**Computer Engineering Department National University of Technology Islamabad, Pakistan**

**Introduction to Data Mining**

**Practice Exercise 12**

Name: Muhammad Sami Uddin Rafay

Roll Number: F18604013

Submitted To: Dr. Kamran Javed

Date: 05 March 2020

**Practice Exercise 12**

**KNN for Regression**

**Objective:**

* Implement KNN for Regression

**Equipment/Software Required:**

* Python (Spyder 4.0 Anaconda Distribution)

**Background:**

KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

**Tasks:**

**Code:**

**# importing necessarily Libraries**

import mglearn

import matplotlib.pyplot as plt

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neighbors import KNeighborsRegressor

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

mglearn.plots.plot\_knn\_regression(n\_neighbors=3)

mglearn.plots.plot\_knn\_regression(n\_neighbors=9)

**# split the wave dataset into a training and a test set**

X, y = mglearn.datasets.make\_wave(n\_samples=40)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

**# instantiate the model and set the number of neighbors to consider to 3**

reg = KNeighborsRegressor(n\_neighbors=3)

**# fit the model using the training data and training targets**

reg.fit(X\_train, y\_train)

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

**# create 1,000 data points, evenly spaced between -3 and 3**

line = np.linspace(-3, 3, 1000).reshape(-1, 1)

for n\_neighbors, ax in zip([1, 3, 9], axes):

**# make predictions using 1, 3, or 9 neighbors**

reg = KNeighborsRegressor(n\_neighbors=n\_neighbors)

reg.fit(X\_train, y\_train)

ax.plot(line, reg.predict(line))

ax.plot(X\_train, y\_train, '^', c=mglearn.cm2(0),

markersize=8)

ax.plot(X\_test, y\_test, 'v', c=mglearn.cm2(1), markersize=8)

ax.set\_title("{} neighbor(s)\n train score: {:.2f} test score: {:.2f}".format(n\_neighbors,

reg.score(X\_train, y\_train),reg.score(X\_test,

y\_test)))

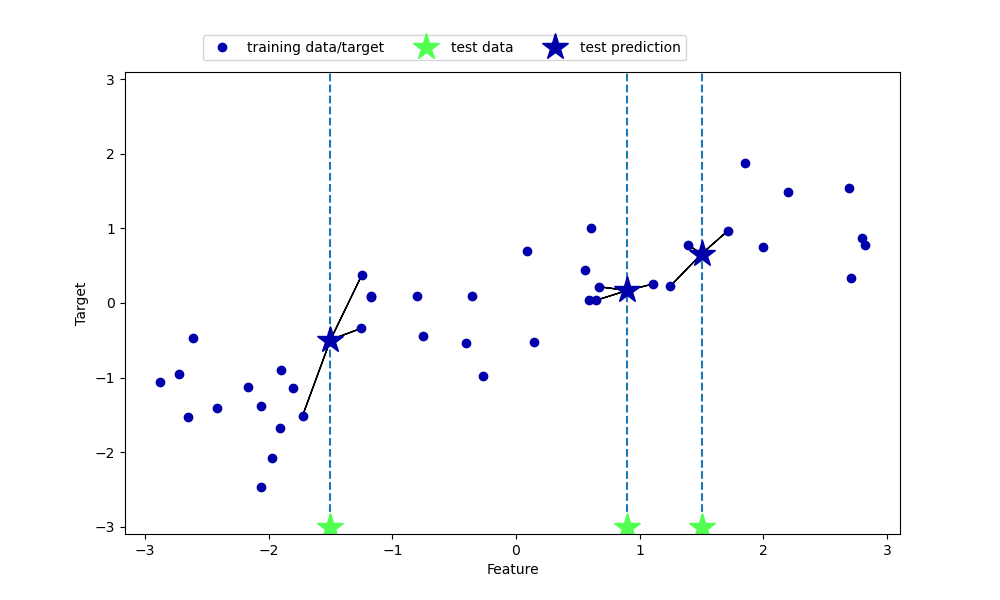
ax.set\_xlabel("Feature")

ax.set\_ylabel("Target")

