Mini Project:5
Machine Learning

Table of Contents

Machine Learning – Project 5

Suprasanna Pradhan
PGPBABI-Dec2018

1. Project Objective and Scope

1.1 Objective

This project requires to understand what mode of transport employees prefers to commute to their office. The attached data 'Cars.csv, includes employees' information about their mode of transport as well as their personal and professional details like age, salary, work exp. We need to predict whether an employee will use Car as a mode of transport. Also, which variables are a significant predictor behind this decision.

1.2 Scope

We will create multiple models and explore how each model perform using appropriate model performance metrics

1. KNN
2. Naive Bayes (is it applicable here? comment and if it is not applicable, how we can build an NB model in this case?)
3. Logistic Regression

Apply both bagging and boosting modeling procedures to create 2 models and compare its accuracy with the best model of the above step.

2 . Project Approach

A typical Development Lifecycle can be adopted for this assignment, as follows:

- Data Preparation
- Model Planning
- Model Building
- Communicating Results.

3. Data Preparation

3.1 Exploratory Data Analysis

3.1.1 Importing Data

```
##Importing Data Set##
cars <- read.csv("C:/Users/SuprasannaPradhan/Documents/My Files/Great Lakes Proje
cts/cars.csv", header= TRUE)
```

3.1.2 Converting Variables

```
#Setting outcome variables as categorical
cars$Gender_Dt<-ifelse(cars$Gender=="Male",1,0)
cars_data <- as.data.frame(cars)
```

3.1.3 Key Observations

The number of columns (Features) in the dataset are 9 and 418 observations existed .

Variables	Levels		
Age	Numbers		
Gender	Male	Female	
Engineer	0	1	
MBA	0	1	
Work Exp	Frequency		
Salary	Frequency		
Distance	Frequency		
license	0	1	
Transport	2Wheeler	Car	Public Transport

Transport is having three classes whereas gender is having two classes , License ,MBA and Engineers has got also two classes

Going further we must be setting outcome variables as categorical for all these above-mentioned variables . Observed there is three variables Work Experience ,Salary and Distance is consisting of continuous frequency .

License variable may not be very helpful to us to analyze the data set , but still we will keep it till checking multicollinearity.

```r
str(cars_data)
```

```
## 'data.frame':    418 obs. of  10 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 ...
##  $ Gender_Dt: num  1 1 0 1 0 1 1 1 1 1 ...
```

```r
head(cars_data)
```

```
##   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
##   Gender_Dt
## 1         1
## 2         1
## 3         0
## 4         1
## 5         0
## 6         1
```

```r
describe(cars_data)
```

```
##            vars   n  mean   sd median trimmed  mad  min  max range  skew
## Age           1 418 27.33 4.15   27.0   26.89 2.97 18.0 43.0  25.0  1.09
## Gender*       2 418  1.71 0.45    2.0    1.76 0.00  1.0  2.0   1.0 -0.93
## Engineer      3 418  0.75 0.43    1.0    0.81 0.00  0.0  1.0   1.0 -1.14
## MBA           4 417  0.26 0.44    0.0    0.20 0.00  0.0  1.0   1.0  1.08
## Work.Exp      5 418  5.87 4.82    5.0    5.12 2.97  0.0 24.0  24.0  1.52
## Salary        6 418 15.42 9.66   13.0   13.22 4.15  6.5 57.0  50.5  2.28
## Distance      7 418 11.29 3.70   10.9   11.08 3.56  3.2 23.4  20.2  0.55
## license       8 418  0.20 0.40    0.0    0.13 0.00  0.0  1.0   1.0  1.47
## Transport*    9 418  2.52 0.81    3.0    2.65 0.00  1.0  3.0   2.0 -1.20
## Gender_Dt    10 418  0.71 0.45    1.0    0.76 0.00  0.0  1.0   1.0 -0.93
##            kurtosis   se
## Age            1.67 0.20
## Gender*       -1.15 0.02
## Engineer      -0.69 0.02
## MBA           -0.83 0.02
## Work.Exp       2.29 0.24
## Salary         4.82 0.47
```

```
## Distance       0.05 0.18
## license        0.16 0.02
## Transport*    -0.38 0.04
## Gender_Dt     -1.15 0.02
```

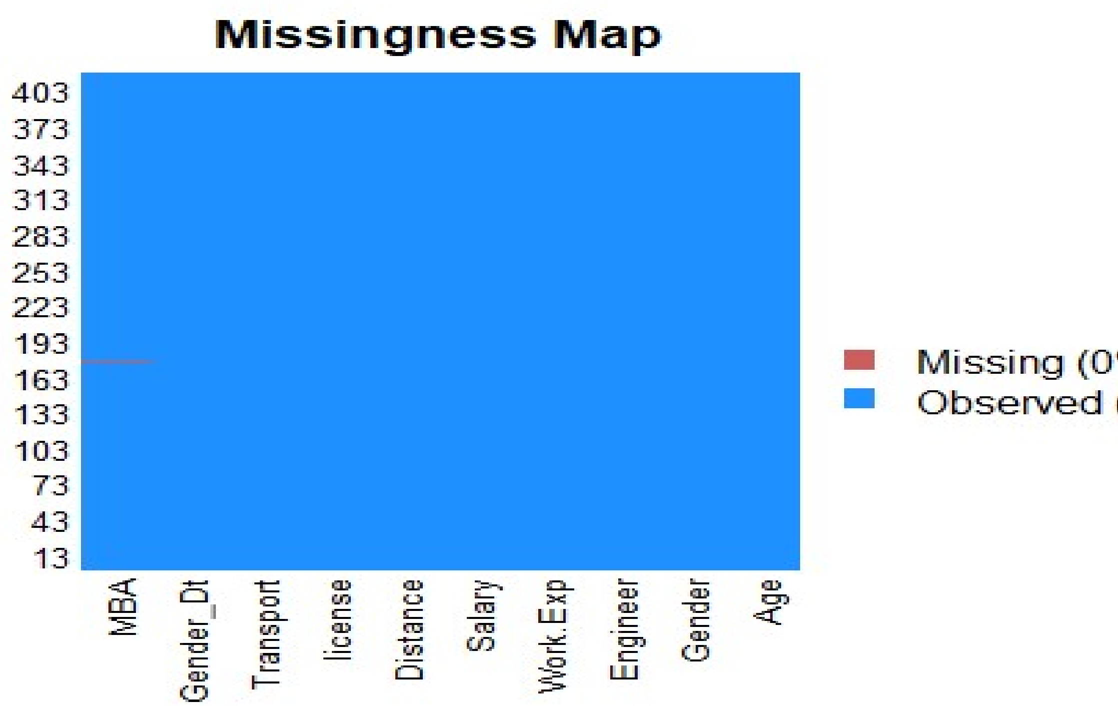
From Age, Work Experience, Distance and Salary , Age is carrying out the maxim average and Salary is maxim SD values

3.1.4 Checking Missing Values

While analyzing the structure of the data set, we found some variable does not have minimum values or zero values . This is not ideal since no variable should missing values Therefore, such values are treated as missing observations.

In the below code snippet, we're setting the zero values to NA's:

```
#visualize the missing data
missmap(cars_data)
```




```
sum(is.na(cars_data))

## [1] 1

cars_data[is.na(cars_data)] <- 0
sum(is.na(cars_data))

## [1] 0
```

The above illustrations show that our data set has only one missing values.

4. Model Planning

Before we study the data, set let's convert the Transport variable into a categorical variable. This is necessary because our transport will be in the form of 3 classes - Car, Two-Wheeler and Public Transport., . Where true will denote that employees uses the mode of the transport.

To analyze further we have been creating separate dummy variable to all these three levels with separate column in the existed data set .

```
levels(cars_data$Transport)

## [1] "2Wheeler"         "Car"              "Public Transport"

cars_data$Car_Dt<-ifelse(cars_data$Transport=="Car",1,0)
cars_data$TW_Dt<-ifelse(cars_data$Transport=="2Wheeler",1,0)
cars_data$Pub_tran<-ifelse(cars_data$Transport=="Public.Transport",1,0)
```

4.1 Data Visualization

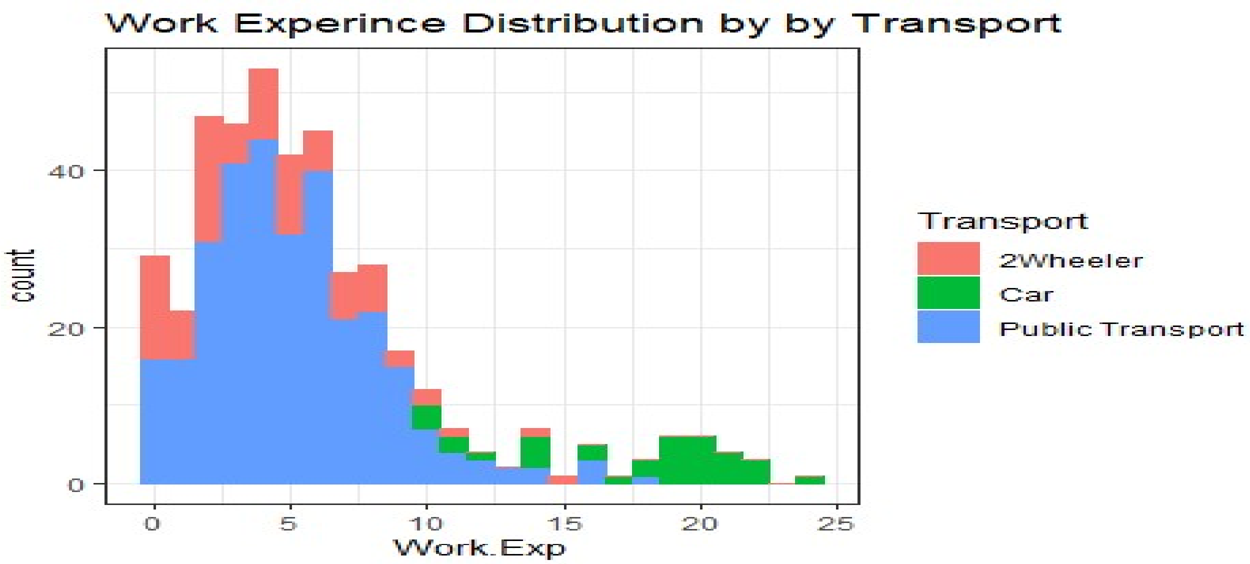
Now let's perform a couple of visualizations to take a better look at each variable, this stage is essential to understand the significance of each predictor variable.

