s

**Prediction Diabetics Using KNN from Diabetics Dataset on Kaggle**

**INTRODUCTION:**

KNN **(K-Nearest Neighbor)** is one of the most popular Machine Learning Algorithm based on the supervised dataset. This technique is followed by selecting the number of K of the neighbors. After selecting the neighbors calculate the Euclidean distance. Among these k neighbors, counted the number of the data points in each category. Assign the new data points to that category for which the number of the neighbor is maximum. Then the model is ready. In the project there were trying to build a model for predicting the diabetics patient from a dataset on Kaggle using KNN- algorithm and R language.

**METHODOLOGY:**

The main data was found from Kaggle (link- <https://www.kaggle.com/datasets/mathchi/diabetes-data-set>). Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

* Pregnancies: Number of times pregnant
* Glucose: Plasma glucose concentration 2 hours in an oral glucose tolerance test
* Blood Pressure: Diastolic blood pressure (mm Hg)
* Skin Thickness: Triceps skin fold thickness (mm)
* Insulin: 2-Hour serum insulin (mu U/ml)
* BMI: Body mass index (weight in kg/ (height in m) ^2)
* Diabetes Pedigree Function: Diabetes pedigree function
* Age: Age (years)
* Outcome: Class variable (0 or 1)

**SOURCES:**

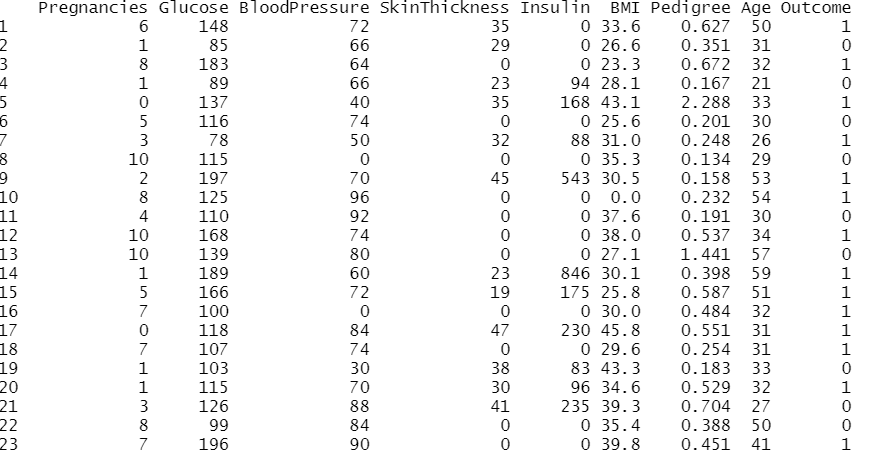
1. Original owners: National Institute of Diabetes and Digestive and  
   kidney diseases  
   (b) Donor of database: Vincent Sigillito (vgs@aplcen.apl.jhu.edu)  
   Research Center, RMI Group Leader  
   Applied Physics Laboratory  
   The Johns Hopkins University  
   Johns Hopkins Road  
   Laurel, MD 20707  
   (301) 953-6231  
   (c) Date received: 9 May 1990

**CODE AND THE STEPS OF THE PROJECTS:**

* **IMPORT DATA:**

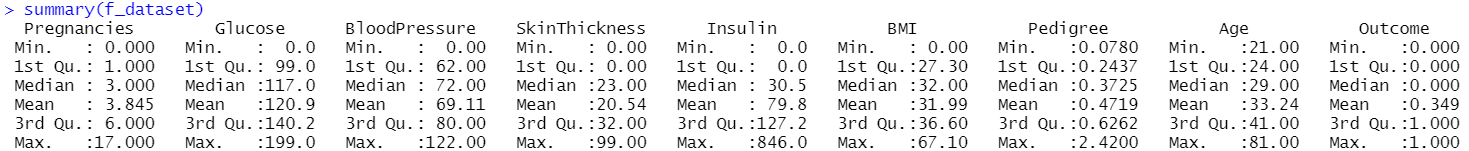
****

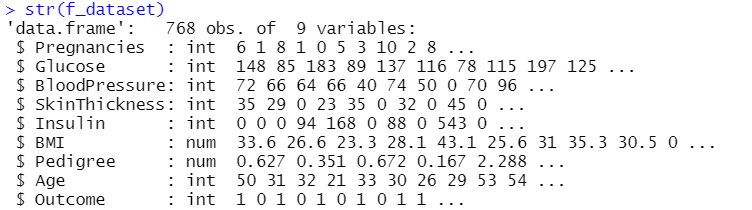
* **OUTPUT:**

****

* **OBSERVE THE DATA SET:**

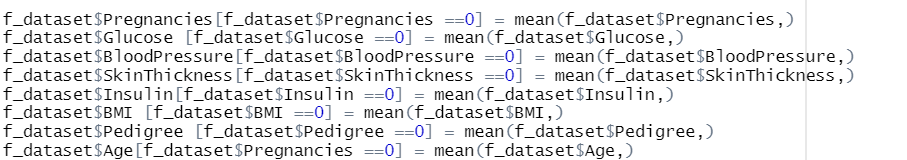
****

* **OUTPUT:**

****

* **Replace by mean value where is the attributes value is zero:**

It may cause problem in KNN-algorithm to deal with the zero value and find the accurate distance.



