#Clean the environment

rm(list = ls())

#Set working directory

setwd("C:/Users/hriti/OneDrive/Documents/Employee-Absenteeism-master")

#Load the librarires

libraries = c("dummies","caret","rpart.plot","plyr","dplyr", "ggplot2","rpart","dplyr","DMwR","randomForest","usdm","corrgram","DataCombine","xlsx")

lapply(X = libraries,require, character.only = TRUE)

rm(libraries)

#Read the csv file

emp\_absent = read.csv("sheet.csv",stringsAsFactors = F)

########################################EXPLORE THE DATA########################################

#Check number of rows and columns

dim(emp\_absent)

#Observe top 5 rows

head(emp\_absent)

#Structure of variables

str(emp\_absent)

########

#emp\_absent$Work.load.Average.day =as.numeric(as.character(emp\_absent$Work.load.Average.day))

#######

#Transform data types

emp\_absent$ID = as.factor(as.character(emp\_absent$ID))

emp\_absent$Reason.for.absence[emp\_absent$Reason.for.absence %in% 0] = 20

emp\_absent$Reason.for.absence = as.factor(as.character(emp\_absent$Reason.for.absence))

emp\_absent$Month.of.absence[emp\_absent$Month.of.absence %in% 0] = NA

emp\_absent$Month.of.absence = as.factor(as.character(emp\_absent$Month.of.absence))

emp\_absent$Day.of.the.week = as.factor(as.character(emp\_absent$Day.of.the.week))

emp\_absent$Seasons = as.factor(as.character(emp\_absent$Seasons))

emp\_absent$Disciplinary.failure = as.factor(as.character(emp\_absent$Disciplinary.failure))

emp\_absent$Education = as.factor(as.character(emp\_absent$Education))

emp\_absent$Son = as.factor(as.character(emp\_absent$Son))

emp\_absent$Social.drinker = as.factor(as.character(emp\_absent$Social.drinker))

emp\_absent$Social.smoker = as.factor(as.character(emp\_absent$Social.smoker))

emp\_absent$Pet = as.factor(as.character(emp\_absent$Pet))

#Structure of variables

str(emp\_absent)

#Make a copy of data

df = emp\_absent

########################################MISSING VALUE ANALYSIS########################################

#Get number of missing values

sum(is.na(df))

sapply(df,function(x){sum(is.na(x))})

missing\_values = data.frame(sapply(df,function(x){sum(is.na(x))}))

#Get the rownames as new column

missing\_values$Variables = row.names(missing\_values)

#Reset the row names

row.names(missing\_values) = NULL

#Rename the column

names(missing\_values)[1] = "Miss\_perc"

head(missing\_values)

#Calculate missing percentage

missing\_values$Miss\_perc = ((missing\_values$Miss\_perc/nrow(emp\_absent)) \*100)

#Reorder the columns

missing\_values = missing\_values[,c(2,1)]

head(missing\_values)

#Sort the rows according to decreasing missing percentage

missing\_values = missing\_values[order(-missing\_values$Miss\_perc),]

head(missing\_values)

#Create a bar plot to visualie top 5 missing values

ggplot(data = missing\_values[1:5,], aes(x=reorder(Variables, -Miss\_perc),y = Miss\_perc))+

geom\_bar(stat = "identity",fill = "grey")+xlab("Parameter")+

ggtitle("Missing data percentage") + theme\_bw()

#Create missing value and impute using mean, median and knn

df = knnImputation(data = df, k = 3)

#Check if any missing values

sum(is.na(df))

###################################

# Saving output result into excel file

write.csv(missing\_values, "Missing\_perc\_R.xlsx", row.names = F)

######################

########################################EXPLORE DISTRIBUTION USING GRAPHS########################################

#Get numerical data

numeric\_index = sapply(df, is.numeric)

numeric\_index

numeric\_data = df[,numeric\_index]

