**#logistic regression**

data <- read.csv("/Users/nirajkc/Desktop/data-final project/winequality-red.csv", header=TRUE)

str(data)

#transform the dependent variable into a binary categorical response

#Values above or equal to seven will be changed to 1,

#meaning a good quality wine.

#On the other hand, amounts less than seven will be converted to 0

#and will indicate bad or mediocre quality

#modified the type of the variable “quality” to factor,

#indicating that the variable is categorical

data$quality <- ifelse(data$quality >= 7, 1, 0)

data$quality <- factor(data$quality, levels = c(0, 1))

#Descriptive statistics

summary(data)

#univariate analysis

#analyzed the dependent variable

#generated a table of frequency count

par(mfrow=c(1,1))

barplot(table(data[[12]]),

main = sprintf('Frequency plot of the variable: %s',

colnames(data[12])),

xlab = colnames(data[12]),

ylab = 'Frequency')

#Check class BIAS

table(data$quality)

round(prop.table((table(data$quality))),2)

# considerably higher amount of 0 values,

#indicating that the data the data is biased.

#86% of 0 outcome values and 14% of 1(good quality)

#that is why it is essential to follow a stratified sampling method

#when splitting the data into the train and test set

#create boxplots and histogram plots for each variable.

#These visualizations will help us identify

#the location of the five-number summary values, the outliers it possesses

#all the variables have outliers

#variables “residual sugar” and “chlorides” are the variables

#that have the most amount of outliers

par(mfrow=c(3,4))

for (i in 1:(length(data)-1)){

boxplot(x = data[i],

horizontal = TRUE,

main = sprintf('Boxplot of the variable: %s',

colnames(data[i])),

xlab = colnames(data[i]))

}

#Histograms

#Visualizing the histogram plots

#there is a right skewness in most of the distributions. However, the

#variables “density” and “pH” show that they follow a normal distribution

par(mfrow=c(3,4))

for (i in 1:(length(data)-1)){

hist(x = data[[i]],

main = sprintf('Histogram of the variable: %s',

colnames(data[i])),

xlab = colnames(data[i]))

}

#bivariate analysis to understand the relationship

#that the variables have with each other

#Bivariate analysis

#Correlation matrix

#positive correlation of 0.67 between the “fixed acidity”

#variable and the variables “citric acid” and “density”.

library(ggcorrplot)

ggcorrplot(round(cor(data[-12]), 2),

type = "lower",

lab = TRUE,

title =

'Correlation matrix of the red wine quality dataset')

#Missing values

sum(is.na(data))

#Outliers

#Identifing outliers

is\_outlier <- function(x) {

return(x < quantile(x, 0.25) - 1.5 \* IQR(x) |

x > quantile(x, 0.75) + 1.5 \* IQR(x))

}

outlier <- data.frame(variable = character(),

sum\_outliers = integer(),

stringsAsFactors=FALSE)

for (j in 1:(length(data)-1)){

variable <- colnames(data[j])

for (i in data[j]){

sum\_outliers <- sum(is\_outlier(i))

}

row <- data.frame(variable,sum\_outliers)

outlier <- rbind(outlier, row)

}

#all of the variables in the data have outliers

#I will accept variables that have

#less than 5% of outlier values

#values because they represent and carry necessary information about the dataset.

#Deleting the outliers

#can bias the result of our model in a significant way

#Identifying the percentage of outliers

for (i in 1:nrow(outlier)){

if (outlier[i,2]/nrow(data) \* 100 >= 5){

print(paste(outlier[i,1],

'=',

round(outlier[i,2]/nrow(data) \* 100, digits = 2),

'%'))

}

}

#I chose to change the outlier values with the mean value of the variables because, as we can see in the histogram plot, both variables have a large concentration

#of value naer the mean.

#Inputting outlier values

for (i in 4:5){

for (j in 1:nrow(data)){

if (data[[j, i]] > as.numeric(quantile(data[[i]], 0.75) +

1.5 \* IQR(data[[i]]))){

if (i == 4){

data[[j, i]] <- round(mean(data[[i]]), digits = 2)

} else{

data[[j, i]] <- round(mean(data[[i]]), digits = 3)

}

}

}

}

#Modelling

#80% of the observations that represent a good quality wine (1 outcome of the “quality” variable) to balance the train set. In other words,

#the dependent variable will have the

#same number of observations of 0 and 1 in the train set.

