Predicting mode of Transport (ML)

[Document subtitle]

SUBMITTED BY-KANUPRIYA MITTAL
UNDER THE GUIDANCE OF – DEEPAK GUPTA
[Company address]

CONTENT:

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:

```
> str(Car)
'data.frame':   443 obs. of  9 variables:
 $ Age      : int  28 23 29 28 27 26 28 26 22 27 ...
 $ Gender   : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 1 2 2 ...
 $ Engineer : int  0 1 1 1 1 1 1 1 1 1 ...
 $ MBA      : int  0 0 0 1 0 0 0 0 0 0 ...
 $ Work.Exp : int  4 4 7 5 4 4 5 3 1 4 ...
 $ Salary   : num  14.3 8.3 13.4 13.4 13.4 12.3 14.4 10.5 7.5 13.5 ...
 $ Distance : num  3.2 3.3 4.1 4.5 4.6 4.8 5.1 5.1 5.1 5.2 ...
 $ license  : int  0 0 0 0 0 1 0 0 0 0 ...
 $ Transport: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "na.action")= 'omit' Named int 145
  ..- attr(*, "names")= chr "145"
>
```

Summary Of Dataset:

```
> summary(Car)
      Age          Gender      Engineer            MBA              Work.Exp
 Min.   :18.00   0:127    Min.   :0.0000   Min.   :0.0000   Min.   : 0.0
 1st Qu.:25.00   1:316    1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.: 3.0
 Median :27.00            Median :1.0000   Median :0.0000   Median : 5.0
 Mean   :27.75            Mean   :0.7562   Mean   :0.2528   Mean   : 6.3
 3rd Qu.:30.00            3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 8.0
 Max.   :43.00            Max.   :1.0000   Max.   :1.0000   Max.   :24.0
     Salary          Distance         license          Transport
 Min.   : 6.50   Min.   : 3.20   Min.   :0.0000   0:382
 1st Qu.: 9.80   1st Qu.: 8.80   1st Qu.:0.0000   1: 61
 Median :13.60   Median :11.00   Median :0.0000
 Mean   :16.24   Mean   :11.33   Mean   :0.2348
 3rd Qu.:15.75   3rd Qu.:13.45   3rd Qu.:0.0000
 Max.   :57.00   Max.   :23.40   Max.   :1.0000
>
```

Dim- will tell the dimension of our dataset

```
> dim(Car)
[1] 443   9
```

Head- will fetch the first 10 records of the dataset

```
> head(Car)
  Age Gender Engineer MBA Work.Exp Salary Distance license Transport
1  28      1        0   0        4   14.3      3.2       0         0
2  23      0        1   0        4    8.3      3.3       0         0
3  29      1        1   0        7   13.4      4.1       0         0
4  28      0        1   1        5   13.4      4.5       0         0
5  27      1        1   0        4   13.4      4.6       0         0
6  26      1        1   0        4   12.3      4.8       1         0
> 
```

Tails- will fetch the last 10 records of the dataset

```
> tail(Car)
    Age Gender Engineer MBA Work.Exp Salary Distance license Transport
439  34      1        1   0       14     38     21.3       1         1
440  40      1        1   0       20     57     21.4       1         1
441  38      1        1   0       19     44     21.5       1         1
442  37      1        1   0       19     45     21.5       1         1
443  37      1        0   0       19     47     22.8       1         1
444  39      1        1   1       21     50     23.4       1         1
> 
```

Name of all columns:

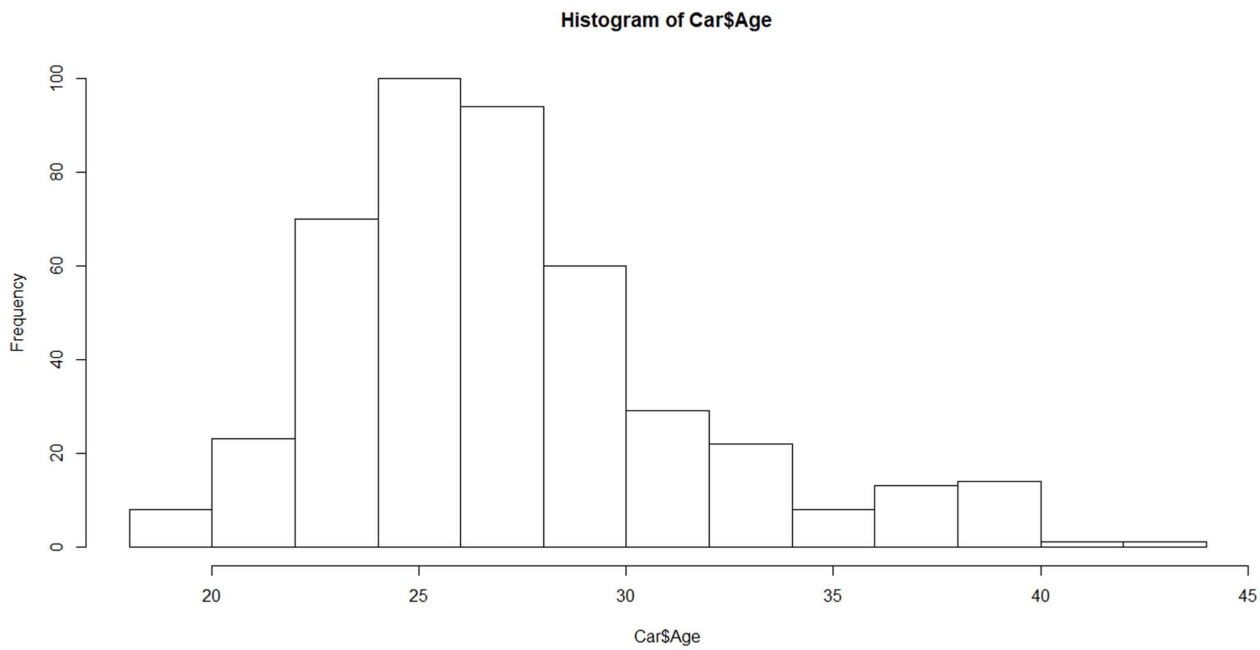
```
> colnames(Car)
[1] "Age"        "Gender"     "Engineer"   "MBA"        "Work.Exp"   "Salary"
[7] "Distance"   "license"    "Transport"
> 
```

