LAB REPORT 1

November 9, 2020

Submitted by:

Kazi Amit Hasan
Roll: 1503089
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

November 9, 2020

0.1 Title

Implementation of nearest neighbor algorithm along with it's performance analysis

0.2 Key Objectives

1. Understanding nearest neighbors.
2. Easy to implement, thus great choice to start with to study more complex methods.
3. Understanding different distance measures.
4. Importance of k value.

0.3 Methodology

Nearest neighbor algorithm is a pretty simple algorithm. It finds out the nearest neighbor of a given point and concludes that the class in which the nearest neighbor belongs to is the desired class of the unknown pattern also. Now, the question is how to determine the distance to find out the nearest neighbor? There are basically many distance measure algorithms. One of them is euclidean distance. The most common way to find this distance is the Euclidean distance, as shown below.

$$d(x, x') = \sqrt{\left(x_1 - x_1'\right)^2 + + \left(x_n - x_n'\right)^2}$$

Nearest neighbor methods pose the problem of finding a reliable way of measuring the distance from one class sample to another. Obviously, we need to specify a distance metric that will allow us to measure the similarity of pattern samples in geometric pattern space. In practice, several methods are used.

Basically, In this lab what we want do is that we try to find the k nearest neighbor and do a majority voting. Typically k is odd when the number of classes is 2. Let's say k = 5 and there are 3 instances of C1 and 2 instances of C2. In this case, KNN says that a new point has to labeled as C1 as it forms the majority. We follow a similar argument when there are multiple classes.

0.4 Implementation

The tools that I used for the implementation purpose of this lab are Jupyter notebook. The codes are written in Python language.

The algorithm that is implemented in the lab is given below:

1. **Importing all the required libraries**
2. **Implementing the knn function**
3. **Generate sample data points**
4. **Fit the model**
5. **Get the results**

0.5 Code

```
# Importing neccesary libraries

import numpy as np
from matplotlib import pyplot as plt

# The main function

class KNearestNeighbor(object):
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        # Store the original points

        self.X = X
        self.y = y

        return self
    def predict(self, X, y=None):
         print("self.X:", self.X)
         print("self.X.shape:", self.X.shape)

        # Initialize a zero distance matrix
        dists = np.zeros((X.shape[0], self.X.shape[0]))
         print("dists.shape:", dists.shape)

        # Loop through all possible pairs and compute their distances
        for i in range(dists.shape[0]):
            for j in range(dists.shape[1]):
                 print(i, j, X[i], self.X[j])
                dists[i, j] = self.distance(X[i], self.X[j])

         print("dists:", dists)

        # Sort the distance array row-wise,
        # and select the top k indexes for each row

        indexes = np.argsort(dists, axis=1)[:,:self.k]
         print("indexes:", indexes)

        # Compute the mean of the values
        mean = np.mean(self.y[indexes], axis=1)
         print("mean:", mean)
        print("mean.shape:", mean.shape)

        return mean

    def distance(self, x, y):
        return np.sqrt(np.dot(x - y, x - y))


# Generate sample data
```

```
x = np.linspace(0, 5, 20)
m = 1.5
c = 1
y = m * x + c + np.random.normal(size=(20,))
plt.plot(x, y, 'x')

# Fit the model

model = KNearestNeighbor(k=3)
model.fit(x, y)

# Getting results

predicted = model.predict(x.reshape(-1, 1))
plt.plot(
    x, y, "x",
    x, model.predict(x), "-o"
)
plt.legend(["actual", "prediction"])
```

0.6 Results

I generated some sample data points shown in Fig. 1. In order to show the results, I plotted a figure of containing actual and predicted values in Fig. 2.

0.7 Performance Analysis

In order to performance analysis, I tested my code with various values of 'k'. K plays an important role in knn algorithm.

The Fig. 3 is representing the results obtained with k = 10. Similarly, the Fig. 4 is representing the results obtained with k=20.

