EMPLOYEE CAR USAGE ANALYSIS

-Vompolu Sai Tanuj

PGP-BABI(JUNE 2019) G1

Table of contents:

1) Project Objective:

The **project** is based on the study of **mode of transport** that is used by the **employees** of an organisation to **commute** on daily basis. We are given a **sample data**(*cars.csv)* containing

the **mode of transport** employees use as well as some of the details such as **salary, Age, etc.** related to the employee. The **management** wants to predict whether an employee working under their company will use a **car** or not. As an **Analyst,** we need to create a model to **predict** the **best model** possible which can predict the **usage of car** by the employees.

2) Exploration of the dataset(EDA):

The dataset provided contains the data of the employees' details like **Salary, Age , Distance from home, etc.** and also their **mode of transport.** The **mode of transport** is our **main concern.** The dataset is given to us in a **csv("Comma Seperated Values")** namely **cars.csv** . This data will help us in creating the **right classification models** to **predict** the **car usage** by the employees.

a) Invoking the required Packages to be used in R Code:

The **required packages** must be **installed** and their **respective libraries** must be **called out** in the R code before calling out any function from that library. The libraries if not found in the R directory, they can be installed by using **install.packages ("package name").** The **Rstudio** must be connected to the internet so that required package gets downloaded and installed. After installation, the library must be called into the R code using the function, **library(packagename).** For this project, the following libraries must be invoked to be used in the R code.

- **readr –** This library is used to **import the .csv file** into the **R** directory.

```
install.packages("readr")
library(readr)
```

- **hmisc –** This library is useful to get **multiple histograms** for **the whole dataframe.**

```
install.packages("Hmisc")
library(Hmisc)
```

- **ggplot2 –** This library is used to get lot of **graphs** that will be useful in the **Uni-Variate and Bi-Variate** analysis.

```
install.packages("ggplot2")
library(ggplot2)
```

- **psych –** This library is used to get the **correlation plot** along with **values.**

```
install.packages("psych")
library(psych)
```

- **caret –** This library can be used to get the **confusion matrix** for various measures.

```
install.packages("caret")
library(caret)
```

- **caTools –** This library is useful to create splits and make **separate training** and **testing data.**

```
install.packages("caTools")
library(caTools)
```

- **class –** This library is used to make **KNN** models.

```
install.packages("class")
library(class)
```

- **e1071 –** This library is useful to create **Naïve Bayes** model.

```
install.packages("e1071")
library(e1071)
```

- **DMwR –** This library is useful in creating **Bagging model.**

```
install.packages("DMwR")
library(DMwR)
```

- **xgboost –** This library is useful in creation of **XG Boost model.**

```
install.packages("xgboost")
library(xgboost)
```

- **gbm –** This library is used in creating **bagging models**

```
install.packages("gbm")
library(gbm)
```

- **ipred –** This library is useful in creating **Adaboost models.**

```
install.packages("ipred")
library(ipred)
```

- **rpart –** This library is useful in control the **tree paratmeters.**

```
install.packages("rpart")
library(rpart)
```

- **Matrix –** This library is useful to **create data matrices** that will be helpful in creating the **XG boost** models.

```
install.packages("Matrix")
library(Matrix)
```

b) Creating a working directory and importing the dataset:

The working directory in the **R Console** must be set to the directory where the XLSX file exists. The working directory can be changed using

the function **setwd().** To get the current directory of the R Console, the function **getwd()** can be used.

```
> ### Setting up the working directory ###
> setwd("C:/R programs great lakes/ML/project")
> getwd()
[1] "C:/R programs great lakes/ML/project"
```

The **dataset** for the **project,** is stored in a **csv file** within the **working directory.** The name of the **csv file** is **"cars.csv".** The **dataset** can be imported into the **R Code** using the function **read.csv()** function with the **argument, header = TRUE.** The imported dataset can be seen using the function **View().** The **dataset** thus imported will be named as **'trans'.**

```
> ### Importing the dataset ####
> trans = read.csv("Cars.csv",header = TRUE)
> View(trans)
```

c) Identification of different variables:

To able to do **uni-variate** and **bi-variate** analysis, we must first be able to understand all the **variables** in the **data-frame** and get a basic idea on the data-frame before the analysis. The following functions can be used to achieve the above:

- **dim() -** This function can be used to get the **dimensions** of the data-frame.

```
> dim(trans)
[1] 418   9
```

- **str()** – This function can be used to the get the **classes** of all the variables available along with the **values** in the variable.

```
> str(trans)
'data.frame':   418 obs. of  9 variables:
 $ Age      : int  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 2 2 2 2 ...
 $ Engineer : int  1 1 1 0 0 0 1 0 1 1 ...
 $ MBA      : int  0 0 0 0 0 0 1 0 0 0 ...
 $ Work.Exp : int  5 6 9 1 3 3 3 0 4 6 ...
 $ Salary   : num  14.4 10.6 15.5 7.6 9.6 9.5 11.7 6.5 8.5 13.7 ...
 $ Distance : num  5.1 6.1 6.1 6.3 6.7 7.1 7.2 7.3 7.5 7.5 ...
 $ license  : int  0 0 0 0 0 0 0 0 0 1 ...
 $ Transport: Factor w/ 3 levels "2Wheeler","Car",..: 1 1 1 1 1 1 1 1 1 1 ...
```

- **head()** – This function is used to give the top first few values of each of the variable. An additional argument can be passed to dictate the number of values to be shown.

```
> head(trans)
  Age Gender Engineer MBA Work.Exp Salary Distance license Transport
1  28   Male        1   0        5   14.4      5.1       0  2Wheeler
2  24   Male        1   0        6   10.6      6.1       0  2Wheeler
3  27 Female        1   0        9   15.5      6.1       0  2Wheeler
4  25   Male        0   0        1    7.6      6.3       0  2Wheeler
5  25 Female        0   0        3    9.6      6.7       0  2Wheeler
6  21   Male        0   0        3    9.5      7.1       0  2Wheeler
```

- **tail()** – This function is used to give the **bottom last few** values of each of the variable. An additional argument can be passed down to dictate the number of values to be shown.

```
> tail(trans)
    Age Gender Engineer MBA Work.Exp Salary Distance license        Transport
413  29 Female        1   0        6   14.9     17.0       0 Public Transport
414  29   Male        1   1        8   13.9     17.1       0 Public Transport
415  25   Male        1   0        3    9.9     17.2       0 Public Transport
416  27 Female        0   0        4   13.9     17.3       0 Public Transport
417  26   Male        1   1        2    9.9     17.7       0 Public Transport
418  23   Male        0   0        3    9.9     17.9       0 Public Transport
```

- **summary()** – This function gives us an basic idea on values on the data-frame by performing the **basic statistics** on it.

```
> summary(trans)
      Age          Gender       Engineer          MBA            Work.Exp    
 Min.   :18.00   Female:121   Min.   :0.0000   Min.   :0.0000   Min.   : 0.000  
 1st Qu.:25.00   Male  :297   1st Qu.:0.2500   1st Qu.:0.0000   1st Qu.: 3.000  
 Median :27.00                Median :1.0000   Median :0.0000   Median : 5.000  
 Mean   :27.33                Mean   :0.7488   Mean   :0.2614   Mean   : 5.873  
 3rd Qu.:29.00                3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 8.000  
 Max.   :43.00                Max.   :1.0000   Max.   :1.0000   Max.   :24.000  
                                               NA's   :1                        
     Salary          Distance        license                   Transport  
 Min.   : 6.500   Min.   : 3.20   Min.   :0.0000   2Wheeler        : 83  
 1st Qu.: 9.625   1st Qu.: 8.60   1st Qu.:0.0000   Car             : 35  
 Median :13.000   Median :10.90   Median :0.0000   Public Transport:300  
 Mean   :15.418   Mean   :11.29   Mean   :0.2033                         
 3rd Qu.:14.900   3rd Qu.:13.57   3rd Qu.:0.0000                         
 Max.   :57.000   Max.   :23.40   Max.   :1.0000                         
```

Inferences:

- The **dataset trans** has **418 Rows** and **9 Columns.**
- **Transport** variable is our **dependent variable** and the **variable** in focus for the project.
- The rest of the **8 variables** are **independent variable** and will prove **useful** in the preparation of the model.
- Except for the **Transport** and **Gender** variable, the rest of the variables contain **numeric values**.
- **Transport** and **Gender** are the only variables not converted into **numeric** variables while the rest of the variables are in **numeric variables**.
- Looking at the **dataset,** we can say that the dataset has been **arranged** from the top in the ascending order of the **distances from the offices.**
- The age of the **youngest employee** in the company is **18 years** and the **oldest employee** of the company is **43 years**
- The **average salaries** of the customers in the company is **15.4.**
- The **least distance commuted** by an employee is **3.20 km,** while the **highest distance commuted** by an employee is **23.40km.**
- The variable **MBA** has one **missing value.**
- The variable **Gender** has two levels namely, **Male** and **Female.**

- The variable **Transport** has **three levels** namely, **2Wheeler, Car** and **Public Transport.**

*Before performing the **Uni-Variate** and **Bi-Variate Analysis,** the variables like **Gender, MBA, etc.** must be converted to **factors** with **labels "1"** and **"0".***

*The variables **MBA, Engineer and license** can be converted to factor variables with **1 as Yes and 0 as No***

*The variable **Gender** can be converted into factor variable containing **1's and 0's** for **Male** and **Female** respectively.*

*The variable **Transport** can be converted into factor variable containing **1's and 0's** for **Car users** and **Non-Car users (Public transport and 2Wheeler)** respectively.*

```
> ### Conversion of cateogrical variable ###
> trans$Gender = as.factor(trans$Gender)
> trans$Gender = ifelse(trans$Gender == "Male","1","0")
> trans$Gender = as.factor(trans$Gender)
> trans$license = as.factor(trans$license)
> trans$Transport = ifelse(trans$Transport == "Car","1","
> class(trans$Transport)
[1] "character"
> trans$Transport = as.factor(trans$Transport)
> trans$Engineer = as.factor(trans$Engineer)
> trans$MBA = as.factor(trans$MBA)
```

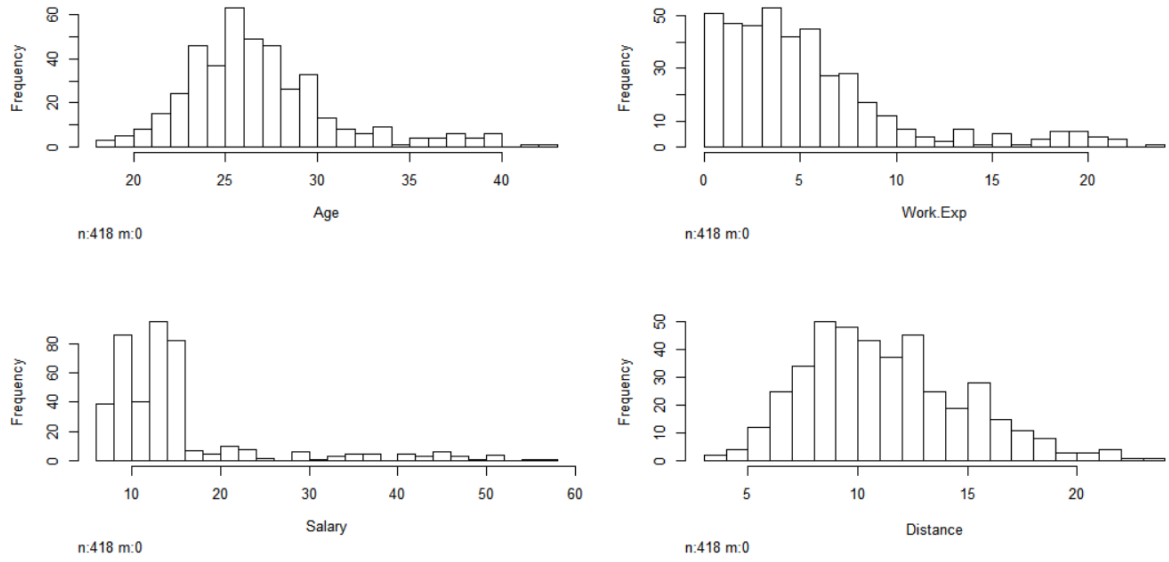
d) Uni-Variate Analysis:

In the data-frame, we have two types of variables namely **Categorical** and **numerical.** The Numerical variables can be analysed plotting **Histograms.** These can be constructed by using **hist.data.frame()** function. The **Categorical variable** can

be analysed by using the **frequency table** and **Bar Plots.** The **frequency table** can be plotted using the function **table()** and the **barplot** can be plotted using the function **plot()**.

Numerical Independent Variables:

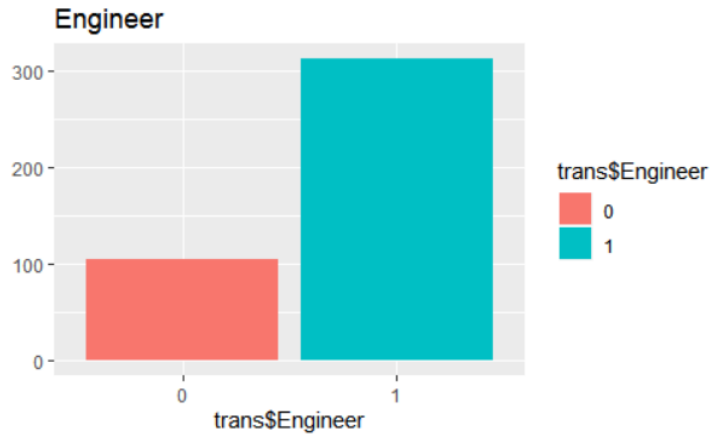
```
> ### Uni-Variate Analysis ####
> ## Analysis of independent numerical variable ##
> cor = trans[,-c(2,9)]
> hist.data.frame(cor)
```



Categorical Independent Variables:

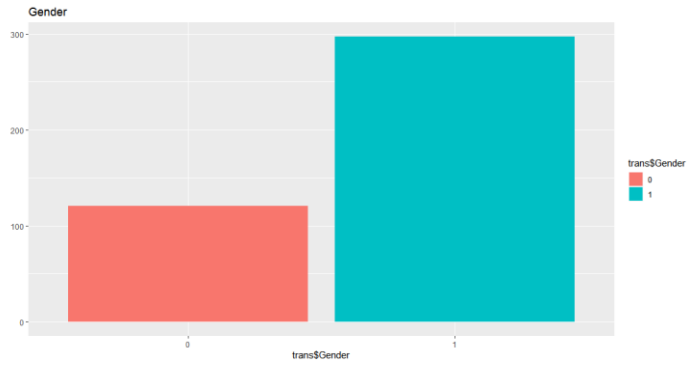
Engineer:

```
> ## Analysis of independent categorical variable ##
> # Engineer #
> qplot(trans$Engineer,fill = trans$Engineer,main = "Engineer")
> prop.table(table(trans$Engineer))

        0         1
0.2511962 0.7488038
```



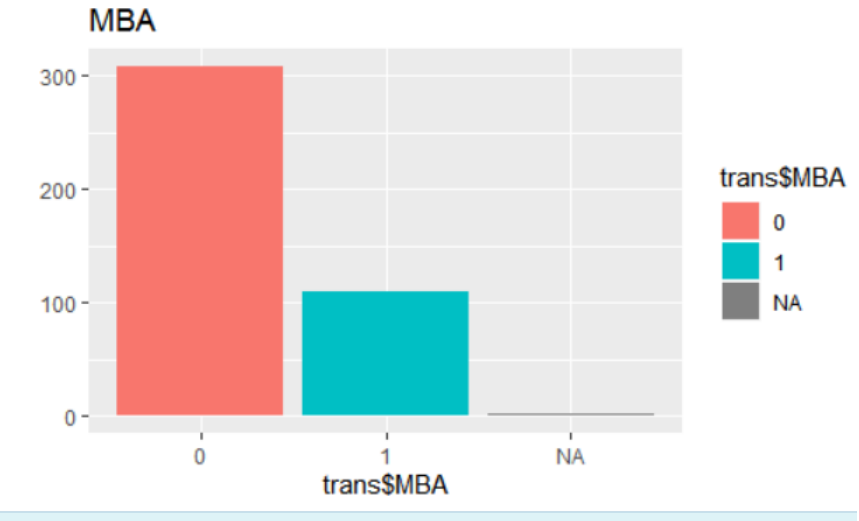
Gender:

```
> # Gender #
> qplot(trans$Gender,fill = trans$Gender,main = "Gender")
> prop.table(table(trans$Gender))

        0         1
0.2894737 0.7105263
```



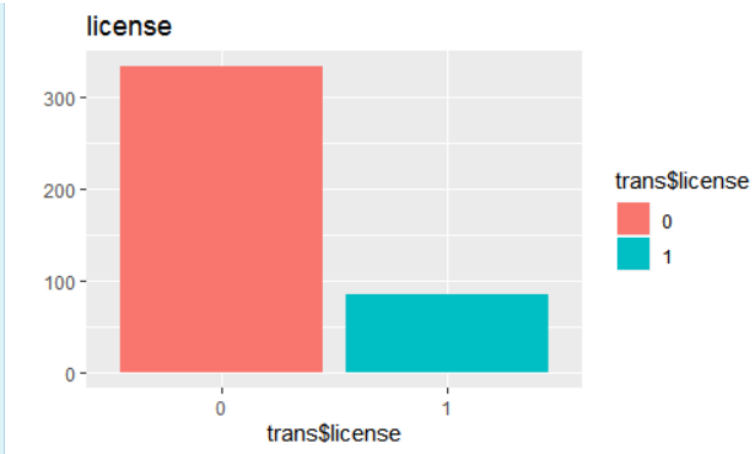
MBA:

```
> # MBA #
> qplot(trans$MBA,fill = trans$MBA,main = "MBA")
> prop.table(table(trans$MBA))

         0          1
0.7386091 0.2613909
```



License:

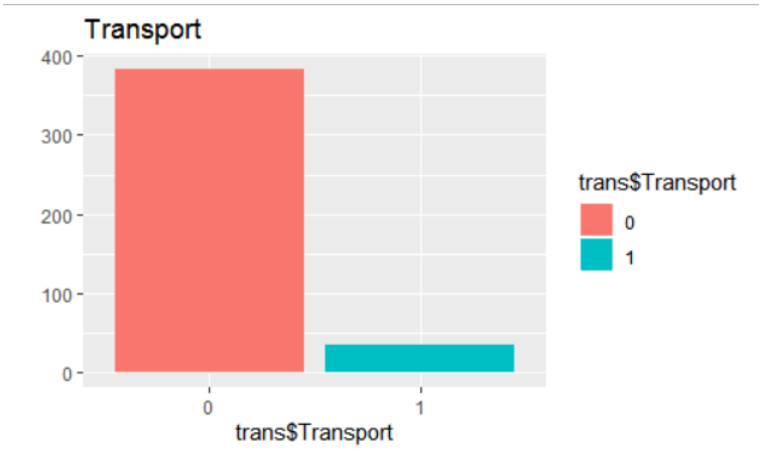
```
> # License #
> qplot(trans$license,fill = trans$license, main = "license")
> prop.table(table(trans$license))

        0         1
0.7966507 0.2033493
```



Categorical Dependent Variable:

Transport:

```
> ## Analysis of dependent categorical variable
> # Transport #
> qplot(trans$Transport,fill = trans$Transport, main = "Transport")
> table(trans$Transport)

  0   1
383  35
> prop.table(table(trans$Transport))

        0         1
0.91626794 0.08373206
```



Inferences:

- The variables **Salary** and **Work Experience** are the only ones that are **extremely left skewed.**
- The variable **Age** is **slightly left skewed.**
- The **Distance** variable is normally distributed with no **skew.**
- All the variables are **normally distributed.**
- We see that most of the employees are aged between **20 and 30 years,** while very few are present **in the range of 35 to 40 years.**
- Most of the employees are fresher with **Work Experience** ranging from **0 to 5 years**
- The company has **more number of employees** whose **Salary** ranges from **10 to 15.**

- The **distance commuted** of most of the employees is around **10 km.**
- We can see that the **Work.Exp** and **Salary** are both extremely skewed indicating a **high collinearity** between them.
- We can see that over **70%** of the employees are **engineers.**
- The company is **male-dominated** as **71%** of the employees are **males.**
- The majority of the **employees** do **not** possess a **MBA.**
- Only the **20%** of the **employees** have a **driving license.**
- Only **35(0.08%)** of the **employees** use **car** as **mode of transport** while **383**(**0.92%**) of the employees **commute** either by **2Wheeler** or **public transport.**

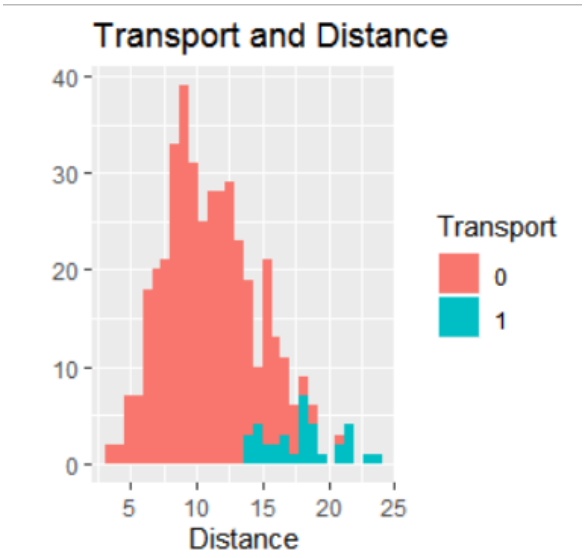
e) Bi-Variate Analysis:

The **Bi-Variate Analysis** can be done with different combination of variables, **(Categorical vs. Categorical, Categorical vs Numerical and Numerical vs Numerical)** by using the **qplot()** function from the **ggplot2** library and changing the arguments accordingly to get the required analysis.

