MINI PROJECT PREDICTING MODE OF TRANSPORT – CARS MACHINE LEARNING

Rithu A M

Table of Contents

➢ **Project Objective:**

This case study is prepared for an organization to study their employees transport preference to commute and need to predict whether an employee will use car as a mode of transport. Also, which variables are a significant predictor behind this decision? The objective is to build the best model using Machine Learning techniques which can identify right employees who prefers cars.

We would be analysing a data-set Cars.csv and performing techniques like logistic regression, KNN, Naïve Bayes and apply boosting and bagging modelling procedures to create 2 models then compare accuracy to come up with best model for our prediction.

➢ **Assumptions:**

There are few assumptions needs to be considered,

- The sample size is adequate to perform techniques like Logistic Regression, KNN, Naïve Bayes.
- All the necessary packages are installed in R
- Working Directly is set to appropriate folder and file is in CSV Format

➢ **Exploratory Data Analysis – Step by step approach:**

Data Dictionary:

Age	Age of the Employee in Years
Gender	Gender of the Employee
Engineer	For Engineer =1, Non-Engineer =0
MBA	For MBA =1, Non-MBA =0
Work Exp	Experience in years
Salary	Salary in Lakhs per Annum
Distance	Distance in Kms from Home to Office
license	If Employee has Driving Licence -1, If not, then 0
Transport	Mode of Transport

The dataset has data of 444 observations and 9 variables about their mode of transport as well as their personal and professional details like Age, salary, Work Exp etc.

Setting up working Directory:

```
setwd("G:/Projects/Machine Learning -Predicting Mode of Transport Project")
> getwd()
[1] "G:/Projects/Machine Learning -Predicting Mode of Transport Project"
```

Import and read the dataset:

```
cars=read.csv ("Cars_edited.csv",header = TRUE)
```

Summary of Data:

```
summary(cars)
Age           Gender            Engineer          MBA            Work.Exp        Salary
 Min.   :18.00   Length:444         Min.   :0.0000   Min.   :0.0000   Min.   : 0.0   Min.   : 6.50
 1st Qu.:25.00   Class :character   1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.: 3.0   1st Qu.: 9.80
 Median :27.00   Mode  :character   Median :1.0000   Median :0.0000   Median : 5.0   Median :13.60
 Mean   :27.75                      Mean   :0.7545   Mean   :0.2528   Mean   : 6.3   Mean   :16.24
 3rd Qu.:30.00                      3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 8.0   3rd Qu.:15.72
 Max.   :43.00                      Max.   :1.0000   Max.   :1.0000   Max.   :24.0   Max.   :57.00
                                                     NA's   :1
   Distance         license          Transport
 Min.   : 3.20   Min.   :0.0000   Length:444
 1st Qu.: 8.80   1st Qu.:0.0000   Class :character
 Median :11.00   Median :0.0000   Mode  :character
 Mean   :11.32   Mean   :0.2342
 3rd Qu.:13.43   3rd Qu.:0.0000
 Max.   :23.40   Max.   :1.0000
```

Structure of Data:

```
str(cars)
```

```
'data.frame':   444 obs. of  9 variables:
 $ Age      : int  28 23 29 28 27 26 28 26 22 27 ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 2 1 2 1 2 2 2 1 2 2 ...
 $ Engineer : int  0 1 1 1 1 1 1 1 1 1 ...
 $ MBA      : int  0 0 0 1 0 0 0 0 0 0 ...
 $ Work.Exp : int  4 4 7 5 4 4 5 3 1 4 ...
 $ Salary   : num  14.3 8.3 13.4 13.4 13.4 12.3 14.4 10.5 7.5 13.5  ...
 $ Distance : num  3.2 3.3 4.1 4.5 4.6 4.8 5.1 5.1 5.1 5.2 ...
 $ license  : int  0 0 0 0 0 1 0 0 0 0 ...
 $ Transport: Factor w/ 3 levels "2Wheeler","Car",..: 3 3 3 3 3 3 1 3 3 3 ...
```

Number of Rows and Columns:

The number of rows in the dataset is 444

The number of columns in the dataset is 9

Target Variable is "Transport"

Data Visualization of the Variables (Plots, charts and Graphs)

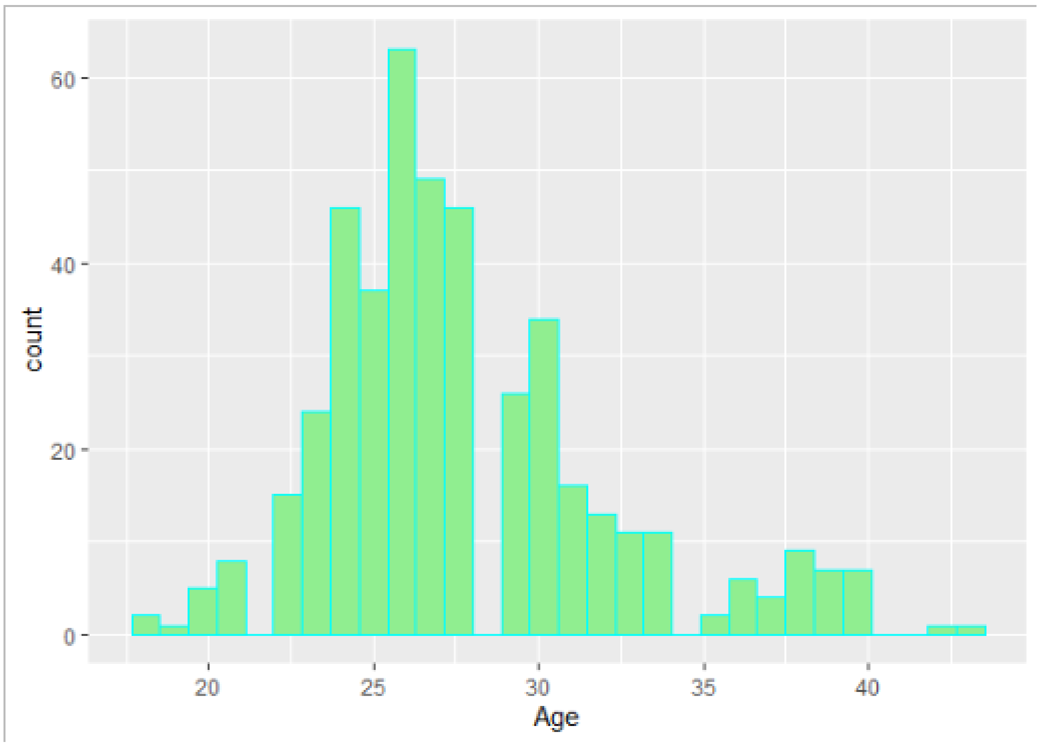
Univariate Analysis:

Data Analysis is done on each feature for better understanding

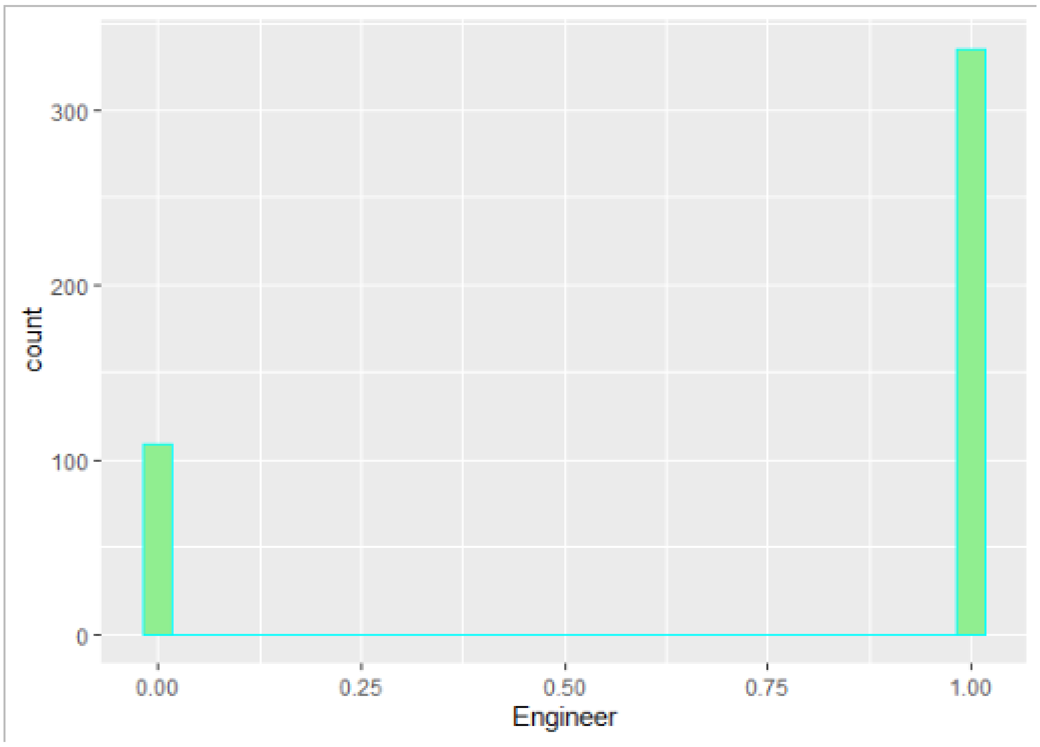
```
colnames(cars[,sapply(cars, is.numeric)])
```

```
[1] "Age"      "Engineer" "MBA"      "Work.Exp" "Salary"   "Distance" "license"
```

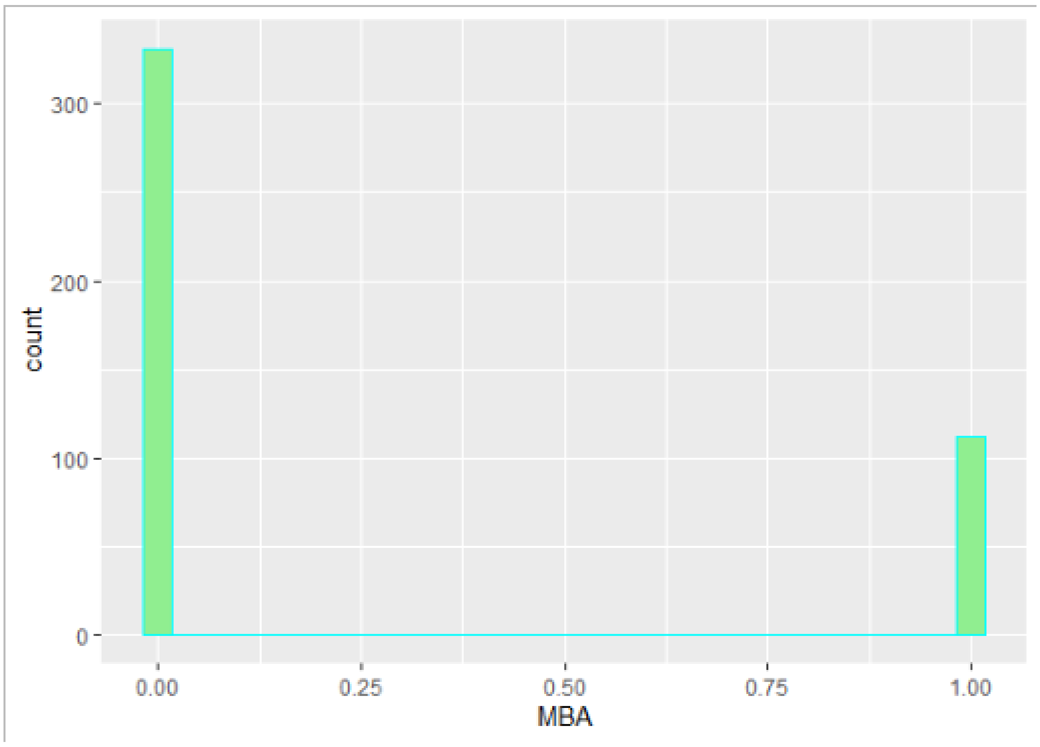
```
ggplotcars, aes(x = Age)) + geom_histogram(fill = "lightgreen", col = "cyan")
```



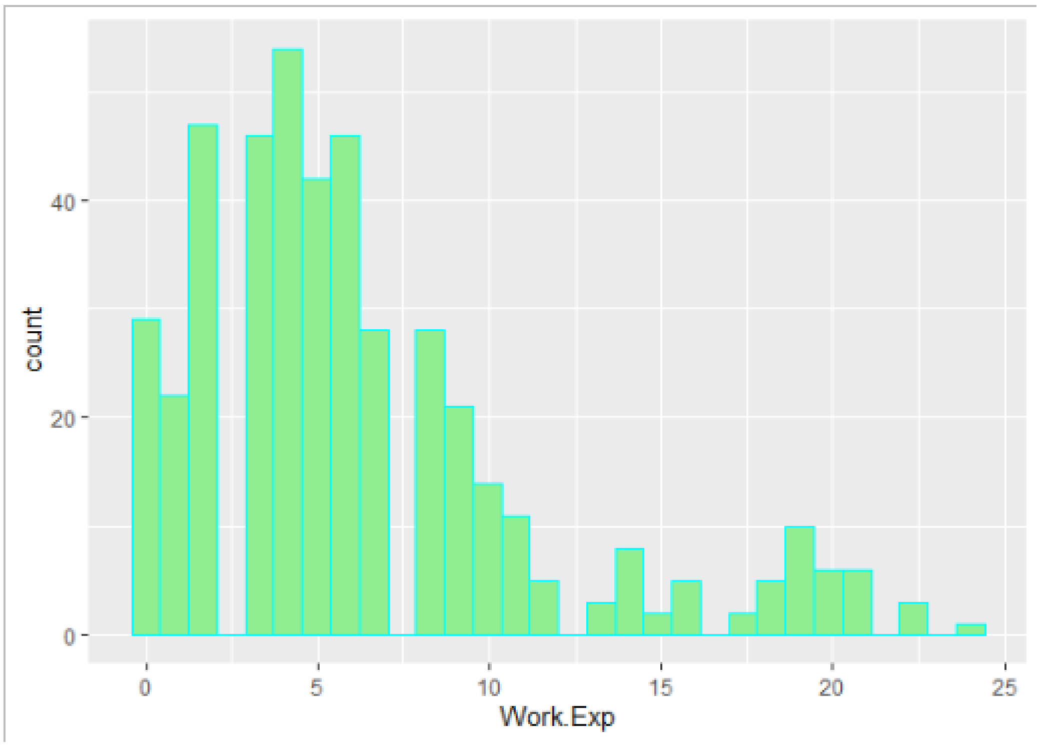
```
ggplot(cars, aes(x = Engineer)) + geom_histogram(fill = "lightgreen", col = "cyan")
```



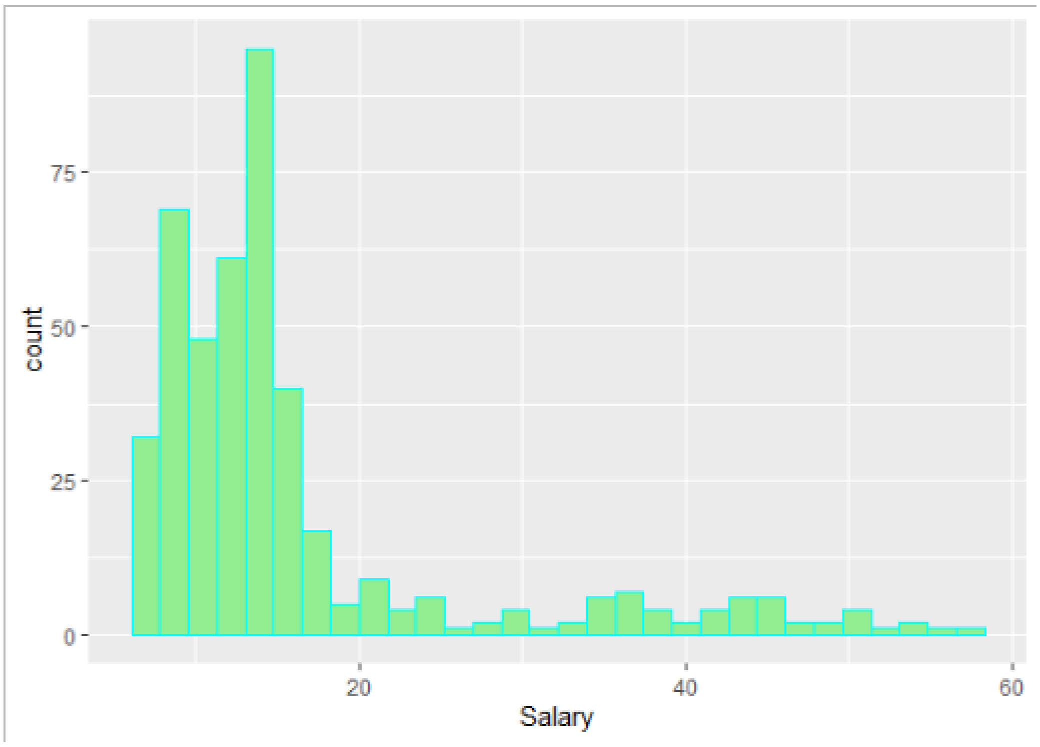
```
ggplot(cars, aes(x = MBA)) + geom_histogram(fill = "lightgreen", col = "cyan")
```



