Project 5 - Machine Learning

Sanju Hyacinth C

22/09/2019

# Problem Statement

This project requires you to understand what mode of transport employees prefers to commute to their office. The attached data 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.

# Problem Objective

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.

# 1. Exploratory Data Analysis

## 1.1 Installing the required packages

#install.packages("DataExplorer")  
#install.packages("car")  
#install.packages("dplyr")  
#install.packages("caTools")  
#install.packages("ROCR")  
#install.packages("e1071")  
#install.packages("mlbench")  
#install.packages("lars")  
#install.packages("elasticnet")  
#install.packages("LiblineaR")  
#install.packages("kknn")  
#install.packages("klaR")  
#install.packages("ppcor")  
#install.packages("fastDummies")  
#install.packages('DMwR')  
#install.packages('gbm')  
#install.packages('xgboost')  
#install.packages("ipred")  
#install.packages("rpart")

## 1.2 Data loading

setwd("D:/R Progms")  
  
car.d = read.csv("Cars.csv")  
  
head(car.d, 10)

## 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  
## 7 23 Male 1 1 3 11.7 7.2 0 2Wheeler  
## 8 23 Male 0 0 0 6.5 7.3 0 2Wheeler  
## 9 24 Male 1 0 4 8.5 7.5 0 2Wheeler  
## 10 28 Male 1 0 6 13.7 7.5 1 2Wheeler

View(car.d)  
  
table(car.d$Gender, car.d$license)

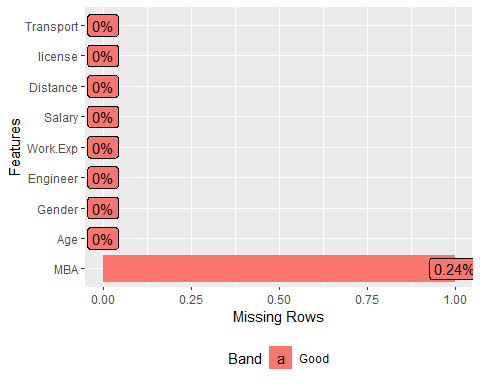
##   
## 0 1  
## Female 114 7  
## Male 219 78

## 1.3 Checking for Missing values

library(DataExplorer)

## Warning: package 'DataExplorer' was built under R version 3.6.1

plot\_missing(car.d)



# We find the presence of NA values in the MBA variable alone from the below plot  
  
car.d = na.omit(car.d)  
# We have omited the one data entry that has an NA value in MBA.  
# Now we have 417 observations in total

## 1.4 Data structure

str(car.d)

## 'data.frame': 417 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 ...  
## - attr(\*, "na.action")= 'omit' Named int 243  
## ..- attr(\*, "names")= chr "243"

# We have 2 existing factor variables (Gender and Transport) and the rest being integer and numeric variables. Later we can convert a few other columns into factor variables.  
  
summary(car.d)

## Age Gender Engineer MBA   
## Min. :18.00 Female:120 Min. :0.0000 Min. :0.0000   
## 1st Qu.:25.00 Male :297 1st Qu.:1.0000 1st Qu.:0.0000   
## Median :27.00 Median :1.0000 Median :0.0000   
## Mean :27.33 Mean :0.7506 Mean :0.2614   
## 3rd Qu.:29.00 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :43.00 Max. :1.0000 Max. :1.0000   
## Work.Exp Salary Distance license   
## Min. : 0.000 Min. : 6.50 Min. : 3.2 Min. :0.0000   
## 1st Qu.: 3.000 1st Qu.: 9.60 1st Qu.: 8.6 1st Qu.:0.0000   
## Median : 5.000 Median :13.00 Median :10.9 Median :0.0000   
## Mean : 5.873 Mean :15.42 Mean :11.3 Mean :0.2038   
## 3rd Qu.: 8.000 3rd Qu.:14.90 3rd Qu.:13.6 3rd Qu.:0.0000   
## Max. :24.000 Max. :57.00 Max. :23.4 Max. :1.0000   
## Transport   
## 2Wheeler : 83   
## Car : 35   
## Public Transport:299   
##   
##   
##

# We have no NA values in the data as we have now removed the one from the MBA variable  
# We can see from the summary that most of the population uses public transport and are men  
# We will have to convert the variables 3, 4 and 8 to factor variables as they can take only two values which will be done later

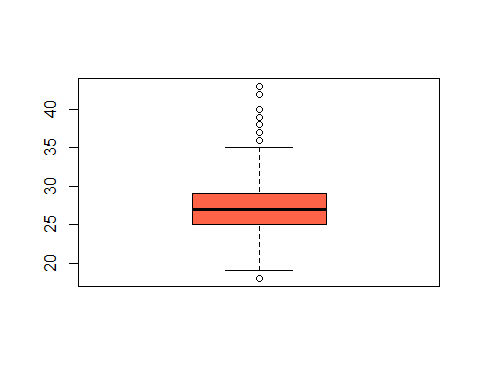
## 1.5 Univariate Analysis - Identifying Outliers

From the below univariate analysis of the variables, we will be able to understand if there are any outliers or extreme values present in the data.

# AGE:  
  
summary(car.d$Age)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 18.00 25.00 27.00 27.33 29.00 43.00

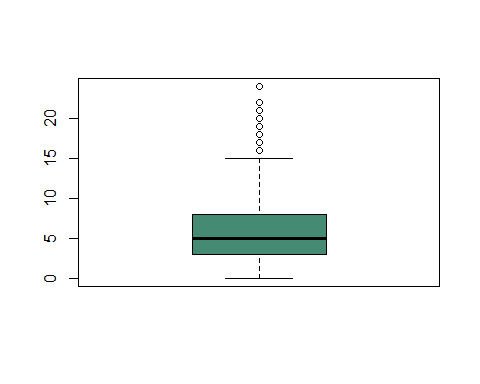
boxplot(car.d$Age, data = car.d, col = "tomato")



# The age variable comprises of outliers where both the min and max range of 18 and 43 (extreme values) lie.  
  
# Work.Exp  
  
summary(car.d$Work.Exp)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 3.000 5.000 5.873 8.000 24.000

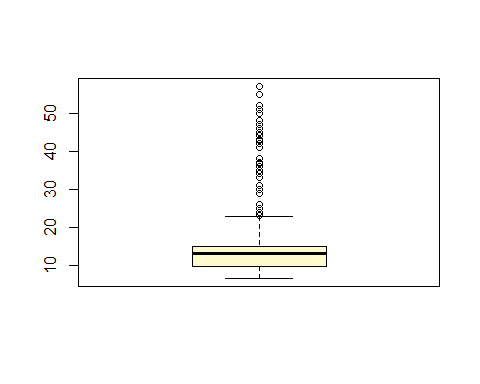
boxplot(car.d$Work.Exp, data = car.d, col = "aquamarine4")



# The variable work experience also comprises of outliers with some values ranging above 15 years  
  
# Salary  
  
summary(car.d$Salary)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 6.50 9.60 13.00 15.42 14.90 57.00

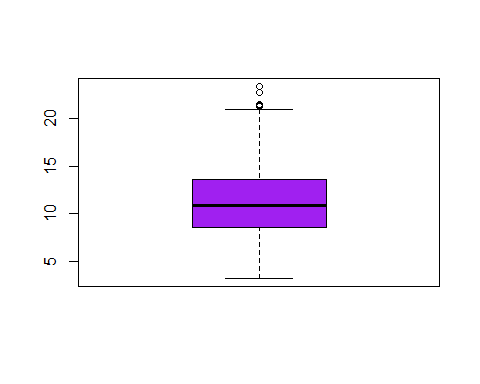
boxplot(car.d$Salary, data = car.d, col = "lemonchiffon")



# This variable salary comprises of a large amount of outliers when compared to the other two fields.  
  
# Distance  
  
summary(car.d$Distance)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.2 8.6 10.9 11.3 13.6 23.4

boxplot(car.d$Distance, data = car.d, col = "purple")