4.1.1 Age Distribution by Transport

```
#visual 1
library(ggplot2)
library(dplyr)
ggplot(cars_data, aes(x=Age, colour = Transport)) +
  geom_freqpoly(binwidth = 1)+ labs(title="Age Distribution by Transport")
```

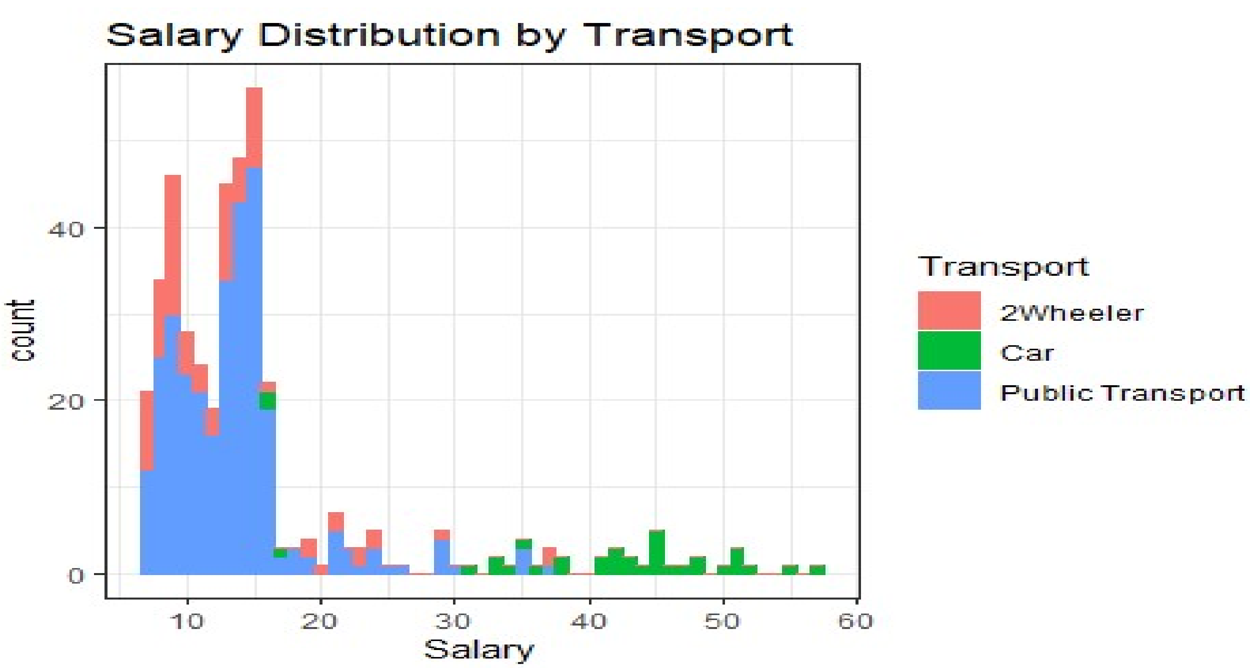
4.1.2 Work Experience Distribution by Transport

```
#visual 2
c <- ggplot(cars_data, aes(x=Work.Exp, fill=Transport, color=Transport)) +
  geom_histogram(binwidth = 1) + labs(title="Work Experince Distribution by by Tr
ansport")
c + theme_bw()
```



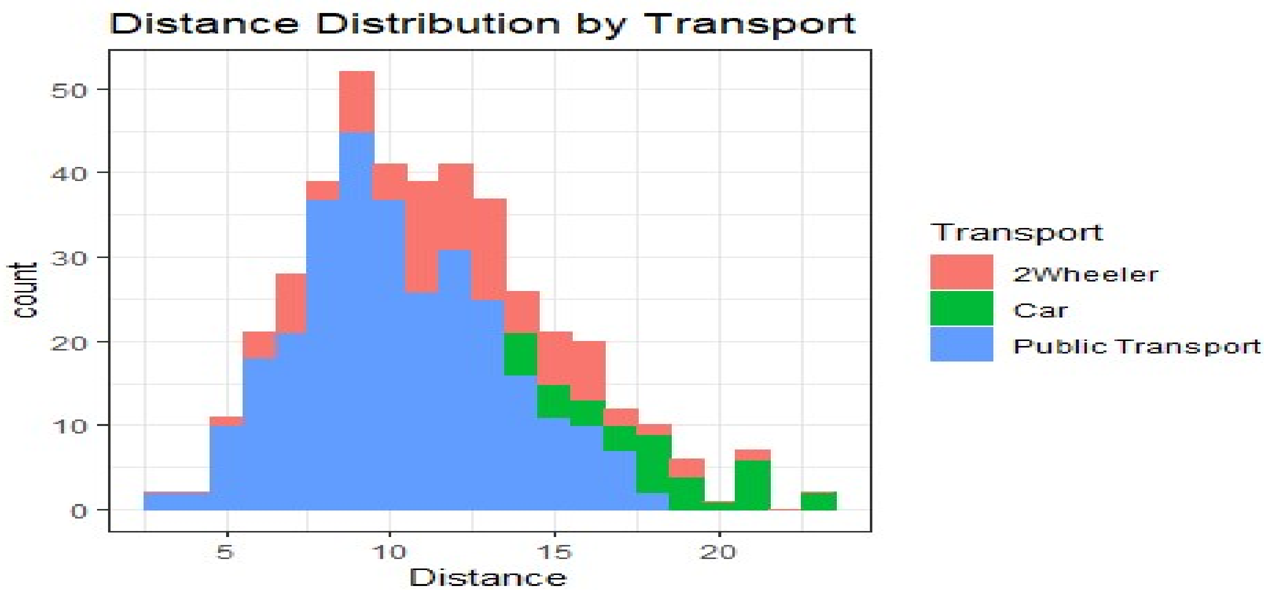
4.1.3 Salary Distribution by Transport

```
#visual 3
s <- ggplot(cars_data, aes(x=Salary, fill=Transport, color=Transport)) +
  geom_histogram(binwidth = 1) + labs(title="Salary Distribution by Transport")
s + theme_bw()
```



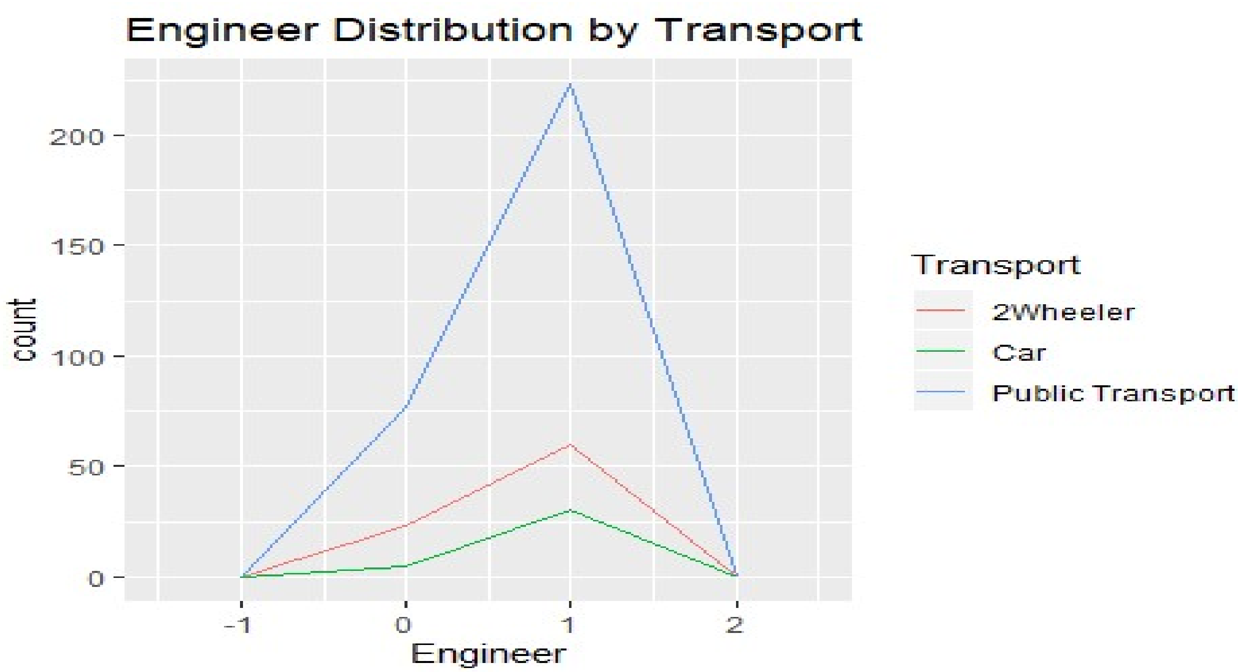
4.1.4 Distance Distribution by Transport

```
#visual 4
d <- ggplot(cars_data, aes(x=Distance, fill=Transport, color=Transport)) +
  geom_histogram(binwidth = 1) + labs(title="Distance Distribution by Transport")
d + theme_bw()
```



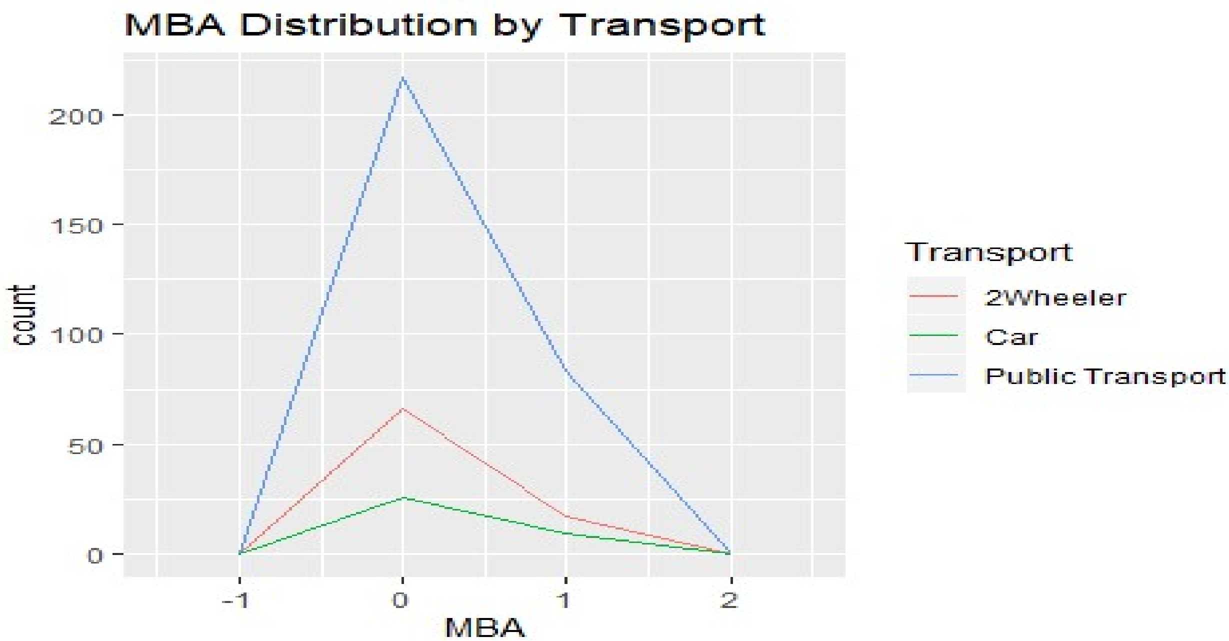
4.1.5 Engineer Distribution by Transport

```
#visual 5
ggplot(cars_data, aes(Engineer, colour = Transport)) +
  geom_freqpoly(binwidth = 1) + labs(title="Engineer Distribution by Transport")
```



