

Predicting mode of Transport (ML)



SUBMITTED BY – SANKALP RAJ MATTA

CONTENT:

1. Project Objective
2. Data Analysis
   1. Descriptive Data Analysis
   2. Exploratory Data Analysis
      1. Univariate Data Analysis
      2. Bivariate Data Analysis
   3. Summary of insights from Data Analysis
3. Check for Integrity of Data
   1. Outlier Detection
   2. Missing Value Detection
4. Multicollinearity Detection
   1. Correlation Matrix
   2. Treating Multicollinearity
5. SMOTE Data Preparation
   1. Splitting the Dataset into Train and Test
   2. Applying SMOTE on the Train Dataset
6. Logistic Regression
   1. Applying Logistic Regression
      1. Creating Logistic Model
      2. Variance Inflation Factor
      3. Predictions + Confusion Matrix On Train Data
      4. Predictions + Confusion Matrix On Test Data
   2. Interpretation of Logistic Regression
7. KNN Model
   1. Applying KNN Model
   2. Interpretation of KNN Model
8. Naïve Bayes’
   1. Applying Naïve Bayes’
   2. Interpretation Of Naïve Bayes’
9. Interpretation Of Confusion Matrix
   1. For Logistic Regression
   2. For KNN Model
   3. For Naïve Bayes’
10. Remarks on the Best Performing Model
11. Bagging Ensemble Method
    1. Applying Bagging Technique
    2. Interpretation of Bagging
12. Boosting Ensemble Method
    1. Converting the Data set into numeric matrices for application
    2. Creating an Initial Model by Boosting
    3. Applying Iteration for tuning the Boosting model
    4. Creating a final model
13. Actionable Insights and Recommendations

# 1. PROJECT OBJECTIVE

This project we attempt to understand what mode of transport employees prefers to commute to their office. We are given a dataset that includes employee information about their mode of transport as well as their personal and professional details like age, salary, work exp. We need to predict whether or not an employee will use Car as a mode of transport and also interpret which variables are a significant predictor behind this decision.

## 2.DATA ANALYSIS

It is a culmination of Descriptive and Exploratory Data Analysis. It is done to understand the basic data structure as well as to visualize the dataset before performing any modelling.

It checks for the relationship between the variables through the use of graphs. And also gives the summary and the class of each variable.

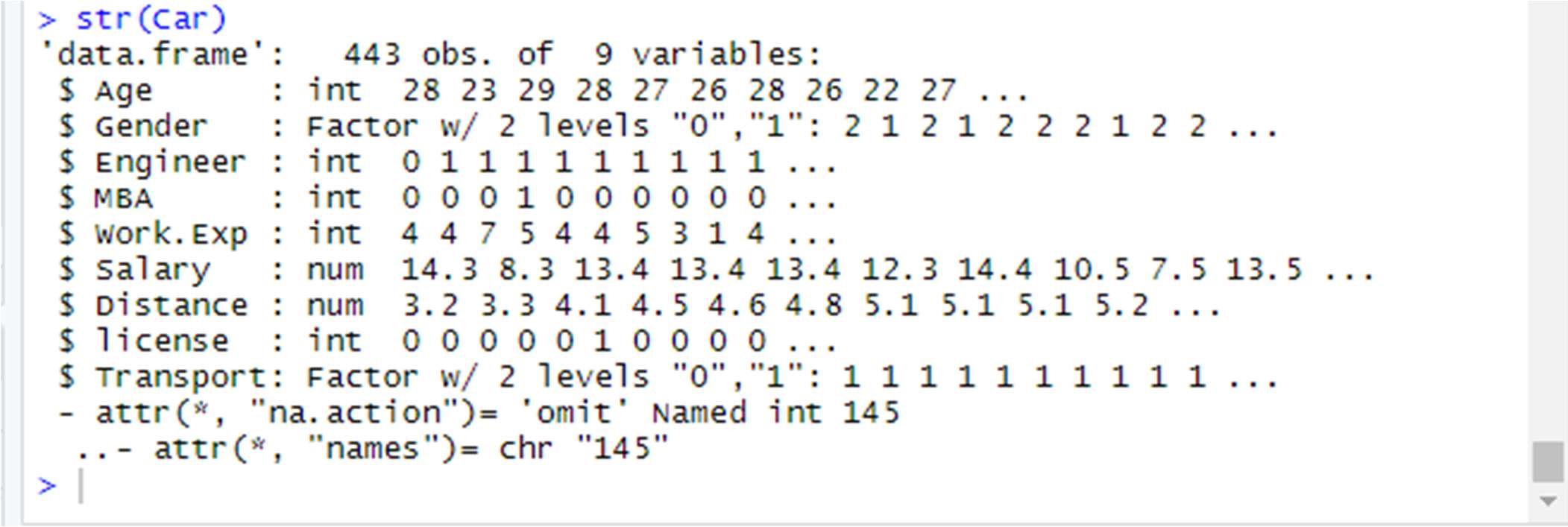
## 2.1 DESCRIPTIVE DATA ANALYSIS

Car= read.csv("Cars\_edited.csv")

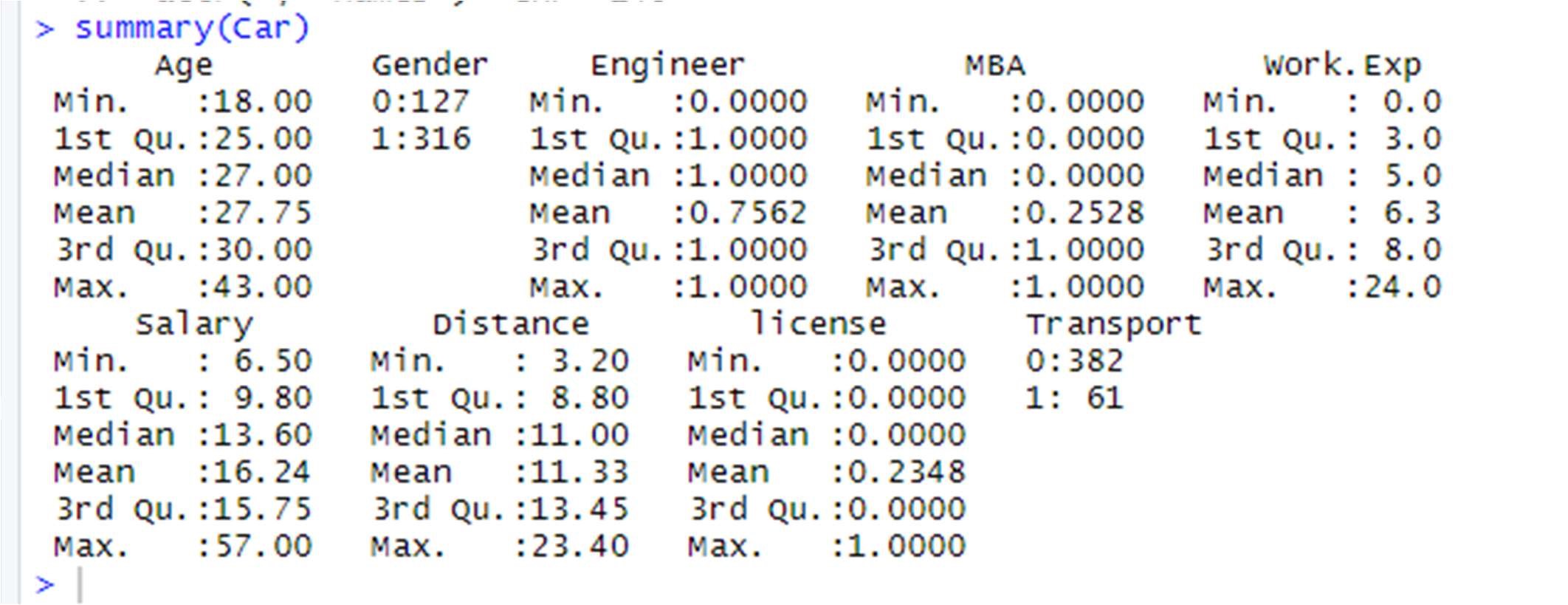
str(Car)

summary(Car) dim(Car)

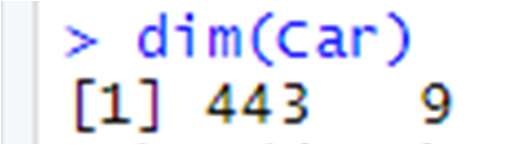
Structure Of Dataset:



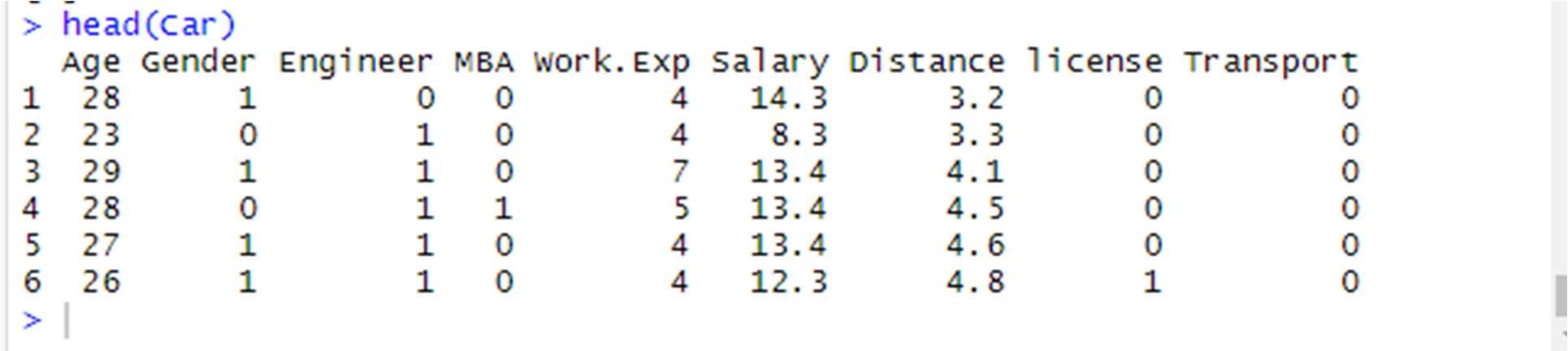
Summary Of Dataset:



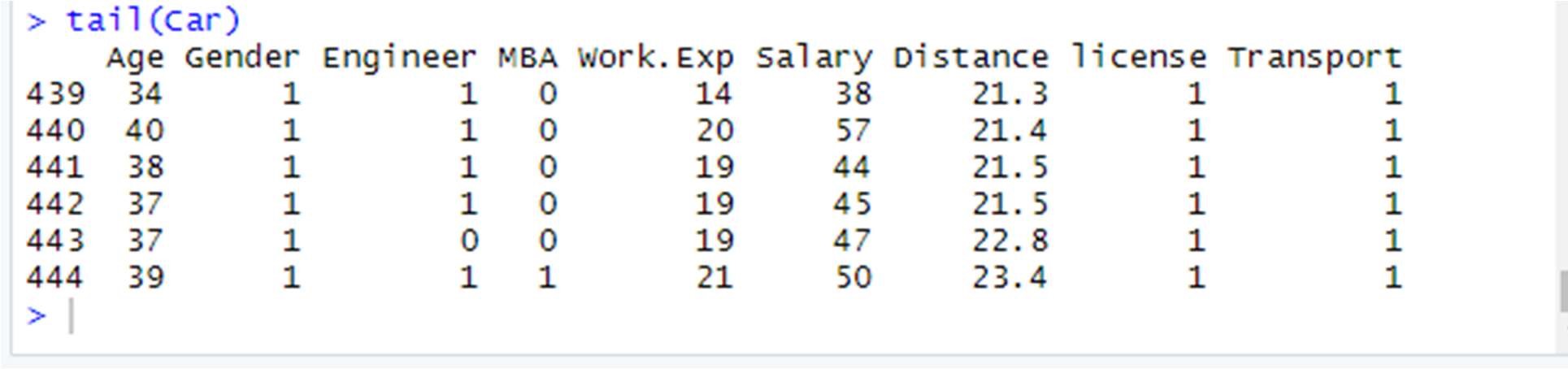
### Dim- will tell the dimension of our dataset



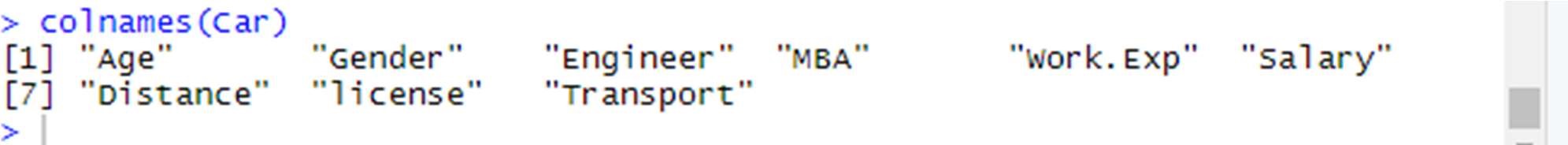
Head- will fetch the first 10 records of the dataset



Tails- will fetch the last 10 records of the dataset



Name of all columns:



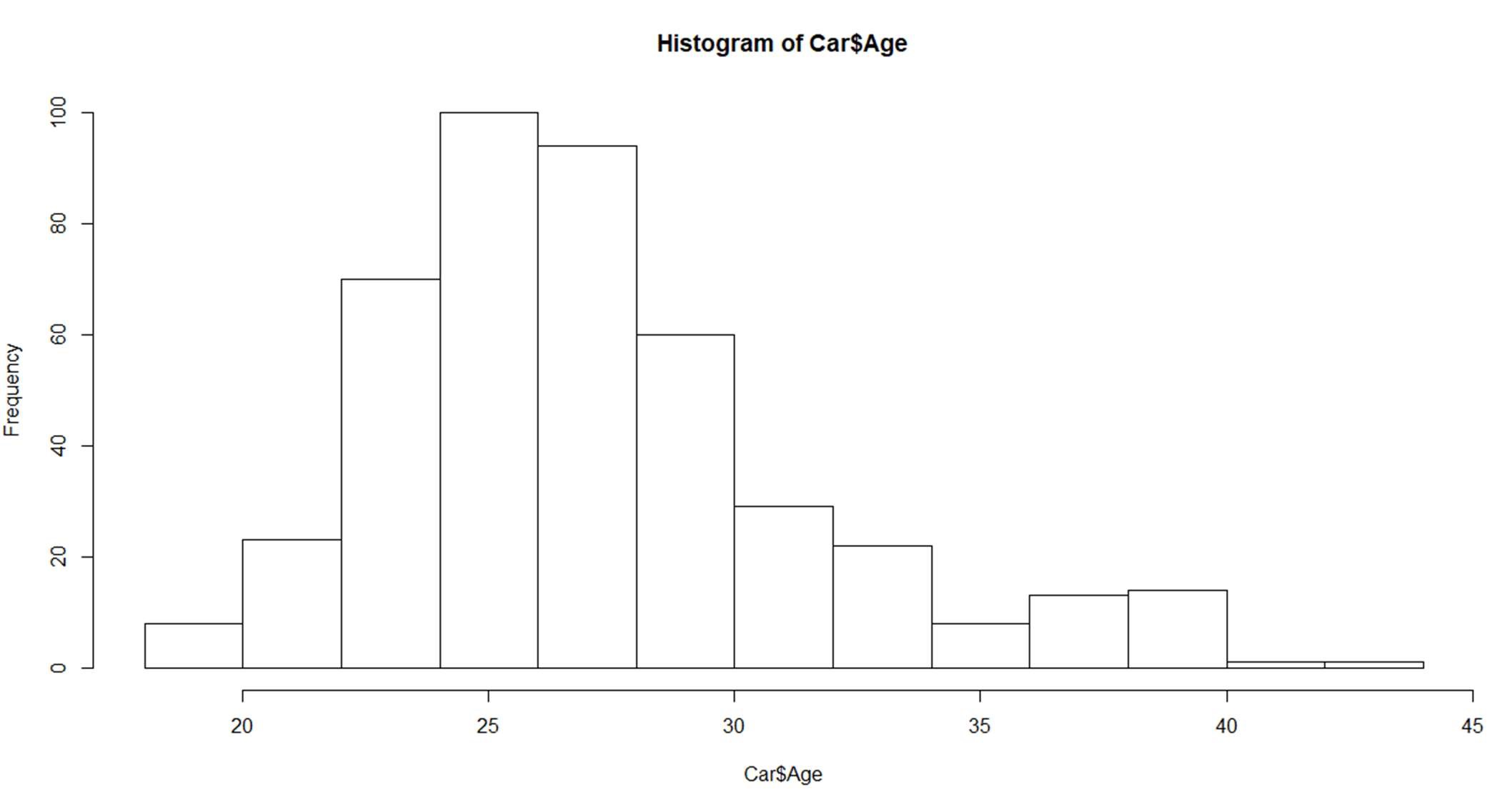
### 2.2.1 UNIVARIATE ANALYISIS

Done for both categorical and continuous variables

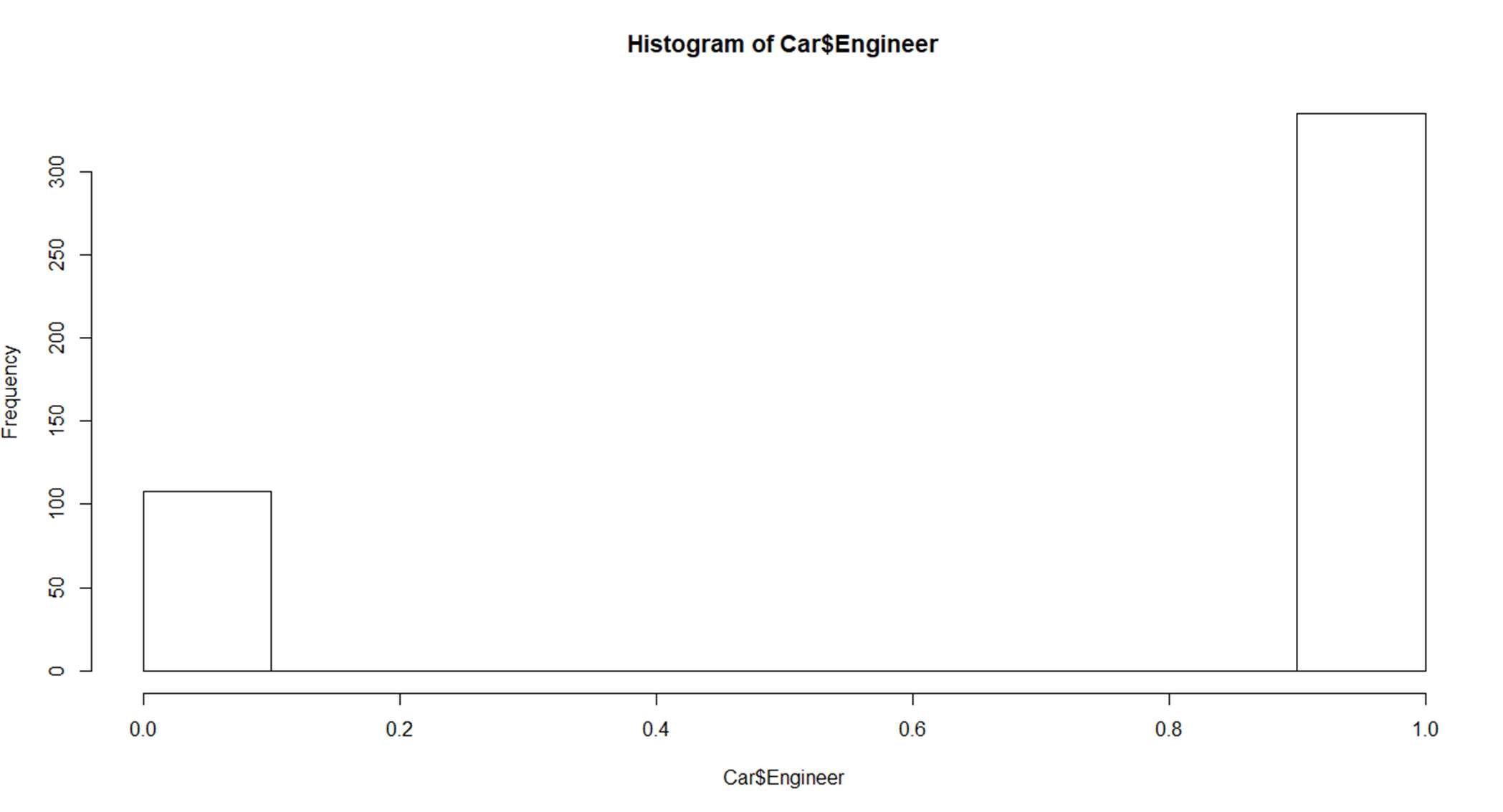
Must be numeric for histogram plotting

\*HISTOGRAM

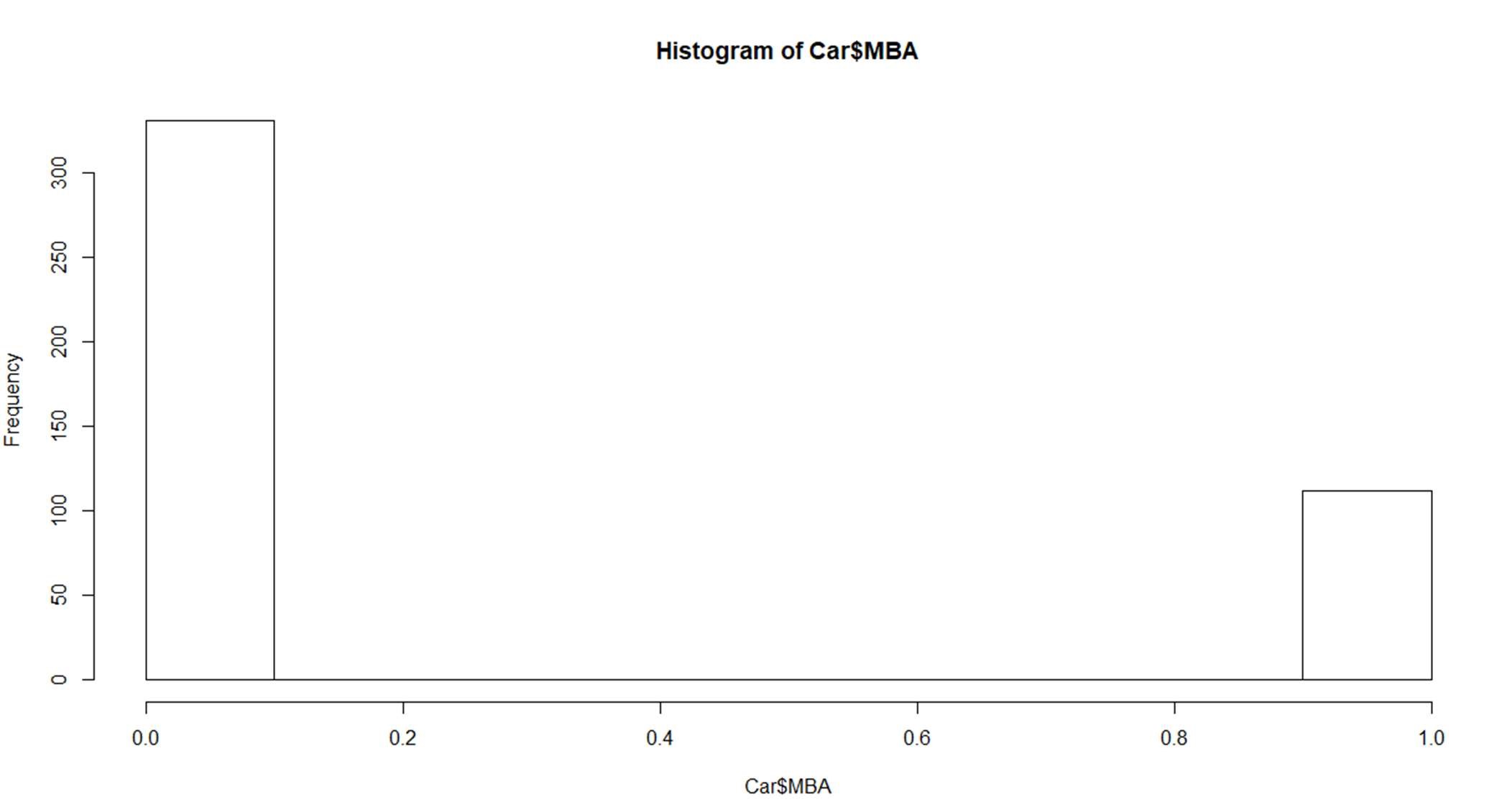
hist(Car$Age)



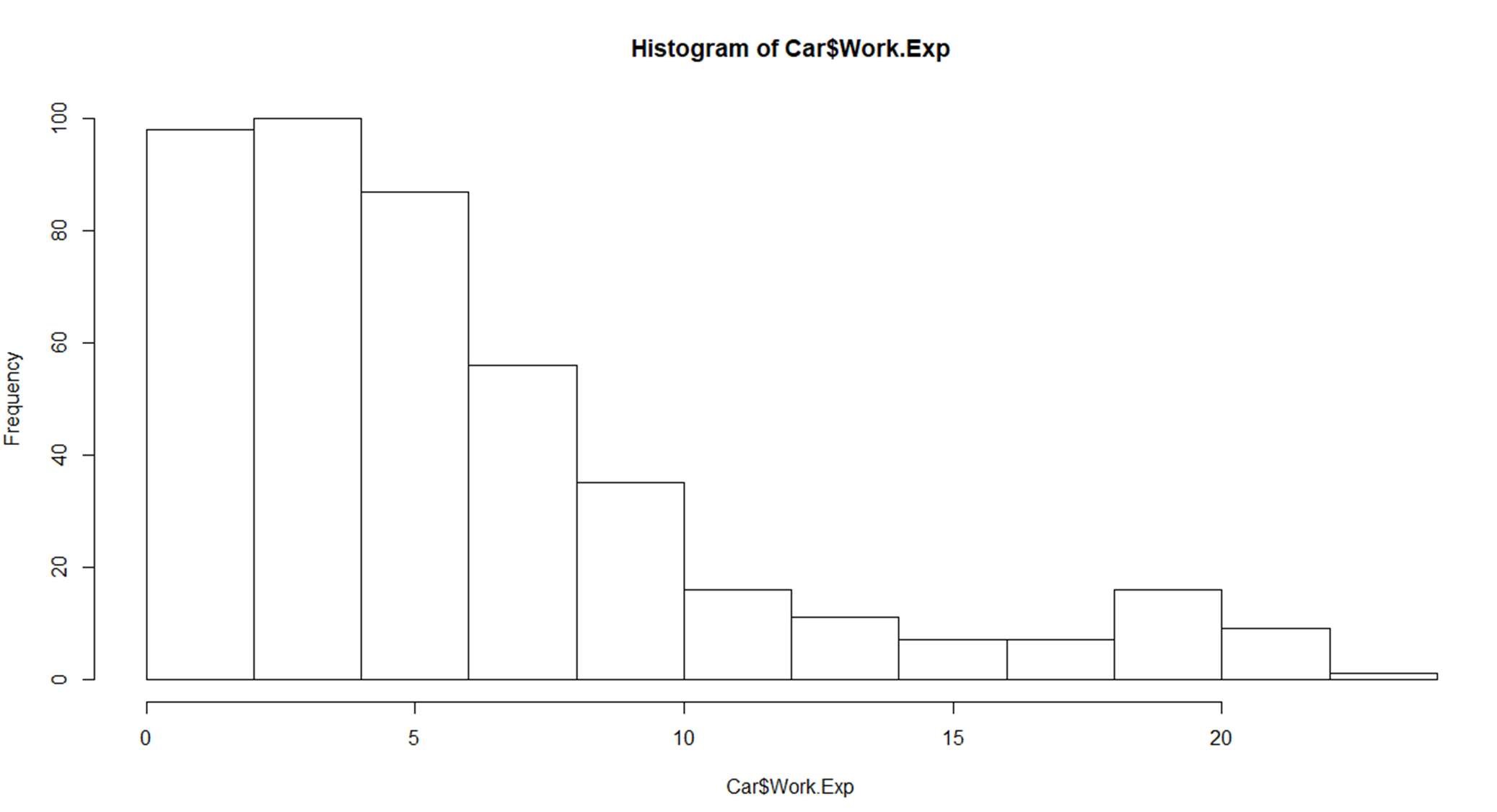
hist(Car$Engineer)

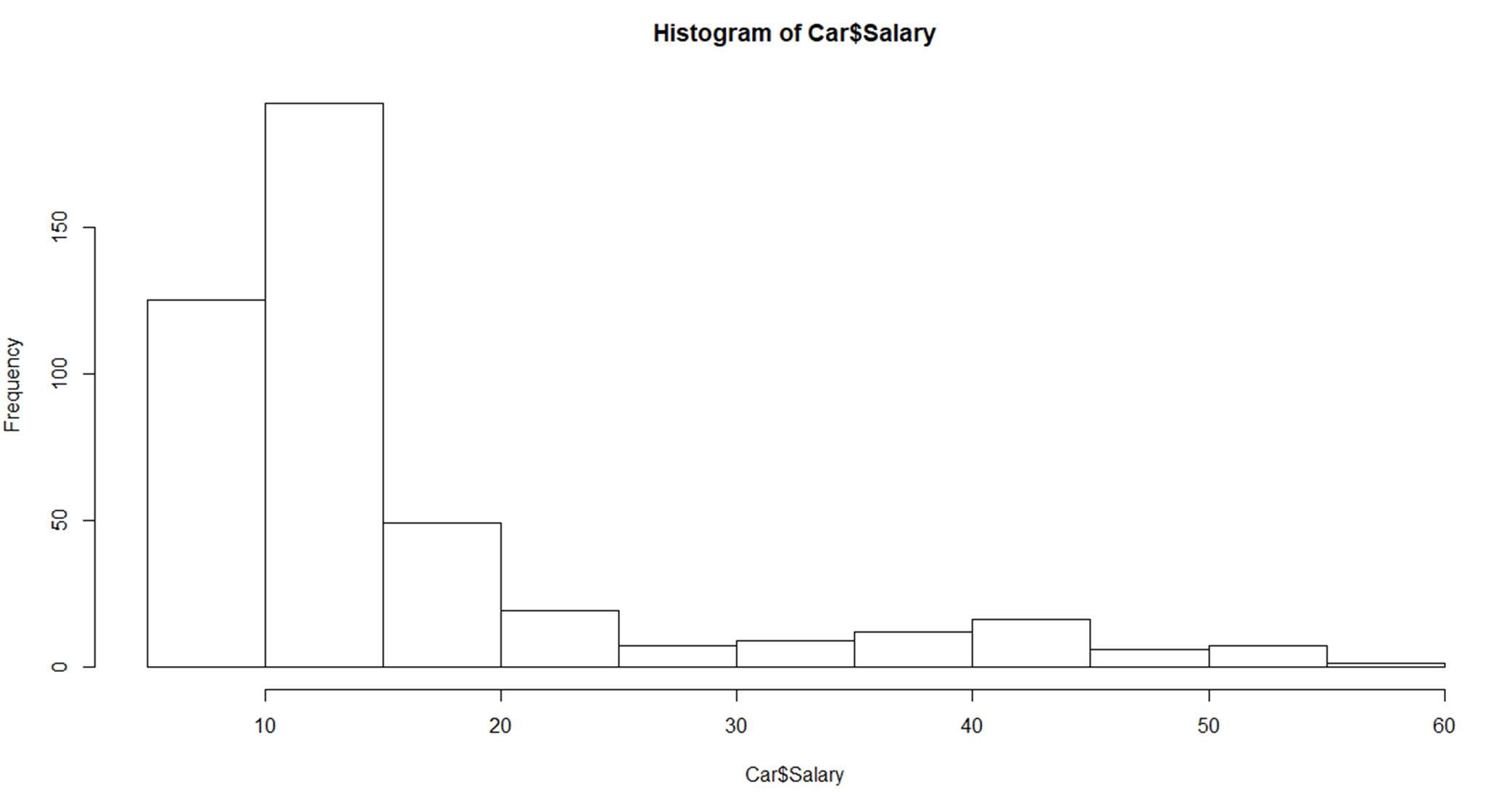


hist(Car$MBA)

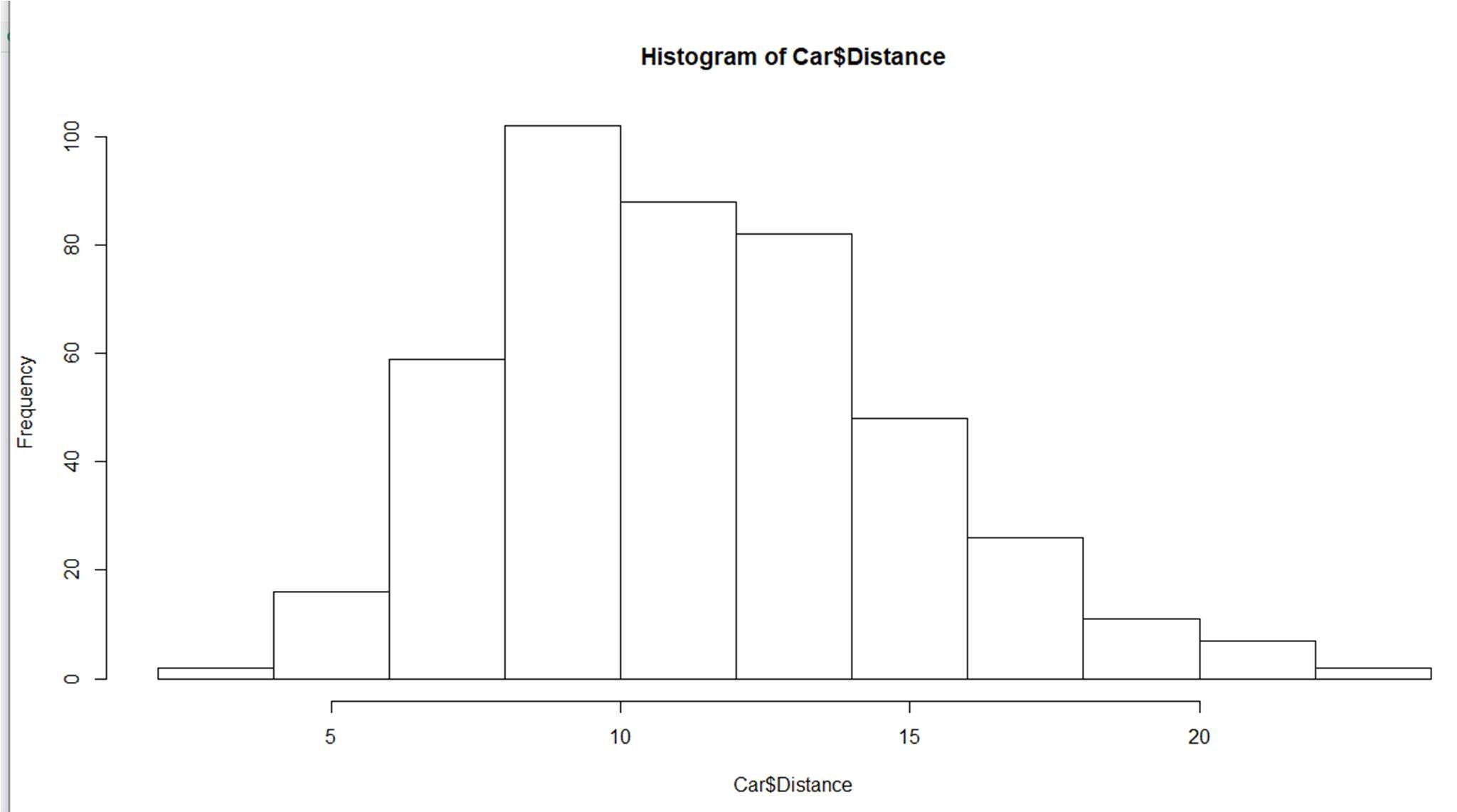


hist(Car$Work.Exp)