Analysis of Dependent variable with Numerical variable:

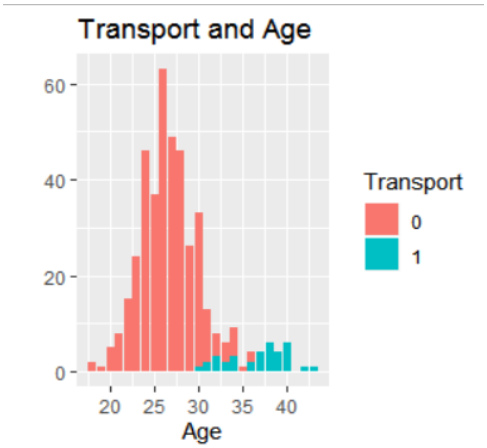
Distance

```
> ### Bi-Variate Analysis ###
> ## Analysis of dependent variable with numerical variables ##
> qplot(Distance,fill = Transport,data = trans,
+       main = "Transport and Distance")
```



Age

```
> qplot(Age,fill = Transport,data = trans,
+       main = "Transport and Age",geom = "bar")
```

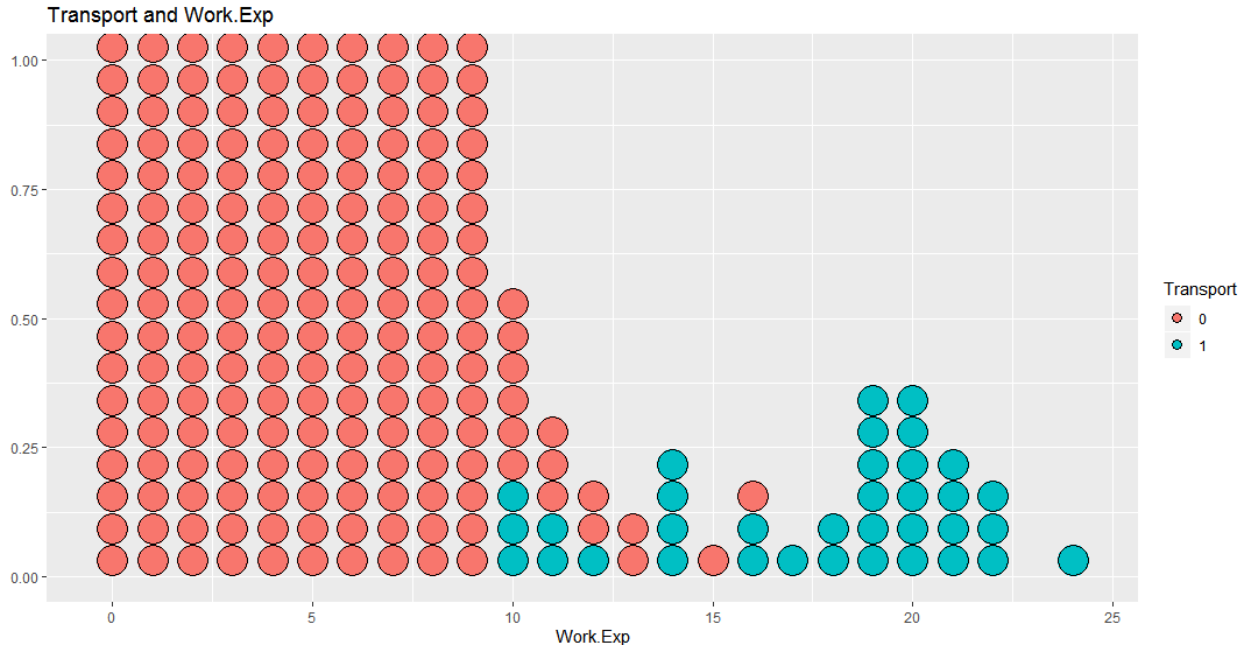


Salary

```
> qplot(Salary,fill = Transport,data = trans,
+       main = "Transport and Salary",
+       geom = "density")
```

Work.Exp

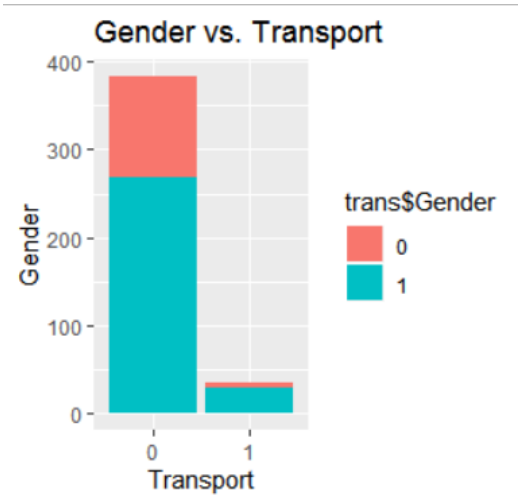
```
> qplot(Work.Exp,fill = Transport,data = trans,
+       main = "Transport and Work.Exp",
+       geom = "dotplot")
```



Analysis of Dependent variable with Categorical variable:

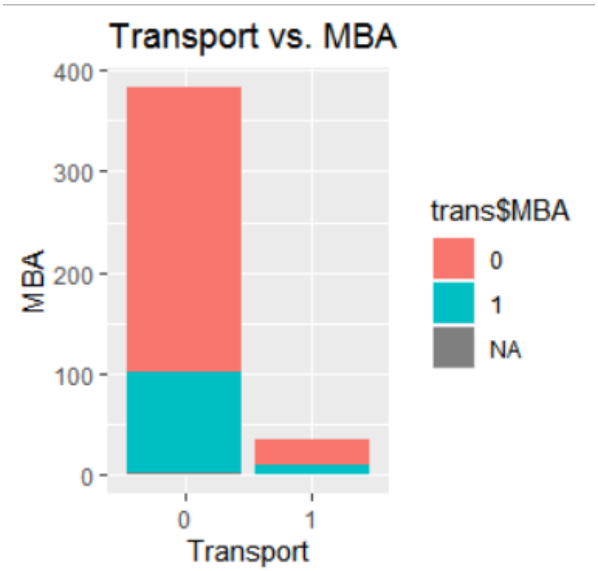
Gender

```
> ## Analysis of dependent variable with categorical variables ##
> qplot(trans$Transport,fill = trans$Gender,xlab = "Transport",
+       ylab = "Gender",main = "Gender vs. Transport")
```



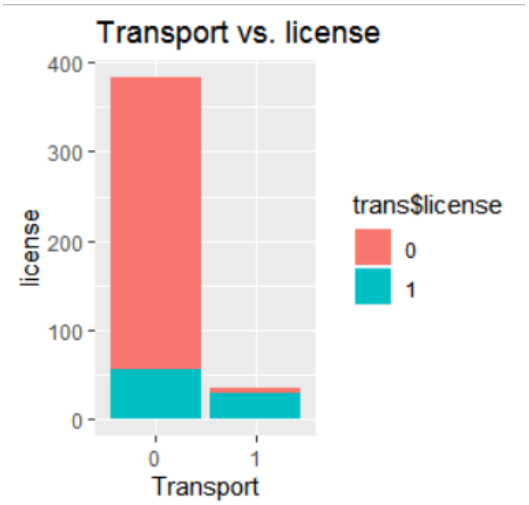
MBA

```
> qplot(trans$Transport,fill = trans$MBA,xlab = "Transport",
+       ylab = "MBA",main = "Transport vs. MBA")
```



License

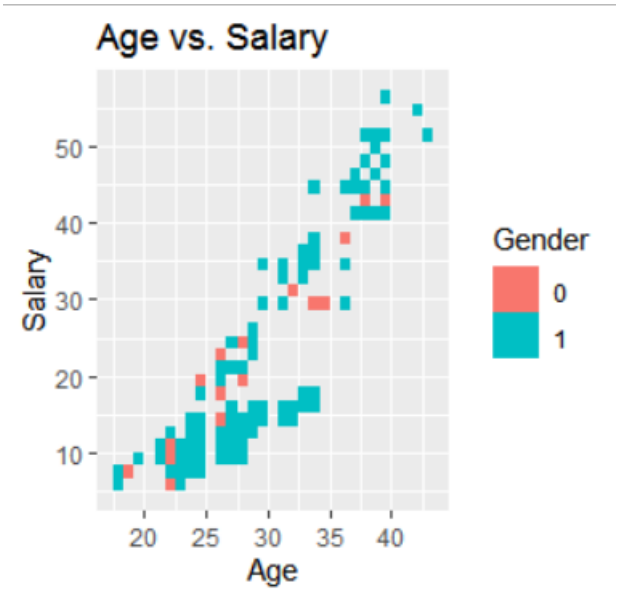
```
> qplot(trans$Transport,fill = trans$license,xlab = "Transport",
+       ylab = "license",main = "Transport vs. license")
```



Analysis of Independent variables with Independent variables:

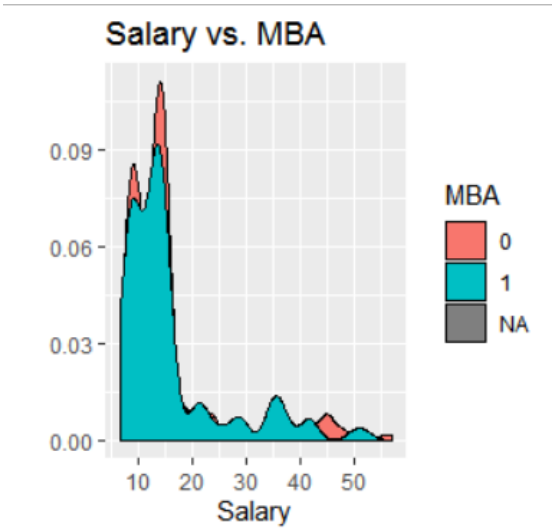
Age vs. Salary

```
> ## Analysis of independent variables with independent variables ##
> qplot(Age,Salary, fill = Gender,data = trans,
+       geom = "bin2d",main = "Age vs. Salary")
```



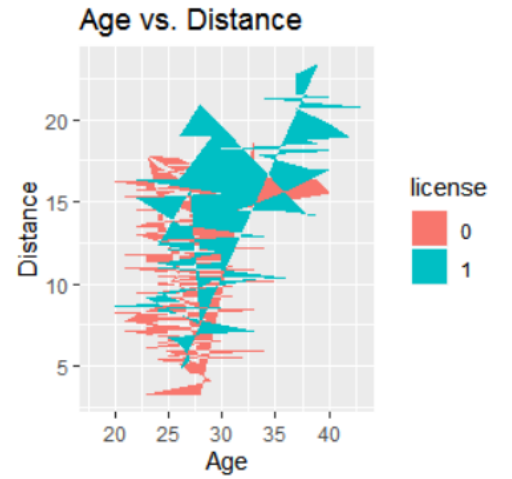
Salary vs. MBA:

```
> qplot(Salary,fill = MBA,data = trans,geom = "density",
+       main = "Salary vs. MBA")
```



Age vs. Distance

```
> qplot(Age,Distance,data = trans,fill = license,geom = "polygon",
+       main = "Age vs. Distance")
```



Work.exp vs. Salary

```
> qplot(Work.Exp,Salary,fill = Engineer,data = trans,
+       geom = "boxplot",main = "Work.Exp vs. Salary")
```

Inferences:

- We can see from above that **employees** who are nearer prefer to go either by **public transport or 2 wheeler** as it is **cheaper**
- The **younger employees** in the company prefer to go either by **2Wheeler** or **public transport** because they can withstand the **strain of the transport by 2 wheeler or public transport** unlike the **older employees** who prefer the **comfort of the car.**
- It is clearly evident that only the employees with **higher salaries** use **car** for commuting.
- Since having higher **work experience** leads to **higher job position** in the company, it is more likely that they are going to **use car** as mode of transport

- We have more number of **Females** who are car users than the **males** as car is more **comfortable and secure** when compared to **2wheeler and Public transport..**
- We can see that **non-car users** consist of more number of people who have **not** done their **MBA** which could because the employees those who don't possess an **MBA** don't hold **high positions** in the company.
- Since the **oldest** employees with **highest salaries** are **males,** we can say that the **all the top level employees are males.**
- We can see that the employees **older in age** stay **farther** than their **younger employees,** and hence it makes sense that they possess **a driving license to use car..**
- We can see **Engineers** with **more work. Exp** earn more than the **Non-Engineer** employees.

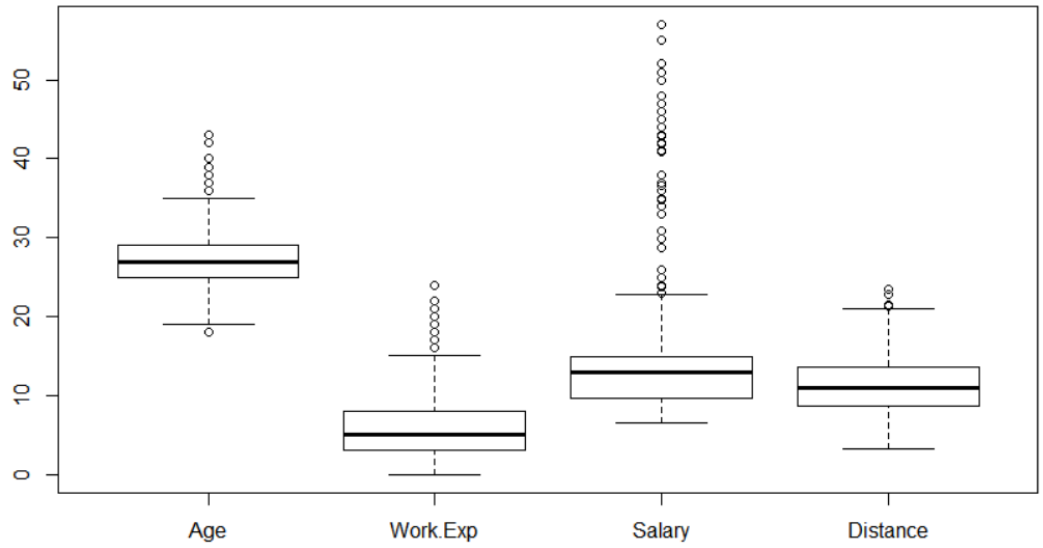
f) Missing values treatment:

From the results in **initial exploration,** it was clear that the **variable MBA** had **one** missing value. Since, it is only **one value,** we can convert that **missing value** into **1** or **0.** We prefer to convert it to **1** since there are more number of **0's** than **1's** in the **MBA** variable. The **missing values** can be found out by using the function **sum(is.na()).**

```
> ### Missing values treatment ####
> sum(is.na(trans))
[1] 1
> trans[is.na(trans)] = 1
> sum(is.na(trans))
```

g) Outlier treatment:

The **Outliers** can be termed as the values that are **1.5 times lesser than first quartile or 1.5 times more than third quartile.** The best way to detect outliers is by plotting **boxplots** using the function **boxplot()**.



Inferences:

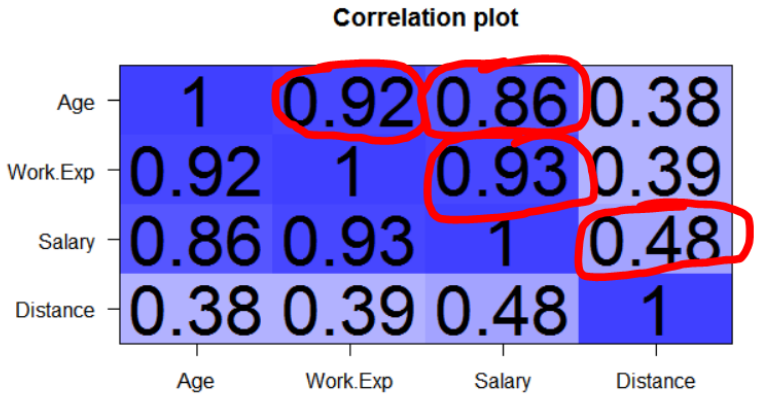
We can see that the variable **salary** has the highest number of **outliers** while the variable **distance** has the least number of **outliers.** The variables **Age and Work. Exp** have a **significant number** of **outliers.**

h) Checking for Multicollinearity:

The Multicollinearity between variables can be termed as **the existence of collinearity between two or more variables.** The **Multicollinearity** can be checked in the following methods.

i. *Using a Correlation Plot:*

The correlation plot can plotted using the **cor.plot()** function.



Inferences:

We can see that all the variables **Age, Work.exp, Salary and Distance have higher Correlation** values between them.

ii. *Checking the eigen values:*

The **Eigen Values** help in explaining the spread of the values in the variables.

```
> eigen = eigen(cor(trans[,c(1,5,6,7)]))
> eigen$values
[1] 3.06161749 0.75740812 0.13516991 0.04580448
```

Inferences:

We can see that there is **one value** which is **near** to the 0. Therefore we can say that **MultiCollinearity** is present between the variables.

iii. ***Using Scatter Plot:***

Plotting a scatter plot for all the numerical variables using the **plot()** function can help in identify the Multicollinearity.

```
> plot(cor)
```

Inferences:

From the above **scatter plots,** we can see that there is a **significant correlation** between **Age, Work. Exp and Salary.**

i) EDA Insights:

The following insights were drawn from the **EDA:**

- Even though they follow **normal distribution,** the variables **Salary and Work Experience** are heavily **skewed** to the left indicating that **low level employees** make up most population of the company.
- There are very few employees who have **High Salaries, High Work. Experience** and are on the **elder side,** which

can be a good reason for the less number of **car users** in the company.

- Many of the employees haven't done their **MBA** but many of them are **engineers** indicating that the company is **more** on the **technical side** and has more **technical employees** rather than **management employees.**
- The target variable is **highly imbalanced** with **92-08 %** distribution. This imbalance will **highly affect** the model performance measures.
- Even the **independent variables** in the dataset have **high imbalance** and can be the reason to affect the **model performance.**
- Even the employees who have done their **MBA,** do not get higher salaries, indicating more that **company is prone** towards the **technical side.**
- The **younger employees** prefer to stay close to the company and use **2wheeler or public transport** rather than **stay away** and use **car.**
- All the **numerical variables** have **many outliers** which maybe affect the **predictions** of the **models.**
- The existence of **multicollinearity** may **decrease the significance** of all the variables accompanying it and sometimes may altogether **make the variable insignificant.**

3) Model Building and Comparison:

The target variable is a **categorical variable.** And hence to make predictions, we must create **classification models** such as **Logistic Regression, Naïve Bayes, KNN, etc.** We create these models and compare their metrics to determine the **best model.** While considering the **metrics**, we must keep in mind that **the management** is interested in the **number of car users** and wants to predict if an employee **uses a car** or **not.** The metrics to be considered are as follows:

- **Precision:** It measures the rate of **False Negatives.** *(The Higher the better)*

  Precision = $\frac{TP}{TP+FP}$

- **Recall:** It measures the **False Negatives** against **True Positives.** *(The Higher the better)*

  Recall = $\frac{TP}{TP+FN}$

- **Accuracy:** It is the number of **correct predictions** over the total output. *(The Higher the better)*

  Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

- **Balanced Accuracy:** It is the balance of accuracy obtained while predicting **1's and 0's** *(The Higher the better)*

  Balanced Accuracy = $\frac{Sensitivity+Specificity}{2}$

Before we go ahead with building models, we first must split the data given to us into two parts, namely **Training and Testing Data** since no separate training and testing data has

been provided to us. The splitting of data is done according to the **industry standards** of **(70%-30%)**. We use the **training data** to create the **predictive model** and use this model to make predictions on **testing data.** The performance measures for these predictions will help in determining the **utility of the model.** The splitting of the dataset according to the **transport variable** can be done using the **sample()** function from **caTools** library.

```
> ### Splitting of the datasets ###
> set.seed(77)
> splt = sample.split(trans$Transport,SplitRatio = 0.7)
> m.train = subset(trans,splt == TRUE)
> m.test = subset(trans,splt == FALSE)
> dim(m.train)
[1] 292   9
> dim(m.test)
[1] 126   9
```

We can see that the dataset **trans** has been **split** into training and testing data namely **m.train** and **m.test respectively.**

a) Logistic Regression:

Logistic Regression is a type of predictive model which is done when the **Dependent variable** is a **categorical variable.** It uses a **Logistic Functions** to predict the **category** by giving a **probability** of a class **as an output**. The logistic regression model can be built using the **glm()** function with **family = "binomial"** as its argument. First we create a **Logistic regression** model using all the variables.

```
> ### Building the logistic regression model ####
> set.seed(77)
> mglm = glm(Transport~.,m.train,family = "binomial",maxit = 100)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(mglm)

Call:
glm(formula = Transport ~ ., family = "binomial", data = m.train,
    maxit = 100)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.251e-05  -2.110e-08  -2.110e-08  -2.110e-08   1.124e-05

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.426e+04  1.627e+07  -0.001    0.999
Age          4.318e+02  4.964e+05   0.001    0.999
Gender1     -5.992e+02  6.920e+05  -0.001    0.999
Engineer1   -3.657e+02  1.119e+06   0.000    1.000
MBA1        -4.743e+02  5.511e+05  -0.001    0.999
Work.Exp    -2.697e+02  3.140e+05  -0.001    0.999
Salary       6.201e+01  7.167e+04   0.001    0.999
Distance     1.631e+02  1.853e+05   0.001    0.999
license1     6.410e+02  7.399e+05   0.001    0.999

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.6591e+02  on 291  degrees of freedom
Residual deviance: 8.1864e-10  on 283  degrees of freedom
AIC: 18

Number of Fisher Scoring iterations: 34
```

As we can see from the above **logistic regression model,** none of the variables prove to be **significant** as none of them have their **p-values** less than **0.05.** This is due to the **existence of multicollinearity variables** and **highly imbalanced independent variables.**

Making Predictions:

From the **logistic regression model** that we created, we can make predictions on the **testing data** using the **predict()** function and putting the argument as **response.** Since we get probabilities from the **predict function,** we can place the **threshold** of **0.5** to make predict the class as the dataset is **highly segregated.**

```
> ## Predictions for Logistic Regression ###
> m.test.pred = predict(mglm,m.test,type = "response")
> m.test.predc = ifelse(m.test.pred > 0.5,"1","0")
> m.test.predc = as.factor(m.test.predc)
> m.test$Transport = as.factor(m.test$Transport)
> cf.lr = caret::confusionMatrix(m.test$Transport,m.test.predc,positive = "1")
```

