Cars Case Study

Benedict Egwuchukwu

10/2/2020

Contents

1. Project Objective

This project requires an understanding of what mode of transport employees prefers to commute to their office. The dataset "Cars-dataset" 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.

The following will be carried out through the assessment.

- Perform an EDA on the data
- Illustrate the insights based on EDA
- What is the most challenging aspect of this problem? What method will you use to deal with this? Comment
- Prepare the data for analysis
- Create multiple models and explore how each model perform using appropriate model performance metrics
  - KNN
  - Naive Bayes (is it applicable here? comment and if it is not applicable, how can you build an NB model in this case?)
  - 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.
- Summarize your findings from the exercise in a concise yet actionable note

2. Exploratory Data Analysis (EDA) - Step by step approach

2.1 Environment Set up and Data Import

```
# Environment set up and data import

# Invoking libraries
library(readr) # To import csv files
library(ggplot2) # To create plots
library(corrplot) # To plot correlation plot between numerical variables
library(gridExtra) # To plot multiple ggplot graphs in a grid
library(DataExplorer) # visual exploration of data
library(caTools) # Split Data into Test and Train Set
library(caret) # for confusion matrix function
library(randomForest) # to build a random forest model
library(rpart) # to build a decision model
library(rattle)
library(gbm) # basic implementation using AdaBoost
library(xgboost) # to build a XGboost model
library(DMwR) # for sMOTE
library(knitr) # Necessary to generate source codes from a .Rmd File
library(markdown) # To convert to HTML
library(rmarkdown) # To convret analyses into high quality documents
```

2.1.1 Install necessary packages and load libraries

```
# Set working directory
setwd("C:/Users/egwuc/Desktop/PGP-DSBA-UT Austin/Machine Learning/Week 5 - Project/")
```

2.1.2 Set up Working Directory

```
# Read input file
cars_dataset <- read.csv("Cars-dataset.csv")
```

2.1.3 Import and Read the Dataset

```
# Global options settings
options(scipen = 999) # turn off scientific notation like 1e+06
```

2.1.4 Global Options Settings

2.2 Variable Identification

In order for us to get familiar with the Cardio Good Fitness data, we would be using the following functions to get an overview

1. dim(): this gives us the dimension of the dataset provided. Knowing the data dimension gives us an idea of how large the data is. 2. head(): this shows the first 6 rows(observations) of the dataset. It is essential for us to get a glimpse of the dataset in a tabular format without revealing the entire dataset if we are to properly analyse the data.
2. tail(): this shows the last 6 rows(observations) of the dataset. Knowing what the dataset looks like at the end rows also helps us ensure the data is consistent.
3. str(): this shows us the structure of the dataset. It helps us determine the datatypes of the features and identify if there are datatype mismatches, so that we handle these ASAP to avoid inappropriate results from our analysis.
4. summary(): this provides statistical summaries of the dataset. This function is important as we can quickly get statistical summaries (mean,median, quartiles, min, frequencies/counts, max values etc.) which can help us derive insights even before diving deep into the data.
5. View(): helps to look at the entire dataset at a glance.

```
# Check dimension of dataset
dim(cars_dataset)
```

2.2.1 Insight(s) from dim():

```
## [1] 418   9
```

- The dataset has 418 rows and 9 columns.

```
# Check first 6 rows(observations) of dataset
head(cars_dataset)
```

2.2.2 Insight(s) from head() and tail():

```
##   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
```

```r
tail(cars_dataset)
```

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

- There are 9 variables.
- Columns names are appropriate.
- Values in all fields are consistent in each column.

```r
# Check structure of dataset
str(cars_dataset)
```

2.2.3 Insight(s) from str():

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

- Age, Engineer, MBA, Work.Exp and License are integer variables.
- Salary and Distance are numerical variables.
- Gender and Transport are factor variables.

```r
# Get summary of dataset
summary(cars_dataset)
```

2.2.4 Insight(s) from summary():

```
##       Age           Gender       Engineer           MBA
##  Min.   :18.00   Female:121   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:25.00   Male  :297   1st Qu.:0.2500   1st Qu.:0.0000
##  Median :27.00                Median :1.0000   Median :0.0000
##  Mean   :27.33                Mean   :0.7488   Mean   :0.2614
##  3rd Qu.:29.00                3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :43.00                Max.   :1.0000   Max.   :1.0000
##                                                NA's   :1
##     Work.Exp          Salary          Distance         license
##  Min.   : 0.000   Min.   : 6.500   Min.   : 3.20   Min.   :0.0000
##  1st Qu.: 3.000   1st Qu.: 9.625   1st Qu.: 8.60   1st Qu.:0.0000
##  Median : 5.000   Median :13.000   Median :10.90   Median :0.0000
```

```
##  Mean   : 5.873   Mean   :15.418   Mean   :11.29   Mean   :0.2033
##  3rd Qu.: 8.000   3rd Qu.:14.900   3rd Qu.:13.57   3rd Qu.:0.0000
##  Max.   :24.000   Max.   :57.000   Max.   :23.40   Max.   :1.0000
##
##             Transport
##  2Wheeler        : 83
##  Car             : 35
##  Public Transport:300
##
##
##
##
```

- The age variable ranges from a minimum value of 18 to a maximum value of 43 with a mean and median of 27.3 and 27.0 respectively.
- In terms of gender, female accounted for 121 while male accounted for 297.
- Number of employees with engineering degree indicated with 1 is 313 while those without engineering degree indicated with 0 is 105.
- Number of employees with MBA indicated with 1 is 109 while those without MBA indicated with 0 is 308. There is a missing value which must be treated.
- Work experience in years ranges from a minimum value of 0 to a maximum value of 24. Mean and median is 5.9 and 5.0 respectively.
- Annual salary of employees (in thousand) ranges from a minimum value of 6.5 to a maximum value of 57. Mean and median is 15.4 and 13.0 respectively.
- Distance from office (in KM) ranges from a minimum value of 3.2 to a maximum value of 23.4. Mean and median is 11.3 and 10.9 respectively.
- Number of employees with a license inidcated with 1 is 85 while those without a license indicated with 0 is 333.
- The transport variable is divided into 3 namely "2Wheeler", "Car" and "Public Transport". 71.8% of employees use Public Transport, 19.9% use 2Wheeler while 8.3% use car.

```r
# How many missing vaues do we have?
sum(is.na(cars_dataset))
```

2.2.5 Missing Data Treatment

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

```r
# What columns contain missing values?
colSums(is.na(cars_dataset))
```

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

```r
# Impute the missing value with the column mean/median
data1 = cars_dataset
data1$MBA[is.na(data1$MBA)] <- median(data1$MBA, na.rm = T)
dim(data1)
```

```
## [1] 418   9
```

```r
cars_dataset <- data1
sum(is.na(cars_dataset))
```

```
## [1] 0
```

```r
# Change Engineer, MBA and license to factor variable
cars_dataset$Engineer <- as.factor(cars_dataset$Engineer)
cars_dataset$MBA <- as.factor(cars_dataset$MBA)
cars_dataset$license <- as.factor(cars_dataset$license)
```

```r
# View the dataset
View(cars_dataset)
```

2.2.7 Insight(s) from View():

- The dataset shows employee information about their mode of transport as well as their personal and professional details.

2.3 Univariate Analysis

```r
# Distribution of the dependent variable
prop.table(table(cars_dataset$Transport))*100
```

```
##
##         2Wheeler              Car Public Transport
##        19.856459         8.373206        71.770335
```

- Under the transport variable, 71.77% of employees use public transport, 19.86% use 2Wheeler while 8.37% use car as a mode of transport.

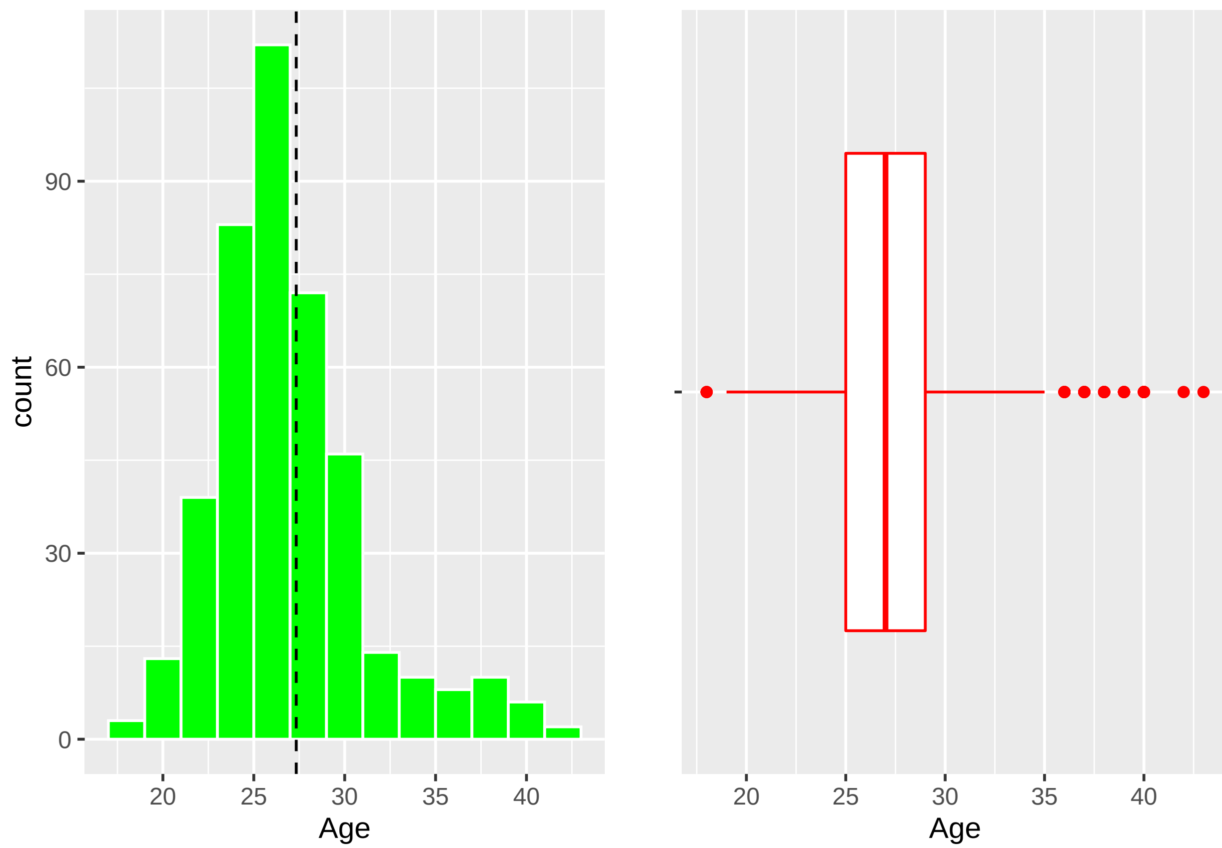
```r
plot_histogram_n_boxplot = function(variable, variableNameString, binw){

  a <- ggplot(data = cars_dataset, aes(x = variable)) +
    labs(x = variableNameString, y = 'count')+
    geom_histogram(fill = 'green', col = 'white', binwidth = binw) +
    geom_vline(aes(xintercept = mean(variable)),
               color = "black", linetype = "dashed", size = 0.5)

  b <- ggplot(data = cars_dataset, aes('',variable))+
    geom_boxplot(outlier.colour = 'red', col = 'red', outlier.shape = 19)+
    labs(x = '', y = variableNameString) + coord_flip()
  grid.arrange(a,b,ncol = 2)
}
```