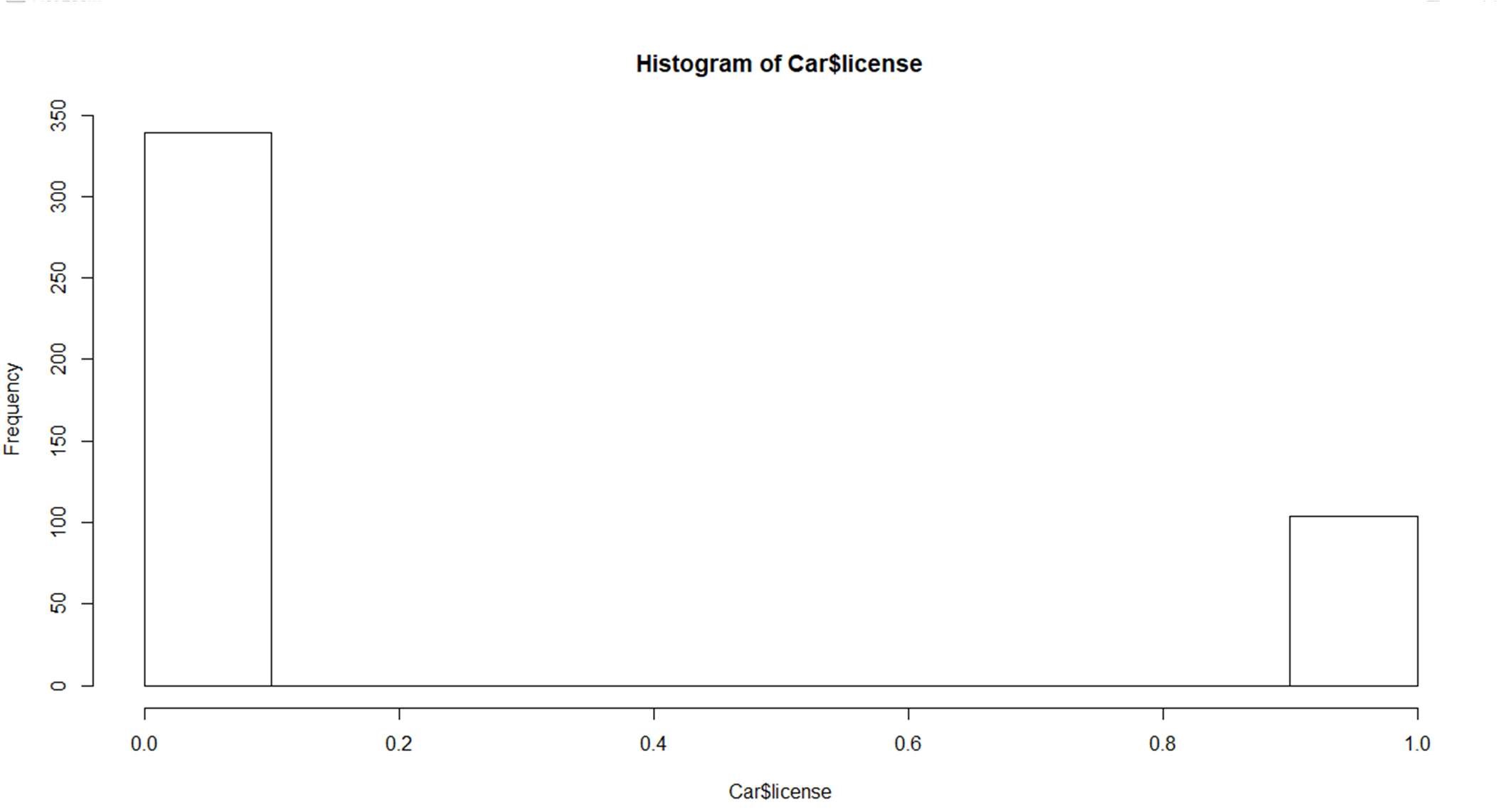
 hist(Car$Salary)



hist(Car$Distance)



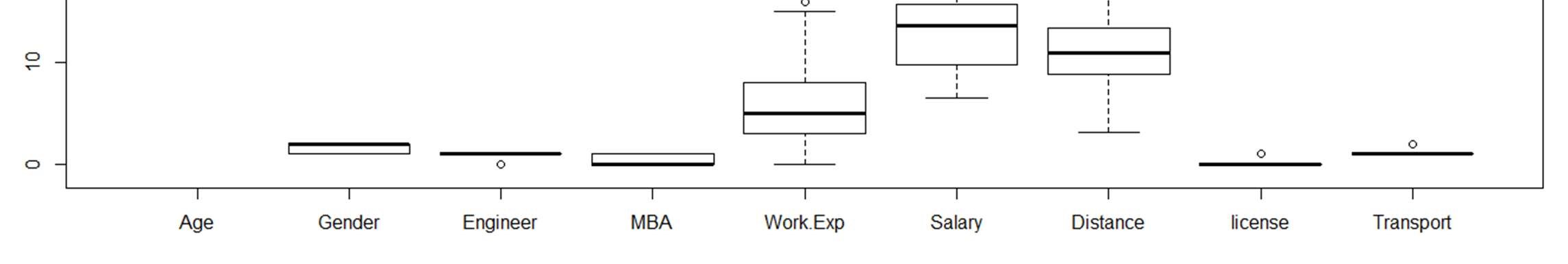
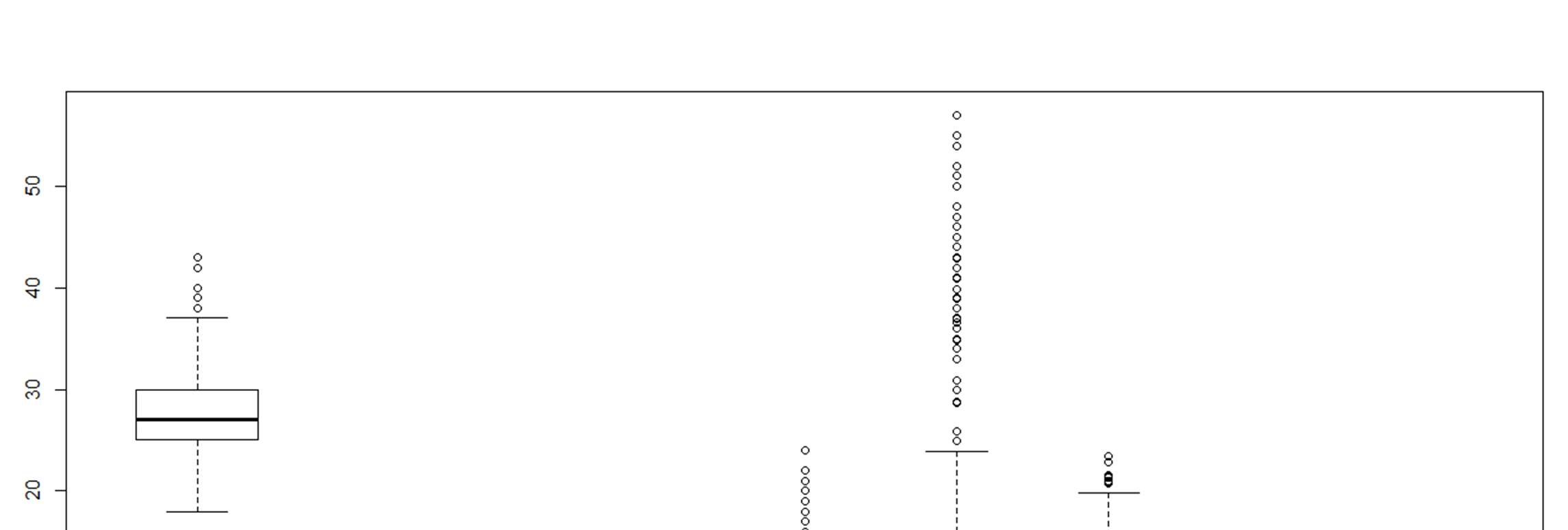
hist(Car$license)



\*BOXPLOT

boxplot(Car)

This function will make the boxplots of all the variables individually on a single chart.

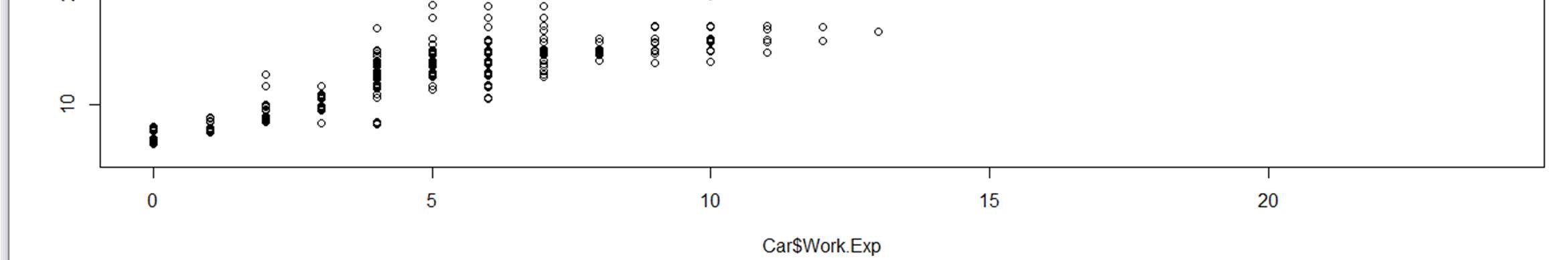
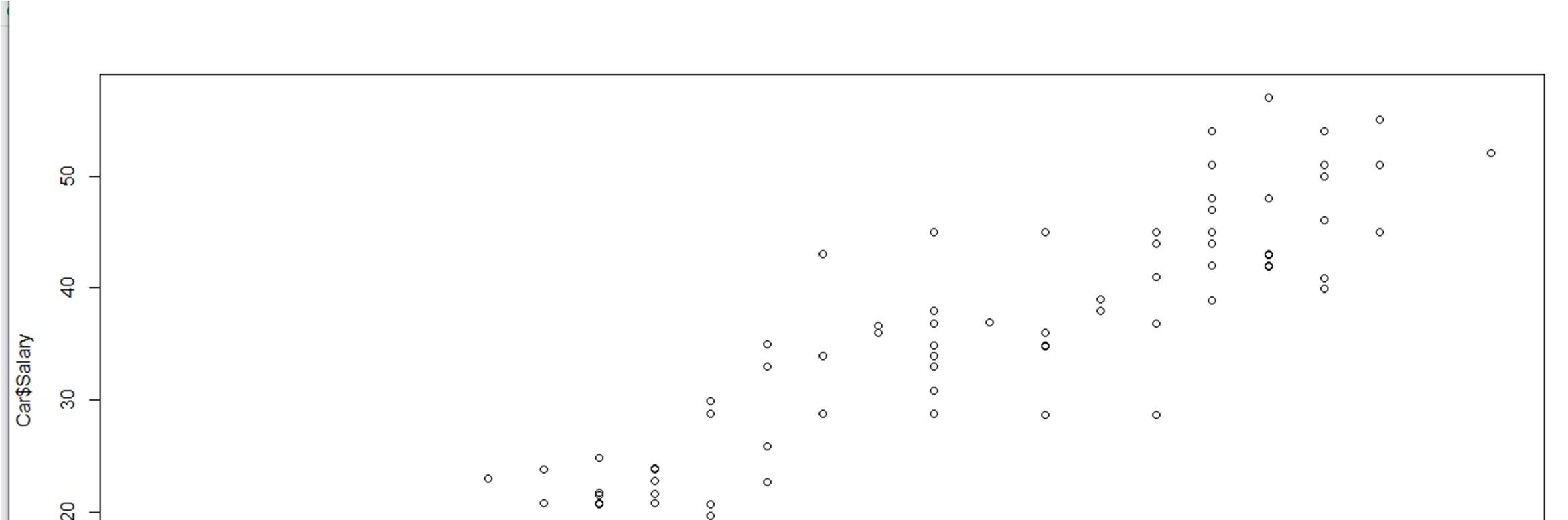


There are outliers in this case. But we are not removing them right now because or data set is small and each row holds great value. We are dealing with an unbalanced dataset in which the minority class is important and is very small in comparison to the majority class.

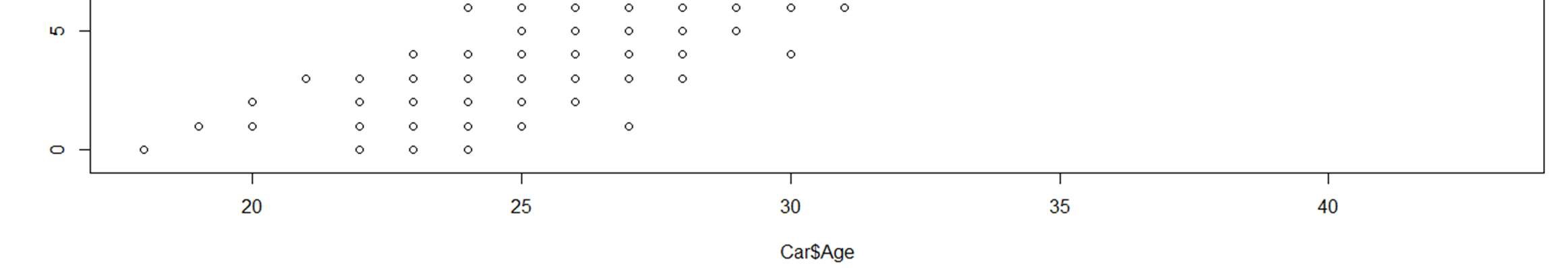
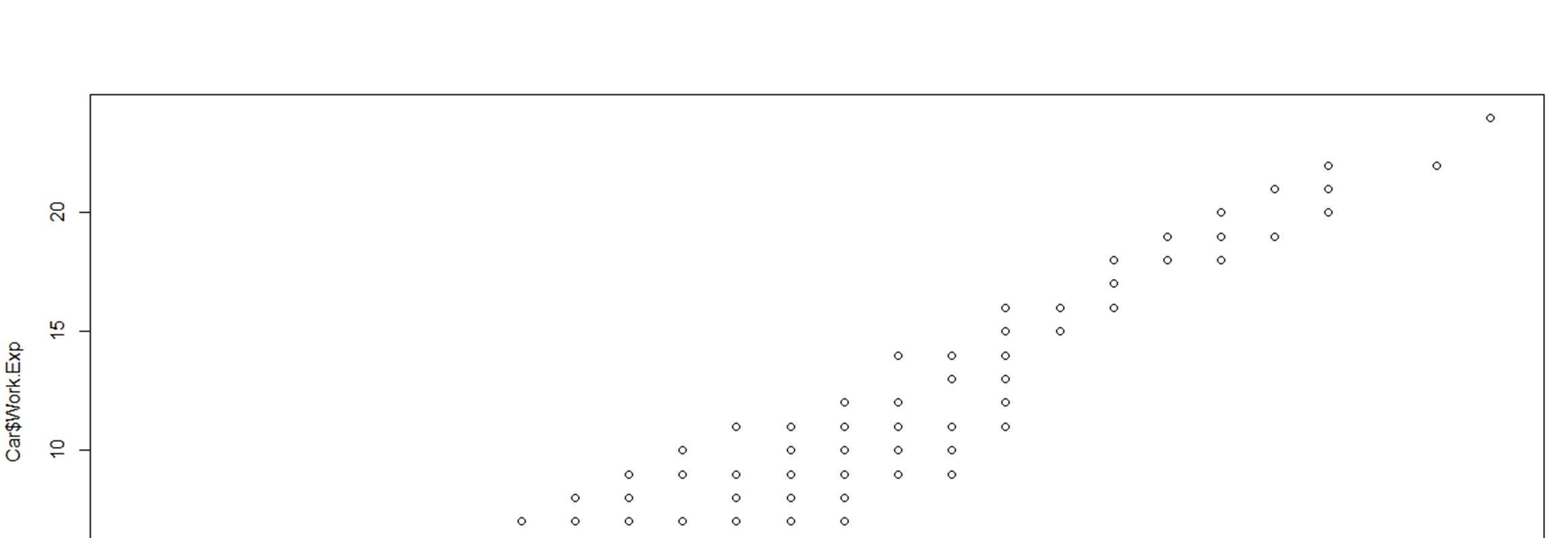
#### 2.2.2 BIVARIATE ANALYSIS

By default these plots will be scatterplots(for bivariate numerical variables).