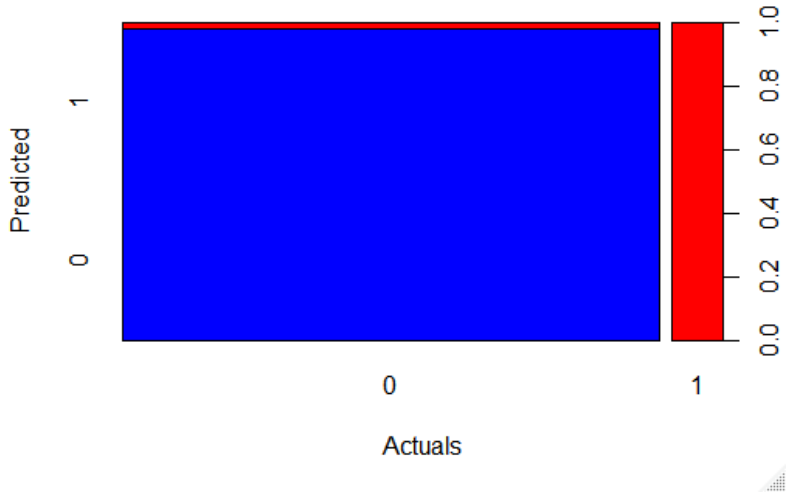
Confusion Matrix:

The **major model metric** for **classification models** is **Confusion Matrix.** It makes a **frequency table** for the **observed values against the predicted values.** The confusion matrix can be obtained from the **function confusionMatrix()** from **caret library.** The **argument positive** must be set as **"1"** since we are concerned with the **number of car users.** *The main measures we are concerned with in the confusion matrix are **high precision, high recall, high balanced accuracy** and **good overall accuracy.***

```
## Confusion Matrix ###
cf.lr = caret::confusionMatrix(m.test$Transport,m.test.predc,positive = "1")
print(cf.lr)
cf.lr$byClass
plot(m.test$Transport,m.test.predc,xlab = "Actuals",ylab = "Predicted",
     col = c("Blue","Red"))
```

```
> print(cf.lr)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   2
         1   0  11

               Accuracy : 0.9841
                 95% CI : (0.9438, 0.9981)
    No Information Rate : 0.8968
    P-Value [Acc > NIR] : 0.0001316

                  Kappa : 0.908

 Mcnemar's Test P-Value : 0.4795001

            Sensitivity : 0.8462
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9826
             Prevalence : 0.1032
         Detection Rate : 0.0873
   Detection Prevalence : 0.0873
      Balanced Accuracy : 0.9231

       'Positive' Class : 1
```

```
> cf.lr$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
          0.84615385           1.00000000           1.00000000           0.98260870
           Precision               Recall                   F1           Prevalence
          1.00000000           0.84615385           0.91666667           0.10317460
Detection Rate Detection Prevalence    Balanced Accuracy
          0.08730159           0.08730159           0.92307692
```



As we can see from the **confusion matrix,** even though we have **high overall accuracy,** our **Balanced Accuracy** is way lesser as we **have managed** to predict **all of the 0's** but **failed to predict two of the 1's,** considering the number of **1's** is only **13.** This is that's why our **Recall** rate is lesser.

b) Naïve Bayes:

The **Naïve Bayes** classifier uses an extension of **Bayes** theorem to predict **class outputs** given any number of **independent variables.** The function **naiveBayes()** can be used to build a **Naïve Bayes** model. We use all the variables to build the **Naïve Bayes** model. (*We are able to build a **naïve bayes model** in this case as we are have been able to convert the multiclass dependent variable into binary variable)*

```
> ### Building a Naive Bayes Model ####
> set.seed(77)
> m.nb = naiveBayes(m.train$Transport~.,data = m.train)
> print(m.nb)

Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
         0          1
0.91780822 0.08219178

Conditional probabilities:
   Age
Y       [,1]     [,2]
  0 26.40672 3.023422
  1 36.50000 3.283688

   Gender
Y           0         1
  0 0.3171642 0.6828358
  1 0.1666667 0.8333333

   Engineer
Y       0    1
  0 0.25 0.75
  1 0.00 1.00

   MBA
Y           0         1
  0 0.7164179 0.2835821
  1 0.6666667 0.3333333


   Work.Exp
Y         [,1]     [,2]
  0  4.735075 3.145766
  1 17.291667 3.861506

   Salary
Y        [,1]      [,2]
  0 13.04216  5.345275
  1 41.27083 10.015444

   Distance
Y        [,1]     [,2]
  0 10.81418 3.176179
  1 17.49583 2.696774

   license
Y           0         1
  0 0.8432836 0.1567164
  1 0.1250000 0.8750000
```

We can see that the **A-priori probabilities** are highly imbalanced as the **dependent variable** is **highly imbalanced**.

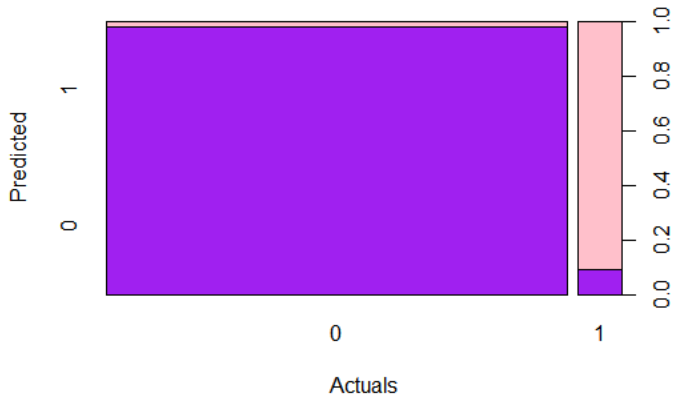
Making Predictions:

From the **Naïve Bayes model** that we created, we can make predictions on the **testing data** using the **predict()** function and putting the argument as **class.**

```
> ## Predictions for Naive Bayes ###
> set.seed(77)
> m.nb.test.pred = predict(m.nb,m.test,type = "class")
```

Confusion Matrix:

```
> ## Confusion Matrix ###
> nb.cf = caret::confusionMatrix(m.test$Transport,m.nb.test.pred,positive = "1")

> print(nb.cf)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   2
         1   1  10

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.9048
    P-Value [Acc > NIR] : 0.001606

                  Kappa : 0.8565

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.83333
            Specificity : 0.99123
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.98261
             Prevalence : 0.09524
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.91228

       'Positive' Class : 1

> nb.cf$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
           0.83333333           0.99122807           0.90909091           0.98260870
            Precision               Recall                   F1           Prevalence
           0.90909091           0.83333333           0.86956522           0.09523810
      Detection Rate Detection Prevalence    Balanced Accuracy
           0.07936508           0.08730159           0.91228070
> plot(m.test$Transport,m.nb.test.pred,xlab = "Actuals",ylab = "Predicted",
+      col = c("Purple","Pink"))
>
```



As we can see from the above results, the **Naïve Bayes** did good but **lacked** compared to **logistic regression** by a small difference.

c) KNN:

KNN(K Nearest Neighbours) algorithm as a **classifier** uses the **current data** and estimates the **class** of the new data point by using certain **similarity measures of distance.** The **KNN** model can be built using the function **knn()**. This function requires an additional argument namely **'k'.** This **K** value denotes the number of nearest neighbours to consider for determining the class of the new data point. The value of k must be in such a way that if it is too low, it will under predict that new data point and if it is too high, it may include other classes into consideration and may result in over-fitting. The exact number of **k** to be taken depends on the size of the dataset. The ideal number would be to take the **square root** of the number of **rows** in the dataset. So for our training dataset, the ideal number would be **17.**

```
> ### Building a KNN model ####
> set.seed(77)
> m.knn = knn(train = m.train,test = m.test,cl = m.train$Transport,k = 17)
```

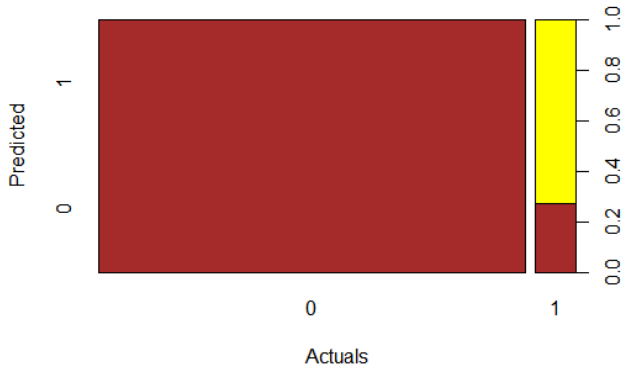
Confusion Matrix:

```
> ## Confusion Matrix ###
> knn.cf = caret::confusionMatrix(m.test$Transport,m.knn,positive = "1")
> print(knn.cf)
```

```
> print(knn.cf)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 115   0
         1   3   8

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.9365
    P-Value [Acc > NIR] : 0.03787

                  Kappa : 0.8296

 Mcnemar's Test P-Value : 0.24821

            Sensitivity : 1.00000
            Specificity : 0.97458
         Pos Pred Value : 0.72727
         Neg Pred Value : 1.00000
             Prevalence : 0.06349
         Detection Rate : 0.06349
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.98729

       'Positive' Class : 1
```

```
> knn.cf$byClass
        Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
         1.00000000            0.97457627            0.72727273            1.00000000
          Precision                Recall                    F1            Prevalence
         0.72727273           1.00000000            0.84210526            0.06349206
     Detection Rate Detection Prevalence     Balanced Accuracy
         0.06349206            0.08730159           0.98728814
```

```
> plot(m.test$Transport,m.knn,xlab = "Actuals",ylab = "Predicted",
+      col = c("White","Yellow"))
```



As we can see from the above results, the **accuracy, recall and balanced accuracy** are very good, we were not able get good **precision** when compared to **logistic regression.**

*As we can see from the above results, we are unable to get good metrics for the models and each of these models seem to trade-off one metric for another. The main reason for this can be **highly imbalanced data.*** The problem of **highly imbalanced data** can be dealt with **SMOTE** technique.

d)SMOTE:

SMOTE (Synthetic Minority Over-Sampling Technique) is an **over-sampling technique** which can be used on **imbalanced datasets.** The basic idea behind **SMOTE** is that it uses the variables present in the dataset to create **more synthetic observations** for the **minority class** instead of making **duplicates** of the same observations so that it comes on par with the **majority class.** The **SMOTE** technique can be done using the **SMOTE()** function. The **arguments** like **perc.over(Decides the number of Minority class to add)** and **perc.under(Decides the number of Majority class to replace)**. The arguments **perc.over** and **perc.under** must be determined based on **how much split** we want **between** both the classes. The function of **SMOTE** will give out dataset **balanced** as per our requirements. The **split percent** we decide for the **SMOTE** function is **50-50%.** We split only the **training data(m.train)** and save the SMOTE data as **s.train.**

Calculations for Perc.Over and Perc.Under:

Perc.Over:

*Let the Perc.Over = x*100*

Since we want the dataset(m.train) to be split into 50-50%, the formula for x will be as follows:

$$x = \frac{146 - 24}{24} = 5.08$$

Therefore, Perc.Over = 508

Perc.Under:

*Let the Perc.under = y*100*

Since we want the dataset(m.train) to be split into 50-50%, the formula for y will be as follows:

$$y = \frac{146}{24 * x} = \frac{146}{24 * 5} = 1.21$$

Therefore Perc.Under = 121

```
> ### Doing a SMOTE analysis for a better dataset ####
> set.seed(77)
> s.train =SMOTE(Transport~.,m.train,perc.over = 501, perc.under = 121)
> s.test = m.test
> prop.table(table(s.train$Transport))

        0         1
0.5017301 0.4982699
> dim(s.train)
[1] 289   9
```

We can see that we were able to **increase** the frequency of **1's** to almost **50%.**

Now that we have a **balanced dataset**, we can now go ahead to create models using this **smote training data.**

e) Logistic Regression after SMOTE:

We now create a **logistic regression model** using the **smote train data (s.train).**

```
> ### Building a Logistic Regression Model after SMOTE ####
> set.seed(77)
> sglm = glm(Transport~.,data = s.train,family = "binomial",maxit = 100)
> summary(sglm)

Call:
glm(formula = Transport ~ ., family = "binomial", data = s.train,
    maxit = 100)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.91407   0.06047   0.16601   0.37641   1.43759

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.51604    2.90527  -2.587  0.00968 **
Age          0.35518    0.13587   2.614  0.00894 **
Gender1      1.29220    0.52216   2.475  0.01333 *
Engineer     0.64652    0.56109   1.152  0.24922
MBA          0.58586    0.68391   0.857  0.39165
Work.Exp    -0.21172    0.19614  -1.079  0.28039
Salary       0.15810    0.09286   1.703  0.08866 .
Distance    -0.17859    0.09009  -1.982  0.04745 *
license     -1.74347    0.64603  -2.699  0.00696 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 179.41  on 288  degrees of freedom
Residual deviance: 118.80  on 280  degrees of freedom
AIC: 136.8

Number of Fisher Scoring iterations: 7
```

We can see that the variables **Age, Gender, salary, distance and license** have become significant **compared to the** previous **logistic regression model.**

Predictions:

We need to use this model to make **predicitons** on the **s.test** testing data.

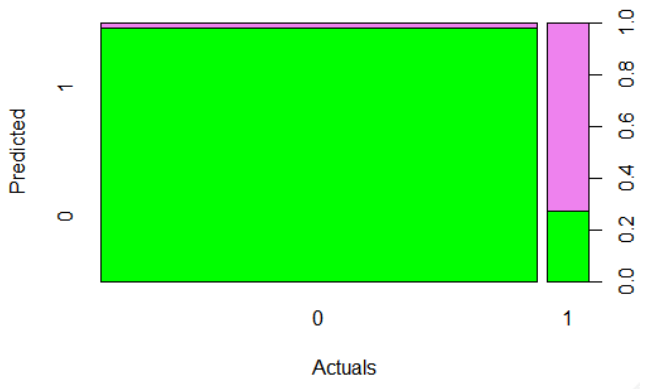
```
> ## Predictions for Logistic Regression after SMOTE ###
> set.seed(77)
> s.test.pred = predict(sglm,s.test,type = "response")
> s.test.predc = ifelse(s.test.pred > 0.5,"1","0")
> s.test.predc = as.factor(s.test.predc)
> s.test$Transport = as.factor(s.test$Transport)
```

Confusion Matrix:

```
## Confusion Matrix ###
glm.c = caret::confusionMatrix(s.test$Transport,s.test.predc,positive = "1")
print(glm.c)
glm.c$byClass
plot(s.test$Transport,s.test.predc,xlab = "Actuals",ylab = "Predicted",
     col = c("Green","Violet"))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   2
         1   3   8

               Accuracy : 0.9603
                 95% CI : (0.9098, 0.987)
    No Information Rate : 0.9206
    P-Value [Acc > NIR] : 0.05955

                  Kappa : 0.7403

 Mcnemar's Test P-Value : 1.00000

            Sensitivity : 0.80000
            Specificity : 0.97414
         Pos Pred Value : 0.72727
         Neg Pred Value : 0.98261
             Prevalence : 0.07937
         Detection Rate : 0.06349
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.88707

       'Positive' Class : 1
```

```
> glm.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
          0.80000000           0.97413793           0.72727273           0.98260870
           Precision               Recall                   F1           Prevalence
          0.72727273           0.80000000           0.76190476           0.07936508
       Detection Rate Detection Prevalence    Balanced Accuracy
          0.06349206           0.08730159           0.88706897
```



We can see that this model has performed **poorly** when compared to the **logistic regression** before **smote.**

f) Naïve Bayes after SMOTE:

We now create a **Naïve Bayes model** using the **smote train data (s.train).**

```
### Building a Naive Bayes Model after SMOTE####
set.seed(77)
s.nb = naiveBayes(s.train$Transport~.,data = s.train)
print(s.nb)
```

```
Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
        0         1
0.5017301 0.4982699

Conditional probabilities:
   Age
Y       [,1]     [,2]
  0 26.52414 3.062050
  1 36.13970 2.816203

   Gender
Y           0         1
  0 0.2965517 0.7034483
  1 0.1527778 0.8472222

   Engineer
Y           0         1
  0 0.2206897 0.7793103
  1 0.0000000 1.0000000
```

```
   MBA
Y           0         1
  0 0.6827586 0.3172414
  1 0.6250000 0.3750000

   Work.Exp
Y        [,1]     [,2]
  0  4.737931 3.327121
  1 16.886585 3.578042

   Salary
Y       [,1]     [,2]
  0 12.68690 5.034647
  1 39.96644 9.460266

   Distance
Y       [,1]     [,2]
  0 10.50966 3.190727
  1 15.95896 1.919303

   license
Y           0          1
  0 0.87586207 0.12413793
  1 0.07638889 0.92361111
```

Predictions:

```
> ## Predictions for Naive Bayes after SMOTE ###
> s.nb.test.pred = predict(s.nb,s.nb.test,type = "class")
```

Confusion Matrix:

```
## Confusion Matrix ###+
nb.c = caret::confusionMatrix(s.test$Transport,s.nb.test.pred,positive = "1")
print(nb.c)
nb.c$byClass
plot(s.test$Transport,s.nb.test.pred,xlab = "Actuals",ylab = "Predicted",
     col = c("Black","Orange"))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 112   3
         1   1  10

               Accuracy : 0.9683
                 95% CI : (0.9207, 0.9913)
    No Information Rate : 0.8968
    P-Value [Acc > NIR] : 0.002604

                  Kappa : 0.8159

 Mcnemar's Test P-Value : 0.617075

            Sensitivity : 0.76923
            Specificity : 0.99115
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.97391
             Prevalence : 0.10317
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.88019

       'Positive' Class : 1
```

```
> nb.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
           0.76923077           0.99115044           0.90909091           0.97391304
            Precision               Recall                   F1           Prevalence
           0.90909091           0.76923077           0.83333333           0.10317460
      Detection Rate Detection Prevalence    Balanced Accuracy
           0.07936508           0.08730159           0.88019061
```

We can see that in every sense, this **Naïve Bayes model** has performed badly compared to the **Naïve Bayes model** before **SMOTE.**

g) KNN after SMOTE:

We now create a **KNN model** using the **smote train data (s.train).**

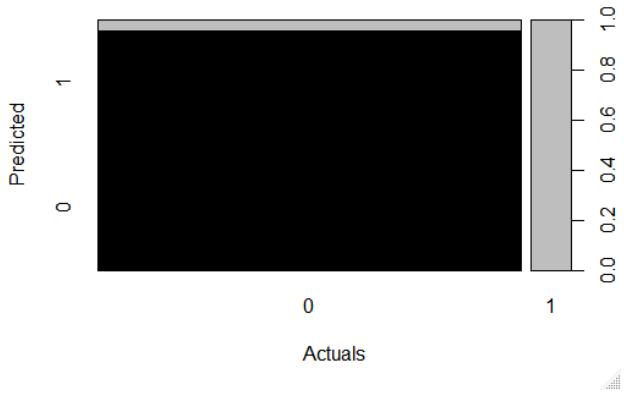
```
> ### Building a KNN model ####
> set.seed(77)
> s.knn = knn(train = s.train,test = s.test,cl = s.train$Transport,k = 17)
> |
```

Predictions:

```
### Confusion Matrix ##
knn.c = caret::confusionMatrix(s.test$Transport,s.knn,positive = "1")
print(knn.c)
knn.c$byClass
plot(s.test$Transport,s.knn,xlab = "Actuals",ylab = "Predicted",
     col = c("Black","Gray"))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 110   5
         1   0  11

               Accuracy : 0.9603
                 95% CI : (0.9098, 0.987)
    No Information Rate : 0.873
    P-Value [Acc > NIR] : 0.0007991

                  Kappa : 0.7934

 Mcnemar's Test P-Value : 0.0736383

            Sensitivity : 0.6875
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9565
             Prevalence : 0.1270
         Detection Rate : 0.0873
   Detection Prevalence : 0.0873
      Balanced Accuracy : 0.8438

       'Positive' Class : 1
```

Sensitivity	Specificity	Pos Pred Value	Neg Pred Value
0.68750000	1.00000000	1.00000000	0.95652174
Precision	**Recall**	**F1**	**Prevalence**
1.00000000	0.68750000	0.81481481	0.12698413
Detection Rate	**Detection Prevalence**	**Balanced Accuracy**	
0.08730159	0.08730159	0.84375000	



As we can see from the above results, this **KNN model** performed **poorly** compared to the **original KNN model.**

Model	Precision	Recall	Accuracy	Balanced Accuracy
Logistic Regression	1	0.84	0.98	0.92
Naïve Bayes	0.9	0.83	0.97	0.91
KNN	0.72	1	0.97	0.98
Logistic Regression(SMOTE)	0.72	0.8	0.96	0.88
Naïve Bayes(SMOTE)	0.99	0.76	0.96	0.88
KNN(SMOTE)	1	0.68	0.96	0.84

From the above models, we can see that models created after SMOTE did not improve in any way over the models before SMOTE. This can be due to the fact that ***the samples created for the minority class are synthetic*** *and the fact that* ***the independent variables are also imbalanced.*** *These factors affect the different metrics of the confusion matrix.*

From the above six models, the best model with emphasis on ***overall measures of precision, recall, accuracy and balanced accuracy,*** *it is* ***Logistic Regression before SMOTE.***

Ensemble Methods:

Even though good enough, we can try to build better models by using several **ensemble methods** known as **bagging and boosting** on the **SMOTE data.** The **ensemble methods** use several **classifiers**(Known as **weak learners)** and utilizes all of them to learn the errors they committed and improve upon them. The most popular method for **classification problems** are **Bagging and Boosting.** Now we try to use **bagging and boosting models** on the **SMOTE** dataset and use them to make predictions on the testing data. Then we use it to compare it with the best model we got, which is **logistic regression.**

h)Bagging:

Bagging or Bootstrap Aggregating is an Ensemble learning method where the **classifiers** use random **subsests** of the original **dataset** to make predictions and then give out the **aggregated** results of the **classifiers.** The **aggregating** can be done either by **voting or averaging.** The process of **bagging** can be done on the by using the function **bagging()** on **s.train.** The **bagging function** has two types of arguments, ones related to the outputs of the **bagging model** and the others related to the **rpart** function of the **bagging model.** The major argument of the **bagging function** is the **coob** which gives out the **out of bag errors.** Then under **rpart.control,** we have **minsplit and maxdepth. Minsplit** determines the **minimum number of** observations should be present in each node

before splitting and **maxdepth** determines the length of the **decision tree.**

```
### BAGGING AFTER SMOTE ####
set.seed(77)
bag.model = bagging(Transport~.,data = s.train,coob = TRUE,
                    control = rpart.control(maxdepth = 10,minsplit = 3))
summary(bag.model)
```

```
> summary(bag.model)

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = Transport ~ ., data = s.train, coob = TRUE,
    control = rpart.control(maxdepth = 10, minsplit = 3))

Out-of-bag estimate of misclassification error:  0.0138

$y
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [56] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[111] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[166] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[221] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[276] 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Levels: 0 1
```

We can see that the **Out of bag** error is **0.0135**.

