Steps Performed

# # Employee Attrition

# ---------------------------------------------------------------------------

# 1. Importing Files in R envirnonment

filepath = "C:/Users/tc186035/Desktop/DSP21\_2/RProject/R\_Project\_Attrition/Attrition.csv"

attrition =read.csv(filepath,header = T)

head(attrition,10)

# ---------------------------------------------------------------------------

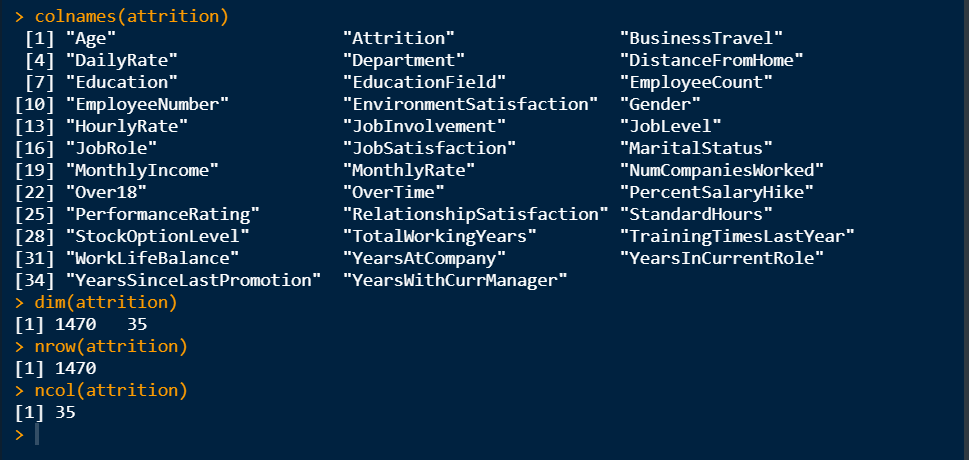
# 2. Data/feature analysis

colnames(attrition)

dim(attrition)

nrow(attrition)

ncol(attrition)



# 2.1 Structure of Datasets

str(attrition)

# Y Variable cosidered to be Attrition column which is of type Factor

# Removing unwanted columns

# Employee number is unique column so removing it

attrition$EmployeeNumber =NULL

attrition$StandardHours=NULL

attrition$EmployeeCount=NULL

attrition$Over18= NULL

# looking at data some columns can be converted to Factor Data

#

attrition$Education=as.factor(attrition$Education)

table(attrition$Education)

#

attrition$EnvironmentSatisfaction=as.factor(attrition$EnvironmentSatisfaction)

table(attrition$EnvironmentSatisfaction)

#

attrition$JobInvolvement=as.factor(attrition$JobInvolvement)

table(attrition$JobInvolvement)

#

attrition$JobLevel=as.factor(attrition$JobLevel)

table(attrition$JobLevel)

#

attrition$JobSatisfaction=as.factor(attrition$JobSatisfaction)

table(attrition$JobSatisfaction)

#

attrition$PerformanceRating=as.factor(attrition$PerformanceRating)

table(attrition$PerformanceRating)

#

attrition$RelationshipSatisfaction=as.factor(attrition$RelationshipSatisfaction)

table(attrition$RelationshipSatisfaction)

#

attrition$StockOptionLevel=as.factor(attrition$StockOptionLevel)

table(attrition$StockOptionLevel)

