Міністерство освіти та науки України

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет комп’ютерних інформаційних технологій

Кафедра ІОСУ

**Модульна робота №1**

З дисципліни «Обчислювальний інтелект»

**Виконав:**

студент групи КНм-11

Коткевич Юрій

**Перевірив:**

доц.Коваль В.С.

Тернопіль 2024

**Тема:** Дослідження побудови класифікатора та регресора методом k-найближчих сусідів (k-nn).

**Мета:** Вивчити можливості аналізу даних з використанням класифкатора та регресора методом k-найближчих сусідів (k-nn).

**Завдання**

1. Створення KNN – класифікатора у Python Розробити програмну реалізацію Python, яка забезпечує виконання наступних кроків:

• Завантажити базу параметрів квітів iris dataset

• Перемішати записи у завантаженій базі

• Нормалізувати параметри квітів ірису

• Розділити існуючі записи на навчальну і тестові вибірки

• Навчити KNN-класифікатор з різними значеннями K

• Вибрати величину K для найкращих показників якості класифікацій у тестовій вибірці

2. Створення KNN – регресора у Python Розробити програмну реалізацію Python, 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 K

• Вибрати величину K для найкращих показників якості регресії у тестовій вибірці

• Здійснити візуалізації отриманих рішень

**Хід роботи:**

**Створення KNN – класифікатора у Python**

**Код програми:**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.preprocessing import StandardScaler

# Loading data

iris = load\_iris()

# Shuffling records

np.random.seed(2021)

np.random.shuffle(iris.data)

np.random.shuffle(iris.target)

# Converting data to a DataFrame

df\_iris = pd.DataFrame(iris.data, columns=iris.feature\_names)

df\_iris['label'] = iris.target

# Saving the label names

features\_dict = {k: v for k, v in enumerate(iris.target\_names)}

df\_iris['label\_names'] = df\_iris.label.apply(lambda x: features\_dict[x])

# Data normalization

scaler = StandardScaler()

# Selecting numerical features (assuming that columns 0 to 3 are numerical)

numerical\_features = df\_iris.iloc[:, :4]

df\_iris\_norm = pd.DataFrame(scaler.fit\_transform(numerical\_features), columns = numerical\_features.columns)

# Combine the normalized features with the original "label" column

df\_iris\_norm['label'] = df\_iris['label']

# Splitting the data into 70% training and 30% test data

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df\_iris\_norm.drop('label', axis=1), df\_iris\_norm['label'], test\_size=0.3, random\_state=2021)

# Training a KNN classifier with different values of K

from sklearn.neighbors import KNeighborsClassifier

# Defining the range of K values

k\_range = list(range(1, 31))

# Creating a dictionary to store the results

scores = {}

for k in k\_range:

  # Training the KNN classifier

  knn = KNeighborsClassifier(n\_neighbors=k)

  knn.fit(X\_train, y\_train)

  # Evaluation of accuracy on the test sample

  score = knn.score(X\_test, y\_test)

  # Storing the results

  scores[k] = score

# Search for the best value of K

k\_best = max(scores, key=scores.get)

score\_best = scores[k\_best]

print('Best k value:', k\_best)

print('Classification accuracy with the best k:', score\_best)

# Visualize the results

import matplotlib.pyplot as plt

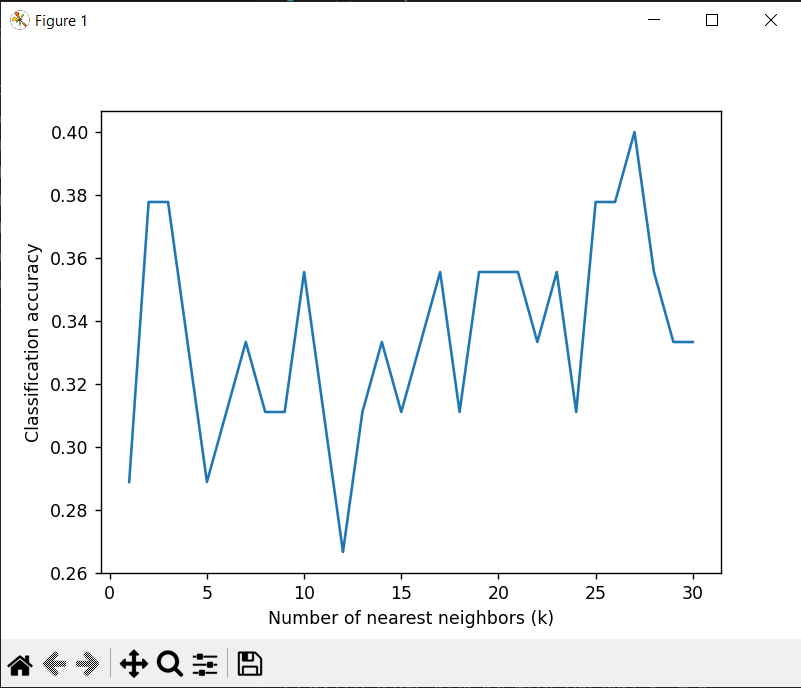
plt.plot(k\_range, scores.values())

plt.xlabel('Number of nearest neighbors (k)')

plt.ylabel('Classification accuracy')

plt.show()