2.2.1 UNIVARIATE ANALYISIS

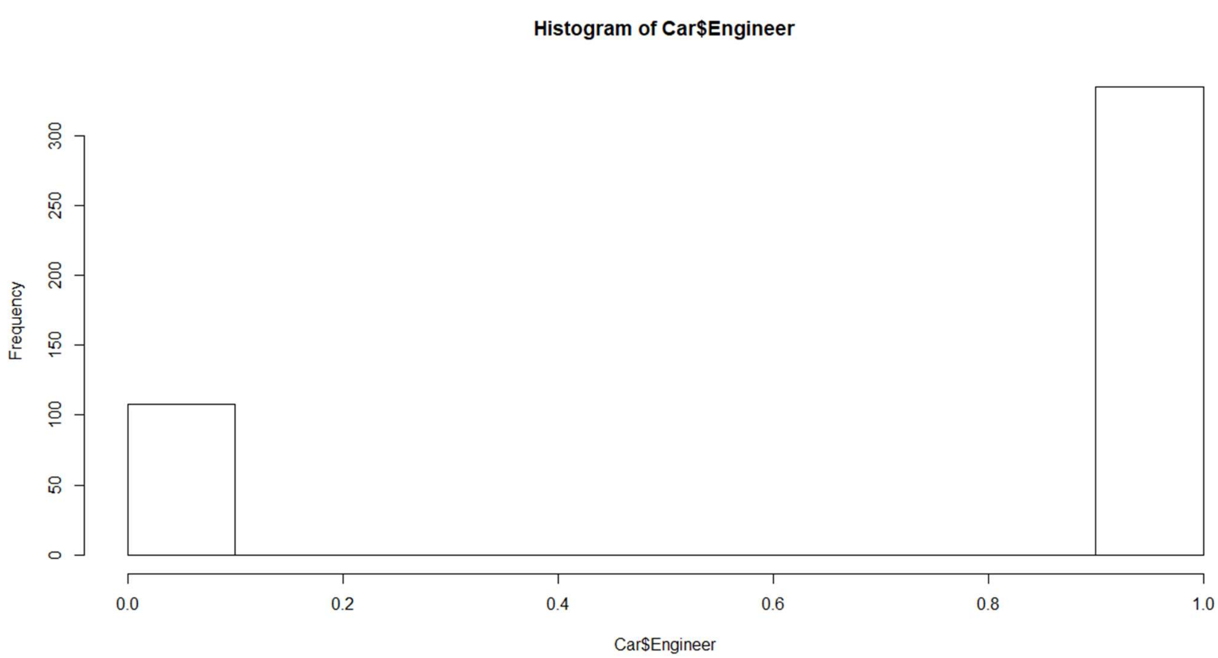
Done for both categorical and continuous variables
Must be numeric for histogram plotting

*HISTOGRAM

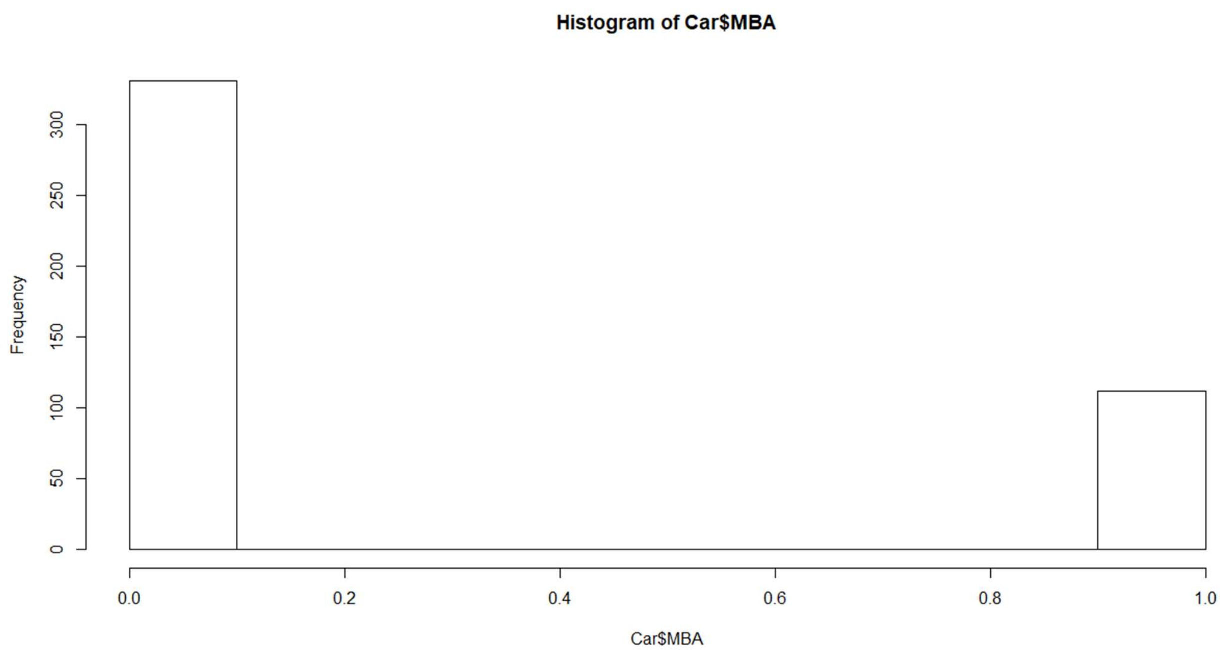
hist(Car$Age)