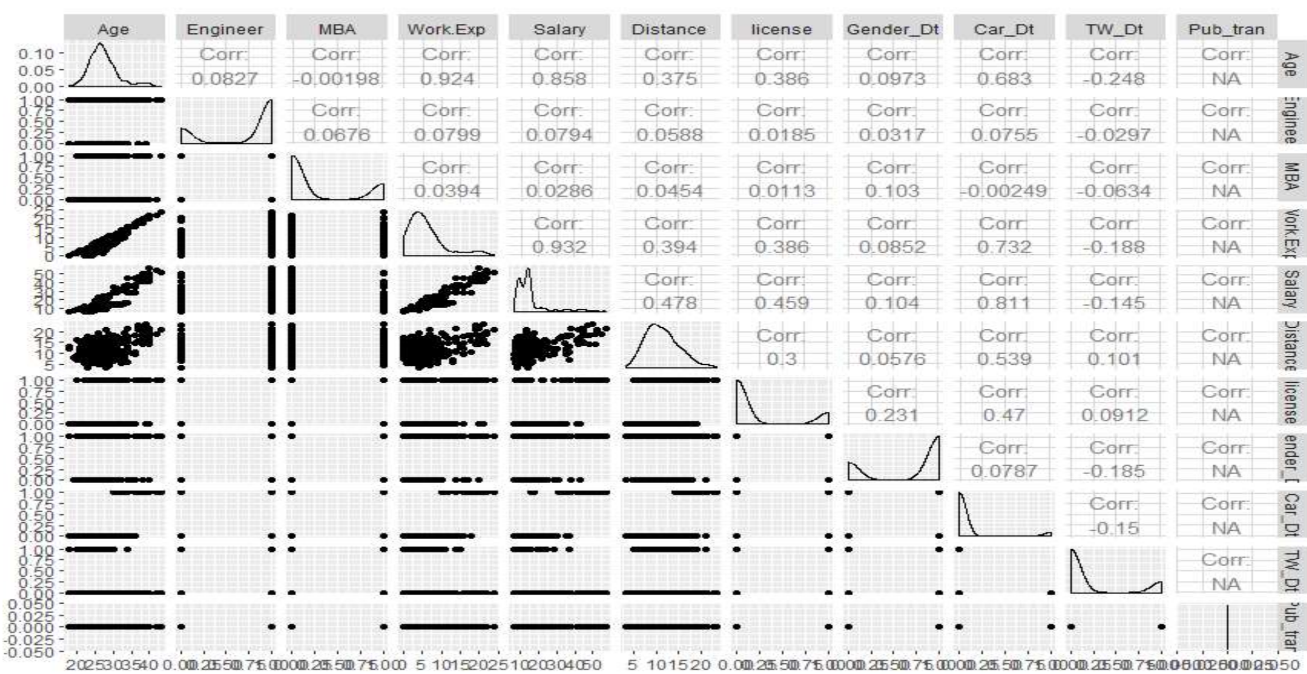
4.1.6 MBA Distribution by Transport

```
#visual 6
ggplot(cars_data, aes(MBA, colour = Transport)) +
  geom_freqpoly(binwidth = 1) + labs(title="MBA Distribution by Transport")
```



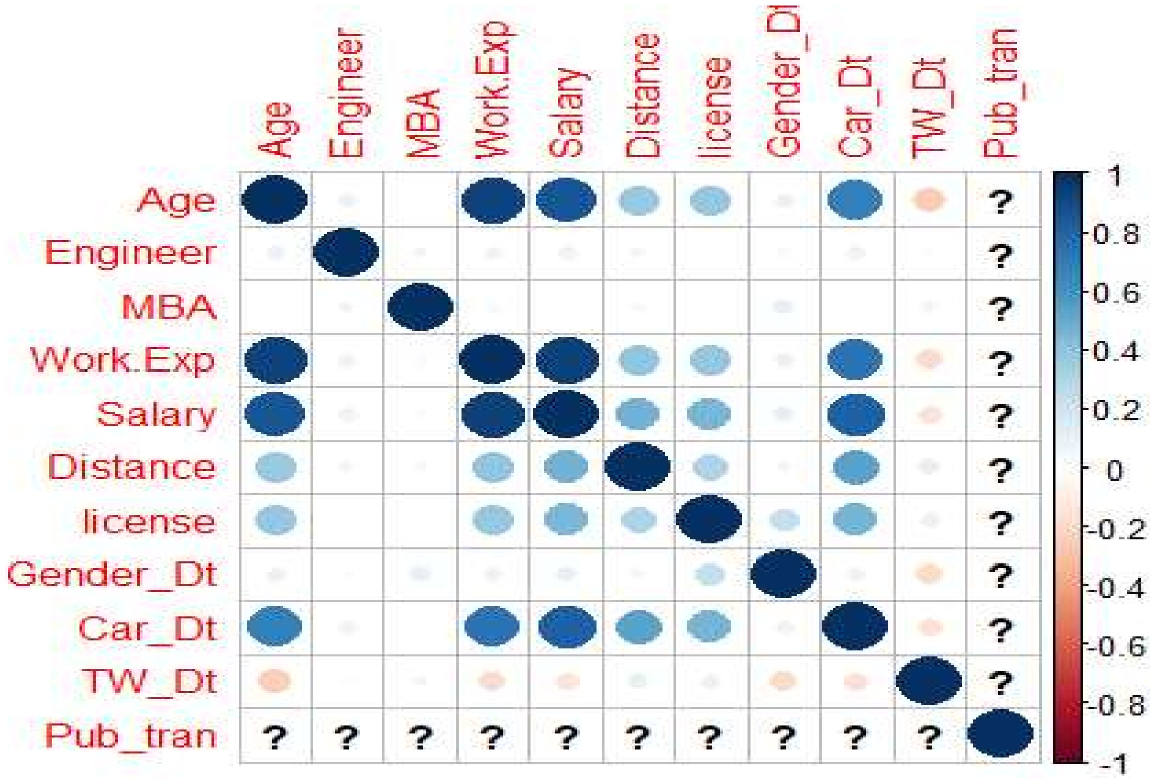
4.1.7 Ggpairs plot

```
#visual 5
cars_data <-  cars_data %>% select (-Gender,-Transport)
ggpairs(cars_data)
```

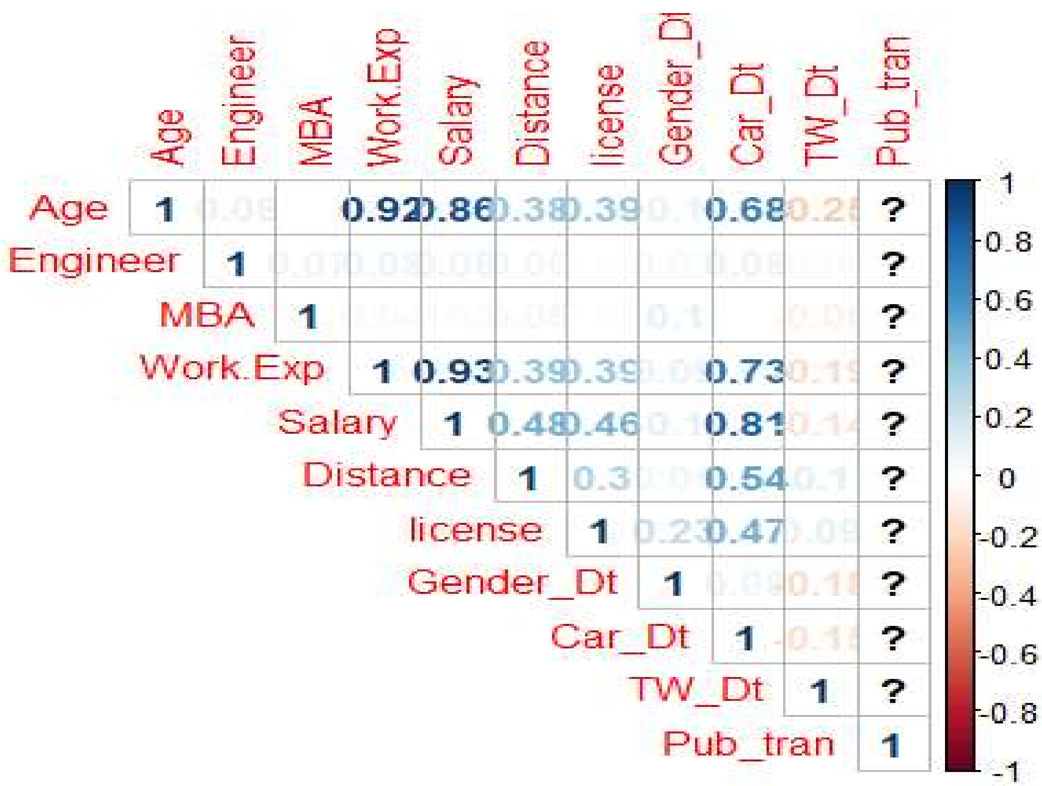


4.2 Checking Multicollinearity

```
#Checking cor plot#
cars_data1 = as.data.frame(cars_data)
cars_matrix = cor(cars_data1)

## Warning in cor(cars_data1): the standard deviation is zero

str(cars_data1 )

## 'data.frame':    418 obs. of  11 variables:
##  $ Age      : int  28 24 27 25 25 21 23 23 24 28 ...
##  $ Engineer : int  1 1 1 0 0 0 1 0 1 1 ...
##  $ MBA      : num  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 ...
##  $ Gender_Dt: num  1 1 0 1 0 1 1 1 1 1 ...
##  $ Car_Dt   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TW_Dt    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Pub_tran : num  0 0 0 0 0 0 0 0 0 0 ...

library(corrplot)

## corrplot 0.84 loaded

corrplot(cars_matrix)
```




```
corrplot(cars_matrix, type="upper", method="number")
```



We have checked Variable like **Work Experience**, **Salary and Cars** is having highly correlated

4.2 Preparation of module and Multicollinearity check

We shall check Regression to find the variable which is significant each other

In the new data module m1 is being performed the coefficient values found Salary, Distance, license and two wheeler variables are significantly correlated with Cars , hence this is sign of multicollinearity.