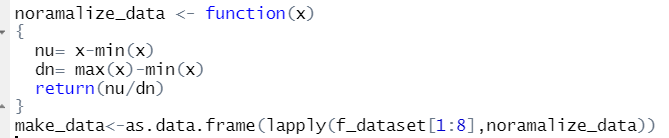
* **Load the library of class for applying the KNN- algorithm:**



* **Normalize the data:**

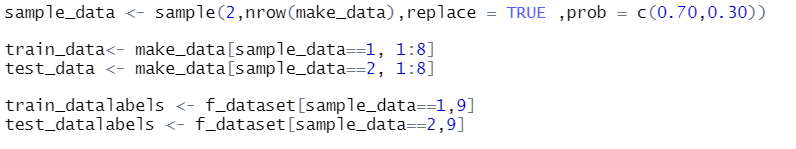
Normalization is a very important part of KNN. It is hard to make the current accuracy of is the data is not in well-shaped. This is because the distance calculation done in KNN uses feature values. When the one feature values are large than other, that feature will dominate the distance hence the outcome of the KNN. It shapes the data in 0 to 1. The main math of the normalization is

= (value-min(value))/(max(value)-min(value))

****

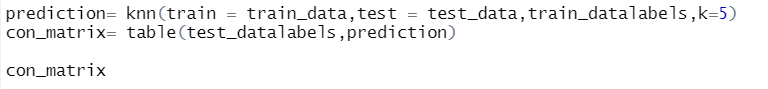
* **Split Data into Test and Train**

There ware taken 70% data in train dataset and 30% data on the test dataset for making the confusion matrix so randomly. After that labels the data with the “Outcome” attribute.

****

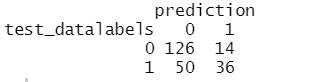
* **Confusion Matrix:**

A confusion matrix is a table that is often used to describe the performance of a classification model or "classifier" on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Here, 1st make prediction using the KNN function. After that, the prediction data was compare with the actual data and make a confusion matrix bellow.

****

* **Output:**

Here, True Negative =126, True Positive=36, False positive =14, False Negative= 50

****

|  |  |  |  |
| --- | --- | --- | --- |
| 0=negative 1=positive | | prediction | |
| 0 | 1 |
| Accentual  result | 0 | 126(TN) | 14(FP) |
| 1 | 50(FN) | 36(TP) |

true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.

true negatives (TN): We predicted no, and they don't have the disease.

false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")

false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Total = 126+14+36+50 = 226

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

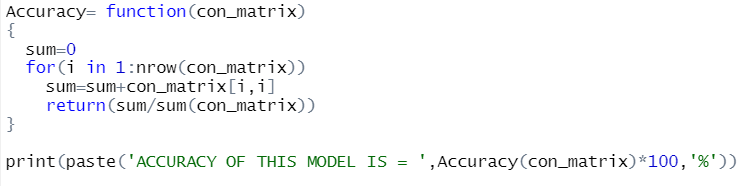
Accuracy: Overall, how often is the classifier correct?

(TP+TN)/total = (36+126)/226 = 0.7168

Misclassification Rate: Overall, how often is it wrong?

