**American International University-Bangladesh**



**Course:** INTRODUCTION TO DATA SCIENCE

**Submitted by:**

Name: Fabiha Tasnim  
Id: 20-43426-1  
Section: B

**Submitted to:**

TOHEDUL ISLAM

Assistant Professor, Computer Science  
American International University-Bangladesh

**Submission Date:** 16/8/2023

**Project Description:**

**Project Title:** Patient Treatment Classification Dataset

The dataset obtained from kaggle <https://www.kaggle.com/datasets/manishkc06/patient-treatment-classification>. There are total 11 attributes and total 3309 instances. The names of attributes are Hematocrit, Hemoglobin, Erythrocyte, Leucocyte, Thrombocyte, MCH, MCHC, MCV, Age, Sex and Source. The dataset was obtained from a private hospital in Indonesia using Electronic Health Record Predicting. It includes the patient's laboratory test findings, which are used to determine the patient's subsequent in- or out-patient care. The amount of time a patient must stay in the institution where they are having the treatment done distinguishes between inpatient and outpatient care. Hospitalization overnight is necessary for inpatient care. Patients are required to spend at least one night at the hospital or other healthcare facility where their procedure was performed. They continue to be supervised by a nurse or doctor during this period. There is no requirement for outpatient patients to spend the night in the hospital. Patients are not required to undergo surgery if there are no significant complications.