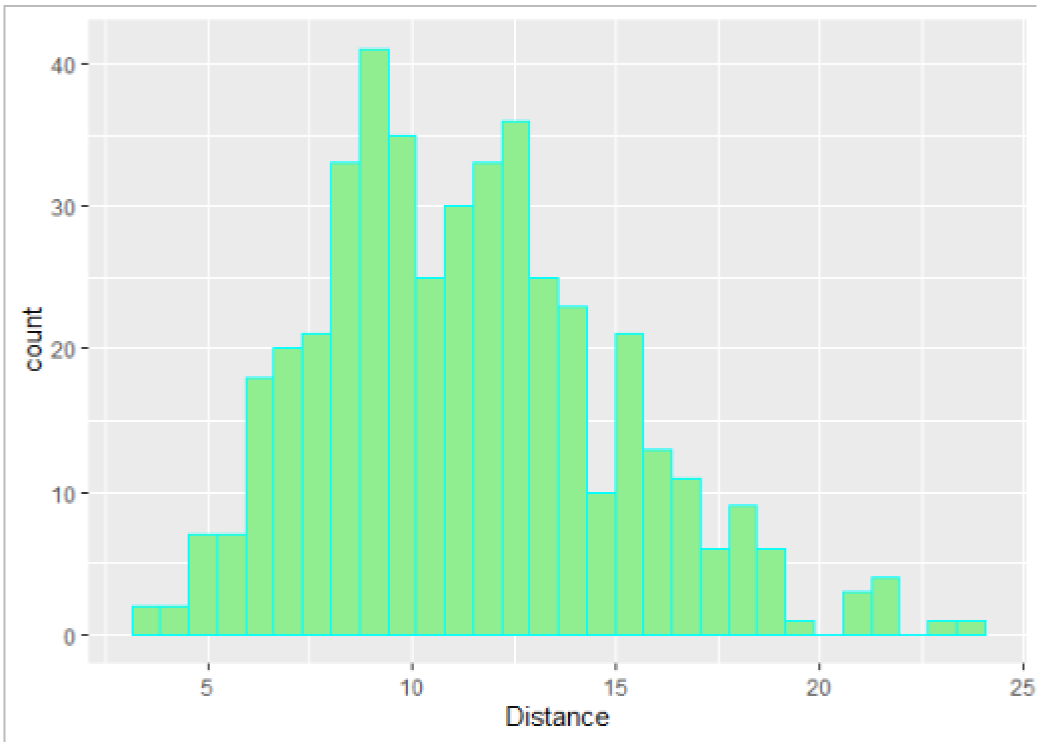
```
ggplot(cars, aes(x = Work.Exp)) + geom_histogram(fill = "lightgreen", col = "cyan")
```



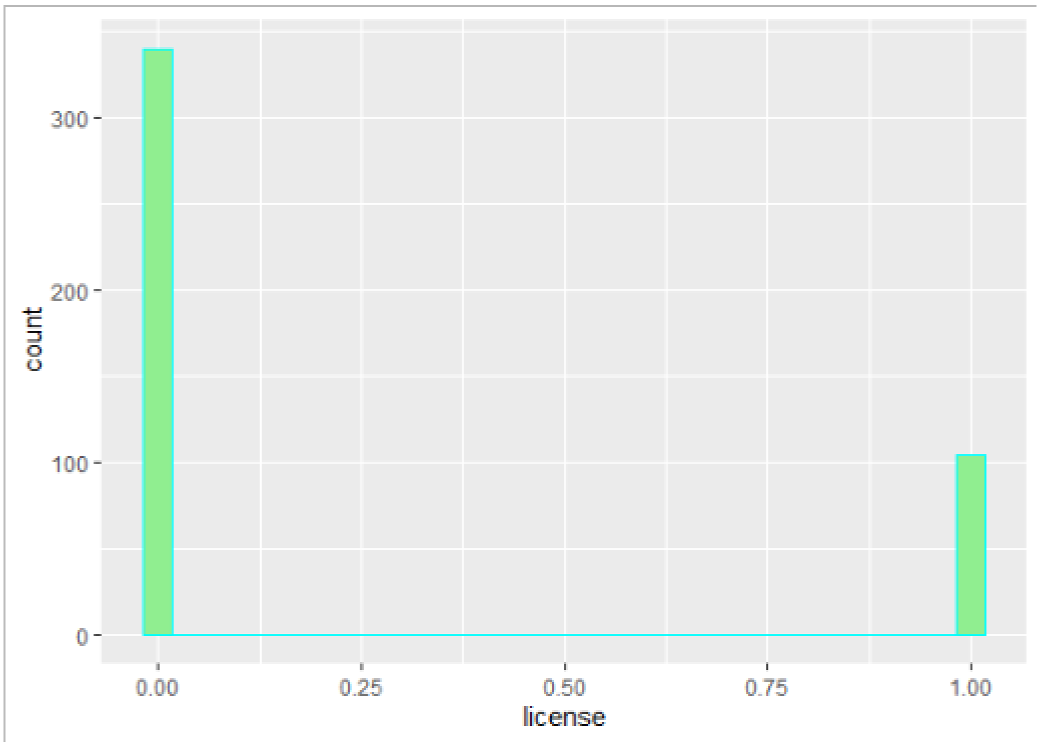
```
ggplot(cars, aes(x = Salary)) + geom_histogram(fill = "lightgreen", col = "cyan")
```



```
ggplot(cars, aes(x = Distance)) + geom_histogram(fill = "lightgreen", col = "cyan")
```



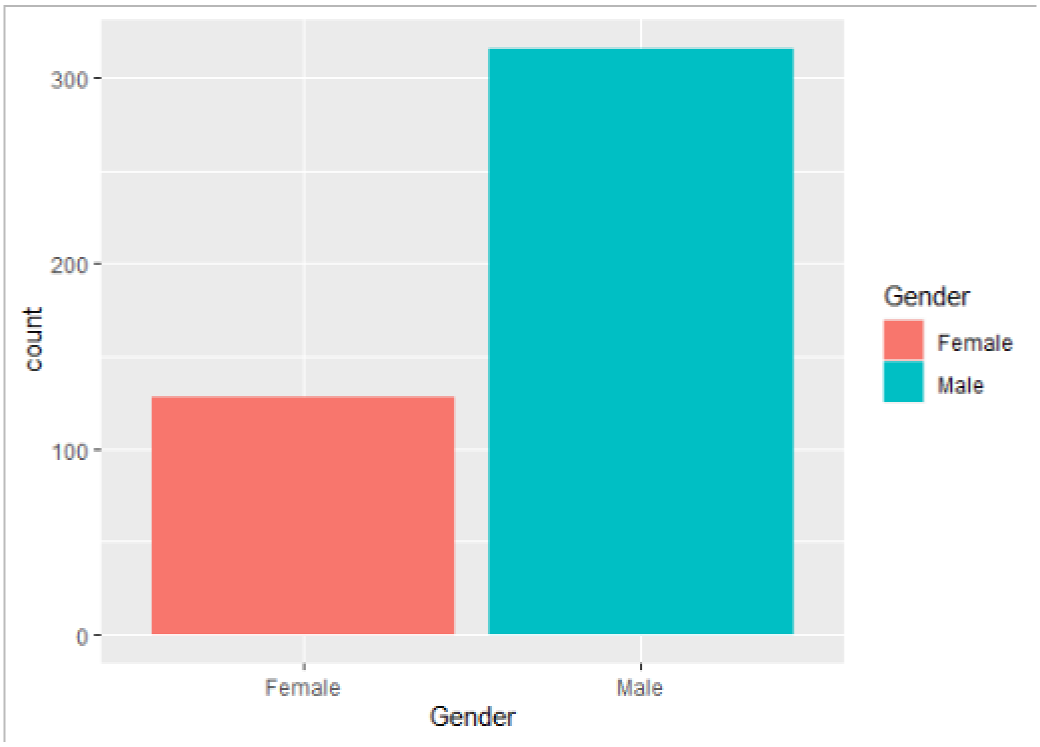
```
ggplot(cars, aes(x = license)) + geom_histogram(fill = "lightgreen", col = "cyan")
```



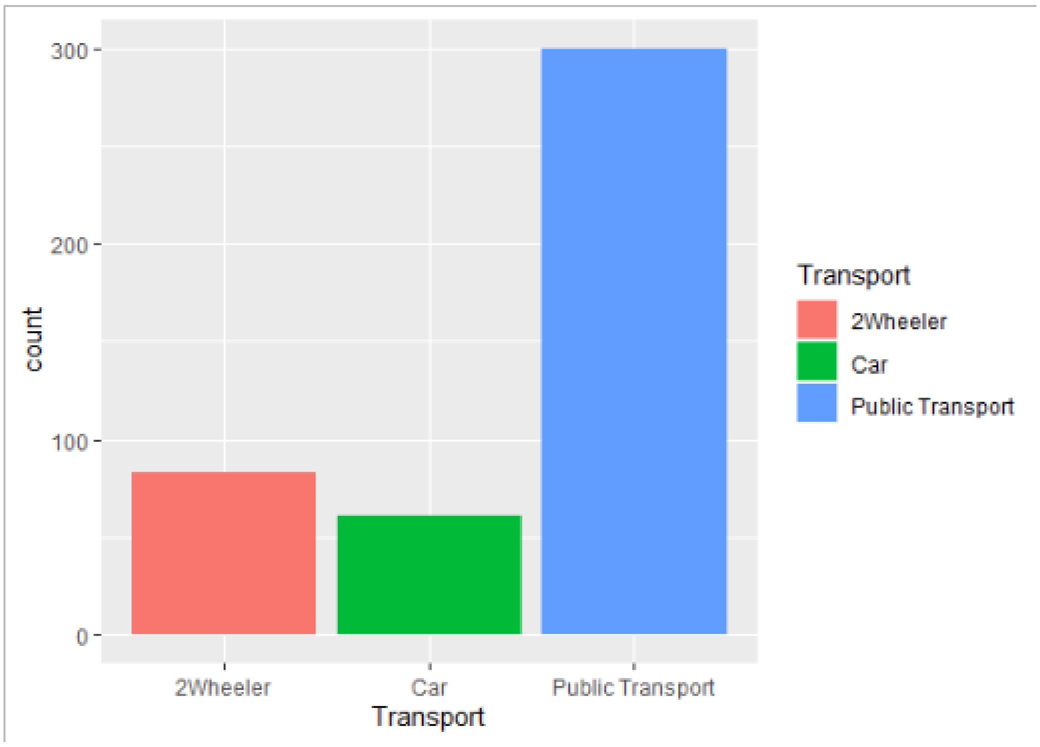
```
colnames(cars[,sapply(cars, is.factor)])
```

```
[1] "Gender"    "Transport"
```

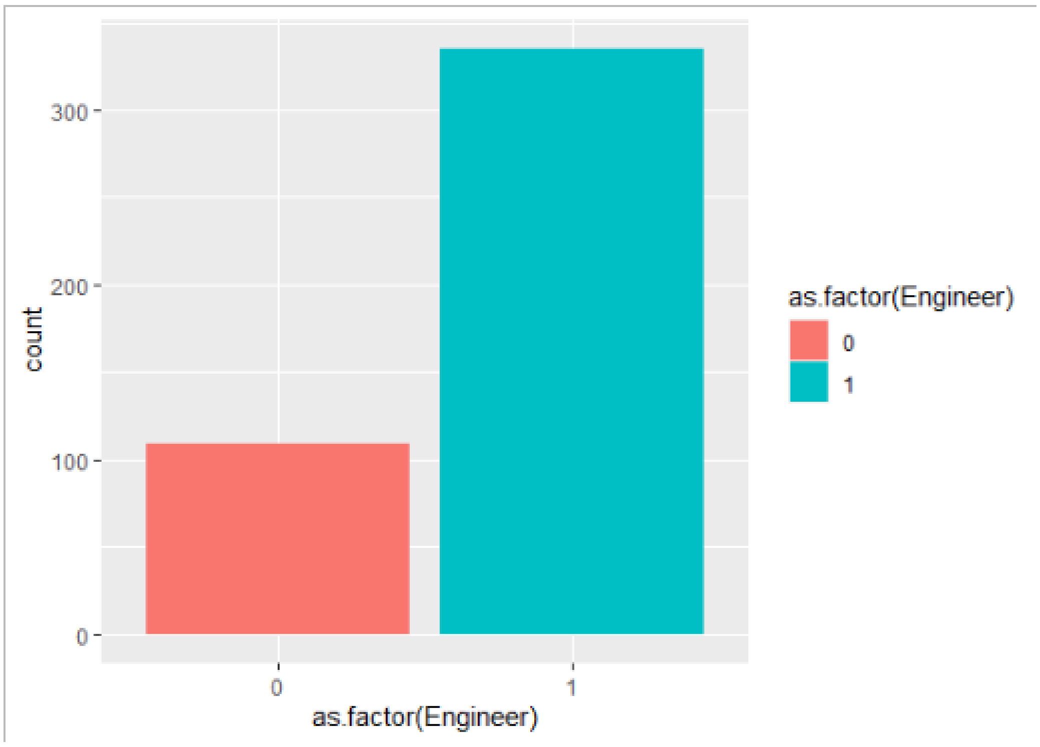
```
ggplot(cars, aes(x = Gender, fill = Gender)) + geom_bar()
```



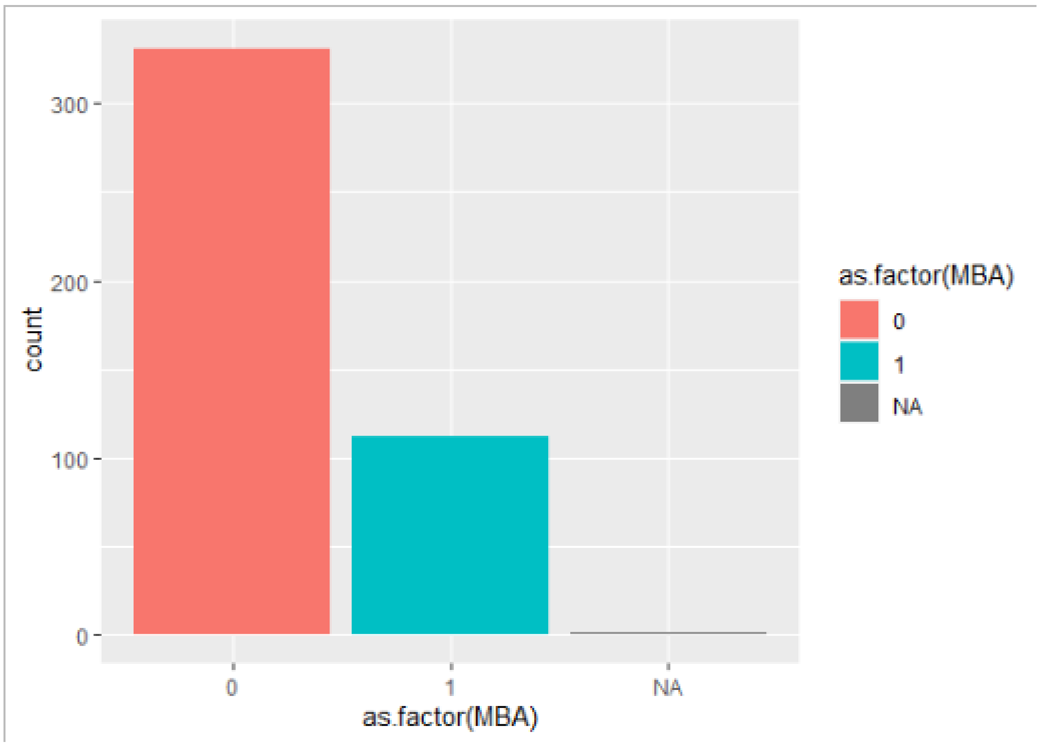
```
ggplot(cars, aes(x = Transport, fill = Transport)) + geom_bar()
```