The R square value is 70, The highest coefficient rate is carried out by Salary.

```
ml <- lm(Car_Dt ~., data=cars_data1)
summary(ml)

##
## Call:
## lm(formula = Car_Dt ~ ., data = cars_data1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50653 -0.06726 -0.00254  0.06402  0.86680
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.313658   0.114882  -2.730 0.006602 **
## Age         -0.002149   0.004877  -0.441 0.659710
## Engineer     0.007441   0.017210   0.432 0.665719
## MBA         -0.021593   0.017189  -1.256 0.209761
## Work.Exp    -0.004347   0.005904  -0.736 0.461988
## Salary       0.021275   0.002330   9.132  < 2e-16 ***
## Distance     0.015125   0.002369   6.384 4.71e-10 ***
## license      0.090510   0.022053   4.104 4.90e-05 ***
## Gender_Dt   -0.028954   0.017309  -1.673 0.095136
## TW_Dt       -0.075182   0.020594  -3.651 0.000296 ***
## Pub_tran           NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1516 on 408 degrees of freedom
## Multiple R-squared:  0.7076, Adjusted R-squared:  0.7012
## F-statistic: 109.7 on 9 and 408 DF,  p-value: < 2.2e-16
```

4.2.2 Bartlett's Test:

We prepared new set of data without the transport(car, 2wheeler,public transport) variable to analyze further where we processed tested Bartlett's tested and KMO

We understand Bartlett's test provides a chi-square output that must be significant. It indicates the matrix is not an identity matrix and accordingly it should be significant (p<.05) . In our case we have seen the P value is less than 0.05 of the significance level indicate that a factor analysis may be useful with this data set.

```r
data2 = subset(cars_data1, select = -c(9:11))
data4 = subset(cars_data1, select = c(9:11))
cormatrix = cor(data2)
str(data2)

## 'data.frame':    418 obs. of  8 variables:
##  $ Age      : int  28 24 27 25 25 21 23 23 24 28 ...
##  $ Engineer : int  1 1 1 0 0 0 1 0 1 1 ...
##  $ MBA      : num  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 ...
##  $ Gender_Dt: num  1 1 0 1 0 1 1 1 1 1 ...

library(psych)
cortest.bartlett(cormatrix,100)

## $chisq
## [1] 440.0267
##
## $p.value
## [1] 1.359223e-75
##
## $df
## [1] 28
```

4.2.3 KMO Test

We have checked Kaiser-Meyer-Olkin (KMO) to find the Test for Sampling Adequacy whereas the values in this case is greater than .5 , hence the data set is occurred with enough samples

```r
KMO(cormatrix)

## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = cormatrix)
## Overall MSA =  0.75
## MSA for each item =
##       Age  Engineer       MBA  Work.Exp    Salary  Distance   license
##      0.80      0.80      0.32      0.67      0.75      0.84      0.83
## Gender_Dt
##      0.63
```

4.2.4 Check Eigen Value

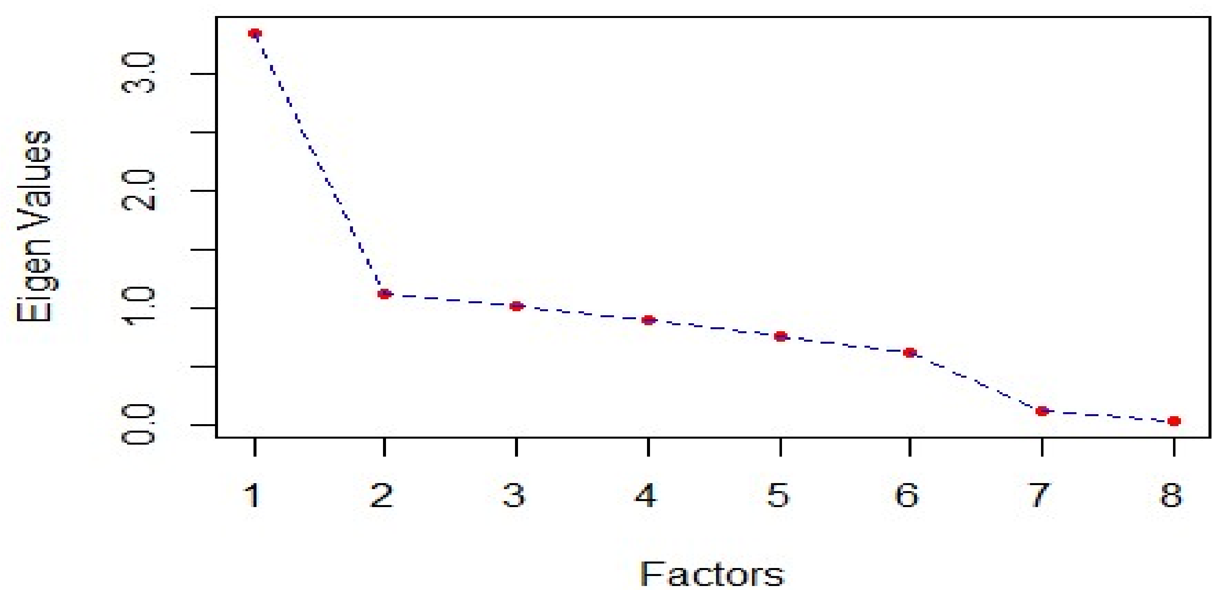
We found there two factors are >1 Eigenvalue above 1 is among the least accurate methods for selecting the number of factors to retain. The number of factors to rotate is the eigenvalues-greater-than-one rule proposed by Kaiser (1960).

It states that there are as many reliable factors as there are eigenvalues greater than one. Eigenvalue less than one implies that the scores on the component would have negative reliability.

4.2.5 Performing Factor Analysis Extracting Four Factors

We have checked without rotate these factors and plotted them

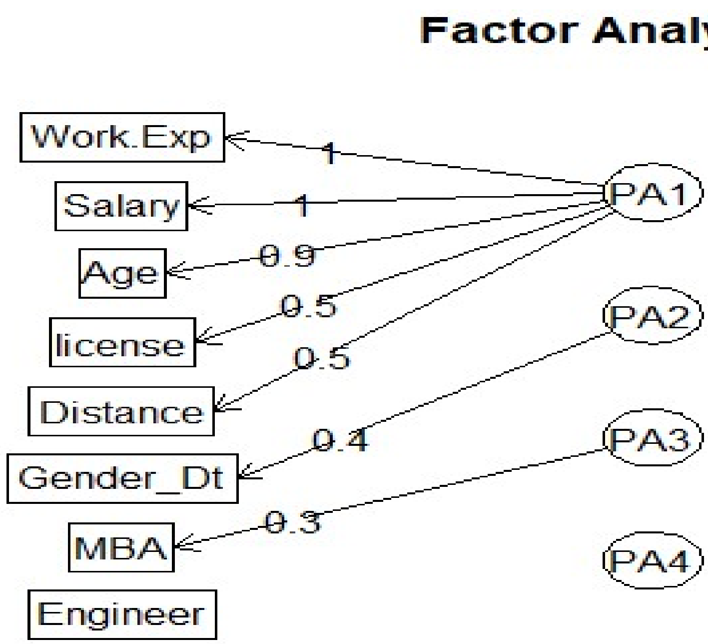
```
#Check eigen values
evector = eigen(cormatrix)
eigen_value = evector$values
eigen_value

## [1] 3.35411628 1.13551068 1.01948784 0.90791568 0.77433856 0.63243008
## [7] 0.13263566 0.04356523

plot(eigen_value, xlab = "Factors", ylab = "Eigen Values", col="red", pch=20)
lines(eigen_value, col="blue", lty = 2)
```




```
fa1 = fa(r= data2, nfactors =4, rotate ="none", fm ="pa")

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : An ultra-Heywood case was detected. Examine the results carefully

print(fa1)

## Factor Analysis using method =  pa
## Call: fa(r = data2, nfactors = 4, rotate = "none", fm = "pa")
## Standardized loadings (pattern matrix) based upon correlation matrix
##            PA1   PA2   PA3   PA4    h2     u2 com
## Age       0.90 -0.14  0.05 -0.10 0.849  0.151 1.1
## Engineer  0.09  0.03  0.12  0.11 0.035  0.965 2.9
```

```
## MBA       0.04  0.17  0.31  0.20 0.169  0.831 2.4
## Work.Exp  0.98 -0.19  0.13 -0.06 1.017 -0.017 1.1
## Salary    0.96 -0.03 -0.05  0.07 0.920  0.080 1.0
## Distance  0.48  0.14 -0.17  0.29 0.361  0.639 2.2
## license   0.49  0.41 -0.19 -0.09 0.452  0.548 2.3
## Gender_Dt 0.14  0.44  0.17 -0.15 0.267  0.733 1.8
##
##                        PA1  PA2  PA3  PA4
## SS loadings           3.19 0.47 0.23 0.19
## Proportion Var        0.40 0.06 0.03 0.02
## Cumulative Var        0.40 0.46 0.49 0.51
## Proportion Explained  0.78 0.11 0.06 0.05
## Cumulative Proportion 0.78 0.90 0.95 1.00
##
## Mean item complexity =  1.9
## Test of the hypothesis that 4 factors are sufficient.
##
## The degrees of freedom for the null model are  28  and the objective function
was  4.61 with Chi Square of  1905.25
## The degrees of freedom for the model are 2  and the objective function was  0.
01
##
## The root mean square of the residuals (RMSR) is  0
## The df corrected root mean square of the residuals is  0.01
##
## The harmonic number of observations is  418 with the empirical chi square  0.3
7  with prob <  0.83
## The total number of observations was  418  with Likelihood Chi Square =  3.2
with prob <  0.2
##
## Tucker Lewis Index of factoring reliability =  0.991
## RMSEA index =  0.039  and the 90 % confidence intervals are  0 0.112
## BIC =  -8.87
## Fit based upon off diagonal values = 1
## Measures of factor score adequacy
##                                                    PA1  PA2   PA3   PA4
## Correlation of (regression) scores with factors   0.99 0.73  0.66  0.55
## Multiple R square of scores with factors          0.99 0.53  0.44  0.30
## Minimum correlation of possible factor scores     0.97 0.07 -0.12 -0.40
```

```
fa.diagram(fa1)
```