Here two classes included in Source attribute,

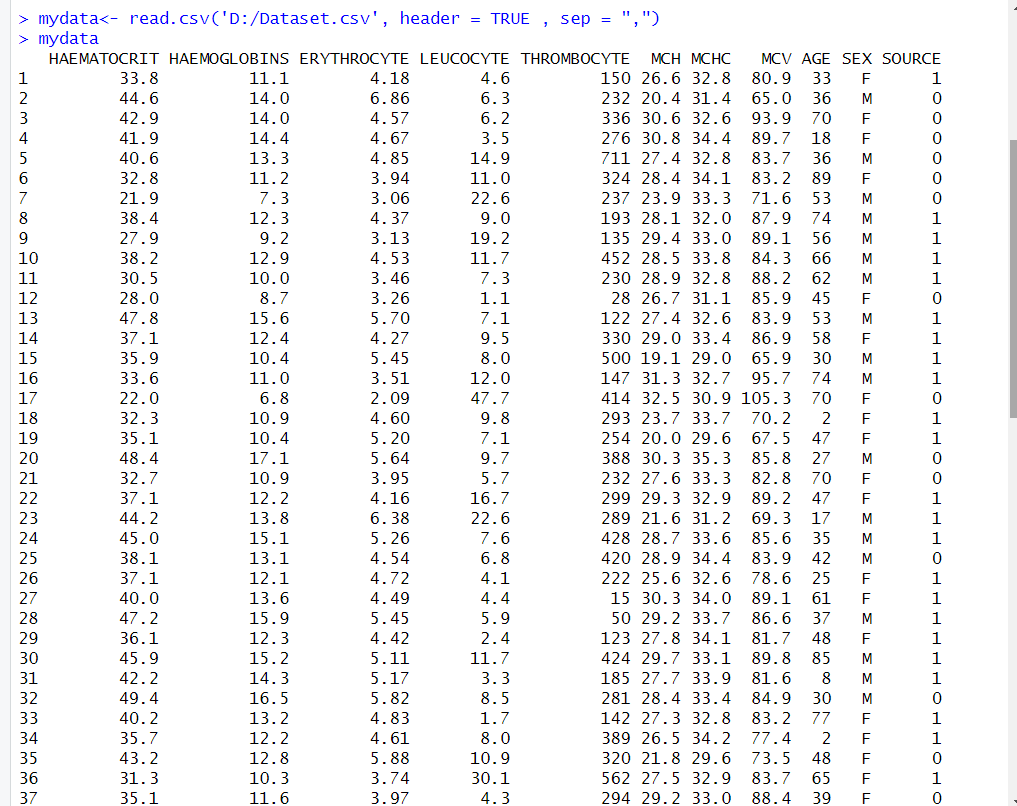
1. In-care patient
2. Out-care patient

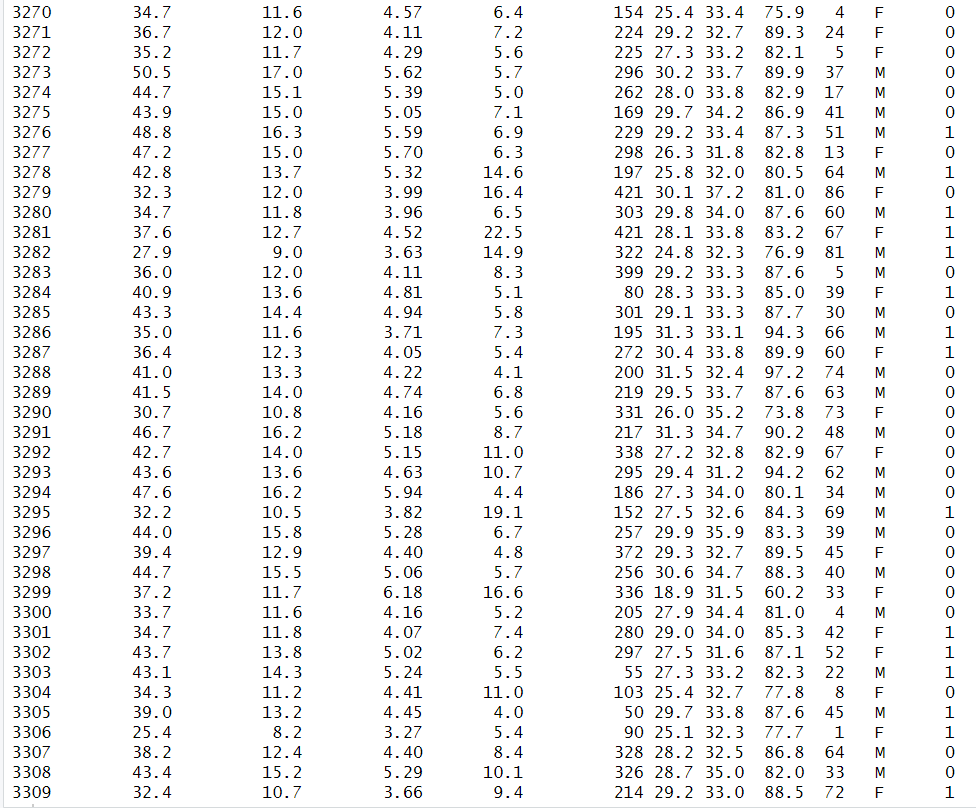
**KNN:** KNN stands for k-Nearest Neighbor; KNN is a supervised machine learning technique that assigns a new data point to the target class based on the characteristics of its nearby data points. Data points are classified into the class to which they are most similar by the KNN algorithm, which examines how similar they are to their neighbors.

**IMPORT DATASET:**

mydata<- read.csv('D:/Dataset.csv', header = TRUE , sep = ",")

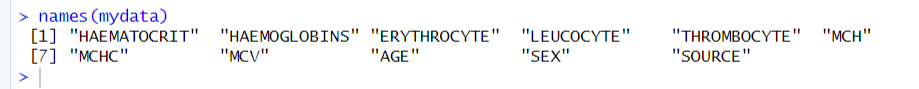
mydata





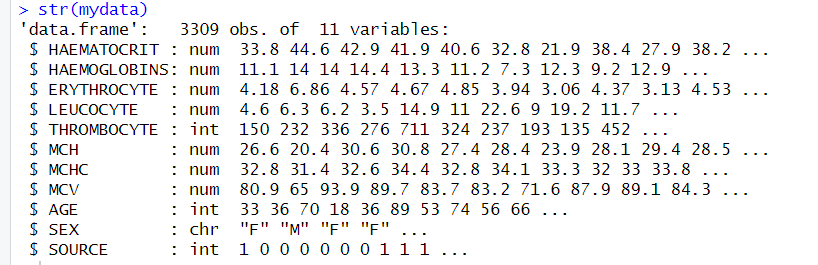
**ATTRIBUTES NAMES:**

names(mydata)



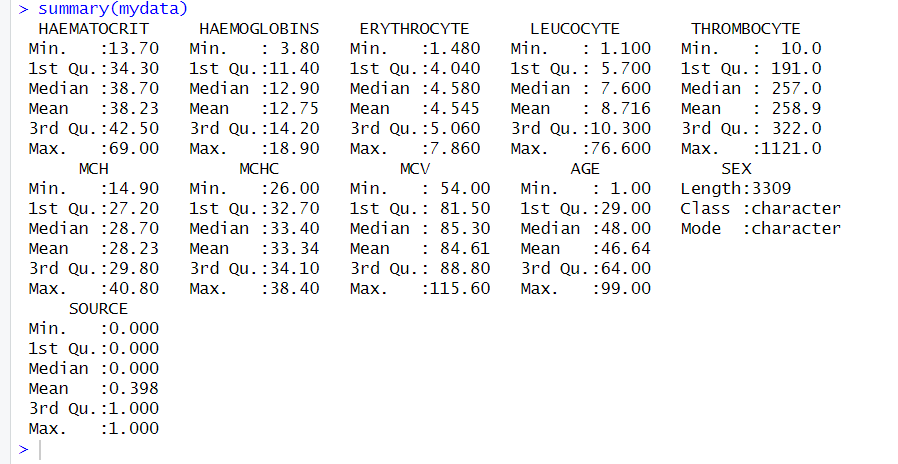
**STRUCTURE OF DATASET:**

str(mydata)



**SUMMARY OF DATASET:**

summary(mydata)



**CONVERT INTO NUMERIC:** For KNN all data have to be in Numeric.

mydata$SEX <- factor(mydata$SEX, levels = c("F","M"), labels = c(0,1))