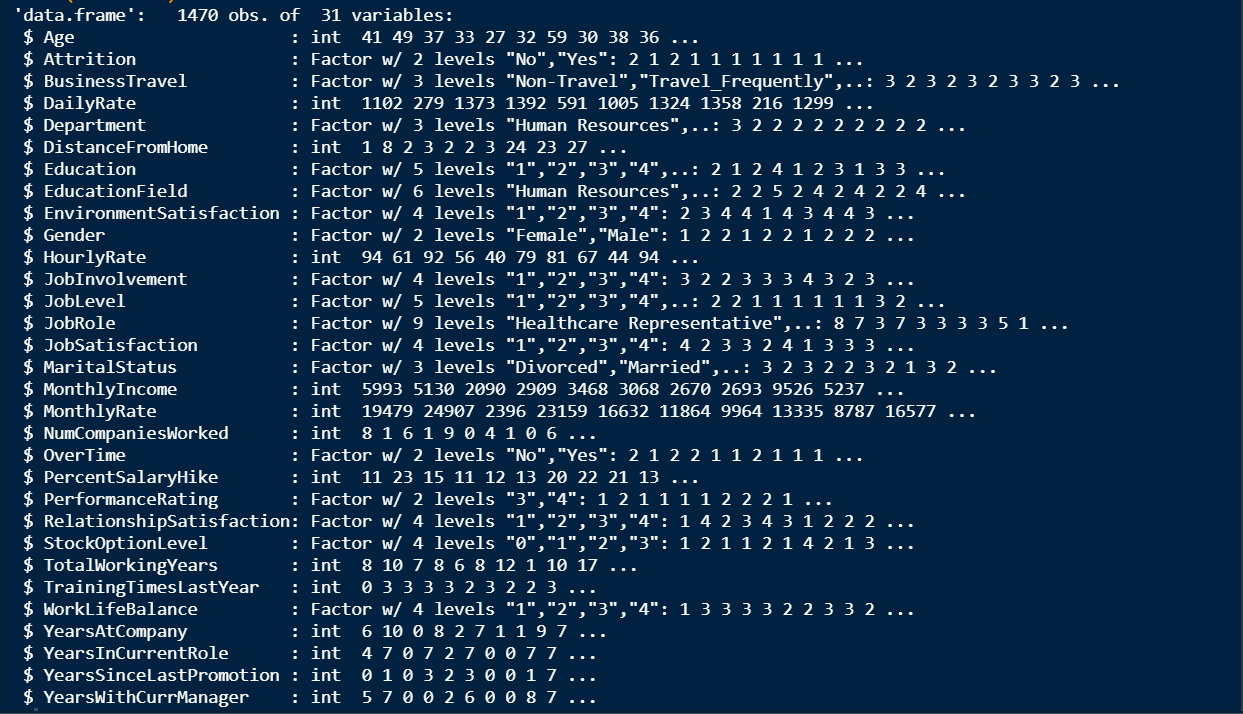
#

attrition$WorkLifeBalance = as.factor(attrition$WorkLifeBalance)

table(attrition$WorkLifeBalance)

prop.table(table(attrition$WorkLifeBalance))

str(attrition)

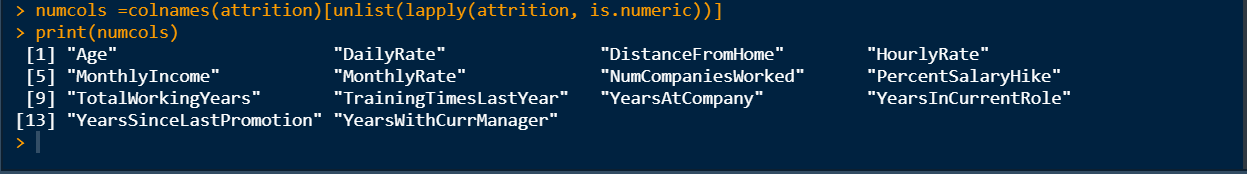


# ---------------------------------------------------------------------------

# 3. Seprate Numeric and Categorical Data.

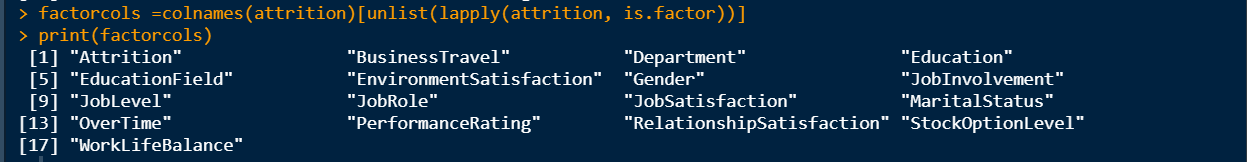
numcols =colnames(attrition)[unlist(lapply(attrition, is.numeric))]

print(numcols)



factorcols =colnames(attrition)[unlist(lapply(attrition, is.factor))]

print(factorcols)



# ---------------------------------------------------------------------------

# 4. Check for Zero in numerical Dataset

# Check for Null

checknull =function(x)

{

return(is.na(x))

}

nullcolnames =colnames(attrition)[apply(attrition, 2, checknull)]

if(length(nullcolnames)==0)

{

print(" No null in Data Sets")

}else

{

print(paste("Null colnames :",nullcolnames))

}

# NO NUll in data



# Check for Zero

checkzero =function(x)

{

return(any(x<=0))

}

zerocolnames =numcols[unlist(lapply(attrition[,numcols],checkzero))]

if(length(zerocolnames)==0)

{

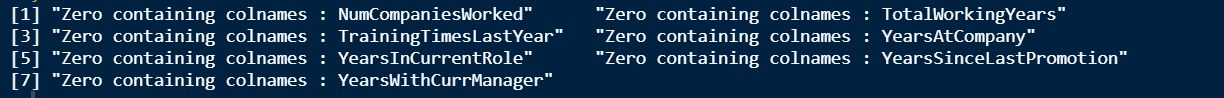
print(" No null in Data Sets")

}else

{

print(paste("Zero containing colnames :",zerocolnames))

}



# Handling Zero's in columns one by one

# i) NumCompaniesWorked-> it cannot be zero -> atleast it can be 1 company

n1min=min(attrition$NumCompaniesWorked)

n1max=max(attrition$NumCompaniesWorked)

n1mean=mean(attrition$NumCompaniesWorked)

n1meandin=median(attrition$NumCompaniesWorked)

n1meanfinal=n1meandin

attrition$NumCompaniesWorked[attrition$NumCompaniesWorked==0]=n1meanfinal

# ii)TotalWorkingYears-> it cannot be zero -> atleas it can be 1year or eperience can be in a month

t1min=min(attrition$TotalWorkingYears)

t1max=max(attrition$TotalWorkingYears)

t1mean=mean(attrition$TotalWorkingYears)

t1meadian=median(attrition$TotalWorkingYears)