#Distribution of factor data using bar plot

bar1 = ggplot(data = df, aes(x = ID)) + geom\_bar() + ggtitle("Count of ID") + theme\_bw()

bar1

bar2 = ggplot(data = df, aes(x = Reason.for.absence)) + geom\_bar() +

ggtitle("Count of Reason for absence") + theme\_bw()

bar2

bar3 = ggplot(data = df, aes(x = Month.of.absence)) + geom\_bar() + ggtitle("Count of Month") + theme\_bw()

bar3

bar4 = ggplot(data = df, aes(x = Disciplinary.failure)) + geom\_bar() +

ggtitle("Count of Disciplinary failure") + theme\_bw()

bar4

bar5 = ggplot(data = df, aes(x = Education)) + geom\_bar() + ggtitle("Count of Education") + theme\_bw()

#(high school (1), graduate (2), postgraduate (3), master and doctor (4)

bar5

bar6 = ggplot(data = df, aes(x = Son)) + geom\_bar() + ggtitle("Count of Son") + theme\_bw()

bar6

bar7 = ggplot(data = df, aes(x = Social.smoker)) + geom\_bar() +

ggtitle("Count of Social smoker") + theme\_bw()

bar7

##### show all #########

gridExtra::grid.arrange(bar1,bar2,bar3,bar4,ncol=2)

gridExtra::grid.arrange(bar5,bar6,bar7,ncol=2)

#######

#Check the distribution of numerical data using histogram

hist1 = ggplot(data = numeric\_data, aes(x =Transportation.expense)) +

ggtitle("Transportation.expense") + geom\_histogram(bins = 25)

hist2 = ggplot(data = numeric\_data, aes(x =Height)) +

ggtitle("Distribution of Height") + geom\_histogram(bins = 25)

hist3 = ggplot(data = numeric\_data, aes(x =Body.mass.index)) +

ggtitle("Distribution of Body.mass.index") + geom\_histogram(bins = 25)

hist4 = ggplot(data = numeric\_data, aes(x =Absenteeism.time.in.hours)) +

ggtitle("Distribution of Absenteeism.time.in.hours") + geom\_histogram(bins = 25)

gridExtra::grid.arrange(hist1,hist2,hist3,hist4,ncol=2)

########################################OUTLIER ANALYSIS########################################

#Get the data with only numeric columns

numeric\_index = sapply(df, is.numeric)

numeric\_index

numeric\_data = df[,numeric\_index]

#Get the data with only factor columns

factor\_data = df[,!numeric\_index]

#Check for outliers using boxplots

for(i in 1:ncol(numeric\_data)) {

assign(paste0("box",i), ggplot(data = df, aes\_string(y = numeric\_data[,i])) +

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour = "red", fill = "grey", outlier.size = 1) +

labs(y = colnames(numeric\_data[i])) +

ggtitle(paste("Boxplot: ",colnames(numeric\_data[i]))))

}

#Arrange the plots in grids

gridExtra::grid.arrange(box1,box2,box3,box4,ncol=2)

gridExtra::grid.arrange(box5,box6,box7,box8,ncol=2)

gridExtra::grid.arrange(box9,ncol=1)

#Get the names of numeric columns

numeric\_columns = colnames(numeric\_data)

#Replace all outlier data with NA

for(i in numeric\_columns){

val = df[,i][df[,i] %in% boxplot.stats(df[,i])$out]

print(paste(i,length(val)))

df[,i][df[,i] %in% val] = NA

}

#Check number of missing values

sapply(df,function(x){sum(is.na(x))})

#Get number of missing values after replacing outliers as NA

missing\_values\_out = data.frame(sapply(df,function(x){sum(is.na(x))}))

missing\_values\_out$Columns = row.names(missing\_values\_out)

row.names(missing\_values\_out) = NULL

names(missing\_values\_out)[1] = "Miss\_perc"

missing\_values\_out$Miss\_perc = ((missing\_values\_out$Miss\_perc/nrow(emp\_absent)) \*100)

missing\_values\_out = missing\_values\_out[,c(2,1)]

missing\_values\_out = missing\_values\_out[order(-missing\_values\_out$Miss\_perc),]

missing\_values\_out

#Compute the NA values using KNN imputation

df = knnImputation(df, k = 5)