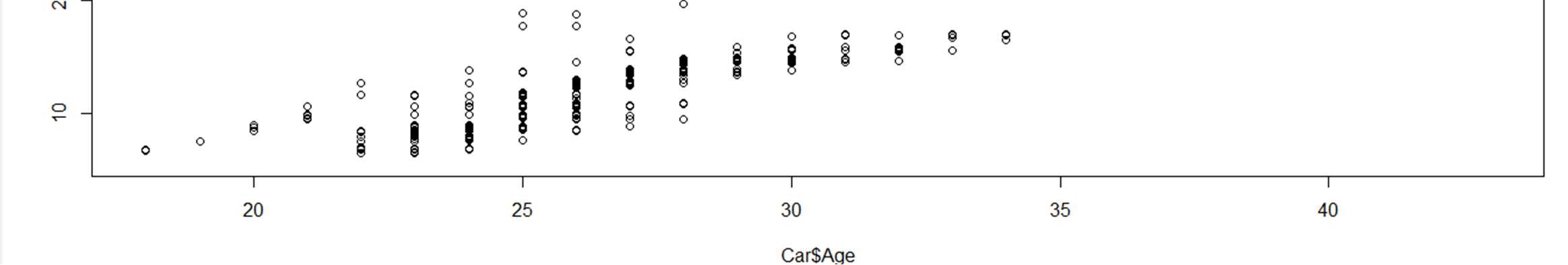
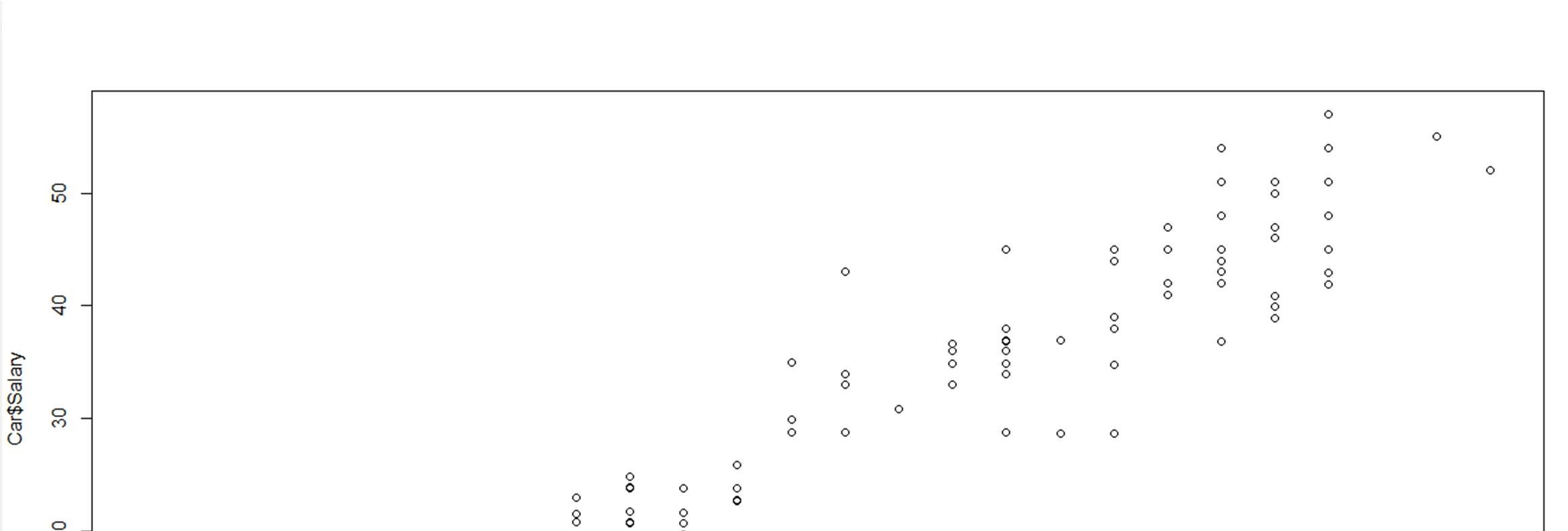
plot(Car$Work.Exp,Car$Salary)



plot(Car$Age,Car$Work.Exp)



plot(Car$Age,Car$Salary)



## 2.3 SUMMARY OF THE INSIGHTS FROM DATA ANALYSIS

The data consists of all numeric, integer variables, and factor variables.

\*There is the issue of multicollinearity amongst the variables.

\*There are outliers in certain variable rows: Age, Engineering, Work.Exp, Salary,Distance, License,Transport.

\*There is one missing value in the data set.

\*The predictor variables/independent variables are:

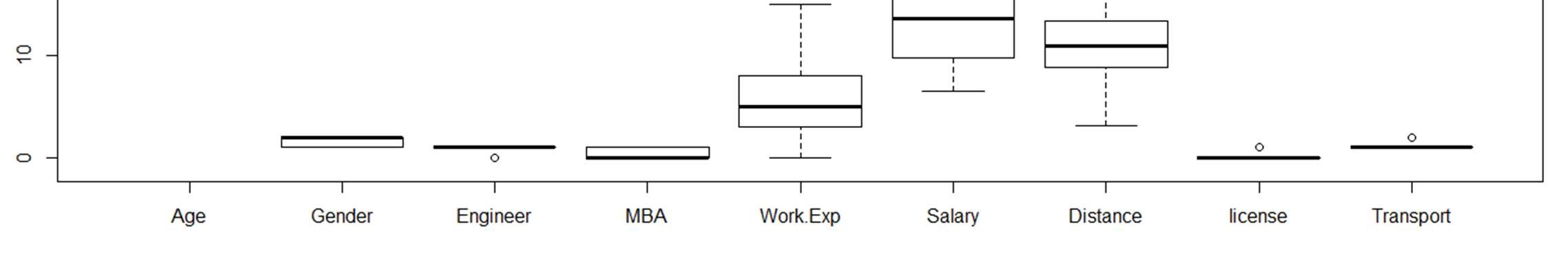
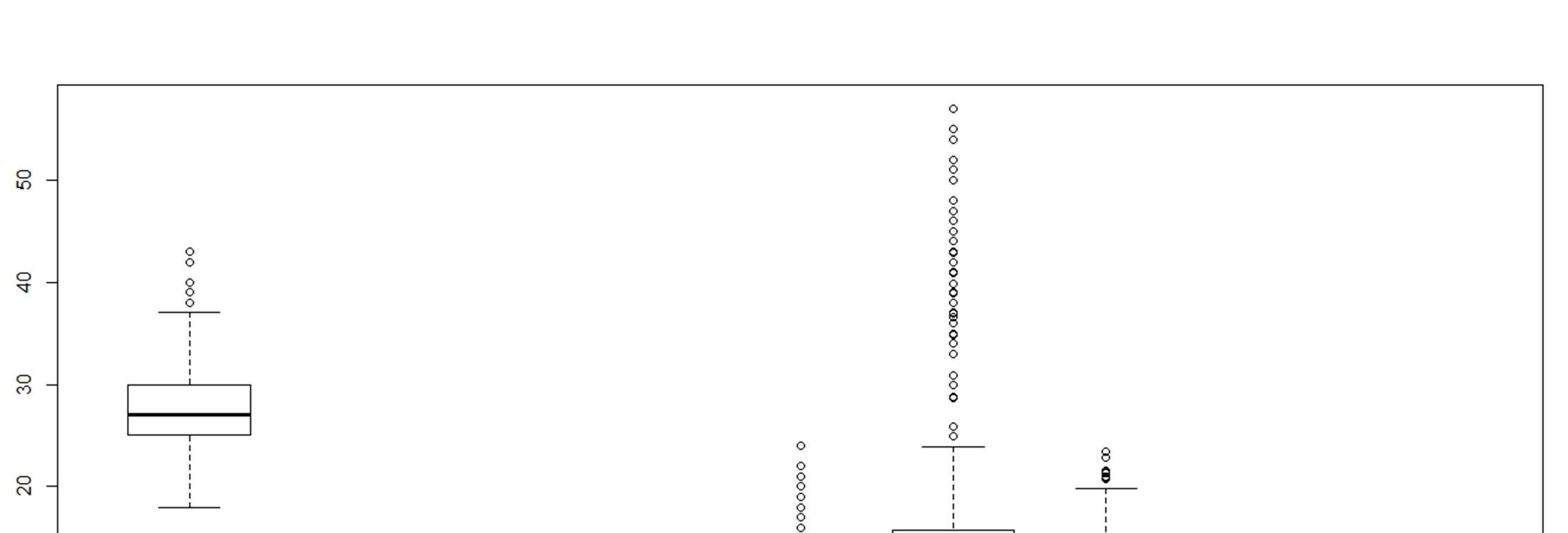
"Age" , "Gender" , "Engineer" , "MBA" , "Work.Exp" , "Salary" , ”Distance" , "license" , "Transport"

\*The Dependent variable is : “Transport”

## 3.CHECKING FOR THE INTEGRITY OF THE DATA

3.1 OUTLIER DETECTION USING BOXPLOT BOXPLOT

Depicts that there are outliers in our data. These plots are being made after seeing the correlation matrix and thus, figuring out which variables are to an extent related to each other and then we can visualize their relationship.



There are outliers in:

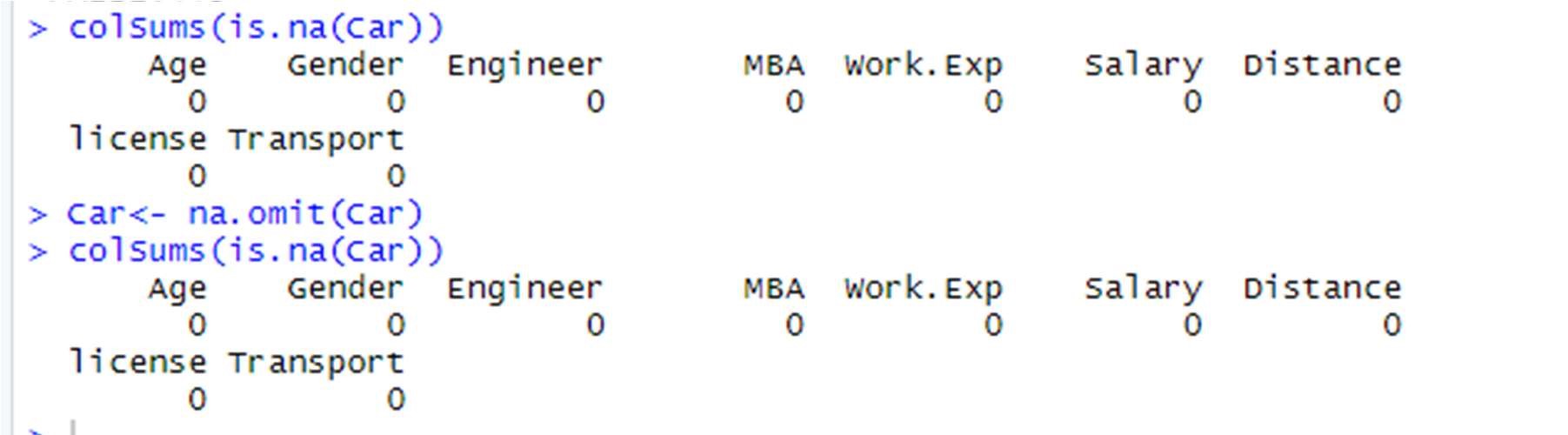
Age, Engineering, Work.Exp, Salary,Distance, License,Transport.

### 3.2MISSING VALUE DETECTION

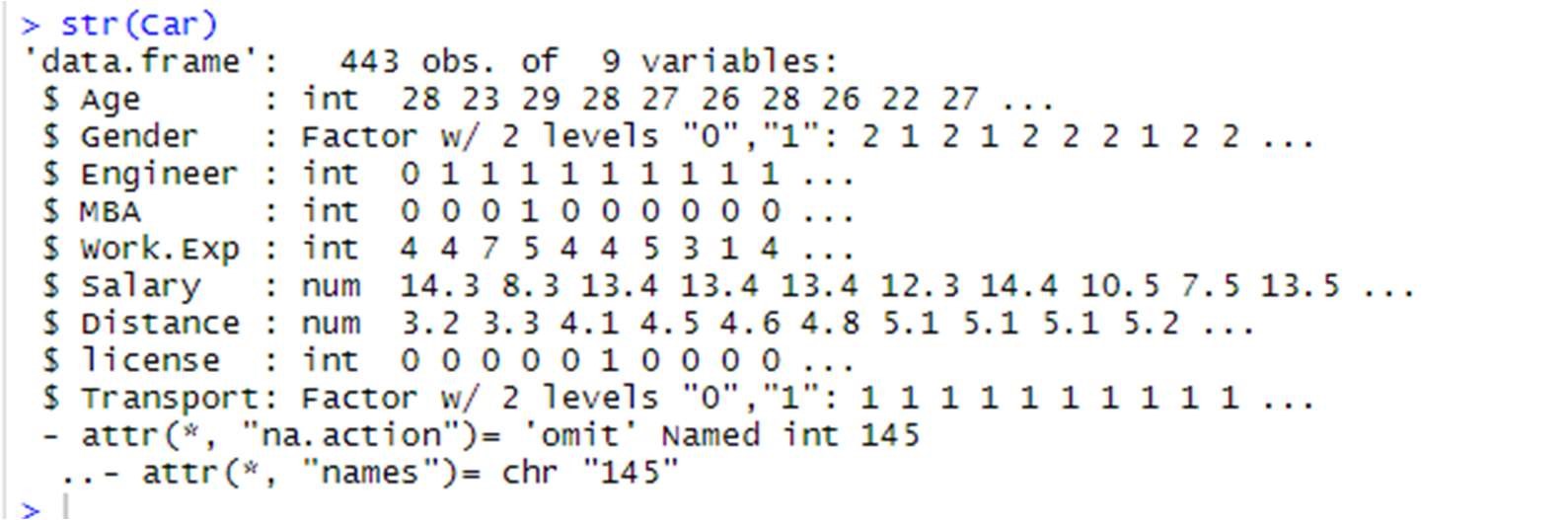
null= is.na(Car) summary(null) colSums(is.na(Car))

Car<- na.omit(Car)

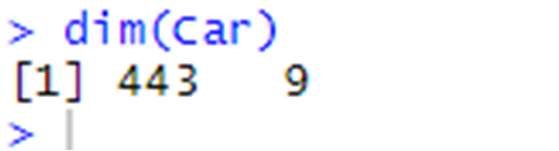
colSums(is.na(Car))



str(Car)



dim(Car)

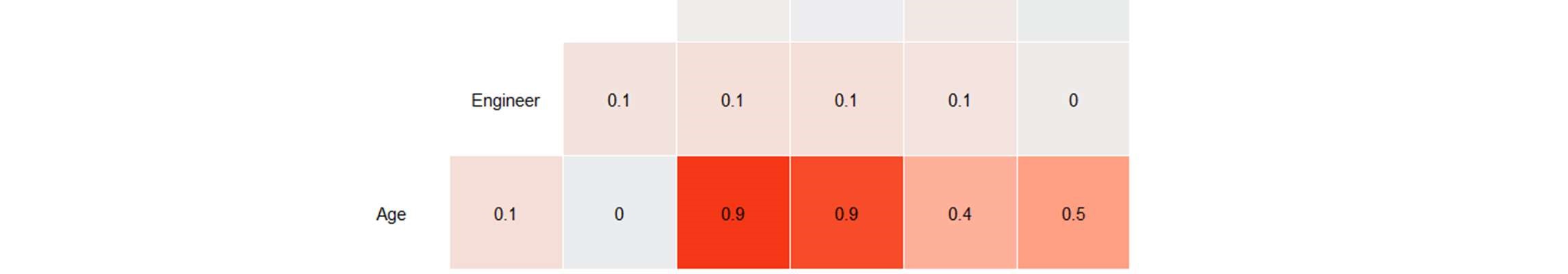
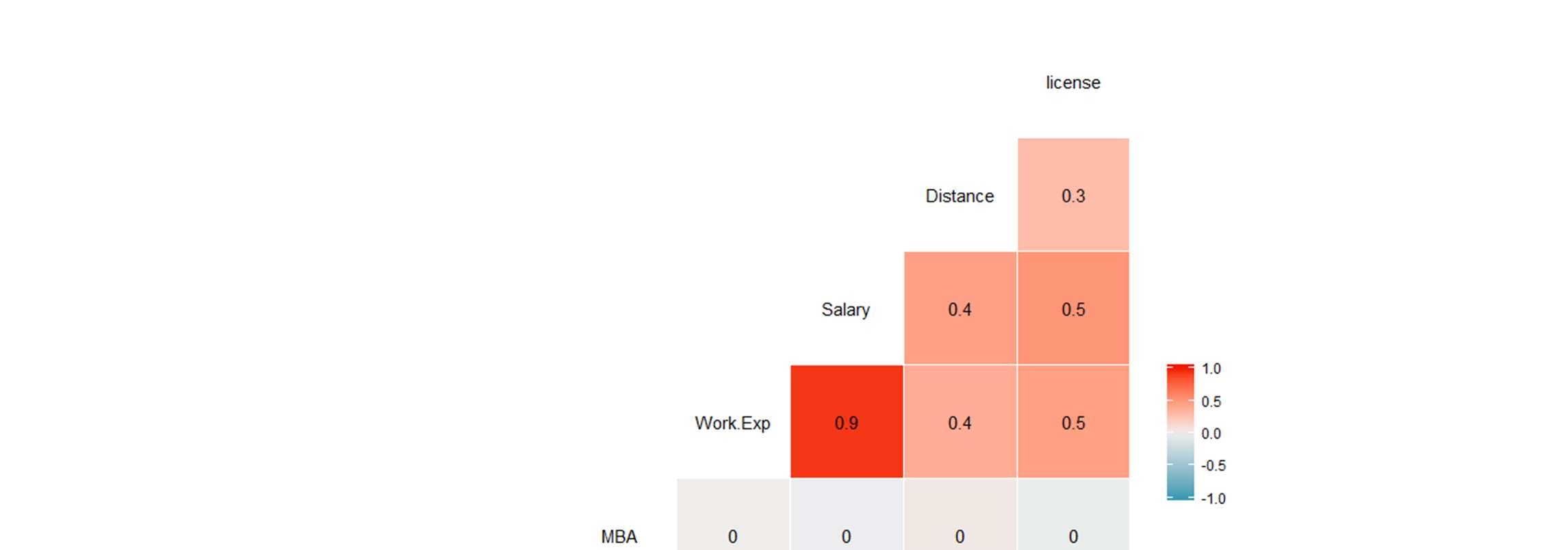


## 4.MULTICOLLINEARITY DETECTION

Multicollinearity occurs when two or more predictors in a regression equation are correlated.