Predictions:

The predictions can be made using the **predict()** function with argument as **class.**

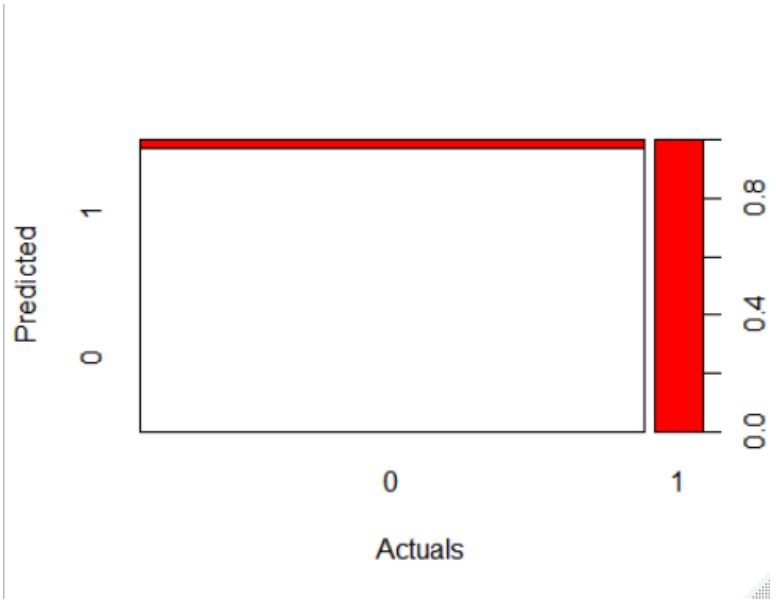
```
## Making predictions with Bagging model ####
set.seed(77)
s.bag.pred = predict(bag.model,s.test,type = "class")
bag.pred = as.factor(s.bag.pred)
```

Confusion Matrix:

```
## Confusion Matrix
bag.c = caret::confusionMatrix(s.test$Transport,bag.pred,positive = "1")
print(bag.c)
bag.c$byClass
plot(s.test$Transport,bag.pred,xlab = "Actuals",ylab = "Predicted",
     col = c("White","Red"))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 112   3
         1   0  11

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.8889
    P-Value [Acc > NIR] : 0.0002782

                  Kappa : 0.867

 Mcnemar's Test P-Value : 0.2482131

            Sensitivity : 0.7857
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9739
             Prevalence : 0.1111
         Detection Rate : 0.0873
   Detection Prevalence : 0.0873
      Balanced Accuracy : 0.8929

       'Positive' Class : 1
```

Sensitivity	Specificity	Pos Pred Value
0.78571429	1.00000000	1.00000000
Neg Pred Value	**Precision**	**Recall**
0.97391304	1.00000000	0.78571429
F1	**Prevalence**	**Detection Rate**
0.88000000	0.11111111	0.08730159
Detection Prevalence	**Balanced Accuracy**	
0.08730159	0.89285714	



The above results state that even though the results are good, they are not as good as **the best model i.e. logistic regression.**

i) Boosting:

Boosting refers to the **Machine Learning** technique where base **machine learning algorithm** is used to create **weak**

learners and all the **weak learners** are utilized to create a **strong rule** that will be used for making predictions.

The two of the most popular types of **boosting algorithms** are:

A. **Adaboost (Adaptive Boosting)**

B. **XG Boost (Extreme Gradient Boosting)**

We will be using both the **boosting algorithms** on the **s.train** model to create models and use the model to make **predictions** on **s.test.**

A. Adaboost (Adaptive Boosting):

Adaptive Boosting or Adaboost is a **boosting algorithm** where the **iteration process** of building a **strong rule** is dependent on the **weightage** given to the **errors** created by the **preceding weak learner.** The **Adaboost** can be performed in **R** with the help of **gbm()** function. The arguments that can be tweaked for the function are **n.trees, distribution, shrinkage, cv.folds and n.cores.**

n.trees – The number of decision trees to build for consideration

distribution – The distribution of the **target variable.**

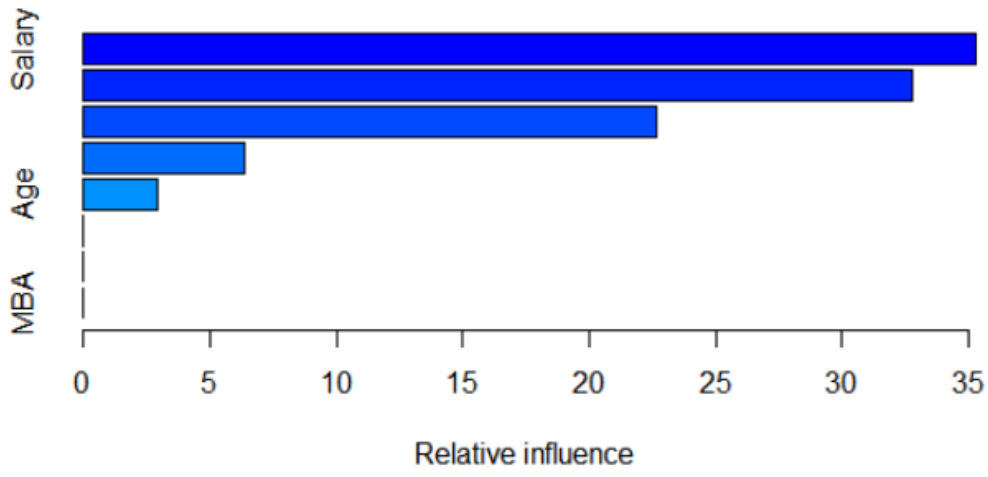
Shrinkage – The learning rate for building the **decision trees.**

cv.folds – The cross validations to perform

n.cores – The number of **cores** to be used for the **algorithm.**

```
### Boosting after SMOTE ####
### Ada boost ####
s.test$Transport = as.numeric(s.test$Transport)
str(s.train)
s.test$Transport = as.factor(s.test$Transport)
s.train$Transport = as.factor(s.train$Transport)
set.seed(77)
s.gbm <- gbm(
        formula = Transport~.,
        data = s.train,
        distribution = "multinomial",
        n.trees = 100,
        interaction.depth = 1,
        shrinkage = 0.1,
        cv.folds = 2,
        n.cores = 2, # will use all cores by default
        verbose = FALSE
)
summary(s.gbm)
```

```
                var      rel.inf
Salary       Salary 35.298959403
Work.Exp   Work.Exp 32.745988200
Distance   Distance 22.645360426
license     license  6.345950201
Age             Age  2.956551766
Engineer   Engineer  0.007190004
Gender       Gender  0.000000000
MBA             MBA  0.000000000
```



As we can see from the above results, **Salary, Work.Exp and Distance** prove to be the major **predictors** for the model.

Predictions:

The predictions can be made using **predict()** function with argument **response.**

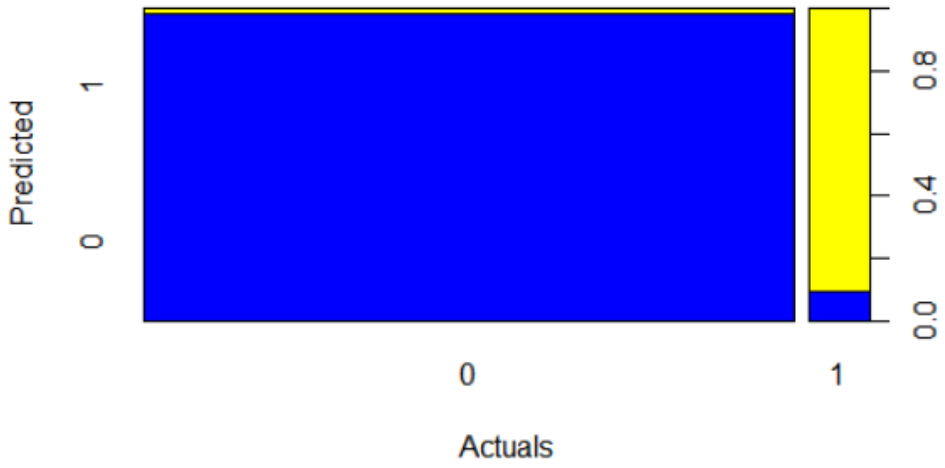
```
## Predictions for Adaboost model ###
set.seed(77)
prob.s.gbm = predict(s.gbm,s.test,type = "response")
prob.s.gbm = as.data.frame(prob.s.gbm)
pred.s.gbm = ifelse(prob.s.gbm[,c(1)] > 0.5,"0","1")
pred.s.gbm = as.factor(pred.s.gbm)
s.test = m.test
```

Confusion Matrix

```
## Confusion Matrix ###
adb.c = caret::confusionMatrix(s.test$Transport,pred.s.gbm,positive = "1")
print(adb.c)
adb.c$byClass
plot(s.test$Transport,pred.s.gbm,xlab = "Actuals",ylab = "Predicted",
     col = c("Blue","Yellow"))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   2
         1   1  10

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.9048
    P-Value [Acc > NIR] : 0.001606

                  Kappa : 0.8565

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.83333
            Specificity : 0.99123
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.98261
             Prevalence : 0.09524
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.91228

       'Positive' Class : 1
```

Sensitivity	Specificity	Pos Pred Value
0.83333333	0.99122807	0.90909091
Neg Pred Value	**Precision**	**Recall**
0.98260870	0.90909091	0.83333333
F1	**Prevalence**	**Detection Rate**
0.86956522	0.09523810	0.07936508
Detection Prevalence	**Balanced Accuracy**	
0.08730159	0.91228070	



We can see that the performance of this model, even though **good**, did not perform as good as our best model which is **logistic regression.**

B. XG Boost (Extreme Gradient Boosting):

The **xgboost** model tries to make a **strong rule** by trying reduce the **error** obtained from the **loss function** from the **weak learners.** The **xgboost** can be performed by using the **xgboost()** function. Since the **xgboost** function only takes matrices in the **arguments,** we need to convert the dataframes into **matrices** using the function **model.matrix()**.

The arguments in the **xgboost()** function are as follows:

eta – The learning rate for tree building.

max depth – **Depth** of the **decision tree.**

Nrounds – Number of **iterations** to be performed

n.fold – Number of **cross-validations** to be performed.

early_stopping_rounds – The number at which **tree building** should stop if there was **no improvent** in **error.**

verbose – Whether to show the **tree building** process or not.

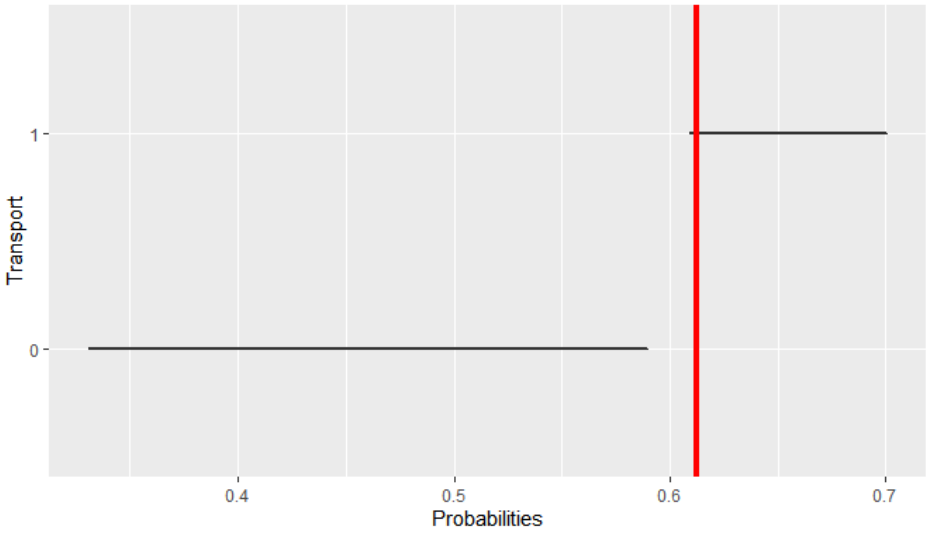
```
### XG BOOSTING ####
set.seed(77)
f = model.matrix(s.train$Transport~.,s.train)

t = s.train[,9]
t = as.matrix(t)
t = as.character(t)
t = as.numeric(t)
xg.train.m = xgb.DMatrix(data = f,label = t)

e = model.matrix(s.test$Transport~.,s.test)
s = s.test[,9]
s = as.matrix(s)
s = as.character(s)
s = as.numeric(s)
xg.test.m = xgb.DMatrix(data = e,label = s)
set.seed(77)
s.xgb <- xgboost(
  data = xg.train.m,
  eta = 0.5,
  max_depth = 5,
  nrounds = 2,
  nfold = 2,
  objective = "binary:logistic",  # for regression models
  verbose = 0,                # silent,
  early_stopping_rounds = 10 # stop if no improvement for 10 consecutive trees
)
print(s.xgb)
```

```
> print(s.xgb)
##### xgb.Booster
raw: 1.5 Kb
call:
  xgb.train(params = params, data = dtrain, nrounds = nrounds,
    watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
    early_stopping_rounds = early_stopping_rounds, maximize = maximize,
    save_period = save_period, save_name = save_name, xgb_model = xgb_model,
    callbacks = callbacks, eta = 0.5, max_depth = 5, nfold = 2,
    objective = "binary:logistic")
params (as set within xgb.train):
  eta = "0.5", max_depth = "5", nfold = "2", objective = "binary:logistic", silent = "1"
xgb.attributes:
  best_iteration, best_msg, best_ntreelimit, best_score, niter
callbacks:
  cb.evaluation.log()
  cb.early.stop(stopping_rounds = early_stopping_rounds, maximize = maximize,
    verbose = verbose)
# of features: 9
niter: 2
best_iteration : 2
best_ntreelimit : 2
best_score : 0
nfeatures : 9
evaluation_log:
 iter train_error
    1     0.00692
    2     0.00000
```

Predictions:

The predictions can be made using the **predict()** function. The **predict()** function gives us **probabilities** which must be converted to **class predictions** using **a certain threshold.** This **threshold** must be decided in such a way that we must be **more** inclined towards **predicting more** number of **1's** than **0's.**

```
## Predictions for XGboost ###
xgb.predict = predict(s.xgb,xg.test.m)
bp = qplot(xgb.predict,s.test$Transport,geom = "boxplot"
      ,xlab = "Probabilities",ylab = "Transport",xintercept = 0.63)
bp+geom_vline(xintercept = 0.6126,color = "Red",size = 1.5)
xgb.predict.c = ifelse(xgb.predict > 0.6126,"1","0")
xgb.predict.c = as.factor(xgb.predict.c)
```



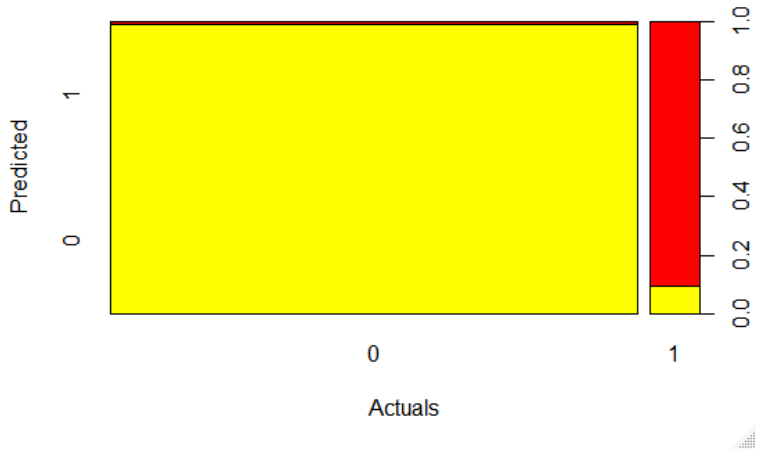
We see that when we set the **threshold** at exactly **0.6126**, we are able to predict **higher number of 1's.**

Confusion Matrix:

```
## Confusion Matrix ###
xgb.c = caret::confusionMatrix(s.test$Transport,xgb.predict.c,positive = "1")
print(xgb.c)
xgb.c$byClass
plot(s.test$Transport,xgb.predict.c,xlab = "Actuals",ylab = "Predicted",
     col = c("Yellow","Red"))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 114   1
         1   1  10

               Accuracy : 0.9841
                 95% CI : (0.9438, 0.9981)
    No Information Rate : 0.9127
    P-Value [Acc > NIR] : 0.0008535

                  Kappa : 0.9004

 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.90909
            Specificity : 0.99130
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.99130
             Prevalence : 0.08730
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.95020

       'Positive' Class : 1
```

Sensitivity	Specificity	Pos Pred Value
0.90909091	0.99130435	0.90909091
Neg Pred Value	**Precision**	**Recall**
0.99130435	0.90909091	0.90909091
F1	**Prevalence**	**Detection Rate**
0.90909091	0.08730159	0.07936508
Detection Prevalence	**Balanced Accuracy**	
0.08730159	0.95019763	



We can see that this model has performed **slightly better** than **the best model** which is **Logistic Regression.**

j) Model Comparison:

Model	Precision	Recall	Accuracy	Balanced Accuracy
Logistic Regression	***1***	***0.84***	***0.98***	***0.92***
Naïve Bayes	0.9	0.83	0.97	0.91
KNN	0.72	1	0.97	0.98
Logistic Regression(SMOTE)	0.72	0.8	0.96	0.88
Naïve Bayes(SMOTE)	0.99	0.76	0.96	0.88
KNN(SMOTE)	1	0.68	0.96	0.84
Bagging(SMOTE)	1	0.78	0.97	0.89
AdaBoost(SMOTE)	0.9	0.83	0.97	0.91
XGBoost(SMOTE)	**0.9**	**0.9**	**0.98**	**0.95**

Remarks:

From the above metrics, we can say that **the best model taking the best of all the metrics into consideration** is **XGBoost after SMOTE**. Even though **xgboost** couldn't perform better than the **logistic regression** in terms of **precision**, it performed **way better** in the terms of **recall** and **slightly better** in terms of **balanced accuracy.**

Since the management is concerned with the **predicting** the number of **car users,** we are **inclined** towards having **better Recall** and **better balanced accuracy** over **precision** and **overall accuracy.** It must be noted that our model may have **more number of False Positives** but should try to **decrease the number of False Negatives** as we might lose **various car users** due to wrong predictions. Terming **someone who is not a car user as a car user** will not affect the management's decision as much as predicting a **car user** as a **non-car user.**

Considering the above parameters, we present the Xgboost model created on SMOTE data as the best model to be used for predictions by the Management.

The following **obstacles** have been placed by the **dataset** while making **classification models:**

- The dataset was **highly imbalanced** with respect to the **target variable** where only **8%** were car users and **92%** non-car users.
- Adding upon the imbalance is the further problem that the class which we needed to consider as positive was the **minority class** (Car Users).
- The whole dataset had many **categorical independent** variables which were very **imbalanced** much like the target variable.
- Only the **numerical variables** and **License** variable contributed much to the **model building** while the rest of the variable **didn't** seem to be **significant** in the same.
- All the **numerical variables** had very **high correlation** values between them making them **insignificant** and making other variables **insignificant** especially during the **building of logistic regression.**

4) Project Conclusion:

The project **Car Usage Analysis** was started with the objective of making a **good classification model** so that they can **predict** given an employee's details. We used the dataset(cars.csv) given to us to make **three classification models, Logistic Regression, Naïve Bayes and KNN.** Since the results weren't up to the mark due to the imbalance in the dataset, we applied **SMOTE** technique to balance the dataset and built all the above **classification models** using the **SMOTE** data. But there was **no improvement** in the model measures from **before SMOTE.** So therefore, we had to go for

ensemble methods like **bagging and boosting. Using** the **SMOTE** training data, we created **3 models, bagging model, Adaptive Boosting model** and **Extreme Boosting model.** Out of these **models, *Extreme boosting model with SMOTE*** as our **best model** which we thought was better than **previous best model** which was **Logistic Regression before SMOTE.**

The suggestions that can be given to the management are as follows:

- The sample data contains many **outliers** and **few missing values.** It must be made sure that the data, if possible, be without any **outliers** or at least **without** any **missing values.**
- The sample data could have contained more **important variables** such as **desire to buy a car, financial capability to afford a car, presence of a car garage, etc.** to make the predictions better**.**
- The sample data if possible, should be made in such a way that the data is not **highly imbalanced.**
- The predictor variables must be selected in such a way there are **not related** to one another, **like Age and Work.Exp** in our case.
- The **dataset** could have been **bigger** as the **model metrics, even though good, sometimes highly overstate** the **results** due to smaller data.
- The **management** must keep in mind that the best model suggested keeping in mind that it wants to **predict** if given an employee, uses a car or not. It was kept in mind that an employee **wrongly identified as car user** costs less than the

ones which we wrongly identified as non-car user. If the management has other objectives, then the best model must be chosen accordingly.