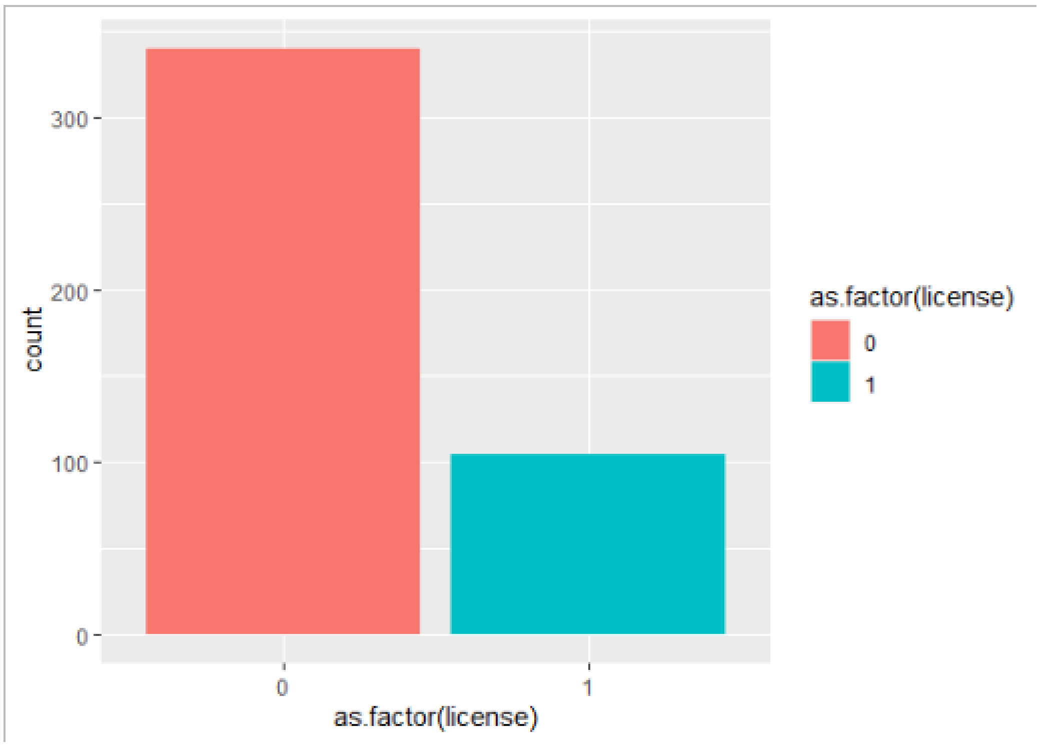
```
ggplot(cars, aes(x = as.factor(Engineer), fill = as.factor(Engineer))) + geom_bar()
```



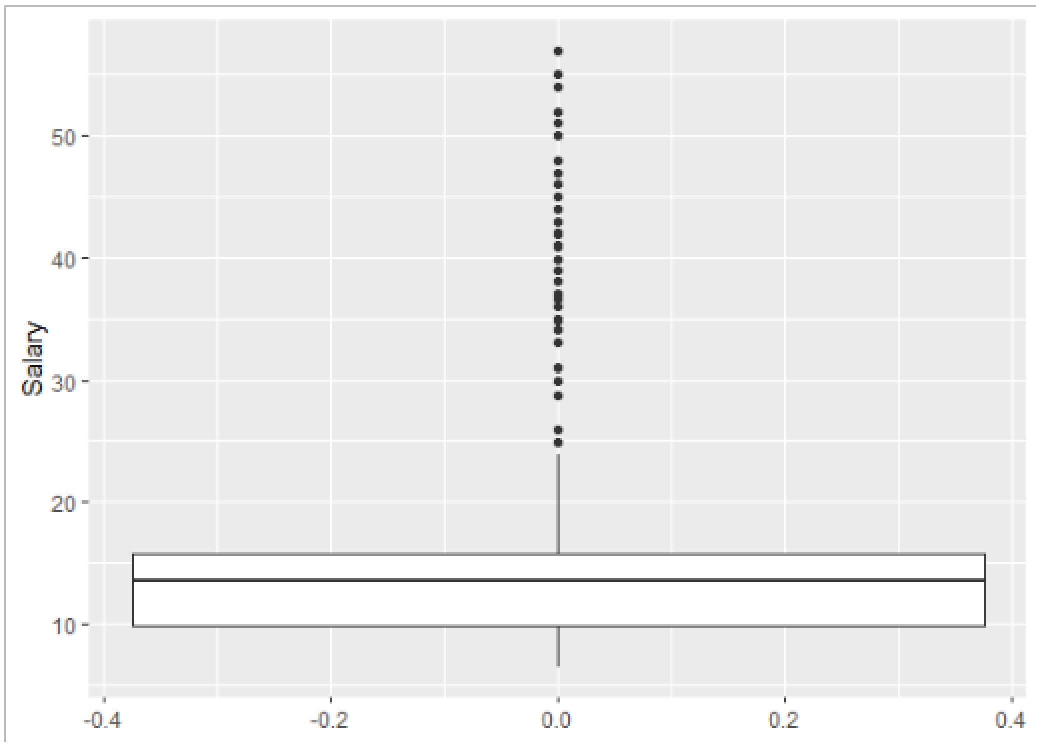
```
ggplot(cars, aes(x = as.factor(MBA), fill = as.factor(MBA))) + geom_bar()
```



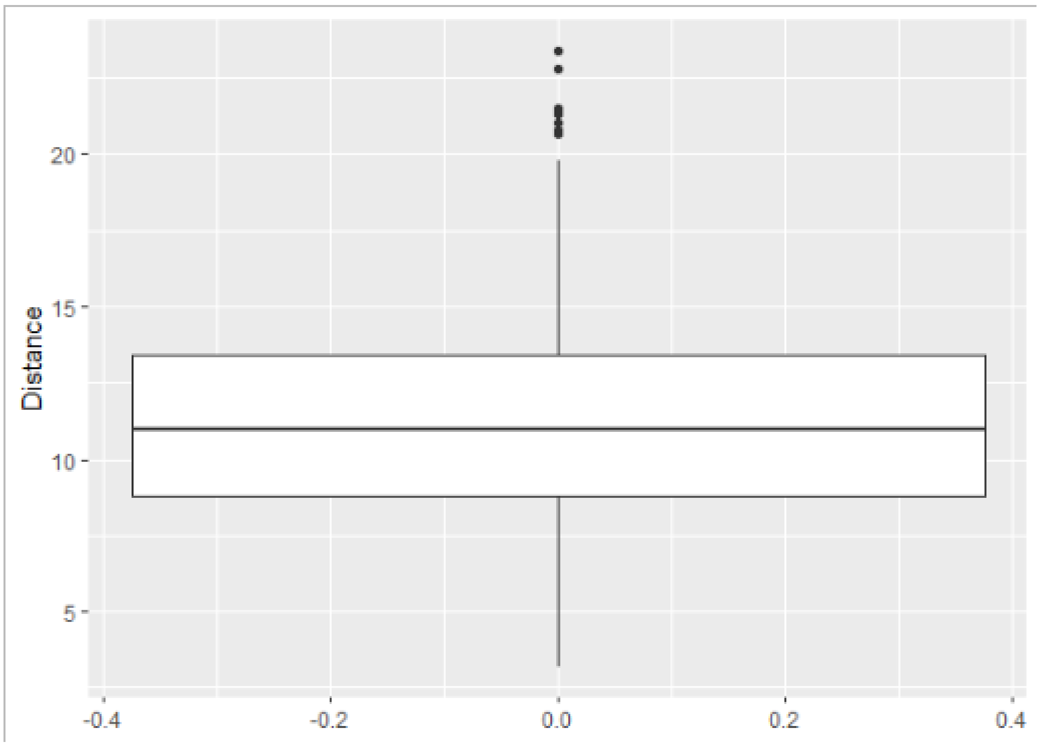
```
ggplot(cars, aes(x = as.factor(license), fill = as.factor(license))) + geom_bar()
```



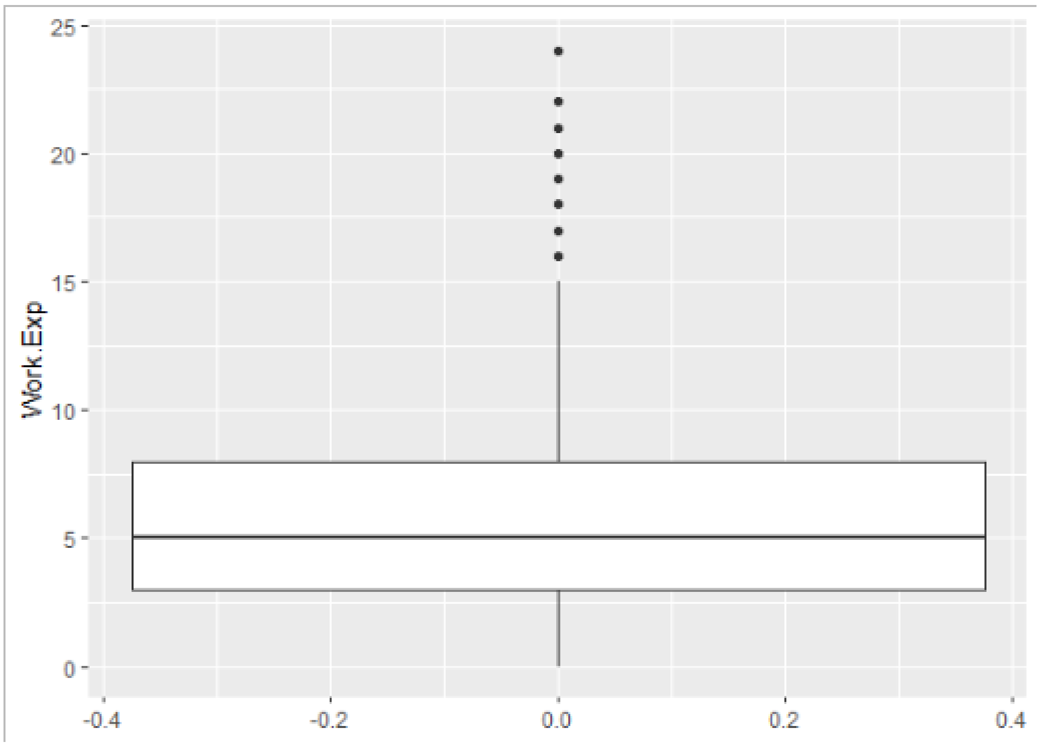
```
ggplot(cars, aes(y = Salary)) + geom_boxplot()
```



```
ggplot(cars, aes(y = Distance)) + geom_boxplot()
```

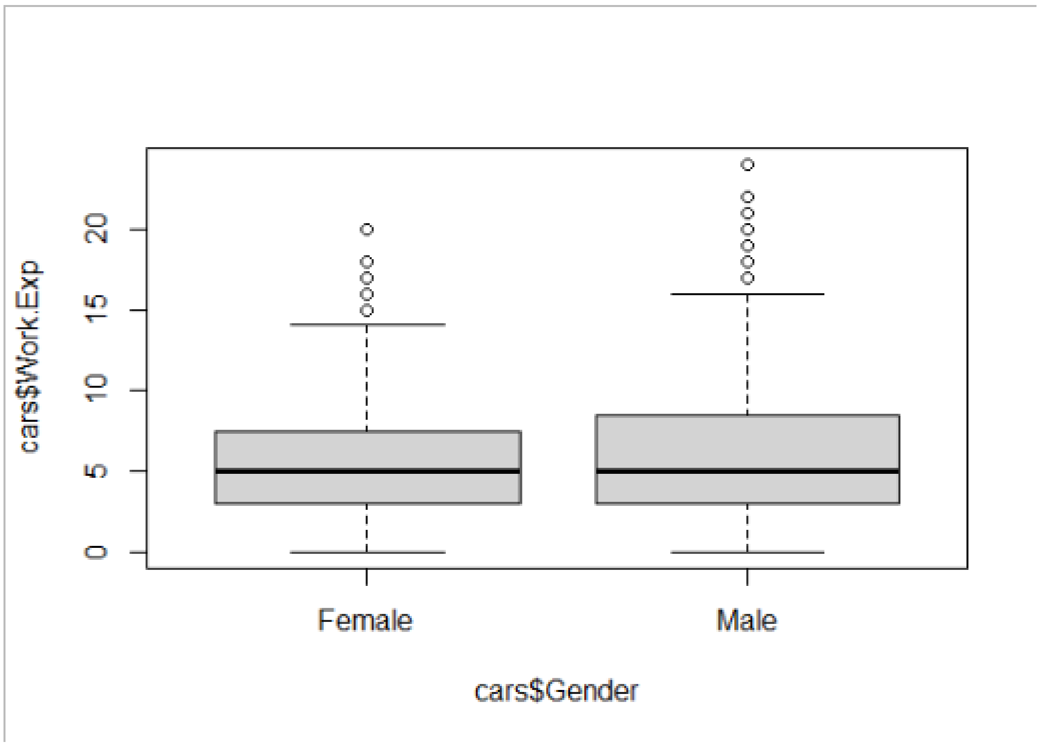


```
ggplot(cars, aes(y = Work.Exp)) + geom_boxplot()
```

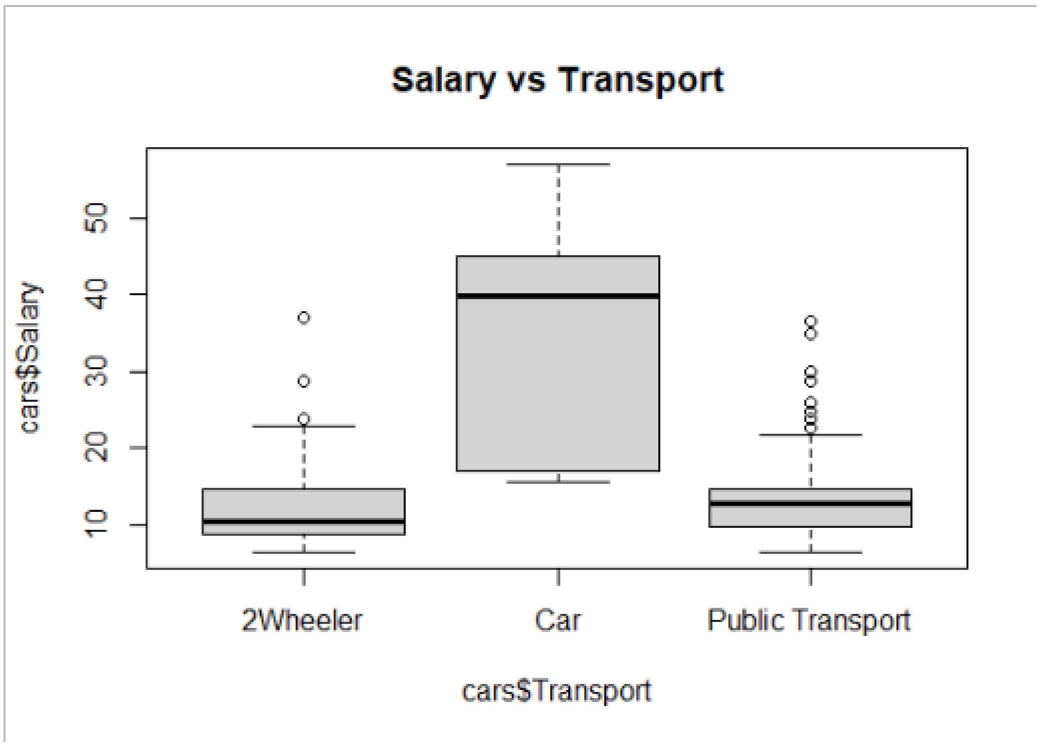


Bi-Variate Analysis:

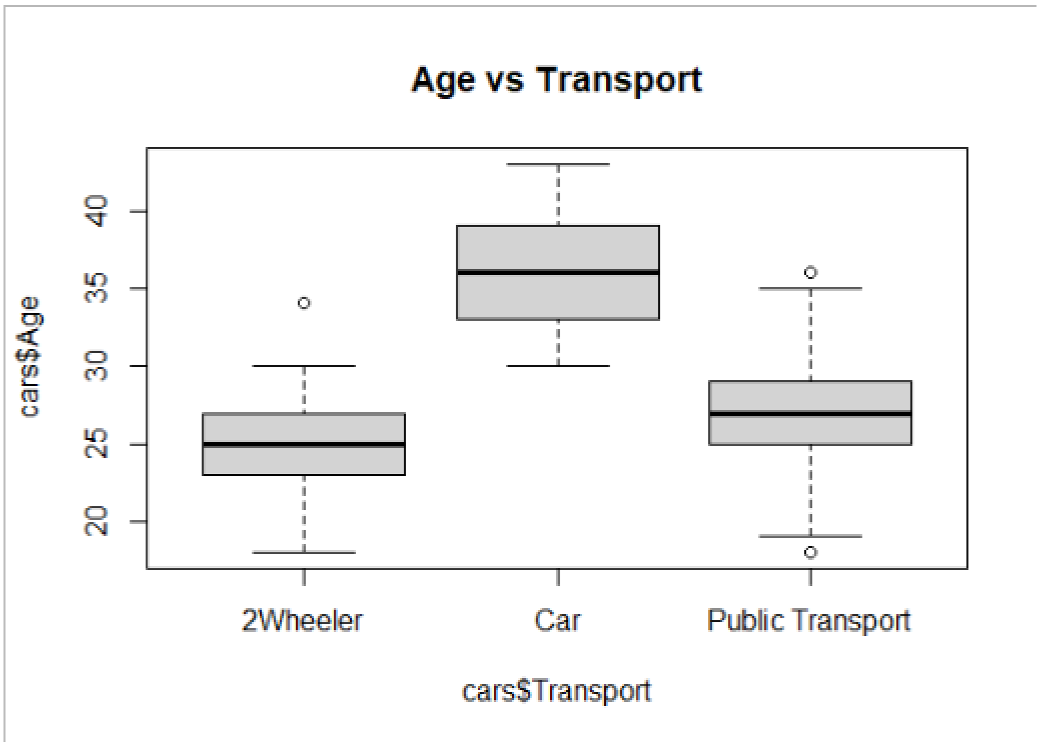
```
boxplot(cars$Work.Exp ~ cars$Gender)
```



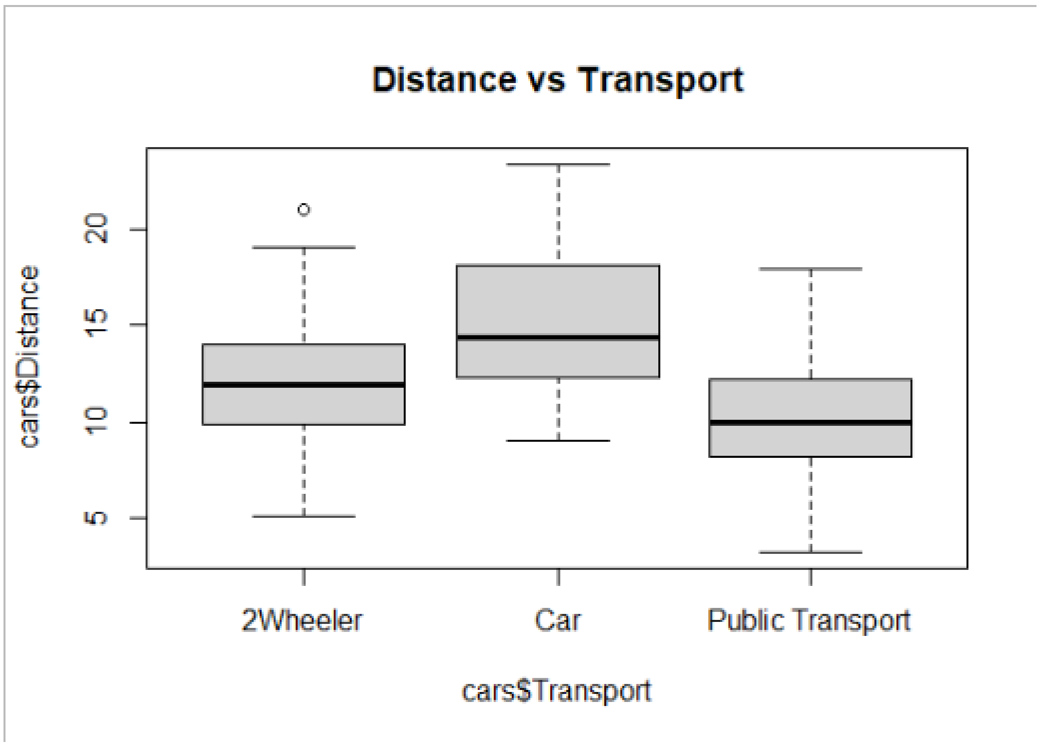
```
boxplot(cars$Salary~cars$Transport, main="Salary vs Transport")
```



```
boxplot(cars$Age~cars$Transport, main="Age vs Transport")
```



```
boxplot(cars$Distance~cars$Transport, main="Distance vs Transport")
```

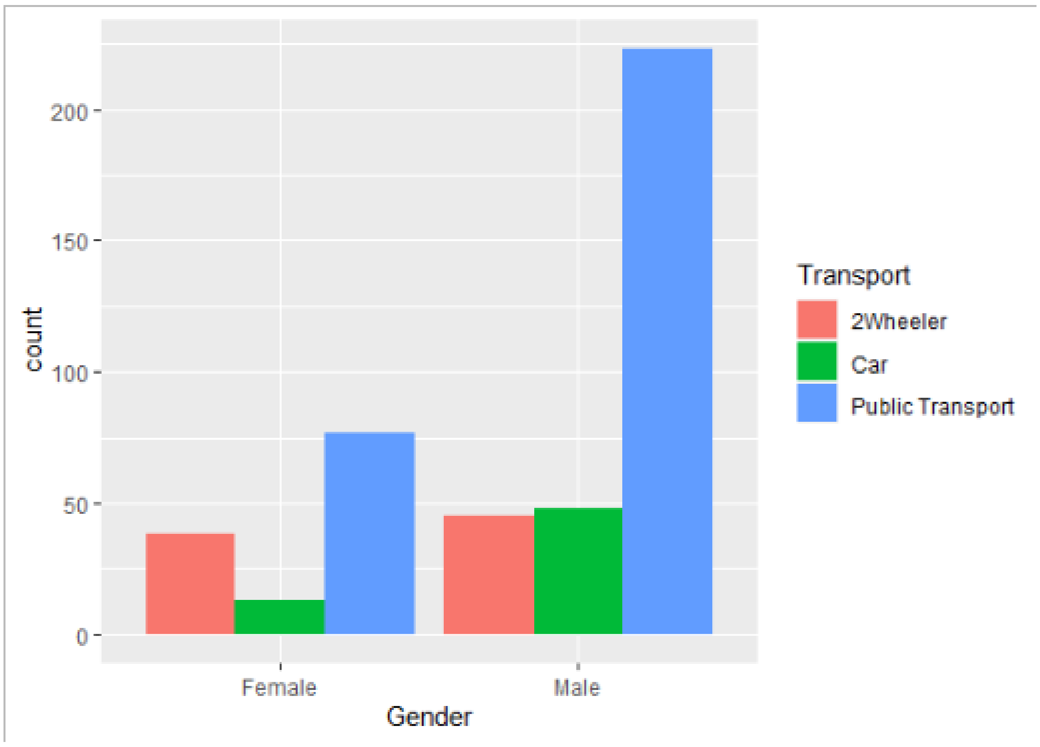


```
table(cars$license,cars$Transport)
2Wheeler Car Public Transport
  0       60  13              267
  1       23  48               33
```

```
cor(cars$Age, cars$Work.Exp)
[1] 0.9322364
```

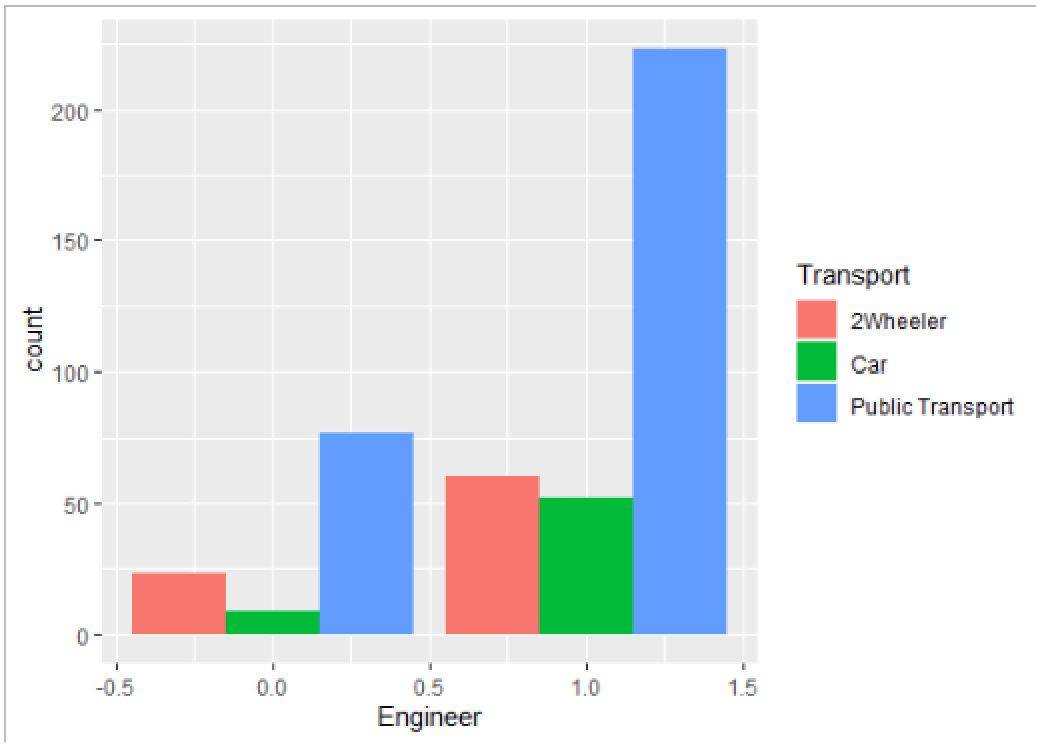
```
table(cars$Gender,cars$Transport)
                  2Wheeler Car Public Transport
  Female      38           13              77
  Male        45           48             223
```

```
ggplot(cars, aes(x = Gender)) + geom_bar(aes(fill = Transport), position = "dodge")
```

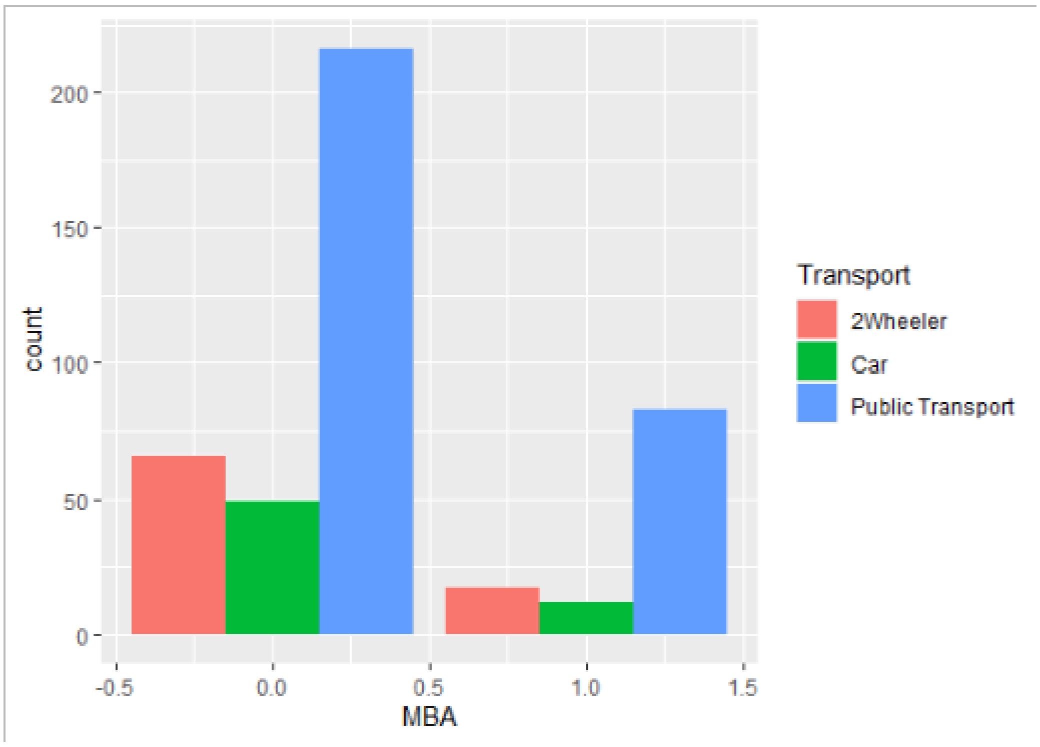


Very few Females use cars compared to Male. Both Male and Female use more of Public Transport. No significant difference due to Gender.

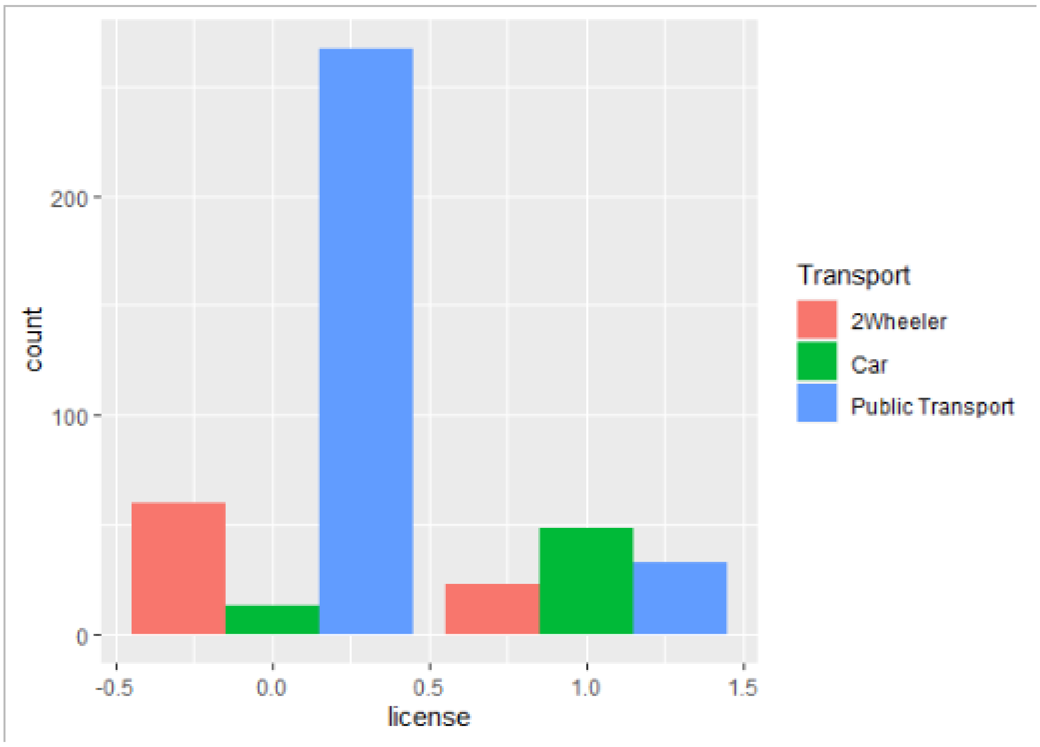
```
ggplot(cars, aes(x = Engineer)) + geom_bar(aes(fill = Transport), position = "dodge")
```



```
ggplot(cars, aes(x = MBA)) + geom_bar(aes(fill = Transport), position = "dodge")
```