# this column is Integer type

t1meanfinal=t1meadian

attrition$TotalWorkingYears[attrition$TotalWorkingYears==0]=t1meanfinal

# iii)YearsAtCompany-> it can be zero

# iv)YearsInCurrentRole-> it can be zero

# v)YearsSinceLastPromotion-> it can be zero

# vi)YearsWithCurrManager-> it can be zero

zerocolnames =numcols[unlist(lapply(attrition[,numcols],checkzero))]

print(zerocolnames)



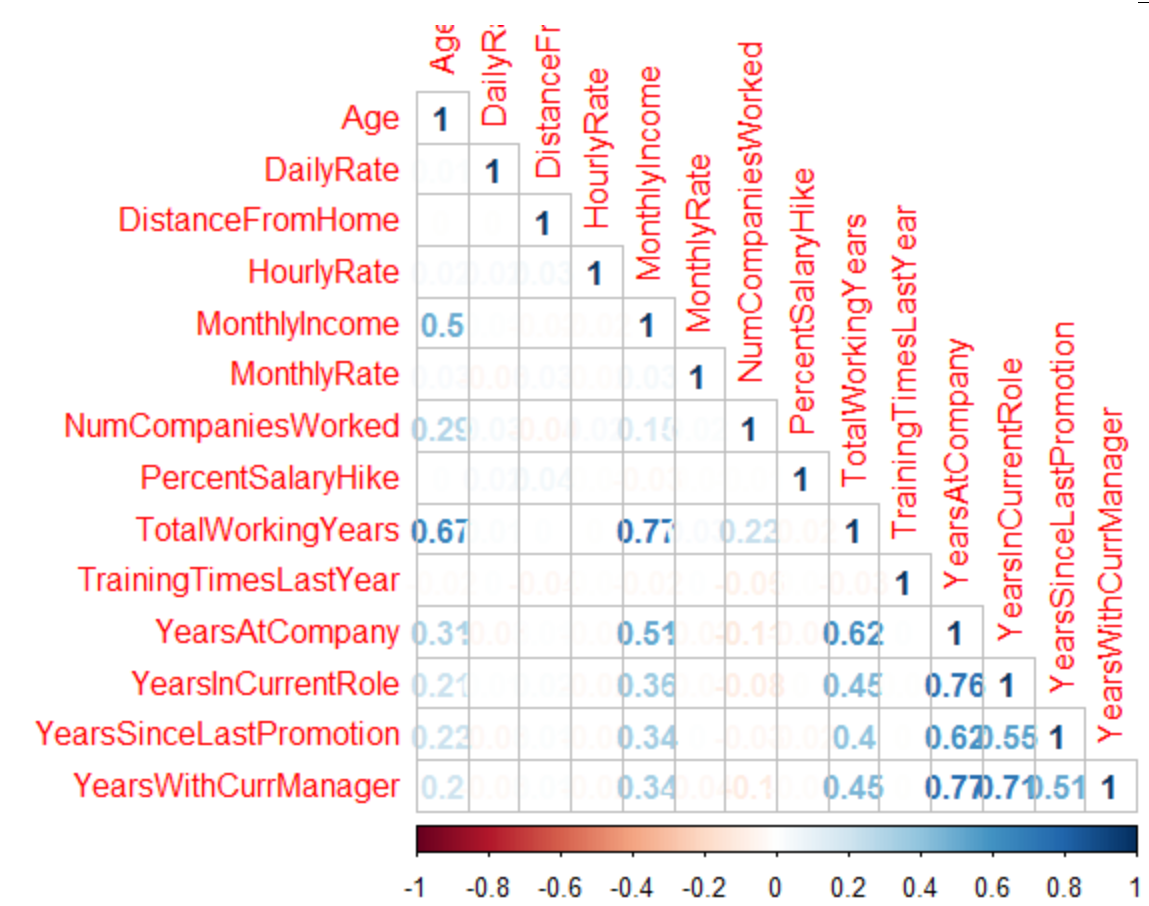
# ---------------------------------------------------------------------------

# 5.Multicolinearity

corr=cor(attrition[,numcols])

library(corrplot)

corrplot(corr,method='number',type='lower')



# ---------------------------------------------------------------------------

# 6. Move Y-Variable to last columns

print(attrition$Attrition)

attrition$Empattrition =attrition$Attrition

print(attrition$Empattrition)

attrition$Attrition =NULL

table(attrition$Empattrition)

View(attrition)