The **actionable insights** that can be taken by the **management** are as follows:

- The company, has **majority** of its employees using **public transport** which is **eco-friendly.** It would be better if even the others were told the **benefits** of the **public transport** so that even goes **eco-friendly.**
- It is seen that **the employees** who stay **far away** from the **company and are older in age** are **forced** to **use car.** If they are provided **accommodation** in the **proximity** of the **company,** even they might **opt public transport.**
- If the company provided **pool car transport** and gave accommodation **for** the **female employees** in the proximity**,** then most of them would **not prefer to commute by car** and hence **staying eco-friendly.**
- If the employee does have to **stay away** from the company and is **older in age,** the company can conduct **driving lessons** which might help them **commute safely** to the **company,** and hence **ensuring** the **safety** of the **employees.**
- It was seen that there were **very less** number of **employees** who have done their **MBA.** It must be noted that even though the company is more **prone** to the **technical side**, presence of **MBA graduates** or **Management employees**

helps company take the **right decisions at the right time** for the **smooth functioning of the company.**

5) Appendix – A (Source Code):

EMPLOYEE CAR USAGE ANALYSIS

```
################## EMPLOYEE CAR USAGE ANALYSIS ####################

 ### Setting up the working directory ####
 setwd("C:/R programs great lakes/ML/project")
 getwd()
[1] "C:/R programs great lakes/ML/project"

### Invoking the necessary libraries ####
install.packages("readr")
library(readr)
install.packages("Hmisc")
library(Hmisc)
install.packages("ggplot2")
library(ggplot2)
install.packages("psych")
library(psych)
install.packages("caret")
library(caret)
install.packages("class")
library(class)
install.packages("caTools")
library(caTools)
install.packages("e1071")
library(e1071)
install.packages("DMwR")
library(DMwR)
install.packages("xgboost")
library(xgboost)
install.packages("gbm")
library(gbm)
install.packages("Matrix")
library(Matrix)
install.packages("ipred")
library(ipred)
install.packages("rpart")
library(rpart)
> trans = read.csv("Cars.csv",header = TRUE)
> ### Identification of different variables ####
> View(trans)
> dim(trans)
[1] 418   9
```

```
> str(trans)
'data.frame': 418 obs. of  9 variables:
 $ Age      : int  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 2 2 2
2 ...
 $ Engineer : int  1 1 1 0 0 0 1 0 1 1 ...
 $ MBA      : int  0 0 0 0 0 0 1 0 0 0 ...
 $ Work.Exp : int  5 6 9 1 3 3 3 0 4 6 ...
 $ Salary   : num  14.4 10.6 15.5 7.6 9.6 9.5 11.7 6.5 8.5 13.7 ...
 $ Distance : num  5.1 6.1 6.1 6.3 6.7 7.1 7.2 7.3 7.5 7.5 ...
 $ license  : int  0 0 0 0 0 0 0 0 0 1 ...
 $ Transport: Factor w/ 3 levels "2Wheeler","Car",..: 1 1 1 1 1 1 1
1 1 1 ...
> head(trans)
  Age Gender Engineer MBA Work.Exp Salary Distance license Transport
1  28   Male        1   0        5   14.4      5.1       0  2Wheeler
2  24   Male        1   0        6   10.6      6.1       0  2Wheeler
3  27 Female        1   0        9   15.5      6.1       0  2Wheeler
4  25   Male        0   0        1    7.6      6.3       0  2Wheeler
5  25 Female        0   0        3    9.6      6.7       0  2Wheeler
6  21   Male        0   0        3    9.5      7.1       0  2Wheeler
> tail(trans)
    Age Gender Engineer MBA Work.Exp Salary Distance license
Transport
413  29 Female        1   0        6   14.9     17.0       0 Public
Transport
414  29   Male        1   1        8   13.9     17.1       0 Public
Transport
415  25   Male        1   0        3    9.9     17.2       0 Public
Transport
416  27 Female        0   0        4   13.9     17.3       0 Public
Transport
417  26   Male        1   1        2    9.9     17.7       0 Public
Transport
418  23   Male        0   0        3    9.9     17.9       0 Public
Transport
> summary(trans)
      Age           Gender       Engineer           MBA             W
ork.Exp
 Min.   :18.00   Female:121   Min.   :0.0000   Min.   :0.0000   Min.
: 0.000
 1st Qu.:25.00   Male  :297   1st Qu.:0.2500   1st Qu.:0.0000   1st
Qu.: 3.000
 Median :27.00                Median :1.0000   Median :0.0000   Medi
an : 5.000
 Mean   :27.33                Mean   :0.7488   Mean   :0.2614   Mean
: 5.873
 3rd Qu.:29.00                3rd Qu.:1.0000   3rd Qu.:1.0000   3rd
Qu.: 8.000
 Max.   :43.00                Max.   :1.0000   Max.   :1.0000   Max.
:24.000
                                               NA's   :1
     Salary          Distance        license                   Transp
ort
 Min.   : 6.500   Min.   : 3.20   Min.   :0.0000   2Wheeler        :
83
```

```
 1st Qu.: 9.625   1st Qu.: 8.60   1st Qu.:0.0000   Car             :
35
 Median :13.000   Median :10.90   Median :0.0000   Public Transport:
300
 Mean   :15.418   Mean   :11.29   Mean   :0.2033
 3rd Qu.:14.900   3rd Qu.:13.57   3rd Qu.:0.0000
 Max.   :57.000   Max.   :23.40   Max.   :1.0000
> ### Conversion of cateogrical variable ###
> trans$Gender = as.factor(trans$Gender)
> trans$Gender = ifelse(trans$Gender == "Male","1","0")
> trans$Gender = as.factor(trans$Gender)
> summary(trans$Gender)
  0   1
121 297
> trans$license = as.factor(trans$license)
> trans$Transport = ifelse(trans$Transport == "Car","1","0")
> class(trans$Transport)
[1] "character"
> trans$Transport = as.factor(trans$Transport)
> trans$Engineer = as.factor(trans$Engineer)
> trans$MBA = as.factor(trans$MBA)
> ### Conversion of cateogrical variable ###
> trans$Gender = as.factor(trans$Gender)
> trans$Gender = ifelse(trans$Gender == "Male","1","0")
> trans$Gender = as.factor(trans$Gender)
> summary(trans$Gender)
  0   1
121 297
> trans$license = as.factor(trans$license)
> trans$Transport = ifelse(trans$Transport == "Car","1","0")
> class(trans$Transport)
[1] "character"
> trans$Transport = as.factor(trans$Transport)
> trans$Engineer = as.factor(trans$Engineer)
> trans$MBA = as.factor(trans$MBA)
> ### Uni-Variate Analysis ####
> ## Analysis of independent numerical variable ##
> cor = trans[,-c(2,3,4,8,9)]
> hist.data.frame(cor,freq = TRUE)
> table(trans[,c(2,3,4,8,9)])
, , MBA = 0, license = 0, Transport = 0

      Engineer
Gender   0   1
     0  25  65
     1  38 113

, , MBA = 1, license = 0, Transport = 0

      Engineer
Gender   0   1
     0   6  14
     1  14  51

, , MBA = 0, license = 1, Transport = 0
```

```
      Engineer
Gender   0   1
     0   0   2
     1  14  25

, , MBA = 1, license = 1, Transport = 0

      Engineer
Gender   0   1
     0   0   2
     1   2  11

, , MBA = 0, license = 0, Transport = 1

      Engineer
Gender   0   1
     0   1   2
     1   0   2

, , MBA = 1, license = 0, Transport = 1

      Engineer
Gender   0   1
     0   0   0
     1   0   1

, , MBA = 0, license = 1, Transport = 1

      Engineer
Gender   0   1
     0   0   2
     1   4  15

, , MBA = 1, license = 1, Transport = 1

      Engineer
Gender   0   1
     0   0   1
     1   0   7

> ## Analysis of independent categorical variable ##
> # Engineer #
> qplot(trans$Engineer,fill = trans$Engineer,main = "Engineer")
> prop.table(table(trans$Engineer))

        0         1
0.2511962 0.7488038
> # Gender #
> qplot(trans$Gender,fill = trans$Gender,main = "Gender")
> prop.table(table(trans$Gender))

        0         1
0.2894737 0.7105263
> # MBA #
> qplot(trans$MBA,fill = trans$MBA,main = "MBA")
> prop.table(table(trans$MBA))
```

```
        0         1
0.7386091 0.2613909
> # License #
> qplot(trans$license,fill = trans$license, main = "license")
> prop.table(table(trans$license))

        0         1
0.7966507 0.2033493
> ## Analysis of dependent categorical variable
> # Transport #
> qplot(trans$Transport,fill = trans$Transport, main = "Transport")
> table(trans$Transport)

  0   1
383  35
> prop.table(table(trans$Transport))

         0          1
0.91626794 0.08373206
> ### Bi-Variate Analysis ####
> ## Analysis of dependent variable with numerical variables ##
> qplot(Distance,fill = Transport,data = trans,
+       main = "Transport and Distance")
> qplot(Age,fill = Transport,data = trans,
+       main = "Transport and Age",geom = "bar")
> qplot(Salary,fill = Transport,data = trans,
+       main = "Transport and Salary",
+       geom = "density")
> qplot(Work.Exp,fill = Transport,data = trans,
+       main = "Transport and Work.Exp",
+       geom = "dotplot")

> ## Analysis of dependent variable with categorical variables ##
> qplot(trans$Transport,fill = trans$Gender,xlab = "Transport",
+       ylab = "Gender",main = "Gender vs. Transport")
> qplot(trans$Transport,fill = trans$MBA,xlab = "Transport",
+       ylab = "MBA",main = "Transport vs. MBA")
> qplot(trans$Transport,fill = trans$license,xlab = "Transport",
+       ylab = "license",main = "Transport vs. license")
 > ## Analysis of independent variables with independent variables ##
 > qplot(Age,Salary, fill = Gender,data = trans,
 +       geom = "bin2d",main = "Age vs. Salary")
 > qplot(Salary,fill = MBA,data = trans,geom = "density",
 +       main = "Salary vs. MBA")
 > qplot(Age,Distance,data = trans,fill = license,geom = "polygon",
 +       main = "Age vs. Distance")
 > qplot(Work.Exp,Salary,fill = Engineer,data = trans,
 +       geom = "boxplot",main = "Work.Exp vs. Salary")
 > ### Missing values treatment ####
 > sum(is.na(trans))
 [1] 1
 > trans[is.na(trans)] = 1
 > sum(is.na(trans))
 [1] 0
 > ### Outlier Treatment ####
 > boxplot(trans[,-c(2,3,4,8,9)])
```

```
> ### Multi collinearity #####
> ## Correlation Plot ###
> cor = trans[,-c(2,3,4,8,9)]
> cor.plot(cor,numbers = TRUE)
> ## Eigen Values ###
> eigen = eigen(cor(trans[,c(1,5,6,7)]))
> eigen$values
[1] 3.06161749 0.75740812 0.13516991 0.04580448
> ## Scatter Plot ###
> plot(cor)
> ### Building the models ###
> ### Splitting of the datasets ###
> set.seed(77)
> splt = sample.split(trans$Transport,SplitRatio = 0.7)
> m.train = subset(trans,splt == TRUE)
> m.test = subset(trans,splt == FALSE)
> dim(m.train)
[1] 292   9
> dim(m.test)
[1] 126   9
> summary(m.train$Transport)
  0   1
268  24
> ### Building the logistic regression model ####
> set.seed(77)
> mglm = glm(Transport~.,m.train,family = "binomial",maxit = 100)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(mglm)

Call:
glm(formula = Transport ~ ., family = "binomial", data = m.train,
    maxit = 100)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.250e-05  -2.110e-08  -2.110e-08  -2.110e-08   1.123e-05

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.426e+04  1.628e+07  -0.001    0.999
Age          4.318e+02  4.969e+05   0.001    0.999
Gender1     -5.993e+02  6.926e+05  -0.001    0.999
Engineer1   -3.657e+02  1.125e+06   0.000    1.000
MBA1        -4.744e+02  5.517e+05  -0.001    0.999
Work.Exp    -2.697e+02  3.143e+05  -0.001    0.999
Salary       6.201e+01  7.174e+04   0.001    0.999
Distance     1.631e+02  1.855e+05   0.001    0.999
license1     6.410e+02  7.406e+05   0.001    0.999

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.6574e+02  on 290  degrees of freedom
Residual deviance: 8.1694e-10  on 282  degrees of freedom
  (1 observation deleted due to missingness)
AIC: 18
```

```
Number of Fisher Scoring iterations: 34
> ## Predictions for Logistic Regression ###
> set.seed(77)
> m.test.pred = predict(mglm,m.test,type = "response")
> m.test.predc = ifelse(m.test.pred > 0.5,"1","0")
> m.test.predc = as.factor(m.test.predc)
> m.test$Transport = as.factor(m.test$Transport)
> ## Confusion Matrix ###
> cf.lr = caret::confusionMatrix(m.test$Transport,m.test.predc,positive = "1")
> print(cf.lr)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   2
         1   0  11

               Accuracy : 0.9841
                 95% CI : (0.9438, 0.9981)
    No Information Rate : 0.8968
    P-Value [Acc > NIR] : 0.0001316

                  Kappa : 0.908

 Mcnemar's Test P-Value : 0.4795001

            Sensitivity : 0.8462
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9826
             Prevalence : 0.1032
         Detection Rate : 0.0873
   Detection Prevalence : 0.0873
      Balanced Accuracy : 0.9231

       'Positive' Class : 1

> cf.lr$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred V
          0.84615385           1.00000000           1.00000000            0.9826
           Precision               Recall                   F1            Preval
          1.00000000           0.84615385           0.91666667            0.1031
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.08730159           0.08730159           0.92307692
> plot(m.test$Transport,m.test.predc,xlab = "Actuals",ylab = "Predicted",
+      col = c("Blue","Red"))
> ### Building a Naive Bayes Model ####
> set.seed(77)
> m.nb = naiveBayes(m.train$Transport~.,data = m.train)
> print(m.nb)

Naive Bayes Classifier for Discrete Predictors

Call:
```

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

A-priori probabilities:
Y
         0          1
0.91780822 0.08219178

Conditional probabilities:
   Age
Y       [,1]     [,2]
  0 26.40672 3.023422
  1 36.50000 3.283688

   Gender
Y           0         1
  0 0.3171642 0.6828358
  1 0.1666667 0.8333333

   Engineer
Y      0    1
  0 0.25 0.75
  1 0.00 1.00

   MBA
Y           0         1
  0 0.7191011 0.2808989
  1 0.6666667 0.3333333

   Work.Exp
Y        [,1]     [,2]
  0  4.735075 3.145766
  1 17.291667 3.861506

   Salary
Y       [,1]      [,2]
  0 13.04216  5.345275
  1 41.27083 10.015444

   Distance
Y       [,1]     [,2]
  0 10.81418 3.176179
  1 17.49583 2.696774

   license
Y           0         1
  0 0.8432836 0.1567164
  1 0.1250000 0.8750000

> ## Predictions for Naive Bayes ###
> set.seed(77)
> m.nb.test.pred = predict(m.nb,m.test,type = "class")
> ## Confusion Matrix ###
> nb.cf = caret::confusionMatrix(m.test$Transport,m.nb.test.pred,positive = "1"
> print(nb.cf)
Confusion Matrix and Statistics
```

```
          Reference
Prediction   0   1
         0 113   2
         1   1  10

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.9048
    P-Value [Acc > NIR] : 0.001606

                  Kappa : 0.8565

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.83333
            Specificity : 0.99123
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.98261
             Prevalence : 0.09524
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.91228

       'Positive' Class : 1

> nb.cf$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred V
          0.83333333           0.99122807           0.90909091           0.9826
           Precision               Recall                   F1           Preval
          0.90909091           0.83333333           0.86956522           0.0952
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.07936508           0.08730159           0.91228070
> plot(m.test$Transport,m.nb.test.pred,xlab = "Actuals",ylab = "Predicted",
+      col = c("Purple","Pink"))
> ### Building a KNN model ####
> set.seed(77)
> m.knn = knn(train = m.train,test = m.test,cl = m.train$Transport,k = 17)
Error in knn(train = m.train, test = m.test, cl = m.train$Transport, k = 17) :
  no missing values are allowed
> ## Confusion Matrix ###
> knn.cf = caret::confusionMatrix(m.test$Transport,m.knn,positive = "1")
> print(knn.cf)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 115   0
         1   3   8

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.9365
    P-Value [Acc > NIR] : 0.03787
```

```
                 Kappa : 0.8296

 Mcnemar's Test P-Value : 0.24821

            Sensitivity : 1.00000
            Specificity : 0.97458
         Pos Pred Value : 0.72727
         Neg Pred Value : 1.00000
             Prevalence : 0.06349
         Detection Rate : 0.06349
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.98729

       'Positive' Class : 1

> knn.cf$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred V
          1.00000000           0.97457627           0.72727273            1.0000
           Precision               Recall                   F1            Preval
          0.72727273           1.00000000           0.84210526            0.0634
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.06349206           0.08730159           0.98728814
> plot(m.test$Transport,m.knn,xlab = "Actuals",ylab = "Predicted",
+      col = c("Brown","Yellow"))
> ### Doing a SMOTE analysis for a better dataset ####
> set.seed(77)
> s.train =SMOTE(Transport~.,m.train,perc.over = 501, perc.under = 121)
> s.test = m.test
> prop.table(table(s.train$Transport))

        0         1
0.5017301 0.4982699
> dim(s.train)
[1] 289   9
> ### Doing a SMOTE analysis for a better dataset ####
> set.seed(77)
> s.train =SMOTE(Transport~.,m.train,perc.over = 501, perc.under = 121)
> s.test = m.test
> prop.table(table(s.train$Transport))

        0         1
0.5017301 0.4982699
> dim(s.train)
[1] 289   9
> ### Building a Logistic Regression Model after SMOTE ####
> set.seed(77)
> sglm = glm(Transport~.,data = s.train,family = "binomial",maxit = 100)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(sglm)

Call:
glm(formula = Transport ~ ., family = "binomial", data = s.train,
    maxit = 100)
```

```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.62051  -0.00001   0.00000   0.00429   1.44261

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -61.90671 3354.44872  -0.018   0.9853
Age          -0.70352    1.46382  -0.481   0.6308
Gender1      -4.05529    2.41141  -1.682   0.0926 .
Engineer1    26.10027 3354.30712   0.008   0.9938
MBA1         -0.05514    1.97100  -0.028   0.9777
Work.Exp      0.99076    1.54787   0.640   0.5221
Salary        0.29917    0.21478   1.393   0.1636
Distance      2.69735    1.45608   1.852   0.0640 .
license1      6.51069    3.74982   1.736   0.0825 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 400.636  on 288  degrees of freedom
Residual deviance:  14.993  on 280  degrees of freedom
AIC: 32.993

Number of Fisher Scoring iterations: 20

> s.train$Gender
  [1] 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 1 0 1 1 1 1
 [41] 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 1 1 0 0 0
 [81] 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1
[121] 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0
[161] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0
[201] 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0
[241] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[281] 1 1 1 0 1 1 1 1 1
Levels: 0 1
> ## Predictions for Logistic Regression after SMOTE ###
> set.seed(77)
> s.test.pred = predict(sglm,s.test,type = "response")
> s.test.predc = ifelse(s.test.pred > 0.5,"1","0")
> s.test.predc = as.factor(s.test.predc)
> s.test$Transport = as.factor(s.test$Transport)
> ## Confusion Matrix ###
> glm.c = caret::confusionMatrix(s.test$Transport,s.test.predc,positive = "1")
> print(glm.c)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   2
         1   3   8

               Accuracy : 0.9603
                 95% CI : (0.9098, 0.987)
    No Information Rate : 0.9206
```

```
    P-Value [Acc > NIR] : 0.05955

                  Kappa : 0.7403

 Mcnemar's Test P-Value : 1.00000

            Sensitivity : 0.80000
            Specificity : 0.97414
         Pos Pred Value : 0.72727
         Neg Pred Value : 0.98261
             Prevalence : 0.07937
         Detection Rate : 0.06349
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.88707

       'Positive' Class : 1

> glm.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred V
          0.80000000           0.97413793           0.72727273           0.9826
           Precision               Recall                   F1            Preval
          0.72727273           0.80000000           0.76190476           0.0793
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.06349206           0.08730159           0.88706897
> plot(s.test$Transport,s.test.predc,xlab = "Actuals",ylab = "Predicted",
+      col = c("Green","Violet"))
> ### Building a Naive Bayes Model after SMOTE####
> set.seed(77)
> s.nb = naiveBayes(s.train$Transport~.,data = s.train)
> print(s.nb)

Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
        0         1
0.5017301 0.4982699

Conditional probabilities:
   Age
Y       [,1]     [,2]
  0 26.52414 3.062050
  1 36.13970 2.816203

   Gender
Y           0         1
  0 0.2965517 0.7034483
  1 0.1527778 0.8472222

   Engineer
Y           0         1
  0 0.2206897 0.7793103
```

```
  1 0.0000000 1.0000000

   MBA
Y           0         1
  0 0.6827586 0.3172414
  1 0.6250000 0.3750000

   Work.Exp
Y        [,1]     [,2]
  0  4.737931 3.327121
  1 16.886585 3.578042

   Salary
Y       [,1]     [,2]
  0 12.68690 5.034647
  1 39.96644 9.460266

   Distance
Y       [,1]     [,2]
  0 10.50966 3.190727
  1 15.95896 1.919303

   license
Y            0          1
  0 0.87586207 0.12413793
  1 0.07638889 0.92361111

> ## Predictions for Naive Bayes after SMOTE ###
> s.nb.test.pred = predict(s.nb,s.test,type = "class")
> ## Confusion Matrix ###+
> nb.c = caret::confusionMatrix(s.test$Transport,s.nb.test.pred,positive = "1")
> print(nb.c)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 112   3
         1   1  10