We should not treat multicollinearity because right now logistic regression does not get affected by it. Although we could perform PCFA in order to reduce the variables and group the similar and highly correlated variables together to form a single variable.

There is multicollinearity between 3 variables- Work.Exp, Age, Salary



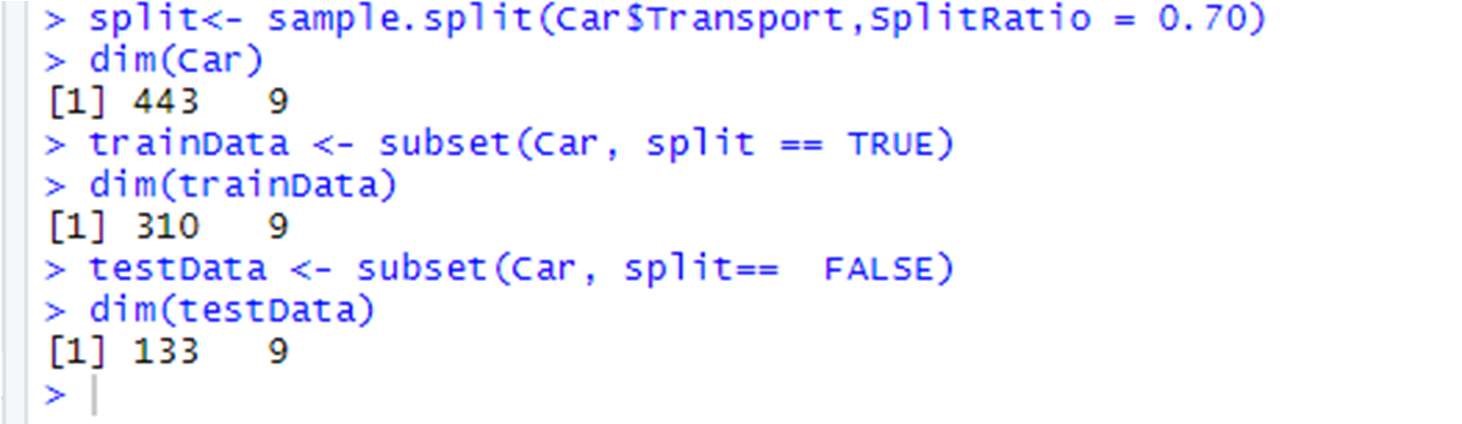
## 5. SMOTE DATA PREPARATION

5.1 Splitting The Dataset Into Train And Test

set.seed(848) table(Car$Transport) split<- sample.split(Car$Transport,SplitRatio = 0.70) dim(Car) trainData <- subset(Car, split == TRUE) testData <- subset(Car, split== FALSE)

Finding the dimensions of the Training and the Testing Dataset dim(trainData)

dim(testData)



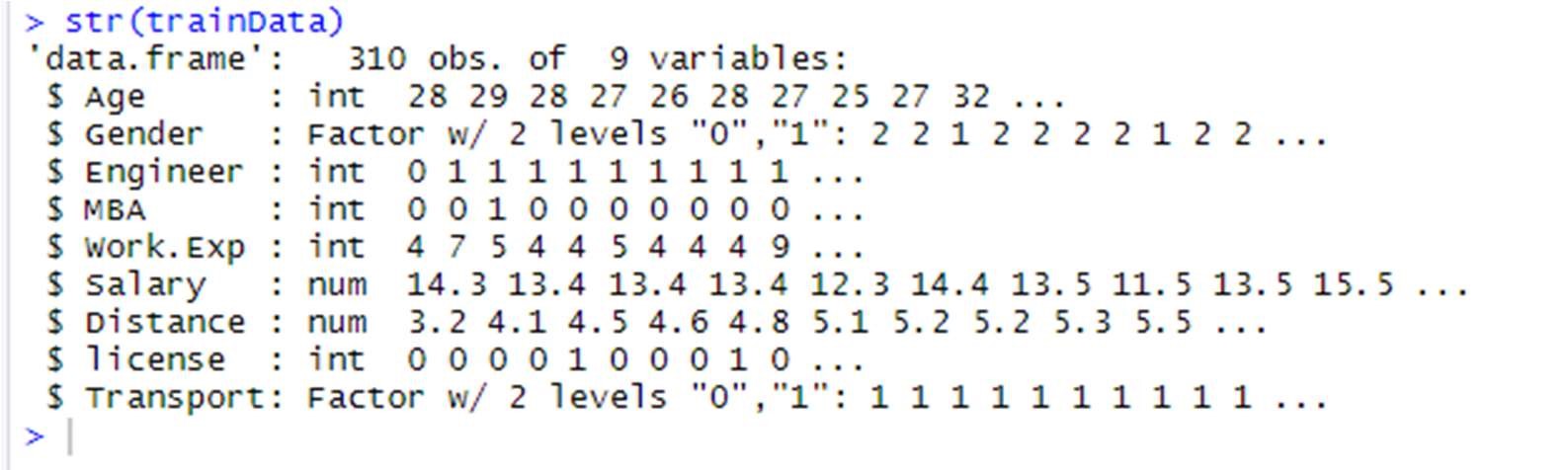
### 5.2 Applying Smote On The Train Dataset

trainData$Transport=as.factor(trainData$Transport)

smote.train= SMOTE(trainData$Transport~., trainData,perc.over = 350, k=7,perc.under=134) smote.test= testData table(smote.train$Transport) prop.table(table(smote.train$Transport))

head(trainData$Age)

#### str(trainData)

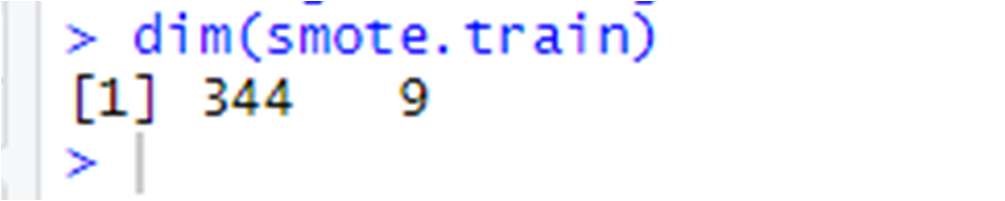


## 6. LOGISTIC REGRESSION

6.1 APPLYING LOGISTIC REGRESSION

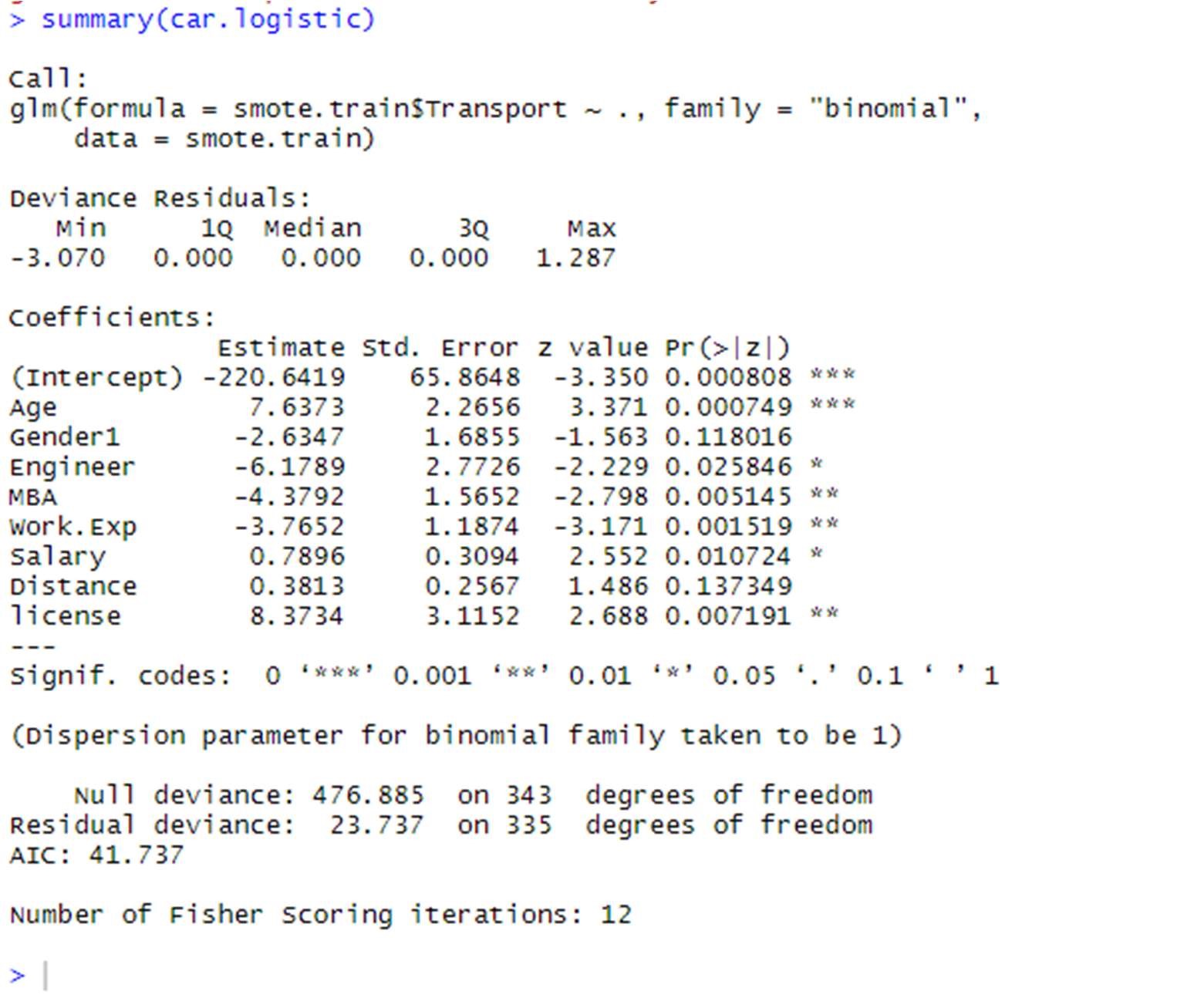
6.1.1 Creating Logistic Model

dim(smote.train)



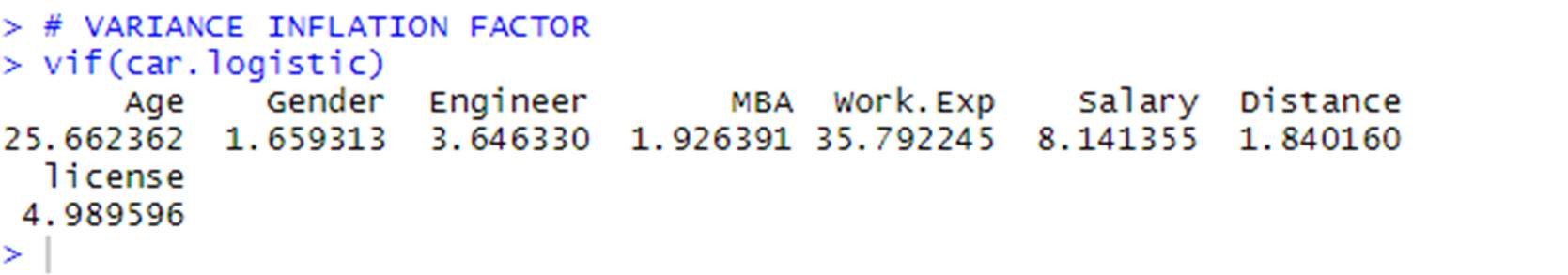
car.logistic <- glm(smote.train$Transport~., data = smote.train,

family = "binomial") summary(car.logistic)



6.1.2 Variance Inflation Factor

vif(car.logistic)



The VIF if more than 5 for all the variables except for the Engineer variable.

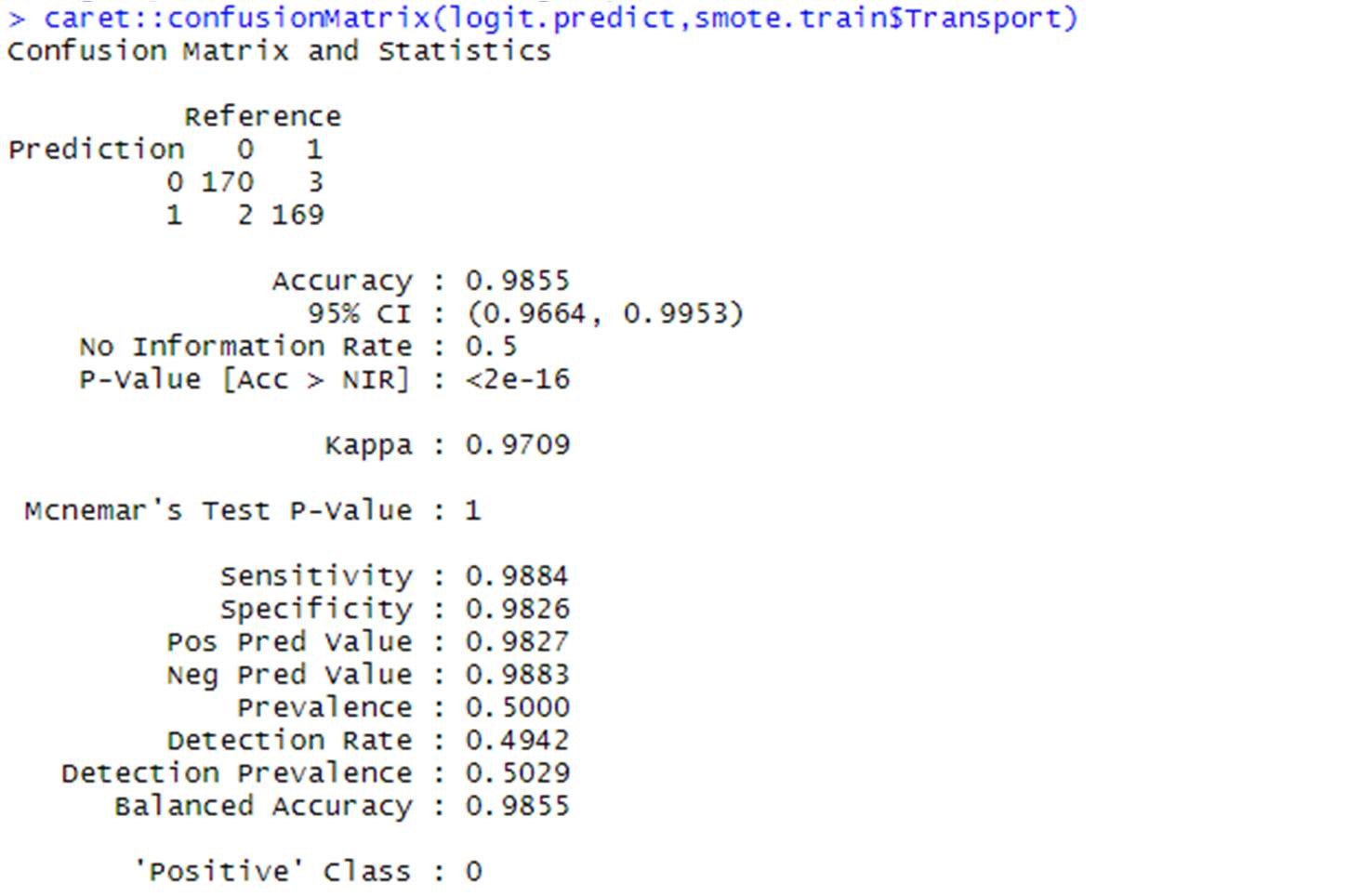
\*Therefore, removing the variable with the highest VIF

i.e., Work.Exp from the dataset will solve the multicollinearity problem.. But it is an important variable so we won’t treat multicollinearity in this case.