(FP+FN)/total = (14+50)/226 = 0.283

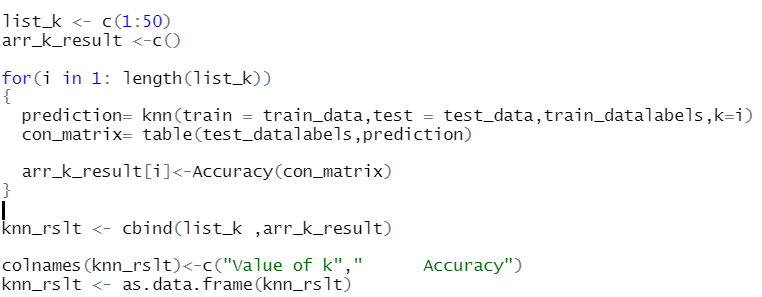
* **Find some Accuracy:**

****

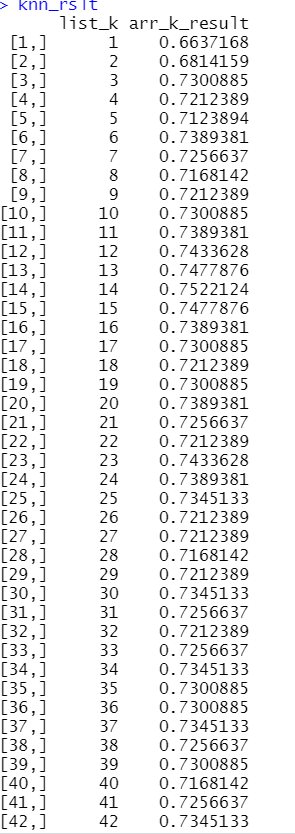
* **Output:**

****

* **Find accuracy based on different K values 1 to 50:**

****

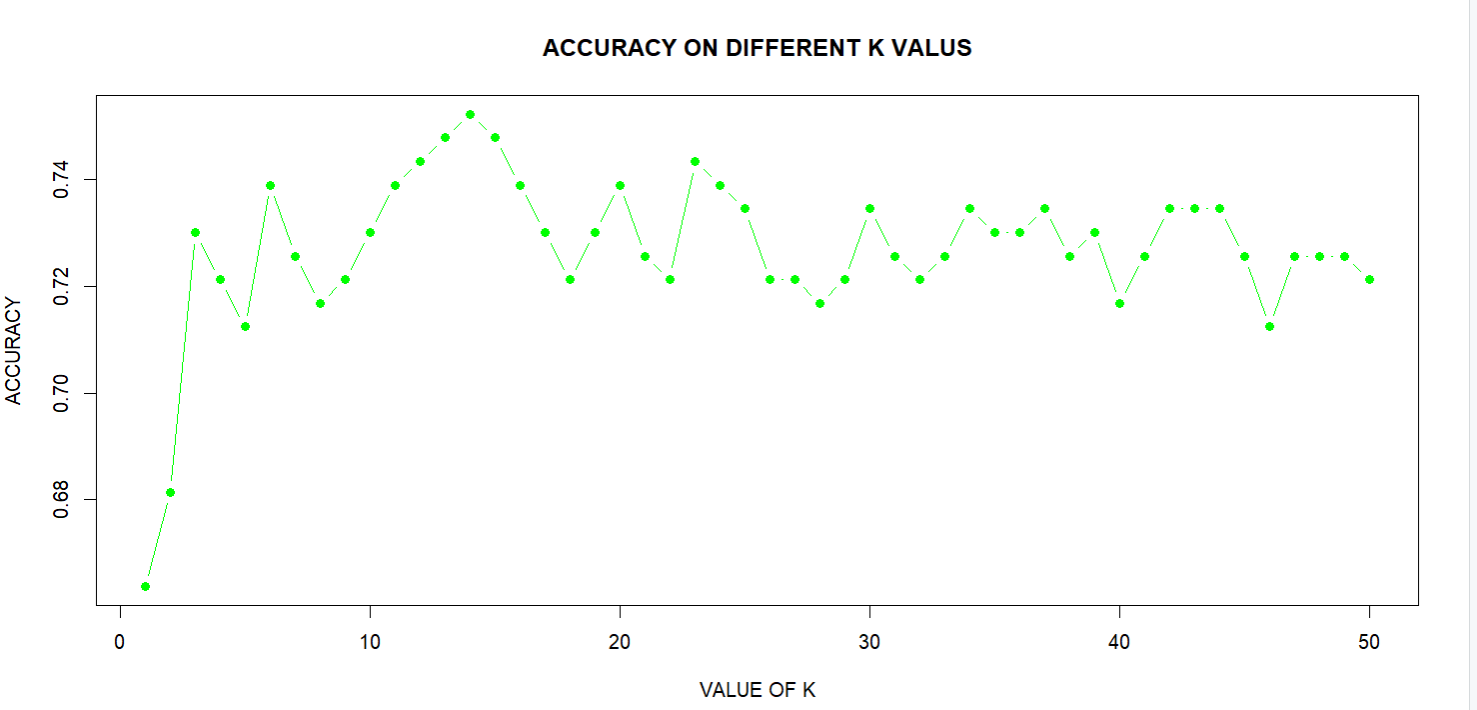
* **Output:**

****

* **Scatter plot of all accuracy with the K value:**

****

* **Output:**

****

Here we can see, there are found maximum accuracy was in between 10 to 20 values of K.

**DISCUSSION:**

There was total 768 data in a dataset. And there predicted the data by 70% train and 30% test dataset selected randomly. In this case if there were more data as well as more attributes given then the prediction will be more meaningful and accurate.