               Accuracy : 0.9683
                 95% CI : (0.9207, 0.9913)
    No Information Rate : 0.8968
    P-Value [Acc > NIR] : 0.002604

                  Kappa : 0.8159

 Mcnemar's Test P-Value : 0.617075

            Sensitivity : 0.76923
            Specificity : 0.99115
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.97391
             Prevalence : 0.10317
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
```

```
      Balanced Accuracy : 0.88019

       'Positive' Class : 1

> nb.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred V
          0.76923077           0.99115044           0.90909091           0.9739
           Precision               Recall                   F1           Preval
          0.90909091           0.76923077           0.83333333           0.1031
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.07936508           0.08730159           0.88019061
> plot(s.test$Transport,s.nb.test.pred,xlab = "Actuals",ylab = "Predicted",
+      col = c("Black","Orange"))
> ### Building a KNN model after SMOTE####
> set.seed(77)
> s.knn = knn(train = s.train,test = s.test,cl = s.train$Transport,k = 17)
> ### Confusion Matrix ##
> knn.c = caret::confusionMatrix(s.test$Transport,s.knn,positive = "1")
> print(knn.c)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 110   5
         1   0  11

               Accuracy : 0.9603
                 95% CI : (0.9098, 0.987)
    No Information Rate : 0.873
    P-Value [Acc > NIR] : 0.0007991

                  Kappa : 0.7934

 Mcnemar's Test P-Value : 0.0736383

            Sensitivity : 0.6875
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9565
             Prevalence : 0.1270
         Detection Rate : 0.0873
   Detection Prevalence : 0.0873
      Balanced Accuracy : 0.8438

       'Positive' Class : 1

> knn.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred V
          0.68750000           1.00000000           1.00000000           0.9565
           Precision               Recall                   F1           Preval
          1.00000000           0.68750000           0.81481481           0.1269
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.08730159           0.08730159           0.84375000
> plot(s.test$Transport,s.knn,xlab = "Actuals",ylab = "Predicted",
+      col = c("Black","Gray"))
```

```
> ### BAGGING AFTER SMOTE ####
> set.seed(77)
> bag.model = bagging(Transport~.,data = s.train,coob = TRUE,
+                     control = rpart.control(maxdepth = 10,minsplit = 3))
> summary(bag.model)
```

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = Transport ~ ., data = s.train, coob = TRUE,
    control = rpart.control(maxdepth = 10, minsplit = 3))

Out-of-bag estimate of misclassification error: 0.0138

$y
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [57] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[113] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[169] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[225] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[281] 1 1 1 1 1 1 1 1 1
Levels: 0 1

$X

	Age	Gender	Engineer	MBA	Work.Exp	Salary	Distance	license
50	25	1	0	0	5	13.7	12.7	1
410	34	1	1	0	12	16.9	16.6	0
310	24	1	1	1	6	11.6	11.3	1
126	22	1	1	0	1	7.5	5.1	0
161	28	0	0	1	5	14.6	7.2	0
397	29	1	0	0	5	14.8	15.4	0
155	25	0	1	0	4	11.5	7.0	0
15	27	1	0	1	8	15.6	9.0	0
253	24	1	1	1	0	7.6	9.5	0
318	26	1	1	0	4	12.8	11.6	0
257	28	1	1	1	5	14.8	9.7	0
311	24	1	1	1	0	7.7	11.3	1
395	23	1	1	0	1	7.9	15.2	0
200	32	1	1	0	11	14.7	8.4	0
373	27	1	1	0	1	8.9	13.6	0
160	30	1	1	0	8	14.6	7.1	0
410.1	34	1	1	0	12	16.9	16.6	0
25	26	0	1	0	2	9.8	10.7	0
267	26	0	0	0	3	9.6	9.9	0
128	25	0	1	0	4	11.5	5.2	0
215	27	1	1	1	5	13.5	8.8	0
200.1	32	1	1	0	11	14.7	8.4	0
24	28	1	1	0	5	14.7	10.5	1
167	26	1	1	0	4	12.5	7.4	0
413	29	0	1	0	6	14.9	17.0	0
145	26	0	0	0	3	9.5	6.3	0

159	23	0	1	1	4	8.4	7.1	0
131	27	1	1	0	4	13.4	5.5	1
14	24	1	1	0	6	12.7	8.7	0
133	25	1	1	1	4	11.5	5.6	0
247	26	0	0	0	4	12.9	9.4	0
9	24	1	1	0	4	8.5	7.5	0
292	27	0	1	0	5	12.9	10.6	0
414	29	1	1	1	8	13.9	17.1	0
238	28	1	1	0	5	13.6	9.3	0
296	24	1	1	1	0	7.8	10.7	0
133.1	25	1	1	1	4	11.5	5.6	0
320	26	0	1	0	8	14.6	11.6	0
48	24	0	1	1	1	8.8	12.6	1
128.1	25	0	1	0	4	11.5	5.2	0
285	30	1	1	1	8	14.9	10.4	0
53	18	1	0	0	0	6.7	13.0	0
340	26	1	0	0	3	9.8	12.2	1
55	25	0	0	0	2	8.9	13.2	0
162	23	0	1	1	1	7.5	7.2	0
198	25	1	1	0	4	11.6	8.3	0
81	29	1	0	1	7	15.0	19.0	1
299	30	1	0	0	8	14.6	10.9	1
16	28	0	0	0	10	19.7	9.0	0
49	24	0	1	1	2	8.7	12.6	0
225	26	0	1	0	3	10.8	9.0	0
259	25	1	1	0	1	8.6	9.7	0
27	25	1	1	1	7	13.6	10.7	0
414.1	29	1	1	1	8	13.9	17.1	0
184	25	1	1	0	3	10.6	8.1	0
377	31	1	1	0	7	15.9	13.7	0
190	30	0	1	0	8	14.6	8.1	0
359	28	1	1	0	6	13.8	12.9	0
125	26	0	1	0	3	10.5	5.1	0
273	24	1	0	1	2	8.7	10.1	0
248	26	1	1	1	3	10.8	9.4	0
229	28	0	1	0	5	14.6	9.0	0
261	31	1	0	1	7	15.9	9.7	0
182	20	0	0	1	1	8.5	7.9	0
187	22	0	1	0	2	11.7	8.1	0
315	27	1	1	0	6	12.7	11.5	0
347	26	1	1	1	5	12.7	12.5	0
414.2	29	1	1	1	8	13.9	17.1	0
367	24	1	1	0	4	13.8	13.2	0
328	30	0	1	0	6	15.6	11.9	0
144	30	0	1	0	8	14.6	6.3	0
23	34	1	1	1	14	36.9	10.4	1
14.1	24	1	1	0	6	12.7	8.7	0
296.1	24	1	1	1	0	7.8	10.7	0
29	21	0	0	0	3	9.8	11.0	0
161.1	28	0	0	1	5	14.6	7.2	0

393	28	0	1	0	6	13.9	15.1	0
209	28	1	1	0	4	14.6	8.6	0
51	34	1	1	1	15	37.0	12.9	1
229.1	28	0	1	0	5	14.6	9.0	0
247.1	26	0	0	0	4	12.9	9.4	0
121	29	1	1	0	7	13.4	4.1	0
14.2	24	1	1	0	6	12.7	8.7	0
246	26	1	0	0	3	9.9	9.4	0
200.2	32	1	1	0	11	14.7	8.4	0
350	27	1	1	0	8	20.7	12.6	0
307	29	1	1	0	5	14.9	11.2	0
241	27	1	1	0	3	10.6	9.3	0
154	28	1	1	0	6	13.6	6.9	1
316	26	1	0	0	4	12.6	11.5	1
247.2	26	0	0	0	4	12.9	9.4	0
349	30	1	1	1	8	14.7	12.6	0
145.1	26	0	0	0	3	9.5	6.3	0
347.1	26	1	1	1	5	12.7	12.5	0
303	24	1	1	1	3	9.9	10.9	0
212	30	1	1	0	7	15.6	8.6	1
9.1	24	1	1	0	4	8.5	7.5	0
341	26	1	1	0	3	10.7	12.2	1
338	34	1	1	1	16	34.9	12.2	0
253.1	24	1	1	1	0	7.6	9.5	0
231	24	1	1	1	0	7.9	9.1	0
203	24	1	0	0	2	8.7	8.4	0
321	26	1	1	1	6	11.7	11.7	0
65	24	0	1	0	2	9.0	15.1	0
54	23	0	1	1	2	8.8	13.1	0
67	25	0	1	0	2	8.8	15.2	0
164	30	1	0	1	7	14.4	7.3	0
208	24	1	1	0	2	8.5	8.5	0
44	25	0	1	0	5	18.9	12.2	0
44.1	25	0	1	0	5	18.9	12.2	0
388	22	0	1	0	0	6.8	14.5	0
415	25	1	1	0	3	9.9	17.2	0
146	24	1	1	1	2	8.6	6.4	0
205	33	1	1	0	11	15.6	8.5	0
205.1	33	1	1	0	11	15.6	8.5	0
198.1	25	1	1	0	4	11.6	8.3	0
14.3	24	1	1	0	6	12.7	8.7	0
415.1	25	1	1	0	3	9.9	17.2	0
389	25	0	0	0	3	9.9	14.6	0
309	28	0	1	0	5	13.7	11.3	0
143	26	1	0	0	2	8.5	6.3	1
297	25	1	1	0	3	10.7	10.8	0
73	23	1	1	0	0	8.0	15.9	0
73.1	23	1	1	0	0	8.0	15.9	0
156	20	1	1	0	2	8.5	7.0	0

[reached 'max' / getOption("max.print") -- omitted 164 rows]

```
$mtrees
$mtrees[[1]]
$bindx
  [1]  95  30 283 215  20  17 251 195 138 206 142  61  11  17  50  10 129 218 280 127 229 208 156  40 172
 [29] 246 142 242 282 113 242 253 229 165 173 274 213 223 201 285  81 202 153 114 247   2  38 140  24
100
 [57]  60  54 116  15 238 150 136  56 261  51 201 141 253 283 231 250  74 225 192 222 206  64 146 146 11
289
 [85] 132  20   2 278  90 100 131  78 148 105  19 216  84  48 161 227  17 156 178 151   4 265  85 282 157 1
[113]  15 262 217 116  75 170 224 142 203 252 174  12 255  43 273 114 196 162 200 201 249 282 153 222
28
[141]  17 170 281 177  75 269 159 202 148 140  97 127 209 150 184 159 288  29 140 266 150 103  94 203
59
[169] 205 177 238  89 199  49 221  49  45 175 284 126 237 286  87  98 112  69 186  69 105 170 182 123 10
59
[197] 192 280 196 242 273  59 285 163 241  10  23 242 277  99  38 168  18  50 177 111  17 251  81 274 184
31
[225]  43 271 233  14 106  31 262 186  60  83 253  77   7 148 192 149 187 251 109 235  82 262  14 158  61
[253]  87 264  71  81  62   6  64 144   3 173 265 136 175  64 194  29  61 169  33 208  35  84 107  51 178   8
[281] 151  16 265  99 238 104 224  59 163

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 140 1 (0.48442907 0.51557093)
   2) Salary< 20.41038 144   7 0 (0.95138889 0.04861111)
     4) Age< 31.5 125   0 0 (1.00000000 0.00000000) *
     5) Age>=31.5 19   7 0 (0.63157895 0.36842105)
      10) Work.Exp>=10.80659 12   0 0 (1.00000000 0.00000000) *
      11) Work.Exp< 10.80659 7   0 1 (0.00000000 1.00000000) *
   3) Salary>=20.41038 145   3 1 (0.02068966 0.97931034)
     6) Distance< 13.15 3   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.15 142   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[2]]
$bindx
  [1] 132   3 144 109 261 277   7  64 175 106 266 193   1 260 195  84 164 253 166  39  65 179 223   6 126  3
 [29] 177 254  65 268  77 190 246  78 270 209 152 113 246  67 150  37  39  23  21 220 167 193 286 170 102
262
 [57] 159 149 258 287   6 167 141 266 186 182 169  44 198  72 105  91 280 104  95  71 158 124 257  93 163
2
```

```
 [85] 152  36  57 255 205 116 289 196 245 126 241  45  89 263  21  22  67  49 275 224 106 149 133 115  12
189
[113] 190 249 154 212 186  19  54 276 181 111  21 289  84 155 106 126 284 245 135 115 265 265 118 196
159 268
[141]  47 176 134 268 164 276 261 191 241  69 150 216  94 227  79  89  83  94 161  16  97 252  93  88 233
[169]  13 143 168 245 154 286 141   2 100 145 287 217 117 271 181 199 178 240 191 222  82 270  59 194
33
[197] 132 179  58  84  47 218 241 185 243  73 273 140   2 182 153 270 150  47 210 213 224 235 275 190 27
93
[225] 287  40 275 102 280  82 141 160 164 135 112  82  81 177 156 196   7 115 140 288 215 192 287  46  3
115
[253] 124  92  60 220  84 158  43  14 276  77 104 192 261 118 159 174 261   7 144  94 149   6 179 107 252
[281]  24 157 115 174 283 197 155  64 211

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 132 1 (0.45674740 0.54325260)
   2) Salary< 15.7 120   0 0 (1.00000000 0.00000000) *
   3) Salary>=15.7 169  12 1 (0.07100592 0.92899408)
     6) Distance< 13.5 8   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.5 161   4 1 (0.02484472 0.97515528)
      14) Salary< 16.95 10   4 1 (0.40000000 0.60000000)
        28) Age>=33.3012 4   0 0 (1.00000000 0.00000000) *
        29) Age< 33.3012 6   0 1 (0.00000000 1.00000000) *
      15) Salary>=16.95 151   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[3]]
$bindx
  [1] 275 206 282  38 121 198  34  66   8  64 167 185 194 180 210 138  83 264 232 163 133 166 190  38 160
201
 [29] 130  53 106  34 103  38 170  44   5 139  33  56 218 244  73 196 239  96 202 169 179  39   4 111 272 18
 [57] 116 283 284  27 112  72 150 124 243 145 161  93  25 257  50  42 252  95  18 104  69 273 230 272 110
 [85]  15 135 142 145 180 122 145 240 221  78 156 247  50  74  85  64 247 164 153  16  25  96   1  30 114  4
[113] 114 236 251 191 111  83  95 275 159 155 122  38 170   9  51 216  56 210 157 212  73  84 237  17 179
[141] 259 106  97  44  37 159 278 124  62  17  21 231 233 171 112  39 269 211 267  68  27 240 170 113  37
233
[169] 201  35 288 211 205 136  39 276 165 255 277 221  90 196 178 248 121 141 229 242 148 107  53  49
136
[197]  50 204 165   8  24 199 119 183 140 104 100 253 119  36 100  52   9  78 197  22 246 203  18 243 277
[225] 216  56 114  14 288 185 169 188 270 166 273 229 128 144 249 181 120 247  86  29 151 106 225 261
22
```

```
[253] 156  12 173 238 159 218 168  52 271 203 103  80 196   6  76 219 166  12 194 124 106 204  65 249 14
197
[281]  95  66 165  20 188 266 171  31 188

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 144 1 (0.49826990 0.50173010)
   2) Salary< 16.95 142   4 0 (0.97183099 0.02816901)
     4) Salary< 15.7 132   0 0 (1.00000000 0.00000000) *
     5) Salary>=15.7 10   4 0 (0.60000000 0.40000000)
      10) Gender=1 6   0 0 (1.00000000 0.00000000) *
      11) Gender=0 4   0 1 (0.00000000 1.00000000) *
   3) Salary>=16.95 147   6 1 (0.04081633 0.95918367)
     6) Distance< 13.35 6   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.35 141   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[4]]
$bindx
  [1] 200 136  67  28  41 113 179  10  26 160  85 122  63 127 124  90 164 200 233 220 167 231 270  55 150
250
 [29] 260  99  84  54 283 117 100 180 244 127 274  74 245 197 195 125 252 226 148  82 253 254  50 269  1
86
 [57] 214 121 108  89 164  79 195 222  72  52  16 222 157 179  20  49 195 213 129 271  43 195 196 182 265
3
 [85] 224  92 187 113 277 159 203 175 261 188  80  42 179  53  37 239 167 160 175  94  73   5  96 147 194
[113]  53 131 173 162 133  72  95  69  25 242 247 265  28  76  92  16 215 200 164 149 227 113 152 255 283
102
[141]  32 262  22  32  45 213 174 110  39  37 225 114 200 174  63 214 187 288 127  40  34 137 134 157 144
206
[169] 250 240  29 127  33 253  26  37 264 179  48 217   9  44  72  24 136 245 209 245 150 169 156 243 244
204
[197]  87  35 205  93 289 104 182 134  62  79 106 188  34  21  94 146 207  81   1 101 213  48  61  52 221 10
[225] 165 120 145 160 288 117 174 284 144 103  34 144 125 199 105  77  44  61 256  69 120 289 174 229
146
[253] 150  18  52  25  76  99  77 210 131 151 163  59 270  93 204 197 183  12 113   8 183 129  58 167  50 2
[281]  83 156  48 148 283 131 196 183 234