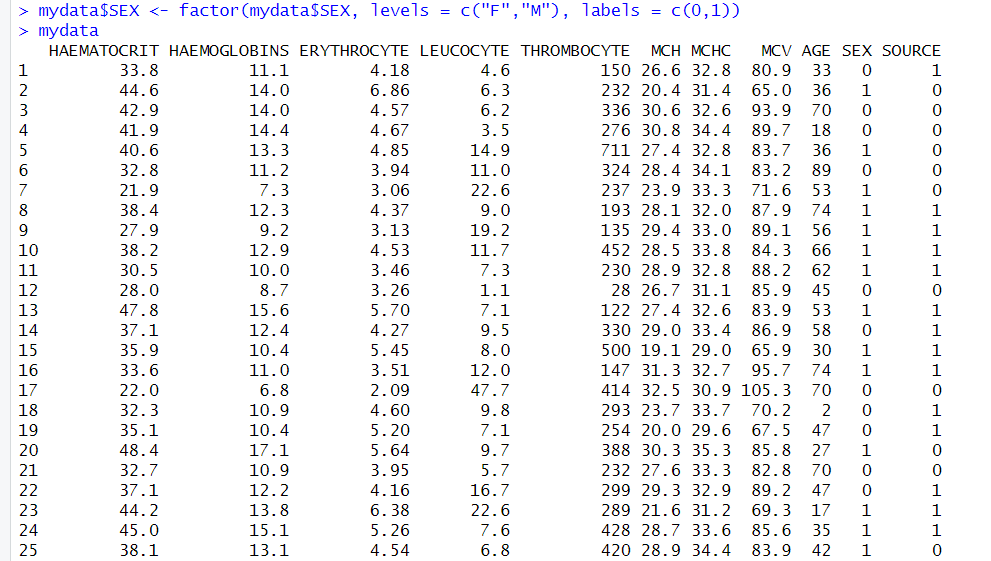
mydata

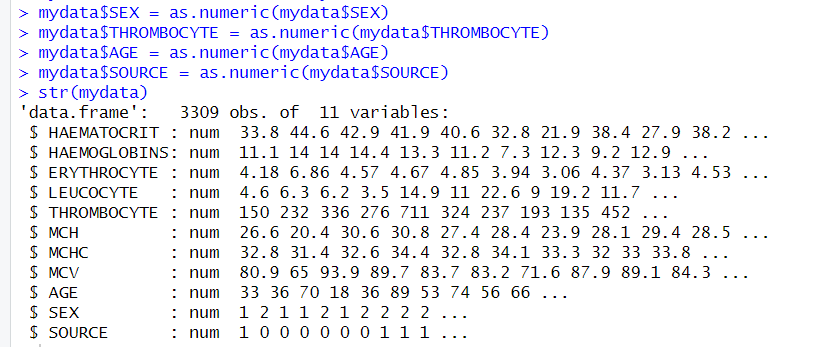
mydata$SEX = as.numeric(mydata$SEX)

mydata$THROMBOCYTE = as.numeric(mydata$THROMBOCYTE)

mydata$AGE = as.numeric(mydata$AGE)

mydata$SOURCE = as.numeric(mydata$SOURCE)



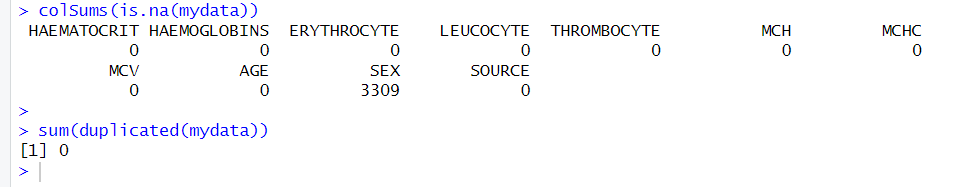


**MISSING VALUE AND DUPLICATE VALUE CHECK:**

To check any missing value and duplicate values increase the KNN algorithm's extra complexity and risk of bias. Before utilizing KNN, it's generally a good idea to clean and preprocess dataset to remove duplicate data points.