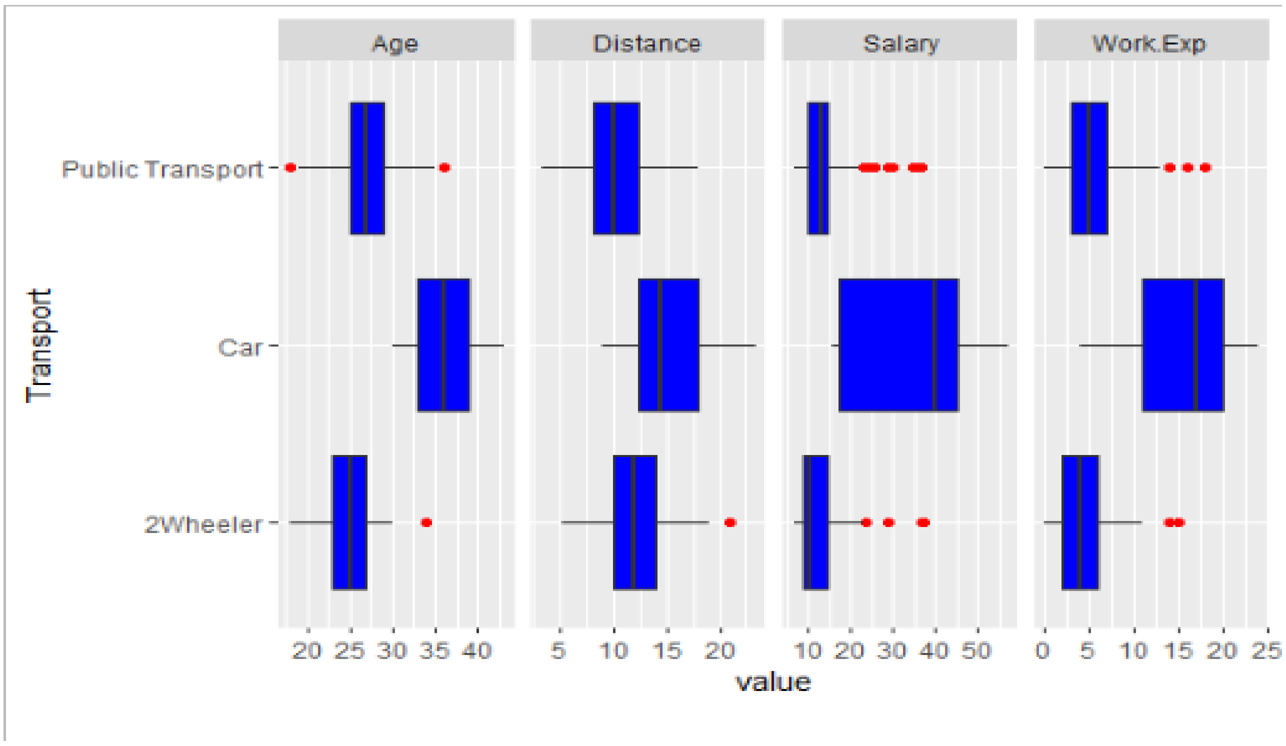


```
ggplot(cars, aes(x = license)) + geom_bar(aes(fill = Transport), position = "dodge")
```



Outlier Treatment:

```
plot_boxplot(cars, by="Transport" , geom_boxplot_args = list("outlier.color" =
"red",fill="blue"))
```



Checking for Outliers:

For all Features, plotted a Box plot to check for Outliers below are the observation and analysis points.

- Age: Outlier: Exist at both end but majorly at right side. Data is evenly skewed which is reflected in plot and graph both.
- Salary: Outlier: Exist at right end. Data is slightly left skewed which is reflected in plot and graph both. Grouping of data can be done based on Histogram.
- Work Experience: Outlier: Exist at right end. Data is slightly right skewed which is reflected in plot and graph both. Grouping of data can be done based on Histogram
- Distance: Few Outlier exist at right end

Missing Values:

We use a sapply to check the number if missing values in each column. There was missing value in MBA data column so it needs to be treated.

```
sapply (cars, function(x) sum(is.na(x)))
```

```
Age    Gender Engineer      MBA Work.Exp    Salary Distance   license Transport
      0          0          0            1          0               0          0            0          0
```

```
cars = knnImputation (cars, 5)
any(is.na(cars))
[1] FALSE
```

KNN is an algorithm that is useful for matching a point with its closed K neighbors in a Multi-Dimensional space. IT can be used that are continuous, discrete, ordinal and categorical which makes it particularly useful for dealing with all kind of missing data. The assumption behind using KNN for missing values is that a point value can be approximated by the values of the points that are closest to it, based on other variables. It is seen here that MBA new logical column has been created.

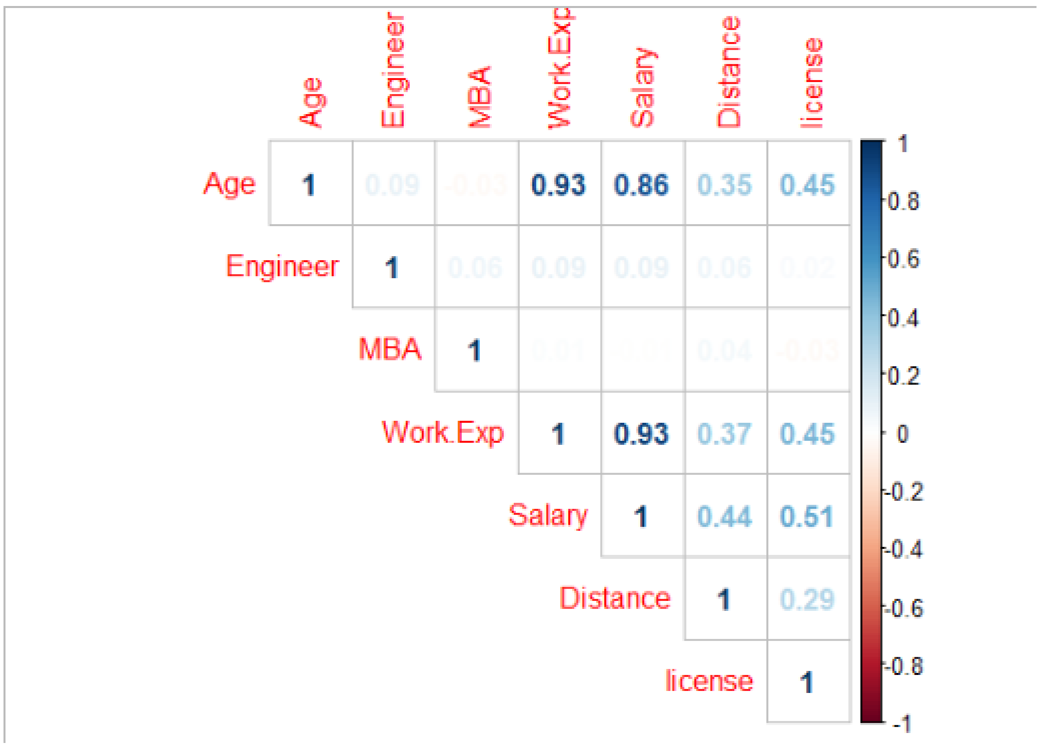
Multicollinearity Check and Treatment:

Let us create a Correlation plot for the given dataset and understand which variables are highly correlated.

```
cars.new = cars %>% select_if(is.numeric)
```

```
a=round(cor(cars.new),2)
```

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



Correlation Interpretation:

- We are using Analysis of significance of features individually using LM model. Above matrix shows the correlation matrix between numerical variables. Highlighted are corelated and either can be removed
- Usage of Variance inflation Factor to assess whether these correlations are really statistically significant or not.
- Multicollinearity exists between variables due to high Variance Inflation Factor values.

Multicollinearity for All Variables:

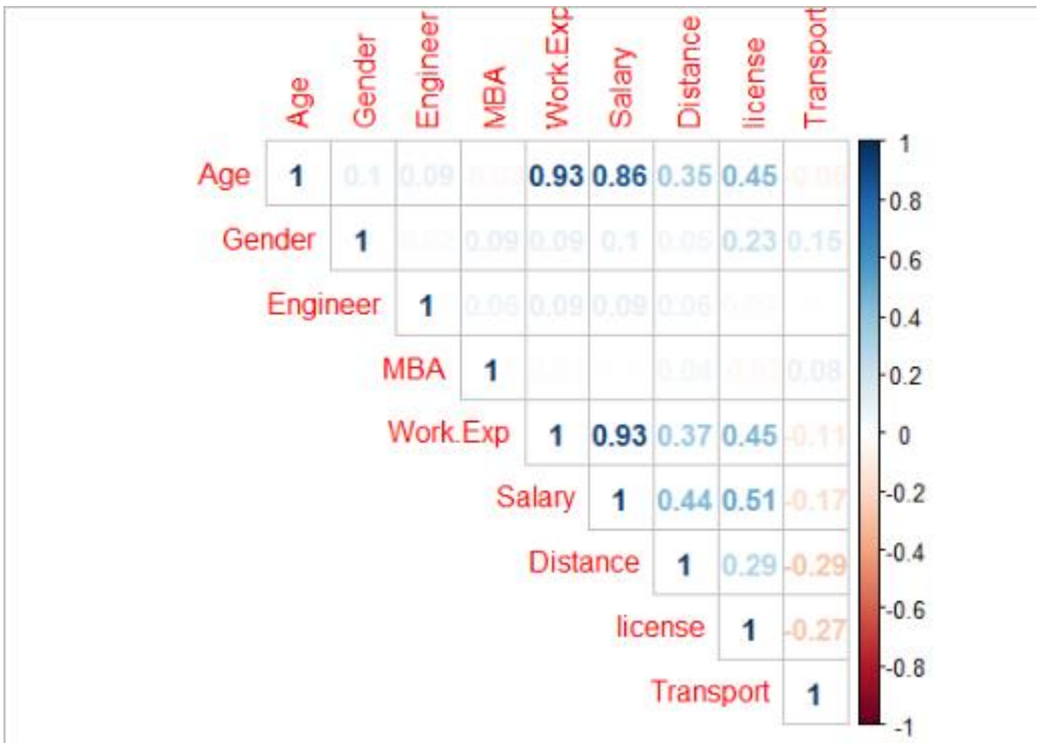
```
carsnew1=cars
```

```
carsnew1$Gender=as.integer(carsnew1$Gender)
```

```
carsnew1$Transport=as.integer(carsnew1$Transport)
```

```
b= round(cor(carsnew1),2)
```

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



Multicollinearity Treatment:

Below are few methods applied on data for treatment of multicollinearity.

- Remove highly correlated predictors from the model. If you have two or more factors with a high VIF, remove one from the model.
- Distance, Age and Salary are statistically significant variables. Remaining all variables are not statistically significant as far as prediction of Car usage is concerned.
- If other variables are not playing significant role the natural thing will be to delete these from further analysis. And recheck the VIF. Let’s remove Work experience and check.
- Post removal, checked the multicollinearity then did data binning and rechecked

```
carsnew1$Work.Exp<-NULL
lModel=lm(Transport~.,data = carsnew1)
summary(lModel)
```

```
Call:
lm(formula = Transport ~ ., data = carsnew1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0604 -0.2633  0.2161  0.4634  1.0810

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.058104   0.375001   2.822  0.00500 **
Age          0.064318   0.015055   4.272 2.38e-05 ***
Gender       0.369527   0.076913   4.804 2.14e-06 ***
Engineer    -0.009348   0.078785  -0.119  0.90560
MBA          0.117394   0.078275   1.500  0.13440
Salary      -0.018538   0.006806  -2.724  0.00671 **
Distance    -0.051246   0.010493  -4.884 1.46e-06 ***
license     -0.546164   0.095090  -5.744 1.74e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7093 on 436 degrees of freedom
Multiple R-squared:  0.208,  Adjusted R-squared:  0.1952
F-statistic: 16.35 on 7 and 436 DF,  p-value: < 2.2e-16
vif(lModel)
Age       Gender       Engineer      MBA   Salary       Distance license
3.893303 1.071184   1.014677   1.019324 4.456966 1.260699 1.431344
```

Now Post treatment i.e. removal of Work Exp, all variables have VIF Value within 5

Key Observations based on EDA Summary are as follows:

- Transport is a target variable as it predicts whether a customer use Car or not.
- Data wrangling can be done for couple of fields like Transport, Salary, Age etc. so to allow for more convenient consumption and organization of the data.
- Based on the model or algorithm used, Flag fields need to be converted from int to Factor
- Outliers noticed in data and treatment done as discussed in earlier sections.
- Multicollinearity exist in data and it is checked by couple of methods and treated by data deletion for less significant variable, data binning etc.

Checking the Proportion of Target Variable in actual Dataset:

```
cars$Transport= ifelse(cars$Transport=='Car',1,0)

table(cars$Transport)

0   1

383  61


sum(cars$Transport==1)/nrow(cars)

0.1373874


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

cars$Engineer=as.factor(cars$Engineer)

cars$MBA=as.factor(cars$MBA)

cars$license=as.factor(cars$license)
```

Checking Target Variable Proportion in Overall Dataset:

```
prop.table(table(cars$Transport)

0                1

0.8626126 0.1373874
```

- **Data Preparation:**

SMOTE

SMOTE: Synthetic Minority Oversampling Technique to handle imbalance in binary classification.

Applying Smote for Data Balancing:

```
balanced.cars=SMOTE(Transport~.,perc.over = 300,cars,k=3,perc.under = 370)
table(balanced.cars$Transport)
```

```
0   1
677 244
```

Step 1: Let’s check structure of Data set Cars.
Step2: To simplify Transport as Car usage, convert to Car 1 or 0 if others.
Step 3: Remove work experience column.
Step 4: Set the seed then partition the data with train and test as 70:30.
Step 5: Check the distribution.
Step 6: Let’s check Car usage rate in both datasets.
Car usage rate is very less in both data set. thus, it falls under minority imbalance data so SMOTE can be used to treat imbalance.

Let’s Create a subset and create Train and Test Dataset:

```
cars2=balanced.cars
```

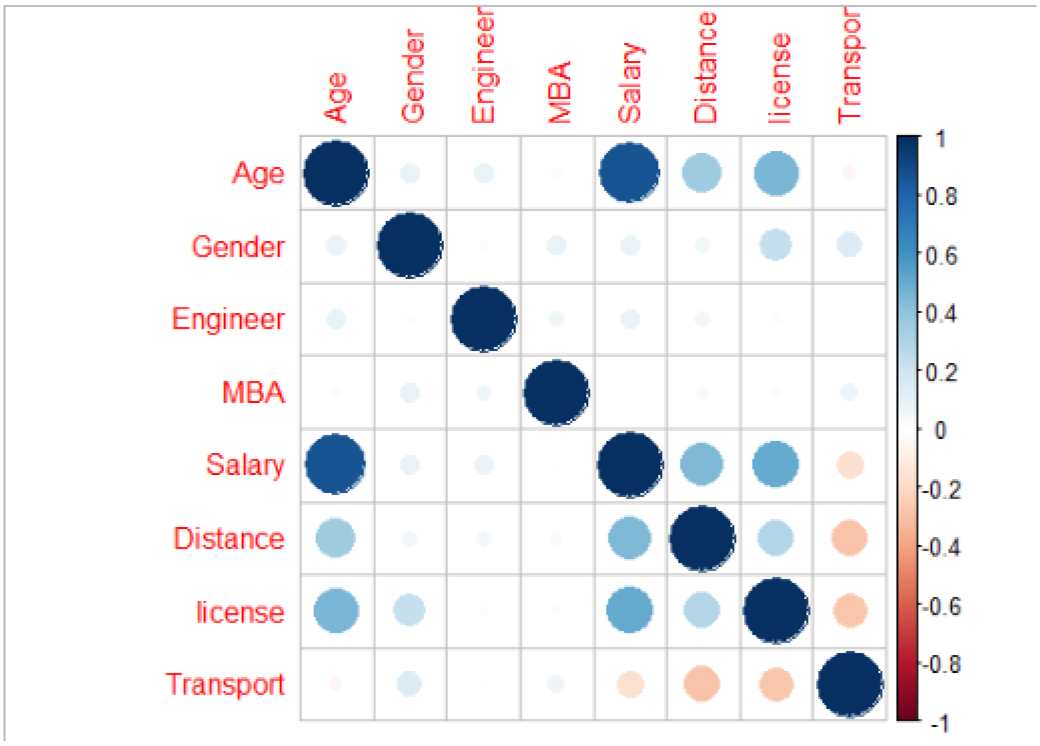
Split the Data into Train and Test:

```
set.seed(500)

spl=sample.split(balanced.cars$Transport,SplitRatio = 0.7)

train=subset(balanced.cars,spl==TRUE)

test=subset(balanced.cars,spl=FALSE)
```

Checking correlation:

Let’s look at the correlation between all variables and treat highly correlated variables accordingly to build the Regression Model.

```
cdata=carsnew1
cdata$Gender=as.integer(cdata$Gender)
cdata$Transport=as.integer(cdata$Transport)
corrplot(cor(cdata))
```



After applying SMOTE, we have now balanced the target variable responder class in the training data.