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

```
1) root 289 143 0 (0.50519031 0.49480969)
  2) Age< 30.43512 134   0 0 (1.00000000 0.00000000) *
  3) Age>=30.43512 155  12 1 (0.07741935 0.92258065)
    6) Distance< 13.5 12   0 0 (1.00000000 0.00000000) *
    7) Distance>=13.5 143   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[5]]
$bindx
  [1] 194 131 287 267  57 186  22 200 151 210  72 249   8 119 271  38 272 187 186 149 151 229   7  19 285
268
 [29]  22 116 212 106  58 196 208 162 197 189 179 168  65  32 158 178 136 113 256 219  60  53 122  82 13
238
 [57] 227  32 263  29  66 251  58 184 211  92 216 177 126  56 187 246  76  12 251 150  71 225 126 216 141
203
 [85] 172 248 153 180   2 162  72 225 239 192 156 101 160 282  62   8 195 114 199 109 112  43  25  48  32
[113]  59   5 181  32 130 185  62 181 116 270 110 205  85 142  54 110  23 203  18  72 175 138  88 188 110
160
[141]  50 190 275  77 243  75 210 128 199 158 195 239 175  65  15 101 184  45  20  41 188  69 216 277 269
186
[169] 267  56 169 171  88 225  12 210 190 170 170 254 200  71  23  63 244 221 161 142  17  69  10  62 217
207
[197] 153  52 228  51  34  66  92 286 199 190 279 133  15  82   9 263 239 195 190  85 271 147 211 250 193
217
[225]  81  34 198 257  88  58 222  82 176   1 122 150  40  75 273 284 148  53 223  49 188 147 117 247 172
139
[253] 158 262 149 160 159  37 228 101 218 242 187 225 261  54 142 262 212  13 161 171 118 275 226 127
103 256
[281] 103 254 233  21  47  69 102 178  94

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 132 1 (0.45674740 0.54325260)
  2) Work.Exp< 9 121   0 0 (1.00000000 0.00000000) *
  3) Work.Exp>=9 168  11 1 (0.06547619 0.93452381)
    6) Distance< 12.25 9   0 0 (1.00000000 0.00000000) *
    7) Distance>=12.25 159   2 1 (0.01257862 0.98742138) *

attr(,"class")
   class
"sclass"
```

```
$mtrees[[6]]
$bindx
  [1] 138  95  65 196  69 275 121  64 268 154 215 119 252 288  61 245  88  67 272 118  16  95 151 286 263
 [29]  25 178  60 266 191 181  75 227 115 145  54 144 176 114  89 189 220 178 121 289 287   8  61 237 140
126
 [57]  89  84  96 107 190 180 205 133  97 106   4 265 166 177  92 122 212  51 134  55 208 139 261  93 138
195
 [85] 106 127 226  90 203 195 241 185 202 126  19 249 145  97 276  50 268 212 124 268  86 279  88 115 22
161
[113] 137 144 118  52 175 177 159  93 104 272 237 227 230 189 133 148 233 228 155 179 155 235 131  19
26
[141] 104  78  13 174  61 187  90 115 282 270 279  35 223   2 266 252 214 138 122 184  40 149 259 159 22
102
[169]  82 103 261 262  48 157 215  26 117 118 145  85 257 253 103   2 178 185 240 109  78 138  30  21  60
113
[197] 143 246  66 200 189  94 122 141 172  46 283  28  81 130 117 153  37 105  55 161 282 168  10   7  47
[225]  90  14  10 186  17  68  62 229  95 230 245   7 114  90 261 228 275  11 234  30 105  55 194 206  46  4
[253] 268   9 240  73  24   6 263 227  26  34  33 173  67  11 244 203  38 201 168 191 101  68 133  96  39  48
[281] 242 104  49 146 275 280 100 275 169

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 132 0 (0.54325260 0.45674740)
   2) Salary< 15.7 149   0 0 (1.00000000 0.00000000) *
   3) Salary>=15.7 140   8 1 (0.05714286 0.94285714)
     6) Distance< 13.9 5   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.9 135   3 1 (0.02222222 0.97777778)
      14) Salary< 17.17943 11   3 1 (0.27272727 0.72727273)
        28) Age>=33.3012 3   0 0 (1.00000000 0.00000000) *
        29) Age< 33.3012 8   0 1 (0.00000000 1.00000000) *
      15) Salary>=17.17943 124   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[7]]
$bindx
  [1]  55  95 245  78 144 186 107  18 104 120 267 243  59 216  90 122 100  98 162 180 140  27 163 223  80
 [29] 193 248   7  71 191 221  17  49  67 194 224 163  38 265 133 162 166  89  49 192  86 140 159 172 246
222
 [57]   5 203 228 144 126 225 159  41 265 214 208 195  56 279 102 165  36 199   1  43 198 184 247 152 136
93
 [85] 262 102 217 283  88  72 149  70 284 228 277 137 143  22 195  82  18   7 145  16 150 269 142 189 158
289
```

```
[113] 288 208 289  27  36  80 140 102 233 210 252 264  52 159 279 160  71  11 229 106 271 232  29   1 199
279
[141]  57  43  64 184 204  77 238  40  94 139  33  14  32 234  83  88  98 115 108  81 235 132 225 257 174 1
[169] 285  86  42 227 149 185 183 243  76 233 133  39 156   2 199  90 235 252 110  25  78 280 272   8 189
[197]  27  80 204  35  29 195  38 278 143  32 100 206  19 142 135  99 260 141 240 251  30 100 261  84   9
[225] 289 247  72 209 241 247 127  71   4 215  93  84 177 176  26 203  10 147 131  57 285  38 195 138 161
181
[253] 219  65  90 103  60 142  48 277  33   2 156  21  71 268  96   2 106 117  33 156  99 154  53 149  30 252
[281] 194 206 196  42 185  51 156 243 167

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 136 0 (0.52941176 0.47058824)
   2) Distance< 13.9 140   0 0 (1.00000000 0.00000000) *
   3) Distance>=13.9 149  13 1 (0.08724832 0.91275168)
     6) Age< 29.5 9   0 0 (1.00000000 0.00000000) *
     7) Age>=29.5 140   4 1 (0.02857143 0.97142857)
      14) license=0 12   4 1 (0.33333333 0.66666667)
        28) Salary< 22.6808 4   0 0 (1.00000000 0.00000000) *
        29) Salary>=22.6808 8   0 1 (0.00000000 1.00000000) *
      15) license=1 128   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[8]]
$bindx
  [1] 147  86  77 127 136  41 208 234 234  21   8 250 227  79   1 266  33  61 273  67  54 113  18  50 190 130
 [29]  47  95 102  64 232  86 250 147 228 121 229 256  29 158  46  33  39  18 122  57   9  14 252 266 180 14
 [57]  50 125 175   5  18 124 115 198  92  97 153  94  83 275 175  38  51 262  66  50 217  75  85  84  14  66
 [85]  62 282  86 250 109 270  90 140 230 227 162 270  80  25 187 214 124  58 182  51 267 152 106 106 12
5
[113] 221 214 249 189 169 113 265  59  98  64 197 113 126   7  23 283  33 123 135 268 221 123 153   8 112
98
[141]  29 275 241 210 244 281  77 115 123 197 208 117 232 122 253  91 168 252 181  73 209 122 208  52
77
[169] 204 214  70 237 192  74 182 227 193 235 200 126 114 166 151  58 230  46 142  31 223 174  66  85 16
164
[197]  85 144 187 136 216 161  64 224 168 145 100 248 252 273 201 239 146 280   7  55 221 176  55 180 2
77
[225]  64 156 201 119 109 101 232 113   9 237 147  37  75  35 105  97 285 158 108 260  71  13 187 188 106
167
[253] 120 186  47  65  92  92 152  36  15  94 175 102 250  80  94 127  41 116 212  14 130  79 138 144 151
[281]  53 274 187 274 196 280 195 115  35
```

```
$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 129 0 (0.553633218 0.446366782)
  2) Salary< 15.7 149   0 0 (1.000000000 0.000000000) *
  3) Salary>=15.7 140  11 1 (0.078571429 0.921428571)
    6) Distance< 13.5 10   0 0 (1.000000000 0.000000000) *
    7) Distance>=13.5 130   1 1 (0.007692308 0.992307692) *

attr(,"class")
   class
"sclass"

$mtrees[[9]]
$bindx
  [1] 184  76  97  13  90 127  72 163  63 185   2  24  65 167  15 244 173 223 144 119 236 113 194  53 216 21
 [29]  47 170  68   2  21  72 160 223 181  10 246  28 175 182 124  43  79 159  72  37 288  99 259  47 190 13
 [57]   3 151 266 229 222 277 237 121  23 191 108 237 138 259 246 173  60 280 139 101  70  43   1  59 233
 [85]  98  69 263 275 241 284 197  95 154 240 173 220  99  36 289 231 215 255  64 229 261 180  50 252  93
142
[113] 125 209  20 123 198  74  34 204 239 203  25 247 278 248 114  92  83  72  28 171 189 253  47 120 102
86
[141]  57 263 153 133 160 143 269 268 255 261 214 151 263  96 206 255  90  71 283 213 160  27 215  24  4
178
[169] 255 243 114  61 265 289  61 244 236 160 287 159  38 133 159 147  58 200   4  60 189 272  82 256 25
178
[197]  45 214 100  40  17 114 219  10 166 239  94 263  93   7 185 180 158 161 205  43 104 219 173  37 158
159
[225] 155 133 222 245 102  56  40  77 219 264  68  14  61 252 230 280  58 289 287  77 210  55 228  71 168
92
[253]  96 280 259 220 103 104 157 224  64 180  82 127 142 235 172  63 163 212  78 178 203  53  96  65  11
123
[281] 205  75 257 193  18  79 152  99 263

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 137 1 (0.47404844 0.52595156)
   2) Salary< 15.7 120   0 0 (1.00000000 0.00000000) *
   3) Salary>=15.7 169  17 1 (0.10059172 0.89940828)
     6) Distance< 13.9 14   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.9 155   3 1 (0.01935484 0.98064516)
      14) Salary< 17.32231 13   3 1 (0.23076923 0.76923077)
        28) Age>=33.3012 3   0 0 (1.00000000 0.00000000) *
```

```
      29) Age< 33.3012 10   0 1 (0.00000000 1.00000000) *
     15) Salary>=17.32231 142   0 1 (0.00000000 1.00000000) *

attr(,"class")
  class
"sclass"

$mtrees[[10]]
$bindx
  [1]  10 284 201 266  92  46 168 111 170 138 198 121  90 218 121 189 105 200 166  21 179  71  44 176 209
274
 [29] 145 133 102 122 158  57 100 129  67 100 176 273  52 202 272  33  34 283  79  58  77 240 135 152 122
3
 [57] 123 118 110  14 267 148 206  43 199 100  71 193  27  43 260 238  86 119 128 145 225 262 193 128 17
14
 [85] 116 191  51  89 163 139 131 246 208  16 154  78 255 123  85 115   9 204 216  74 216  57  90 203 103
100
[113] 156 137 251  70 249  61  85  19  16 268 171 245  26 273 209  57  33 105 229 154 209 237   6  58 181
[141] 246  91  37   4  97 159 115  79  83 117   2 271 214  99 117 179 120 128 205 107 191  10  69  32 210 2
[169] 146 255  98  97 235 150 234  54 113 142  12  85 203 262 227 193 211 233 262  38 117  99 221 201  3
226
[197] 176  13 206 190  40   9 140 200 106 137  71 180  69  57 120 262   4  17 228  74  28 176 233 171  63 1
[225] 189  44  70 283 238  71 237   1 229  38 241 288   4 132 183   7 192 183 101 153  32 146 116  37 115
[253] 150  69 229 136 155 104 230 139 127 128 159 234 178  65 267  25 281 278 104 158 281 167 268  90
123
[281] 224 283  67 276  14 232  57 289 263

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 135 0 (0.53287197 0.46712803)
  2) Salary< 15.7 141   0 0 (1.00000000 0.00000000) *
  3) Salary>=15.7 148  13 1 (0.08783784 0.91216216)
    6) Distance< 13.5 11   0 0 (1.00000000 0.00000000) *
    7) Distance>=13.5 137   2 1 (0.01459854 0.98540146) *

attr(,"class")
  class
"sclass"

$mtrees[[11]]
$bindx
  [1]  30  56 148  74 207 186 158 243  12 183  86 275 182 114 153  69 172 184 215 141 111 121  93  70 126
125
 [29] 256 142  61 239   9  36 205 100 193 165  43 157  65 234 188 268  23  37 287   6 233  34 102  77  64 15
 [57]  37 169 265  36 171   7 127 152  96 172  81 279 122 256 254 237 221 183 202 200 190 145 196 211 1
213
```

```
 [85] 113 123 171  39 257 227 176  71 162 175 108 280  38 272 180  12 191 210  38 226 186  65 120 159  7
49
[113]  33 161 174 131 187   8 195 108 244 136 168   4 138 149 153 140 258  22 133 250  68 246   7 196 16
130
[141]  49  50   6 204 286   8  57  88 146 143 125 191  14 126   2 150 132  85 187  85  90 195 272 155  10 27
[169]  65 258  28 186   9 131  48 102 172 148  35   3  47 149 100  91 272 210  48 256  76 157 123 226 122 2
[197] 274  77 254  85 247  42 287  40  43 195 238 271  69 251 250 196 166  86 149 152 138 160 173 140 2
6
[225]  18 201  95  46  96  97  21 171  91 135 230 170  32 151  37 267 147 241  95 196 151  39 137  38  27 2
[253]  28  34 137  28 238 265  85 143 231 180 213 270  47 227 283 251 182 123  36 198 247  31   1 256 17
[281] 159  94   3  79 178 112 218 275 121

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 144 0 (0.501730104 0.498269896)
  2) Salary< 15.7 133   0 0 (1.000000000 0.000000000) *
  3) Salary>=15.7 156  12 1 (0.076923077 0.923076923)
    6) Distance< 13.9 11   0 0 (1.000000000 0.000000000) *
    7) Distance>=13.9 145   1 1 (0.006896552 0.993103448) *

attr(,"class")
   class
"sclass"

$mtrees[[12]]
$bindx
  [1]  91 182 210 181 156 254 215 161 113 133 147 218 218 122 259 151 244 232 191 267 230  21  73  78  9
204
 [29] 199  48  62  77  31  40  90 282 243 266 277  69   7 163 271 189  11 206 188  95  36 216 155  21 247 20
 [57] 214 150 195 270  13 169  94 124  63 122 149   7  89  38 275 155  33 198  33 212 122 236 172 218 173
 [85]   2  76 141 254 125 184 227 108 130   4  89 118  70  94  86 198 115   4  19 168 242 208 112 120 104 2
[113]  90 191 171 104 236 200 174  98 245 167  76 274 210 134 248  28 150  59  23 181 249  18 269 159 1
179
[141] 231  16 231 244 229 190 154  19 254 289  16 235  44  59 142 104 259  19 117 178 186 230 245 168 2
35
[169] 182 119  41 126 217 194 141 114  44  54  23 207  13  43 277  31 122  28 158 156 161  24  15 183 112
[197] 113 226 157  70 223  99 132 214   7 153  81 215 287 206  84 175  11  71 285 278  42 287 220 176  81
152
[225] 233 156  49 287  90 265 211  10 243 216 190 161 117 218 209  93  26 264 116  53 211 279 281 198 2
273
[253] 206  15  29 189 116 132 127 269 117   5 262  45 245  82 179 270  13  85 144 251 264 252 137  13 109
257
[281] 102 223 167 273 280 240   5 206  80

$btree
n= 289
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 132 1 (0.456747405 0.543252595)
  2) Work.Exp< 9 124   0 0 (1.000000000 0.000000000) *
  3) Work.Exp>=9 165   8 1 (0.048484848 0.951515152)
    6) Distance< 13.15 7   0 0 (1.000000000 0.000000000) *
    7) Distance>=13.15 158   1 1 (0.006329114 0.993670886) *

attr(,"class")
   class
"sclass"

$mtrees[[13]]
$bindx
  [1] 130 140 159 209 158 252 258  96 104 239 210  59 240   2 137 287 196 125  88  31 237 205 233 171 22
52
 [29] 141 104  19 243 158  37 184 237 177 280 199 239 259 127 201  24 233 192  24 221 221 137 284  37  5
123
 [57] 117 275 162  66 265  97 225  49 175 199 126  94 239 148  54  55 187 247 219 201  24  26 253 107  43
268
 [85]   3 120  12 247 154  63 107 238 283 208 172  19 200  30 246  77 154  93 146 253 214  96 281 153  94
[113] 119 154  51  27  48  46  22 244 117 161  55 242 255  41  51 270 283  38 142 251  15  74 217  13  29 1
[141] 228 251 263 253 137 223  28 277  77 133 236 152  56 244  30 289  82 139 215  67 184  17 102 254 18
22
[169] 185 163 200 147 173 149 198 137 183 278 266 218  86  39   4 187  90 280 156  29  88 158  98 205 27
19
[197] 118  65 191 285 176 235  23  50  67  56  48 269 201 186 217 277 216  47 110 242 257  83  63 221  25
282
[225] 246 120 181 178  82  92   2 221 248 132 163  54 283 268  59 221  76  70 251 135 264 227  39 206 229
[253]  41 278 182 206 202  52  42  69  34  97 114 134  52 279 287   6 189 224  69  50  97 167  51  55   7 280
[281]  55 225   6 155 201 164 196  58  60

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 137 1 (0.47404844 0.52595156)
   2) Work.Exp< 9 126   0 0 (1.00000000 0.00000000) *
   3) Work.Exp>=9 163   11 1 (0.06748466 0.93251534)
     6) Distance< 13.5 7   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.5 156   4 1 (0.02564103 0.97435897)
      14) Salary< 16.95 9   4 1 (0.44444444 0.55555556)
        28) Age>=33.3012 4   0 0 (1.00000000 0.00000000) *
        29) Age< 33.3012 5   0 1 (0.00000000 1.00000000) *
      15) Salary>=16.95 147   0 1 (0.00000000 1.00000000) *
```

```
attr(,"class")
   class
"sclass"

$mtrees[[14]]
$bindx
  [1]  60 147 128 127 278 104 240 104 190 234 219   9 229  46  93 228  86 222 148   6 273 252 111  90 132
 [29] 103 170  30 181 138  41  82 276 137 124 149 154 223 280 121 282 184 192  84 177 213 205 167 206 2
164
 [57] 228 161 247 129 276 124 112  53  34 220  98  93   3  54 275  58 242 201 251  67 126 251  36 149 152
 [85]  62 262  97 110   3 182  55 283 177 227   6  87 254 235 234  15 255 108 257  73 225 250 194  19  59 2
[113] 186 177 285 249  53   2 235 186 124 112 167 152  93  94  81  22  79 232  68 267 169  82 153  49  76
[141]  50  30 251  25 197 177 260 169 273 171 165 161 192 218 281  57 255  16  51  36  99 202  17 166 223
59
[169]  14 150  92 243 143 109 120   7 128 134 270 257 224 100 285 177  85  20 289 113  65  34   6 189 139
[197] 268  29 101 268   7 202 134 129 198  28 217 222 248 102 135 281  70 254  18 276 104   7 169 158  76
194
[225] 199 180  49 125 200 249   3 225 192 165 285   7  64 114 118 176 267 175 162  49   9  91  91 207  92 2
[253] 257 113  61  49  76  17 264  13  36 273 238   5 272 170 133  75 274 193 145 206  97 287 243 101 212
167
[281] 168 110  45 166  44 228 206 189 237

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 137 1 (0.47404844 0.52595156)
   2) Salary< 20.98691 138   6 0 (0.95652174 0.04347826)
     4) Age< 31 125   0 0 (1.00000000 0.00000000) *
     5) Age>=31 13   6 0 (0.53846154 0.46153846)
      10) MBA=0 7   0 0 (1.00000000 0.00000000) *
      11) MBA=1 6   0 1 (0.00000000 1.00000000) *
   3) Salary>=20.98691 151   5 1 (0.03311258 0.96688742)
     6) Distance< 13.5 5   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.5 146   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[15]]
$bindx
  [1] 272 150 134 107   1 229  51 199   8 163  93 137  87 247 242 236  47  60  61  90 262 206 287 170 106 2
 [29]  44 120  63  56   8 167  41  50 125 226 109 101 220  33 241  80 203 259  27 143  65 186 112  93 155 2
 [57] 191  41  43 226 280  84 220 285  86  87 122 220  80   4 167 115 171 193 189 261  20 166 220   3 278
 [85]  45 194 164  80 136 267 190 175 225  22  62 133 193  10 252 270 159 185 264 145  80 146 231  96  8
102
```

```
[113] 222 152 102 282 188  42  72 178  18 200 230 144 133 125 119  50 289  43 288 274 126 232  73 177 2
143
[141]  99 166 250  71 116  51 194 282 157 278 170  63 246  60 275 209 204   5 103   7 185  24  82  55 228
[169]  97  46 248 122  69 161 210   4 191 247 188 278 105 212 164  93 245  78   6 222 155 172 126 110 154
42
[197] 282 146 257 168   4 242  47 172 155 121 160  41 100  21   2  74  71  64  69 197 276 204 120 176 245
[225] 113 136 153  93   5 163 112 160  95 107 178 221 140 116 173 204 281  22 252 226 172 156 243  71 1
175
[253]  90 197 200 188 107 160  92  11 207 174  77  85 231 287 116  55 255 280  13 283  71 193 189  49  36
203
[281] 285  77  98 264 199  83 269  65  12

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 143 1 (0.494809689 0.505190311)
  2) Salary< 22.30747 144   3 0 (0.979166667 0.020833333)
    4) Distance< 19.05 142   1 0 (0.992957746 0.007042254) *
    5) Distance>=19.05 2   0 1 (0.000000000 1.000000000) *
  3) Salary>=22.30747 145   2 1 (0.013793103 0.986206897)
    6) Distance< 13.15 2   0 0 (1.000000000 0.000000000) *
    7) Distance>=13.15 143   0 1 (0.000000000 1.000000000) *

attr(,"class")
   class
"sclass"

$mtrees[[16]]
$bindx
  [1] 202  83  26  94  26 144 262  16 175  14 204 252 122 211 126 249 260  59  83  32  50 262 171  73 147  7
 [29]  39 122  46 236  17  60 257   8 209 196 126  65  34  71 278 221 168 108  94 254  83  20 244 227 142 2
 [57] 215   6 178 133 231 256  70 217 209 242 286  23 219 159 138 128 171 277 147 113  15 104 246  39 10
211
 [85]  18 107 224  84 170 171 205  34 216 245 162 145 257 151 187  11 287 130 273 289 145 226 258 162 1
96
[113]  16  68 278 149 269  36 129  36  46  27 206 276 151 249  61   9 172  28 109 275 236  54  18 221 158 2
[141] 128 172  33 203 135  95 192 106 209 167 283 210 138  95 275  98  58 120  34 109  10 243 179  64  88
237
[169] 200  64 207 133 107 149 145 280 205  97  55 212 120  30  81 260 115 235 255 242 194  91 250 123 1
167
[197] 114 236 276  27  47 179  50 195 161 110 127  30 233  35 220 136  32 179  21 193 220  71 206   9 239
69
[225] 177 247 210  22 106  46  40  22 254 204 163 144   2 246  56 123 206 255 134 263 215  75  33  72 237
46
[253] 114  87 149  94 182 138 271 220 227  14  72 225  56 154 162 202 278 211  66  10 182 261  61 159  29
192
[281] 114  43 117 102 211 100 207 137 156
```

```
$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 142 0 (0.50865052 0.49134948)
   2) Salary< 17.17943 145   4 0 (0.97241379 0.02758621)
     4) Salary< 15.7 136   0 0 (1.00000000 0.00000000) *
     5) Salary>=15.7 9   4 0 (0.55555556 0.44444444)
      10) MBA=0 5   0 0 (1.00000000 0.00000000) *
      11) MBA=1 4   0 1 (0.00000000 1.00000000) *
   3) Salary>=17.17943 144   6 1 (0.04166667 0.95833333)
     6) Distance< 13.15 6   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.15 138   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[17]]
$bindx
  [1] 248  16 247 156 162  97 217 230 209 248  91 199 142 279  96  16  45 118 182 144  99  11 115  78 271
 [29] 218 188 159  70  93  43  91 267  79 177  69 232 175 275 189 185   2 141 178  86   5 181  57 278 121 1
 [57]  55  16   8 256 149 281 198 266  18  94  37  12 147   5  22 242 184 179 171 262 193 239  60 284 139 1
 [85]   7 156  93 197  25  70 230 114 104 268 246 281 273 278 135 219 239 133  20  53 270  82  43 219 109
167
[113] 206 150  50  51   1 176 248 144 108  99 199  79 262 221 110 169 237 278 192  15 243 112 285  62 27
155
[141]   2 214 284  28  86 155 274 178  74  76 200  96 248  21 195 166 281 126 159  51 145   9 224 143 235
218
[169]  12 107 223 198 206  34 263   7 211 261  75 174 213 159 254 227 129 242 270 191 158 137 130 135
278
[197] 146 260 130 174  71   9 189 152 275 281 278 200 122 251  32  53  63 253 239  62  28  29  56  65 223
154
[225]  52   6 276  50 124 160  16  75 116 283  61 147  55 130 284 154 258 215 203 128 248  68 228 280 26
56
[253] 257 288  62 244 280 232 129 245 254   5 171 215 143 183  27 287 119  44  62  19 207 218  68 106 24
136
[281]  34  45 111  34 102  71  91 139  70

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 134 1 (0.46366782 0.53633218)
  2) Work.Exp< 9 123   0 0 (1.00000000 0.00000000) *
```

```
  3) Work.Exp>=9 166  11 1 (0.06626506 0.93373494)
    6) Distance< 13.5 9   0 0 (1.00000000 0.00000000) *
    7) Distance>=13.5 157   2 1 (0.01273885 0.98726115) *

attr(,"class")
   class
"sclass"

$mtrees[[18]]
$bindx
  [1] 130 126 215 149 158 245 174  30  93 233  14 272 146 122 260 204 171  71  95 211  37 260  55 120 244
147
 [29] 202  46 225 182 165  69 141 105 151 257 288  93 162   9  24  20 124  19  97  57 134 229  76 225   1 14
 [57] 267 132 125 208  48  78 163 220  63  93  92 120   3 234 289  24 181  86  78 106 132  13 247 197  18 2
 [85] 119 115  80 200 261 174  98  57 228  61  10 233 171 180 267 169 286  70  62 224   7   6  60 245 113 1
[113] 208  12 225  32  86 141 101 284 187 140 123  56  25 139 263   1  77 272 241 137 103 176 181 211 18
145
[141] 242 202 226 191 253 239 106 216  88 188 244 199  47 276  48 167 131 112 209  46  51   1 100 113 1
111
[169] 234 228 209  67 162 213 178 284  72 288 196   4 113 169 280 276 286  44  80 229  96   7  37  39  49
[197]  12  54 171 176  62 245  25 124 186  80 175   2 235 209 283  45  27 202  30 179  87 120 202  24 215
[225] 289 159 139 230 197 239 248  88  25  48  44 289 150  98  98  98 262  68  30  54   9  89 213  82 102 2
[253] 179 225  77 158  83 166  52 142 144  31 100  57 132 169 177  78 158  17 157 148  36  51 122 206  74
112
[281] 207 169   9 230 214  14  87 212 148

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 136 0 (0.52941176 0.47058824)
  2) Age< 30.43512 145   0 0 (1.00000000 0.00000000) *
  3) Age>=30.43512 144   8 1 (0.05555556 0.94444444)
    6) Distance< 13.9 6   0 0 (1.00000000 0.00000000) *
    7) Distance>=13.9 138   2 1 (0.01449275 0.98550725) *

attr(,"class")
   class
"sclass"

$mtrees[[19]]
$bindx
  [1]  15 140 219 242 272  51  40  43 102 122 268  13 132   4 169  88 147 170  85  98  92 162 190  68 133 1
 [29] 208 121 236 163 157  12  69 127 279  96 208  49  14  71  93  32  84 243 286  40 201  23 252 281 248 2
 [57]  44 109  55 161 133 241 121 191  90 113  89  94  62 253 193 109  86  45 156 190 257  99 109 190 263
190
 [85] 258 217 199  22 243 200  54 287 152 100  71 200 266  64 135  23  22 191 271  69  80 122  56  13  96
[113] 208  35  26  44 229 256   4 227  88 182 236 195  14 238 152  83 210 188  58  17 137 128   7 208  43 1
```

```
[141] 206  71 193 179 242 128 107  71  75 142 272  21 185 120  95  20 167 133  12  56  75  68 257  60 130
[169] 158  79 120 133  83 164 130 143 114  25  84 264 100  57 277  72 108 282 274 182  28 126 241  15 26
95
[197] 222 117 153  53 186  13 228  63  50 106 172 191 282 178  50  18 153  53 213 149 130  60   1 133 120
200
[225] 199  35 131   9 195 160   5 215  92 168  27  58 110 169  59 189 251 253  83 208  52 129  96  80  83  7
[253] 168  60  47 268 200  31  62 259  27 125 116 236 161  78 210  49 134 255 240 260 190 179  54 117  41
4
[281] 219 156   8 262 106  92 111 246  28

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 122 0 (0.577854671 0.422145329)
  2) Work.Exp< 9 152   0 0 (1.000000000 0.000000000) *
  3) Work.Exp>=9 137  15 1 (0.109489051 0.890510949)
    6) Distance< 13.5 14   0 0 (1.000000000 0.000000000) *
    7) Distance>=13.5 123   1 1 (0.008130081 0.991869919) *

attr(,"class")
   class
"sclass"

$mtrees[[20]]
$bindx
  [1]  83  17  81 289 138  38  36 176 109 279 244  76  94 158 150 144 218  67  94 181  19 115 197 171  55  6
 [29]  15 259 278  96 285 202 276 251  93 118 135  52 147 171  72 285  52 160 248 131 162 162 278  21  46
243
 [57]  59 251  14 221  21 117 128  44 142  99 139 181 108 267 239 251 100  68 149 170 167  13   6 106 123
201
 [85] 268 173  70 215 261 230 264  45 111  35  93 236 245   4 224  25   6 195 272 254  98 266 227 286 144
228
[113]  29  51 155 202 187  31 166 226 266  93  32 123 241 121 251  26  58 150  83 223  35 224  87 235 155
127
[141] 277  63  77 253 136  99  58 201 146 180 118 234   9  86 219 165  65  54 176 163  94 161 128 164   2
[169]  75 215 266 146  14 217 131  33  14 209 116 139 107 162 193  45 172 101  87  27 162 226 112 117 11
62
[197] 138 238 184 227 201 133  91 160 109 198 102 231 251  80 221 216 110 256 187  17 127 184  59 282
21
[225]  13  41 220   7   3  92 192 105  41 194  64 158 180 176 197  91 144  81 162 109 211 182  45 156  60 1
[253] 115  32 241 253  87 239 106 234 174  39  18  25 234 121  83 164 129 118 201  20 164 152  93 103 18
14
[281]  53  58 220 282 185  76  71 210 273

$btree
n= 289
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 140 0 (0.51557093 0.48442907)
   2) Salary< 16.95 144   4 0 (0.97222222 0.02777778)
     4) Salary< 15.7 134   0 0 (1.00000000 0.00000000) *
     5) Salary>=15.7 10   4 0 (0.60000000 0.40000000)
      10) Gender=1 6   0 0 (1.00000000 0.00000000) *
      11) Gender=0 4   0 1 (0.00000000 1.00000000) *
   3) Salary>=16.95 145   9 1 (0.06206897 0.93793103)
     6) Distance< 13.35 9   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.35 136   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[21]]
$bindx
  [1]   6 207 106  62  81  19 268  75 166  28 179  25 263  26 117 282  28  43   5 213  49  26  98  81 278 201 2
 [29] 141  42 147 234   3 184  50 237  62  51 136  19 218 137 168 254  64  72  51 204  94 162 180 172 134
 [57]   2 169 288  55 265  10 152  80 151 226 281 225 209 134 170  64 173  37  57  77  98 230 132  25  40 2
 [85] 198  49  49 136 219 143 264 124  13 239 213 173  54  44  54 185 235 232 145 141 264 179 139 246 27
84
[113] 256 257 227  32 183 224 194 260 279 227 284 214  23 192 205  54 260  47  91 199  57  14 154  66 27
164
[141]  96 127 283 195  37 268 251 126 150 215 242 191 241 149 132 212 261  38 126 242 270 140 177  74
253
[169]  30  28  41 240  82 145  64 186  97 183 100 240 186 282  72 249 193 242  17 289 163 177  30   5 113
[197]  40 171 221 195 157  38  94 267  80 203  68 279 163 193 267 163 254  62 204   6 226 100 156  61 230
275
[225] 184  39 160 144  89 128  41 202 273 178 212 128   7  86 111  39   6 232  67 129 103 155  37 285 182
[253] 204 225 162 117 220 143  26 241  29 227 263 286  48 240   1  46 252  18 144 257 177 250  69 237 23
16
[281]  33 186  76  69 207 263 206 102 214

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 141 1 (0.48788927 0.51211073)
  2) Age< 30.43532 135   0 0 (1.00000000 0.00000000) *
  3) Age>=30.43532 154   6 1 (0.03896104 0.96103896)
    6) Distance< 12.25 4   0 0 (1.00000000 0.00000000) *
    7) Distance>=12.25 150   2 1 (0.01333333 0.98666667) *

attr(,"class")
   class
```