1. Observations on Age

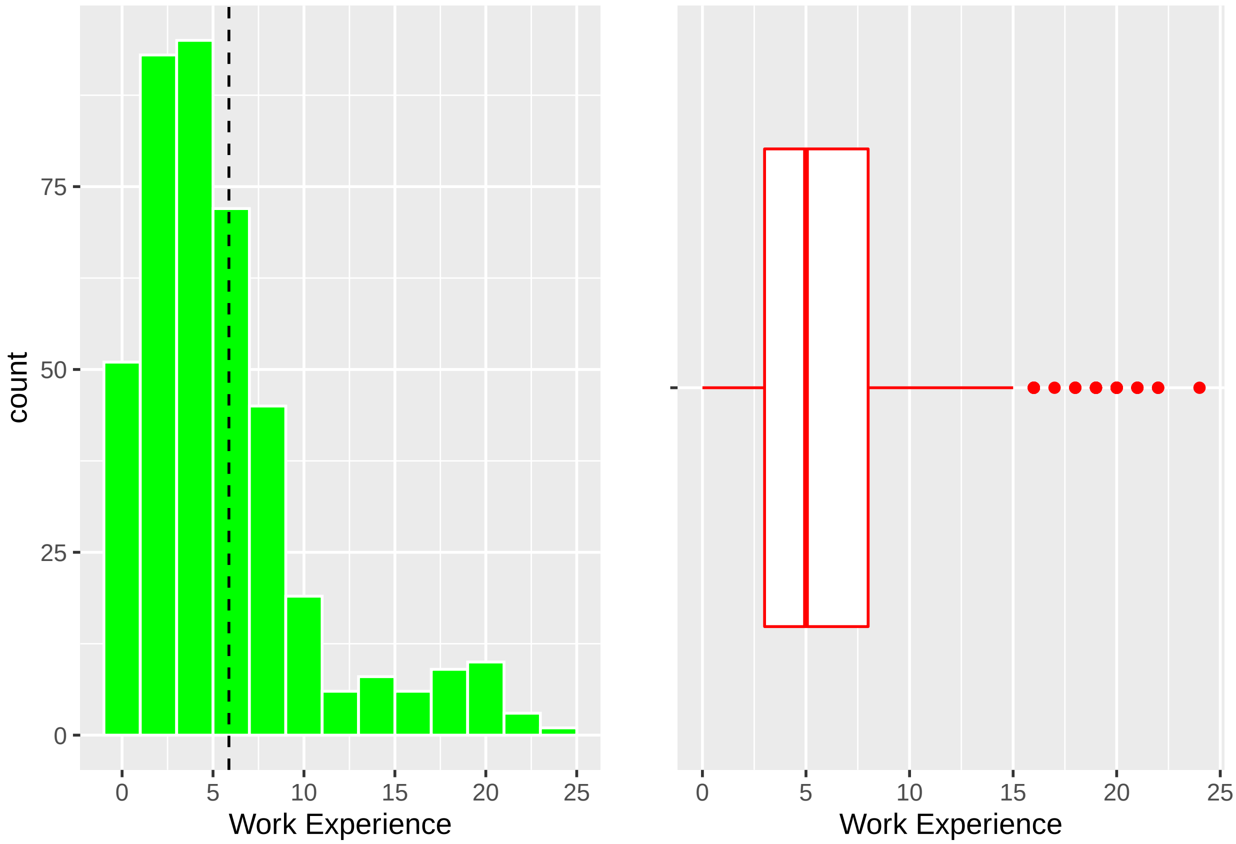
```r
plot_histogram_n_boxplot(cars_dataset$Age, 'Age', 2)
```



- It is slightly uniform though it is skewed to the right. There are few outliers.

2. Observations on WorkExp

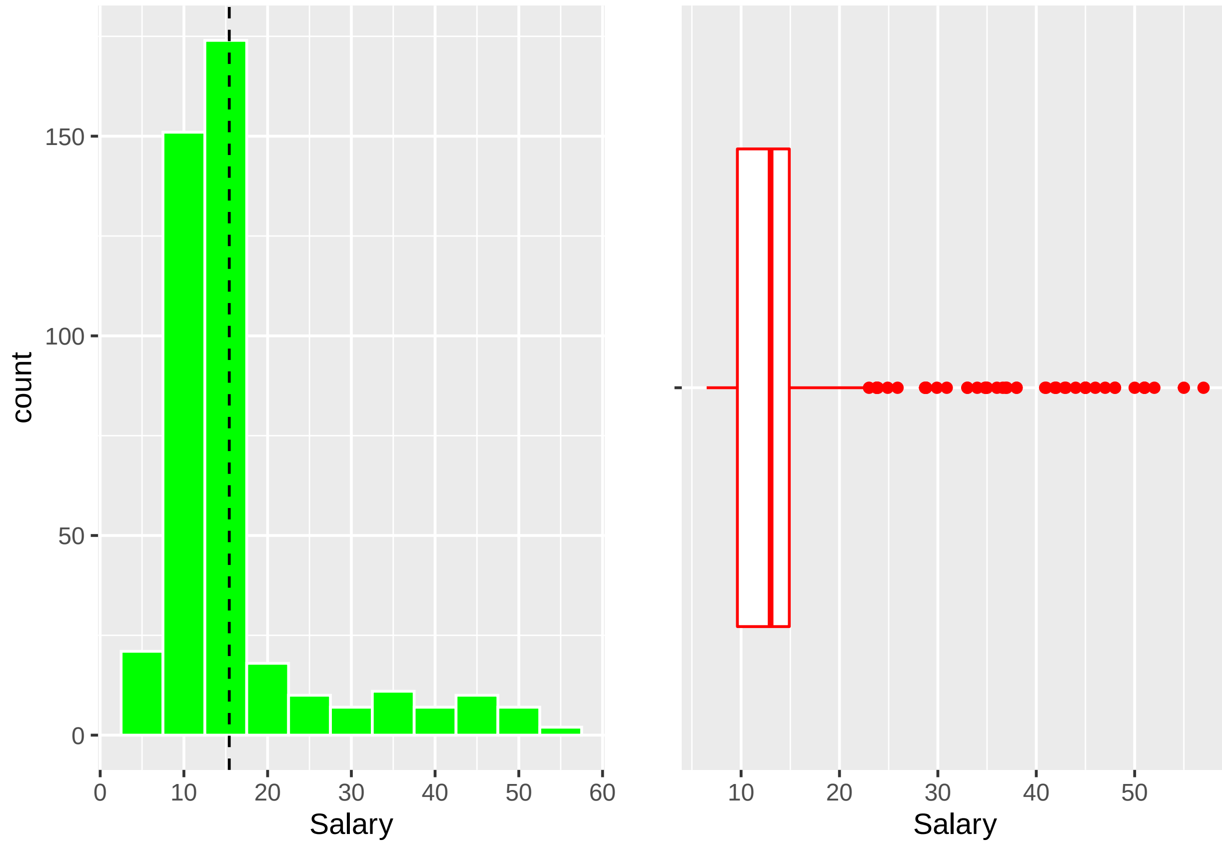
```
plot_histogram_n_boxplot(cars_dataset$Work.Exp, 'Work Experience', 2)
```



- It is skewed to the right. There are few outliers.

3. Observations on Salary

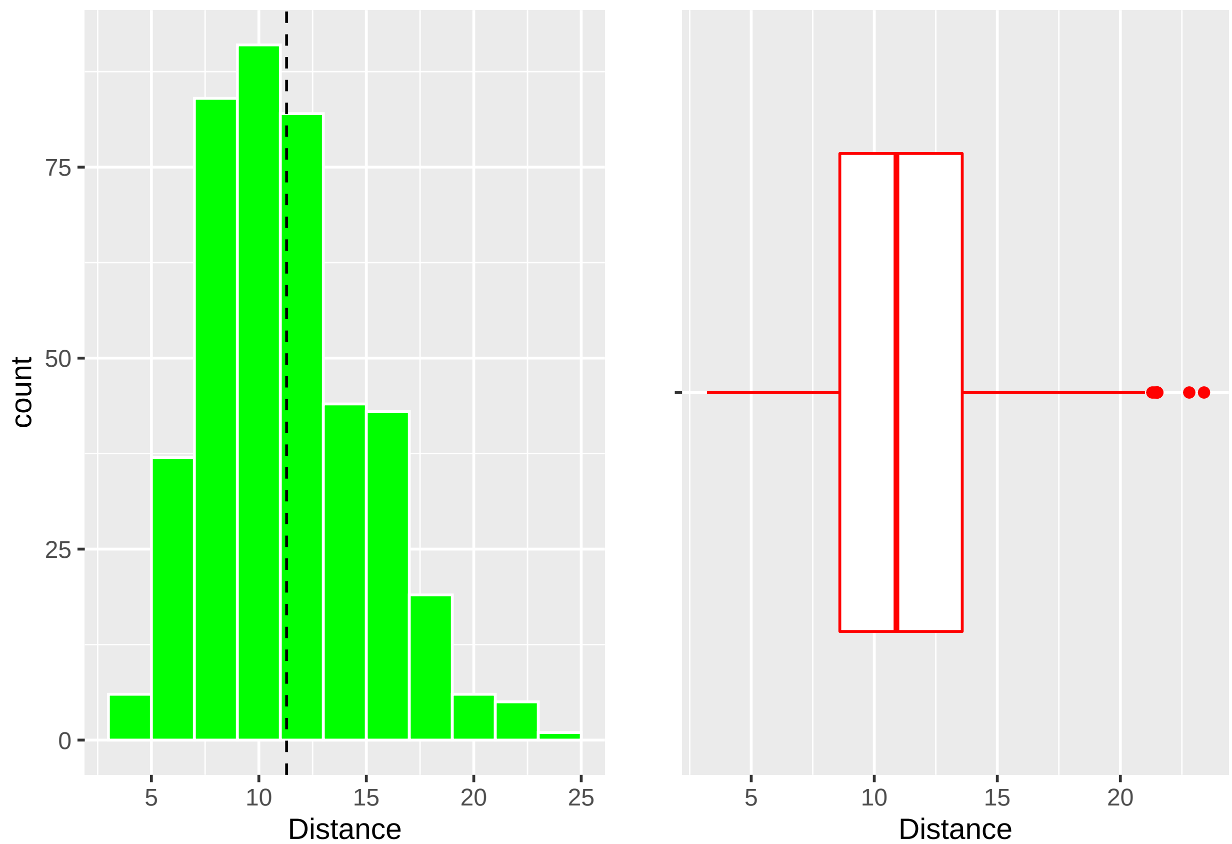
```
plot_histogram_n_boxplot(cars_dataset$Salary, 'Salary', 5)
```



- It is skewed to the right. There are numerous outliers.