- **Prediction Models:**

Model Using Logistic Regression Technique on SMOTE Data:

Logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

Building Logistic Regression Model

Will Start Logistic Regression Analysis as it will give us clear insight that what are those variables which are significant in building model so that we can achieve more precision by eliminating irrelevant variables.

Logistic Regression on all Variables

```
logreg=glm(balanced.cars$Transport~.,data=balanced.cars,family=binomial)
summary(logreg)
```

```
Call:
glm(formula = balanced.cars$Transport ~ ., family = binomial,
    data = balanced.cars)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.80341  -0.04585  -0.00681   0.00154   2.23090

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          -66.64574    8.58093  -7.767 8.05e-15 ***
Age                    2.10824    0.28628   7.364 1.78e-13 ***
GenderMale            -1.76073    0.43125  -4.083 4.45e-05 ***
Engineer1             -0.11967    0.45137  -0.265   0.7909
MBA0.375278655473355  -9.36578  840.27466  -0.011   0.9911
MBA1                  -1.07441    0.45055  -2.385   0.0171 *
Work.Exp              -1.05208    0.20265  -5.192 2.08e-07 ***
Salary                 0.18358    0.04635   3.960 7.49e-05 ***
Distance               0.55449    0.08943   6.200 5.64e-10 ***
license1               1.92423    0.41220   4.668 3.04e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1064.95  on 920  degrees of freedom
Residual deviance:  187.56  on 911  degrees of freedom
AIC: 207.56

Number of Fisher Scoring iterations: 14
```

Checking for Logistic Regression Model Multicollinearity

```
vif(logreg)
```

```
GVIF Df GVIF^(1/(2*Df))
Age       10.529273  1         3.244884
Gender     1.192036  1         1.091804
Engineer  1.017589  1         1.008756
MBA        1.284626  2         1.064619
Work.Exp 15.356488  1         3.918735
Salary     3.945175  1         1.986246
Distance  1.779312  1         1.333909
license   1.222617  1         1.105720
```

Interpretation

- The multicollinearity has caused the inflated VIF values for correlated variables, making the model unreliable for model building.
- VIF values for Salary and Work Exp are 3.94 and 15.35 respectively which are not inflated as such
- Being conservative and not considering VIF values above 5, we will remove Salary & Work Exp (Highly Correlated)

Steps for Variable Reduction

- Use full set of explanatory variables.
- Calculate VIF for each variable.
- Remove variable with highest value.
- Recalculates all VIF values for logistic model built with the new set of variables.
- Removes again variable which has highest value, until all values are within threshold.

Start estimating a Logistic Regression Model using the **glm** (generalized linear model) function.

- GLM does not assume a linear relationship between dependent and independent variables. However, it assumes a linear relationship between link function and independent variables in logit model.
- The dependent variable need not to be normally distributed.
- It does not use OLS (Ordinary Least Square) for parameter estimation. Instead, it uses maximum likelihood estimation (MLE).
- Errors need to be independent but not normally distributed.

Logistic Regression after removing highly correlated Variables – Salary & WorkExp

```
logreg2=glm(balanced.cars$Transport~.SalaryWork.Exp,data=balanced.cars,family=binomial)

summary(logreg2)
```

```
Call:
glm(formula = balanced.cars$Transport ~ . - Salary - Work.Exp,
    family = binomial, data = balanced.cars)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.77234  -0.11139  -0.02464   0.00811   2.74679

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -35.51365    3.35467 -10.586  < 2e-16 ***
Age                        0.97301    0.09580  10.157  < 2e-16 ***
GenderMale                -1.29881    0.38095  -3.409 0.000651 ***
Engineer1                 -0.25372    0.41912  -0.605 0.544939
MBA0.375278655473355 -11.21498  840.27437  -0.013 0.989351
MBA1                      -1.49519    0.41382  -3.613 0.000303 ***
Distance                   0.41684    0.06646   6.272 3.56e-10 ***
license1                   1.63049    0.36200   4.504 6.66e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1064.95  on 920  degrees of freedom
Residual deviance:  221.33  on 913  degrees of freedom
AIC: 237.33
```

```
Number of Fisher Scoring iterations: 14
```

```
vif(logreg2)
```

```
GVIF Df GVIF^(1/(2*Df))
Age      1.399467  1        1.182991
Gender   1.061131  1        1.030112
Engineer 1.024984  1        1.012415
MBA      1.208658  2        1.048518
Distance 1.201792  1        1.096263
license  1.054577  1        1.026926
```

Engineer, Distance, Gender and MBA are insignificant, so will remove them as well and create new model based upon the rest of the variables.

<u>Logistic Regression Built after removing all insignificant variables</u>

```
logreg3=glm(balanced.cars$Transport~.-Salary-Work.Exp-MBA-license-Gender-Engineer,data=balanced.cars,family=binomial)

summary(logreg3)
```

```
Call:
glm(formula = balanced.cars$Transport ~ . - Salary - Work.Exp -
    MBA - license - Gender - Engineer, family = binomial, data = balanced.cars)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.64288  -0.14671  -0.04436   0.01780   2.42852

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -33.75287    2.84103 -11.881  < 2e-16 ***
Age           0.89107    0.07948  11.211  < 2e-16 ***
Distance      0.41676    0.06046   6.893 5.48e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1064.95  on 920  degrees of freedom
Residual deviance:  270.08  on 918  degrees of freedom
AIC: 276.08

Number of Fisher Scoring iterations: 8
```

Now based on this new built model we can see that all values are significant, and we can verify the same by checking the multicollinearity as well.

```
vif(logreg3)
Age        Distance
1.101922 1.101922
```

Now we can see that VIF values are within range and all variables are significant and results are making more sense and are in line with the results which we obtained from EDA.

- **Confusion Matrix:**

We will start the model evaluation on train and test data by executing below code and will see that how accurate our model will be in identification of employee who will be performing car as mode of Transport.

Regression Model Performance on Train and Test Data

Confusion Matrix of Train data with 0.5 Threshold

We are Predicting classification of 0 and 1 for each row and then we are putting our actual and predicted into a table to build confusion matrix to check that how accurate our model is by executing below R Code.

```
ctrain=predict(logreg3,newdata=train[,-9],type="response")
tab1=table(train$Transport,ctrain>0.5)
sum(diag(tab1))/sum(tab1)
[1] 0.9333333
```

Confusion Matrix of Test data with 0.5 Threshold

```
ctest=predict(logreg3,newdata=test[,-9],type="response")
tab2=table(test$Transport,ctest>0.5)
sum(diag(tab2))/sum(tab2)
[1] 0.9391965
```

Confusion Matrix Output:

- From Confusion Matrix we can clearly see that our Train data is 93.33% accurate in predicting and Train data is 93.91% accuracy.
- We can see there is slight variation but that is within the range so we can confirm that our model is good model.

ROC

The ROC Curve is the plot between sensitivity and (1-specificity)

(1-specificity is also known as false positive rate and sensitivity is also known as True Positive rate.

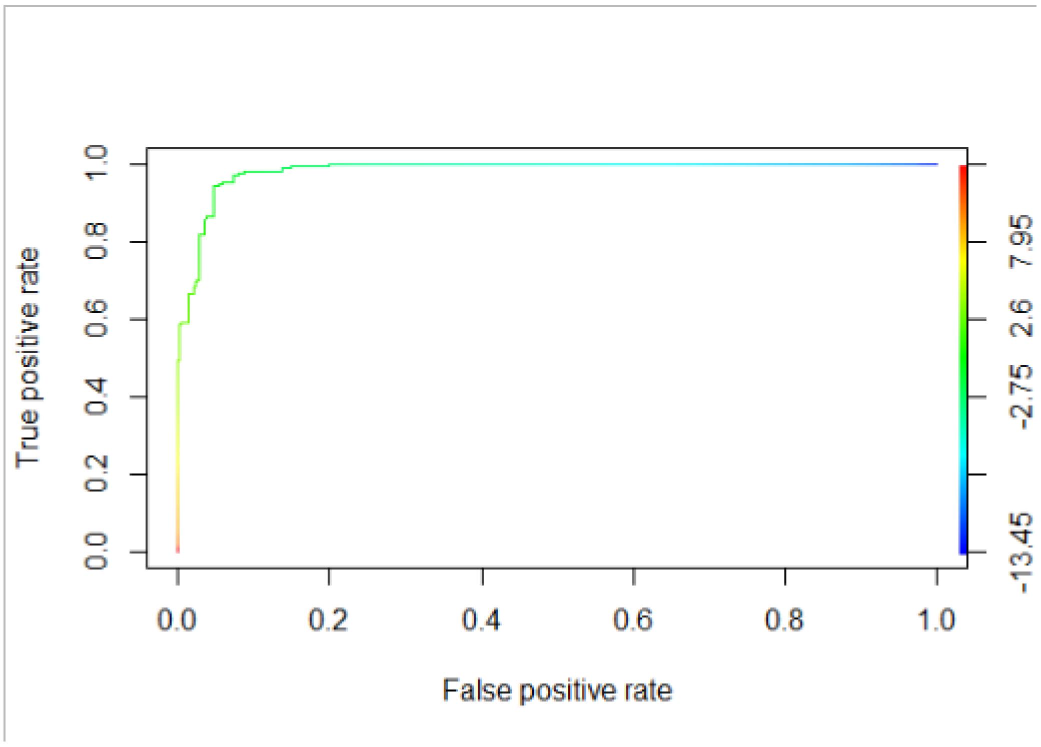
Calculating ROC on Train Data

```
predictroc1=predict(logreg3,newdata=train)
```

```
pred1=prediction(predictroc1,train$Transport)
```

```
perf1=performance(pred1,"tpr","fpr")
```

```
plot(perf1,colorize=T)
```

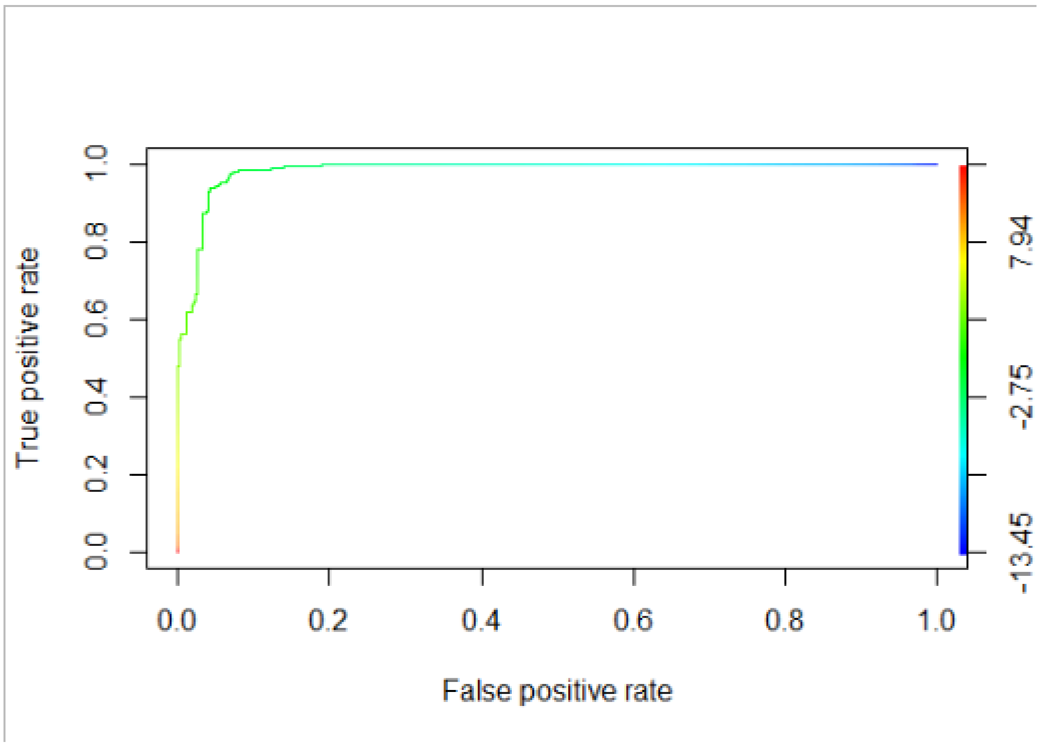


```
as.numeric(performance(pred1,"auc")@y.values)
```

```
[1] 0.9836344
```

Calculating ROC on Test Data

```
predictroc2=predict(logreg3,newdata=test)

pred2=prediction(predictroc2,test$Transport)

perf2=performance(pred2,"tpr","fpr")

plot(perf2,colorize=T)
```



```
as.numeric(performance(pred2,"auc")@y.values)

[1] 0.9841665
```

ROC Output Analysis

- So, from the plot above we can see that plot is covering large area under the curve and we are able to predict on the True Positive side.
- In Train data my True positive rate is 98.36% and in Test data my True Positive rate is 98.41%, so there is no major variation in our test and train data and this proves that our model is more stable.

K-S Chart:

K-S will measure the degree of separation between car users and non-car users
By executing below code on Train and Test model, we will be able to see K-S Analysis result: -

K-S on Train Data

```
KSLRTrain=max(attr(perf1,'y.values')[[1]]-attr(perf1,'x.values')[[1]])

KSLRTrain

[1] 0.8969206
```

K-S on Test Data

```
KSLRTest=max(attr(perf2,'y.values')[[1]]-attr(perf2,'x.values')[[1]])
KSLRTest
[1] 0.907463
```

K-S Output Analysis

- From K- S Analysis we can clearly see that out train data can distinguish between people likely to prefer car or not 89.69% on Train and 90.74% on Test accuracy.
- We can see there is a slight variation but that is within the range so we can confirm that our model is ok.

Gini Chart:

Gini is the ratio between area between the ROC Curve and the diagonal line and the area of the above triangle.

Gini on Train Data

```
giniLRTrain=ineq(ctrain,type="Gini")
giniLRTrain
[1] 0.7030732
```

Gini on Test Data

```
giniLRTest=ineq(ctest,type="Gini")
giniLRTest
[1] 0.7094383
```

Gini Output Analysis

- From Gini analysis we can clearly see that our Train data covering maximum area of car and non-car use employee with 70.30% and test data with 70.94% of accuracy.
- We can see there is a slight variation but that is within the range so we can confirm that our model is ok.

K-NN Classification

k-NN is a supervised learning algorithm. It uses labelled input data to learn a function that produces an appropriate output when given new unlabelled data and also K-NN is executed on numerical variable so below is the structure of data should be converted to numeric.