# ---------------------------------------------------------------------------

# 7. Outlier

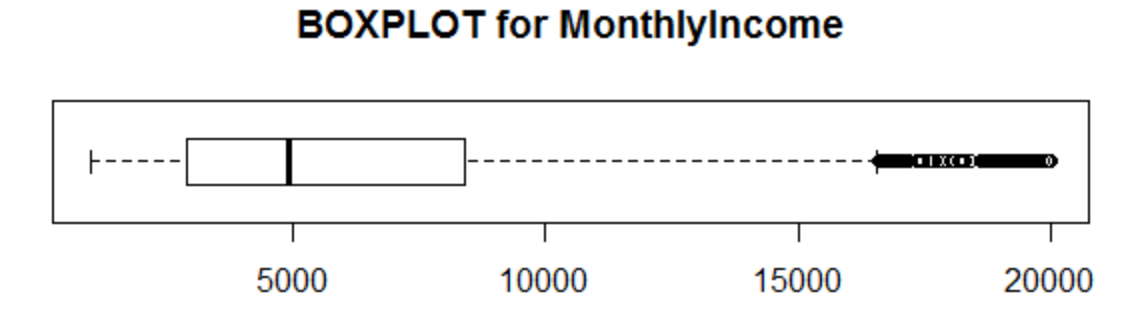
for(x in numcols)

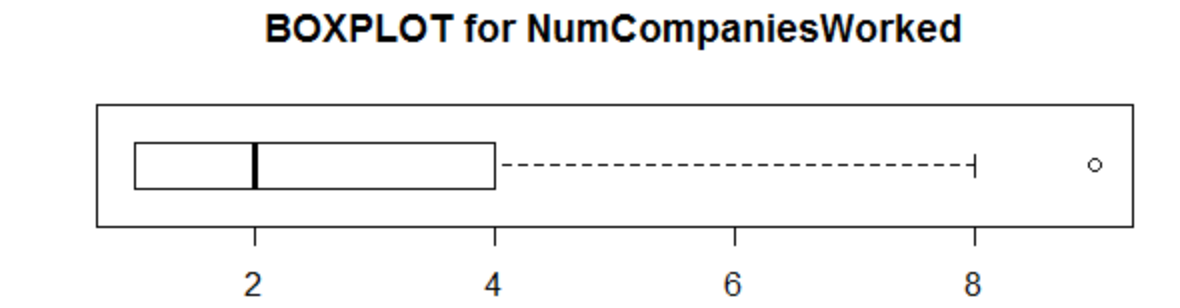
{

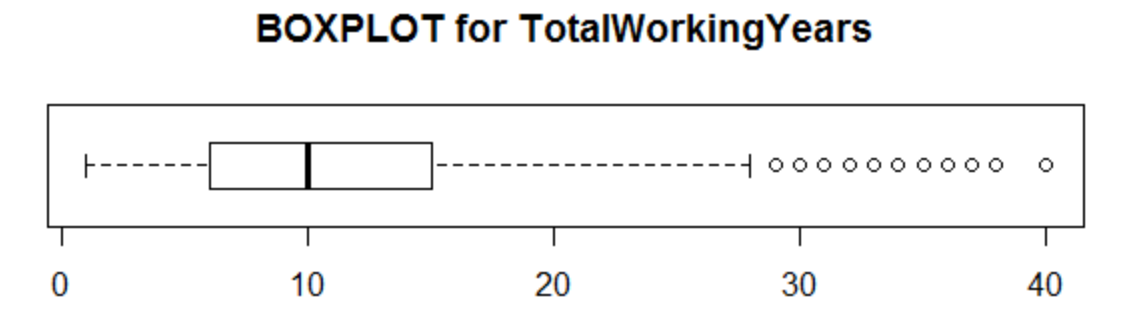
title =paste("BOXPLOT for",x)

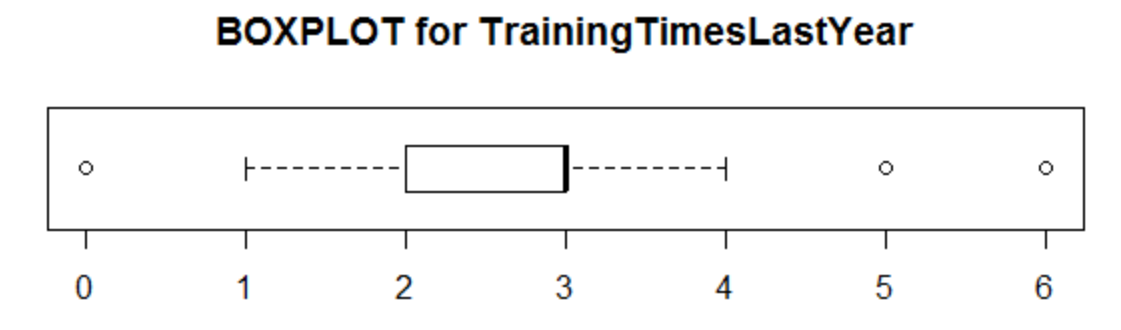
boxplot(attrition[,x],main =title,horizontal = T)

}









# Need to handle Outliers As needed

# MonthlyIncome -> it can have outlier

# NumCompaniesWorked -> perople can change the companiews No of time(can have outlier)

# TotalWorkingYears -> it may varies person to person

# TrainingTimesLastYear -> this outlier can be handle for less traing hours

# YearsAtCompany -> it may varies person to person

# YearsInCurrentRole -> should be less than or equal to YearsAtCompany

# YearsSinceLastPromotion -> this outlier can be handle for more then year

# YearsWithCurrManager -> should be less than or equal to YearsAtCompany

impute\_outliers <- function(x)