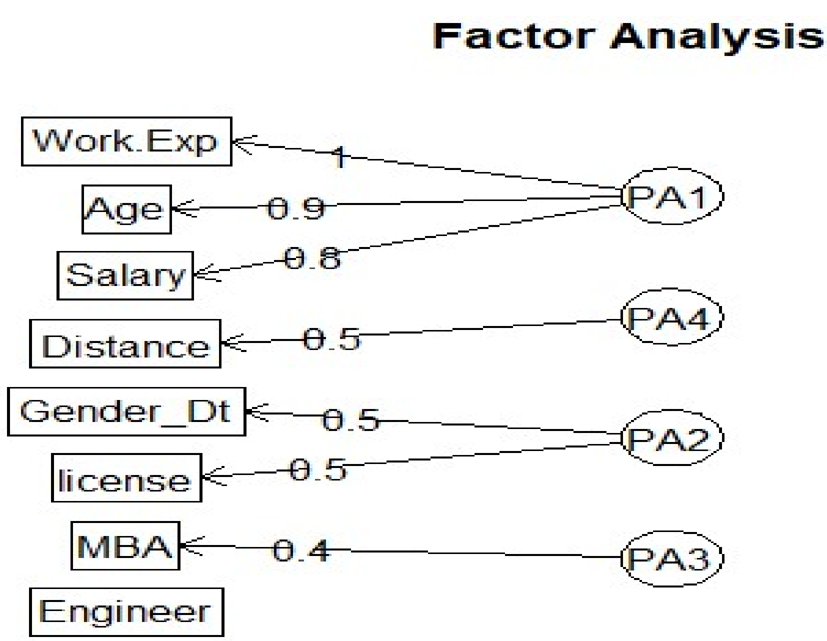


4.2.6 Factors Rotated

```
fa2 = fa(r= data2, nfactors = 4
         , rotate ="varimax", fm ="pa")

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs
## = np.obs, : The estimated weights for the factor scores are probably
## incorrect. Try a different factor extraction method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : An ultra-Heywood case was detected. Examine the results carefully

print(fa2)

## Factor Analysis using method =  pa
## Call: fa(r = data2, nfactors = 4, rotate = "varimax", fm = "pa")
## Standardized loadings (pattern matrix) based upon correlation matrix
##             PA1  PA4  PA2   PA3    h2     u2 com
## Age        0.89 0.20 0.12  0.05 0.849  0.151 1.1
## Engineer   0.06 0.03 0.01  0.17 0.035  0.965 1.3
## MBA       -0.03 0.00 0.09  0.40 0.169  0.831 1.1
## Work.Exp   0.98 0.18 0.08  0.14 1.017 -0.017 1.1
## Salary     0.85 0.42 0.12  0.10 0.920  0.080 1.5
## Distance   0.29 0.51 0.05  0.12 0.361  0.639 1.8
## license    0.29 0.39 0.46 -0.06 0.452  0.548 2.7
## Gender_Dt  0.03 0.01 0.50  0.14 0.267  0.733 1.2
##
##                        PA1  PA4  PA2  PA3
## SS loadings           2.65 0.66 0.50 0.26
## Proportion Var        0.33 0.08 0.06 0.03
```

```
## Cumulative Var        0.33 0.41 0.48 0.51
## Proportion Explained  0.65 0.16 0.12 0.06
## Cumulative Proportion 0.65 0.81 0.94 1.00
##
## Mean item complexity =  1.5
## Test of the hypothesis that 4 factors are sufficient.
##
## The degrees of freedom for the null model are  28  and the objective function
was  4.61 with Chi Square of  1905.25
## The degrees of freedom for the model are 2  and the objective function was  0.
01
##
## The root mean square of the residuals (RMSR) is  0
## The df corrected root mean square of the residuals is  0.01
##
## The harmonic number of observations is  418 with the empirical chi square  0.3
7  with prob <  0.83
## The total number of observations was  418  with Likelihood Chi Square =  3.2
with prob <  0.2
##
## Tucker Lewis Index of factoring reliability =  0.991
## RMSEA index =  0.039  and the 90 % confidence intervals are  0 0.112
## BIC =  -8.87
## Fit based upon off diagonal values = 1
```

```
fa.diagram(fa2)
```



We have changed factor names to PA1, PA2 ,PA3,PA4

Professional
Distance
License
MBA

```
Carsdata_Mic = cbind(data4,fa2$scores)
head(Carsdata_Mic)

##   Car_Dt TW_Dt Pub_tran        PA1        PA4         PA2         PA3
## 1      0     1        0 -0.1315237 -0.4467651  0.20259100 -0.34126862
## 2      0     1        0  0.4483541 -1.5530375 -0.08023245  0.32422828
## 3      0     1        0  1.1544105 -1.4364795 -0.99503567  0.05182746
## 4      0     1        0 -1.0069960 -0.2519568  0.28808071 -0.74157021
## 5      0     1        0 -0.4101696 -0.3982738 -0.72322719 -0.77508798
## 6      0     1        0 -0.3557918 -0.7985349 -0.03510892 -0.05420258

colnames(Carsdata_Mic) = c("Cars", "2Whleers","PublicTran", "Professional", "Dist
ance ","License","MBA")
str(Carsdata_Mic)

## 'data.frame':    418 obs. of  7 variables:
##  $ Cars        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ 2Whleers    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ PublicTran  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Professional: num  -0.132 0.448 1.154 -1.007 -0.41 ...
##  $ Distance    : num  -0.447 -1.553 -1.436 -0.252 -0.398 ...
##  $ License     : num  0.2026 -0.0802 -0.995 0.2881 -0.7232 ...
##  $ MBA         : num  -0.3413 0.3242 0.0518 -0.7416 -0.7751 ...
```

We have renamed variables are attached in our earlier data set where Transport class are lying with separate variable , so the data set going to be comprised these variables

Cars ,2Whleers, PublicTran, Professional, Distance, License, MBA

5. Model Building

Our final data is now ready to prepare the modules

Splitting the dataset into the Training set and Test set

```
> # Splitting the dataset into the Training set and Test set

> # install.packages('caTools')

> library(caTools)

> library(dplyr)

> table(Carsdata_Mic$Cars)
```

```
  0   1 
383 35 
> set.seed(108)

> split = sample.split(Carsdata_Mic$Cars, SplitRatio = 0.75)

> training_set = subset(Carsdata_Mic, split == TRUE)

> test_set = subset(Carsdata_Mic, split == FALSE)

> table(training_set$Cars)

  0   1 
287  26 
> table(test_set$Cars)

 0  1 
96  9 
```

In Above illustration we got the main data set is consist of 418 observation with 7 variables Since we target classify Cars variable found it has been having 383 false cases and 35 cases are true .

We have splinted with 75% of ratio into train and test set.

```
Base ratio: 383/35 =  10.94286
Training set ratio: 287/6 = 47.83333
Testing set ration:96/9 = 10.66667
```

5.1 KNN Classifier

```r
library(class)
knn_fit<- knn(train = training_set[-1], test = test_set[-1], cl=training_set$Cars
,k =3,prob=TRUE)
knn_chk= table(test_set$Cars,knn_fit)
knn_chk

##    knn_fit
##      0  1
##   0 94  2
##   1  2  7

accuracy.knn = sum(diag(knn_chk))/sum(knn_chk)*100
accuracy.knn

## [1] 96.19048
```

KNN Algorithm accuracy print It prints accuracy of our knn model. Here our accuracy is 98.80%. That's pretty good ???? for our randomly selected dummy dataset.

5.2 Naive Bays

5.2.1 Why we will use here Naive Bays ?

Naive Bayes is used to solve classification problems by following a probabilistic approach. It is based on the idea that the predictor variables in a Machine Learning model are independent of each other. Meaning that the outcome of a model depends on a set of independent variables that have nothing to do with each other. Hence we have prepare our data set using multicollinearity to make the variable independent

The Bayes theorem is used to calculate the conditional probability, which is nothing but the probability of an event occurring based on information about the events in the past. Mathematically, the Bayes theorem is represented as: Bayes Theorem - Naive Bayes

```
library(e1071)
training_set$Cars = as.factor(training_set$Cars)
test_set$Cars = as.factor(test_set$Cars)
NB = naiveBayes(x =training_set[-1], y =training_set$Cars)
pred.NB = predict(NB, newdata =test_set[-1])
pred.NB

##   [1] 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 0 0 0 0 0 0 0 0 0
##  [36] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
##  [71] 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0
## Levels: 0 1

tab.NB =table(test_set[,1], pred.NB)
tab.NB

##    pred.NB
##      0  1
##   0 85 11
##   1  1  8

accuracy.NB = sum(diag(tab.NB))/sum(tab.NB)
accuracy.NB

## [1] 0.8857143
```

To check the efficiency of the model, we are now going to run the testing data set on the model, after which we will evaluate the accuracy of the model by using a Confusion matrix.

5.2.2 Model Performance Measure – Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of classes being predicted. For the problem in hand, we have N=2, and hence we get a 2 X 2 matrix.

Few performance parameters we can obtain with the help of confusion matrix are as follows:

Accuracy: the proportion of the total number of predictions that were correct.

Positive Predictive Value or Precision: the proportion of positive cases that were correctly identified.

Negative Predictive Value: the proportion of negative cases that were correctly identified.

Sensitivity or Recall: the proportion of actual positive cases which are correctly identified.

Specificity: the proportion of actual negative cases which are correctly identified.

Confusion Matrix for our given Model is as follow

```
confusionMatrix(pred.NB, test_set$Car)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 85  1
##          1 11  8
##
##                Accuracy : 0.8857
##                  95% CI : (0.8089, 0.9395)
##     No Information Rate : 0.9143
##     P-Value [Acc > NIR] : 0.885635
##
##                   Kappa : 0.515
##
##  Mcnemar's Test P-Value : 0.009375
##
##             Sensitivity : 0.8854
##             Specificity : 0.8889
##          Pos Pred Value : 0.9884
##          Neg Pred Value : 0.4211
##              Prevalence : 0.9143
##          Detection Rate : 0.8095
##    Detection Prevalence : 0.8190
##       Balanced Accuracy : 0.8872
##
##        'Positive' Class : 0
##
```

The final output shows that we built a Naive Bayes classifier that can predict whether an an employee will use Car as a mode of transport, with an accuracy of approximately 88%.

The model observed to perform fairly decent on majority of the model performance measures, indicating it to be a good model.

5.2.3 Create a classification task

Classifying all variable with Transport using Naive Bays classifier

```
names(cars)

##  [1] "Age"       "Gender"    "Engineer"   "MBA"       "Work.Exp"
##  [6] "Salary"    "Distance"  "license"    "Transport" "Gender_Dt"

#install.packages(mlr)
library(mlr)

## Loading required package: ParamHelpers

##
## Attaching package: 'mlr'

## The following object is masked from 'package:e1071':
##
##     impute

## The following object is masked from 'package:caret':
##
##     train

#Create a classification task for learning oncars Dataet and specify Transport fe
ature
task = makeClassifTask(data = cars,target = "Transport")
#Initialize the Naive Bayes classifier
selected_model = makeLearner("classif.naiveBayes")
#Train the model
NB_mlr = train(selected_model, task)
#Read the model learned
NB_mlr$learner.model

##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         2Wheeler              Car Public Transport
##       0.19856459       0.08373206       0.71770335
##
## Conditional probabilities:
##                      Age
## Y                         [,1]     [,2]
##   2Wheeler           25.26506 2.858809
##   Car                36.71429 3.391784
##   Public Transport 26.81333 2.957642
##
```

```
##                     Gender
## Y                       Female      Male
##   2Wheeler         0.4578313 0.5421687
##   Car              0.1714286 0.8285714
##   Public Transport 0.2566667 0.7433333
##
##                     Engineer
##   Y                      [,1]      [,2]
##   2Wheeler         0.7228916 0.4502913
##   Car              0.8571429 0.3550358
##   Public Transport 0.7433333 0.4375237
##
##                     MBA
## Y                        [,1]      [,2]
##   2Wheeler         0.2048193 0.4060228
##   Car              0.2571429 0.4434396
##   Public Transport 0.2775920 0.4485617
##
##                     Work.Exp
## Y                        [,1]     [,2]
##   2Wheeler          4.060241 3.317909
##   Car              17.514286 3.988007
##   Public Transport  5.016667 3.163820
##
##                     Salary
## Y                       [,1]      [,2]
##   2Wheeler         12.60964  6.048495
##   Car              41.29429 10.012432
##   Public Transport 13.17667  4.806996
##
##                     Distance
## Y                       [,1]    [,2]
##   2Wheeler         12.04458 3.32497
##   Car              17.88000 2.66809
##   Public Transport 10.31500 3.00537
##
##                     license
## Y                        [,1]      [,2]
##   2Wheeler         0.2771084 0.4502913
##   Car              0.8285714 0.3823853
##   Public Transport 0.1100000 0.3134125
##
##                     Gender_Dt
## Y                        [,1]      [,2]
##   2Wheeler         0.5421687 0.5012473
##   Car              0.8285714 0.3823853
##   Public Transport 0.7433333 0.4375237
```

Here we observed that salary is most significant variable which shows most Conditional probability

5.2.4 Confusion matrix to check accuracy for Naive classification task

```
predictions_mlr = as.data.frame(predict(NB_mlr, newdata = cars[,1:7]))

## Warning in predict.naiveBayes(.model$learner.model, newdata = .newdata, :
## Type mismatch between training and new data for variable 'license'. Did you
## use factors with numeric labels for training, and numeric values for new
## data?

## Warning in predict.naiveBayes(.model$learner.model, newdata = .newdata, :
## Type mismatch between training and new data for variable 'Gender_Dt'. Did
## you use factors with numeric labels for training, and numeric values for
## new data?

table(predictions_mlr[,1],cars$Transport)

##
##                     2Wheeler Car Public Transport
##   2Wheeler                24   0               15
##   Car                      2  32                8
##   Public Transport        57   3              277
```

As we see, the predictions are exactly same 35 true values . The only way to improve is to have more features or more data. We may arrive at a better model using Naive Bayes.

