1000]. Ось перші 10 елементів з цього набору даних:

[567.51688384 211.21529768 714.48898361 574.97267908 734.70386453

187.19850117 588.72238562 667.53542215 575.21501683 64.84162643]

Цей набір даних буде використовуватись для подальших операцій.

Нормалізація даних

Для нормалізації використовувалася мінімаксна нормалізація, щоб привести дані в діапазон [0, 1]. Нормалізовані дані представлені нижче:

[[0.56747037]

[0.21065163]

[0.71465579]

[0.57493698]

[0.73490001]

[0.18659998]

[0.58870665]

[0.66763408]

[0.57517967]

[0.06406551]]

Нормалізація важлива для алгоритмів, таких як KNN, оскільки різні шкали можуть впливати на результати моделі.

Розділення на навчальну та тестову вибірки

Дані були розділені на навчальну та тестову вибірки у пропорції 80% для навчання і 20% для тестування:

Розмір вибірок:

- Навчальна вибірка: 800 записів.
- Тестова вибірка: 200 записів.

Цей розподіл забезпечує достатню кількість даних для навчання моделі і її подальшої оцінки.

Навчання KNN-регресора з різними значеннями K

Найкраще значення K:

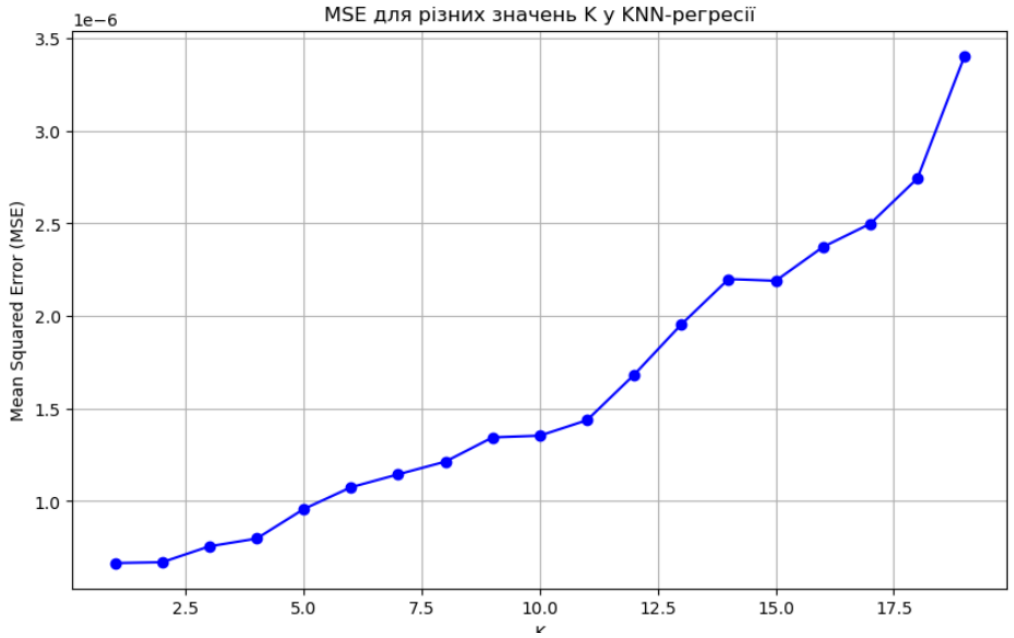
1

Результати для кожного K:

{1: 6.652303740982771e-07, 2: 6.699009699004706e-07, 3: 7.556479134795837e-07, 4: 7.974114168449872e-07, 5: 9.571191136255913e-07, 6: 1.0747634685319368e-06, 7: 1.14454101359057e-06, 8: 1.2133171399349794e-06, 9: 1.3441641779587656e-06, 10: 1.3535669422090282e-06, 11: 1.4369607197583688e-06, 12: 1.6830600497753708e-06, 13: 1.955019095674708e-06, 14: 2.1995559621132655e-06, 15: 2.190294317567268e-06, 16: 2.372888203348804e-06, 17: 2.498267396049074e-06, 18: 2.7431221946971863e-06, 19: 3.4048816245102463e-06}

Візуалізація результатів

На основі результатів навчання побудовано графік, який відображає середньоквадратичну помилку (MSE) для кожного значення K у KNN-регресії.



Опис візуалізації: Графік показує, що при значенні K = 1, модель дає найнижчу MSE, що підтверджує обрання цього значення як найкращого. Зі збільшенням K помилка поступово зростає, що свідчить про те, що більші значення K не забезпечують кращих результатів у даному випадку.

Висновки:

В рамках цієї лабораторної роботи були виконані всі завдання: згенеровано дані, виконана нормалізація, проведено розділення на вибірки, навчена модель KNN-регресії з різними значеннями K, обрано найкраще значення K, і результати були візуалізовані. Значення K = 1 виявилося оптимальним для даного набору даних, забезпечуючи найменшу помилку моделі на тестовій вибірці.

Код програми:

```
import numpy as np

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean_squared_error

import matplotlib.pyplot as plt



# Крок 1: Генерація випадкових даних

data = np.random.rand(1000) * 1000

print("Згенеровані дані:", data[:10])  # Виведемо перші 10 елементів для перевірки



# Крок 2: Нормалізація даних

scaler = MinMaxScaler()
```

```
data_normalized = scaler.fit_transform(data.reshape(-1, 1))

print("Нормалізовані дані:", data_normalized[:10])  # Перевірка перших 10 елементів


# Крок 3: Розділення на навчальну та тестову вибірки

X_train, X_test, y_train, y_test = train_test_split(data_normalized, data_normalized, test_size=0.2, random_state=42)

print(f"Розмір навчальної вибірки: {len(X_train)}")

print(f"Розмір тестової вибірки: {len(X_test)}")


# Крок 4: Навчання KNN-регресора з різними значеннями K

def evaluate_knn(X_train, X_test, y_train, y_test, k_values):

    best_k = None

    best_mse = float('inf')

    results = {}


    for k in k_values:

        knn = KNeighborsRegressor(n_neighbors=k)

        knn.fit(X_train, y_train)


        y_pred = knn.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)


        results[k] = mse


        if mse < best_mse:
```

```
            best_mse = mse
            best_k = k

    return best_k, results

k_values = range(1, 20)
best_k, results = evaluate_knn(X_train, X_test, y_train, y_test, k_values)
print(f"Найкраще значення K: {best_k}")
print(f"Результати для кожного K: {results}")

# Крок 5: Візуалізація результатів
plt.figure(figsize=(10, 6))
plt.plot(list(results.keys()), list(results.values()), marker='o', color='b')
plt.title('MSE для різних значень K у KNN-регресії')
plt.xlabel('K')
plt.ylabel('Mean Squared Error (MSE)')
plt.grid(True)
plt.show()
```