6.1.3 Predictions + Confusion Matrix On Train Data

logistic.pred= predict(car.logistic,smote.train) logit.predict <- ifelse(logistic.pred<.5,0,1) logit.predict<- as.factor(logit.predict)

caret::confusionMatrix(logit.predict,smote.train$Transport)



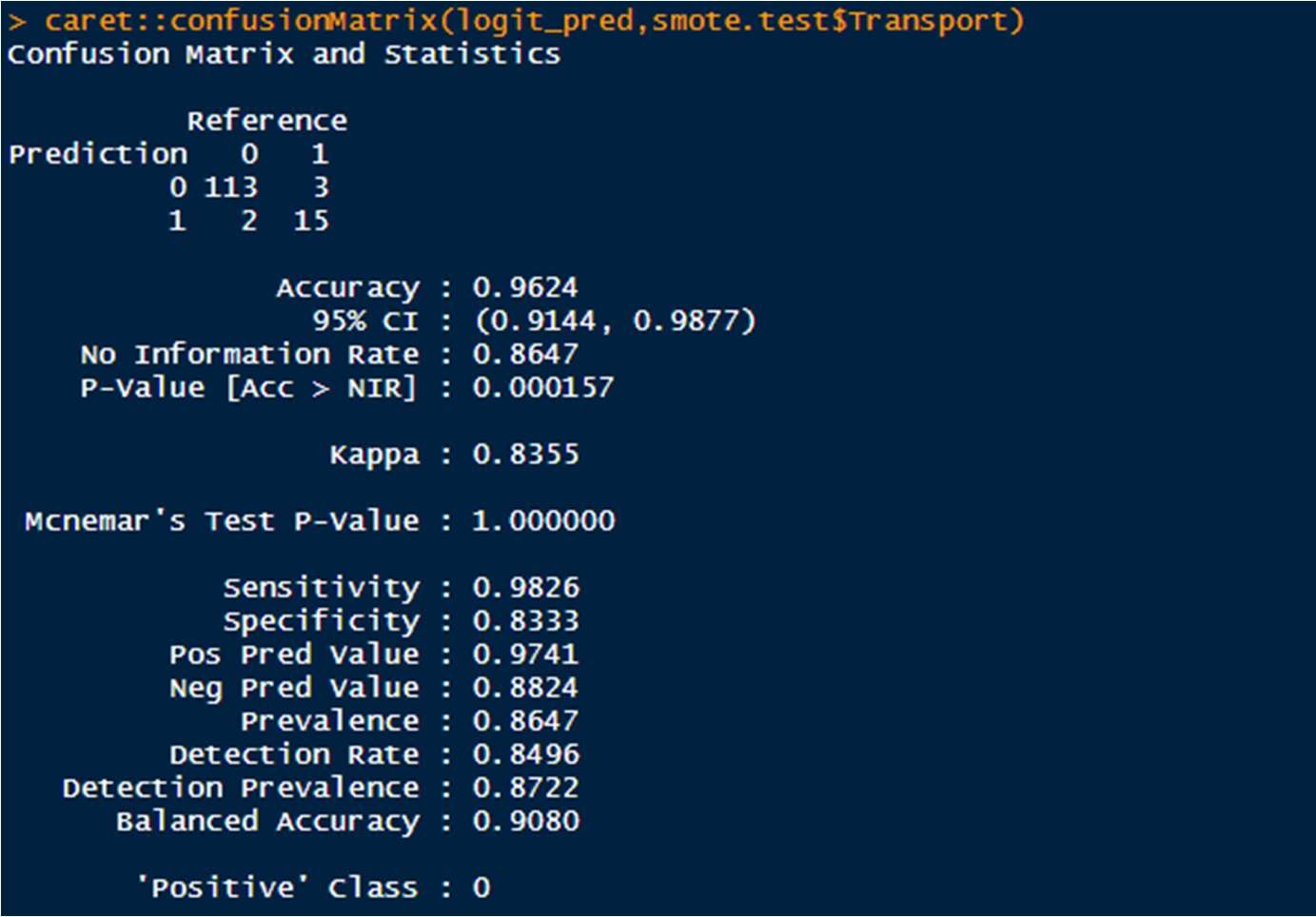
#Accuracy : 0.9855

#Sensitivity : 0.9884

#Specificity : 0.9826

6.1.4 Predictions + Confusion Matrix On Test Data

logistic.pred= predict(car.logistic,smote.test) logit\_pred <- ifelse(logistic.pred>.5,1,0) logit\_pred<- as.factor(logit\_pred) caret::confusionMatrix(logit\_pred,smote.test$Transport)



#Accuracy : 0.9624

#Sensitivity : 0.9826

#Specificity : 0.8333

# 15 out of 18 people who travelled by Car were predicted right.

### 6.2 INTERPRETATION OF LOGISTIC REGRESSION

The Logistic Regression model gives us an accuracy of 96.24%.

The True positive rate is 98.26% which is very good.

15 predictions were accurate put of a total of 18. Though we can strive to improve the sensitivity.

3 times , it predicted the transport wrongly. So, we’ll make other models to improve the accuracy of our predictions.

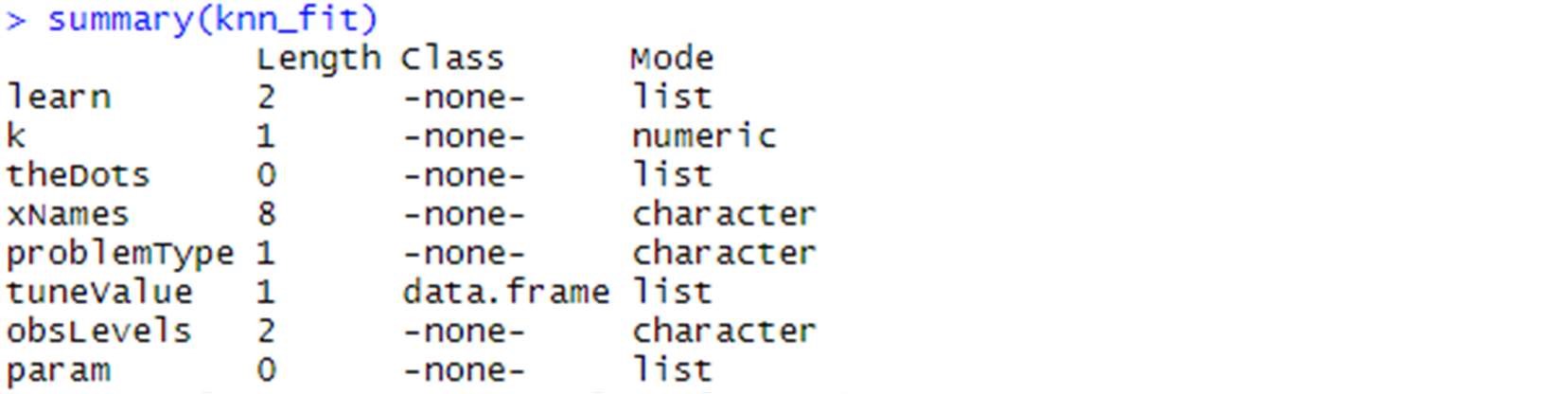
## 7.KNN MODEL

7.1 APPLYING KNN MODEL

str(smote.train)

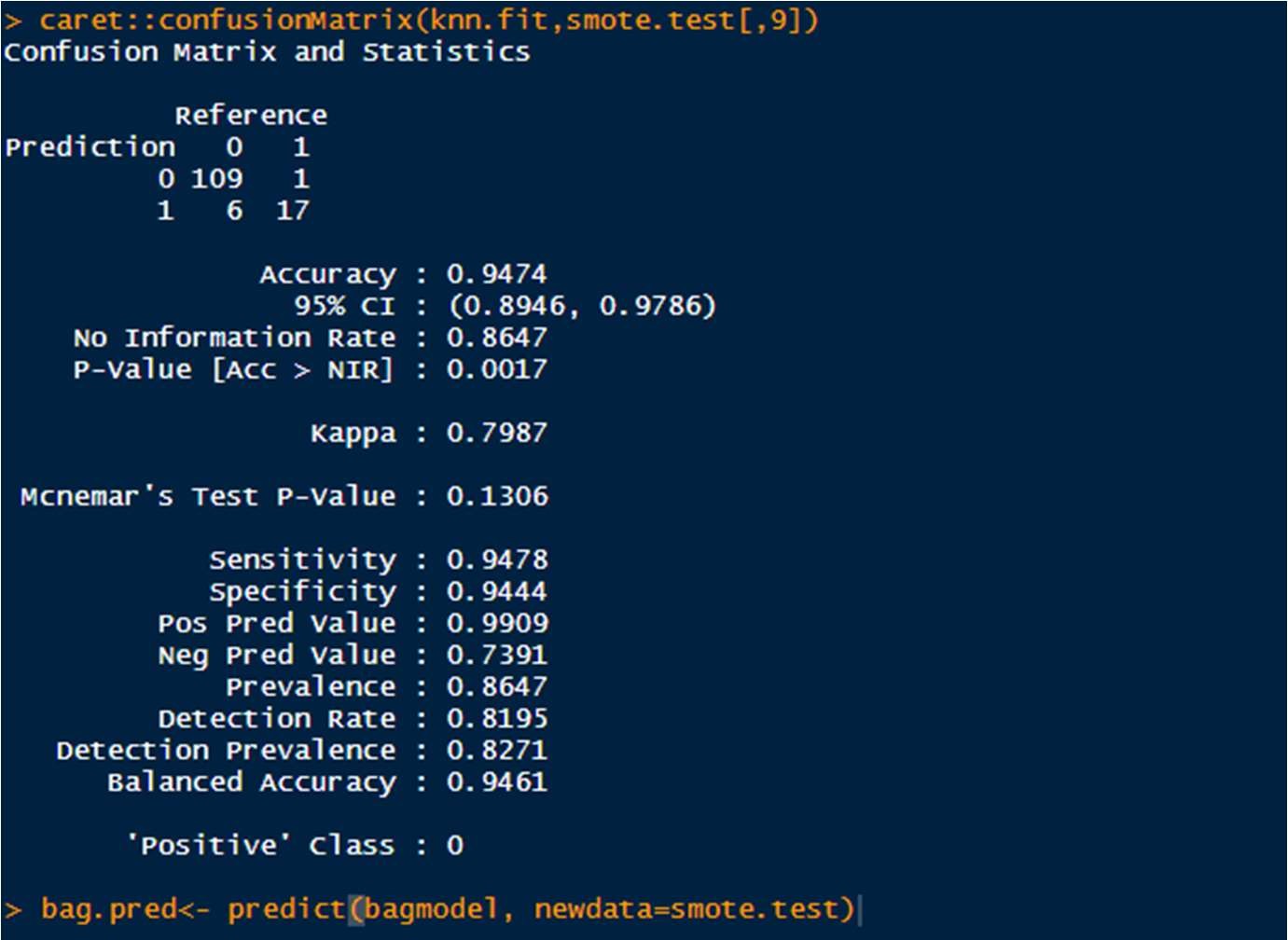
knn\_fit <- caret::train(Transport~.,data = smote.train,method = "knn",trControl= trainControl(method = "cv", number = 10), tuneLength = 10)

summary(knn\_fit)



# Therefore, we get the value of K to be used as = 5 knn.fit= knn(smote.train[,-9],smote.test[,-9],smote.train[,9],k=5) knn.fit= as.factor(knn.fit)

caret::confusionMatrix(knn.fit,smote.test[,9])



# Accuracy : 0.9474

# Sensitivity : 0.9478

# Specificity : 0.9444

\*17 people out of 18 people who travelled by Car were predicted right.

7.2 INTERPRETATION OF KNN MODEL

KNN supports non-linear solutions and can output only the labels.

The KNN predictions are calculated by simultaneously creating a KNN model and come out to be 0 or 1 depending on car and not-car. The Accuracy comes out to be= 94.74%