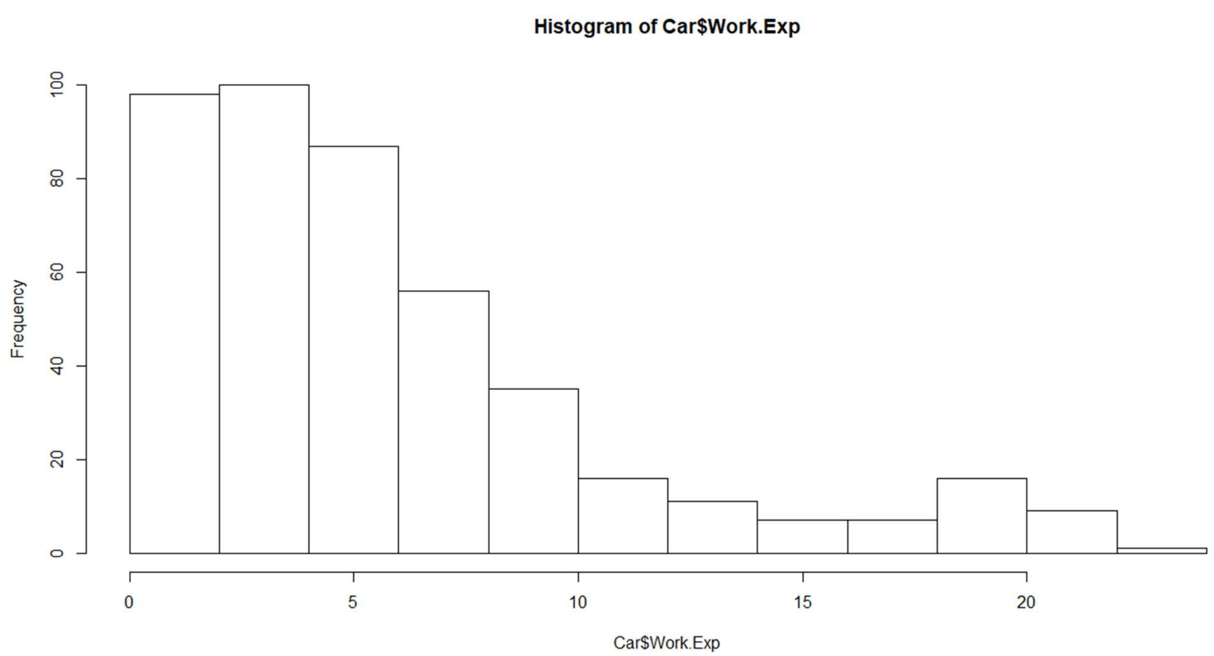
hist(Car$Engineer)



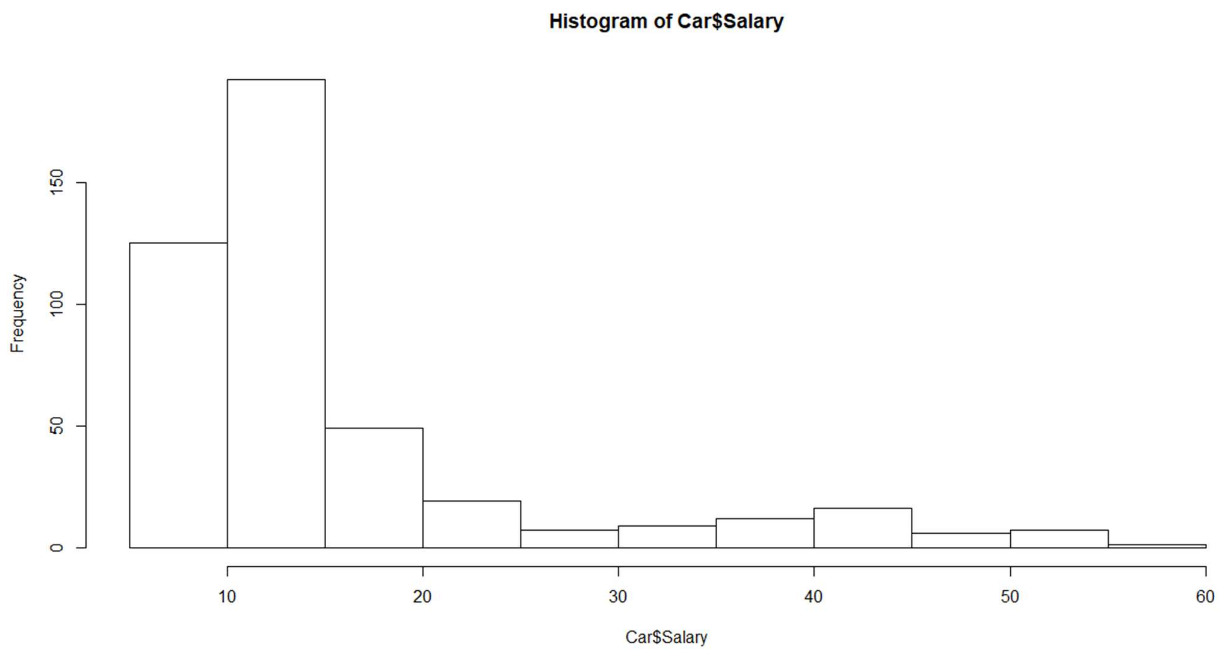
hist(Car$MBA)