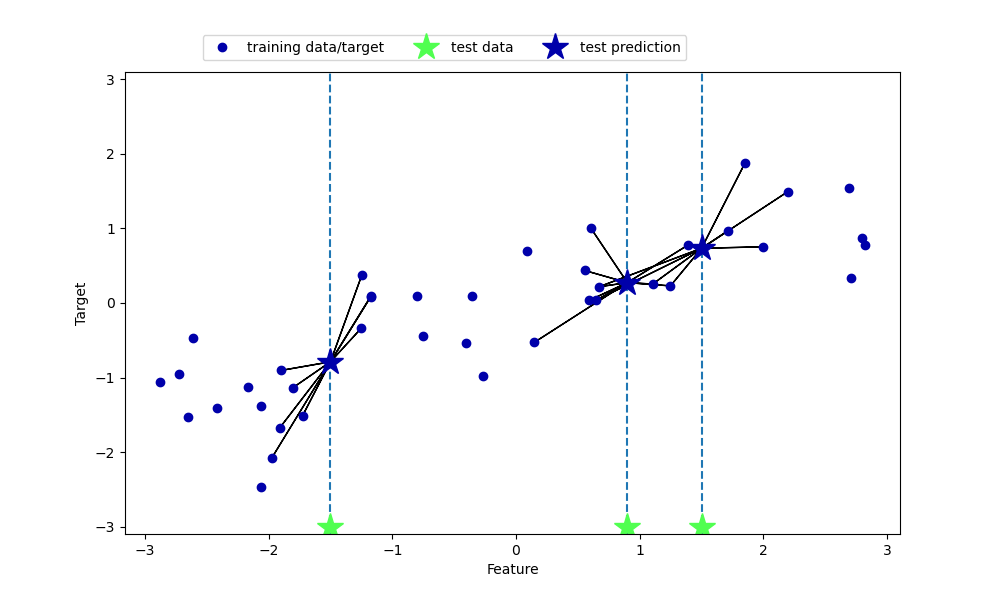
axes[0].legend(["Model predictions", "Training data/target","Test data/target"], loc="best")

**Output:**

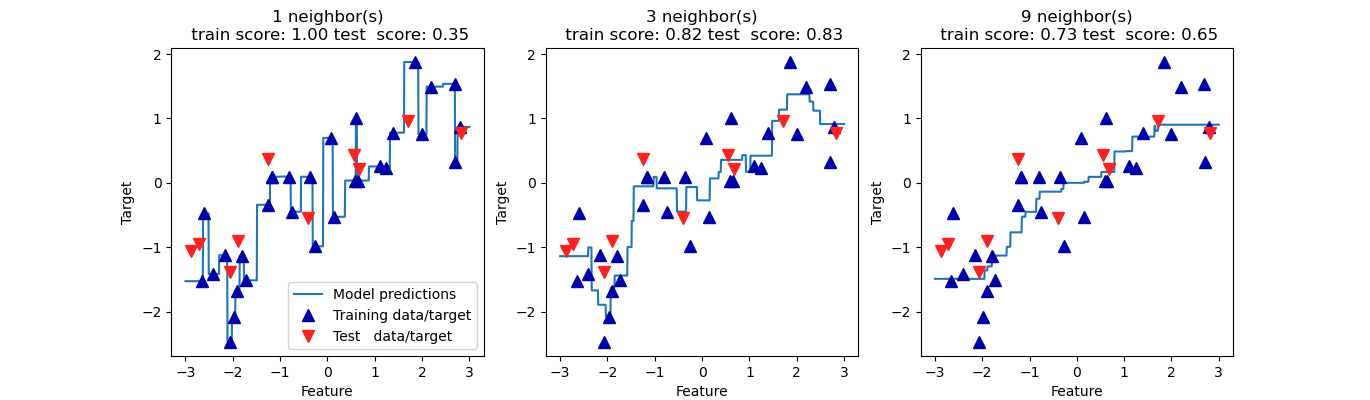
For 3 Neighbors

****

For 9 Neighbors

****

Here we can see that the model with one neighbor is overfitted, the model with nine neighbor is underfitted and the model with three neighbor is best fitted.

****

**Reference:**

[**https://medium.com/analytics-vidhya/k-neighbors-regression-analysis-in-python-61532d56d8e4**](https://medium.com/analytics-vidhya/k-neighbors-regression-analysis-in-python-61532d56d8e4)

The basic purpose of this exercise is implementing the Regression with KNN approach so I used the example and code is given in above link. This example suits the best for Regression using KNN.

**Note:** This is not my own generated code, I just dry run this code and understand the proof of concept.

**Results and Discussions:**

In this Exercise, I have learned how to do regression analysis with KNN approach.

**Conclusion:**

Regression analysis is a statistical method that helps us to analyse and understand the relationship between two or more variables of interest. The process that is adapted to perform regression analysis helps to understand which factors are important, which factors can be ignored and how they are influencing each other.