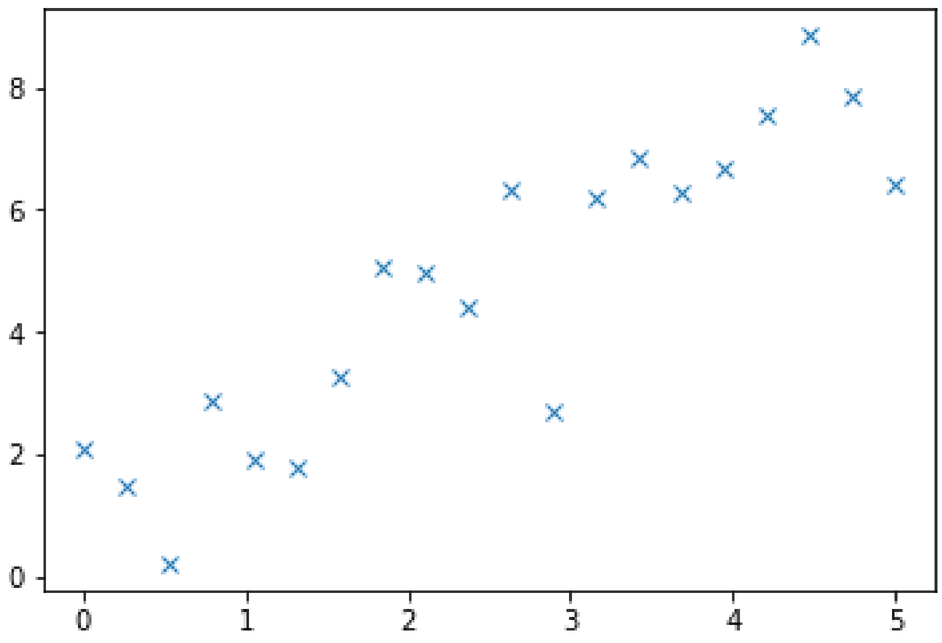

Figure 1: Generated data points

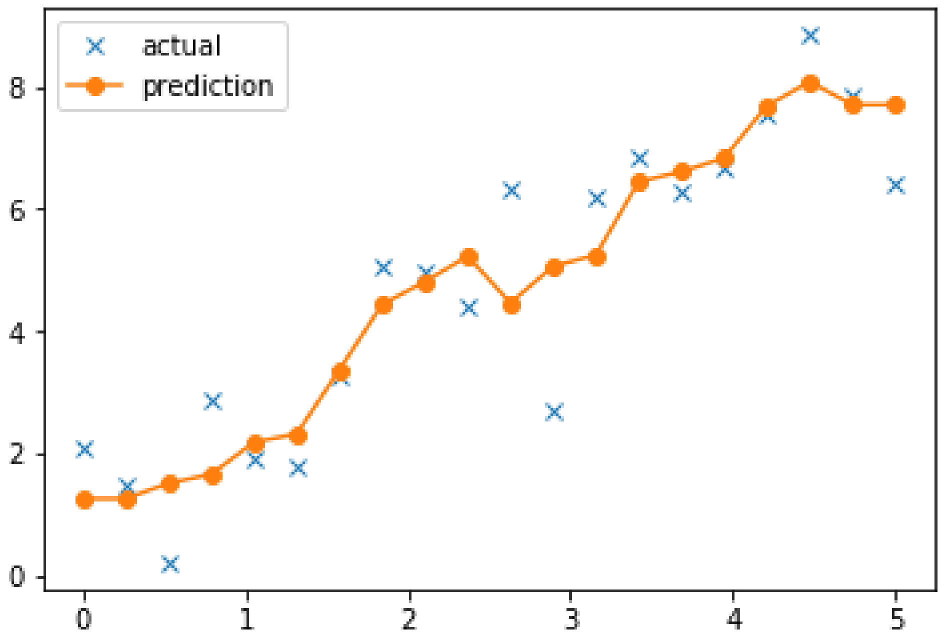

Figure 2: Results by using knn algorithm

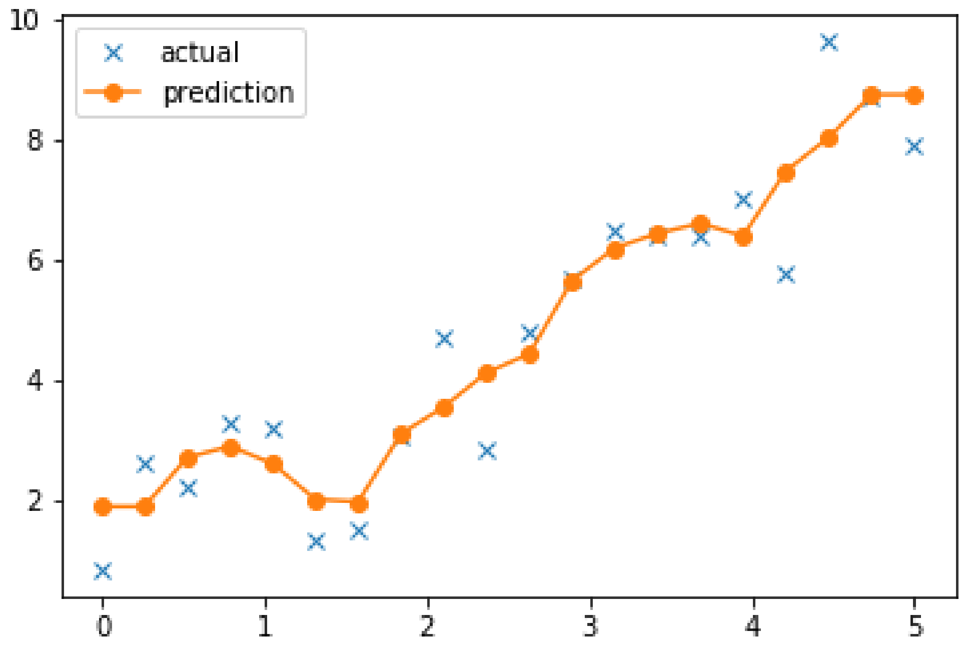

Figure 3: Obtained results using KNN while k=10

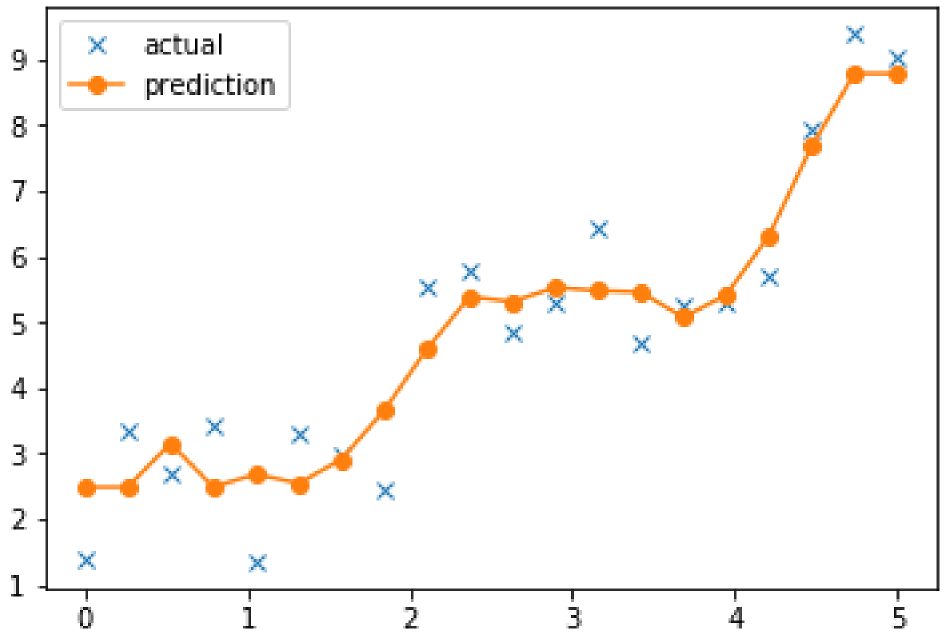

Figure 4: Obtained results using KNN while k=20

0.8 Conclusion

KNN is a basic algorithm which is commonly used in different machine learning problems. The KNN holds it's importance because of it's simplicity and understandable ability. All the codes and neccesary files are available in my github profile. The codes will be publicly available after my finals grades. My Github profile: https://github.com/AmitHasanShuvo/