**Prepare a model for glass classification using KNN**

**Logic:** This problem statement can be solved using the KNN algorithm that will classify the

Glass classification.

**Algorithm:**

* Calculate the distance from x to all points in Glass data.
* Sort the points in Glass data by increasing distance from x.
* Predict the majority label of the k closest points.

Note that the value of k effects the results, its ideal to test the model for different values of k for better results and there by a better model.

**Data**

**Glass Identification Database**  contains 9 attributes. The response is glass type which has 7 values.

**Import the dataset**

After importing the dataset, let’s take a look at the structure of the dataset:

The glass data set contain 214 observations and 10 variables

Variables are

* **RI**: refractive index
* **Na**: Sodium
* **Mg**: Magnesium
* **Al**: Aluminum
* **Si**: Silicon
* **K**: Potassium
* **Ca**: Calcium
* **Ba**: Barium
* **Fe**: Iron

Note that, the ‘Type’ variable is our output variable or the target variable. The value of the Type variable represents glass classification.

**Data Normalization**

Always normalize the data set so that the output remains unbiased. To explain this, let’s take a look at the first few observations in our data set.

> head(glass)

RI Na Mg Al Si K Ca Ba Fe Type

1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75 0 0.00 1

2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83 0 0.00 1

3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78 0 0.00 1

4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22 0 0.00 1

5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07 0 0.00 1

6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07 0 0.26 1

Notice the ‘Na’ variable, its value scale is in 2 digits, whereas the rest of the variables are in single digits or 2 digits. If the data isn’t normalized it will lead to a baised outcome.

**# create normalization function**

normalize <- function(x) {

return ((x - min(x)) / (max(x) - min(x)))

}

In the below code , we’re storing the normalized data set in the ‘glass\_n’ variable and also we’re removing the ‘Type’ variable since it’s the response variable that needs to be predicted.

glass\_n <- as.data.frame(lapply(glass[1:9], normalize))

**Data Splicing**

After cleaning the data set and formatting it, the next step is data splicing. Data splicing basically involves splitting the data set into training and testing data set.

**#random sampling**

n <- nrow(glass\_n)

n1 <- n\*0.8

n2 <- n-n1

train\_index <- sample(1:n,n1)

glass\_train <- glass[train\_index, ]

glass\_test <- glass[-train\_index, ]

After deriving the training and testing data set, the below code is going to create a separate data frame for the ‘Type’ variable so that our final outcome can be compared with the actual value.

glass\_train\_labels <- glass[train\_index,10]

glass\_test\_labels <- glass[-train\_index,10]

**Building a Machine Learning model**

At this stage, we have to build a model by using the training data set. Since we’re using the KNN algorithm to build the model, we must first install the ‘class’ package provided by R. This package has the KNN function in it:

Predict the the data set using KNN function

glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,cl = glass\_train\_labels, k=1)

after that predict in error

error <- mean(glass\_test\_pred!=glass\_test\_labels)

**Model Evaluation**

After building the model, it is time to calculate the accuracy of the created models:

We can check the predicted outcome against the actual value in tabular form:

table(glass\_test\_pred ,glass\_test\_labels)

We can also use the confusion matrix to calculate the accuracy. To do this we must first install the infamous Caret package

Now, let’s use the confusion matrix to calculate the accuracy of the KNN model with **K value set to 1**

confusionMatrix(table(glass\_test\_pred, glass\_test\_labels))

Confusion Matrix and Statistics

glass\_test\_labels

glass\_test\_pred 1 2 3 5 6 7

1 19 0 0 0 0 0

2 0 10 0 0 0 0

3 0 0 3 0 0 0

5 0 0 0 1 0 0

6 0 0 0 0 2 1

7 0 0 0 0 2 5

Overall Statistics

Accuracy : 0.9302

95% CI : (0.8094, 0.9854)

No Information Rate : 0.4419

P-Value [Acc > NIR] : 1.474e-11

Kappa : 0.9026

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: 1 Class: 2 Class: 3 Class: 5 Class: 6 Class: 7