#Check if any missing values

sum(is.na(df))

########################################FEATURE SELECTION########################################

#Check for multicollinearity using corelation graph

library(corrgram)

corrgram(numeric\_data, order = NULL, upper.panel=panel.pie,

text.panel=panel.txt, main = "Correlation Plot")

##############################################################################

#Variable Reduction

df = subset.data.frame(df, select = -c(Body.mass.index))

#Make a copy of Clean Data

clean\_data = df

#write.xlsx(clean\_data, "clean\_data.xlsx", row.names = F)

########################################DIMENSION REDUCTION USING PCA########################################

#Principal component analysis

#Splitting data into train and test data

set.seed(1)

train\_index = sample(1:nrow(df), 0.8\*nrow(df))

train = df[train\_index,]

test = df[-train\_index,]

prin\_comp = prcomp(train)

#Compute standard deviation of each principal component

pr\_stdev = prin\_comp$sdev

#Compute variance

pr\_var = pr\_stdev^2

#Proportion of variance explained

prop\_var = pr\_var/sum(pr\_var)

#Cumulative scree plot

plot(cumsum(prop\_var), xlab = "Principal Component",

ylab = "Cumulative Proportion of Variance Explained",

type = "b")

#Add a training set with principal components

train.data = data.frame(Absenteeism.time.in.hours = train$Absenteeism.time.in.hours, prin\_comp$x)

# From the above plot selecting 45 components since it explains almost 95+ % data variance

train.data =train.data[,1:45]

#Transform test data into PCA

test.data = predict(prin\_comp, newdata = test)

test.data = as.data.frame(test.data)

#Select the first 45 components

test.data=test.data[,1:45]

########################################DECISION TREE########################################

#Build decsion tree using rpart

dt\_model = rpart(Absenteeism.time.in.hours ~., data = train.data, method = "anova")

#Predict the test cases

dt\_predictions = predict(dt\_model,test.data)

#Create data frame for actual and predicted values

df\_pred = data.frame("actual"=test[,114], "dt\_pred"=dt\_predictions)

head(df\_pred)

#Calcuate MAE, RMSE, R-sqaured for testing data

print(postResample(pred = dt\_predictions, obs = test$Absenteeism.time.in.hours))

#Plot a graph for actual vs predicted values

plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="green")

lines(dt\_predictions,col="blue")

#RMSE Rsquared MAE

#0.2108736 0.1660149 0.1443652

########################################RANDOM FOREST########################################

#Train the model using training data

rf\_model = randomForest(Absenteeism.time.in.hours~., data = train.data, ntrees = 500)

#Predict the test cases

rf\_predictions = predict(rf\_model,test.data)

#Create dataframe for actual and predicted values

df\_pred = cbind(df\_pred,rf\_predictions)

head(df\_pred)

#Calcuate MAE, RMSE, R-sqaured for testing data

print(postResample(pred = rf\_predictions, obs = test$Absenteeism.time.in.hours))

#Plot a graph for actual vs predicted values

plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="green")

lines(rf\_predictions,col="blue")

#RMSE Rsquared MAE

#0.1571539 0.5486220 0.1051845

########################################LINEAR REGRESSION########################################

#Train the model using training data

lr\_model = lm(Absenteeism.time.in.hours ~ ., data = train.data)

#Get the summary of the model

summary(lr\_model)

#Predict the test cases

lr\_predictions = predict(lr\_model,test.data)

#Create dataframe for actual and predicted values

df\_pred = cbind(df\_pred,lr\_predictions)

head(df\_pred)

#Calcuate MAE, RMSE, R-sqaured for testing data

print(postResample(pred = lr\_predictions, obs =test$Absenteeism.time.in.hours))

#Plot a graph for actual vs predicted values

plot(test$Absenteeism.time.in.hours,type="l",lty=2,col="green")

lines(lr\_predictions,col="blue")

# RMSE Rsquared MAE

#0.11901781 0.70385587 0.07940121

################