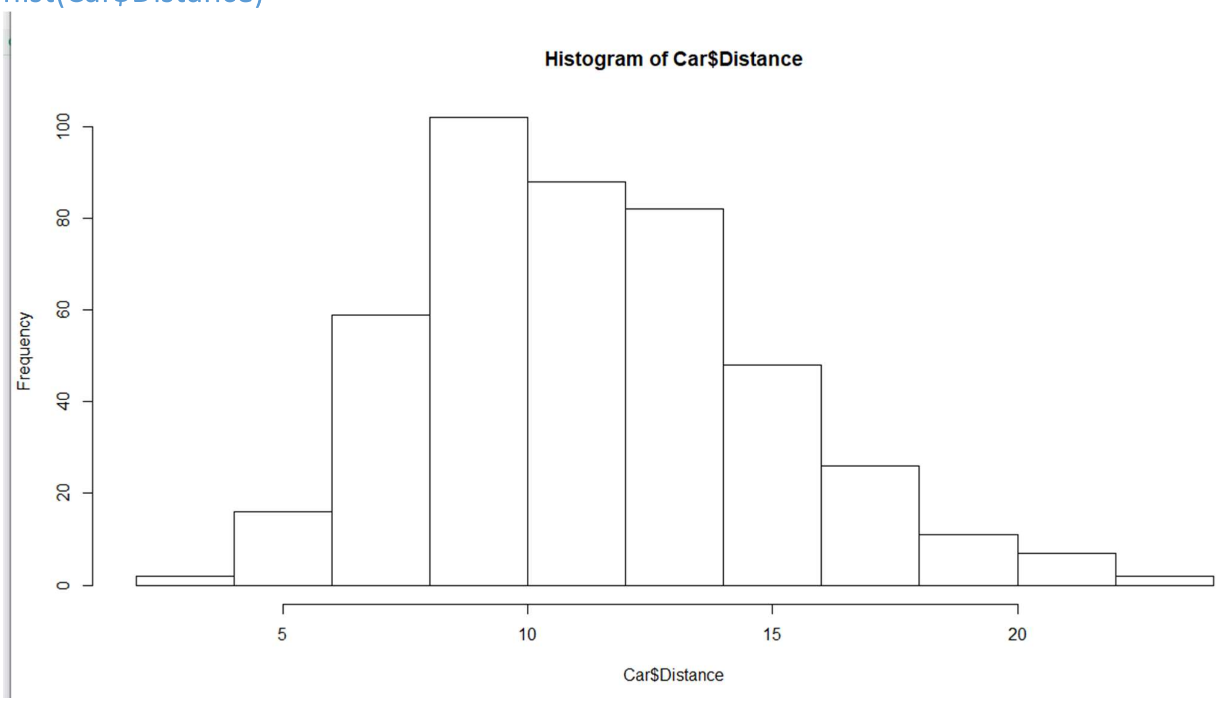
hist(Car$Work.Exp)



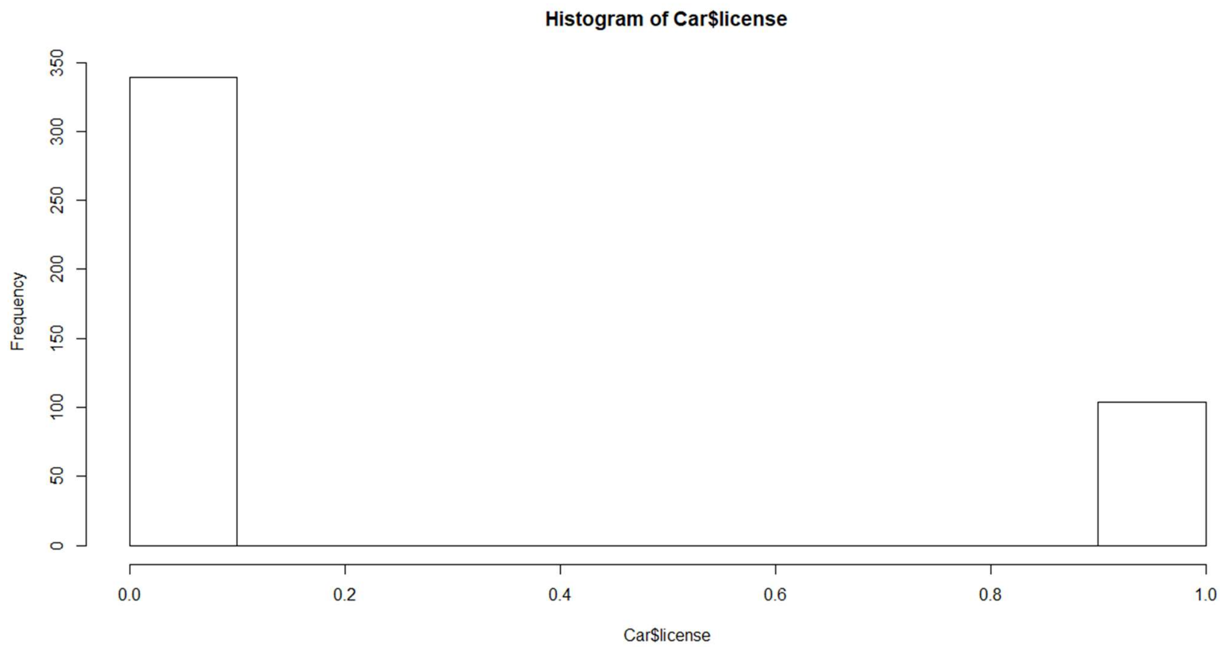
hist(Car$Salary)



hist(Car$Distance)



hist(Car$license)

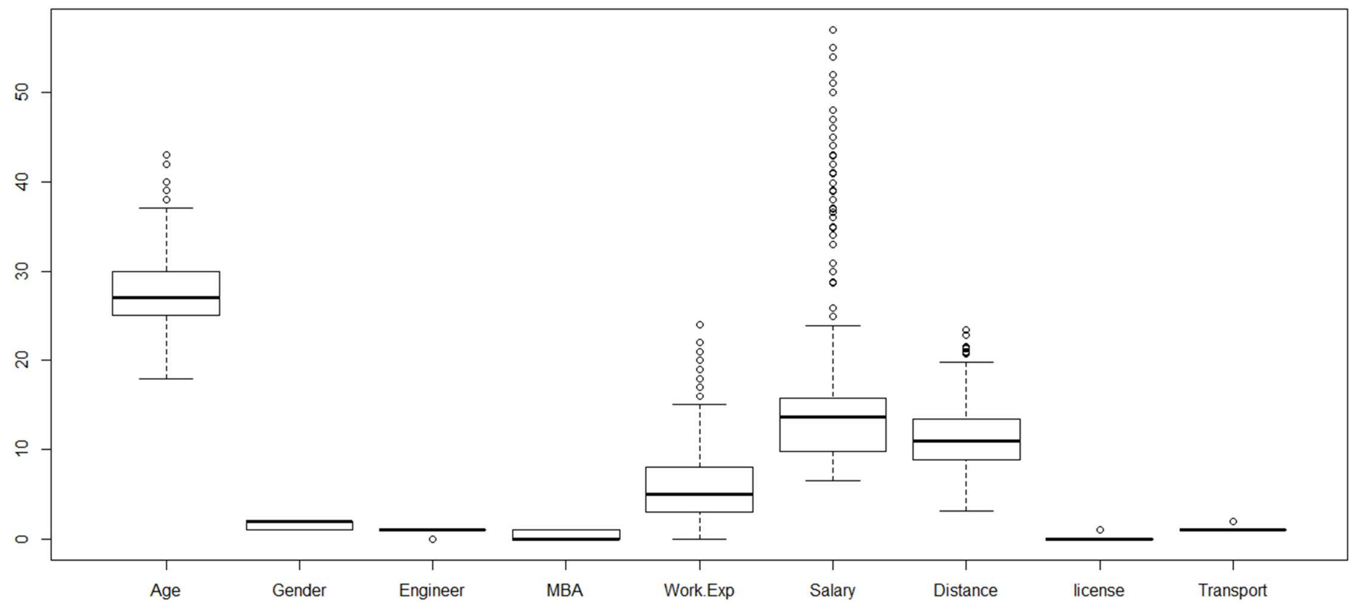


Histogram of Car$license
Frequency
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0
Car$license