**Результат:**

****

****

**Створення KNN – регресора у Python**

**Код програми:**

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsRegressor

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap

# Generate a random data set

np.random.seed(2021)

X = np.random.rand(1000, 2)

# Defining a color map

cmap\_bold = ListedColormap(['blue','#FFFF00','black','green'])

# Data normalization

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split into training and test samples

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, np.arange(1000), test\_size=0.25, random\_state=42)

# Training a KNN regressor with different values of K

k\_values = list(range(1, 21))

scores = []

for k in k\_values:

  knn = KNeighborsRegressor(n\_neighbors=k)

  knn.fit(X\_train, y\_train)

  score = knn.score(X\_test, y\_test)

  scores.append(score)

# Selecting the value of K for the best performance

best\_k = k\_values[scores.index(max(scores))]

# Estimation of KNN regressor with optimal K

best\_score = scores[best\_k - 1]

# Visualization of results

plt.plot(k\_values, scores)

plt.xlabel("Кількість найближчих сусідів (K)")

plt.ylabel("Точність регресії")

plt.show()

# Prediction using a KNN regressor with optimal K

knn\_best = KNeighborsRegressor(n\_neighbors=best\_k)

knn\_best.fit(X\_train, y\_train)

y\_pred = knn\_best.predict(X\_test)

# Visualization of forecasts

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_test, marker='o', s=30, cmap='viridis')

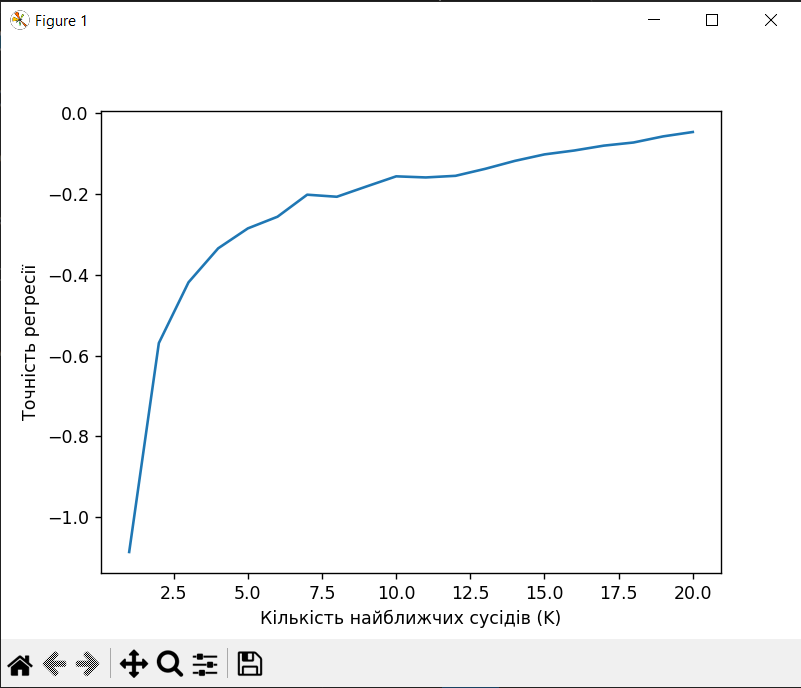
plt.plot(X\_test[:, 0], y\_pred, color='red')