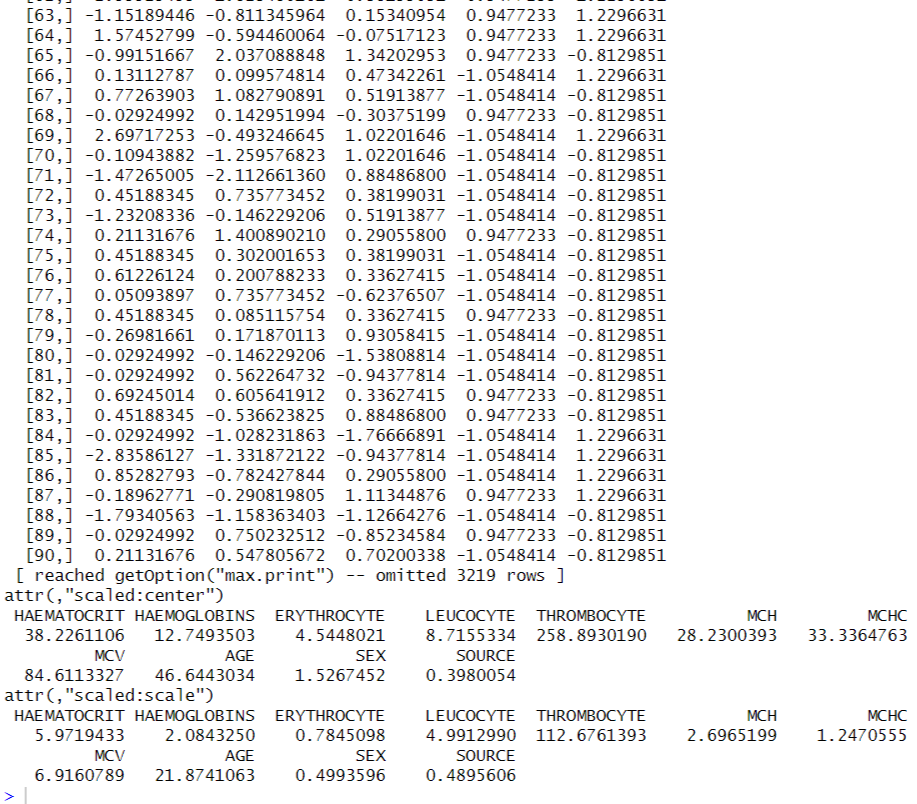
colSums(is.na(mydata))

sum(duplicated(mydata))



**SCALING DATASET:**

scale(mydata)

  
One of the most important phases in pre-processing data before building a model is data scaling. Normalization is the method of feature scaling that is most frequently used. values will exist between [0,1] or [-1,1].

**NORMALIZE DATASET:**

After normalization all data will be in a range of [0,1]. Normalization gives rise to predictions that are more accurate and meaningful comparisons between data points, which is important when using KNN's nearest neighbors to make judgments.

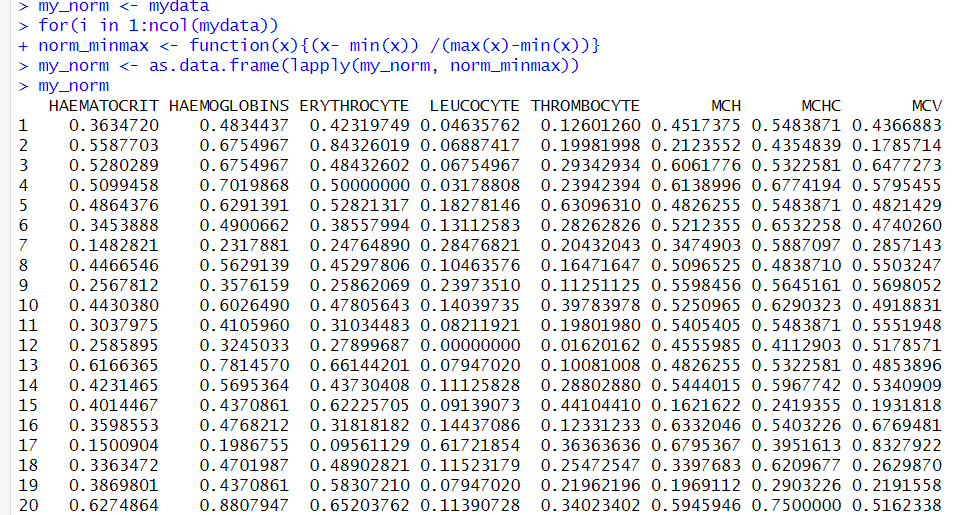
my\_norm <- mydata

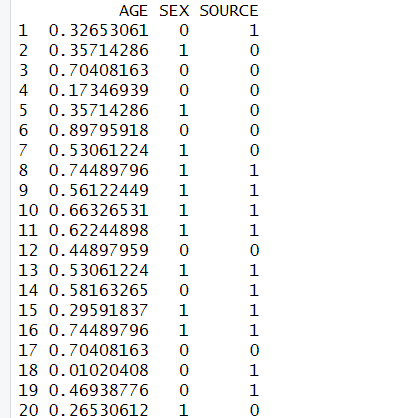
for(i in 1:ncol(mydata))

norm\_minmax <- function(x){(x- min(x)) /(max(x)-min(x))}

my\_norm <- as.data.frame(lapply(my\_norm, norm\_minmax))