Sensitivity 1.0000 1.0000 1.00000 1.00000 0.50000 0.8333

Specificity 1.0000 1.0000 1.00000 1.00000 0.97436 0.9459

Pos Pred Value 1.0000 1.0000 1.00000 1.00000 0.66667 0.7143

Neg Pred Value 1.0000 1.0000 1.00000 1.00000 0.95000 0.9722

Prevalence 0.4419 0.2326 0.06977 0.02326 0.09302 0.1395

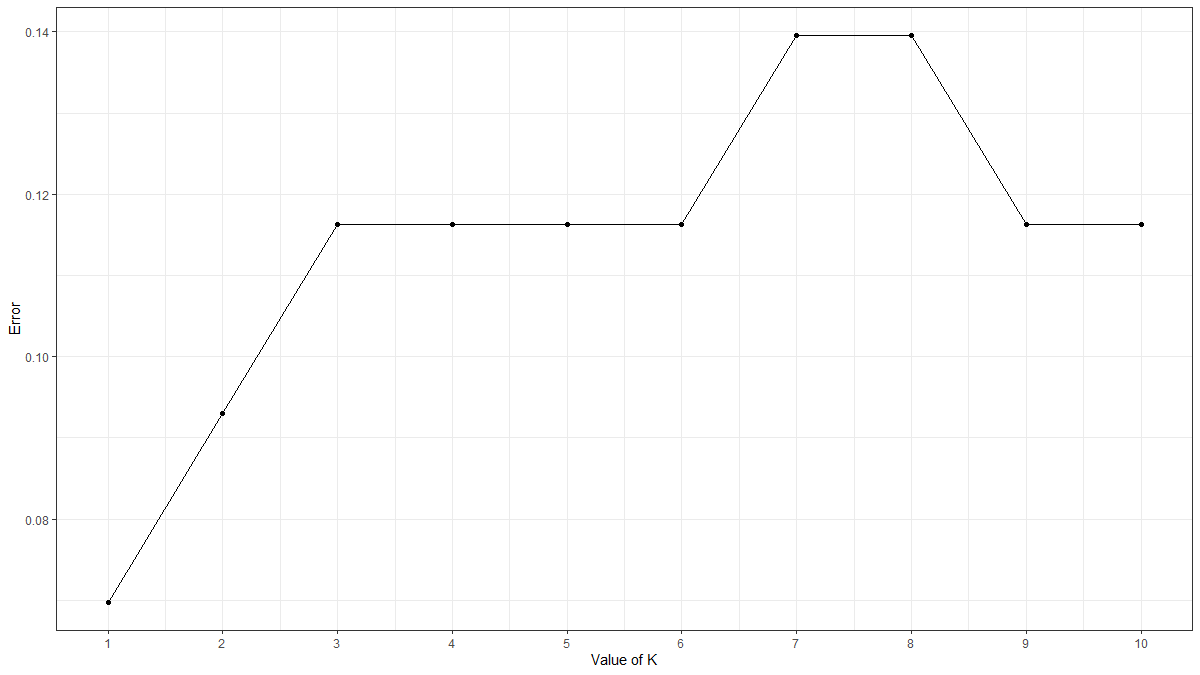
Detection Rate 0.4419 0.2326 0.06977 0.02326 0.04651 0.1163

Detection Prevalence 0.4419 0.2326 0.06977 0.02326 0.06977 0.1628

Balanced Accuracy 1.0000 1.0000 1.00000 1.00000 0.73718 0.8896

So, from the output, we can see that our model predicts the outcome with an accuracy of 93%

In order to improve the accuracy of the model, created a loop that calculates the accuracy of the KNN model for ‘K’ values ranging from 1 to 10. This way you can check which ‘K’ value will result in the most accurate model.



**R-Code:**

**install.packages('ggplot2') #for Data Visualization**

**library(ggplot2)**

**install.packages('corrplot') #Correlation Plot**

**library(corrplot)**

**glass <- read.csv("C:/RAVI/Data science/Assignments/Module 18 KNN/KNN Assignment1 dataset/glass.csv/glass.csv")**

**View(glass)**

**attach(glass)**

**table(glass$Type)**

**str(glass$Type)**

**# table or proportation of enteries in the datasets. What % of glass of Type 1 and what % of glass of Type 2**

**# summarize any three numeric features**

**summary(glass[c("RI", "Na", "Fe")])**

**summary(glass$Fe)**

**head(glass)**

**# create normalization function**

**normalize <- function(x) {**

**return ((x - min(x)) / (max(x) - min(x)))**

**}**

**# normalize the glass data #normalization function - result should be identical#**

**glass\_n <- as.data.frame(lapply(glass[1:9], normalize))**

**glass\_n**

**# confirm that normalization worked**

**summary(glass\_n$Mg)**

**head(glass\_n)**

**#Data Visualization**

**plot(glass\_n)**

**corrplot(cor(glass\_n))**

**# create training and test datasets**

**#random sampling**

**n <- nrow(glass\_n)**

**n1 <- n\*0.8**

**n1**

**n2 <- n-n1**

**n2**

**train\_index <- sample(1:n,n1)**

**glass\_train <- glass[train\_index, ]**

**glass\_test <- glass[-train\_index, ]**

**#Creating seperate dataframe for 'Type' feature which is our target.**

**glass\_train\_labels <- glass[train\_index,10]**

**glass\_test\_labels <- glass[-train\_index,10]**

**#---- Training a model on the data ----**

**#Find the number of observation**

**NROW(glass\_train\_labels)**

**sqrt(171) # k=13**

**# load the "class" library**

**install.packages("class") ##KNN**

**library(class)**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=1)**

**glass\_test\_pred**

**#Error in prediction**

**error <- mean(glass\_test\_pred!=glass\_test\_labels)**

**error**

**install.packages('caret')**

**library(caret)**

**##--------Evaluating model performance ----**

**# load the "gmodels" library**

**#library(gmodels)**

**#Model Evaluation**

**#Calculate the proportion of correct classification for k = 1**

**install.packages('e1071', dependencies=TRUE)**

**# Check prediction against actual value in tabular form for k=1**

**table(glass\_test\_pred ,glass\_test\_labels)**

**confusionMatrix(table(glass\_test\_pred, glass\_test\_labels))**

**glass\_test\_pred <- NULL**

**error\_rate <- NULL**

**for (i in 1:10) {**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,cl = glass\_train\_labels,k=i)**

**error\_rate[i] <- mean(glass\_test\_pred!=glass\_test\_labels)**

**}**

**knn\_error <- as.data.frame(cbind(k=1:10,error\_type =error\_rate))**

**#K Value by Visualization**

**ggplot(knn\_error,aes(k,error\_type))+**

**geom\_point()+**

**geom\_line() +**

**scale\_x\_continuous(breaks=1:10)+**

**theme\_bw() +**

**xlab("Value of K") +**

**ylab('Error')**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=4)**

**glass\_test\_pred**

**#Error in prediction**

**error <- mean(glass\_test\_pred!=glass\_test\_labels)**

**error**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=1)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=2)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=3)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=4)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=5)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=7)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=9)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**

**glass\_test\_pred <- knn(train = glass\_train, test = glass\_test,**

**cl = glass\_train\_labels, k=10)**

**confusionMatrix(table(glass\_test\_pred,glass\_test\_labels))**