# The distance variable does contain outliers but less when compared to the other variables.  
  
# From the boxplots, we find that all the numeric variables contain outliers (extreme values).

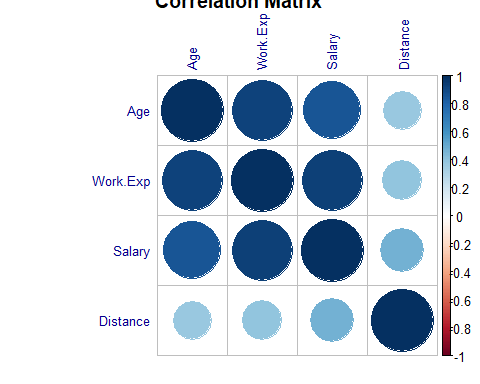
## 1.6 Correlation matrix

library(corrplot)

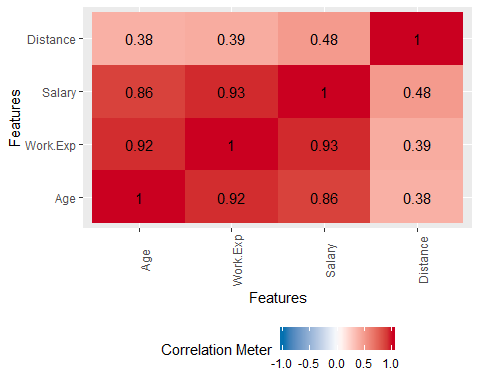
## Warning: package 'corrplot' was built under R version 3.6.1

## corrplot 0.84 loaded

carcorr = corrplot(cor(car.d[, c(1, 5:7)]), method = "circle", title = "Correlation Matrix", tl.cex = 0.8, tl.col = "dark blue")



library(DataExplorer)  
  
plot\_correlation(car.d[, c(1, 5:7)], type = "c")



#Pairwise correlation  
  
library(ppcor)

## Warning: package 'ppcor' was built under R version 3.6.1

## Loading required package: MASS

pcor(car.d[, c(1, 5:7)], method = "pearson")

## $estimate  
## Age Work.Exp Salary Distance  
## Age 1.00000000 0.6687590 -0.03788803 0.04134212  
## Work.Exp 0.66875901 1.0000000 0.70953116 -0.14681864  
## Salary -0.03788803 0.7095312 1.00000000 0.33278916  
## Distance 0.04134212 -0.1468186 0.33278916 1.00000000  
##   
## $p.value  
## Age Work.Exp Salary Distance  
## Age 0.000000e+00 3.991923e-55 4.414267e-01 4.008961e-01  
## Work.Exp 3.991923e-55 0.000000e+00 9.138348e-65 2.716108e-03  
## Salary 4.414267e-01 9.138348e-65 0.000000e+00 3.444049e-12  
## Distance 4.008961e-01 2.716108e-03 3.444049e-12 0.000000e+00  
##   
## $statistic  
## Age Work.Exp Salary Distance  
## Age 0.000000 18.279985 -0.770529 0.840890  
## Work.Exp 18.279985 0.000000 20.462477 -3.016395  
## Salary -0.770529 20.462477 0.000000 7.171862  
## Distance 0.840890 -3.016395 7.171862 0.000000  
##   
## $n  
## [1] 417  
##   
## $gp  
## [1] 2  
##   
## $method  
## [1] "pearson"

### 1.6.1 Inference from Correlation Plot

From the correlation plot, we infer the below:

1. All the numeric variables (Age, Work.Exp, Salary and Distance) are highly correlated to one another.
2. Work experience and Salary has high positive correlation of **92%** and **86%** with Age, as we can understand that the older the person is, the higher the work experience. And the correlation between Salary and Age is also explanatory.
3. Next, we find that there is a high positive correlation of **93%** between Work Experience and Salary too, which is understandable. Higher the work experience, higher is the Salary, hence we can explain the high positive correlation between them.
4. On the other hand, we find that all the 3 variables (Age, Work.Exp and Salary) are having some sort of a correlation with Distance. This can be neglected as it is not too much.

From the above inference we can assume that there is a presence of multicollinearity between the variables as some of the independent variables seem to show a good amount of correlation.

### 1.6.2 Remedy for multicollinearity:

This data comprises of all the numerical columns being correlated with each other. The above inference gives the picture of multicollinearity. As discussed in our previous project, we have options to deal with this. We can either use the **Variance Inflation Factor** method, where we will have to do a regression model on the dependent variable with all the independent variables. If all the variables are found to be having a VIF value less than 5, we can go about the problem without the need to remove any variables. But if we find variables having a higher VIF than 5, we will need to remove the variables one by one from the highest VIF. Upon performing regression with the other variables, we will eventually repeat the same steps to arrive at a consistent model with independent variables having VIF less than 5.

The other method will be of performing **Principal Components Analysis**, where we will group the variables with a commom group or the variables that provide more or less the **same kind of information**. By this way, we may arrive at the variables or the groups that will not be correlated with the other independent variables of the data.

For this data, we may deploy either of the methods PCA method where we can combine the variables that provide similar details, like the work experience and age which contribute to the salary. This may be combined to a group with **superiority** as a header. Or VIF to remove the most correlated varibles.

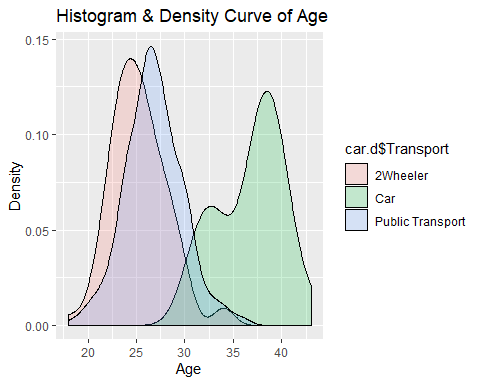
This is a remedy to remove multicollinearity in order to construct stable models

## 1.7 Bivariate Analysis - Plots and Inferences:

# Converting the required fields to factor format:  
# The below fields can only hold two values 0 and 1  
  
car.d$Engineer = factor(car.d$Engineer)  
car.d$MBA = factor(car.d$MBA)  
car.d$license = factor(car.d$license)  
  
str(car.d)

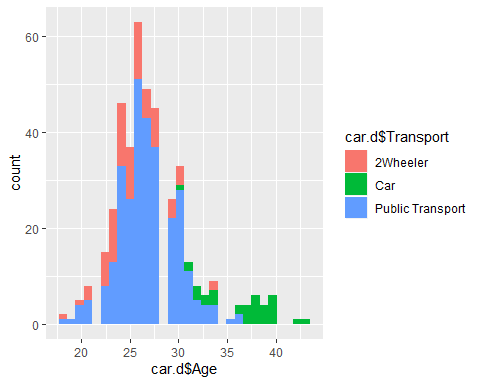
## 'data.frame': 417 obs. of 9 variables:  
## $ Age : int 28 24 27 25 25 21 23 23 24 28 ...  
## $ Gender : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 2 2 2 2 ...  
## $ Engineer : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 2 1 2 2 ...  
## $ MBA : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...  
## $ 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 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...  
## $ Transport: Factor w/ 3 levels "2Wheeler","Car",..: 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(\*, "na.action")= 'omit' Named int 243  
## ..- attr(\*, "names")= chr "243"

library(ggplot2)  
  
den.plot <- ggplot(data=car.d, aes(x=car.d$Age, fill=car.d$Transport))  
den.plot2 = den.plot + geom\_density(stat="density", alpha=I(0.2)) + xlab("Age") + ylab("Density") + ggtitle("Histogram & Density Curve of Age")  
  
den.plot2



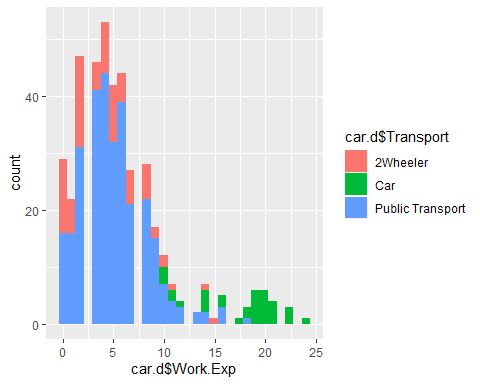
# The above density plot shows that most of the youngsters between the age group of 20-30 mostly use 2 wheelers and public transport. Whilst the older people between age 30-43 choose to travel by car.  
  