5.3 Logistic Regression

```
#logistic regression
table(training_set$Cars)

##
##   0   1
## 287  26

table(test_set$Cars)

##
##  0  1
## 96  9

cars_lg <-glm(Cars ~ .,data = training_set, family=binomial(link="logit"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

pred.lg <- predict.glm(cars_lg, newdata=test_set, type="response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading

table(test_set$Car,pred.lg>0.5)
```

```
##
##     FALSE TRUE
##   0    94    2
##   1     2    7
```

In Logistic regression we are able get true value 7 but somehow, we are not fine with the false values output .

Checking which variables are a significant predictor behind this decision

```
## > summary(cars_lg)

Call:
## glm(formula = Cars ~ ., family = binomial(link = "logit"), data = training_set)

## Deviance Residuals:
     Min        1Q    Median        3Q       Max
## -1.33207  -0.10075  -0.05877  -0.00002   2.67279

## Coefficients: (1 not defined because of singularities)
              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.8032     0.8554  -5.615 1.97e-08 ***
## `2Whleers`    -18.8993  2967.8678  -0.006  0.99492
## PublicTran          NA         NA      NA       NA
## Professional   2.3275     0.5537   4.204 2.63e-05 ***
## Distance `     2.2389     0.5879   3.808  0.00014 ***
## License       -0.3224     0.7187  -0.449  0.65370
## MBA            0.3433     0.8824   0.389  0.69719
---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 192.405  on 333  degrees of freedom
Residual deviance:  35.952  on 328  degrees of freedom
AIC: 47.952

Number of Fisher Scoring iterations: 20
```

According to the logistic regression model **Professionals** and **Distance** is highly significant

Pay close attention to the Pr(>|z|) or the p-value of the coefficients. A logistic regression model is said to be statistically significant only when the p-Values are less than the pre-determined statistical significance level, which is ideally 0.05. The p-value for each coefficient is represented as a probability Pr(>|z|).

We see here that both the coefficients have a very low p-value which means that both the coefficients are essential in computing the response variable.

The stars corresponding to the p-values indicate the significance of that respective variable. Since in our model, both the p values have a 3 star, this indicates that both the variables are extremely significant in predicting the response variable.

5.4 Bagging

```
#Bagging
library(gbm)

## Loaded gbm 2.1.5

#install.packages('xgboost')
library(xgboost)

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice

#install.packages('caret')
library(caret)
library(ipred)
library(rpart)

cars_bagging <- bagging(Cars ~.,data=training_set,
                         control=rpart.control(maxdepth=5, minsplit=4))

pred_class <- predict(cars_bagging, test_set)

table(test_set$Cars,pred_class)

##    pred_class
##      0  1
##   0 96  0
##   1  2  7
```

In Bagging also we are able predict exactly 7 true values ,which shows models full of accuracy but the false values are not correct classified correctly

5.5 Boosting

```
# XGBoost
# install.packages('xgboost')
library(xgboost)
#cars_data <-  cars_data %>% select (-Gender, -Transport)
set.seed(123)
split = sample.split(Carsdata_Mic$Cars, SplitRatio = 0.8)
training_set = subset(Carsdata_Mic, split == TRUE)
test_set = subset(Carsdata_Mic, split == FALSE)
classifier = xgboost(data = as.matrix(training_set[-1]), label = training_set$Car
s, nrounds = 10)

## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121

#Predicting the Test set results
y_pred <- predict(classifier, newdata = as.matrix(test set[-1]))
y_pred = (y_pred >= 0.5)

# Making the Confusion Matrix
cm = table(test_set$Cars, y_pred)
cm

##    y_pred
##     FALSE TRUE
##   0    75    2
##   1     0    7
```

We found in Boosting we may achieve somehow very near of false and true value as exactly it supposed to be classified

5.6 K Fold Cross Validation

```
# install.packages('caret')
library(caret)
folds = createFolds(training_set$Cars, k =10)
cv = lapply(folds, function(x) {
training_fold = training_set[-x,]
test_fold = training_set[x,]
classifier = xgboost(data = as.matrix(training_set[-1]),label = training_set$Cars
, nrounds = 10)
y_pred = predict(classifier, newdata = as.matrix(test_fold[-1]))
y_pred = (y_pred >= 0.5)
```

```
cm = table(test_fold[,1], y_pred)
cm
accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
return(accuracy)
})
```

```
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
```

```
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
```

```
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
```

```
accuracy = mean(as.numeric(cv))

accuracy

[1] 100
```

6. Communicating Results – Conclusion

Summary of comparison :

Accuracy.KNN	0.9619048
Accuracy.NB	0.8857143
Accuracy. Logistic Regression	0.9619048
Accuracy. Bagging	0.9809524
Accuracy.Boosting	0.9761905
Accuracy_K flod	100

7. Appendix A - Source Code

Project Objective and Scope

This project requires to understand what mode of transport employees prefers to commute to their office. The attached data 'Cars.csv, includes employee information about their mode of transport as well as their personal and professional details like age, salary, work exp. We need to predict whether or not an employee will use Car as a mode of transport. Also, which variables are a significant predictor behind this decision.

```r
#Loading required packages
library(tidyverse)
## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## -- Conflicts ------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
library(ggplot2)
library(caret)
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
library(caretEnsemble)
##
## Attaching package: 'caretEnsemble'
## The following object is masked from 'package:ggplot2':
##
##     autoplot
library(psych)
##
## Attaching package: 'psych'
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
library(Amelia)
## Loading required package: Rcpp
## ##
## ## Amelia II: Multiple Imputation
```

```
## ## (Version 1.7.5, built: 2018-05-07)
## ## Copyright (C) 2005-2019 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```r
library(mice)
```

```
##
## Attaching package: 'mice'
## The following object is masked from 'package:tidyr':
##
##     complete
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
##
## Attaching package: 'GGally'
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```r
library(gutenbergr)
library(tidytext)
library(dplyr)
library(janeaustenr)
library(stringi)
library(tidyr)
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:psych':
##
##     outlier
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
knitr::opts_chunk$set(echo = TRUE)
```

Importing Data

```
##Importing Data Set##
cars <- read.csv("C:/Users/SuprasannaPradhan/Documents/My Files/Great Lakes Projec
ts/cars.csv", header= TRUE)
```

Changing Variables

```
#Setting outcome variables as categorical
cars$Gender_Dt<-ifelse(cars$Gender=="Male",1,0)
cars_data <- as.data.frame(cars)
```

Key Observations

```
str(cars_data)
## 'data.frame':    418 obs. of  10 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 ...
```

```
##  $ Gender_Dt: num  1 1 0 1 0 1 1 1 1 1 ...
```

```r
head(cars_data)
```

```
##   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
##   Gender_Dt
## 1         1
## 2         1
## 3         0
## 4         1
## 5         0
## 6         1
```

```r
describe(cars_data)
```

```
##            vars   n  mean   sd median trimmed  mad  min  max range  skew
## Age           1 418 27.33 4.15   27.0   26.89 2.97 18.0 43.0  25.0  1.09
## Gender*       2 418  1.71 0.45    2.0    1.76 0.00  1.0  2.0   1.0 -0.93
## Engineer      3 418  0.75 0.43    1.0    0.81 0.00  0.0  1.0   1.0 -1.14
## MBA           4 417  0.26 0.44    0.0    0.20 0.00  0.0  1.0   1.0  1.08
## Work.Exp      5 418  5.87 4.82    5.0    5.12 2.97  0.0 24.0  24.0  1.52
## Salary        6 418 15.42 9.66   13.0   13.22 4.15  6.5 57.0  50.5  2.28
## Distance      7 418 11.29 3.70   10.9   11.08 3.56  3.2 23.4  20.2  0.55
## license       8 418  0.20 0.40    0.0    0.13 0.00  0.0  1.0   1.0  1.47
## Transport*    9 418  2.52 0.81    3.0    2.65 0.00  1.0  3.0   2.0 -1.20
## Gender_Dt    10 418  0.71 0.45    1.0    0.76 0.00  0.0  1.0   1.0 -0.93
##            kurtosis   se
## Age            1.67 0.20
## Gender*       -1.15 0.02
## Engineer      -0.69 0.02
## MBA           -0.83 0.02
## Work.Exp       2.29 0.24
```

```
## Salary         4.82 0.47
## Distance       0.05 0.18
## license        0.16 0.02
## Transport*    -0.38 0.04
## Gender_Dt     -1.15 0.02
```

Checking Missing Values

```
#visualize the missing data
missmap(cars_data)
sum(is.na(cars_data))
## [1] 1
cars_data[is.na(cars_data)] <- 0
sum(is.na(cars_data))
## [1] 0
```

The above illustrations show that our data set has only one missing values.