Reference

Prediction 0 1

1. 109 1
2. 6 17

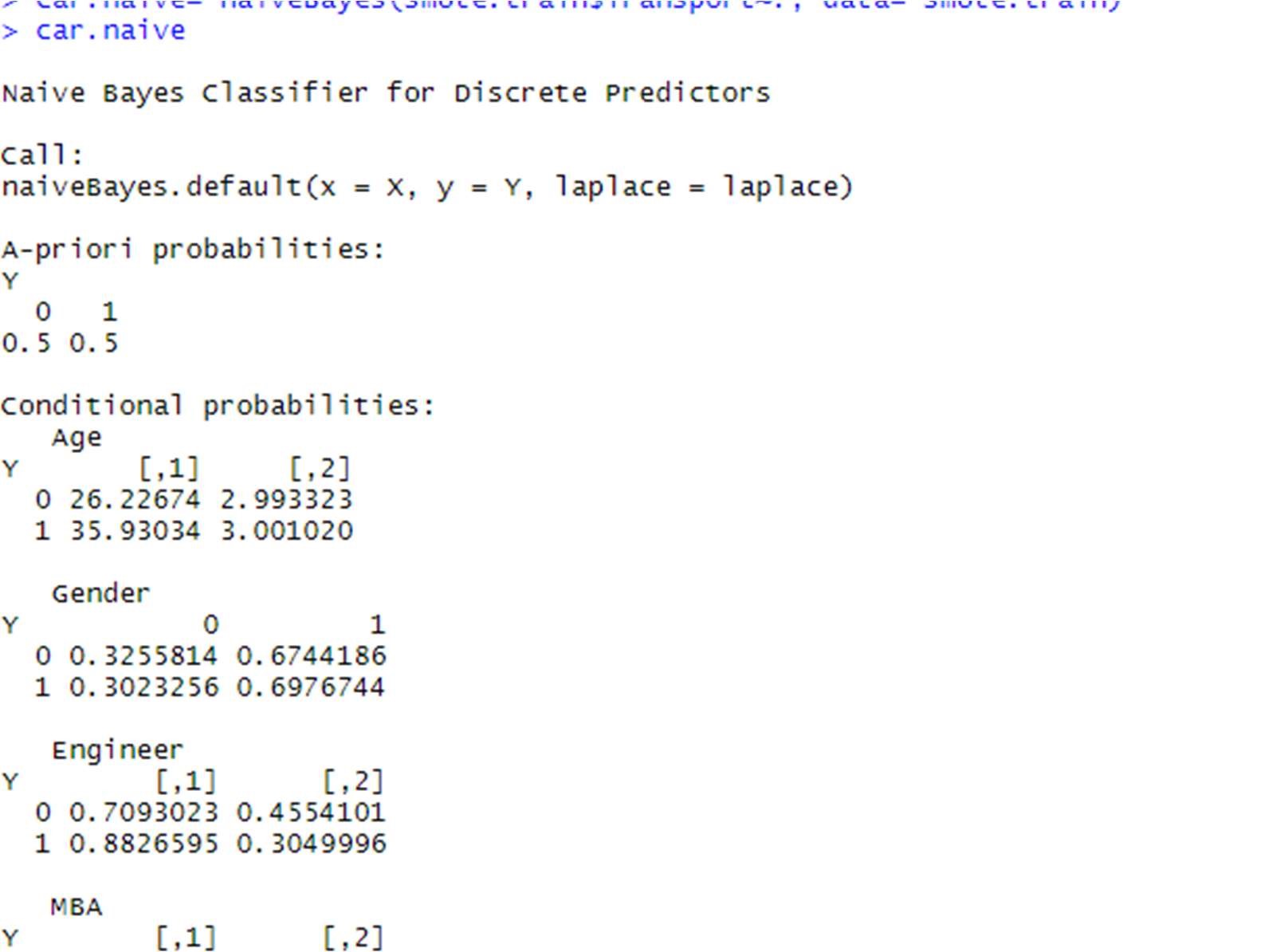
Sensitivity : 0.9478

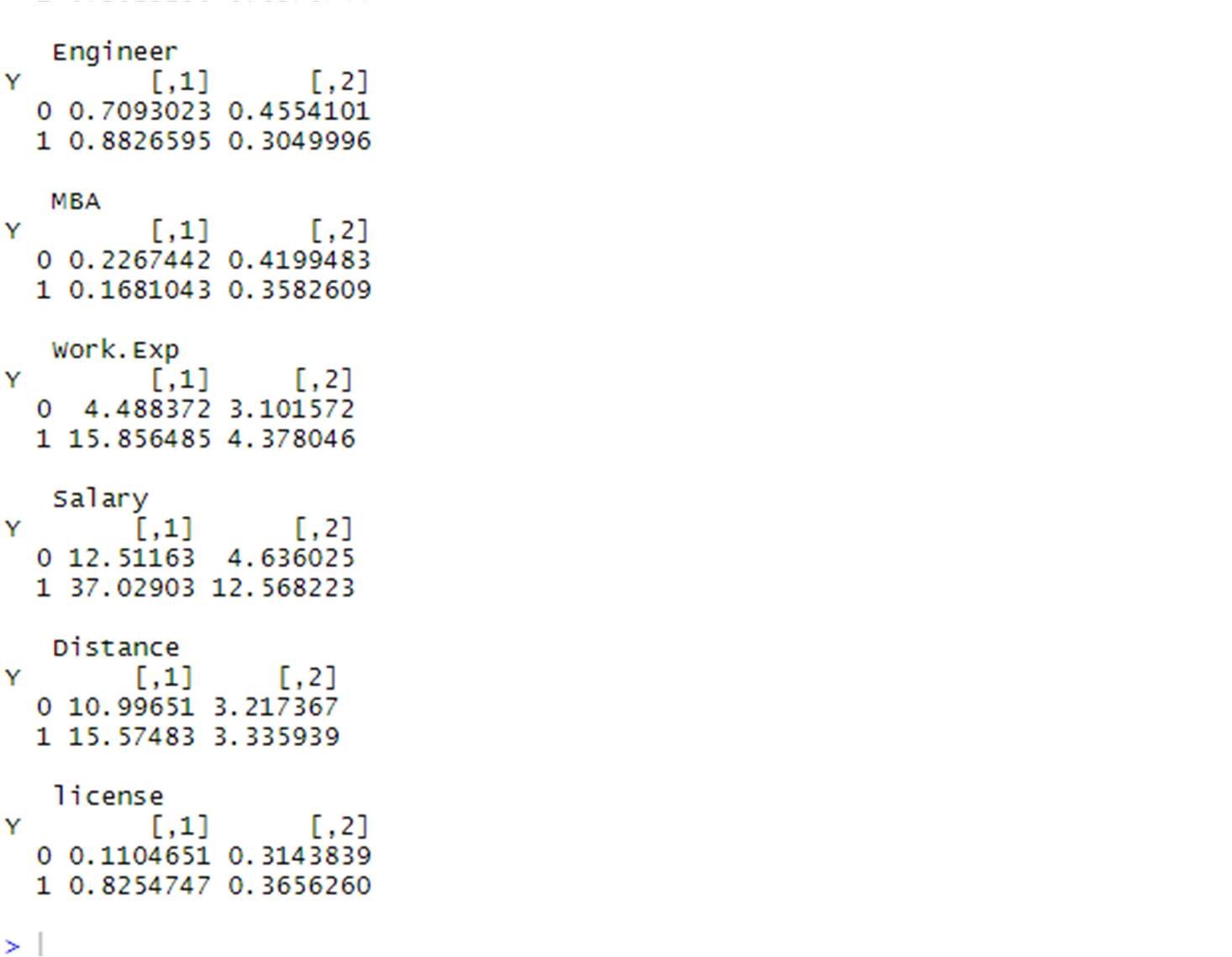
Specificity : 0.9444

## 8 NAÏVE BAYES’

8.1 APPLYING NAÏVE BAYES’

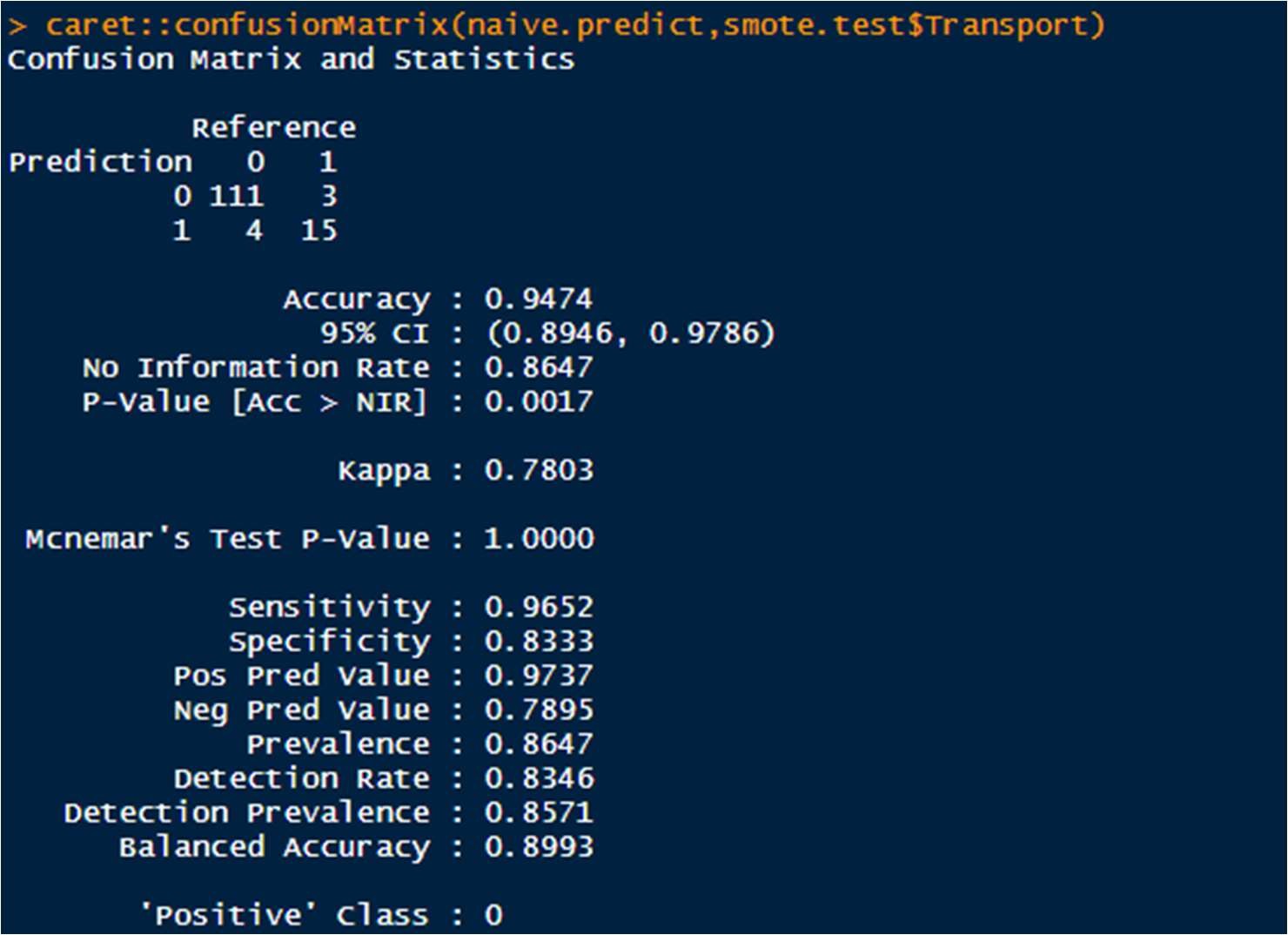
car.naive= naiveBayes(smote.train$Transport~., data= smote.train) car.naive





# The average Salary given Car as a means of transport is 12.73 and the standard # deviation for the same means is 5.40.

naive.predict= predict(car.naive,newdata = smote.test) caret::confusionMatrix(naive.predict,smote.test$Transport)



Accuracy : 0.9474

Sensitivity : 0.9652

Specificity : 0.8333

\*15 people out of 18 were predicted right for using car as a means of transport.

### 8.2 INTERPRETATION OF NAÏVE BAYES’

In Naïve Bayes, we can take each feature separately and determine it statistically giving certain conditions.

Let's consider Salary:

The average Salary given Car as a means of transport is 12.73 and the standard deviation for the same means is 5.40

Reference

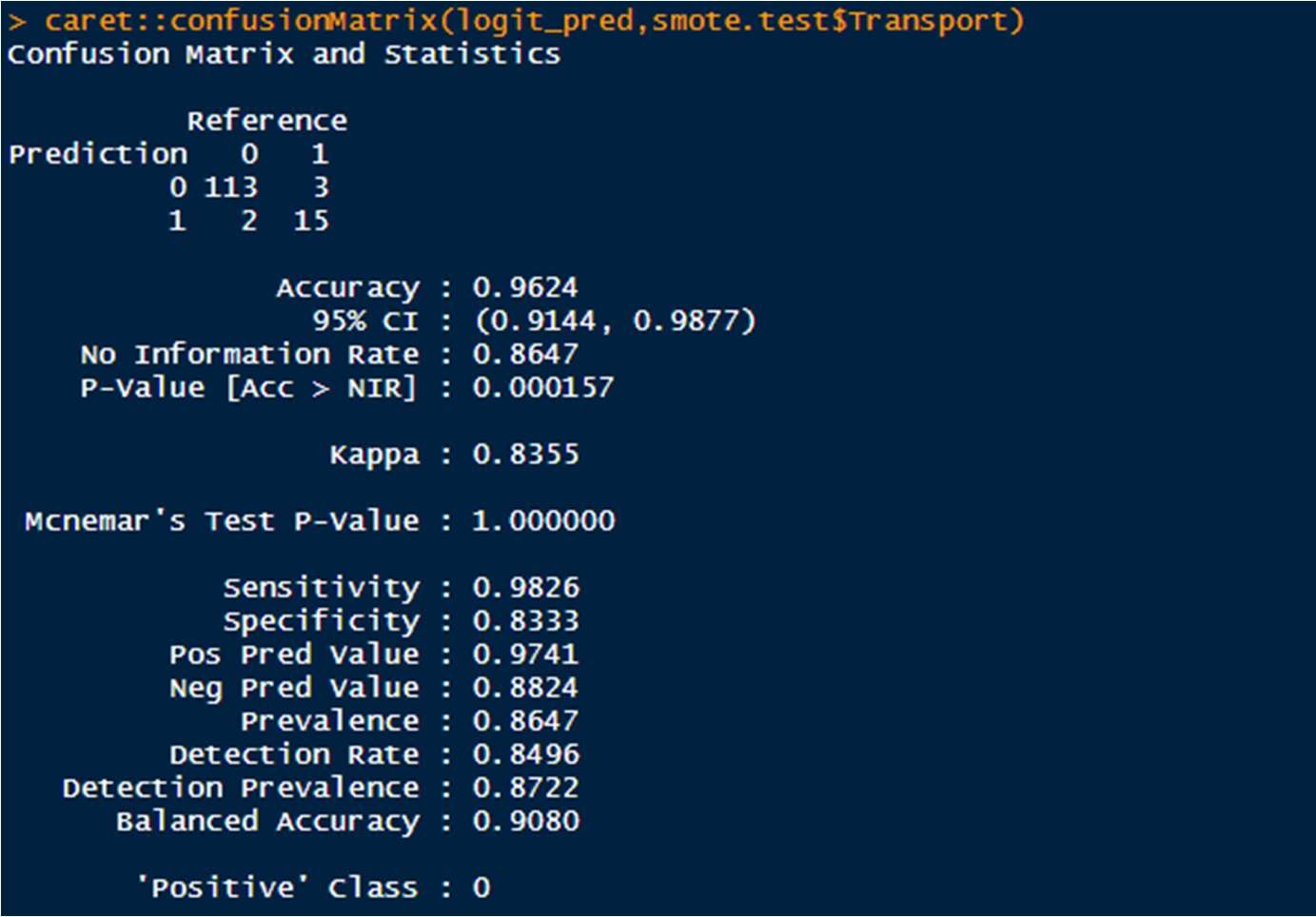
Prediction 0 1

1. 111 3
2. 4 15

The Accuracy is= 94.74%

## 9 .INTERPRETATION OF CONFUSION MATRIX

9.1For Logistic Regression

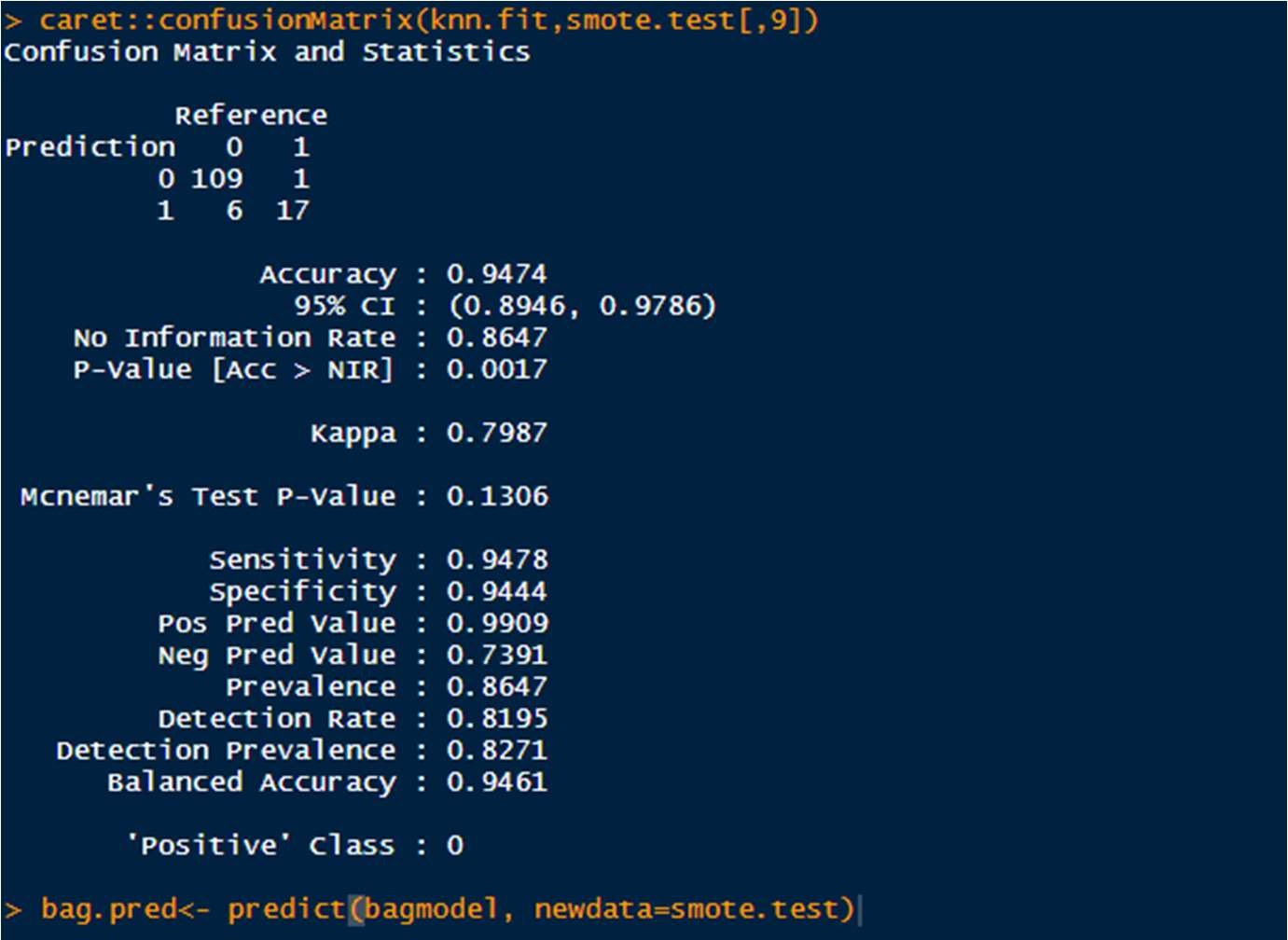


#Accuracy : 0.9624

#Sensitivity : 0.9826

#Specificity : 0.8333

### 9.2 For KNN Model

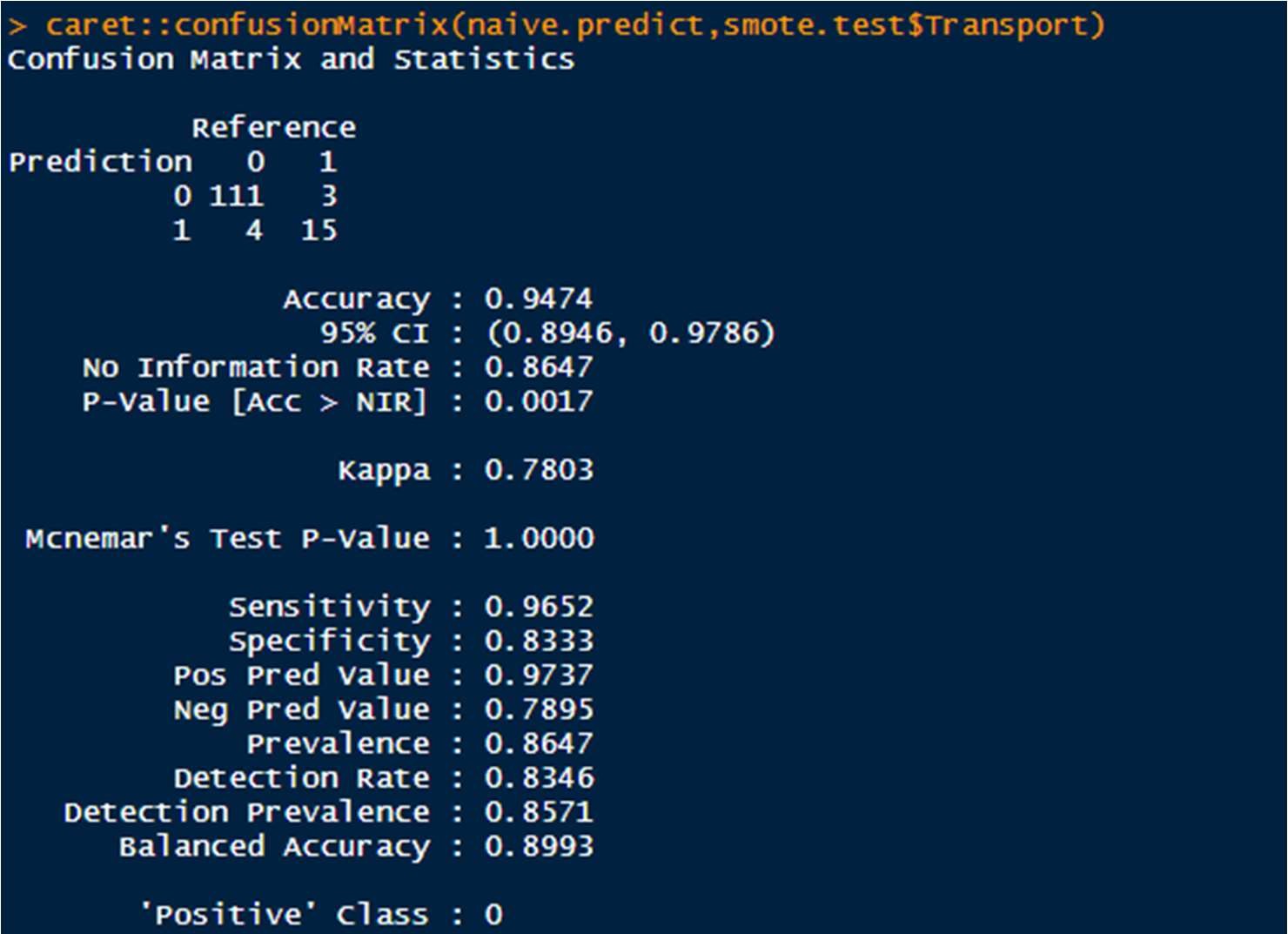


# Accuracy : 0.9474

# Sensitivity : 0.9478

# Specificity : 0.9444

### 9.3 For Naïve Bayes’



#Accuracy : 0.9474

#Sensitivity : 0.9652

#Specificity : 0.8333

### 10. REMARKS ON THE BEST PERFORMING MODEL

The best model amongst Logistic, KNN, and naïve bayes is KNN.

Although the Accuracy of Logistic is greatest but the ability of KNN model to identify Car as the transport is the highest.

Reference

Prediction 0 1

1. 109 1
2. 6 17

11. Bagging Ensemble Method

* 1. Applying Bagging Technique

bagmodel= bagging(as.numeric(smote.train$Transport)~.,data= smote.train, control= rpart.control(maxdepth = 5,minsplit = 4)) bag.pred<- predict(bagmodel, newdata=smote.test) table.bag<- table(smote.test$Transport,bag.pred>0.5) table.bag



This is a highly overfit model.There is no false. All the predictions which are 1, are correctly predicted.

\*Hence the sensitivity is (18+0)/18=100% Accuracy is = (18+0)/(115+18)=13.533%

* 1. INTERPRETATION OF BAGGING

Using Bagging Ensemble method for predictions gave us an overfit model.