# We also get a clear picture from the plot that no one below the age of 25 uses a car for transportation.  
  
p1 = ggplot(data = car.d) + geom\_histogram(mapping = aes(x = car.d$Age, fill = car.d$Transport))  
p1

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

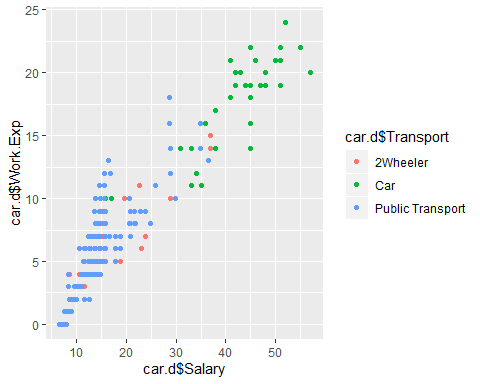


# This plot is another illustration of the measure of people who use all modes of transport. We can say that almost all the employees between the age group of 22-28 use only either 2 wheelers or public transport, where many of them use the latter. People between the age of 30-43 mix the choices and use cars too, but beyond the age of 36, we can be sure from the plot that the employees use only cars and not any other transport.  
  
p2 = ggplot(data = car.d) + geom\_histogram(mapping = aes(x = car.d$Work.Exp, fill = car.d$Transport))  
p2

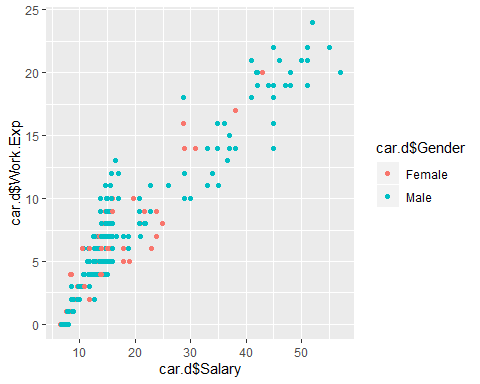
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# From this plot, we see that employees below work experience of 9 years use either 2 wheelers or public transport only, whereas we can also see that employees with experience of 18+ years use only cars as their mode of commute. Inshort, employees in their beginning years use the 2 wheelers and public mode, and those in their senior level of work prefer to use only cars to commute. And there are people in between these two ranges who use all three.  
   
# DEEPER ANALYSIS  
p3 = ggplot(data = car.d,aes(y= car.d$Work.Exp, x= car.d$Salary))+  
 geom\_point(aes(color = car.d$Transport))   
p3



# First of all, we can see the positive correlation between work experience and salary  
  
# Secondly, this plot shows accurately how many employees choose to commute by car. Let's break it down. We have 2-3 employees commuting by car having 10 years of work experience and ideal salary. The employees whose salary range is between 31 and 34 also commute only by car. Then comes the employees who are the seniors with work experience over 17 and salary over 38 who also commute only by car.   
  
p4 = ggplot(data = car.d,aes(y= car.d$Work.Exp, x= car.d$Salary))+  
 geom\_point(aes(color = car.d$Gender))   
p4



# We know that the population on this data has most amount of men who are more than double the population of women. Most of them are under 15 years of work experience. To be gender specific we can fairly assume that most of the employees in the top order are men with experience over 15 years and salary package over 35.  
  
table(car.d$Transport, car.d$Gender)

##   
## Female Male  
## 2Wheeler 38 45  
## Car 6 29  
## Public Transport 76 223

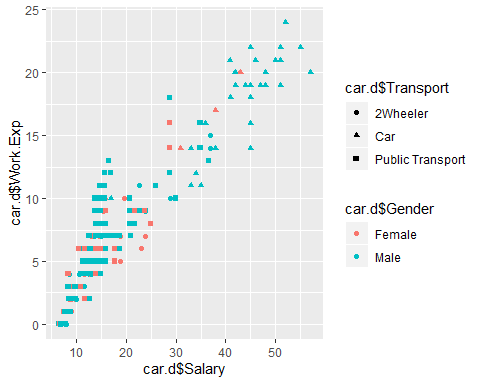
# Most of the population travels by public transport is something that we are already aware of. But from this table, we confirm that about 63% of the entire female population and almost 75% of the entire male population make use of the public commute system. The male employees travelling by car is almost 5 times that of the female employees.  
  
# Based on the population percentage, we can see that only 25% of the male population commute on their own whereas it is about 37% for the female population   
  
table(car.d$license, car.d$Transport)

##   
## 2Wheeler Car Public Transport  
## 0 60 6 266  
## 1 23 29 33

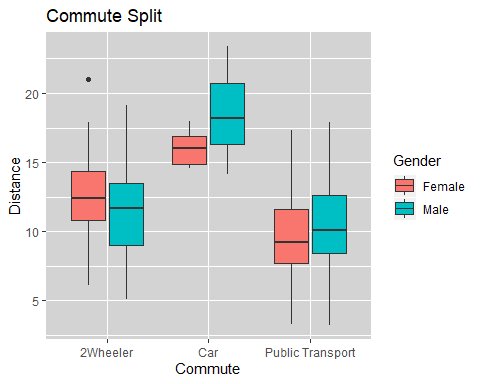
# This table shows the number of people who own a licence against the commute modes  
  
table(car.d$Gender, car.d$license)

##   
## 0 1  
## Female 113 7  
## Male 219 78

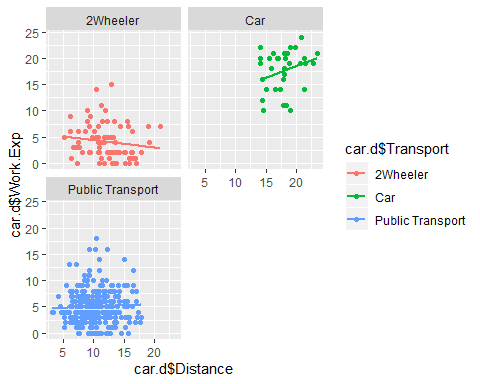
# From the above observation and from this table, we can see that only about 5% of the female employees and 26% of the male employees own a license. This license could be assumed as car licence, as no one would be eligible to ride two wheelers without proper licence. This explains the numbers for public transport as well.  
  