Exploratory Data Analysis

```
levels(cars_data$Transport)
## [1] "2Wheeler"         "Car"              "Public Transport"
cars_data$Car_Dt<-ifelse(cars_data$Transport=="Car",1,0)
cars_data$TW_Dt<-ifelse(cars_data$Transport=="2Wheeler",1,0)
cars_data$Pub_tran<-ifelse(cars_data$Transport=="Public.Transport",1,0)
```

Data Visualization

```
#visual 1
library(ggplot2)
library(dplyr)
ggplot(cars_data, aes(x=Age, colour = Transport)) +
  geom_freqpoly(binwidth = 1)+ labs(title="Age Distribution by Transport")
#visual 2
c <- ggplot(cars_data, aes(x=Work.Exp, fill=Transport, color=Transport)) +
  geom_histogram(binwidth = 1) + labs(title="Work Experince Distribution by by Tra
nsport")
```

```r
c + theme_bw()
#visual 3
s <- ggplot(cars_data, aes(x=Salary, fill=Transport, color=Transport)) +
  geom_histogram(binwidth = 1) + labs(title="Salary Distribution by Transport")
s + theme_bw()
#visual 4
d <- ggplot(cars_data, aes(x=Distance, fill=Transport, color=Transport)) +
  geom_histogram(binwidth = 1) + labs(title="Distance Distribution by Transport")
d + theme_bw()
#visual 5
ggplot(cars_data, aes(Engineer, colour = Transport)) +
  geom_freqpoly(binwidth = 1) + labs(title="Engineer Distribution by Transport")
#visual 6
ggplot(cars_data, aes(MBA, colour = Transport)) +
  geom_freqpoly(binwidth = 1) + labs(title="MBA Distribution by Transport")
#visual 5
cars_data <-  cars_data %>% select (-Gender,-Transport)
ggpairs(cars_data)
```

```
## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
```

```
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero

## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero
```

#Checking Multicollinearity#

```
#Checking cor plot#
cars_data1 = as.data.frame(cars_data)
cars_matrix = cor(cars_data1)
## Warning in cor(cars_data1): the standard deviation is zero
str(cars_data1 )
## 'data.frame':    418 obs. of  11 variables:
##  $ Age      : int  28 24 27 25 25 21 23 23 24 28 ...
##  $ Engineer : int  1 1 1 0 0 0 1 0 1 1 ...
##  $ MBA      : num  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 ...
##  $ Gender_Dt: num  1 1 0 1 0 1 1 1 1 1 ...
##  $ Car_Dt   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TW_Dt    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Pub_tran : num  0 0 0 0 0 0 0 0 0 0 ...
library(corrplot)
## corrplot 0.84 loaded
```

```
corrplot(cars_matrix)
corrplot(cars_matrix, type="upper", method="number")
```

Preparation of Module and Multicollinearity Check

```
ml <- lm(Car_Dt ~., data=cars_data1)
summary(ml)
```

```
##
## Call:
## lm(formula = Car_Dt ~ ., data = cars_data1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50653 -0.06726 -0.00254  0.06402  0.86680
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.313658   0.114882  -2.730 0.006602 **
## Age         -0.002149   0.004877  -0.441 0.659710
## Engineer     0.007441   0.017210   0.432 0.665719
## MBA         -0.021593   0.017189  -1.256 0.209761
## Work.Exp    -0.004347   0.005904  -0.736 0.461988
## Salary       0.021275   0.002330   9.132  < 2e-16 ***
## Distance     0.015125   0.002369   6.384 4.71e-10 ***
## license      0.090510   0.022053   4.104 4.90e-05 ***
## Gender_Dt   -0.028954   0.017309  -1.673 0.095136 .
## TW_Dt       -0.075182   0.020594  -3.651 0.000296 ***
## Pub_tran           NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1516 on 408 degrees of freedom
## Multiple R-squared:  0.7076, Adjusted R-squared:  0.7012
## F-statistic: 109.7 on 9 and 408 DF,  p-value: < 2.2e-16
```

Bartlett's Test:

```
data2 = subset(cars_data1, select = -c(9:11))
data4 = subset(cars_data1, select = c(9:11))
cormatrix = cor(data2)
str(data2)
## 'data.frame':    418 obs. of  8 variables:
##  $ Age      : int  28 24 27 25 25 21 23 23 24 28 ...
##  $ Engineer : int  1 1 1 0 0 0 1 0 1 1 ...
##  $ MBA      : num  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 ...
##  $ Gender_Dt: num  1 1 0 1 0 1 1 1 1 1 ...
library(psych)
cortest.bartlett(cormatrix,100)
## $chisq
## [1] 440.0267
##
## $p.value
## [1] 1.359223e-75
##
## $df
## [1] 28
```

KMO Test

```
KMO(cormatrix)
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = cormatrix)
## Overall MSA =  0.75
## MSA for each item =
##       Age  Engineer       MBA  Work.Exp    Salary  Distance   license
##      0.80      0.80      0.32      0.67      0.75      0.84      0.83
```

```
## Gender_Dt
##      0.63
```

Check Eigen Value

```
#Check Eigen Values
evector = eigen(cormatrix)
eigen_value = evector$values
eigen_value
## [1] 3.35411628 1.13551068 1.01948784 0.90791568 0.77433856 0.63243008
## [7] 0.13263566 0.04356523
plot(eigen_value, xlab = "Factors", ylab = "Eigen Values", col="red", pch=20)
lines(eigen_value, col="blue", lty = 2)
fa1 = fa(r= data2, nfactors =4, rotate ="none", fm ="pa")
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : An ultra-Heywood case was detected. Examine the results carefully
print(fa1)
## Factor Analysis using method =  pa
## Call: fa(r = data2, nfactors = 4, rotate = "none", fm = "pa")
## Standardized loadings (pattern matrix) based upon correlation matrix
##            PA1   PA2   PA3   PA4    h2     u2 com
## Age       0.90 -0.14  0.05 -0.10 0.849  0.151 1.1
## Engineer  0.09  0.03  0.12  0.11 0.035  0.965 2.9
## MBA       0.04  0.17  0.31  0.20 0.169  0.831 2.4
## Work.Exp  0.98 -0.19  0.13 -0.06 1.017 -0.017 1.1
## Salary    0.96 -0.03 -0.05  0.07 0.920  0.080 1.0
## Distance  0.48  0.14 -0.17  0.29 0.361  0.639 2.2
## license   0.49  0.41 -0.19 -0.09 0.452  0.548 2.3
## Gender_Dt 0.14  0.44  0.17 -0.15 0.267  0.733 1.8
##
##                        PA1  PA2  PA3  PA4
## SS loadings           3.19 0.47 0.23 0.19
## Proportion Var        0.40 0.06 0.03 0.02
## Cumulative Var        0.40 0.46 0.49 0.51
## Proportion Explained  0.78 0.11 0.06 0.05
```

```
## Cumulative Proportion 0.78 0.90 0.95 1.00
##
## Mean item complexity =  1.9
## Test of the hypothesis that 4 factors are sufficient.
##
## The degrees of freedom for the null model are  28  and the objective function w
as  4.61 with Chi Square of  1905.25
## The degrees of freedom for the model are 2  and the objective function was  0.0
1
##
## The root mean square of the residuals (RMSR) is  0
## The df corrected root mean square of the residuals is  0.01
##
## The harmonic number of observations is  418 with the empirical chi square  0.37
with prob <  0.83
## The total number of observations was  418  with Likelihood Chi Square =  3.2  w
ith prob <  0.2
##
## Tucker Lewis Index of factoring reliability =  0.991
## RMSEA index =  0.039  and the 90 % confidence intervals are  0 0.112
## BIC =  -8.87
## Fit based upon off diagonal values = 1
## Measures of factor score adequacy
##                                                    PA1  PA2   PA3   PA4
## Correlation of (regression) scores with factors   0.99 0.73  0.66  0.55
## Multiple R square of scores with factors          0.99 0.53  0.44  0.30
## Minimum correlation of possible factor scores     0.97 0.07 -0.12 -0.40
```

```
fa.diagram(fa1)
```

Factors with rotated and their plotted figures

```
fa2 = fa(r= data2, nfactors = 4
         , rotate ="varimax", fm ="pa")
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs
## = np.obs, : The estimated weights for the factor scores are probably
## incorrect. Try a different factor extraction method.
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : An ultra-Heywood case was detected. Examine the results carefully
```

```
print(fa2)
```

```
## Factor Analysis using method =  pa
## Call: fa(r = data2, nfactors = 4, rotate = "varimax", fm = "pa")
## Standardized loadings (pattern matrix) based upon correlation matrix
##              PA1  PA4  PA2   PA3    h2     u2 com
## Age         0.89 0.20 0.12  0.05 0.849  0.151 1.1
## Engineer    0.06 0.03 0.01  0.17 0.035  0.965 1.3
## MBA        -0.03 0.00 0.09  0.40 0.169  0.831 1.1
## Work.Exp    0.98 0.18 0.08  0.14 1.017 -0.017 1.1
## Salary      0.85 0.42 0.12  0.10 0.920  0.080 1.5
## Distance    0.29 0.51 0.05  0.12 0.361  0.639 1.8
## license     0.29 0.39 0.46 -0.06 0.452  0.548 2.7
## Gender_Dt   0.03 0.01 0.50  0.14 0.267  0.733 1.2
##
##                        PA1  PA4  PA2  PA3
## SS loadings           2.65 0.66 0.50 0.26
## Proportion Var        0.33 0.08 0.06 0.03
## Cumulative Var        0.33 0.41 0.48 0.51
## Proportion Explained  0.65 0.16 0.12 0.06
## Cumulative Proportion 0.65 0.81 0.94 1.00
##
## Mean item complexity =  1.5
## Test of the hypothesis that 4 factors are sufficient.
##
## The degrees of freedom for the null model are  28  and the objective function w
as  4.61 with Chi Square of  1905.25
## The degrees of freedom for the model are 2  and the objective function was  0.0
1
##
## The root mean square of the residuals (RMSR) is  0
## The df corrected root mean square of the residuals is  0.01
##
## The harmonic number of observations is  418 with the empirical chi square  0.37
with prob <  0.83
```

```
## The total number of observations was  418  with Likelihood Chi Square =  3.2  w
ith prob <  0.2
##
## Tucker Lewis Index of factoring reliability =  0.991
## RMSEA index =  0.039  and the 90 % confidence intervals are  0 0.112
## BIC =  -8.87
## Fit based upon off diagonal values = 1
fa.diagram(fa2)
```

##We have changed factor names to ## 1. Professional 2. Distance 3. License 4. MBA

```
Carsdata_Mic = cbind(data4,fa2$scores)
head(Carsdata_Mic)
##   Car_Dt TW_Dt Pub_tran        PA1        PA4         PA2         PA3
## 1      0     1        0 -0.1315237 -0.4467651  0.20259100 -0.34126862
## 2      0     1        0  0.4483541 -1.5530375 -0.08023245  0.32422828
## 3      0     1        0  1.1544105 -1.4364795 -0.99503567  0.05182746
## 4      0     1        0 -1.0069960 -0.2519568  0.28808071 -0.74157021
## 5      0     1        0 -0.4101696 -0.3982738 -0.72322719 -0.77508798
## 6      0     1        0 -0.3557918 -0.7985349 -0.03510892 -0.05420258
colnames(Carsdata_Mic) = c("Cars", "2Whleers","PublicTran", "Professional", "Dista
nce ","License","MBA")
str(Carsdata_Mic)
## 'data.frame':    418 obs. of  7 variables:
##  $ Cars        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ 2Whleers    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ PublicTran  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Professional: num  -0.132 0.448 1.154 -1.007 -0.41 ...
##  $ Distance    : num  -0.447 -1.553 -1.436 -0.252 -0.398 ...
##  $ License     : num  0.2026 -0.0802 -0.995 0.2881 -0.7232 ...
##  $ MBA         : num  -0.3413 0.3242 0.0518 -0.7416 -0.7751 ...
```