*BOXPLOT

boxplot(Car)

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

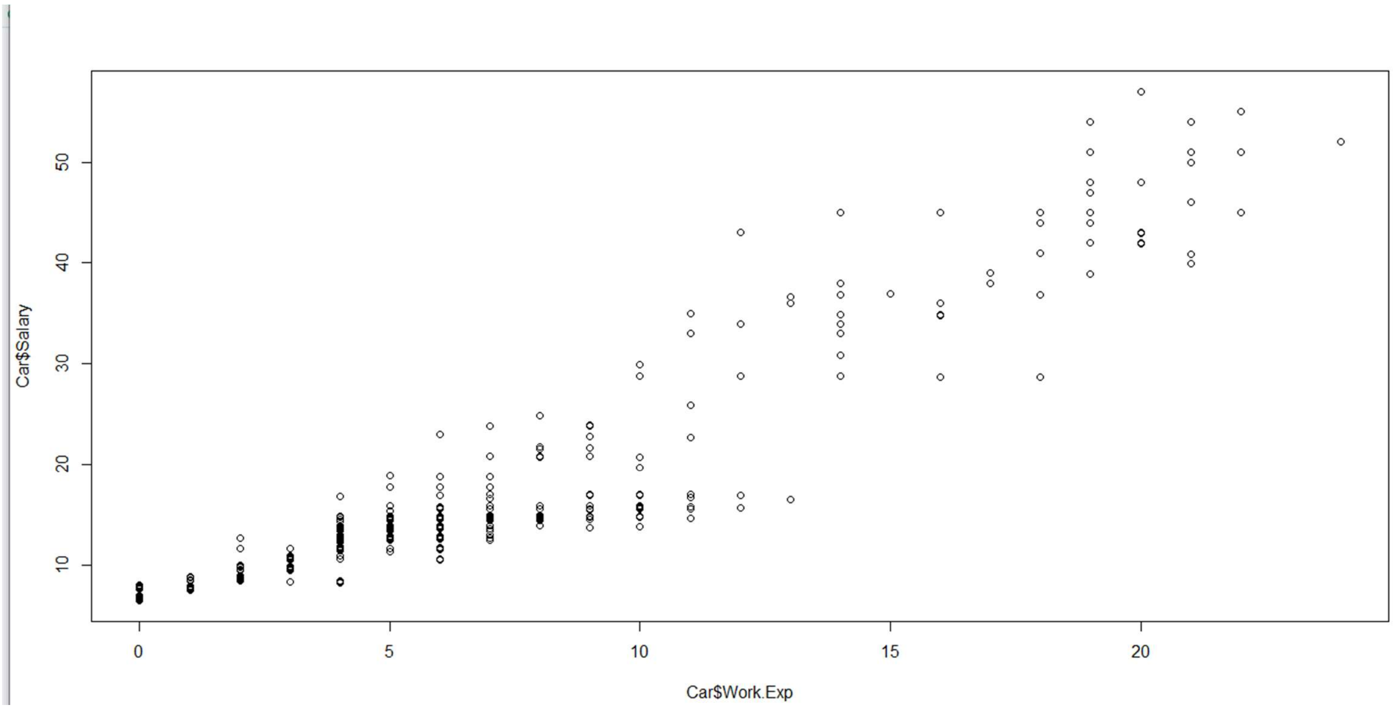


There are outliers in this case. But we are not removing them right now because or data set is small and each row holds great value. We are dealing with an unbalanced dataset in which the minority class is important and is very small in comparison to the majority class.