# A better visualisation of the above statements is given in the below plot  
p5 = ggplot(data = car.d,aes(y= car.d$Work.Exp, x= car.d$Salary))+  
 geom\_point(aes(color = car.d$Gender, shape = car.d$Transport))   
p5



p6 = ggplot(data = car.d, aes(x=car.d$Transport, y=car.d$Distance)) + geom\_boxplot(aes(fill = car.d$Gender))  
p61 = p6 + labs(title = "Commute Split",x="Commute", y="Distance",fill = "Gender")  
p62 = p61 + theme(panel.background = element\_rect(fill = "light grey"))  
p62

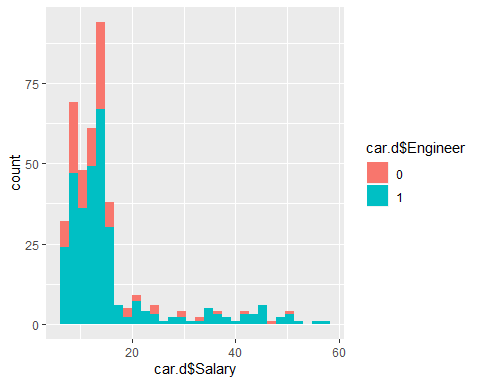


# In this plot we find that in the order of the choice of commute, we see the employees choosing public transport for shorter distances with a median of 9 and 10 units for female and male employees respectively. The next choice of commute for a longer distance is 2 wheelers with a median distance of 12.5 and 12 units for female and male employees respectively. And finally we have the employees commuting by car for longer distances of median of 16 and 18 units for female and male employees respectively. Here we have one female employee commuting by 2 wheeler for a 21 unit distance  
  
p7 = ggplot(data = car.d,aes(y= car.d$Work.Exp, x= car.d$Distance, col = car.d$Transport))+ geom\_point() + geom\_smooth(method = "lm",se = F)+ facet\_grid(~car.d$Transport) + facet\_wrap(~car.d$Transport, ncol = 2)  
p7



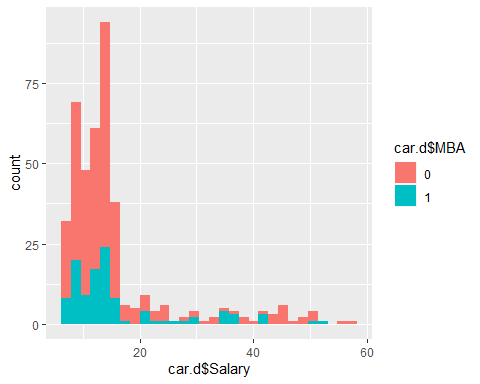
# From this plot, we have 3 graphs showing the relation between work experience and distance for all the modes of commute. For 2 wheelers, we see that the relation shows a negative trend (distance seems to decrease with an increase in work experience) For Cars, there is a positive trend line that with increase in experience, the distance seems to increase as well. There is also a positive trend for public transport but is not that visible. This work experience is also related to age hence the result would be the same.  
  
p8 = ggplot(data = car.d) + geom\_histogram(mapping = aes(x = car.d$Salary, fill = car.d$Engineer))  
p8

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# We can see the spread of the employees with an Engineering degree. They have branched out to the senior level, but mot of them are within the salary line of 20. The same also applies for employees without an engineering degree. The highest paid employee holds an engineering degree is quite evident  
  
p9 = ggplot(data = car.d) + geom\_histogram(mapping = aes(x = car.d$Salary, fill = car.d$MBA))  
p9

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# This plot is almost like the opposite of the previous plot. From this we can find most of them not having an MBA degree. The highest paid employee does not have an MBA degree as we have previously found the employee having an engineering degree.

# 2. Data Preparation:

We make use of the library **fastDummies** and the function **dummy\_cols**to generate some dummy variables for the response field in order to faciltate constructing a good and effective regression model. As we are asked to predict if the employees would take the Car as a commute or not, we are trying to make the variable easy for a binomial prediction model.

# Creating dummy variables for the response category column as we have more than 2 levels in it.  
  
library(fastDummies)

## Warning: package 'fastDummies' was built under R version 3.6.1

# creating dummy variables for the target column  
  
car.dn = dummy\_cols(car.d, select\_columns = "Transport")  
  
# As we need only the column with the probabilities of employees travelling by car, we may remove the other two dummy variables  
  
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.6.1

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':  
##   
## select

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

car.dn = select(car.dn, c(-9,-10,-12))  
  
View(car.dn)

## 2.1 Checking the balance in data:

# Imbalanced data  
  
library(caTools)

## Warning: package 'caTools' was built under R version 3.6.1

set.seed(248)  
sample = sample.split(car.dn,SplitRatio = 0.7)  
train = subset(car.dn,sample == TRUE)  
test = subset(car.dn, sample == FALSE)  
  
nrow(train)

## [1] 279

nrow(test)

## [1] 138

View(train)  
View(test)  
  
table(train$Transport\_Car)

##   
## 0 1   
## 256 23

table(test$Transport\_Car)

##   
## 0 1   
## 126 12

table(car.dn$Transport\_Car)

##   
## 0 1   
## 382 35

# testing data distribution  
  
prop.table(table(train$Transport\_Car))

##   
## 0 1   
## 0.91756272 0.08243728

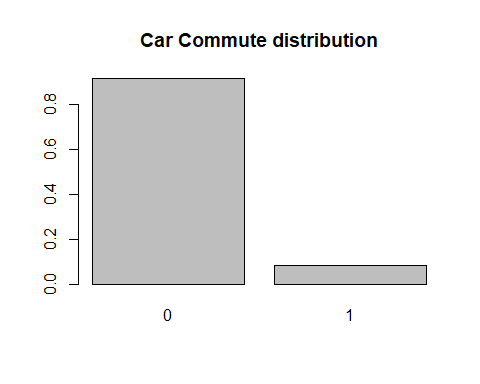
prop.table(table(test$Transport\_Car))

##   
## 0 1   
## 0.91304348 0.08695652

prop.table(table(car.dn$Transport\_Car))

##   
## 0 1   
## 0.91606715 0.08393285

# We find that the data has very less of people commuting by car (only 8% of th total population)  
  
# Plotting to visualise  
  
barplot(prop.table(table(car.dn$Transport\_Car)),  
 main = "Car Commute distribution")



# Even the barplot shows less than 10% of the people using car

## 2.2 Model on imbalanced data:

We have found that the data here is imbalanced with the target level not being sufficient for constructing an unbiased model. Let us now run a logistic regression model on the present unaltered data to find some results.

# Logistic model on the imbalanced data:  
# Logistic Model 1  
  
logit1 = glm(Transport\_Car~., data = train, family = "binomial")  
summary(logit1)

##   
## Call:  
## glm(formula = Transport\_Car ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.95812 -0.01512 -0.00289 -0.00044 2.09207   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -44.85298 28.20496 -1.590 0.1118   
## Age 0.76664 1.03459 0.741 0.4587   
## GenderMale 0.86537 2.44963 0.353 0.7239   
## Engineer1 -0.45085 2.61537 -0.172 0.8631   
## MBA1 -2.91740 2.47623 -1.178 0.2387   
## Work.Exp 0.19067 0.94223 0.202 0.8396   
## Salary 0.08695 0.21412 0.406 0.6847   
## Distance 1.02478 0.42109 2.434 0.0149 \*  
## license1 0.85326 2.23149 0.382 0.7022   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 158.853 on 278 degrees of freedom  
## Residual deviance: 12.808 on 270 degrees of freedom  
## AIC: 30.808  
##   
## Number of Fisher Scoring iterations: 11

# Only the distance seems to be a significant variable  
  
# Prediction on test data:  
  
test.pred = predict(logit1, newdata = test, type = "response")  
table(test$Transport\_Car, test.pred>0.5)

##   
## FALSE TRUE  
## 0 125 1  
## 1 1 11

# Our logit1 model has almost predicted correctly except one identified incorrectly for both instances  
  
# Accuracy   
  
(125+11)/nrow(test)

## [1] 0.9855072

# This model is a 98.5% accurate keeping in mind how the model has identified almost correctly all the time.  
  
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.6.1

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.6.1

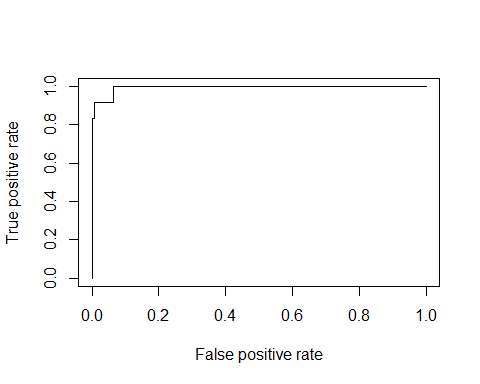
##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

rocr.pred = prediction(test.pred, test$Transport\_Car)  
  
  
# Area Under the Curve is calculated  
as.numeric(performance(rocr.pred, "auc")@y.values)

## [1] 0.9940476

# 99 percent of the time, the model has given correct predictions  
  
perf = performance(rocr.pred, "tpr", "fpr")  
plot(perf)



# For this data, we can justify that the model has made very good predictions on this data, missing out just one or two predictions and giving a 98% accuracy

We find that even on an imbalanced data, the accuracy we have got is over 98%. We may still have to proceed with balancing the data to check if the accuracies change or do not change any further after applying the SMOTE technique.