4. Observations on Distance

```
plot_histogram_n_boxplot(cars_dataset$Distance, 'Distance', 2)
```



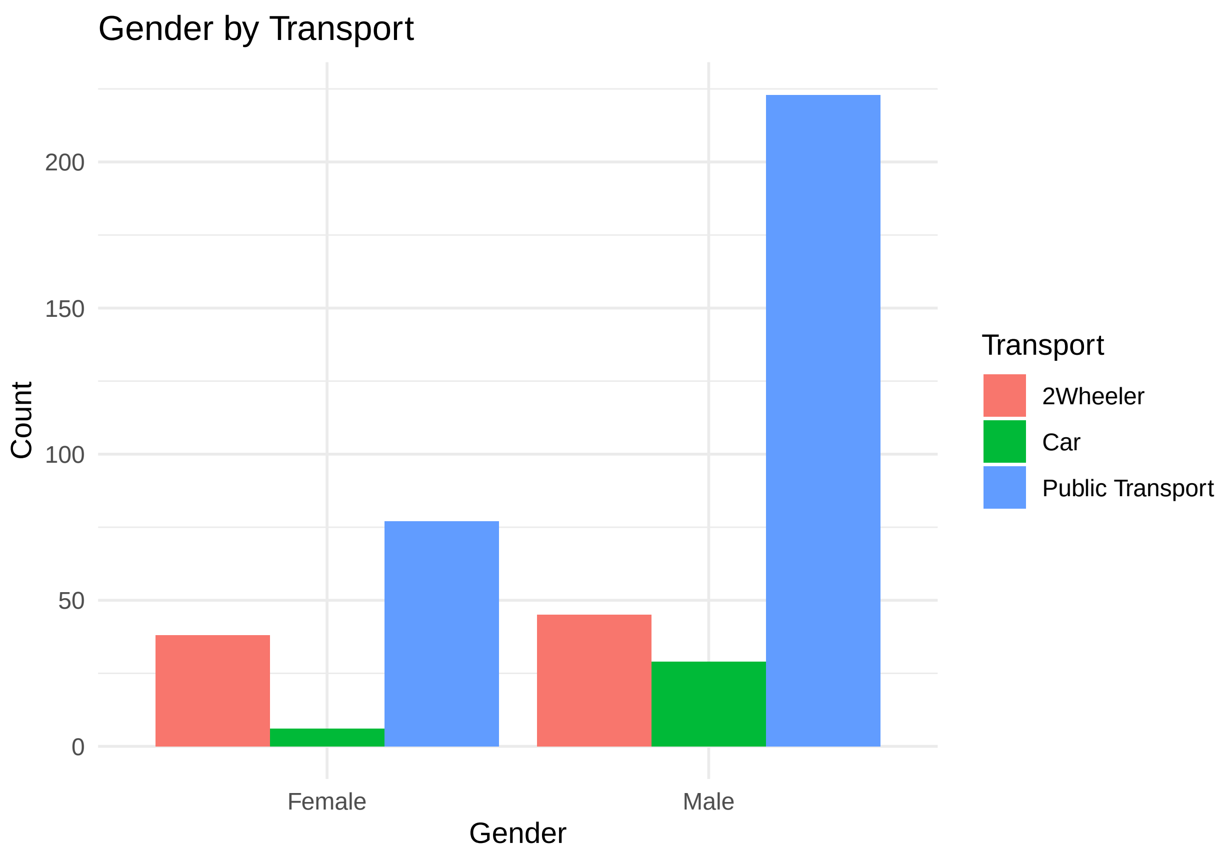
- There is a uniform distribution. There are few outliers.

2.4 Bivariate Analysis

Plot bivariate charts between variables to understand their relationship with each other.

1. Relationship between Transport and Gender

```
ggplot(cars_dataset, aes(x = Gender, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "Gender",
       title = "Gender by Transport") +
  theme_minimal()
```

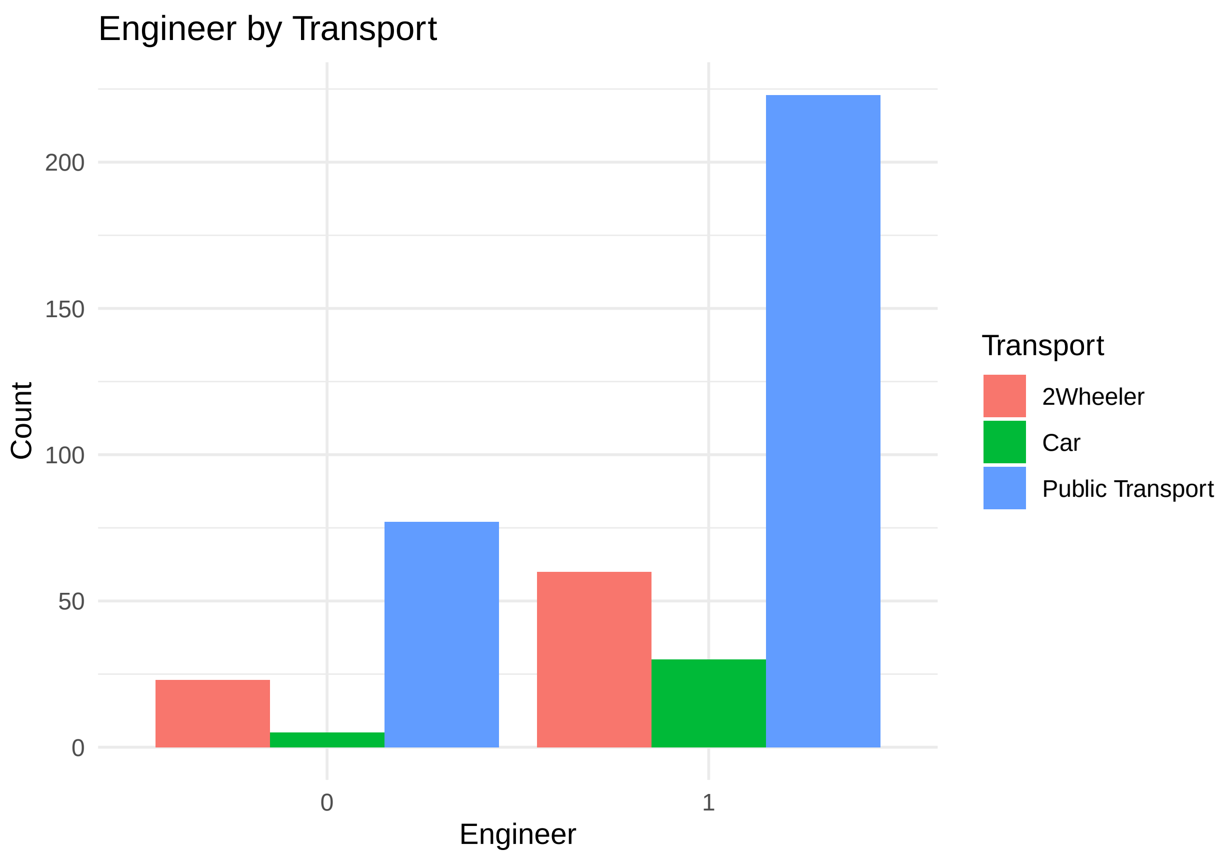


- Data reveals across all modes of transport that male employees usage rate is higher compared to female employees.
- Female employees have a lower usage rate of the transport available. Possible reasons include an alternative mode of transport and female employees live closer to work than their male counterpart.

- Public transport is more common among both gender, trailed by 2Wheeler and car.

2. Relationship between Transport and Engineer

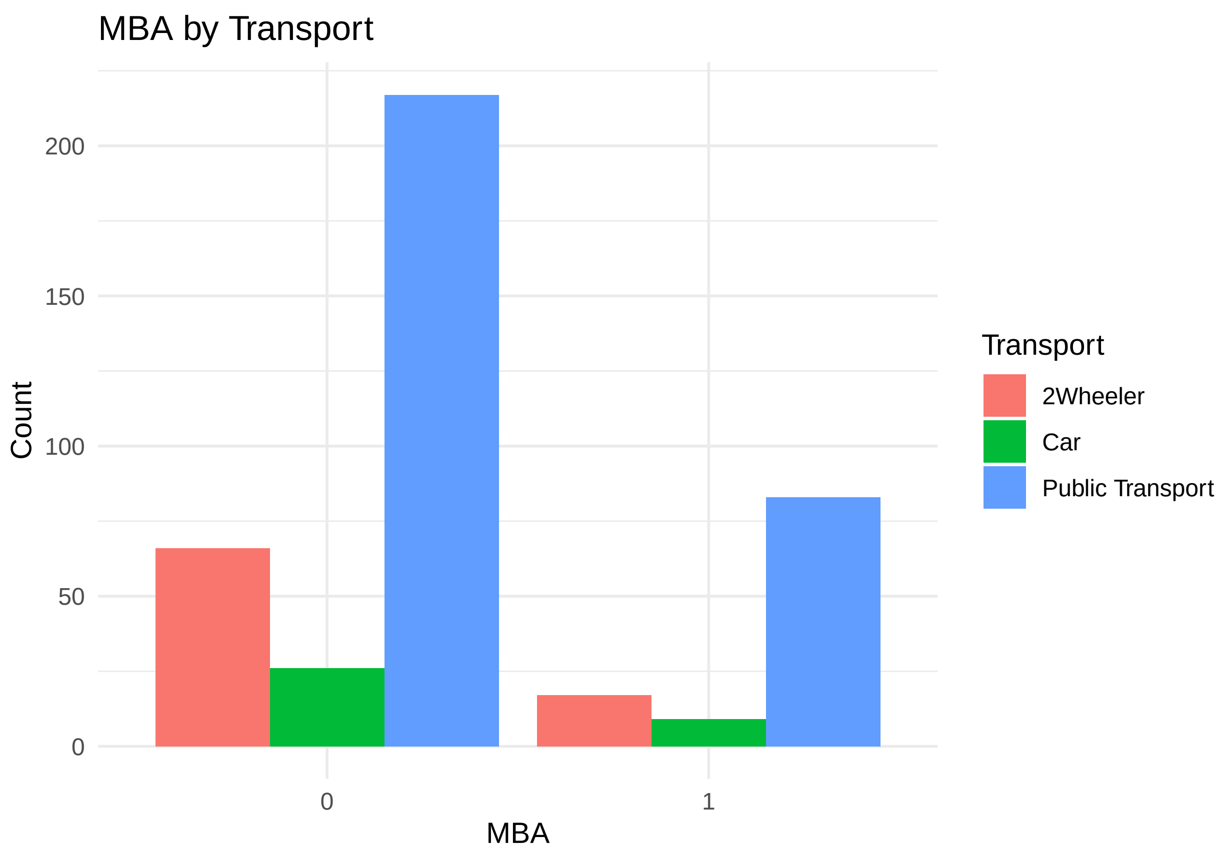
```
ggplot(cars_dataset, aes(x = Engineer, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "Engineer",
       title = "Engineer by Transport") +
  theme_minimal()
```



- Across all modes of transport, employees with an engineering degree have a higher usage rate compared to employees without an engineering degree.
- Similar to gender, public transport is more common among employees with/without an engineering degree. Trailing is 2Wheeler and car.