It identified all the transports correctly to an accuracy of 100%.

TRUE

1. 115
2. 18

18 times transport was car and was predicted correctly without any false rates.

### 12. BOOSTING ENSEMBLE METHOD

12.1 CONVERTING THE DATA SET INTO NUMERIC MATRICES FOR APPLICATION

tp\_xgb<- vector() str(smote.train) str(smote.test)

# setting The Transport Variable of the Train set as numeric smote.train$Transport=as.numeric(smote.train$Transport) smote.train$Transport= ifelse(smote.train$Transport==2,1,0) # setting the other variables od Train Data as numeric smote.train$Gender<- as.numeric(smote.train$Gender)

# setting The Transport Variable of the TEST set as numeric smote.test$Transport=as.numeric(smote.test$Transport) smote.test$Transport= ifelse(smote.test$Transport==2,1,0) # setting the other variables od Train Data as numeric smote.test$Gender<- as.numeric(smote.test$Gender)

# Converting into Matrix features\_smote.train<-as.matrix(smote.train[,1:8]) label\_smote.train<- as.matrix(smote.train$Transport) features\_smote.test= as.matrix(smote.test[,1:8])

12.2 CREATING AN INITIAL MODEL BY BOOSTING

xgbmodel<-xgboost(data= features\_smote.train,

label = label\_smote.train, eta=0.001, max\_depth=5, min\_child\_weight=3, nrounds=1000, nfold=5, objective= "binary:logistic", verbose =0,

early\_stopping\_rounds =100) xgb.predict= predict(xgbmodel,newdata=features\_smote.test) xgb.pred= ifelse(xgb.predict>0.5,1,0) tab\_xgb=table(xgb.pred,smote.test$Transport)

sum(diag(tab\_xgb))/nrow(smote.test)

0.9172932

# The accuracy is 91.72

12.3 APPLYING ITERATION FOR TUNING THE BOOSTING MODEL

# Making the FIT Model for the "max\_depth" car\_xgb<- vector() md<- c(1,3,5,7,9,15)

for(i in md)

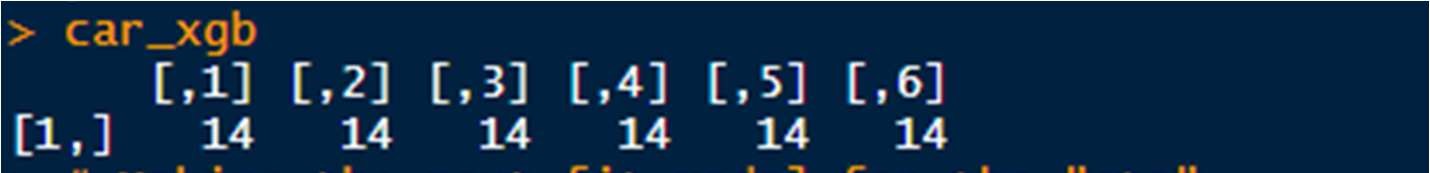
{ xgb.fit<-xgboost(data= features\_smote.train,

label = label\_smote.train, eta=0.001, max\_depth=md, min\_child\_weight=3, nrounds=1000, nfold=5, objective= "binary:logistic", verbose =0,

early\_stopping\_rounds =100)

smote.test$xgb.pred= predict(xgb.fit,features\_smote.test) car\_xgb<- cbind(car\_xgb,sum(smote.test$Transport==1 & smote.test$xgb.pred>0.5)) }

car\_xgb



\*Since we get the same values in all the iterations , we select the least value of max\_depth= 1

# Making the next fit model for the "eta" car\_xgb2<-vector() lr<- c(0.001,0.01,0.1,0.3,0.5,0.7,1)

for(i in lr) { xgb.fit<-xgboost(data= features\_smote.train,

label = label\_smote.train,

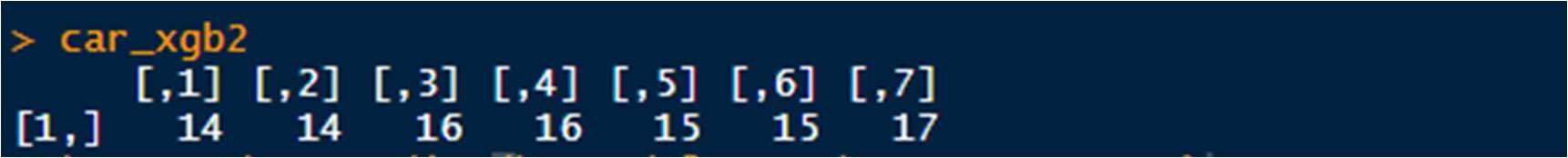
eta=i,

max\_depth=1, min\_child\_weight=3, nrounds=1000, nfold=5, objective= "binary:logistic", verbose =1,

early\_stopping\_rounds =100)

smote.test$xgb.pred= predict(xgb.fit,features\_smote.test) car\_xgb2<- cbind(car\_xgb2,sum(smote.test$Transport==1 & smote.test$xgb.pred>0.5)) }

car\_xgb2



As the number of true positives at eta=1 is the highest =17, we'll select eta=1

# Making the next fit model for the "nrounds" car\_xgb3<- vector() nr<-c(2,10,50,100,1000,10000)

for(i in nr)

{ xgb.fit<-xgboost(data= features\_smote.train,

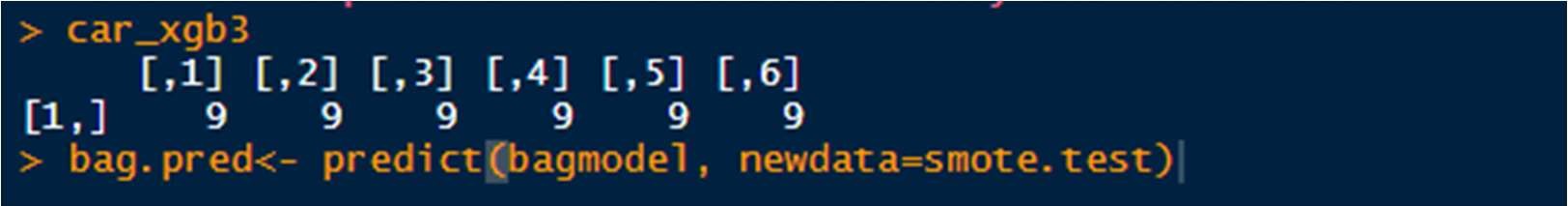
label = label\_smote.train, eta=0.001, max\_depth=5, min\_child\_weight=3, nrounds=nr, nfold=5, objective= "binary:logistic", verbose =1,

early\_stopping\_rounds =100)

smote.test$xgb.pred= predict(xgb.fit,features\_smote.test) car\_xgb3<- cbind(car\_xgb3,sum(smote.test$Transport==1 & smote.test$xgb.pred>0.5))

}

car\_xgb3



\*Since we get the same values in all the iterations , we select the least value of nrounds= 10

12.4 CREATING A FINAL MODEL

xgbmodel.final<-xgboost(data= features\_smote.train,

label = label\_smote.train,

eta=1,

max\_depth=1, min\_child\_weight=3, nrounds=100, nfold=5, objective= "binary:logistic", verbose =0,

early\_stopping\_rounds =100)

xgb.predict= predict(xgbmodel.final,newdata=features\_smote.test) xgb.pred= ifelse(xgb.predict>0.5,1,0) tab\_xgb=table(xgb.pred,smote.test$Transport) sum(diag(tab\_xgb))/nrow(smote.test)

0.9774436

### 13. Actionable Insights and Recommendations

We would suggest to use boosting as the technique to predict the mode of transport. We can tone the models and get great results through boosting ensemble method.

It’s much more versatile than KNN as we can alter variety of factors in this method and iterate until we get the maximum possible number of TRUE POSITIVES.

RCODE:

setwd("D:/BABI/BABI projects/Cars-Mode of transport") getwd()

install.packages("readr") install.packages("dplyr") install.packages("ggplot2") install.packages("GGally") install.packages("caTools") install.packages("rms") install.packages("ROCR") install.packages("pROC") install.packages("Hmisc") install.packages("DMwR") install.packages("caret") install.packages("InformationValue") install.packages("blorr") install.packages("ineq") install.packages("e1071") install.packages("xgboost") install.packages("ipred") install.packages("rpart")

install.packages("car")

library(ggplot2) library(readr) library(dplyr) library(GGally) library(caTools) library(Hmisc) library(stringi) library(rms) library(ROCR) library(pROC) library(DMwR)

library(caret) library(InformationValue) library(blorr) library(ineq) library(class) library(e1071) library(xgboost) library(ipred) library(rpart)

library(car)

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

Car= read.csv("Cars\_edited.csv")

str(Car) summary(Car) dim(Car) head(Car) tail(Car)

colnames(Car)

-------------------------------------------------------------------

#UNIVARIATE ANALYISIS

#done for both categorical and continuous variables

#must be numeric for histogram plotting

#HISTOGRAM

hist(Car$Age) hist(Car$Engineer) hist(Car$MBA) hist(Car$Work.Exp) hist(Car$Salary) hist(Car$Distance)

hist(Car$license)

#Using Boxplot now boxplot(Car)

# There are outliers in:Age,Work.Exp, Salary,Distance

#BIVARIATE ANALYSIS

plot(Car$Work.Exp,Car$Salary) plot(Car$Age,Car$Work.Exp)

plot(Car$Age,Car$Salary)

#MISSING VALUE DETECTION

null= is.na(Car) summary(null) colSums(is.na(Car)) Car<- na.omit(Car) colSums(is.na(Car)) str(Car)

#Dimension

dim(Car)

#Labelling the Dependent Variable

Car$Transport <- ifelse(Car$Transport=="Car",1,0)

Car$Transport <- as.factor(Car$Transport)

Car$Gender <- ifelse(Car$Gender =="Male",1,0)

Car$Gender <- as.factor(Car$Gender)

str(Car)

#----------------------------------------------------------------------------- # Multicollinearity

ggcorr(Car,label= TRUE)

# There is multicollinearity between 3 variables: Age, Work.Exp and Salary # Multicollinearity occurs when two or more predictors in a regression # equation are correlated.

#We should not treat multicollinearity because right now logistic

#regression does notget affected by it. Although we could perform PCFA #in order to reduce the variables and group the similar and highly #correlated variables together to forma single variable.

dim(Car) head(Car,10) tail(Car,10)

# We can use Principle Component Analysis

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

set.seed(848) table(Car$Transport) split<- sample.split(Car$Transport,SplitRatio = 0.70)

dim(Car) trainData <- subset(Car, split == TRUE) dim(trainData) testData <- subset(Car, split== FALSE) dim(testData)

#SMOTE DATA PREPARATION

trainData$Transport=as.factor(trainData$Transport) smote.train= SMOTE(trainData$Transport~., trainData,perc.over = 350, k=7,perc.under=134) smote.test= testData table(smote.train$Transport)

prop.table(table(smote.train$Transport))

head(trainData$Age) str(trainData)

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

# Logistic Resgression

dim(smote.train)

car.logistic <- glm(smote.train$Transport~., data = smote.train,

family = "binomial")

summary(car.logistic)

# VARIANCE INFLATION FACTOR

vif(car.logistic)

# The VIF if more than 5 for all the variables except for the # Engineer variable.

# Predicting on the Train Data itself logistic.pred= predict(car.logistic,smote.train) logit.predict <- ifelse(logistic.pred<.5,0,1) logit.predict<- as.factor(logit.predict) caret::confusionMatrix(logit.predict,smote.train$Transport)

#Accuracy : 0.9855

#Sensitivity : 0.9884

#Specificity : 0.9826

logistic.pred= predict(car.logistic,smote.test) logit\_pred <- ifelse(logistic.pred>.5,1,0) logit\_pred<- as.factor(logit\_pred) caret::confusionMatrix(logit\_pred,smote.test$Transport)

#Accuracy : 0.9624

#Sensitivity : 0.9826

#Specificity : 0.8333

# 15 out of 18 people who travelled by Car were predicted right.

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

#KNN

str(smote.train)

knn\_fit <- caret::train(Transport~.,data = smote.train,method = "knn", trControl= trainControl(method = "cv", number = 10), tuneLength = 10)

summary(knn\_fit)

# Therefore, we get the value of K to be used as = 5 knn.fit= knn(smote.train[,-9],smote.test[,-9],smote.train[,9],k=5) knn.fit= as.factor(knn.fit) caret::confusionMatrix(knn.fit,smote.test[,9])

# Accuracy : 0.9474

# Sensitivity : 0.9478

# Specificity : 0.9444

# 17 people out of 18 people who travelled by Car were predicted right.

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

car.naive= naiveBayes(smote.train$Transport~., data= smote.train) car.naive

# The average Salary given Car as a means of transport is 12.73 and the standard # deviation for the same means is 5.40.

naive.predict= predict(car.naive,newdata = smote.test)

caret::confusionMatrix(naive.predict,smote.test$Transport)

#Accuracy : 0.9474

#Sensitivity : 0.9652

#Specificity : 0.8333

#15 people out of 18 were predicted right for using car as a means of transport.

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

# Bagging

bagmodel= bagging(as.numeric(smote.train$Transport)~.,data= smote.train,

control= rpart.control(maxdepth = 5,minsplit = 4)) bag.pred<- predict(bagmodel, newdata=smote.test) table.bag<- table(smote.test$Transport,bag.pred>0.5) table.bag

# This is a highly overfit model.

# There is no false. All the predictions which are 1, are correctly predicted.

# Hence the

#sensitivity is (18+0)/18=100%

#Accuracy is = (18+0)/(115+18)=13.533%

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

# Boosting

tp\_xgb<- vector() str(smote.train) str(smote.test)

# setting The Transport Variable of the Train set as numeric smote.train$Transport=as.numeric(smote.train$Transport) smote.train$Transport= ifelse(smote.train$Transport==2,1,0) # setting the other variables od Train Data as numeric

smote.train$Gender<- as.numeric(smote.train$Gender)

# setting The Transport Variable of the TEST set as numeric smote.test$Transport=as.numeric(smote.test$Transport) smote.test$Transport= ifelse(smote.test$Transport==2,1,0) # setting the other variables od Train Data as numeric

smote.test$Gender<- as.numeric(smote.test$Gender)

# Converting into Matrix features\_smote.train<-as.matrix(smote.train[,1:8]) label\_smote.train<- as.matrix(smote.train$Transport)

features\_smote.test= as.matrix(smote.test[,1:8])

# Making an initial Model

xgbmodel<-xgboost(data= features\_smote.train,

label = label\_smote.train,

eta=0.001, max\_depth=5, min\_child\_weight=3, nrounds=1000, nfold=5,

objective= "binary:logistic", verbose =0, early\_stopping\_rounds =100)

xgb.predict= predict(xgbmodel,newdata=features\_smote.test) xgb.pred= ifelse(xgb.predict>0.5,1,0) tab\_xgb=table(xgb.pred,smote.test$Transport) sum(diag(tab\_xgb))/nrow(smote.test)

# The accuracy is 93.2

# Finding the Best Model(Tuning XGB Model)

# Making the FIT Model for the "max\_depth" car\_xgb<- vector()

md<- c(1,3,5,7,9,15)

for(i in md) { xgb.fit<-xgboost(data= features\_smote.train, label = label\_smote.train,

eta=0.001, max\_depth=md, min\_child\_weight=3, nrounds=1000, nfold=5,

objective= "binary:logistic", verbose =0, early\_stopping\_rounds =100)

smote.test$xgb.pred= predict(xgb.fit,features\_smote.test) car\_xgb<- cbind(car\_xgb,sum(smote.test$Transport==1 & smote.test$xgb.pred>0.5)) }

car\_xgb

# Since we get the same values in all the iterations , we select the least value

# of max\_depth= 1

# Making the next fit model for the "eta" car\_xgb2<-vector() lr<- c(0.001,0.01,0.1,0.3,0.5,0.7,1)

for(i in lr) {

xgb.fit<-xgboost(data= features\_smote.train, label = label\_smote.train,

eta=i, max\_depth=1, min\_child\_weight=3, nrounds=1000, nfold=5, objective= "binary:logistic", verbose =1, early\_stopping\_rounds =100)

smote.test$xgb.pred= predict(xgb.fit,features\_smote.test)

car\_xgb2<- cbind(car\_xgb2,sum(smote.test$Transport==1 & smote.test$xgb.pred>0.5)) }

car\_xgb2

# As the number of true positives at eta=1 is the highest =17, we'll select

# eta=1

# Making the next fit model for the "nrounds" car\_xgb3<- vector()

nr<-c(2,10,50,100,1000,10000) for(i in nr) { xgb.fit<-xgboost(data= features\_smote.train, label = label\_smote.train,

eta=0.001, max\_depth=5, min\_child\_weight=3,

nrounds=nr, nfold=5, objective= "binary:logistic", verbose =1, early\_stopping\_rounds =100)

smote.test$xgb.pred= predict(xgb.fit,features\_smote.test) car\_xgb3<- cbind(car\_xgb3,sum(smote.test$Transport==1 & smote.test$xgb.pred>0.5))

} car\_xgb3

# Since we get the same values in all the iterations , we select the least value

# of nrounds= 10

# FINAL MODEL

xgbmodel.final<-xgboost(data= features\_smote.train, label = label\_smote.train,

eta=1, max\_depth=1, min\_child\_weight=3, nrounds=100, nfold=5, objective= "binary:logistic", verbose =0, early\_stopping\_rounds =100)

xgb.predict= predict(xgbmodel.final,newdata=features\_smote.test) xgb.pred= ifelse(xgb.predict>0.5,1,0) tab\_xgb=table(xgb.pred,smote.test$Transport) sum(diag(tab\_xgb))/nrow(smote.test)