## 2.3 SMOTE (Synthetic Minority Oversampling Technique)

SMOTE is a function that is used generate synthetic data points for an imbalanced data, in order to tweak up the minority class in the data. For an imbalanced data where the majority class is more than 90%, a model constructed on that would be having results dominated by the majority class. This will have an impact on the model, where the machine learning algorithm could simply classify everything as the majority class and still be correct 90% of the time. In order to effectively control this situation, we employ the function SMOTE.

Let us now work on pur data and see how well SMOTE can control this situation. We will run the same models as we did on the data originally.

# Working with SMOTE  
  
library(DMwR)

## Warning: package 'DMwR' was built under R version 3.6.1

## Loading required package: lattice

## Loading required package: grid

## Registered S3 method overwritten by 'xts':  
## method from  
## as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

sm.train<-subset(car.dn, sample == TRUE)  
sm.test<-subset(car.dn, sample == FALSE)  
  
# The amount of 0 nd 1 on the smote train data  
table(sm.train$Transport\_Car)

##   
## 0 1   
## 256 23

sm.train$Transport\_Car<-as.factor(sm.train$Transport\_Car)  
# converting to factor  
  
bald.car.tr <- SMOTE(Transport\_Car ~., sm.train, perc.over = 100, k = 5, perc.under = 500)  
  
#perc.over means that 1 minority class will be added for every value of perc.over  
  
# We are adding 100 percent of our existing minority class to the value   
  
# perc under - for every minority class generated we have retained 500 percent of the minority class. That will be 23\*5  
  
table(bald.car.tr$Transport\_Car)

##   
## 0 1   
## 115 46

# This table has increased the minority class by 100 perc over and has also decreased the majority class  
  
# Checking the proportion of the smote train data:  
  
prop.table(table(bald.car.tr$Transport\_Car))

##   
## 0 1   
## 0.7142857 0.2857143

As we see from the above split of data, we have 71% on the majority class (which indicates employees who do not commute by car) and the minority class raise to roughly about 29% .Now that our data is ready for the models to be performed, let us now go to the modelling section

# 3. Modeling:

## 3.1 Logistic Regression:

Let us first look at our logistic regression model, continuing from the previous task. We will use the smoted train and test data here in order to produce an effective and unbiased model.

# Logistic model on the smoted data:  
# Logistic Model 2  
  
logit2 = glm(Transport\_Car~., data = sm.train, family = "binomial")  
summary(logit2)

##   
## Call:  
## glm(formula = Transport\_Car ~ ., family = "binomial", data = sm.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.95812 -0.01512 -0.00289 -0.00044 2.09207   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -44.85298 28.20496 -1.590 0.1118   
## Age 0.76664 1.03459 0.741 0.4587   
## GenderMale 0.86537 2.44963 0.353 0.7239   
## Engineer1 -0.45085 2.61537 -0.172 0.8631   
## MBA1 -2.91740 2.47623 -1.178 0.2387   
## Work.Exp 0.19067 0.94223 0.202 0.8396   
## Salary 0.08695 0.21412 0.406 0.6847   
## Distance 1.02478 0.42109 2.434 0.0149 \*  
## license1 0.85326 2.23149 0.382 0.7022   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 158.853 on 278 degrees of freedom  
## Residual deviance: 12.808 on 270 degrees of freedom  
## AIC: 30.808  
##   
## Number of Fisher Scoring iterations: 11

# Here too the distance seems to be a significant variable  
  
# Prediction on test data:  
  
test.pred2 = predict(logit2, newdata = sm.test, type = "response")  
table(test$Transport\_Car, test.pred2>0.5)

##   
## FALSE TRUE  
## 0 125 1  
## 1 1 11

# Our logit2 model has almost predicted correctly except one identified incorrectly for both instances  
  
# Accuracy  
(125+11)/nrow(sm.test)

## [1] 0.9855072

# This model is a 98.5% accurate keeping in mind how the model has identified almost correctly all the time.  
  
library(ROCR)  
rocr.pred2 = prediction(test.pred2, sm.test$Transport\_Car)  
  
# Area Under the Curve is calculated  
as.numeric(performance(rocr.pred2, "auc")@y.values)

## [1] 0.9940476

# 99 percent of the time, the model has given correct predictions  
  
# ROC Calculation  
perf2 = performance(rocr.pred2, "tpr", "fpr")  
plot(perf2)  
  
# For this data, we can justify that the model has made very good predictions on this data, missing out just one or two predictions and giving a 98% accuracy  
# This is exactly the same as our model for the imbalanced data.  
  
# Let us tweak up the probability threshold to 0.8 for prediction  
# Prediction on test data:  
  
test.pred3 = predict(logit2, newdata = sm.test, type = "response")  
table(test$Transport\_Car, test.pred3>0.8)

##   
## FALSE TRUE  
## 0 125 1  
## 1 2 10

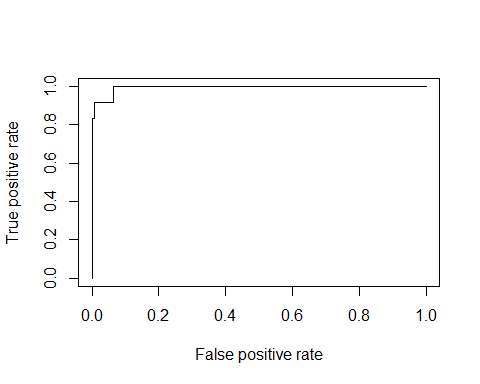
# Our logit2 model with the prob to 0.8 has falsely identified 2 predictions which were true but still has got 4/5th of it correctly  
  
# Accuracy  
(125+10)/nrow(sm.test)

## [1] 0.9782609

# This model is a 97.8% accurate and little less when compared to our previous model with prob threshold of 0.5  
  
library(ROCR)  
rocr.pred3 = prediction(test.pred3, sm.test$Transport\_Car)  
  
# Let us look at the Area Under the Curve  
as.numeric(performance(rocr.pred3, "auc")@y.values)

## [1] 0.9940476

# This has not changed at all and still has given 99.4% accurate results  
  
# ROC Calculation for perf3  
perf3 = performance(rocr.pred3, "tpr", "fpr")  
plot(perf3)



# There is not much difference in the model except that the accuracy has come down to 97.8%   
# almost around 98% from 98.5% as we have increased the threshold from 0.5 to 0.8

## 3.2 K Nearest Neighbour Algorithm:

KNN, as it is popularly called, is a Supervised Machine Learning algorithm that helps to classify a data point to it’s assigned variable or target class, by analysing the nearest neighbours of the data point. Hence the name, nearest neighbour. And K is the value we assign to the model in order for the algorithm to focus on the desired K number of neighbours for analysis. KNN works only on numerical variables as it involves calculation of distances between each datapoint of the numeric variable. Mostly, the assigned K value is **odd** to avoid any ties between the selected data point.

Let us go about the algorithm

# Preparing data:  
  
library(dplyr)  
library(fastDummies)  
  
setwd("D:/R Progms")  
  
cardata = read.csv("Cars.csv")  
  
cardata = na.omit(cardata)  
  
# creating dummy variables for the target column  
  
cardata = dummy\_cols(cardata, select\_columns = "Transport")  
  
cardata = select(cardata, c(-9,-10,-12))  
  
View(cardata)  
# Let us remove the first two columns as knn requires only numeric data for distance  
cardatanew = select(cardata, c(-2))  
  
View(cardatanew)

### 3.2.1 Normalising data

We all know the KNN algorithm performs well with all the data being in numerical format. But we cannot be sure if all the numerical data are of the same units of measurement. Hence, in order to perform the model effectively and to obtain unbiased result, we will have to **normalise the numerical data** we are using for the algorithm. After going about the standardisation, we will obtain a data ranging between the values of 0 and 1, thus making it even.

