KNN OF IRIS DATA SET

Here, we try to create a KNN Model on the Iris Flowers's Data set in R (using Google Colaab)

First we run the under mentioned programme to run R programms in Google Colaab.

```
%load_ext rpy2.ipython
```

```
The rpy2.ipython extension is already loaded. To reload it, use:
  %reload_ext rpy2.ipython
```

INSTALL THE REQUIRED PACKAGES FOR THIS PROJECT

Here we need the following packages:

1. gt
2. class
3. caret
4. GGally

```
%%R
install.packages(c("gt","class","caret","GGally"))
```

```
R[write to console]:

R[write to console]: downloaded 1.6 MB

R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/prodlim_2019.11

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 126048 bytes (123 KB)
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

IMPORTING LIBRARIES AND DATA SET

Then we Import the given data set dataset.csv into this project.

```
%%R
```

```
%%R
library(gt)
library(class)
library(caret)
library(GGally)
iris_ds<-read.csv('dataset.csv',stringsAsFactors = FALSE)
```

OVERLOOK OF OUR DATA SET

Here we can observe our data set, which have 150 observations with 6 variables, with target variable "Species".

```
%%R
head(iris_ds)
```

```
  Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm     Species
1  1           5.1          3.5           1.4          0.2 Iris-setosa
2  2           4.9          3.0           1.4          0.2 Iris-setosa
3  3           4.7          3.2           1.3          0.2 Iris-setosa
4  4           4.6          3.1           1.5          0.2 Iris-setosa
5  5           5.0          3.6           1.4          0.2 Iris-setosa
6  6           5.4          3.9           1.7          0.4 Iris-setosa
```

```
%%R
str(iris_ds)
```

```
'data.frame':	150 obs. of  6 variables:
 $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ SepalLengthCm: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ SepalWidthCm : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ PetalLengthCm: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ PetalWidthCm : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : chr  "Iris-setosa" "Iris-setosa" "Iris-setosa" "Iris-setosa" ...
```

CHECKING IS THERE ANY MISSING VALUE?

In the next step, we check if there are any missing values in the dataset or not.

```
%%R
sum(is.na(iris_ds))
```

```
[1] 0
```

ANALYZING THE DATA SET

Now we Start the Analysis Stage.

Well first we normalize the data set using Z-scores so that there are no biases in the data due to difference in location and scale:

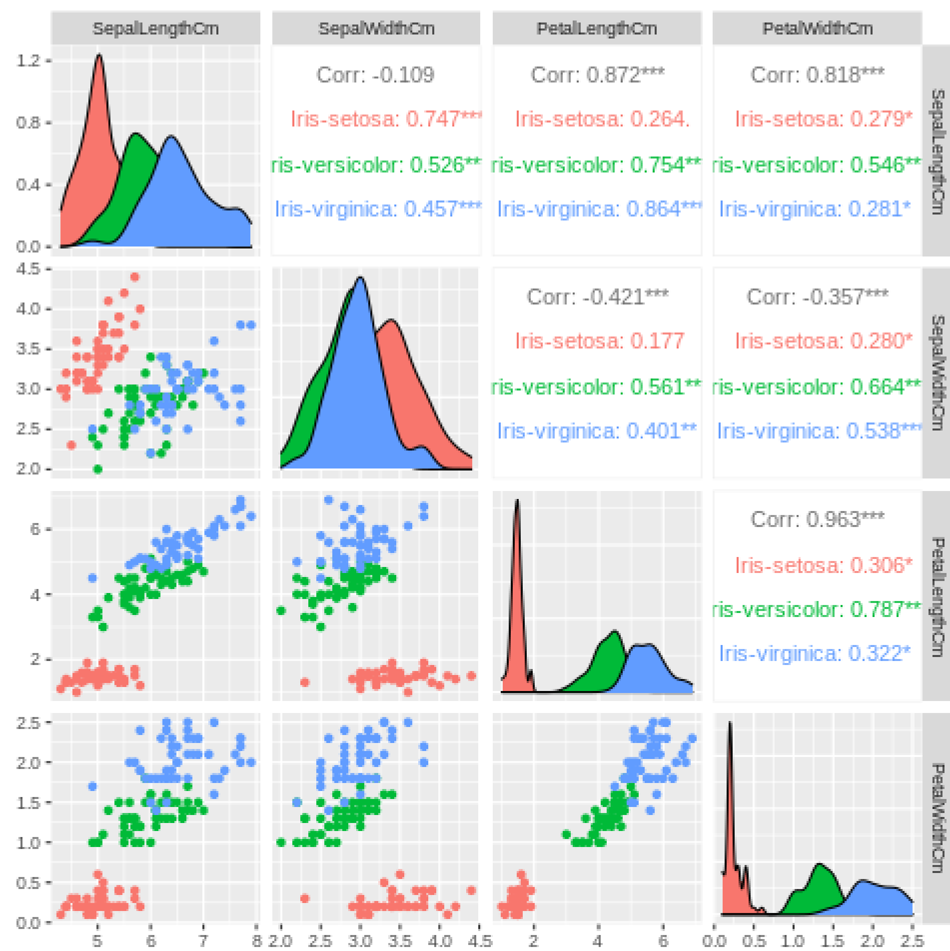
```
%%R
normlz <-function(x) {return(x-as.numeric(mean(x)))/as.numeric(sd(x)) }
iris_norm<-as.data.frame(lapply(iris_ds[,2:5], normlz))
summary(iris_norm)
```