3. Relationship between Transport and MBA

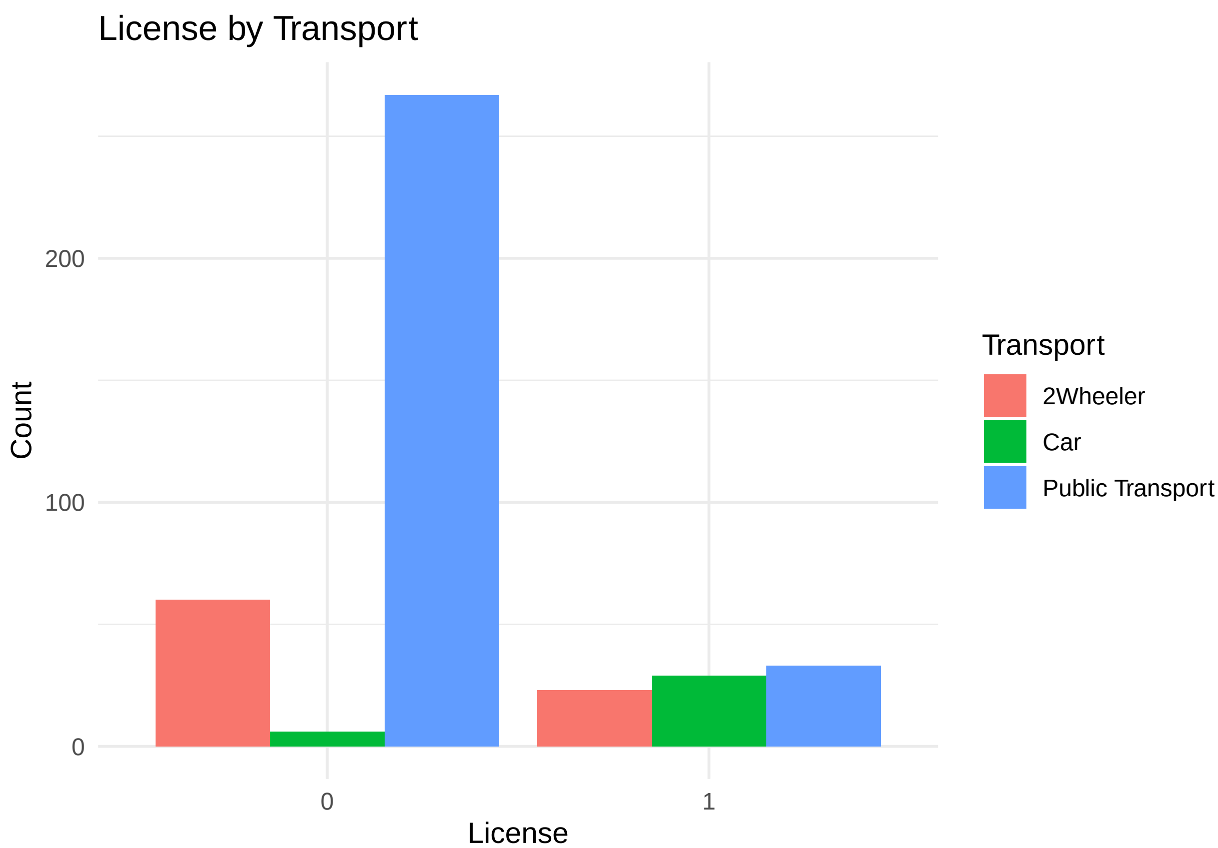
```
ggplot(cars_dataset, aes(x = MBA, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "MBA",
       title = "MBA by Transport") +
  theme_minimal()
```



- Across all modes of transport, employees without an MBA have a higher usage rate compared to employees with an MBA.
- Similar to gender and engineer, public transport is more common with employees with/without an MBA. This is trailed by 2Wheeler and car.

4.Relationship between Transport and License

```
ggplot(cars_dataset, aes(x = license, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "License",
       title = "License by Transport") +
  theme_minimal()
```



- Across all modes of transport except car, data reveals that employees without a license have a higher usage rate compared to employees with a license.
- This should be the case given that a license is required to drive a car. Therefore, it is reasonable for employees without a license to use a 2Wheeler and public transport.
- However, the data suggests some employees have access to a car in the absence of a license. A possible reason could be employees are awaiting license approval or renewal.

2.4.2 Correlation Plot between Numerical Variables Plot bivariate charts between variables to understand their relationship with each other.

Check for correlation among numerical variables

```
# Numeric variables in the data
num_vars = sapply(cars_dataset, is.numeric)

# Correlation Plot
corrplot(cor(cars_dataset[,num_vars]), method = 'number')
```

- There is a high correlation between age and work experience. As an individual grows older, there is a tendency to gain more years of work experience.
- There is a high correlation between age and salary.
- There is a high correlation between work experience and salary. The likelihood that an employee's salary increases with respect to a higher work experience is high.

2.5 The Problem

The case requires us to determine the factors that influence an employees decision to use a car as a mode of transport. In order to achieve this, we have to understand the factors that will cause an employee to use a car or not use a car. The dataset presented has a transport variable with three levels namely "2Wheeler", "Car" and "Public Transport". Since the objective is to predict an employees decision to use a car, we will create a new column (Carusage) segmenting employees mode of transport to "car" or "not car". Under the "Carusage" variable, we will ascribe the word "Car" to employees who use "Car" while the word "Not Car" to employees who use "2Wheeler" and "Public Transport".

```
# Distribution of the Transport variable
prop.table(table(cars_dataset$Transport))*100
```

```
##
##         2Wheeler              Car Public Transport
##        19.856459         8.373206        71.770335
```

```
# Adding a new column titled "Carusage"
# Given we want to determine employees who use a car or not, we will use
# "Car" to represent "Car" and "Not Car" to represent "2Wheeler" and "Public Transport".
cars_dataset$Carusage <- ifelse(cars_dataset$Transport == "Car", "Car", "Not.Car")
table(cars_dataset$Carusage)
```

```
##
##     Car Not.Car
##      35     383
```

```
prop.table(table(cars_dataset$Carusage))*100
```

```
##
##       Car   Not.Car
##  8.373206 91.626794
```

From the above, the proportion of employees using a car as a mode of transport is 8.37% compared to 91.63% of employees not using a car. The data reveals that the number of employees using a car is in the minority. This poses an imbalance problem given that the aim of this report is to accurately predict whether or not an employee will use Car as a mode of transport. In order to solve this, we will use a methodology known as Synthetic Minority Over-sampling Technique (SMOTE).

```
# The Carusage variable needs to be converted to a factor variable
cars_dataset$Carusage <- as.factor(cars_dataset$Carusage)
summary(cars_dataset)
```

```
##       Age           Gender    Engineer MBA        Work.Exp
##  Min.   :18.00   Female:121   0:105    0:309   Min.   : 0.000
##  1st Qu.:25.00   Male  :297   1:313    1:109   1st Qu.: 3.000
##  Median :27.00                                 Median : 5.000
##  Mean   :27.33                                 Mean   : 5.873
##  3rd Qu.:29.00                                 3rd Qu.: 8.000
##  Max.   :43.00                                 Max.   :24.000
##      Salary           Distance     license           Transport      Carusage
##  Min.   : 6.500   Min.   : 3.20   0:333   2Wheeler        : 83   Car    : 35
##  1st Qu.: 9.625   1st Qu.: 8.60   1: 85   Car             : 35   Not.Car:383
##  Median :13.000   Median :10.90           Public Transport:300
##  Mean   :15.418   Mean   :11.29
##  3rd Qu.:14.900   3rd Qu.:13.57
##  Max.   :57.000   Max.   :23.40
```

3.Data Preparation

```
# Remove the Transport variable
cars_dataset <- cars_dataset[,-9]
view(cars_dataset)
```

```
# Split the data into train and test
set.seed(123)
carsdataset_index <- createDataPartition(cars_dataset$Carusage, p = 0.70, list = FALSE)

carsdataset_train <- cars_dataset[carsdataset_index,]
carsdataset_test <- cars_dataset[-carsdataset_index,]

prop.table(table(cars_dataset$Carusage))*100
```

```
##
##       Car   Not.Car
##  8.373206 91.626794
```

```
prop.table(table(carsdataset_train$Carusage))*100
```

```
##
```

```
##       Car   Not.Car
##  8.503401 91.496599
```

```r
prop.table(table(carsdataset_test$Carusage))*100
```

```
##
##       Car   Not.Car
##  8.064516 91.935484
```

- The train and test dataset have almost the same car usage percentage as the base dataset.

```r
# Apply SMOTE on the Train dataset
table(carsdataset_train$Carusage)
```

```
##
##     Car Not.Car
##      25     269
```

```r
prop.table(table(carsdataset_train$Carusage))*100
```

```
##
##       Car   Not.Car
##  8.503401 91.496599
```

```r
smote_carsdataset_train <- SMOTE(Carusage ~ ., data = carsdataset_train,
                      perc.over = 500,
                      perc.under = 200,
                      k = 5)

table(smote_carsdataset_train$Carusage)
```

```
##
##     Car Not.Car
##     150     250
```

```r
prop.table(table(smote_carsdataset_train$Carusage))*100
```

```
##
##     Car Not.Car
##    37.5    62.5
```

```r
# perc.over
# how many extra cases from the minority class are generated (known as over-sampling)

# smoted_minority_class = perc.over/100 * minority_class_cases + minority_class_cases

# perc.under
# how many extra cases from the majority classes are selected for each case generated from the minority

# k: number of nearest neighbours that are used to generate the new examples of the minority class.
```

- After applying SMOTE, we have a 37.5:62.5 split in the dataset between car users and non car users.