{

quantiles = quantile( x, c(0.25, 0.75 ))

x[ x < quantiles[1] ] = mean(x )

x[ x > quantiles[2] ] = mean(x)

x

}

#attrition$MonthlyIncome = impute\_outliers(attrition$MonthlyIncome)

attrition$TotalWorkingYears = impute\_outliers(attrition$TotalWorkingYears)

attrition$TrainingTimesLastYear = impute\_outliers(attrition$TrainingTimesLastYear)

attrition$YearsAtCompany = impute\_outliers(attrition$YearsAtCompany)

attrition$YearsInCurrentRole = impute\_outliers(attrition$YearsInCurrentRole)

attrition$YearsSinceLastPromotion = impute\_outliers(attrition$YearsSinceLastPromotion)

attrition$YearsWithCurrManager = impute\_outliers(attrition$YearsWithCurrManager)

# ---------------------------------------------------------------------------

# 8.Distribution

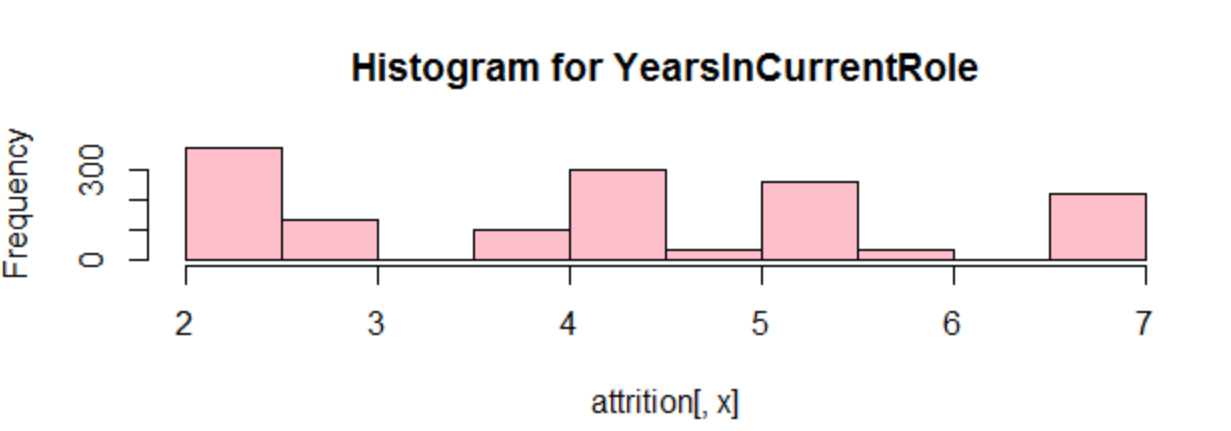
for(x in numcols)