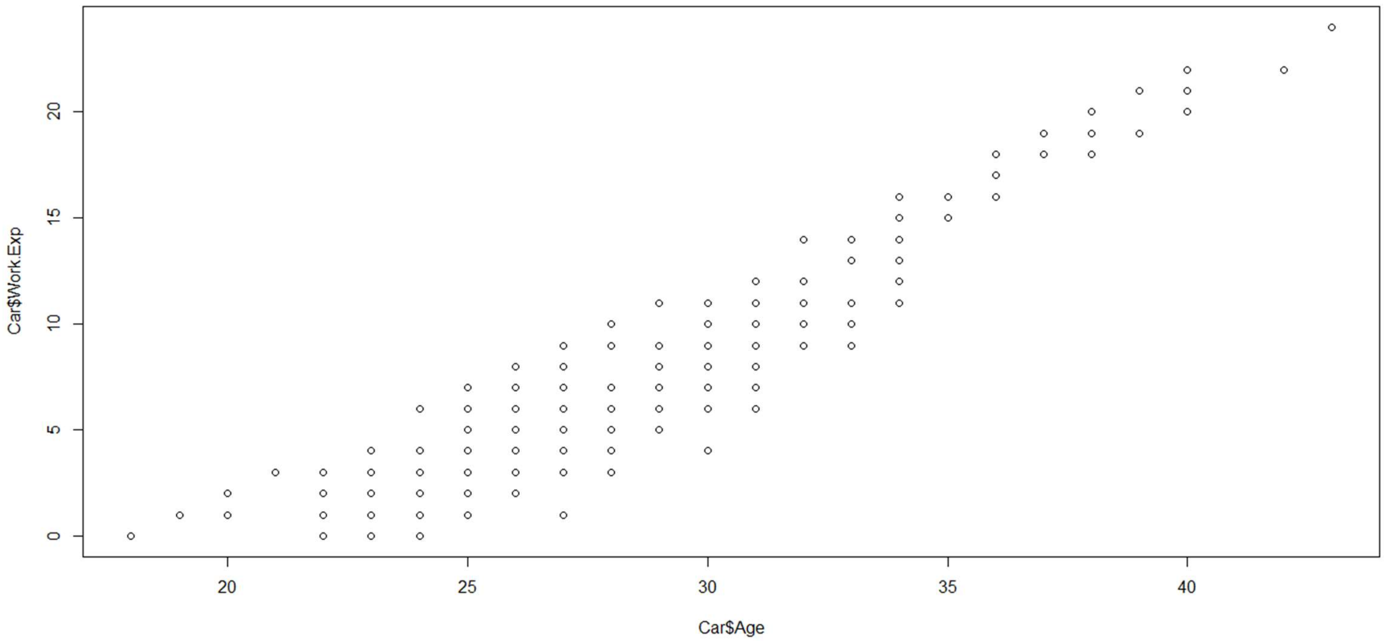
2.2.2 BIVARIATE ANALYSIS

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

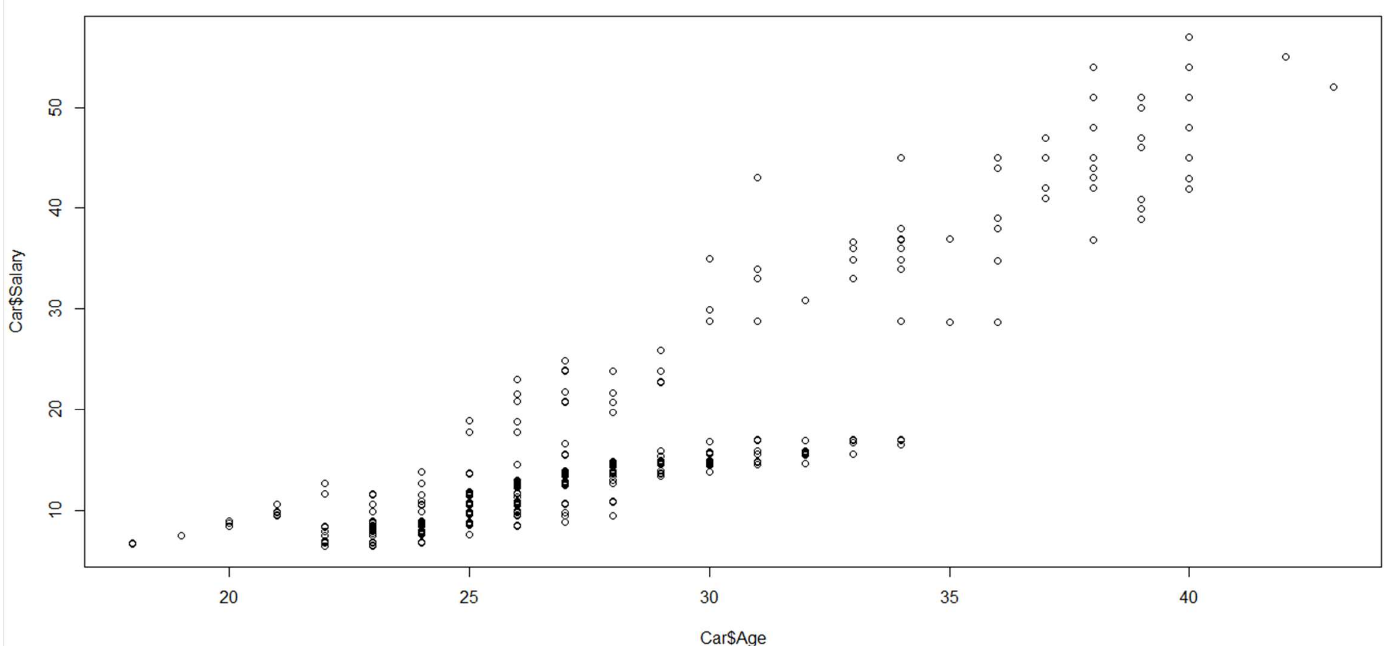
plot(Car$Work.Exp,Car$Salary)



plot(CarAge,CarWork.Exp)



plot(CarAge,CarSalary)



2.3 SUMMARY OF THE INSIGHTS FROM DATA ANALYSIS

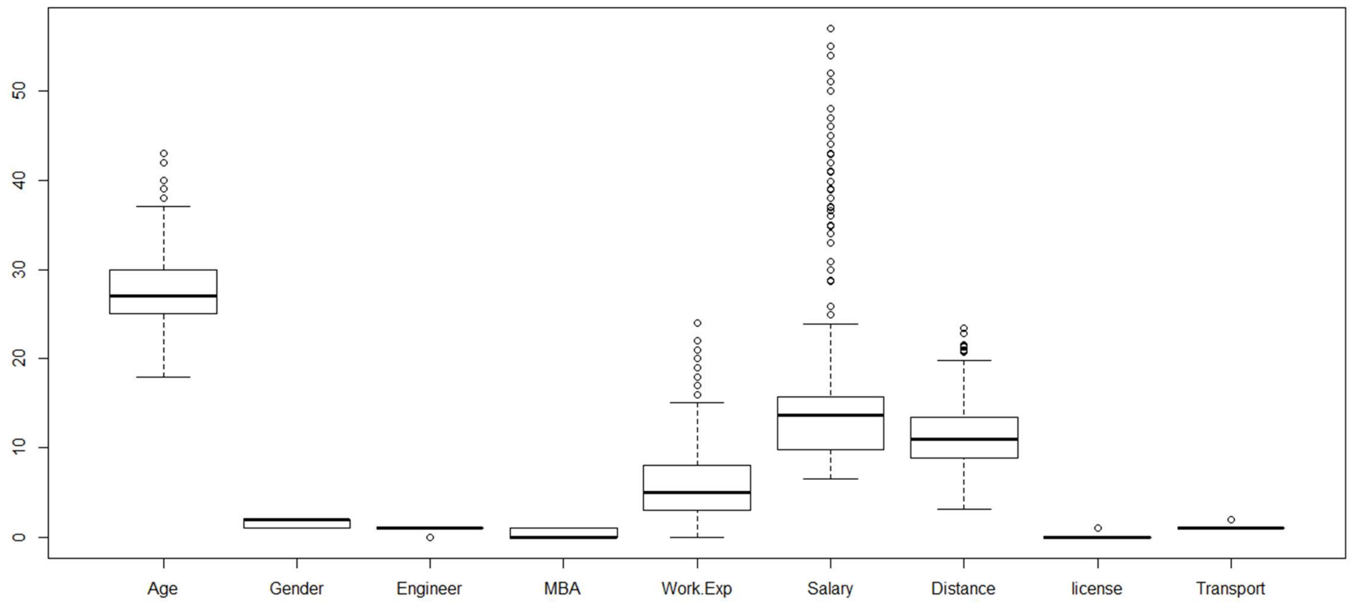
The data consists of all numeric, integer variables, and factor variables.
*There is the issue of multicollinearity amongst the variables.
*There are outliers in certain variable rows: Age, Engineering, Work.Exp, Salary,Distance, License,Transport.
*There is one missing value in the data set.
*The predictor 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))

```
> colSums(is.na(Car))
      Age    Gender  Engineer       MBA  Work.Exp    Salary  Distance
        0         0         0         0         0         0         0
  license Transport
        0         0
> Car<- na.omit(Car)
> colSums(is.na(Car))
      Age    Gender  Engineer       MBA  Work.Exp    Salary  Distance
        0         0         0         0         0         0         0
  license Transport
        0         0
```