4. Model Building

4.1 Setting up the general parameters for training multiple models

```r
# Define the training control
fitControl <- trainControl(
```

```r
              method = 'repeatedcv',           # k-fold cross validation
              number = 3,                      # number of folds or k
              repeats = 1,                     # repeated k-fold cross-validation
              allowParallel = TRUE,
              classProbs = TRUE,
              summaryFunction = twoClassSummary # should class probabilities be returned
    )
```

4.2 Model_1: KNN

```r
knn_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                   preProcess = c("center", "scale"),
                   method = "knn",
                   tuneLength = 3,
                   trControl = fitControl)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```r
knn_model
```

```
## k-Nearest Neighbors
##
## 400 samples
##   8 predictor
##   2 classes: 'Car', 'Not.Car'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (3 fold, repeated 1 times)
## Summary of sample sizes: 267, 266, 267
## Resampling results across tuning parameters:
##
##   k  ROC        Sens       Spec
##   5  0.9890361  0.9400000  0.9800631
##   7  0.9924139  0.9533333  0.9840792
##   9  0.9924560  0.9533333  0.9840792
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```r
knn_prediction_test <- predict(knn_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(knn_prediction_test, carsdataset_test$Carusage)
```

4.2.1 Predict using the trained model & check performance on test set

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Car Not.Car
##    Car       8       1
##    Not.Car   2     113
##
##                   Accuracy : 0.9758
##                     95% CI : (0.9309, 0.995)
```

```
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.008296
##
##                   Kappa : 0.829
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.80000
##             Specificity : 0.99123
##          Pos Pred Value : 0.88889
##          Neg Pred Value : 0.98261
##              Prevalence : 0.08065
##          Detection Rate : 0.06452
##    Detection Prevalence : 0.07258
##       Balanced Accuracy : 0.89561
##
##        'Positive' Class : Car
##
```

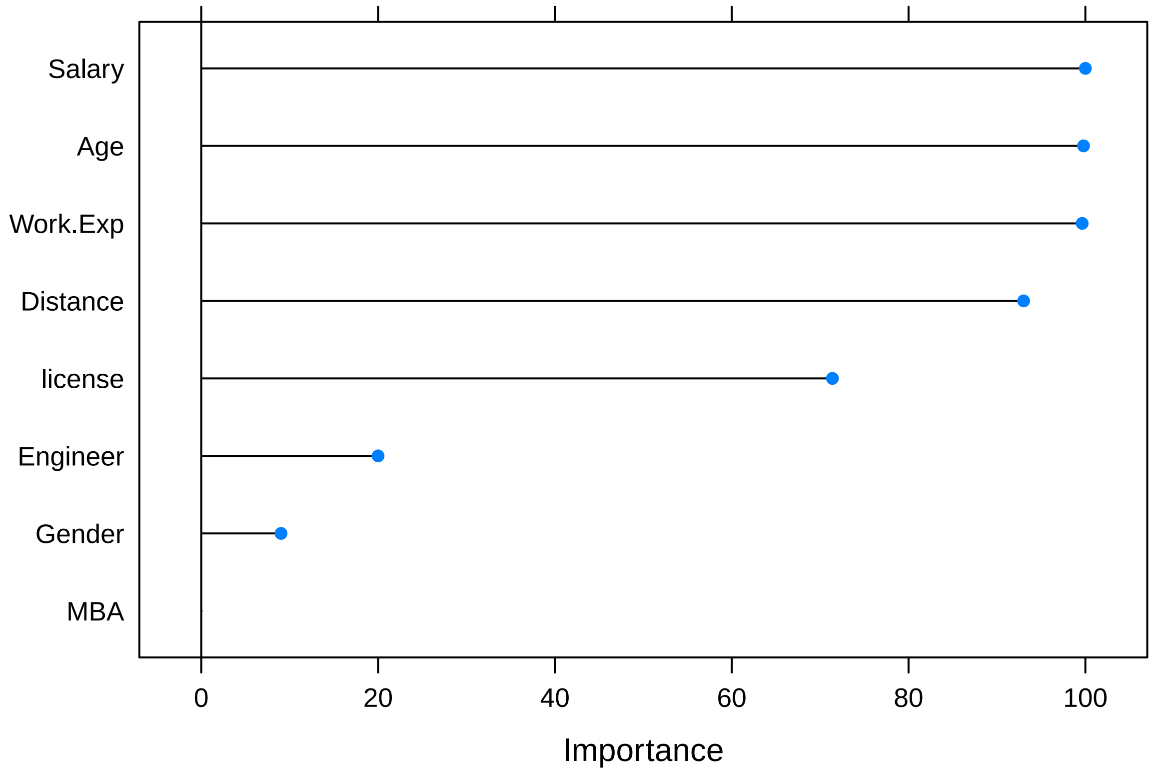
- Accuracy : 97.6%
- Sensitivity : 80.0%
- Specificity : 99.1%
- The accuracy of prediction is 97.5% with almost all non-users of a car predicted accurately. On the other hand, there is an 80.0% accuracy in predicting employees that will use a car.
- From the above metrics we can conclude that KNN is performing very well on the data and is able to differentiate between employees using a car and those not using a car.

```r
varImp(object = knn_model)
```

4.2.2 KNN Variable Importance

```
## ROC curve variable importance
##
##          Importance
## Salary      100.000
## Age          99.792
## Work.Exp     99.639
## Distance     93.014
## license      71.389
## Engineer     20.000
## Gender        9.028
## MBA           0.000
```

```r
plot(varImp(object = knn_model))
```



- The most important variables influencing an employee's decision to use a car or not are salary, age, work experience, distance and license.

4.3 Model_2: Naive Bayes

```
nb_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                  method = "naive_bayes",
                  trControl = fitControl)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
summary(nb_model)
```

```
##
## ================================== Naive Bayes ==================================
##
## - Call: naive_bayes.default(x = x, y = y, laplace = param$laplace, usekernel = TRUE,      adjust = pa
## - Laplace: 0
## - Classes: 2
## - Samples: 400
## - Features: 8
## - Conditional distributions:
##     - KDE: 8
## - Prior probabilities:
##     - Car: 0.375
##     - Not.Car: 0.625
```

```
##
## ----------------------------------------------------------------------------------
```

```r
nb_prediction_test <- predict(nb_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(nb_prediction_test, carsdataset_test$Carusage)
```

4.3.1 Predict using the trained model & check performance on test set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Car Not.Car
##    Car       9       2
##    Not.Car   1     112
##
##                Accuracy : 0.9758
##                  95% CI : (0.9309, 0.995)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.008296
##
##                   Kappa : 0.844
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.90000
##             Specificity : 0.98246
##          Pos Pred Value : 0.81818
##          Neg Pred Value : 0.99115
##              Prevalence : 0.08065
##          Detection Rate : 0.07258
##    Detection Prevalence : 0.08871
##       Balanced Accuracy : 0.94123
##
##        'Positive' Class : Car
##
```

- Accuracy : 97.6%
- Sensitivity : 90.0%
- Specificity : 98.2%
- The accuracy of prediction is 97.5% with almost all non-users of a car predicted accurately. On the other hand, there is a 90.0% accuracy in predicting employees that will use a car.
- Naives Bayes is applicable in this case and surprisingly performs better than KNN on the data. It is capable of differentiating between those using a car and not using a car.

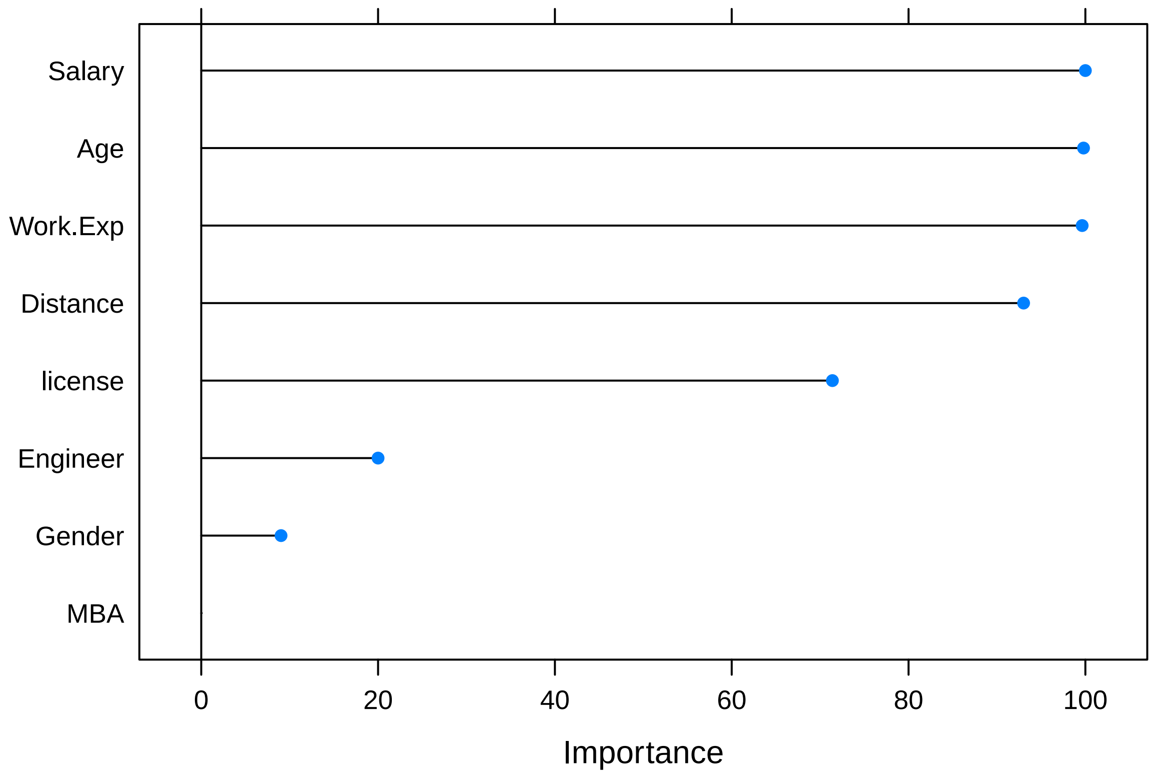
```r
varImp(object = nb_model)
```

4.3.2 Naive-Bayes Variable Importance

```
## ROC curve variable importance
##
##          Importance
## Salary      100.000
```

```
## Age           99.792
## Work.Exp      99.639
## Distance      93.014
## license       71.389
## Engineer      20.000
## Gender         9.028
## MBA            0.000
```

```r
plot(varImp(object = nb_model))
```



- Similar to KNN, the most important variables here are salary, age, work experience, distance and license.