```
 SepalLengthCm        SepalWidthCm     PetalLengthCm      PetalWidthCm
 Min.   :-1.54333   Min.   :-1.054   Min.   :-2.7587   Min.   :-1.0987
 1st Qu.:-0.74333   1st Qu.:-0.254   1st Qu.:-2.1587   1st Qu.:-0.8987
 Median :-0.04333   Median :-0.054   Median : 0.5913   Median : 0.1013
 Mean   : 0.00000   Mean   : 0.000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.55667   3rd Qu.: 0.246   3rd Qu.: 1.3413   3rd Qu.: 0.6013
 Max.   : 2.05667   Max.   : 1.346   Max.   : 3.1413   Max.   : 1.3013
```

DIAGRAMS

Here these plots can easily describe the associationship (using correlation) between the variables.

```
%%R
ggpairs(iris_ds,columns=2:5,mapping =aes(color=Species))
```



SPLITTING THE DATA SET INTO TRAINING AND TEST DATA SET

Here we split the total observations into 70%-30% as Training and Testing data set.

Total observations is 150, so the 1st 105 (70% of 150) observations are selected for training purpose, and the rest are for testing purpose.

```
%%R
iris_train<-iris_norm[1:105,1:4]
iris_test<-iris_norm[106:150,1:4]
iris_train_labels<-as.array(iris_ds[1:105,6])
iris_test_labels<-as.array(iris_ds[106:150,6])
```

Finding the Optimal Number Of Neighbours

Here we create a loop to find the Percentage of Accuracy for each k from 1 to 15:

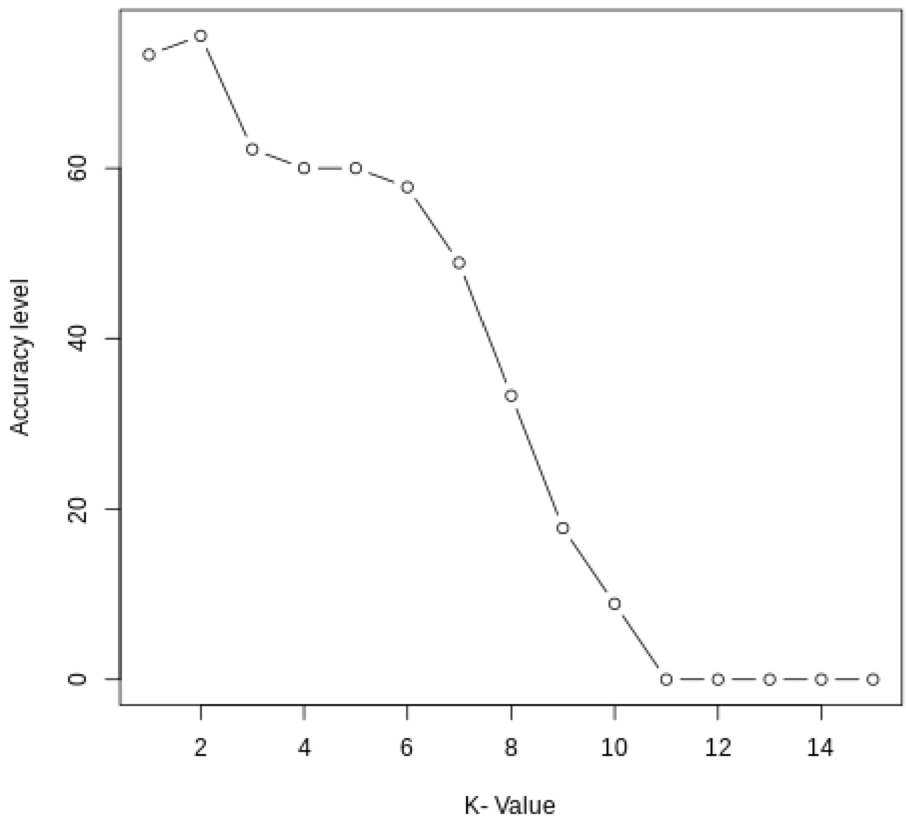
```
%%R
set.seed(123)
i=1
k.optm=1
for (i in 1:15)
{
knn.pred <- knn(train=iris_train, test=iris_test, cl=iris_train_labels, k=i)
k.optm[i] <- 100 * sum(iris_test_labels == knn.pred)/NROW(iris_test_labels)
k=i
cat(k,'=',k.optm[i],'
')
}
```

```
1 = 73.33333
2 = 75.55556
3 = 62.22222
4 = 60
5 = 60
6 = 57.77778
7 = 48.88889
8 = 33.33333
9 = 17.77778
10 = 8.888889
11 = 0
12 = 0
13 = 0
14 = 0
15 = 0
```

Accuracy vs. k graph

Then we draw the accuracy plot and determine the value of k for which we have the highest .

```
%%R
acc_plot<-plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```



Thus, from the graph we see that the model has the best accuracy of 75.56% for K=2 neighbours.

```
**THANK YOU**
```

✓ 0s completed at 9:49 PM