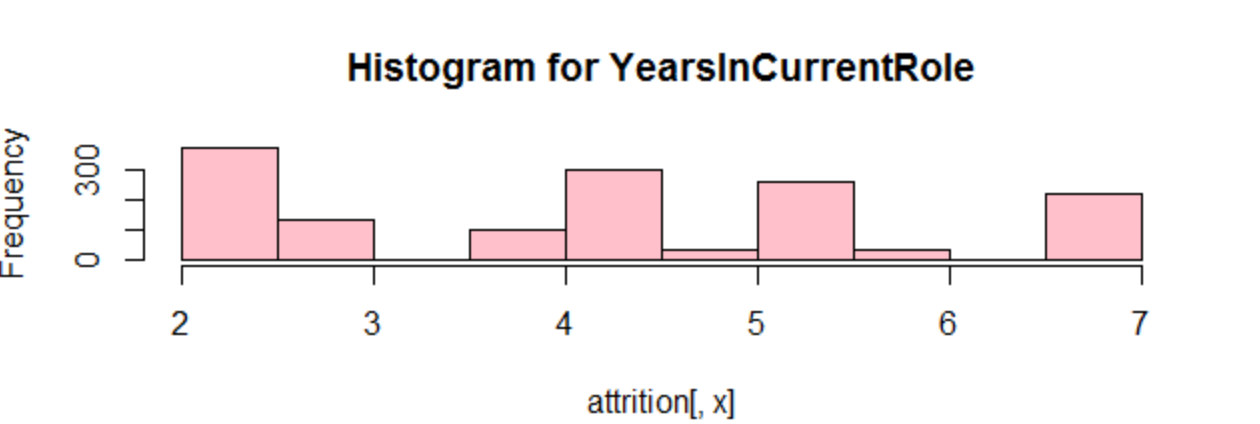
{

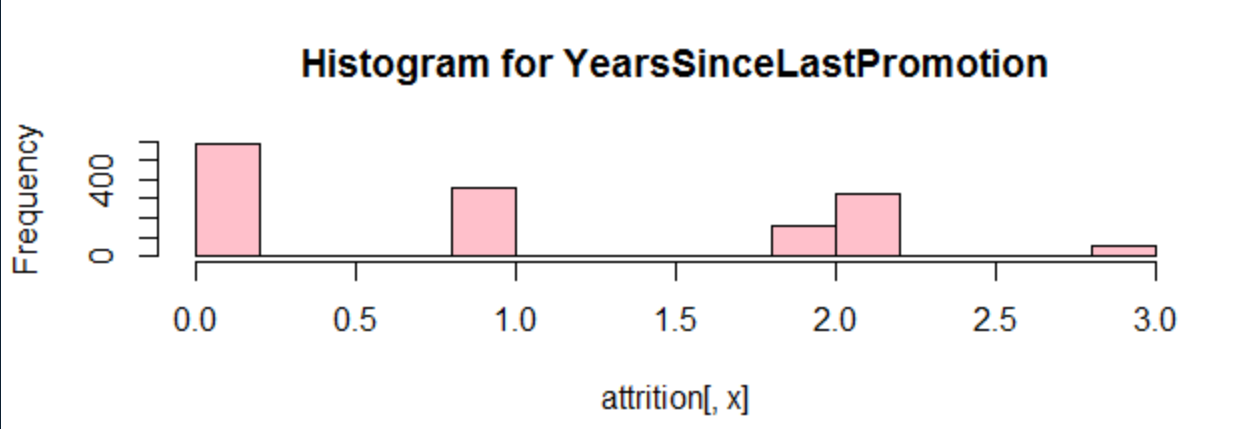
title =paste("Histogram for",x)

hist(attrition[,x],main =title ,col ="pink")

}







# ---------------------------------------------------------------------------

# 9. split data

rows=nrow(attrition)

s=sample(seq(1,rows),0.7\*rows)

train=attrition[s,]

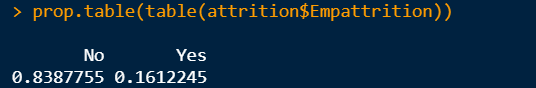
test=attrition[-s,]

print(paste('train :',nrow(train),'test :',nrow(test)))



# check class distribution of class

prop.table(table(attrition$Empattrition))



levels(factor(attrition$Empattrition))



train\_level=length(levels(factor(train$Empattrition)))

test\_level =length(levels(factor(test$Empattrition)))

if(train\_level>=test\_level)

{

print("Traing and Testing level are ok")

}else

{

print("Testing data have more levels than traing..Data need to sample Again")

}



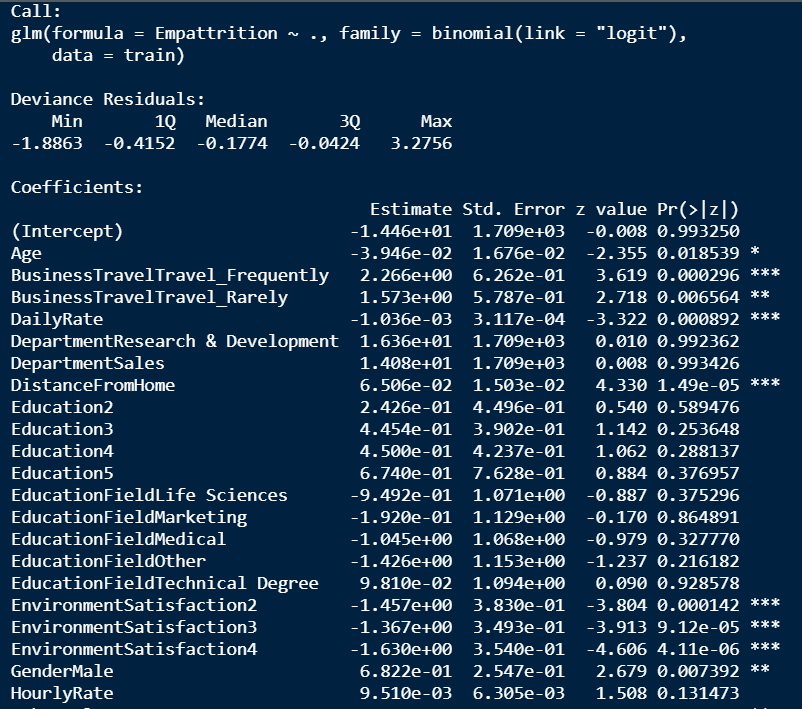
# ---------------------------------------------------------------------------