4.4 Model_3: GLM: Simple Logistic Regression Model

```r
slr_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                   method = "glm",
                   family = "binomial",
                   trControl = fitControl)

summary(slr_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -2.07766  -0.00614   0.00072   0.00920   1.89060
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  58.9724    21.8380   2.700  0.00692 **
## Age          -1.4186     0.7707  -1.841  0.06565 .
## GenderMale    0.6881     1.1312   0.608  0.54300
## Engineer1    -1.2447     2.3858  -0.522  0.60186
## MBA1          0.2393     0.9557   0.250  0.80230
## Work.Exp      0.7236     0.7048   1.027  0.30459
## Salary       -0.2222     0.1323  -1.679  0.09317 .
## Distance     -0.9484     0.2575  -3.684  0.00023 ***
## license1     -2.1087     1.0639  -1.982  0.04748 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 529.251  on 399  degrees of freedom
## Residual deviance:  36.866  on 391  degrees of freedom
## AIC: 54.866
##
## Number of Fisher Scoring iterations: 10
```

```
slr_prediction_test <- predict(slr_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(slr_prediction_test, carsdataset_test$Carusage)
```

4.4.1 Predict using the trained model & check performance on test set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Car Not.Car
##    Car       9       1
##    Not.Car   1     113
##
##                Accuracy : 0.9839
##                  95% CI : (0.943, 0.998)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.002091
##
##                   Kappa : 0.8912
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.90000
##             Specificity : 0.99123
##          Pos Pred Value : 0.90000
##          Neg Pred Value : 0.99123
##              Prevalence : 0.08065
##          Detection Rate : 0.07258
##    Detection Prevalence : 0.08065
##       Balanced Accuracy : 0.94561
```

```
##
##        'Positive' Class : Car
##
```

```r
# se"N"sitivity : True "P"ositive rate
# s"P"ecificity : True "N"egative rate
```

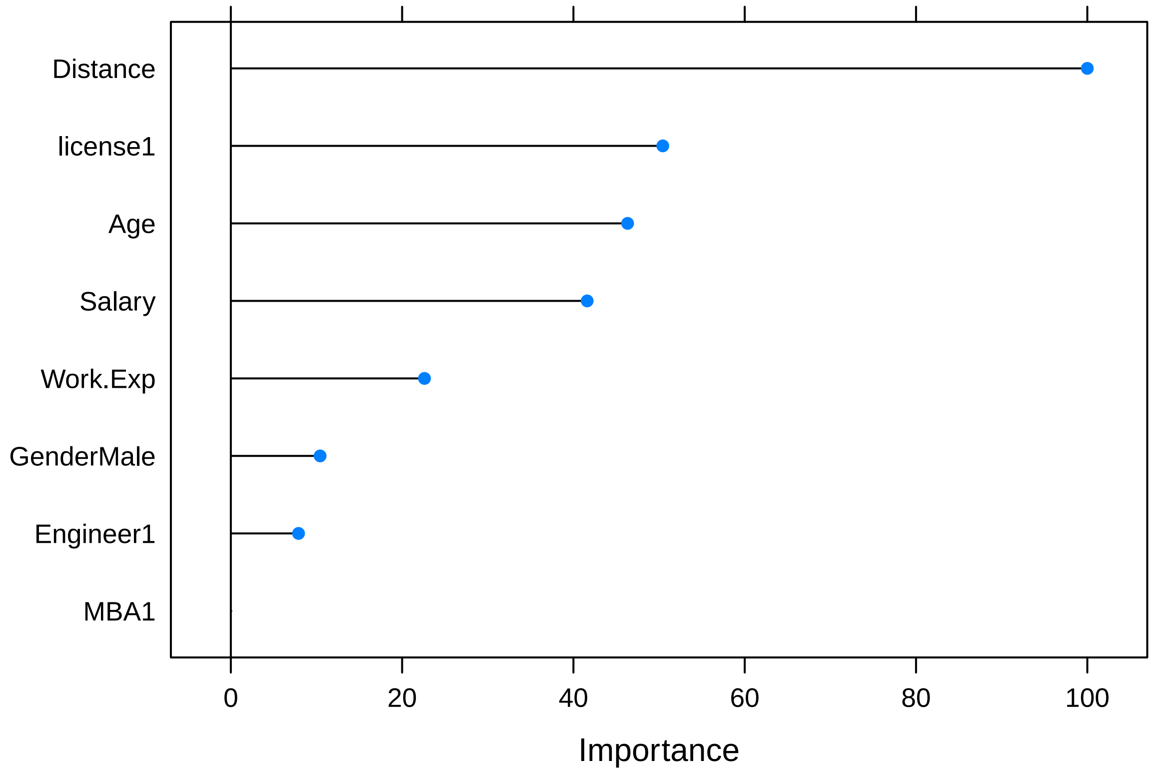
- Accuracy : 98.4%
- Sensitivity : 90.0%
- Specificity : 99.1%
- The accuracy of prediction is 98.4% with almost all non-users of a car predicted accurately. On the other hand, there is a 90.0% accuracy in predicting employees that will use a car.
- Thus far, the logistic regression model performs better than the KNN and Naives Bayes models. It is capable of differentiating between those using a car and not using a car.

```r
varImp(object = slr_model)
```

4.4.2 Logistic Regression Variable Importance

```
## glm variable importance
##
##             Overall
## Distance   100.000
## license1    50.436
## Age         46.323
## Salary      41.607
## Work.Exp    22.610
## GenderMale  10.424
## Engineer1    7.904
## MBA1         0.000
```

```r
plot(varImp(object = slr_model))
```



- Unlike KNN and Naive Bayes, and using a threshold of 70, the most important variable influencing an employee's decision to use a car or not is distance.

4.5 Model_4: Bagging - Random Forest

```
rf_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                     method = "rf",
                     ntree = 30,
                     maxdepth = 5,
                     tuneLength = 10,
                     trControl = fitControl)
```

```
## note: only 7 unique complexity parameters in default grid. Truncating the grid to 7 .
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
rf_prediction_test <- predict(rf_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(rf_prediction_test, carsdataset_test$Carusage)
```

4.5.1 Predict using the trained model & check performance on test set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Car Not.Car
```

```
##    Car       10       1
##    Not.Car    0     113
##
##                Accuracy : 0.9919
##                  95% CI : (0.9559, 0.9998)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.0003521
##
##                   Kappa : 0.948
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 1.00000
##             Specificity : 0.99123
##          Pos Pred Value : 0.90909
##          Neg Pred Value : 1.00000
##              Prevalence : 0.08065
##          Detection Rate : 0.08065
##    Detection Prevalence : 0.08871
##       Balanced Accuracy : 0.99561
##
##        'Positive' Class : Car
##
```

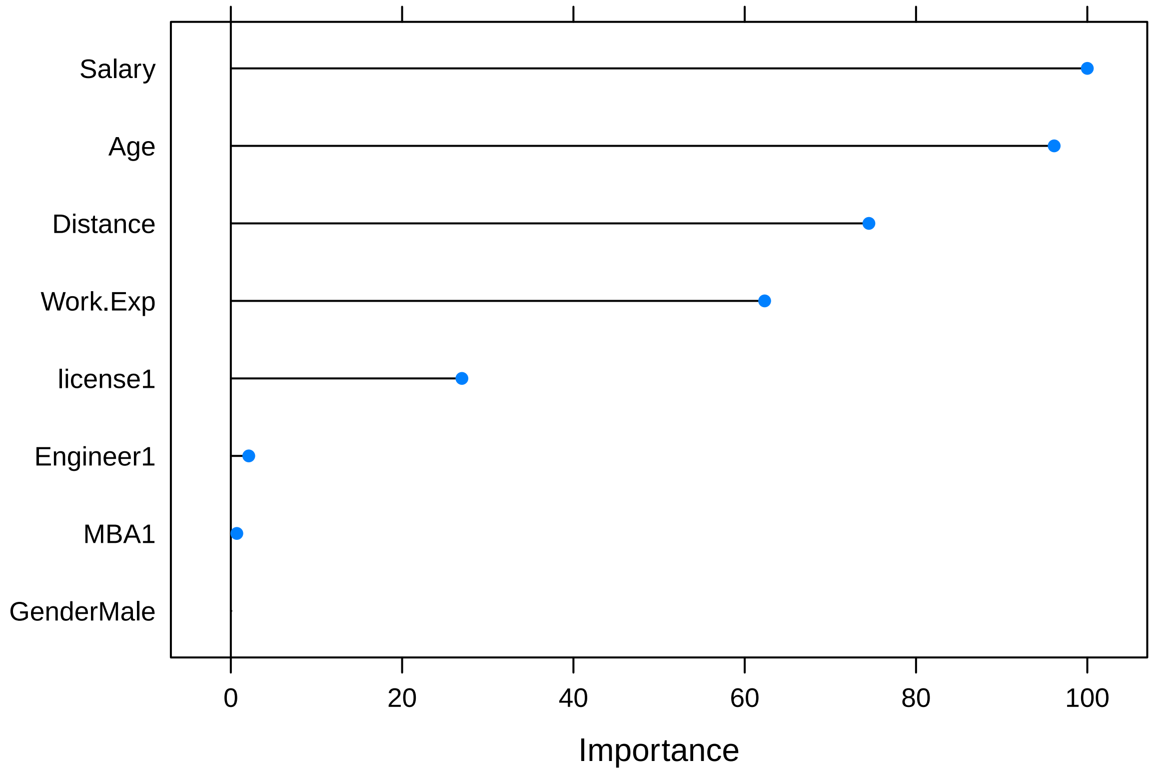
- Accuracy : 100.0%
- Sensitivity : 100.0%
- Specificity : 100.0%
- The accuracy of prediction is 100.0% with all non-users of a car predicted accurately. Similarly, there is a 100.0% accuracy in predicting employees that will use a car.

```r
varImp(object = rf_model)
```

4.5.2 Random Forest Variable Importance

```
## rf variable importance
##
##             Overall
## Salary     100.0000
## Age         96.1315
## Distance    74.4999
## Work.Exp    62.3268
## license1    26.9806
## Engineer1    2.0935
## MBA1         0.7002
## GenderMale   0.0000
```

```r
plot(varImp(object = rf_model))
```



- Here, the most important variables using a threshold of 70 include work experience, age and salary.

4.6 Model_5: Gradient Boosting Machines

```
gbm_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                   method = "gbm",
                   trControl = fitControl,
                   verbose = FALSE)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
gbm_prediction_test <- predict(gbm_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(gbm_prediction_test, carsdataset_test$Carusage)
```

4.6.1 Predict using the trained model & check performance on test set

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction Car Not.Car
##    Car      10       0
##    Not.Car   0     114
##
##                Accuracy : 1
##                  95% CI : (0.9707, 1)
```