str(Car)

```
> str(Car)
'data.frame':   443 obs. of  9 variables:
 $ Age      : int  28 23 29 28 27 26 28 26 22 27 ...
 $ Gender   : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 1 2 2 ...
 $ Engineer : int  0 1 1 1 1 1 1 1 1 1 ...
 $ MBA      : int  0 0 0 1 0 0 0 0 0 0 ...
 $ Work.Exp : int  4 4 7 5 4 4 5 3 1 4 ...
 $ Salary   : num  14.3 8.3 13.4 13.4 13.4 12.3 14.4 10.5 7.5 13.5 ...
 $ Distance : num  3.2 3.3 4.1 4.5 4.6 4.8 5.1 5.1 5.1 5.2 ...
 $ license  : int  0 0 0 0 0 1 0 0 0 0 ...
 $ Transport: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "na.action")= 'omit' Named int 145
  ..- attr(*, "names")= chr "145"
>
```

dim(Car)

```
> dim(Car)
[1] 443   9
>
```

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)

```
> split<- sample.split(Car$Transport,SplitRatio = 0.70)
> dim(Car)
[1] 443   9
> trainData <- subset(Car, split == TRUE)
> dim(trainData)
[1] 310   9
> testData <- subset(Car, split==  FALSE)
> dim(testData)
[1] 133   9
> 
```

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)
```

```
> str(trainData)
'data.frame':   310 obs. of  9 variables:
 $ Age      : int  28 29 28 27 26 28 27 25 27 32 ...
 $ Gender   : Factor w/ 2 levels "0","1": 2 2 1 2 2 2 2 1 2 2 ...
 $ Engineer : int  0 1 1 1 1 1 1 1 1 1 ...
 $ MBA      : int  0 0 1 0 0 0 0 0 0 0 ...
 $ Work.Exp : int  4 7 5 4 4 5 4 4 4 9 ...
 $ Salary   : num  14.3 13.4 13.4 13.4 12.3 14.4 13.5 11.5 13.5 15.5 ...
 $ Distance : num  3.2 4.1 4.5 4.6 4.8 5.1 5.2 5.2 5.3 5.5 ...
 $ license  : int  0 0 0 0 1 0 0 0 1 0 ...
 $ Transport: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> |
```

6. LOGISTIC REGRESSION

6.1 APPLYING LOGISTIC REGRESSION

6.1.1 Creating Logistic Model

dim(smote.train)

```
> dim(smote.train)
[1] 344    9
> 
```

car.logistic <- glm(smote.train$Transport~., data = smote.train,
family = "binomial")
summary(car.logistic)

```
> summary(car.logistic)

Call:
glm(formula = smote.train$Transport ~ ., family = "binomial",
    data = smote.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-3.070   0.000   0.000   0.000   1.287

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -220.6419    65.8648  -3.350 0.000808 ***
Age            7.6373     2.2656   3.371 0.000749 ***
Gender1       -2.6347     1.6855  -1.563 0.118016
Engineer      -6.1789     2.7726  -2.229 0.025846 *
MBA           -4.3792     1.5652  -2.798 0.005145 **
Work.Exp      -3.7652     1.1874  -3.171 0.001519 **
Salary         0.7896     0.3094   2.552 0.010724 *
Distance       0.3813     0.2567   1.486 0.137349
license        8.3734     3.1152   2.688 0.007191 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 476.885  on 343  degrees of freedom
Residual deviance:  23.737  on 335  degrees of freedom
AIC: 41.737

Number of Fisher Scoring iterations: 12

> 
```

6.1.2 Variance Inflation Factor

vif(car.logistic)

```
> # VARIANCE INFLATION FACTOR
> vif(car.logistic)
      Age    Gender  Engineer       MBA  Work.Exp    Salary  Distance 
25.662362  1.659313  3.646330  1.926391 35.792245  8.141355  1.840160 
  license 
 4.989596 
> |
```

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)
```

```
> caret::confusionMatrix(logit.predict,smote.train$Transport)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 170   3
         1   2 169

               Accuracy : 0.9855
                 95% CI : (0.9664, 0.9953)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9709

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9884
            Specificity : 0.9826
         Pos Pred Value : 0.9827
         Neg Pred Value : 0.9883
             Prevalence : 0.5000
         Detection Rate : 0.4942
   Detection Prevalence : 0.5029
      Balanced Accuracy : 0.9855

       'Positive' Class : 0
```

#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)
```

```
> caret::confusionMatrix(logit_pred,smote.test$Transport)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   3
         1   2  15

               Accuracy : 0.9624
                 95% CI : (0.9144, 0.9877)
    No Information Rate : 0.8647
    P-Value [Acc > NIR] : 0.000157