By using the below set of codes, we will arrive at the data that is standardised for the model.

# Data Normalization function  
  
norm = function(x) { (x- min(x))/(max(x) - min(x)) }  
  
# Eliminating the dependent variable   
# As we only normalise the independent variables with the norm function  
  
norm.data = as.data.frame(lapply(cardatanew[,-8], norm))  
  
View(norm.data)  
  
# Let us now bind back the dependent variable to the normalised data  
# To facilitate model construction  
  
usable.data = cbind(cardatanew[,8], norm.data)  
  
str(usable.data)

## 'data.frame': 417 obs. of 8 variables:  
## $ cardatanew[, 8]: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Age : num 0.4 0.24 0.36 0.28 0.28 0.12 0.2 0.2 0.24 0.4 ...  
## $ Engineer : num 1 1 1 0 0 0 1 0 1 1 ...  
## $ MBA : num 0 0 0 0 0 0 1 0 0 0 ...  
## $ Work.Exp : num 0.2083 0.25 0.375 0.0417 0.125 ...  
## $ Salary : num 0.1564 0.0812 0.1782 0.0218 0.0614 ...  
## $ Distance : num 0.0941 0.1436 0.1436 0.1535 0.1733 ...  
## $ license : num 0 0 0 0 0 0 0 0 0 1 ...

View(usable.data)

### 3.2.2 KNN Classifier

Let us now partition the data into training and testing datasets in order to run the KNN algorithm. Here we have to keep in mind is an **optimum number for K is what that will produce a good model**. A smaller k will result in a rather overfitting model and a larger k may not capture as much details from the data, thereby missing on important ones.

By default, K can take a value of 19 or the square root of the number of observations in the data. But we have to make sure to select the optimum number so as to avoid over or under fitting of the model.

# Data Partioning  
  
library(caTools)  
  
set.seed(248)  
  
spl = sample.split(usable.data$`cardatanew[, 8]`, SplitRatio = 0.7)  
c.train = subset(usable.data, spl == T)  
c.test = subset(usable.data, spl == F)  
  
# KNN Classifier:  
table(sqrt(291), sqrt(126))

##   
## 11.2249721603218  
## 17.058722109232 1

# Let us take the k value to be 19 per usual.  
# As the sqrt of the observations results to about 20  
# As per our guess, either 19 or 21 would give us the best model accuracy  
  
library(class)  
  
pred1 = knn(c.train[-1], c.test[-1], c.train[,1], k = 19)   
table.knn1 = table(c.test[,1], pred1)  
sum(diag(table.knn1)/sum(table.knn1))

## [1] 0.9761905

# Accuracy of 97.62% keeping k as 19  
  
# Assigning k to be 21 and checking for accuracy:  
pred2 = knn(c.train[-1], c.test[-1], c.train[,1], k = 21)   
table.knn2 = table(c.test[,1], pred2)  
sum(diag(table.knn2)/sum(table.knn2))

## [1] 0.9761905

# Accuracy of 97.62% still remains he same as model 1  
  
pred3 = knn(c.train[-1], c.test[-1], c.train[,1], k = 23)   
table.knn3 = table(c.test[,1], pred3)  
sum(diag(table.knn3)/sum(table.knn3))

## [1] 0.9761905

# Accuracy still maintains the same to 97.62% as the previous model   
  
pred4 = knn(c.train[-1], c.test[-1], c.train[,1], k = 25)   
table.knn4 = table(c.test[,1], pred4)  
sum(diag(table.knn4)/sum(table.knn4))

## [1] 0.968254

# Accuracy falls down to 96.82%

### 3.2.3 Inference from KNN:

From this we may understand that a higher k value than the optimum would tend to miss out on many details and provide an underfitting model. While on the other hand, a model with k as low as 3 would give an overfitting model with accuracy seemingly greater than the other models. This may happen as the model tends to capture all the minute details including the noise from the data, which results in a model that tackles all minute issues.

Back to our models, the difference in accuracy from 19 to 25 revolved around the same level.The accuracy given by k=19 to k=23 are the same. And the accuracy given by the model with k=25 tends to be a little less than the first three models. From the above model accuracies we can confirm the model 2 or 3 with k = 21 and k=23 are good fit models as it is technically equal to the square root of the observations

## 3.3 Naïve Bayes:

Let us take the same cardata Naïve Bayes, an extremely powerful tool, is technically used only when the data set has all of its variables in a categorical format. However, it can also be used with continuous features but is more suited to categorical variables. Naïve Bayes is a recommended algorithm if all the independent and dependent variables are categorical. Naïve Bayes is a parametric algorithm. Hence, we may not obtain different results upon re executing the function unless the data remains unchanged.

In our case, although our dependent variable is categorical (as in the case of any data with a problem statement), we have equal number of independent variables in numeric format. Let us now convert the required variables to categorical variables

# Let us load the package e1071   
# Since e1071 has the naiveBayes function   
  
library(e1071)

## Warning: package 'e1071' was built under R version 3.6.1

library(caret)

## Warning: package 'caret' was built under R version 3.6.1

# Let us use the data cardata for this model and split for train and test data  
  
library(caTools)  
set.seed(248)  
sample2 = sample.split(cardata,SplitRatio = 0.7)  
train2 = subset(cardata,sample2 == TRUE)  
test2 = subset(cardata, sample2 == FALSE)  
  
# train2 has 279 obs and test2 has 138 obs  
  
# training the model on the train dataset  
NB = naiveBayes(Transport\_Car~., data = train2)  
print(NB)

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.91756272 0.08243728   
##   
## Conditional probabilities:  
## Age  
## Y [,1] [,2]  
## 0 26.48047 3.062680  
## 1 36.34783 3.127716  
##   
## Gender  
## Y Female Male  
## 0 0.2851562 0.7148438  
## 1 0.2173913 0.7826087  
##   
## Engineer  
## Y [,1] [,2]  
## 0 0.7148438 0.4523732  
## 1 0.8260870 0.3875534  
##   
## MBA  
## Y [,1] [,2]  
## 0 0.2539062 0.4360972  
## 1 0.1304348 0.3443502  
##   
## Work.Exp  
## Y [,1] [,2]  
## 0 4.792969 3.211516  
## 1 17.217391 3.930724  
##   
## Salary  
## Y [,1] [,2]  
## 0 12.98711 5.072497  
## 1 40.55217 9.695400  
##   
## Distance  
## Y [,1] [,2]  
## 0 10.70703 3.201279  
## 1 17.88261 2.652726  
##   
## license  
## Y [,1] [,2]  
## 0 0.1562500 0.3638034  
## 1 0.7826087 0.4217412

# From this model, we find that the classification given for gender is perfect as it is categorical  
# Whereas the other variables do not seem to be explained well by Naïve Bayes  
  
# Let us convert the required fields to categorical  
  
train2$Engineer = factor(train2$Engineer)  
train2$MBA = factor(train2$MBA)  
train2$license = factor(train2$license)  
train2$Transport\_Car = factor(train2$Transport\_Car)  
  
test2$Engineer = factor(test2$Engineer)  
test2$MBA = factor(test2$MBA)  
test2$license = factor(test2$license)  
test2$Transport\_Car = factor(test2$Transport\_Car)  
  
# Creating naiveBayes model only on categorical variables  
NB1 = naiveBayes(Transport\_Car~Gender+Engineer+MBA+license, data = train2)  
print(NB1)

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.91756272 0.08243728   
##   
## Conditional probabilities:  
## Gender  
## Y Female Male  
## 0 0.2851562 0.7148438  
## 1 0.2173913 0.7826087  
##   
## Engineer  
## Y 0 1  
## 0 0.2851562 0.7148438  
## 1 0.1739130 0.8260870  
##   
## MBA  
## Y 0 1  
## 0 0.7460938 0.2539062  
## 1 0.8695652 0.1304348  
##   
## license  
## Y 0 1  
## 0 0.8437500 0.1562500  
## 1 0.2173913 0.7826087

NB1.pred = predict(NB1, test2, type = "class" )  
  
table(NB1.pred, test2$Transport\_Car, dnn = c("Prediction", "Actual"))

## Actual  
## Prediction 0 1  
## 0 126 12  
## 1 0 0

# This table gives an image that the model has predicted all the observations to be 0   
# while there are 12 entries that are actually 1. And the model has made wrong prediction.  
  