```
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.00002964
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 1.00000
##              Prevalence : 0.08065
##          Detection Rate : 0.08065
##    Detection Prevalence : 0.08065
##       Balanced Accuracy : 1.00000
##
##        'Positive' Class : Car
##
```

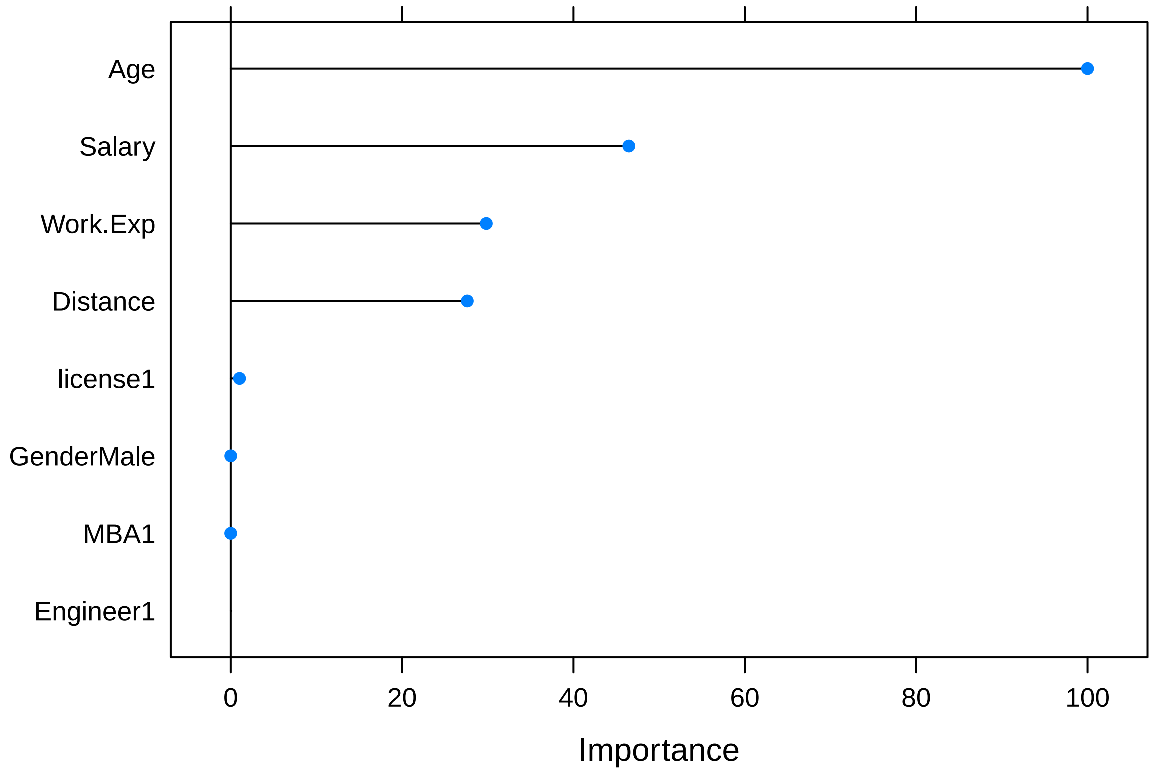
- Accuracy : 99.2%
- Sensitivity : 90.0%
- Specificity : 100.0%
- The accuracy of prediction is 99.2% with almost all non-users of a car predicted accurately. On the other hand, there is a 90.0% accuracy in predicting employees that will use a car.

```
varImp(object = gbm_model)
```

4.6.2 Gradient Boosting Variable Importance

```
## gbm variable importance
##
##               Overall
## Age        100.000000
## Salary      46.467600
## Work.Exp    29.829032
## Distance    27.609793
## license1     1.030269
## GenderMale   0.010295
## MBA1         0.003126
## Engineer1    0.000000
```

```
plot(varImp(object = gbm_model))
```



- With a threshold of 70, the most important variables include age and salary.

4.7 Model_6: Xtreme Gradient boosting Machines

```
cv.ctrl <- trainControl(method = "repeatedcv", repeats = 1,number = 3,
                        summaryFunction = twoClassSummary,
                        classProbs = TRUE,
                        allowParallel=T)

    xgb.grid <- expand.grid(nrounds = 500,
                            eta = c(0.01),
                            max_depth = c(2,4),
                            gamma = 0,               #default=0
                            colsample_bytree = 1,    #default=1
                            min_child_weight = 1,    #default=1
                            subsample = 1            #default=1
    )

    xgb_model <-train(Carusage~.,
                      data=smote_carsdataset_train,
                      method="xgbTree",
                      trControl=cv.ctrl,
                      tuneGrid=xgb.grid,
                      verbose=T,
                      nthread = 2
    )
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
xgb_prediction_test <- predict(xgb_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(xgb_prediction_test, carsdataset_test$Carusage)
```

4.7.1 Predict using the trained model & check performance on test set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Car Not.Car
##    Car       9       0
##    Not.Car   1     114
##
##                Accuracy : 0.9919
##                  95% CI : (0.9559, 0.9998)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.0003521
##
##                   Kappa : 0.943
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.90000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.99130
##              Prevalence : 0.08065
##          Detection Rate : 0.07258
##    Detection Prevalence : 0.07258
##       Balanced Accuracy : 0.95000
##
##        'Positive' Class : Car
##
```

- Accuracy : 99.2%
- Sensitivity : 90.0%
- Specificity : 100.0%
- The accuracy of prediction is 99.2% with almost all non-users of a car predicted accurately. On the other hand, there is a 90.0% accuracy in predicting employees that will use a car.

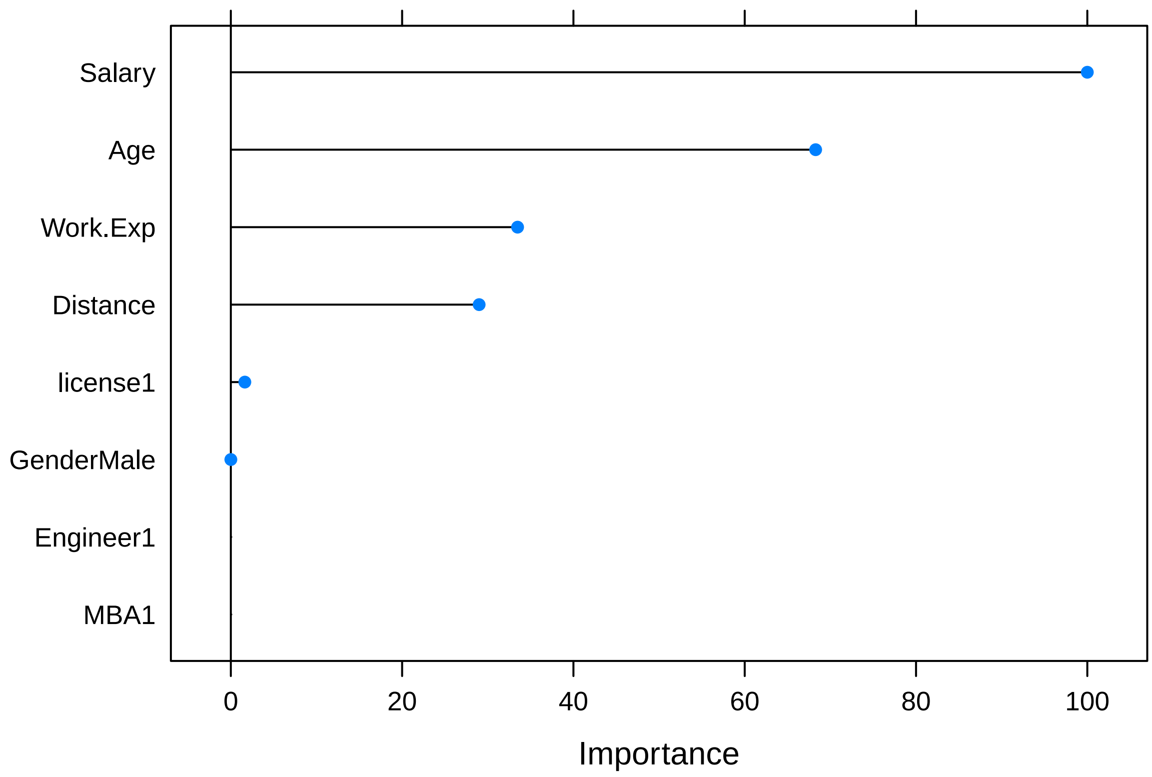
```
varImp(object = xgb_model)
```

4.7.2 Xtreme Gradient Boosting Variable Importance

```
## xgbTree variable importance
##
##             Overall
## Salary     100.00000
## Age         68.28220
## Work.Exp    33.48036
```

```
## Distance    28.99201
## license1     1.63774
## GenderMale   0.00396
## MBA1         0.00000
## Engineer1    0.00000
```

```
plot(varImp(object = xgb_model))
```



- Here, with a threshold of 70, the only important variable is salary.

5. Comparing Models

```
models_to_compare <- list(KNN = knn_model,
                   Naive_Bayes = nb_model,
                   Logistic_Regression = slr_model,
                   Random_Forest = rf_model,
                   Gradient_Boosting = gbm_model,
                   Xtreme_Gradient_Boosting = xgb_model)
resamp <- resamples(models_to_compare)
resamp
```

```
##
## Call:
## resamples.default(x = models_to_compare)
##
## Models: KNN, Naive_Bayes, Logistic_Regression, Random_Forest, Gradient_Boosting, Xtreme_Gradient_Boos
## Number of resamples: 3
```

```
## Performance metrics: ROC, Sens, Spec
## Time estimates for: everything, final model fit
```

```r
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: KNN, Naive_Bayes, Logistic_Regression, Random_Forest, Gradient_Boosting, Xtreme_Gradient_Boos
## Number of resamples: 3
##
## ROC
##                              Min.   1st Qu.    Median      Mean   3rd Qu.
## KNN                     0.9784524 0.9889250 0.9993976 0.9924560 0.9994578
## Naive_Bayes             0.9980952 0.9983247 0.9985542 0.9988832 0.9992771
## Logistic_Regression     0.9925301 0.9926936 0.9928571 0.9948078 0.9959466
## Random_Forest           1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## Gradient_Boosting       0.9971084 0.9985542 1.0000000 0.9990361 1.0000000
## Xtreme_Gradient_Boosting 0.9997590 0.9998795 1.0000000 0.9999197 1.0000000
##                              Max. NA's
## KNN                     0.9995181    0
## Naive_Bayes             1.0000000    0
## Logistic_Regression     0.9990361    0
## Random_Forest           1.0000000    0
## Gradient_Boosting       1.0000000    0
## Xtreme_Gradient_Boosting 1.0000000    0
##
## Sens
##                         Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
## KNN                     0.94    0.94   0.94 0.9533333    0.96 0.98    0
## Naive_Bayes             1.00    1.00   1.00 1.0000000    1.00 1.00    0
## Logistic_Regression     0.88    0.92   0.96 0.9400000    0.97 0.98    0
## Random_Forest           1.00    1.00   1.00 1.0000000    1.00 1.00    0
## Gradient_Boosting       1.00    1.00   1.00 1.0000000    1.00 1.00    0
## Xtreme_Gradient_Boosting 0.98    0.99   1.00 0.9933333    1.00 1.00    0
##
## Spec
##                              Min.   1st Qu.    Median      Mean   3rd Qu. Max.
## KNN                     0.9642857 0.9761188 0.9879518 0.9840792 0.9939759    1
## Naive_Bayes             0.9523810 0.9581182 0.9638554 0.9720788 0.9819277    1
## Logistic_Regression     0.9638554 0.9698795 0.9759036 0.9799197 0.9879518    1
## Random_Forest           1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    1
## Gradient_Boosting       0.9759036 0.9879518 1.0000000 0.9919679 1.0000000    1
## Xtreme_Gradient_Boosting 0.9879518 0.9880235 0.9880952 0.9920157 0.9940476    1
##                         NA's
## KNN                        0
## Naive_Bayes                0
## Logistic_Regression        0
## Random_Forest              0
## Gradient_Boosting          0
## Xtreme_Gradient_Boosting    0
```

```
##                        Name Accuracy Sensitivity Specificity
## 1                       KNN     0.97         0.8        0.99
```

```
## 2              Naive_Bayes     0.97          0.9        0.98
## 3      Logistic_Regression     0.98          0.9        0.99
## 4            Random_Forest     1.00          1.0        1.00
## 5        Gradient_Boosting     0.99          0.9        1.00
## 6 Xtreme_Gradient_Boosting     0.99          0.9        1.00
```

6. Conclusion

- The bagging algorithm known as random forest has the highest accuracy and sensitivity compared to every other model, and performs the best on our data.
- Using the random forest model, all predictors influencing an employee's decision to use a car or not are work experience, age, salary, distance and those with a license. However, the most significant predictors using a threshold of 70 include work experience, age and salary.
- From the comparison, bagging and boosting modelling procedures perform better than other model performance metrics such as KNN, naive bayes and logistic regression.
- Using navie bayes as a benchmark, only the KNN model underperforms.
- Due to low sensitivity on the KNN model, it can be tuned to see if it can outperform the naive bayes model. However, accuracy and specificity may be affected significantly.