Perform Data Modelling & Evaluation

```
# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
```

```
library(dplyr)
str(Carsdata_Mic)
## 'data.frame':    418 obs. of  7 variables:
##  $ Cars        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ 2Whleers    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ PublicTran  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Professional: num  -0.132 0.448 1.154 -1.007 -0.41 ...
##  $ Distance    : num  -0.447 -1.553 -1.436 -0.252 -0.398 ...
##  $ License     : num  0.2026 -0.0802 -0.995 0.2881 -0.7232 ...
##  $ MBA         : num  -0.3413 0.3242 0.0518 -0.7416 -0.7751 ...
table(Carsdata_Mic$Cars)
##
##   0   1
## 383  35
set.seed(108)
split = sample.split(Carsdata_Mic$Cars, SplitRatio = 0.75)
training_set = subset(Carsdata_Mic, split == TRUE)
test_set = subset(Carsdata_Mic, split == FALSE)
table(training_set$Cars)
##
##   0   1
## 287  26
table(test_set$Cars)
##
##  0  1
## 96  9
```

KNN Classifier

```
library(class)
knn_fit<- knn(train = training_set[-1], test = test_set[-1], cl=training_set$Cars,
k =3,prob=TRUE)
knn_chk= table(test_set$Cars,knn_fit)
knn_chk
##    knn_fit
```

```
##       0  1
##    0 94  2
##    1  2  7
accuracy.knn = sum(diag(knn_chk))/sum(knn_chk)*100
accuracy.knn
## [1] 96.19048
```

Naive Bays

```
library(e1071)
training_set$Cars = as.factor(training_set$Cars)
test_set$Cars = as.factor(test_set$Cars)
NB = naiveBayes(x =training_set[-1], y =training_set$Cars)
pred.NB = predict(NB, newdata =test_set[-1])
pred.NB
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
##  [36] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
##  [71] 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0
## Levels: 0 1
tab.NB =table(test_set[,1], pred.NB)
tab.NB
##     pred.NB
##       0  1
##    0 85 11
##    1  1  8
accuracy.NB = sum(diag(tab.NB))/sum(tab.NB)
accuracy.NB
## [1] 0.8857143
confusionMatrix(pred.NB, test_set$Car)
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##            0 85  1
##            1 11  8
```

```
##
##                Accuracy : 0.8857
##                  95% CI : (0.8089, 0.9395)
##     No Information Rate : 0.9143
##     P-Value [Acc > NIR] : 0.885635
##
##                   Kappa : 0.515
##
##  Mcnemar's Test P-Value : 0.009375
##
##             Sensitivity : 0.8854
##             Specificity : 0.8889
##          Pos Pred Value : 0.9884
##          Neg Pred Value : 0.4211
##              Prevalence : 0.9143
##          Detection Rate : 0.8095
##    Detection Prevalence : 0.8190
##       Balanced Accuracy : 0.8872
##
##        'Positive' Class : 0
##
```

Clasifiying all variable with Transport using NaiveBays classifer

```
names(cars)
##  [1] "Age"       "Gender"    "Engineer"  "MBA"       "Work.Exp"
##  [6] "Salary"    "Distance"  "license"   "Transport" "Gender_Dt"
#install.packages(mlr)
library(mlr)
## Loading required package: ParamHelpers
##
## Attaching package: 'mlr'
## The following object is masked from 'package:e1071':
##
##     impute
```

```
## The following object is masked from 'package:caret':
##
##     train
```

```r
#Create a classification task for learning oncars Dataet and specify Transport fea
ture
task = makeClassifTask(data = cars,target = "Transport")
#Initialize the Naive Bayes classifier
selected_model = makeLearner("classif.naiveBayes")
#Train the model
NB_mlr = train(selected_model, task)
#Read the model learned
NB_mlr$learner.model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##          2Wheeler              Car Public Transport
##        0.19856459       0.08373206       0.71770335
##
## Conditional probabilities:
##                   Age
## Y                      [,1]     [,2]
##   2Wheeler         25.26506 2.858809
##   Car              36.71429 3.391784
##   Public Transport 26.81333 2.957642
##
##                   Gender
## Y                     Female      Male
##   2Wheeler         0.4578313 0.5421687
##   Car              0.1714286 0.8285714
##   Public Transport 0.2566667 0.7433333
```

```
##
##                     Engineer
## Y                          [,1]      [,2]
##   2Wheeler         0.7228916 0.4502913
##   Car              0.8571429 0.3550358
##   Public Transport 0.7433333 0.4375237
##
##                     MBA
## Y                          [,1]      [,2]
##   2Wheeler         0.2048193 0.4060228
##   Car              0.2571429 0.4434396
##   Public Transport 0.2775920 0.4485617
##
##                     Work.Exp
## Y                          [,1]     [,2]
##   2Wheeler          4.060241 3.317909
##   Car              17.514286 3.988007
##   Public Transport  5.016667 3.163820
##
##                     Salary
## Y                         [,1]      [,2]
##   2Wheeler         12.60964  6.048495
##   Car              41.29429 10.012432
##   Public Transport 13.17667  4.806996
##
##                     Distance
## Y                         [,1]    [,2]
##   2Wheeler         12.04458 3.32497
##   Car              17.88000 2.66809
##   Public Transport 10.31500 3.00537
##
##                     license
## Y                          [,1]      [,2]
##   2Wheeler         0.2771084 0.4502913
```

```
##   Car                0.8285714 0.3823853
##   Public Transport 0.1100000 0.3134125
##
##                     Gender_Dt
## Y                          [,1]      [,2]
##   2Wheeler          0.5421687 0.5012473
##   Car                0.8285714 0.3823853
##   Public Transport 0.7433333 0.4375237
```

Confusion matrix to check accuracy

```
predictions_mlr = as.data.frame(predict(NB_mlr, newdata = cars[,1:7]))
## Warning in predict.naiveBayes(.model$learner.model, newdata = .newdata, :
## Type mismatch between training and new data for variable 'license'. Did you
## use factors with numeric labels for training, and numeric values for new
## data?
## Warning in predict.naiveBayes(.model$learner.model, newdata = .newdata, :
## Type mismatch between training and new data for variable 'Gender_Dt'. Did
## you use factors with numeric labels for training, and numeric values for
## new data?
table(predictions_mlr[,1],cars$Transport)
##
##                      2Wheeler Car Public Transport
##   2Wheeler                 24   0               15
##   Car                       2  32                8
##   Public Transport         57   3              277
```

As we see, the predictions are exactly same 35 true values . The only way to improve is to have more features or more data.we may arrive at a better model using Naive Bayes. ###logistic regression##

```
#logistic regression
table(training_set$Cars)
##
##   0   1
## 287  26
table(test_set$Cars)
```

```
##
##   0  1
## 96  9
cars_lg <-glm(Cars ~ .,data = training_set, family=binomial(link="logit"))
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
summary(cars_lg)
##
## Call:
## glm(formula = Cars ~ ., family = binomial(link = "logit"), data = training_set)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.47483  -0.04200  -0.02149   0.00000   2.84980
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -6.1004     1.7640  -3.458 0.000544 ***
## `2Whleers`      -20.6425  4481.8419  -0.005 0.996325
## PublicTran            NA         NA      NA       NA
## Professional      2.5819     0.8498   3.038 0.002380 **
## `Distance `       3.6894     1.3251   2.784 0.005364 **
## License          -0.0761     1.1703  -0.065 0.948154
## MBA               1.1135     1.7705   0.629 0.529411
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 179.159  on 312  degrees of freedom
## Residual deviance:  15.637  on 307  degrees of freedom
## AIC: 27.637
##
## Number of Fisher Scoring iterations: 21
pred.lg <- predict.glm(cars_lg, newdata=test_set, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading
tab.lg <- table(test_set$Car,pred.lg>0.5)
tab.lg
##
##      FALSE TRUE
##    0    94    2
##    1     2    7
accuracy.lg = sum(diag(tab.lg))/sum(tab.lg)
accuracy.lg
## [1] 0.9619048
```

Bagging

```
#Bagging
library(gbm)
## Loaded gbm 2.1.5
#install.packages('xgboost')
library(xgboost)
##
## Attaching package: 'xgboost'
## The following object is masked from 'package:dplyr':
##
##     slice
#install.packages('caret')
library(caret)
library(ipred)
library(rpart)

cars_bagging <- bagging(Cars ~.,data=training_set,
                         control=rpart.control(maxdepth=5, minsplit=4))

pred_class <- predict(cars_bagging, test_set)
tab.bg <- table(test_set$Cars,pred_class)
accuracy.bg = sum(diag(tab.bg))/sum(tab.bg)
```

```
accuracy.bg
## [1] 0.9809524
```

In Bagging also we are able predict exactly 7 trule values ,which shows models full of accuracy but the false values are not correct clasifed correctly

Boosting

```
# XGBoost
# install.packages('xgboost')
library(xgboost)
#cars_data <-  cars_data %>% select (-Gender, -Transport)
set.seed(123)
split = sample.split(Carsdata_Mic$Cars, SplitRatio = 0.8)
training_set = subset(Carsdata_Mic, split == TRUE)
test_set = subset(Carsdata_Mic, split == FALSE)
classifier = xgboost(data = as.matrix(training_set[-1]), label = training_set$Cars
, nrounds = 10)
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
#Predicting the Test set results
y_pred <- predict(classifier, newdata = as.matrix(test_set[-1]))
y_pred = (y_pred >= 0.5)

# Making the Confusion Matrix
cm = table(test_set$Cars, y_pred)
cm
##    y_pred
```

```
##      FALSE TRUE
##    0    75    2
##    1     0    7
```

```r
accuracy.bs = sum(diag(cm))/sum(cm)
accuracy.bs
```

```
## [1] 0.9761905
```

K Fold Cross Validation

```r
# install.packages('caret')
library(caret)
folds = createFolds(training_set$Cars, k =10)
cv = lapply(folds, function(x) {
training_fold = training_set[-x,]
test_fold = training_set[x,]
classifier = xgboost(data = as.matrix(training_set[-1]),label = training_set$Cars,
nrounds = 10)
y_pred = predict(classifier, newdata = as.matrix(test_fold[-1]))
y_pred = (y_pred >= 0.5)
cm = table(test_fold[,1], y_pred)
cm
accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
return(accuracy)
})
```

```
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
```

```
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
```

```
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
```

```
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
## [1]  train-rmse:0.354530
## [2]  train-rmse:0.251027
## [3]  train-rmse:0.178121
## [4]  train-rmse:0.126761
## [5]  train-rmse:0.092198
## [6]  train-rmse:0.067090
## [7]  train-rmse:0.049546
## [8]  train-rmse:0.036644
## [9]  train-rmse:0.027902
## [10] train-rmse:0.021121
```

```
accuracy_kf = mean(as.numeric(cv))*100
accuracy_kf
```

```
## [1] 100
```

Comparison of Accuracy

```
accuracy.knn
```

```
## [1] 96.19048
```

```
accuracy.NB
```

```
## [1] 0.8857143
```

```
accuracy.lg
```

```
## [1] 0.9619048
```

```
accuracy.bg
```

```
## [1] 0.9809524
```

```
accuracy.bs
## [1] 0.9761905
accuracy_kf
## [1] 100
```

"Thank You"