# Acuracy  
  
(126+0)/nrow(test2)

## [1] 0.9130435

confusionMatrix(NB1.pred, test2$Transport\_Car, dnn = c("Prediction", "Actual"))

## Confusion Matrix and Statistics  
##   
## Actual  
## Prediction 0 1  
## 0 126 12  
## 1 0 0  
##   
## Accuracy : 0.913   
## 95% CI : (0.853, 0.9543)  
## No Information Rate : 0.913   
## P-Value [Acc > NIR] : 0.576044   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 0.001496   
##   
## Sensitivity : 1.000   
## Specificity : 0.000   
## Pos Pred Value : 0.913   
## Neg Pred Value : NaN   
## Prevalence : 0.913   
## Detection Rate : 0.913   
## Detection Prevalence : 1.000   
## Balanced Accuracy : 0.500   
##   
## 'Positive' Class : 0   
##

# the confusion matrix gives an accuracy of 91.3% and since the p value being 0.0015 it is a significant model

We can see here from the confusion matrix that the model has not made correct predictions when compared to the other two models of KNN and Logistic Regression. Though the model has made correct prediction of 126 people not commuting by car and has made only 12 wrong assumptions, contributing to about 91% accuracy. The fact that it did not predict any number of employees using the car is an indicator that the model fails in its purpose.

## 3.4 Model comparison - Classification:

library(kknn)

## Warning: package 'kknn' was built under R version 3.6.1

##   
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':  
##   
## contr.dummy

library(klaR)

## Warning: package 'klaR' was built under R version 3.6.1

library(LiblineaR)  
library(rpart)

## Warning: package 'rpart' was built under R version 3.6.1

library(fastDummies)  
library(caret)  
library(dplyr)  
  
# loading the fresh data:  
  
setwd("D:/R Progms")  
  
car.d = read.csv("Cars.csv")  
  
# creating dummy variables for the target column  
  
car.dn = dummy\_cols(car.d, select\_columns = "Transport")  
  
car.dn$Transport\_Car = factor(car.dn$Transport\_Car, labels = c("No", "Yes"))  
  
car.dn = na.omit(car.dn)  
  
car.dn = select(car.dn, c(-9,-10,-12))  
  
View(car.dn)  
  
str(car.dn)

## 'data.frame': 417 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\_Car: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(\*, "na.action")= 'omit' Named int 243  
## ..- attr(\*, "names")= chr "243"

# CROSS VALIDATION: technique used is K Fold cross validation   
  
ctrl <- trainControl(method = "repeatedcv", classProbs = T, number = 10, summaryFunction = twoClassSummary, repeats = 2)  
  
# Logistic Regression model  
  
set.seed(248)  
  
modellog <- train(Transport\_Car ~ ., data = car.dn, method = "regLogistic", metric = 'ROC', trControl = ctrl)  
  
print(modellog)

## Regularized Logistic Regression   
##   
## 417 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 2 times)   
## Summary of sample sizes: 376, 376, 375, 376, 376, 374, ...   
## Resampling results across tuning parameters:  
##   
## cost loss epsilon ROC Sens Spec   
## 0.5 L1 0.001 0.9909329 0.9908232 0.8958333  
## 0.5 L1 0.010 0.9909329 0.9934548 0.8958333  
## 0.5 L1 0.100 0.9879976 0.9947706 0.8458333  
## 0.5 L2\_dual 0.001 0.9920294 0.9895074 0.8958333  
## 0.5 L2\_dual 0.010 0.9920294 0.9895074 0.8958333  
## 0.5 L2\_dual 0.100 0.9920294 0.9895074 0.8958333  
## 0.5 L2\_primal 0.001 0.9920294 0.9895074 0.8958333  
## 0.5 L2\_primal 0.010 0.9917004 0.9895074 0.8958333  
## 0.5 L2\_primal 0.100 0.9846070 0.9960526 0.9208333  
## 1.0 L1 0.001 0.9913715 0.9895074 0.8958333  
## 1.0 L1 0.010 0.9907220 0.9881916 0.8958333  
## 1.0 L1 0.100 0.9887567 0.9921390 0.8541667  
## 1.0 L2\_dual 0.001 0.9919197 0.9895074 0.8958333  
## 1.0 L2\_dual 0.010 0.9919197 0.9895074 0.8958333  
## 1.0 L2\_dual 0.100 0.9915908 0.9895074 0.8958333  
## 1.0 L2\_primal 0.001 0.9920294 0.9908232 0.8958333  
## 1.0 L2\_primal 0.010 0.9913715 0.9895074 0.8958333  
## 1.0 L2\_primal 0.100 0.9846070 0.9960526 0.9208333  
## 2.0 L1 0.001 0.9906123 0.9881916 0.8958333  
## 2.0 L1 0.010 0.9922487 0.9881916 0.8958333  
## 2.0 L1 0.100 0.9897689 0.9855601 0.8166667  
## 2.0 L2\_dual 0.001 0.9908232 0.9907895 0.8958333  
## 2.0 L2\_dual 0.010 0.9908232 0.9907895 0.8833333  
## 2.0 L2\_dual 0.100 0.9901738 0.9907895 0.8833333  
## 2.0 L2\_primal 0.001 0.9909329 0.9907895 0.8958333  
## 2.0 L2\_primal 0.010 0.9917004 0.9895074 0.8958333  
## 2.0 L2\_primal 0.100 0.9842780 0.9960526 0.9208333  
##   
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were cost = 2, loss = L1 and epsilon  
## = 0.01.

# Naive bayes model  
  
modelnb <- train(Transport\_Car ~ ., data = car.dn, method = "nb",   
 metric = 'ROC', trControl = ctrl)

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 10

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 10

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 31  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 31

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 2

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 31

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 2

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 31

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 24  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 24

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 25

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 25

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 39

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 39

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 12

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 12

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 8  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 7

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 32

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 7

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 32

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 7

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 14

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 22

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 7

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 14

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 22

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 25  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 25

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 39

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 39

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 12

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 15

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 12

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 15

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 7

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 35

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 7

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 11

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 35

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 3

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 28

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 31

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 3

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 28

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 31

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 5

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 36

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 5

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 27

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 36

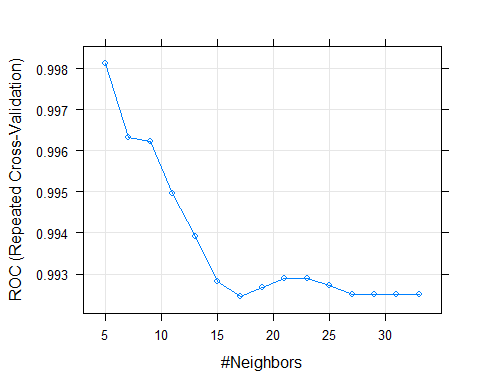
print(modelnb)

## Naive Bayes   
##   
## 417 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 2 times)   
## Summary of sample sizes: 374, 375, 376, 375, 376, 376, ...   
## Resampling results across tuning parameters:  
##   
## usekernel ROC Sens Spec   
## FALSE 0.9934548 0.9764170 0.9208333  
## TRUE 0.9963057 0.9948043 0.9208333  
##   
## Tuning parameter 'fL' was held constant at a value of 0  
## Tuning  
## parameter 'adjust' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were fL = 0, usekernel = TRUE  
## and adjust = 1.