                  Kappa : 0.8355

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.9826
            Specificity : 0.8333
         Pos Pred Value : 0.9741
         Neg Pred Value : 0.8824
             Prevalence : 0.8647
         Detection Rate : 0.8496
   Detection Prevalence : 0.8722
      Balanced Accuracy : 0.9080

       'Positive' Class : 0
```

#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)

```
> summary(knn_fit)
            Length Class      Mode
learn       2      -none-     list
k           1      -none-     numeric
theDots     0      -none-     list
xNames      8      -none-     character
problemType 1      -none-     character
tuneValue   1      data.frame list
obsLevels   2      -none-     character
param       0      -none-     list
```

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])

```
> caret::confusionMatrix(knn.fit,smote.test[,9])
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 109   1
         1   6  17

               Accuracy : 0.9474
                 95% CI : (0.8946, 0.9786)
    No Information Rate : 0.8647
    P-Value [Acc > NIR] : 0.0017

                  Kappa : 0.7987

 Mcnemar's Test P-Value : 0.1306

            Sensitivity : 0.9478
            Specificity : 0.9444
         Pos Pred Value : 0.9909
         Neg Pred Value : 0.7391
             Prevalence : 0.8647
         Detection Rate : 0.8195
   Detection Prevalence : 0.8271
      Balanced Accuracy : 0.9461

       'Positive' Class : 0

> bag.pred<- predict(bagmodel, newdata=smote.test)
```

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
         0 109   1
         1   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

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

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  0   1
0.5 0.5

Conditional probabilities:
   Age
Y       [,1]     [,2]
  0 26.22674 2.993323
  1 35.93034 3.001020

   Gender
Y           0         1
  0 0.3255814 0.6744186
  1 0.3023256 0.6976744

   Engineer
Y        [,1]      [,2]
  0 0.7093023 0.4554101
  1 0.8826595 0.3049996

   MBA
Y        [,1]      [,2]
```

```
   Engineer
Y        [,1]      [,2]
  0 0.7093023 0.4554101
  1 0.8826595 0.3049996

   MBA
Y        [,1]      [,2]
  0 0.2267442 0.4199483
  1 0.1681043 0.3582609

   Work.Exp
Y        [,1]     [,2]
  0  4.488372 3.101572
  1 15.856485 4.378046

   Salary
Y       [,1]      [,2]
  0 12.51163  4.636025
  1 37.02903 12.568223

   Distance
Y       [,1]     [,2]
  0 10.99651 3.217367
  1 15.57483 3.335939

   license
Y        [,1]      [,2]
  0 0.1104651 0.3143839
  1 0.8254747 0.3656260

>
```

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)
```

```
> caret::confusionMatrix(naive.predict,smote.test$Transport)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 111   3
         1   4  15

               Accuracy : 0.9474
                 95% CI : (0.8946, 0.9786)
    No Information Rate : 0.8647
    P-Value [Acc > NIR] : 0.0017

                  Kappa : 0.7803

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.9652
            Specificity : 0.8333
         Pos Pred Value : 0.9737
         Neg Pred Value : 0.7895
             Prevalence : 0.8647
         Detection Rate : 0.8346
   Detection Prevalence : 0.8571
      Balanced Accuracy : 0.8993

       'Positive' Class : 0
```

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
         0 111   3
         1   4  15
```

The Accuracy is= 94.74%

9 .INTERPRETATION OF CONFUSION MATRIX

9.1For Logistic Regression

```
> caret::confusionMatrix(logit_pred,smote.test$Transport)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   3
         1   2  15

               Accuracy : 0.9624
                 95% CI : (0.9144, 0.9877)
    No Information Rate : 0.8647
    P-Value [Acc > NIR] : 0.000157

                  Kappa : 0.8355

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.9826
            Specificity : 0.8333
         Pos Pred Value : 0.9741
         Neg Pred Value : 0.8824
             Prevalence : 0.8647
         Detection Rate : 0.8496
   Detection Prevalence : 0.8722
      Balanced Accuracy : 0.9080

       'Positive' Class : 0
```

#Accuracy : 0.9624
#Sensitivity : 0.9826
#Specificity : 0.8333

9.2 For KNN Model

```
> caret::confusionMatrix(knn.fit,smote.test[,9])
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 109   1
         1   6  17

               Accuracy : 0.9474
                 95% CI : (0.8946, 0.9786)
    No Information Rate : 0.8647
    P-Value [Acc > NIR] : 0.0017

                  Kappa : 0.7987

 Mcnemar's Test P-Value : 0.1306

            Sensitivity : 0.9478
            Specificity : 0.9444
         Pos Pred Value : 0.9909
         Neg Pred Value : 0.7391
             Prevalence : 0.8647
         Detection Rate : 0.8195
   Detection Prevalence : 0.8271
      Balanced Accuracy : 0.9461

       'Positive' Class : 0

> bag.pred<- predict(bagmodel, newdata=smote.test)
```

Accuracy : 0.9474

Sensitivity : 0.9478

Specificity : 0.9444

9.3 For Naïve Bayes’

```
> caret::confusionMatrix(naive.predict,smote.test$Transport)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 111   3
         1   4  15

               Accuracy : 0.9474
                 95% CI : (0.8946, 0.9786)
    No Information Rate : 0.8647
    P-Value [Acc > NIR] : 0.0017

                  Kappa : 0.7803

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.9652
            Specificity : 0.8333
         Pos Pred Value : 0.9737
         Neg Pred Value : 0.7895
             Prevalence : 0.8647
         Detection Rate : 0.8346
   Detection Prevalence : 0.8571
      Balanced Accuracy : 0.8993

       'Positive' Class : 0
```

#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
         0 109   1
         1   6  17
```

11. Bagging Ensemble Method

11.1 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

```
> table.bag

   TRUE
 0  115
 1   18
```

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%

11.2 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
0  115
1   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
```

```
> car_xgb
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   14   14   14   14   14   14
```

*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
```

```
> car_xgb2
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   14   14   16   16   15   15   17
```

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
```

```
> car_xgb3
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    9    9    9    9    9    9
> bag.pred<- predict(bagmodel, newdata=smote.test)
```

*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)
```