my\_norm

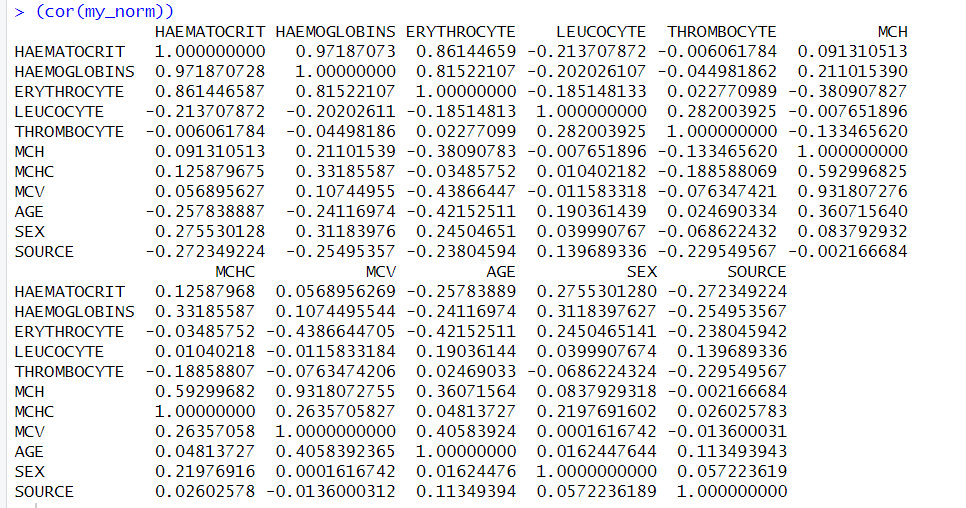




**CORRELATION BETWEEN ATTRIBUTES:**

A statistical measure known as correlation evaluates the strength of the relationship or association between two variables. The range of correlation values is from -1 to 1, where -1 denotes a perfect negative correlation, 1 a perfect positive correlation, and 0 denotes no linear correlation between the variables. As one variable rises, the other falls.

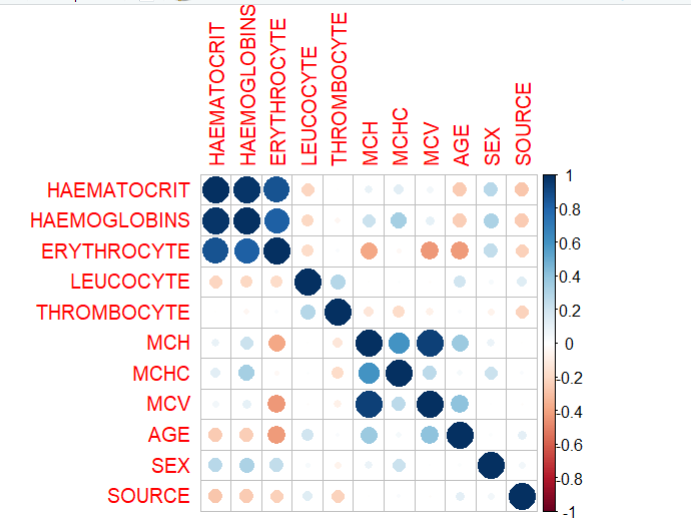
(cor(my\_norm))



library(corrplot)

correlation\_matrix <- cor(my\_norm)

corrplot(correlation\_matrix, method = "circle")



Here, negatively and positively correlation has shown.

**REMOVE LESS CORRELATED ATTRIBUTES:**

Removing less correlated attributes to make more easier for KNN apply.

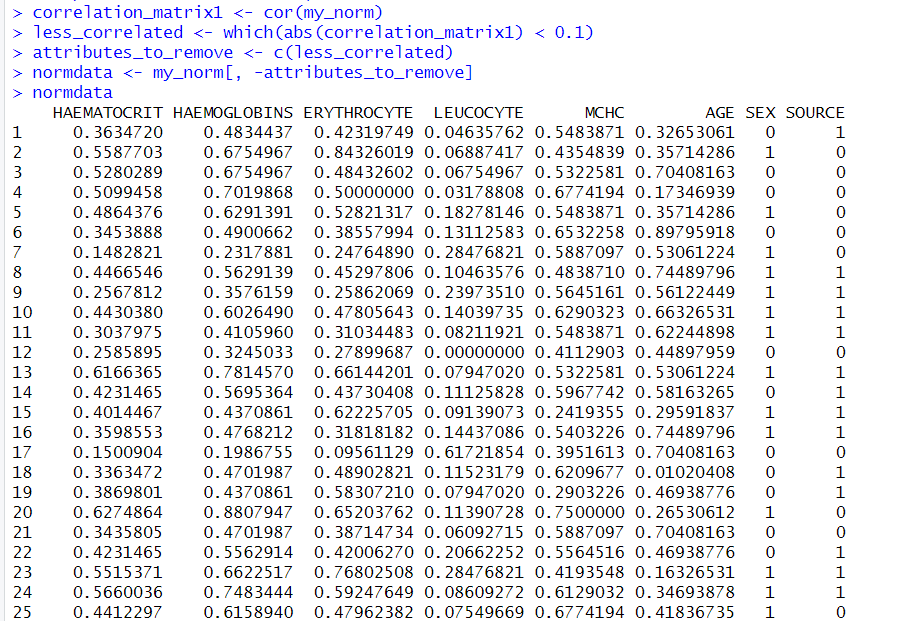
correlation\_matrix1 <- cor(my\_norm)

less\_correlated <- which(abs(correlation\_matrix1) < 0.1)

attributes\_to\_remove <- c(less\_correlated)

normdata <- my\_norm[, -attributes\_to\_remove]

normdata



Here, which correlation value is less than 0.1 have removed and after removing those dataset only 8 attributes remaining.