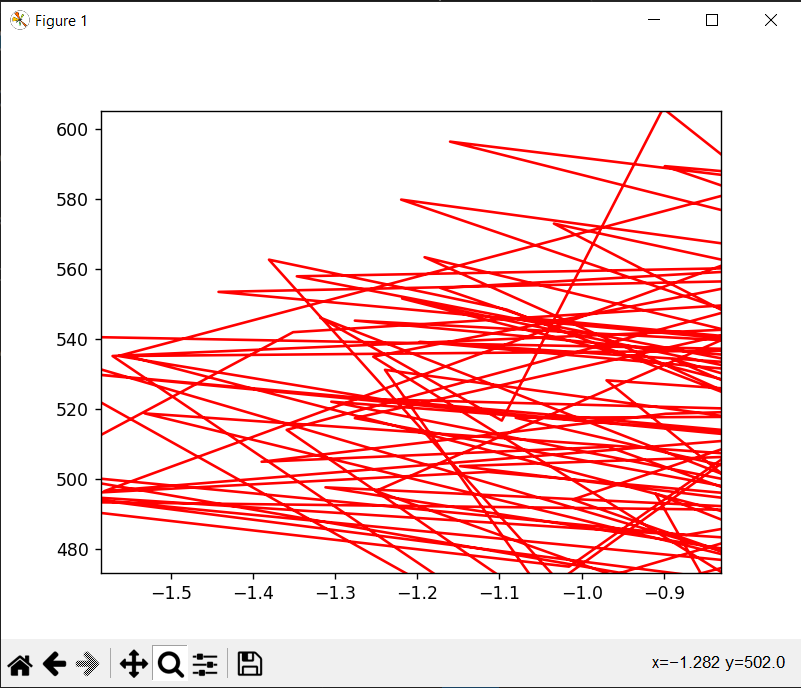
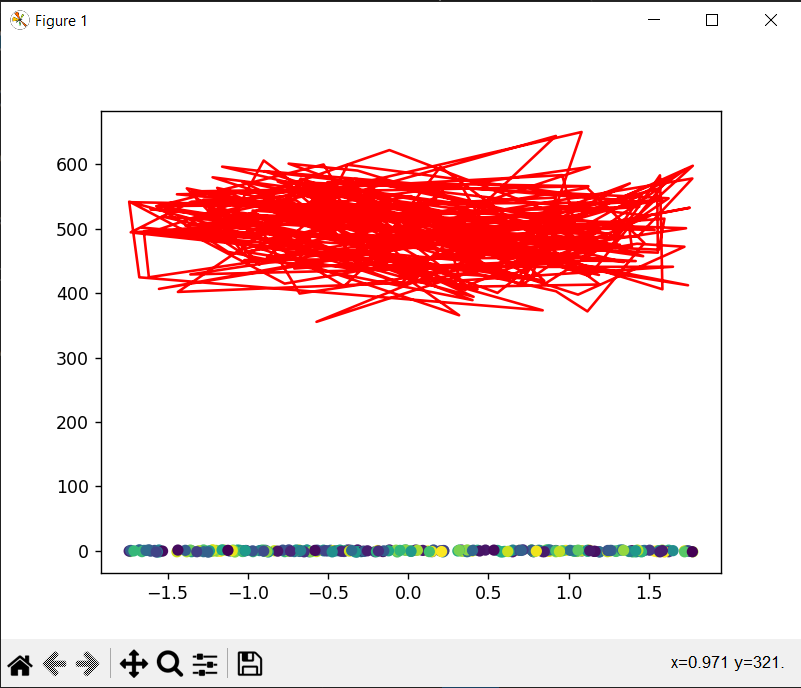
plt.show()

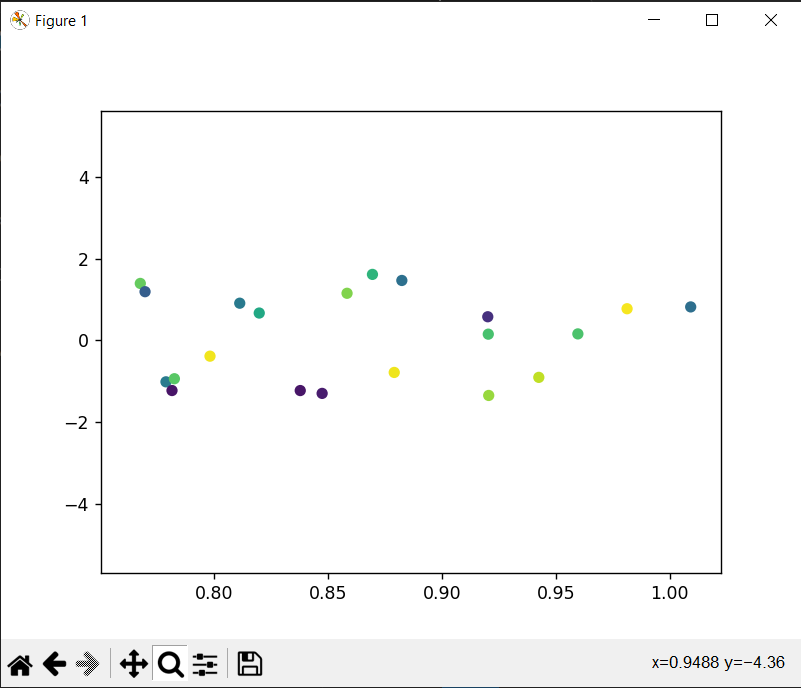
print(f"The best value of K: {best\_k}")

print(f"The accuracy of the regression with K={best\_k}: {best\_score:.2%}")

**Результат:**

****

****

****

****

**Висновок**:

У ході даної модульної роботи було досліджено KNN-алгоритми класифікації та регресії. Розроблено програмні реалізації KNN-класифікатора та KNN-регресора, а саме: завантажено та перемішано записи датасету Iris, згенерувано вибірковий набір даних з 1000 значень, нормалізовано значення, розділено записи на навчальну та тестові вибірки, навчено KNN-класифікатор та KNN-регресор з різними значеннями K, вибрано значення K з найкращою точністю для класифікації та регресії, здійснено візуалізацію отриманих рішень для регресора.