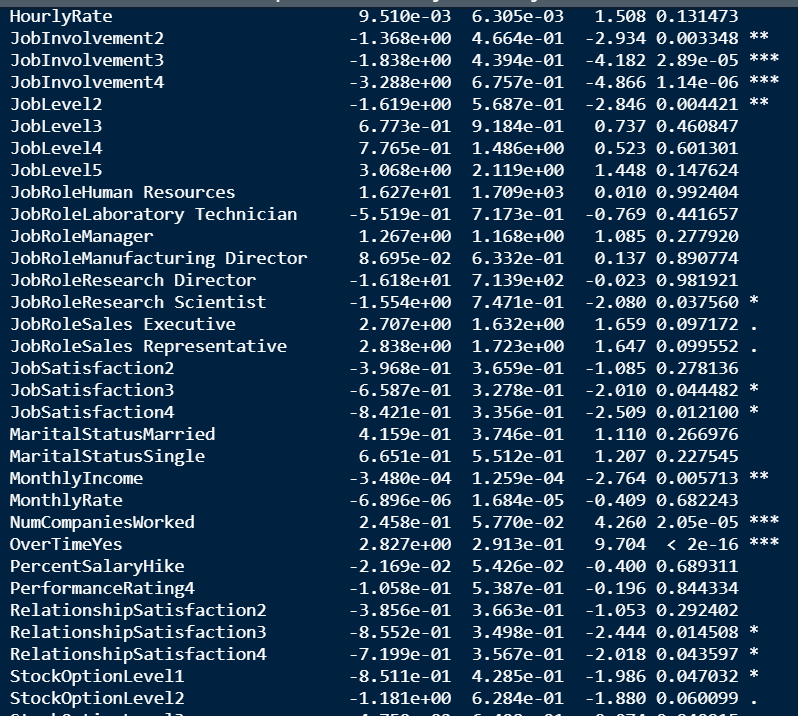
# 10. Model Building

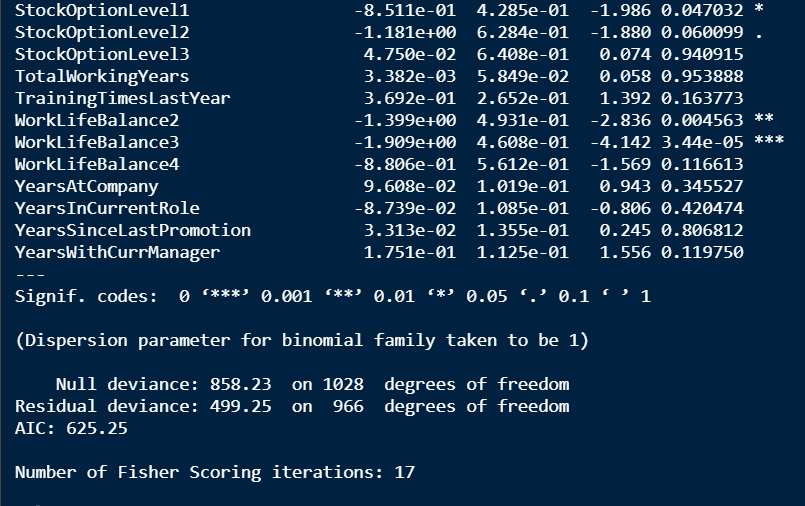
#m1=glm(Empattrition~.,data = train,gaussian(link = "identity"))

m1 =glm(Empattrition~. , data = train,binomial(link = "logit"))

summary(m1)







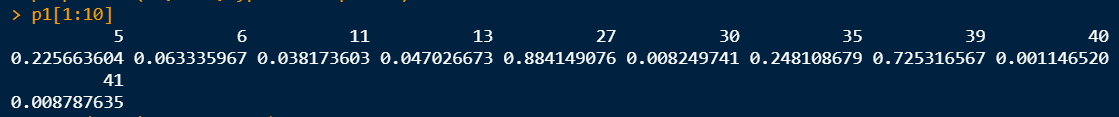
#logitstic regression gives you log oods -> exp(ordss) gives you odds

# ---------------------------------------------------------------------------

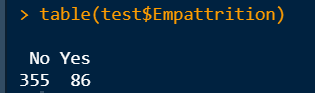
# 11.prediction

p1=predict(m1,test,type = "response")

p1[1:10]



table(test$Empattrition)



length(p1[p1<=0.5])

length(p1[p1>=0.5])

length(p1)

#convertin likhood estimated into classes 0/1

pred1 =as.factor(ifelse(p1<=0.5,'No','Yes'))

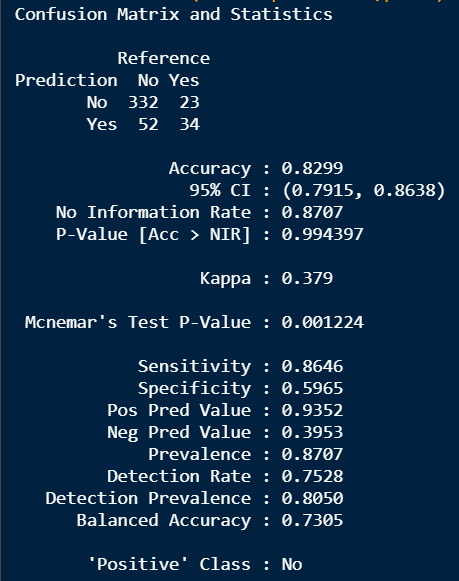
print(pred1[1:10])

cbind(p1[1:10],pred1[1:10])

library(ggplot2)

library(caret)

confusionMatrix(test$Empattrition,pred1)