**TRAIN DATASET & TEST DATASET:**

For KNN, total dataset need to divided into 2 datasets that are train dataset and test dataset. Test data represents real-world scenarios for KNN algorithm training, assessing its ability to generalize and predict on unseen instances. Performance metrics, such as accuracy, are calculated using test data.

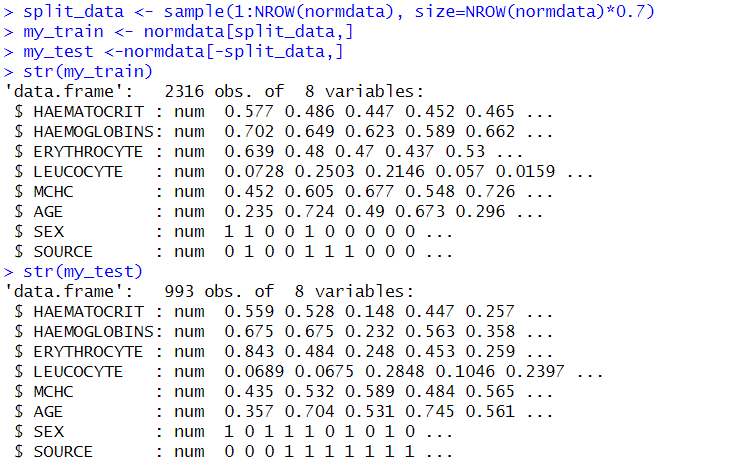
split\_data <- sample(1:NROW(normdata), size=NROW(normdata)\*0.7)

my\_train <- normdata[split\_data,]

my\_test <-normdata[-split\_data,]

str(my\_train)

str(my\_test)



Here, total 100% of data split into 70% train data and 30% of data become test data. So 2316 data becomes train data and 993 data becomes test data.

**APPLYING KNN:**

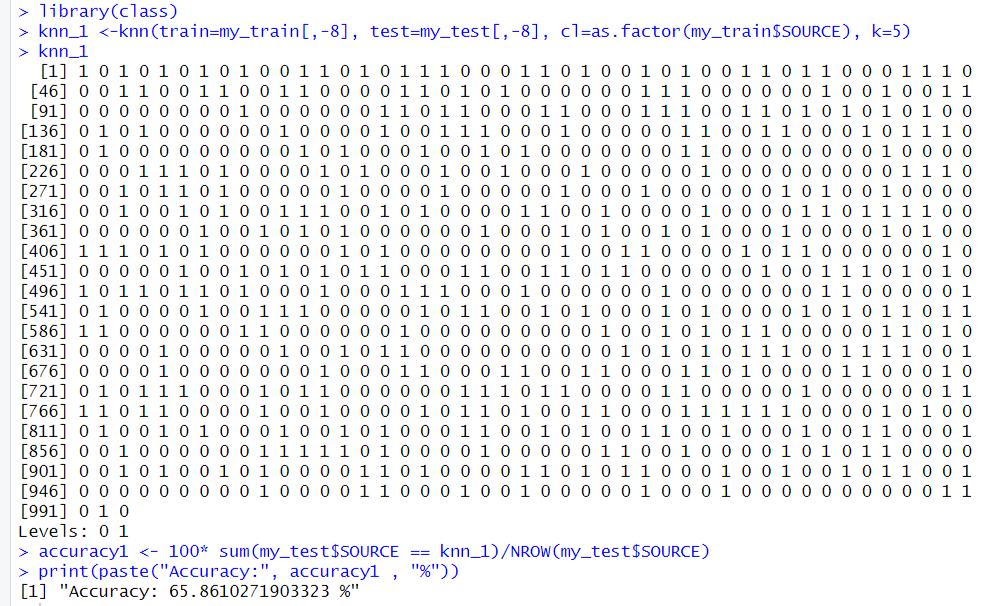
library(class)

knn\_1 <-knn(train=my\_train[,-8], test=my\_test[,-8], cl=as.factor(my\_train$SOURCE), k=5)

knn\_1

accuracy1 <- 100\* sum(my\_test$SOURCE == knn\_1)/NROW(my\_test$SOURCE)

print(paste("Accuracy:", accuracy1 , "%"))

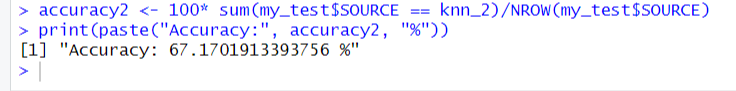


knn\_2 <- knn(train = my\_train[,-8], test = my\_test[,-8], cl = my\_train$SOURCE, k = 50)

knn\_2

accuracy2 <- 100\* sum(my\_test$SOURCE == knn\_2)/NROW(my\_test$SOURCE)

print(paste("Accuracy:", accuracy2, "%"))



For KNN\_1 the value of K=5, the 5 iterations have completed and the result is approximately 65.86% and another KNN\_2 is taken and the value of K=50, 50 iterations have completed and the result is 67.17%. So by this, the maximum iterated (K) is given better result.