Loading Original Data

```
cdata1=cars
str(cdata1)
'data.frame':   444 obs. of  9 variables:
 $ Age      : int  28 23 29 28 27 26 28 26 22 27 ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 2 1 2 1 2 2 2 1 2 2 ...
 $ Engineer : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 2 ...
 $ MBA      : Factor w/ 3 levels "0","0.375278655473355",..: 1 1 1 3 1 1 1 1 1 1 ...
 $ Work.Exp : int  4 4 7 5 4 4 5 3 1 4 ...
 $ Salary   : num  14.3 8.3 13.4 13.4 13.4 12.3 14.4 10.5 7.5 13.5 ...
 $ Distance : num  3.2 3.3 4.1 4.5 4.6 4.8 5.1 5.1 5.1 5.2 ...
 $ license  : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
 $ Transport: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Converting all Factor Variable into Integer:

```
cdata1$Gender=as.integer(cdata1$Gender)
cdata1$Transport=as.integer(cdata1$Transport)
cdata1$MBA=as.integer(cdata1$MBA)
cdata1$Engineer=as.integer(cdata1$Engineer)
cdata1$license=as.integer(cdata1$license)

knn.data=cdata1[,-9]
norm.data=scale(knn.data)
usable.data=cbind(Transport=cdata1[,9],norm.data)
usable.data=as.data.frame(usable.data)
```

Splitting data into Test and Train set in 70:30 ratio:

```
set.seed(500)
spl=sample.split(usable.data$Transport,SplitRatio = 0.7)
train1=subset(usable.data,spl==TRUE)
test1=subset(usable.data,spl==FALSE)
```

K-NN Test and Train Data:

```
library(class)
knn_fit_test<- knn(train = train1[,1:8], test = test1[,1:8], cl= train1[,9],k =
3,prob=TRUE)
knn_fit_train<- knn(train = train1[,1:8],  train1[,1:8], cl= train1[,9],k = 3,prob=TRUE)
```

```
table(train1[,9],knn_fit_train)
```

```
knn_fit_train
                    -0.55244314524016         1.80606412866975
  -0.55244314524016              219               18
  1.80606412866975                31               43
```

```
table(test1[,9],knn_fit_test)
```

```
knn_fit_test
                    -0.55244314524016 1.80606412866975
  -0.55244314524016               93               10
  1.80606412866975                14               16
```

Confusion Matrix on Train Data:

```
table.knn3=table(train1[,9],knn_fit_train)
sum(diag(table.knn3))/sum(table.knn3)
```

```
[1] 0.8424437
```

Confusion Matrix on Test Data:

```
table.knn3=table(test1[,9],knn_fit_test)
sum(diag(table.knn3))/sum(table.knn3)
```

```
[1] 0.8195489
```

Confusion Matrix Output

- From Confusion matrix we can clearly see that our Train data is 84.24% accurate in predicting and Train data confirms the same with 81.95% of accuracy.
- We can see there is a slight variation but that is within the range so we can confirm that our model is good model.

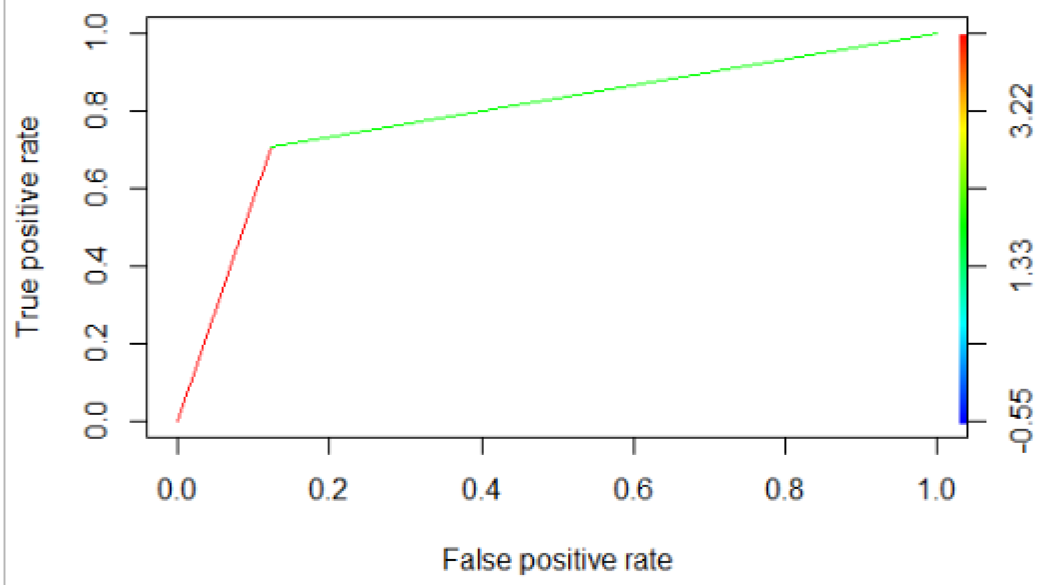
ROC

The ROC Curve is the plot between sensitivity and (1-specificity)

(1- specificity) is also known as false positive rate and sensitivity is also known as True Positive rate.

Calculating ROC on Train Data:

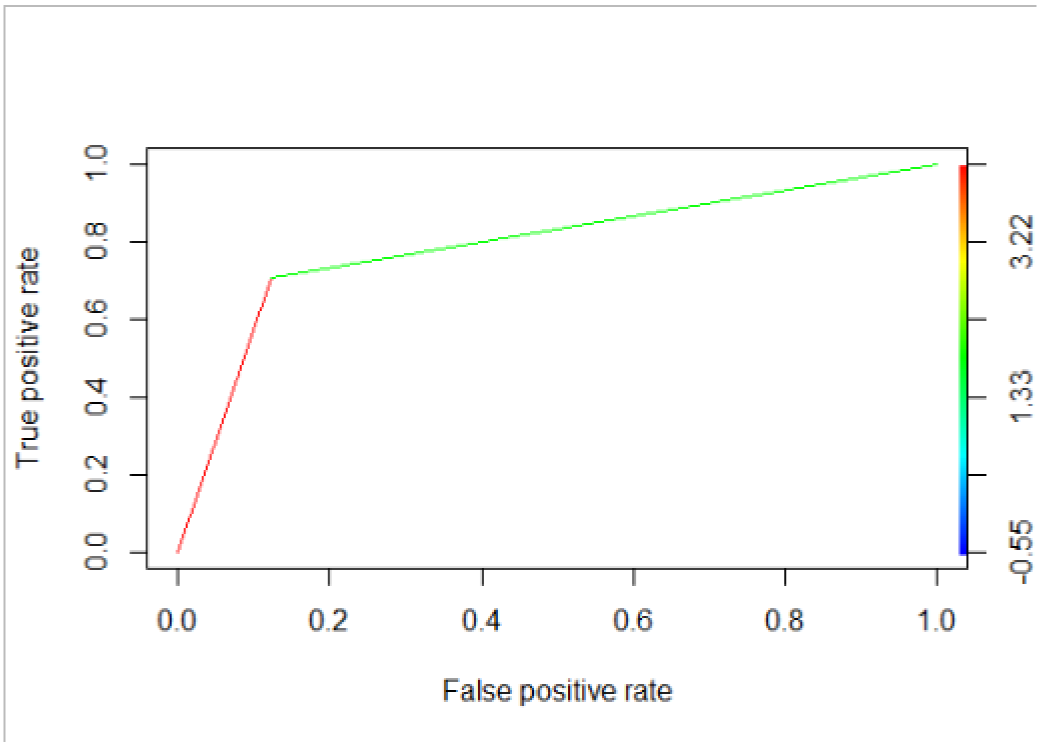
```
predRoc3=ROCR::prediction(train1[,9],knn_fit_train)
perf3=performance(predRoc3,"tpr","fpr")
plot(perf3,colorize=T)
```



```
as.numeric(performance(predRoc3,"auc")@y.values)
[1] 0.790459
```

Calculating ROC on Test Data:

```
predRoc4=ROCR::prediction(test1[,9],knn_fit_test)
perf4=performance(predRoc4,"tpr","fpr")
plot(perf3,colorize=T)
```



```
as.numeric(performance(predRoc4,"auc")@y.values)
```

```
[1] 0.7422717
```

ROC Output Analysis

- So, from the plot we can see that plot is covering large area under the curve and we are able to predict on the True Positive side.
- In Train data my true positive rate is 79.05% and in test data it's 74.22%. so, there is major variation in our Test and Train data, and this proves that our model is stable.

K-S Chart

K-S will measure the degree of separation between car users and non-car users
By executing below code on Train and Test model, we will be able to see K-S Analysis result

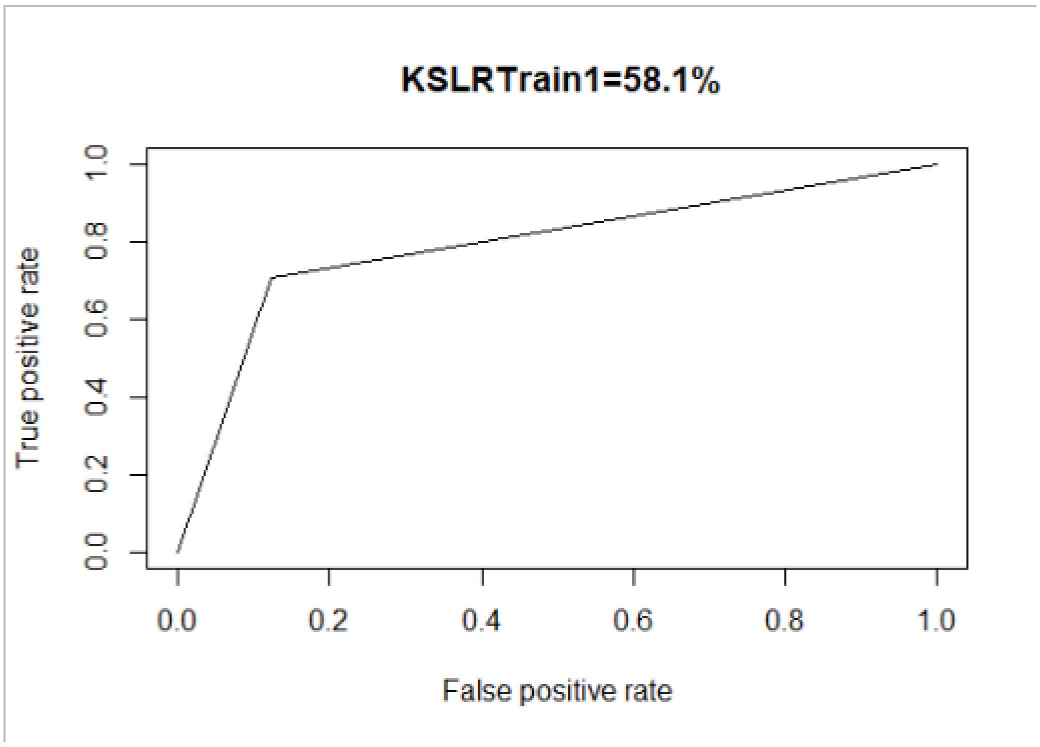
K-S on Train Data:

```
KSLRTrain1=max(attr(perf3,'y.values')[[1]]-attr(perf3,'x.values')[[1]])
```

```
KSLRTrain1
```

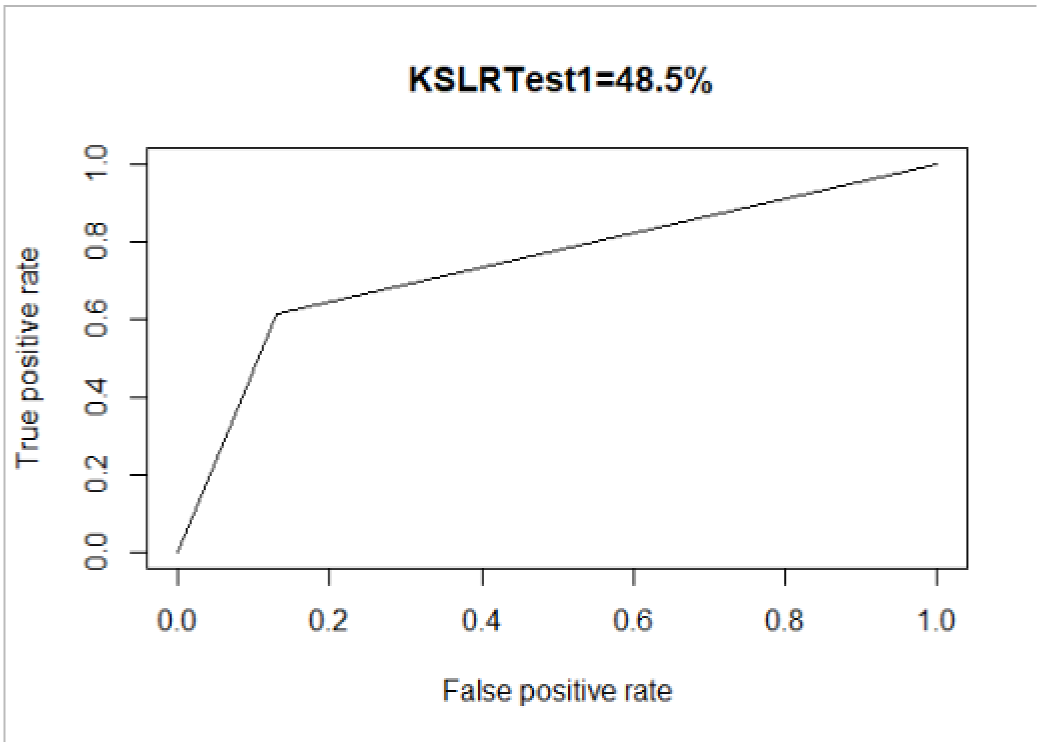
```
[1] 0.580918
```

```
plot(perf3,main=paste0(' KSLRTrain1=',round(KSLRTrain1*100,1),'%'))
```



K-S on Test Data:

```
KSLRTest1=max(attr(perf4,'y.values')[[1]]-attr(perf4,'x.values')[[1]])
KSLRTest1
[1] 0.4845435
plot(perf4,main=paste0(' KSLRTest1=',round(KSLRTest1*100,1),'%'))
```



K-S Output Analysis

- From K-S analysis we can clearly see that our Train data can distinguish between people likely to prefer car or not 58.09% on Train and 48.45% on Test accuracy.
- We can see there is a variation and that is not within the range so we can confirm that our model is not stable.

Gini on Train Data:

Gini is the ratio between area between the ROC curve and the diagonal line & the area of the above triangle.

```
giniKnnTrain=ineq(knn_fit_train,type="Gini")
giniKnnTrain
[1] 0.1318155
```

Gini on Test Data:

```
giniKnnTest=ineq(knn_fit_test,type="Gini")
giniKnnTest
[1] 0.1315553
```

Gini Output Analysis

- From Gini analysis we can clearly see that our Train data not covering maximum area of car and non-car use employee with 13.18% and test data with 13.15% of accuracy.
- We can see there is a slight variation but that is within the range so we can confirm that our model is ok.

Naïve Bayes

Naive Bayes classifier presume that the presence of a feature in a class is unrelated to the presence of any other feature in the same class, so let’s build the model and see how good our model is as per this classification model.

Creating Naïve Bayes Model:

```
NB1=naiveBayes(as.factor(train1$Transport)~.,data=train1,method="class")
NB2=naiveBayes(as.factor(test1$Transport)~.,data=test1,method="class")
```

Performing Classification Model Performance Measures for NB

Confusion Matrix on Train Data

```
predNB1=predict(NB1,newdata=train1,type="class")
table.NB1=table(train1$Transport,predNB1)
sum(diag(table.NB1))/sum(table.NB1)
[1] 0.9421222
```

Confusion Matrix on Test Data

```
predNB2=predict(NB2,newdata=test1,type="class")
table.NB2=table(test1$Transport,predNB2)
sum(diag(table.NB2))/sum(table.NB2)
```

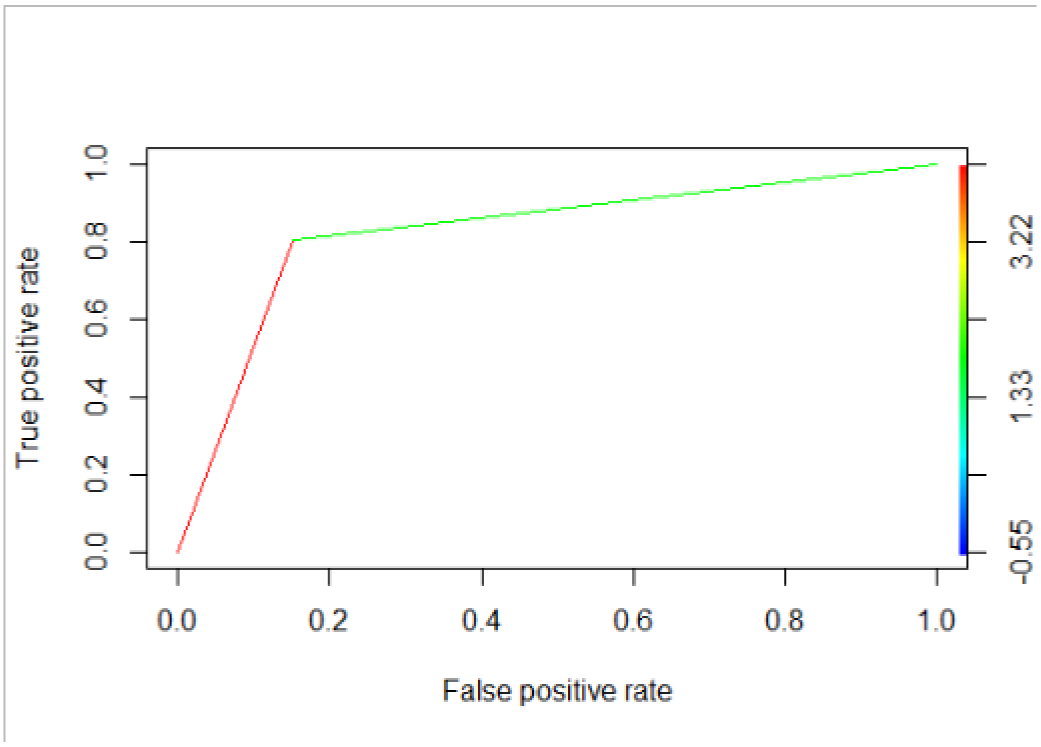
```
[1] 0.962406
```

Confusion Matrix Output

From Confusion matrix we can clearly see that our Train data is 94.21% accurate in predicting and Test data has 96.24% accuracy in prediction the churn rate.

Area Under the ROC Curve on Train Data:

```
predROC7=ROCR::prediction(train1[,9],predNB1)
perf7=performance(predROC7,"tpr","fpr")
plot(perf7,colorize=T)
```

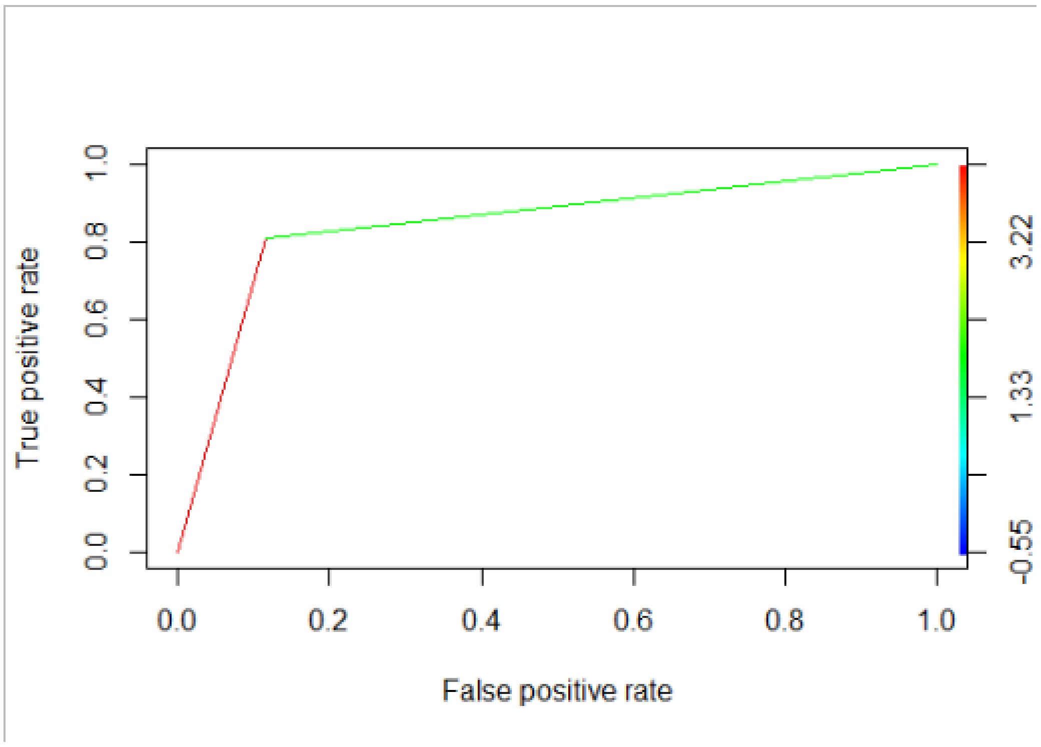


```
as.numeric(performance(predROC7,"auc")@y.values)
```

```
[1] 0.8265131
```

Area Under the ROC Curve on Test Data:

```
predROC8=ROCR::prediction(test1[,9],predNB2)
perf8=performance(predROC8,"tpr","fpr")
plot(perf8,colorize=T)
```



```
as.numeric(performance(predROC8,"auc")@y.values)
[1] 0.8467262
```

ROC Output Analysis

- So, from the plot we can see that plot is covering large area under the curve and we are able to predict on the True Positive side.
- In Train data my true positive rate is 82.65% and in test data it's 84.67%. so, there is major variation in our Test and Train data, and this proves that our model is not stable.

K-S Chart

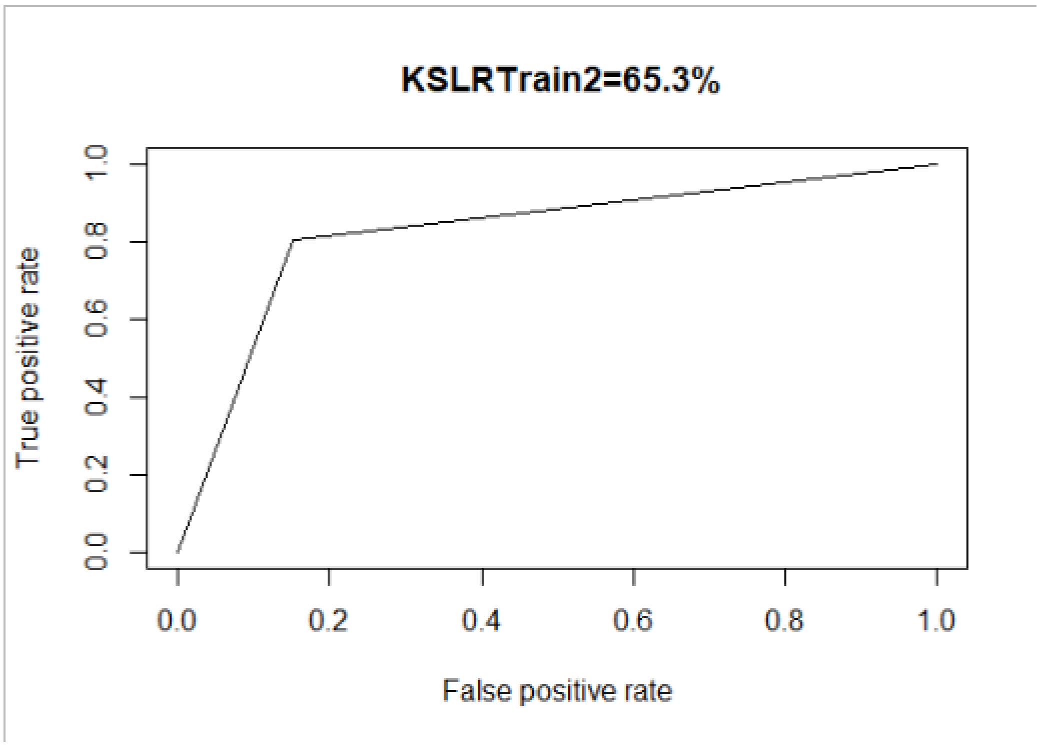
K-S will measure the degree of separation between car users and non-car users
By executing below code on Train and Test model, we will be able to see K-S Analysis result

K-S On Train Data

```
KSLRTrain2=max(attr(perf7,'y.values')[[1]]-attr(perf7,'x.values')[[1]])
KSLRTrain2
```

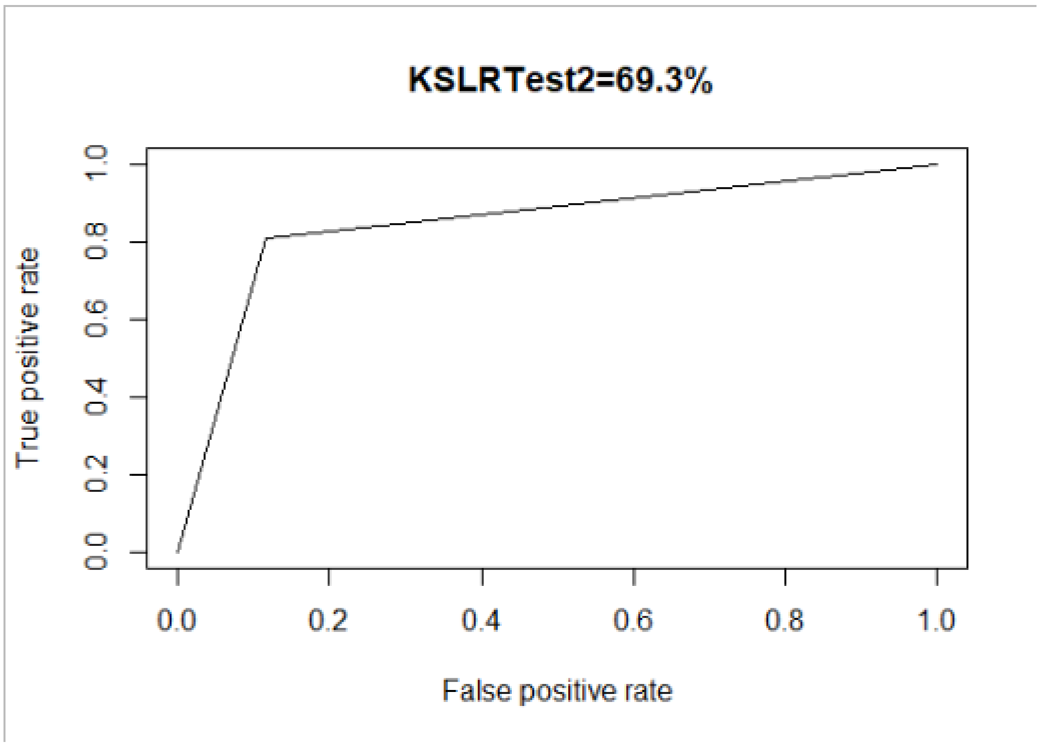
```
[1] 0.6530262
```

```
plot(perf7,main=paste0(' KSLRTrain2=',round(KSLRTrain2*100,1),'%'))
```



K-S On Test Data

```
KSLRTest2=max(attr(perf8,'y.values')[[1]]-attr(perf8,'x.values')[[1]])
KSLRTest2
[1] 0.6934524
plot(perf8,main=paste0(' KSLRTest2=',round(KSLRTest2*100,1),'%'))
```



K-S Output Analysis

From K-S analysis we can clearly see that our Train data can distinguish between people likely to prefer car or not 65.30% on Train and 69.34% on Test accuracy. We can see there is a slight variation and that is within the range so we can confirm that our model is stable.

Remarks on Model Validation “Which is the best Model”

As all 3 models performance measures are quite close but considering accuracy and AUC, Logistic Regression are providing better performance as compared to Naïve Bayes and KNN considering 4 to 5 performance measures.

- ## Prediction using Bagging and Boosting Techniques

Bagging and Boosting:

This method is ensemble method that helps in training the multiple models using the same algorithm and helps in creating the strong learner from weak one.

Bagging:(Aka Bootstrap Aggregating)

It is a way to decrease the variance of prediction by generating additional data for training from your original dataset using combinations with repetitions to produce multisets of the same size as original data.

Boosting:

Way of training the weak learners subsequently.

Applying Bagging Model

```
BAGmodel=bagging(as.numeric(Transport)~.,data=train1,control=rpart.control(maxdepth=10,m
insplit=50))
BAGpredTest=predict(BAGmodel,test1)
tabBAG=table(test1$Transport,BAGpredTest>0.5)
tabBAG
```

```
   TRUE
  1 115
  2   18
```

Interpretation

Bagging is going with baseline approach calling everything as true hence it’s an extreme, hence bagging is going with minority therefore it is not preferable.

Convert the Dependent Variable to numeric

```
train1$Gender=as.numeric(train1$Gender)
train1$Transport=as.numeric(train1$Transport)
train1$Engineer=as.numeric(train1$Engineer)
train1$MBA=as.numeric(train1$MBA)
train1$license=as.numeric(train1$license)
str(train1)
```

```
'data.frame':	311 obs. of  9 variables:
 $ Transport: num  1 1 1 1 1 1 1 1 1 1 ...
 $ Age      : num  -1.075 0.0571 -0.3957 0.0571 -0.1693 ...
 $ Gender   : num  -1.569 -1.569 0.636 0.636 0.636 ...
 $ Engineer : num  0.57 0.57 0.57 0.57 0.57 ...
 $ MBA      : num  -0.583 1.717 -0.583 -0.583 -0.583 ...
 $ Work.Exp : num  -0.45 -0.254 -0.45 -0.254 -0.45 ...
 $ Salary   : num  -0.759 -0.272 -0.377 -0.176 -0.262 ...
 $ Distance : num  -2.22 -1.89 -1.81 -1.73 -1.7 ...
 $ license  : num  -0.552 -0.552 1.806 -0.552 -0.552 ...
```

```
test1$Gender=as.numeric(test1$Gender)
test1$Transport=as.numeric(test1$Transport)
test1$Engineer=as.numeric(test1$Engineer)
test1$MBA=as.numeric(test1$MBA)
test1$license=as.numeric(test1$license)
str(test1)
```

```
'data.frame':	133 obs. of  9 variables:
 $ Transport: num  1 1 1 1 1 1 1 1 1 1 ...
 $ Age      : num  0.0571 0.2835 -0.1693 -0.3957 -1.3014 ...
 $ Gender   : num  0.636 0.636 0.636 -1.569 0.636 ...
 $ Engineer : num  -1.75 0.57 0.57 0.57 0.57 ...
 $ MBA      : num  -0.583 -0.583 -0.583 -0.583 -0.583 ...
 $ Work.Exp : num  -0.45 0.137 -0.45 -0.645 -1.037 ...
 $ Salary   : num  -0.185 -0.272 -0.272 -0.549 -0.836 ...
 $ Distance : num  -2.25 -2 -1.86 -1.73 -1.73 ...
 $ license  : num  -0.552 -0.552 -0.552 -0.552 -0.552 ...
```

Boosting Model

We are using XG Method that is a specialized implementation of Gradient Boosting decision trees designed for performance.

Convert everything to numeric after passing the metrics

XGBoost works with matrixes that contain all numeric variables. Hence changing the data to matrix.

```
features_train1=as.matrix(train1[,1:8])
label_train1=as.matrix(train1[,9])
features_test1=as.matrix(test1[,1:8])
```

XGBoost Model

```
XGBmodel=xgboost (data=features_train1, label=label_train1,
eta=.01,
max_depth=5,
min_child_weight=3,
nrounds=10,
nfold=5,
objective="reg:linear",verbose = 0,
early_stopping_rounds = 10)
```

Functions used in above Model

- Eta=A learning rate at which the values are updated, it’s a slow learning rate
- Max_depth=Represents how many nodes to expand the trees. Larger the depth, more complex the model, higher chances of overfitting. There is no standard value for max_depth. Larger data sets require deep trees to learn the rules from data.
- Min_child_weight= It blocks the potential feature interactions to prevent overfitting
- Nrounds=controls the maximum number of iterations. For classification, its similar to number of trees to grow
- nfold=used for cross validation
- Verbose=do not want to see the output printed
- early_stopping_rounds=stop if no improvement for 10 consecutive trees

Confusion Matrix Output

```
XGBpredTest=predict(XGBmodel,features_test1)
tabXGB=table(test1$Transport,XGBpredTest>0.5)
tabXGB
```

```
      FALSE TRUE
  1   112      3
  2     0     18
```

Interpretation

Shows a prediction of 100% accuracy that the customers are using cars

This model same as bagging and therefore is a proper representation of both majority and minority class.

Actionable Insights and Recommendations

- Logistic Regression Model,K-NN and Naïve Bayes models are all able to predict the transport mode with very high accuracy.
- However, using Bagging and Boosting we can predict the choice of transport mode with 100%accuracy.
- In this case any of the models Logistic Regression,K-NN,Naïve Bayes or Bagging/Boosting can be used for high accuracy prediction.
- However, the key aspect is SMOTE for balancing the minority and majority class, without which our models will not be so accurate.

***************************************THANKYOU***