"sclass"

$mtrees[[22]]
$bindx
  [1] 251 222 191 165 13 126 53 155 158 243 34 171 31 183 40 274 106 23 209 56 85 51 265 207 277
 [29] 134   6 276   1 253 259 259 280 287 258 95 213 146 152 84 105 106   7 232 179 24   3 150 149 124
 [57] 55 20   7 147 114 127 94 107 59 140   5 129 84 250 99 148 139 64 90 61 228 69 243 167 229 13
 [85] 234 203 164 275 87 213   7 269 103 159 243 53 48 109 46 225 250 287 280 12 214 198 105 264 24
46
[113] 17 239 241 28   5 16 281 58 55 151 176 11 169 125 258   1 219   1 20 106 197 147 268 81 120 7
[141] 129 223 126 282 41 41 102 48 273 120 19 205 112 177 40 79 220 241 73 211 231 132   4 235 25
[169] 100 237 84 126 127 152 184 277   3 240 159 215 183 78 83 172 190 49 78 115 182 161 222 72   4
232
[197] 142 189 242 124 200 33 96 189 254 252 118 129 112 156 164 114 223 56 218 89 222 55 245 51 1
125
[225] 111 102 125 112 282 129 154 76 157 46 272   1 69 272 184 40 66 134   5 225 122 159 68 62 92 2
[253] 188 154 83 145 253 39 199 35 278 276 180 145 83 31 271   6 249 224 152 100 271 274 264 110 23
123
[281] 267 26 274 123 70 260 270 96 196

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 138 0 (0.52249135 0.47750865)
  2) Work.Exp< 9 140   0 0 (1.00000000 0.00000000) *
  3) Work.Exp>=9 149 11 1 (0.07382550 0.92617450)
    6) Distance< 13.5 9   0 0 (1.00000000 0.00000000) *
    7) Distance>=13.5 140   2 1 (0.01428571 0.98571429) *

attr(,"class")
   class
"sclass"

$mtrees[[23]]
$bindx
  [1] 73 201 162 263 49 43 273 256 51 269 234 69 236 131 57 119 30 279 246 24 122 149 36 204 154
 [29] 154 78 198 93 123 207 95 101 202 289 250 122 187 273 281 220 123 182 23 56 166 261 141 175 2
89
 [57] 243 266 192 91 51 136 64 208 155 271 142 133 146 20 203 61 122 92 273 191 177 119 266 64 27
124
 [85] 79 266 135 55 94 241 61 110 120 46 247 188 66 114 195 251 134 84 147 194 183 263 193 144 4
182
[113] 176 81 113 142 269 177 229 101 171 25 149 45 247 39   5 61 274 168 200 253 181 276 165 75 22
28
[141] 286 87 116 249 113 114 82 246 88 218 266 54 84 205 288 287 40 34 234 261 207 104 53 81   2
[169] 103 142 96 152 198 127 31 47 48 162 72 60 210 66 260 247 249 210 105 77 14 200 218 108 20

```
[197]  21  91  47 141 140 210 244  69 209  43  48 185 285 176  52 100 147 191 286 284 166 162 109 273 1
219
[225] 270 170 120  96  21 268 156 107 288  38  70 234  17 283   8  53 121   8 279 129 145  44  76 228 125
[253] 238 107 222 116 165 166 152 199 252  80  61 286 264 202  51 127 148 113 112 110 240 149  46  53
26
[281]  31 215 264  10 251 258  69  95  28

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 289 144 1 (0.49826990 0.50173010)
   2) Age< 30.43512 134   0 0 (1.00000000 0.00000000) *
   3) Age>=30.43512 155  10 1 (0.06451613 0.93548387)
     6) Distance< 13.9 7   0 0 (1.00000000 0.00000000) *
     7) Distance>=13.9 148   3 1 (0.02027027 0.97972973)
      14) license=0 12   3 1 (0.25000000 0.75000000)
        28) Salary< 22.71038 3   0 0 (1.00000000 0.00000000) *
        29) Salary>=22.71038 9   0 1 (0.00000000 1.00000000) *
      15) license=1 136   0 1 (0.00000000 1.00000000) *

attr(,"class")
   class
"sclass"

$mtrees[[24]]
$bindx
  [1] 144 287 202 233 143 208 238  34  55  79 194 245  86 224 128  12 161 170 189 133 133  24  53 225 173
130
 [29] 164 263 171 284 275 126 150 117 286 212 166 288 100 201 265 122 254   9 221 242 222 245  10 258
263 287
 [57]  25 265 248 188 139 130 231 128 108  56 155 156  27  78 195 219 243 108 224  96 265 153  39 184 10
141
 [85] 205  43 247 165  73  66 112 269  42  57  12 211  44 238 260  14 165 121 108  18 123 199   7  82  58 20
[113]  51  83   7 242 202  67 159 280 165 239 261  50 252 259 159  68 143   9 225 107  51  70   1 124 113
[141] 269  15 261 136 130 254  79 258 131 231 144 278 278 174  79 164  90 259 172 177   9 228 241 230 2
133
[169] 102 137  94 255  49 276 141  82 145 230 172 237 238 148  23 179 172 194  76 147 100 104 272 271
217
[197]  61 125 179 258 270 108 142  32 177 253 280 246 152 153 205 206  12  79 164 158   1  32 189 239 12
51
[225]   5  45  46 226 115 247 253  50  46 227 189 224 242 200  25  86  37 269 139 133 183 144  80  46 193
[253] 117  57 126  14 208 107 168 223 126 119 265 209 113  67  23 242  15  32 135 170 263 269  25 279  2
245
[281]  11 282  49  15 124 121 111 187 221

$btree
n= 289
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 137 1 (0.4740484 0.5259516)
  2) Work.Exp< 9 127   0 0 (1.0000000 0.0000000) *
  3) Work.Exp>=9 162  10 1 (0.0617284 0.9382716)
    6) Distance< 13.5 10   0 0 (1.0000000 0.0000000) *
    7) Distance>=13.5 152   0 1 (0.0000000 1.0000000) *

attr(,"class")
   class
"sclass"

$mtrees[[25]]
$bindx
  [1]  47  44 128 279 153  12  41  65   3  12 175  54 178 259 211  19 203  40 227 275  40 264  89  61 248 242
 [29] 285  28 197 112  17  14  27 151 277 104 194 179  88  98  89  99 243  74  26 132  43  86 271  26 142 22
 [57] 193  98  99  32 253 206  10 188 231  63  72  44 140 186  16 184  12 168 190  13 204 191  99 238  36  3
 [85] 228  84  12 122 122  18 188 244 129  22  31  22 220 233  90  76 285  14 167  69  34  10 270 234   5  30
[113]  73  44  88 263 224  87 204 124  66 105 125  37 155  84  52 118 213   5 184  51 113  31  72 259 270  6
[141] 288 179 151 146 217 233 258 233 135 180  59 276 217 203 180 285 127 258 248 108  47 182  21 249
274
[169]  25  65  68  59 144  68 158  22 269 244  80 207 166 198  62 208 259 126 216 118 114 170 222  42 116
252
[197] 196 274   3 254  12 221 123 273 125 198 132 252   9 229 286 205  27  23 157 154  55 114 253  17 134
167
[225] 207 261 209 226 229  74  65 236 126  56  12 164  66 218 169 146 256  29 124  15 214 163 273  55  85
27
[253] 183 275 207  75 168  27  43 215  14 104 143 112 198  18 175 245  29 227 158 119  91 111   7 138 149
192
[281] 188 224 174 120  84 261 175 225 130

$btree
n= 289

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 289 135 0 (0.53287197 0.46712803)
  2) Salary< 15.7 139   0 0 (1.00000000 0.00000000) *
  3) Salary>=15.7 150  15 1 (0.10000000 0.90000000)
    6) Distance< 13.9 13   0 0 (1.00000000 0.00000000) *
    7) Distance>=13.9 137   2 1 (0.01459854 0.98540146) *

attr(,"class")
   class
"sclass"
```

```
$OOB
[1] TRUE

$comb
[1] FALSE

$err
[1] 0.01384083

$call
bagging.data.frame(formula = Transport ~ ., data = s.train, coob = TRUE,
   control = rpart.control(maxdepth = 10, minsplit = 3))

attr(,"class")
[1] "summary.bagging"
> ## Making predictions with Bagging model ####
> set.seed(77)
> s.bag.pred = predict(bag.model,s.test,type = "class")
> bag.pred = as.factor(s.bag.pred)
> ## Confusion Matrix
> bag.c = caret::confusionMatrix(s.test$Transport,bag.pred,positive = "1")
> print(bag.c)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 112   3
         1   0  11

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.8889
    P-Value [Acc > NIR] : 0.0002782

                  Kappa : 0.867

 Mcnemar's Test P-Value : 0.2482131

            Sensitivity : 0.7857
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9739
             Prevalence : 0.1111
         Detection Rate : 0.0873
   Detection Prevalence : 0.0873
      Balanced Accuracy : 0.8929

       'Positive' Class : 1

> bag.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
          0.78571429           1.00000000           1.00000000           0.97391304
           Precision               Recall                   F1           Prevalence
          1.00000000           0.78571429           0.88000000           0.11111111
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.08730159           0.08730159           0.89285714
> plot(s.test$Transport,bag.pred,xlab = "Actuals",ylab = "Predicted",
```

```
+      col = c("White","Red"))
> ### Boosting after SMOTE ####
> ### Ada boost ####
> s.test$Transport = as.numeric(s.test$Transport)
> str(s.train)
'data.frame':  289 obs. of  9 variables:
 $ Age      : num  25 34 24 22 28 29 25 27 24 26 ...
 $ Gender   : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 2 2 2 ...
 $ Engineer : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 2 1 2 2 ...
 $ MBA      : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 2 2 1 ...
 $ Work.Exp : num  5 12 6 1 5 5 4 8 0 4 ...
 $ Salary   : num  13.7 16.9 11.6 7.5 14.6 14.8 11.5 15.6 7.6 12.8 ...
 $ Distance : num  12.7 16.6 11.3 5.1 7.2 15.4 7 9 9.5 11.6 ...
 $ license  : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 1 1 1 1 ...
 $ Transport: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> s.test$Transport = as.factor(s.test$Transport)
> s.train$Transport = as.factor(s.train$Transport)
> set.seed(77)
> s.gbm <- gbm(
+         formula = Transport~.,
+         data = s.train,
+         distribution = "multinomial",
+         n.trees = 100,
+         interaction.depth = 1,
+         shrinkage = 0.1,
+         cv.folds = 2,
+         n.cores = 2, # will use all cores by default
+         verbose = FALSE
+ )
> summary(s.gbm)
                var      rel.inf
Salary     Salary 35.298959403
Work.Exp Work.Exp 32.745988200
Distance Distance 22.645360426
license   license  6.345950201
Age           Age  2.956551766
Engineer Engineer  0.007190004
Gender     Gender  0.000000000
MBA           MBA  0.000000000
> ## Predictions for Adaboost model ###
> set.seed(77)
> prob.s.gbm = predict(s.gbm,s.test,type = "response")
Using 99 trees...
> prob.s.gbm = as.data.frame(prob.s.gbm)
> qplot(prob.s.gbm[,c(1)],s.test$Transport,geom = "boxplot")
> pred.s.gbm = ifelse(prob.s.gbm[,c(1)] > 0.50,"0","1")
> pred.s.gbm = as.factor(pred.s.gbm)
> s.test = m.test
> ## Confusion Matrix ###
> adb.c = caret::confusionMatrix(s.test$Transport,pred.s.gbm,positive = "1")
> print(adb.c)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 113   2
         1   1  10

               Accuracy : 0.9762
                 95% CI : (0.932, 0.9951)
    No Information Rate : 0.9048
```

```
    P-Value [Acc > NIR] : 0.001606

                  Kappa : 0.8565

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.83333
            Specificity : 0.99123
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.98261
             Prevalence : 0.09524
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.91228

       'Positive' Class : 1

> adb.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
           0.83333333           0.99122807           0.90909091           0.98260870
            Precision               Recall                   F1           Prevalence
           0.90909091           0.83333333           0.86956522           0.09523810
      Detection Rate Detection Prevalence    Balanced Accuracy
           0.07936508           0.08730159           0.91228070
> plot(s.test$Transport,pred.s.gbm,xlab = "Actuals",ylab = "Predicted",
+      col = c("Blue","Yellow"))
> ### XG BOOSTING ####
> set.seed(77)
> f = model.matrix(s.train$Transport~.,s.train)
>
> t = s.train[,9]
> t = as.matrix(t)
> t = as.character(t)
> t = as.numeric(t)
> xg.train.m = xgb.DMatrix(data = f,label = t)
>
> e = model.matrix(s.test$Transport~.,s.test)
> s = s.test[,9]
> s = as.matrix(s)
> s = as.character(s)
> s = as.numeric(s)
> xg.test.m = xgb.DMatrix(data = e,label = s)
> set.seed(77)
> s.xgb <- xgboost(
+   data = xg.train.m,
+   eta = 0.5,
+   max_depth = 5,
+   nrounds = 2,
+   nfold = 2,
+   objective = "binary:logistic",  # for regression models
+   verbose = 0,                # silent,
+   early_stopping_rounds = 10 # stop if no improvement for 10 consecutive trees
+ )
> summary(s.xgb)
                Length Class              Mode
handle             1   xgb.Booster.handle externalptr
raw             1453   -none-             raw
best_iteration     1   -none-             numeric
best_ntreelimit    1   -none-             numeric
best_score         1   -none-             numeric
niter              1   -none-             numeric
```

```
evaluation_log     2   data.table         list
call              17   -none-             call
params             5   -none-             list
callbacks          2   -none-             list
feature_names      9   -none-             character
nfeatures          1   -none-             numeric
> ## Predictions for XGboost ###
> xgb.predict = predict(s.xgb,xg.test.m)
> bp = qplot(xgb.predict,s.test$Transport,geom = "boxplot"
+       ,xlab = "Probabilities",ylab = "Transport",xintercept = 0.63)
> bp+geom_vline(xintercept = 0.6126,color = "Red",size = 1.5)
> xgb.predict.c = ifelse(xgb.predict > 0.6126,"1","0")
> xgb.predict.c = as.factor(xgb.predict.c)
> ## Confusion Matrix ###
> xgb.c = caret::confusionMatrix(s.test$Transport,xgb.predict.c,positive = "1")
> print(xgb.c)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 114   1
         1   1  10

               Accuracy : 0.9841
                 95% CI : (0.9438, 0.9981)
    No Information Rate : 0.9127
    P-Value [Acc > NIR] : 0.0008535

                  Kappa : 0.9004

 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.90909
            Specificity : 0.99130
         Pos Pred Value : 0.90909
         Neg Pred Value : 0.99130
             Prevalence : 0.08730
         Detection Rate : 0.07937
   Detection Prevalence : 0.08730
      Balanced Accuracy : 0.95020

       'Positive' Class : 1

> xgb.c$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
          0.90909091           0.99130435           0.90909091           0.99130435
           Precision               Recall                   F1           Prevalence
          0.90909091           0.90909091           0.90909091           0.08730159
      Detection Rate Detection Prevalence    Balanced Accuracy
          0.07936508           0.08730159           0.95019763
> plot(s.test$Transport,xgb.predict.c,xlab = "Actuals",ylab = "Predicted",
+      col = c("Yellow","Red"))
```