**K-FOLD CROSS VALIDATION:**

K-Fold cross validation usually Ks value is 10. In 10-fold cross-validation, the dataset is divided into ten equal "folds." Nine folds are used to train the model, and one fold is saved for testing. Ten times this procedure is carried out, one for each fold acting as the test set. The model's performance is evaluated by averaging the performance measures from each iteration. This method guarantees thorough analysis, avoids over fitting, and offers a trustworthy prediction of the model's performance on new data.

accuracy <- numeric(10)

all\_fold\_predicted\_data <- character(0)

all\_labels <- character(0)

for (fold in 1:10) {

fold\_data <- sample(seq\_len(nrow(normdata)), size = floor(0.1 \* nrow(normdata)))

test\_data <- normdata[fold\_data, ]

train\_data <- normdata[-fold\_data, ]

library(class)

predicted\_data <- knn(train\_data[, -8], test\_data[, -8], train\_data$SOURCE, k)

all\_fold\_predicted\_data <- c(all\_fold\_predicted\_data, predicted\_data)

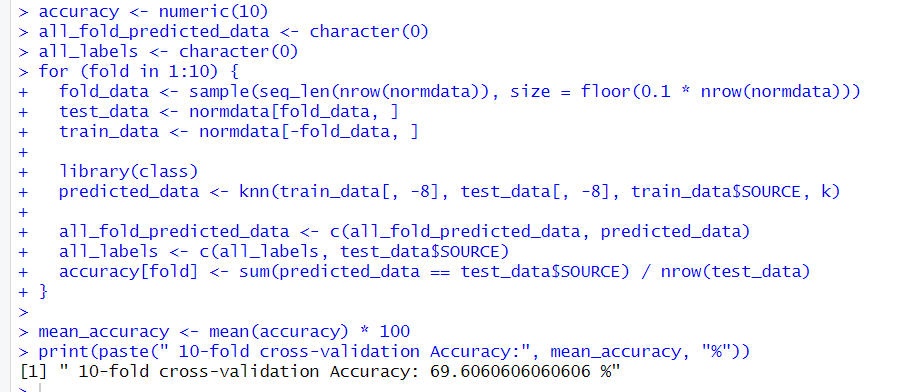
all\_labels <- c(all\_labels, test\_data$SOURCE)

accuracy[fold] <- sum(predicted\_data == test\_data$SOURCE) / nrow(test\_data)

}

mean\_accuracy <- mean(accuracy) \* 100

print(paste(" 10-fold cross-validation Accuracy:", mean\_accuracy, "%"))

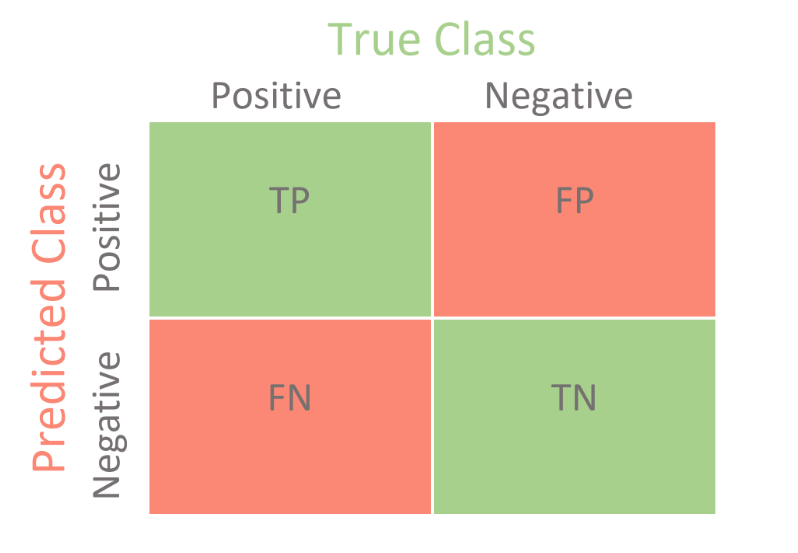


In 10-fold cross validation accuracy is 69.6%.

After KNN and 10-fold cross validation, we can see in KNN the accuracy is 67.17% and 10-fold cross validation accuracy is 69.6%. So, we can say 10-fold cross validation is more appropriate and gives better result.

**CONFUSION MATRIX:**

A table that displays how well predictions from a categorization model match real data is called a confusion matrix. Understanding the model's strengths (true positives and true negatives) and weaknesses (false positives and false negatives) will help reveal its accuracy and flaws.