# ---------------------------------------------------------------------------

# 12. Model 2 with Limited Attributes

m2data=attrition

m2data$DailyRate=NULL

m2data$EducationField=NULL

m2data$HourlyRate=NULL

m2data$MaritalStatus=NULL

m2data$PercentSalaryHike=NULL

rows=nrow(m2data)

s=sample(seq(1,rows),0.7\*rows)

train2=m2data[s,]

test2=m2data[-s,]

print(paste('train :',nrow(train2),'test :',nrow(test2)))

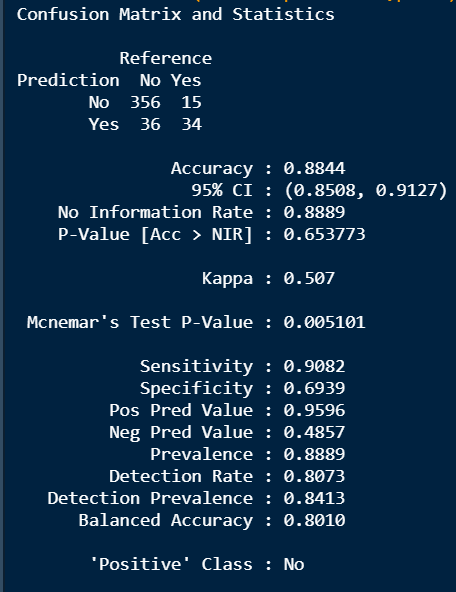
m2 =glm(Empattrition~. , data = train2,binomial(link = "logit"))

summary(m2)

p2=predict(m2,test2,type = "response")

pred2 =as.factor(ifelse(p2<=0.5,'No','Yes'))

confusionMatrix(test2$Empattrition,pred2)



# ---------------------------------------------------------------------------

# 12. Model 3 with oversampling for negative Class

library(ROSE)

m3data=attrition

nrow(m3data)

rows=nrow(m3data)+1000

print(paste("oversample rows",rows))

over\_m3data =ovun.sample(Empattrition~.,data = m3data,method= "over",N=rows,seed =1)$data

# Over Sampling check

table(over\_m3data$Empattrition)

prop.table(table(over\_m3data$Empattrition))

o\_rows=nrow(over\_m3data)

s=sample(seq(1,rows),0.7\*o\_rows)

train3=over\_m3data[s,]

test3=over\_m3data[-s,]

print(paste('train :',nrow(train3),'test :',nrow(test3)))

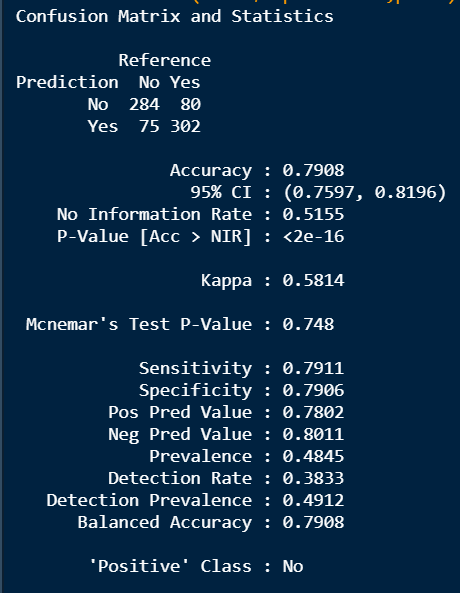
m3 =glm(Empattrition~. , data = train3,binomial(link = "logit"))

summary(m3)

p3=predict(m3,test3,type = "response")

pred3 =as.factor(ifelse(p3<=0.5,'No','Yes'))

confusionMatrix(test3$Empattrition,pred3)



# ---------------------------------------------------------------------------

# 13. Model 4 with Decision Tree Alogorithm.

library(rpart)

library(rpart.plot)

library(caret)

library(rattle)

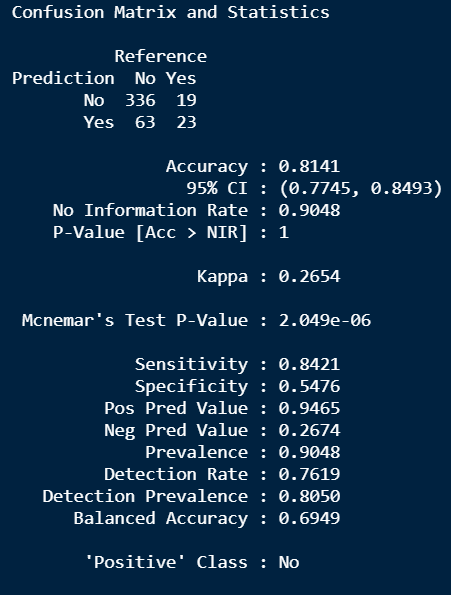
library(RColorBrewer)

m4dt = rpart(Empattrition~.,data = train,method = "class")

rpart.plot(m4,type = 4,extra = 101,box.palette = "GnBu" ,branch.lty=3,shadow.col="gray",nn=T)

p4=predict(m4dt,test,type="class")

confusionMatrix(test$Empattrition,p4)



# ---------------------------------------------------------------------------