7. Appendix A – Source Code

```
#==========================================================================
#
# Exploratory Data Analysis - CardioGoodFitness
#
#==========================================================================

# Environment set up and data import

# Invoking libraries
library(readr) # To import csv files
library(ggplot2) # To create plots
library(corrplot) # To plot correlation plot between numerical variables
library(gridExtra) # To plot multiple ggplot graphs in a grid
library(DataExplorer) # visual exploration of data
library(caTools) # Split Data into Test and Train Set
library(caret) # for confusion matrix function
library(randomForest) # to build a random forest model
library(rpart) # to build a decision model
library(rattle)
library(gbm) # basic implementation using AdaBoost
library(xgboost) # to build a XGboost model
library(DMwR) # for sMOTE
library(knitr) # Necessary to generate source codes from a .Rmd File
library(markdown) # To convert to HTML
library(rmarkdown) # To convret analyses into high quality documents

# Set working directory
setwd("C:/Users/egwuc/Desktop/PGP-DSBA-UT Austin/Machine Learning/Week 5 - Project/")

# Read input file
cars_dataset <- read.csv("Cars-dataset.csv")
```

```
# Global options settings
options(scipen = 999) # turn off scientific notation like 1e+06

# Check dimension of dataset
dim(cars_dataset)

# Check first 6 rows(observations) of dataset
head(cars_dataset)
tail(cars_dataset)

# Check structure of dataset
str(cars_dataset)

# Get summary of dataset
summary(cars_dataset)

# How many missing vaues do we have?
sum(is.na(cars_dataset))

# What columns contain missing values?
colSums(is.na(cars_dataset))

# Impute the missing value with the column mean/median
data1 = cars_dataset
data1$MBA[is.na(data1$MBA)] <- median(data1$MBA, na.rm = T)
dim(data1)
cars_dataset <- data1
sum(is.na(cars_dataset))

# Change Engineer, MBA and license to factor variable
cars_dataset$Engineer <- as.factor(cars_dataset$Engineer)
cars_dataset$MBA <- as.factor(cars_dataset$MBA)
cars_dataset$license <- as.factor(cars_dataset$license)

# View the dataset
View(cars_dataset)

# Distribution of the dependent variable
prop.table(table(cars_dataset$Transport))*100

plot_histogram_n_boxplot = function(variable, variableNameString, binw){

  a <- ggplot(data = cars_dataset, aes(x = variable)) +
    labs(x = variableNameString, y = 'count')+
    geom_histogram(fill = 'green', col = 'white', binwidth = binw) +
    geom_vline(aes(xintercept = mean(variable)),
               color = "black", linetype = "dashed", size = 0.5)

  b <- ggplot(data = cars_dataset, aes('',variable))+
    geom_boxplot(outlier.colour = 'red', col = 'red', outlier.shape = 19)+
    labs(x = '', y = variableNameString) + coord_flip()
  grid.arrange(a,b,ncol = 2)
}
```

```
plot_histogram_n_boxplot(cars_dataset$Age, 'Age', 2)

plot_histogram_n_boxplot(cars_dataset$Work.Exp, 'Work Experience', 2)

plot_histogram_n_boxplot(cars_dataset$Salary, 'Salary', 5)

plot_histogram_n_boxplot(cars_dataset$Distance, 'Distance', 2)

ggplot(cars_dataset, aes(x = Gender, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "Gender",
       title = "Gender by Transport") +
  theme_minimal()

ggplot(cars_dataset, aes(x = Engineer, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "Engineer",
       title = "Engineer by Transport") +
  theme_minimal()

ggplot(cars_dataset, aes(x = MBA, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "MBA",
       title = "MBA by Transport") +
  theme_minimal()

ggplot(cars_dataset, aes(x = license, fill = Transport)) +
  geom_bar(position = "dodge") +
  labs(y = "Count",
       fill = "Transport",
       x = "License",
       title = "License by Transport") +
  theme_minimal()

# Numeric variables in the data
num_vars = sapply(cars_dataset, is.numeric)

# Correlation Plot
corrplot(cor(cars_dataset[,num_vars]), method = 'number')

# Distribution of the Transport variable
prop.table(table(cars_dataset$Transport))*100

# Adding a new column titled "Carusage"
# Given we want to determine employees who use a car or not, we will use
# "Car" to represent "Car" and "Not Car" to represent "2Wheeler" and "Public Transport".
cars_dataset$Carusage <- ifelse(cars_dataset$Transport == "Car", "Car", "Not.Car")
```

```
table(cars_dataset$Carusage)
prop.table(table(cars_dataset$Carusage))*100

# The Carusage variable needs to be converted to a factor variable
cars_dataset$Carusage <- as.factor(cars_dataset$Carusage)
summary(cars_dataset)

# Remove the Transport variable
cars_dataset <- cars_dataset[,-9]
view(cars_dataset)

# Split the data into train and test
set.seed(123)
carsdataset_index <- createDataPartition(cars_dataset$Carusage, p = 0.70, list = FALSE)

carsdataset_train <- cars_dataset[carsdataset_index,]
carsdataset_test <- cars_dataset[-carsdataset_index,]

prop.table(table(cars_dataset$Carusage))*100
prop.table(table(carsdataset_train$Carusage))*100
prop.table(table(carsdataset_test$Carusage))*100

# Apply SMOTE on the Train dataset
table(carsdataset_train$Carusage)
prop.table(table(carsdataset_train$Carusage))*100

smote_carsdataset_train <- SMOTE(Carusage ~ ., data = carsdataset_train,
                     perc.over = 500,
                     perc.under = 200,
                     k = 5)

table(smote_carsdataset_train$Carusage)
prop.table(table(smote_carsdataset_train$Carusage))*100

# perc.over
# how many extra cases from the minority class are generated (known as over-sampling)

# smoted_minority_class = perc.over/100 * minority_class_cases + minority_class_cases

# perc.under
# how many extra cases from the majority classes are selected for each case generated from the minority

# k: number of nearest neighbours that are used to generate the new examples of the minority class.

# Define the training control
fitControl <- trainControl(
              method = 'repeatedcv',           # k-fold cross validation
              number = 3,                      # number of folds or k
              repeats = 1,                     # repeated k-fold cross-validation
              allowParallel = TRUE,
              classProbs = TRUE,
              summaryFunction = twoClassSummary # should class probabilities be returned
    )
```

```
knn_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                   preProcess = c("center", "scale"),
                   method = "knn",
                   tuneLength = 3,
                   trControl = fitControl)
knn_model

knn_prediction_test <- predict(knn_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(knn_prediction_test, carsdataset_test$Carusage)

varImp(object = knn_model)
plot(varImp(object = knn_model))

nb_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                 method = "naive_bayes",
                 trControl = fitControl)

summary(nb_model)

nb_prediction_test <- predict(nb_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(nb_prediction_test, carsdataset_test$Carusage)

varImp(object = nb_model)
plot(varImp(object = nb_model))

slr_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                 method = "glm",
                 family = "binomial",
                 trControl = fitControl)

summary(slr_model)

slr_prediction_test <- predict(slr_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(slr_prediction_test, carsdataset_test$Carusage)

# se"N"sitivity : True "P"ositive rate
# s"P"ecificity : True "N"egative rate

varImp(object = slr_model)
plot(varImp(object = slr_model))

rf_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                     method = "rf",
                     ntree = 30,
                     maxdepth = 5,
                     tuneLength = 10,
                     trControl = fitControl)

rf_prediction_test <- predict(rf_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(rf_prediction_test, carsdataset_test$Carusage)

varImp(object = rf_model)
plot(varImp(object = rf_model))
```

```
gbm_model <- train(Carusage ~ ., data = smote_carsdataset_train,
                     method = "gbm",
                     trControl = fitControl,
                     verbose = FALSE)

gbm_prediction_test <- predict(gbm_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(gbm_prediction_test, carsdataset_test$Carusage)

varImp(object = gbm_model)
plot(varImp(object = gbm_model))

cv.ctrl <- trainControl(method = "repeatedcv", repeats = 1,number = 3,
                        summaryFunction = twoClassSummary,
                        classProbs = TRUE,
                        allowParallel=T)

    xgb.grid <- expand.grid(nrounds = 500,
                            eta = c(0.01),
                            max_depth = c(2,4),
                            gamma = 0,               #default=0
                            colsample_bytree = 1,    #default=1
                            min_child_weight = 1,    #default=1
                            subsample = 1            #default=1
    )

    xgb_model <-train(Carusage~.,
                     data=smote_carsdataset_train,
                     method="xgbTree",
                     trControl=cv.ctrl,
                     tuneGrid=xgb.grid,
                     verbose=T,
                     nthread = 2
    )

xgb_prediction_test <- predict(xgb_model, newdata = carsdataset_test, type = "raw")
confusionMatrix(xgb_prediction_test, carsdataset_test$Carusage)

varImp(object = xgb_model)
plot(varImp(object = xgb_model))

models_to_compare <- list(KNN = knn_model,
                   Naive_Bayes = nb_model,
                   Logistic_Regression = slr_model,
                   Random_Forest = rf_model,
                   Gradient_Boosting = gbm_model,
                   Xtreme_Gradient_Boosting = xgb_model)
resamp <- resamples(models_to_compare)
resamp
summary(resamp)

Name = c("KNN", "Naive_Bayes", "Logistic_Regression", "Random_Forest", "Gradient_Boosting", "Xtreme_Gra
Accuracy = c(0.97, 0.97, 0.98, 1.0, 0.99, 0.99)
Sensitivity=c(0.80, 0.90, 0.90, 1.0, 0.90, 0.90)
Specificity=c(0.99, 0.98, 0.99, 1.0, 1.0, 1.0)
```

```
output = data.frame(Name, Accuracy, Sensitivity, Specificity)
output

#=========================================================================
#
# T H E - E N D
#
#=========================================================================

# Generate the .R file from this .Rmd to hold the source code

purl("Thera Bank Project.Rmd", documentation = 0)
```

Generate .R file from this Rmd. The .R will contain only the R source code.

```r
# Generate the .R file from this .Rmd to hold the source code

purl("Cars Case Study.Rmd", documentation = 0)
```

To create word or pdf report -> click on Knit in the toolbar above, select knit to pdf.