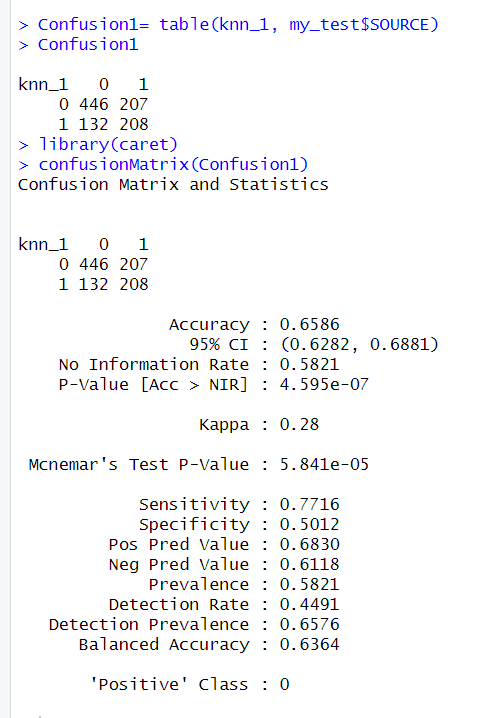
For KNN\_1 and KNN\_2 the confusion matrix are,

Confusion1= table(knn\_1, my\_test$SOURCE)

Confusion1

library(caret)

confusionMatrix(Confusion1)

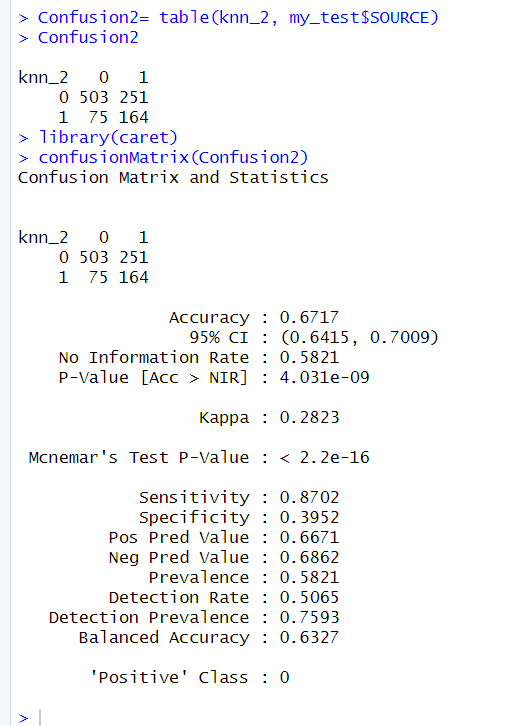


Confusion1= table(knn\_1, my\_test$SOURCE)

Confusion1

library(caret)

confusionMatrix(Confusion1)



**RECALL & PRECISION:**

Recall and precision are two vital performance metrics derived from the confusion matrix in classification tasks. Precision assesses the accuracy of the model's accurate positive predictions, whereas recall reflects a model's capacity to accurately identify all pertinent occurrences of a class. Both measures help to find a balance between avoiding false negatives (high recall) and eliminating false positives (high accuracy), providing useful insights into many facets of a model's performance.

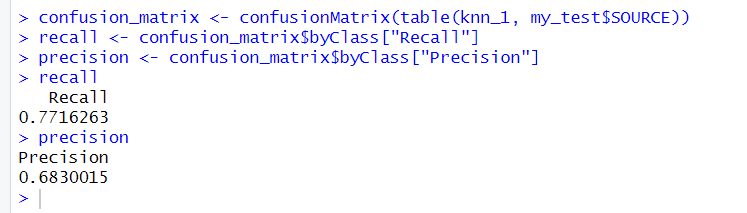
confusion\_matrix <- confusionMatrix(table(knn\_1, my\_test$SOURCE))

recall <- confusion\_matrix$byClass["Recall"]

precision <- confusion\_matrix$byClass["Precision"]

recall

precision



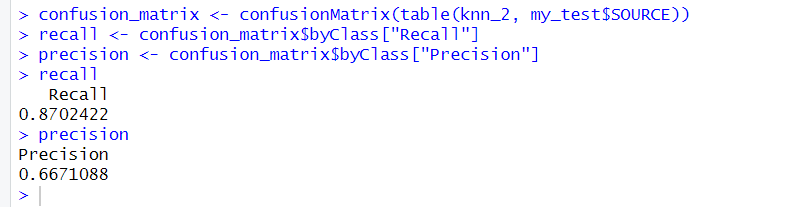
confusion\_matrix <- confusionMatrix(table(knn\_2, my\_test$SOURCE))

recall <- confusion\_matrix$byClass["Recall"]

precision <- confusion\_matrix$byClass["Precision"]

recall

precision



For KNN\_1 & KNN\_2 recall and precision have measured and finds the difference.