#Splitting the dataset into the Training set and Test set

#Stratified sample

data\_ones <- data[which(data$quality == 1), ]

data\_zeros <- data[which(data$quality == 0), ]

#Train data

set.seed(123)

train\_ones\_rows <- sample(1:nrow(data\_ones), 0.8\*nrow(data\_ones))

train\_zeros\_rows <- sample(1:nrow(data\_zeros), 0.8\*nrow(data\_ones))

train\_ones <- data\_ones[train\_ones\_rows, ]

train\_zeros <- data\_zeros[train\_zeros\_rows, ]

train\_set <- rbind(train\_ones, train\_zeros)

table(train\_set$quality)

#Test Data

test\_ones <- data\_ones[-train\_ones\_rows, ]

test\_zeros <- data\_zeros[-train\_zeros\_rows, ]

test\_set <- rbind(test\_ones, test\_zeros)

table(test\_set$quality)

#develop the models and determine which model can

#accurately predict the quality of red wine.

#Logistic Regression

lr = glm(formula = quality ~.,

data = train\_set,

family = binomial)

#Predictions

prob\_pred = predict(lr,

type = 'response',

newdata = test\_set[-12])

library(InformationValue)

optCutOff <- optimalCutoff(test\_set$quality, prob\_pred)[1]

y\_pred = ifelse(prob\_pred > optCutOff, 1, 0)

#Once the model is created, with the training set,

#I proceed to predict the values with the test set data.

#Since the logistic regression will deliver probability values, I proceed to calculate the optimal cut-off

#point, which will categorize the outcome values into 1 or 0

#Making the confusion matrix

cm\_lr = table(test\_set[, 12], y\_pred)

cm\_lr

#Accuracy

accuracy\_lr = (cm\_lr[1,1] + cm\_lr[1,1])/

(cm\_lr[1,1] + cm\_lr[1,1] + cm\_lr[2,1] + cm\_lr[1,2])

accuracy\_lr

#ROC curve

library(ROSE)

par(mfrow = c(1, 1))

roc.curve(test\_set$quality, y\_pred)

#F1 SCORE

install.packages('caret')

library(caret)

confusionMatrix(y\_pred, test\_set$quality)

confusionMatrix(as.factor(y\_pred), as.factor(test\_set$quality))

confusionMatrix(pred,as.factor(testing$Final))

confusionMatrix(

as.factor(y\_pred),

test\_set$quality,

mode = "everything",

)

**#Decision Tree**

library(rpart)

dt = rpart(formula = quality ~ .,

data = train\_set,

method = 'class')

#Predictions

y\_pred = predict(dt,

type = 'class',

newdata = test\_set[-12])

#Making the confusion matrix

cm\_dt = table(test\_set[, 12], y\_pred)

cm\_dt

#Accuracy

accuracy\_dt = (cm\_dt[1,1] + cm\_dt[1,1])/

(cm\_dt[1,1] + cm\_dt[1,1] + cm\_dt[2,1] + cm\_dt[1,2])

accuracy\_dt

#ROC curve

library(ROSE)

roc.curve(test\_set$quality, y\_pred)

library(ROSE)

par(mfrow = c(1, 1))

roc.curve(test\_set$quality, y\_pred)

install.packages("rpart")

library(rpart)

tm <- rpart(quality~., train\_set, method = "class")

install.packages("rpart.plot")

library(rpart.plot)

rpart.plot(tm, tweak = 1.4)

rpart.plot(tm, type =4, extra = 101, tweak = 1.5)

#F1 SCORE

confusionMatrix(data=y\_pred, reference = test\_set[, 12], mode = "everything")

**#Random forest**

library(randomForest)

rf = randomForest(x = train\_set[-12],

y = train\_set$quality,

ntree = 10)

#Predictions

y\_pred = predict(rf,

type = 'class',

newdata = test\_set[-12])

#Making the confusion matrix

cm\_rf = table(test\_set[, 12], y\_pred)

cm\_rf

#Accuracy

accuracy\_rf = (cm\_rf[1,1] + cm\_rf[1,1])/

(cm\_rf[1,1] + cm\_rf[1,1] + cm\_rf[2,1] + cm\_rf[1,2])

accuracy\_rf

#ROC curve

library(ROSE)

roc.curve(test\_set$quality, y\_pred)

**# Basic knn model**

set.seed(123)

wineknns<- knn(train = train\_set1, test=test\_set1, cl=wineTrain\_label, k=19)

# Evaluating model performance

confusionMatrix(wineTest\_label, wineknns, )

confusionMatrix(

as.factor(wineknns),

wineTest\_label,

mode = "everything",

)

library(ROSE)

roc.curve(wineTest\_label, wineknns)