# KNN model :  
  
modelknn <- train(Transport\_Car ~ ., data = car.dn, method = "knn", metric = 'ROC',   
 trControl = ctrl, tuneLength = 15)  
  
print(modelknn)

## k-Nearest Neighbors   
##   
## 417 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 2 times)   
## Summary of sample sizes: 375, 374, 375, 376, 376, 376, ...   
## Resampling results across tuning parameters:  
##   
## k ROC Sens Spec   
## 5 0.9981360 0.9973684 0.9083333  
## 7 0.9963268 0.9973684 0.8958333  
## 9 0.9962171 0.9973684 0.8958333  
## 11 0.9949646 0.9960864 0.8208333  
## 13 0.9939229 0.9974022 0.8208333  
## 15 0.9928264 0.9974022 0.7833333  
## 17 0.9924426 0.9974022 0.7833333  
## 19 0.9926619 0.9947706 0.7833333  
## 21 0.9928812 0.9895074 0.7833333  
## 23 0.9928812 0.9855601 0.7833333  
## 25 0.9927168 0.9842443 0.7833333  
## 27 0.9924975 0.9842443 0.7833333  
## 29 0.9924975 0.9842443 0.7833333  
## 31 0.9924975 0.9855601 0.7833333  
## 33 0.9924975 0.9868758 0.7833333  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was k = 5.

plot(modelknn)



# The plot has identified the optimum number of k as 5 by using ROC

## 3.5 Remarks on the best model:

We have constructed 3 models plus a regression model with the SMOTE data. And based on the accuracies, we have validated the best model

* Our logistic regression model gives a better accuracy of 98.5% when compared to our other models on KNN and Naive bayes.
* The KNN model has also done well in giving a good accuracy of 97.8%
* Though the Naive Bayes model also gave an accuracy of 91%, it did not predict any employee who used cars, though there were actually employees who used cars. The model predicted everyone as non car users.

Hence we classify the Logistic regression model to be a good fit.

## 3.6 Bootstrap Aggregating (BAGGING):

Bootstrap Aggregating, better known as Bagging, is a machine learning ensemble technique which helps to imrpove the stability and accuracy of a model. Bagging helps in reducing the variance in a model, thus preventing overfitting a model. What bagging does is, it splits the train data into many more chunks or combinations with replacements from the training set. This helps in finalising a better model by aggregating the results from the model conducted on the many more subsets of the training data.

Let us look at the code

library(fastDummies)  
library(dplyr)  
library(ipred)

## Warning: package 'ipred' was built under R version 3.6.1

library(rpart)  
  
setwd("D:/R Progms")  
  
car.data = read.csv("Cars.csv")  
  
# creating dummy variables for the target column  
  
car.data = dummy\_cols(car.data, select\_columns = "Transport")  
  
car.data = na.omit(car.data)  
  
car.data <- select(car.data, c(-9,-10,-12))  
  
View(car.data)  
  
str(car.data)

## 'data.frame': 417 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\_Car: int 0 0 0 0 0 0 0 0 0 0 ...  
## - attr(\*, "na.action")= 'omit' Named int 243  
## ..- attr(\*, "names")= chr "243"

# Splitting data to train and test.  
  
library(caret)  
library(caTools)  
  
set.seed(248)  
split = sample.split(car.data,SplitRatio = 0.7)  
cartrain = subset(car.data,split == TRUE)  
cartest = subset(car.data, split == FALSE)  
  
# cartrain has 279 obs  
# cartest has 138 obs  
  
# control details for rpart  
  
rpctrl = rpart.control(maxdepth=7, minsplit=5)  
  
# Model for bagging  
car.bagging <- bagging(Transport\_Car~., data = cartrain, control = rpctrl)  
  
# Running on the test data for prediction  
cartest$pred = predict(car.bagging, cartest)  
  
table(cartest$Transport\_Car,cartest$pred >0.5)

##   
## FALSE TRUE  
## 0 125 1  
## 1 1 11

# Accuracy  
  
(125+11)/nrow(cartest)

## [1] 0.9855072

# This has given an accuracy of 98.55% which a little much more than our KNN model on the data.  
# This accuracy is almost equal to our logistic regression model  
  
  
# Tweaking the threshold to 0.7  
table(cartest$Transport\_Car,cartest$pred >0.7)

##   
## FALSE TRUE  
## 0 126 0  
## 1 1 11

(126+11)/nrow(cartest)

## [1] 0.9927536

# This tweak shows that the model with some bagging has booted the accuracy to 99.28%   
# This is significantly higher than the first model as the model has identified all those who do not   
# use a car correctly. It has only mispredicted at one instance and correctly predicted those who used  
# a car  
  
# Thus by bagging the imbalanced data, we have obtained the best model of all after bagging.   
# And the accuracy is higher than all of our other models (logistic regression, knn and naive bayes)

## 3.7 Boosting:

### 3.7.1 GBM model 1

# Basic boosting model:  
  
library(gbm)

## Warning: package 'gbm' was built under R version 3.6.1

## Loaded gbm 2.1.5

library(xgboost)

## Warning: package 'xgboost' was built under R version 3.6.1

##   
## Attaching package: 'xgboost'

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

gbm.fit <- gbm(  
 formula = Transport\_Car ~ .,  
 distribution = "bernoulli",  
 data = cartrain,  
 n.trees = 500,   
 interaction.depth = 1,  
 shrinkage = 0.001,  
 cv.folds = 5,  
 n.cores = NULL,   
 verbose = FALSE  
)   
  
# we are using bernoulli because we are doing a logistic and want probabilities  
# n.trees - the number of stumps  
# interaction depth - number of splits it has to perform on a tree (starting from a single node)  
# shrinkage is used for reducing, or shrinking the impact of each additional fitted base-learner(tree)  
# c.v. folds - cross validation folds  
# n.cores will use all cores by default  
# verbose - after every tree/stump it is going to show the error and how it is changing  
  
cartest$pred <- predict(gbm.fit, cartest, type = "response")

## Using 500 trees...

# we have to put type="response" just like in logistic regression else we will have log odds  
  
table(cartest$Transport\_Car,cartest$pred>0.5)

##   
## FALSE  
## 0 126  
## 1 12

(126+0)/nrow(cartest)

## [1] 0.9130435

# This model using the gbm boost has done a very good fit of correctly predicting all the 0s. So a 100% correct prediction  
# But when it comes to predicting the employees using car as a mode of transport, the model has not predicted any of it correctly, making all predictions as 0

Let us tweak the gbm fit a little to get the best fit. Since XGBoost works with matrices that contain all numeric variables, we are unable to deplot the same with our data. Moreover, our data is quie small having only 417 observations and the model does not take a lot of time to fit the model to our data. Hence the Gradient Boost Model would suit best for this data.

### 3.7.2 GBM model 2

# GBM model 2  
  
gbm.fit2 <- gbm(  
 formula = Transport\_Car ~ .,  
 distribution = "bernoulli",  
 data = cartrain,  
 n.trees = 1000,   
 interaction.depth = 1,  
 shrinkage = 0.001,  
 cv.folds = 5,  
 n.cores = NULL,   
 verbose = FALSE  
)   
  
# for the above, we are increasing the n.trees to 1000. Thus the model will be build using 100 trees  
  
# Let us test it on our validation set  
cartest$pred <- predict(gbm.fit2, cartest, type = "response")

## Using 1000 trees...

table(cartest$Transport\_Car,cartest$pred>0.5)

##   
## FALSE TRUE  
## 0 126 0  
## 1 3 9

(126+9)/nrow(cartest)

## [1] 0.9782609

# This model has got better than the last one  
# In this fit, we still find that the model has correctly predicted the false values a 100% correctly  
# And also a better identification (75%) of the true values of employees using car as a commute   
# And this gives an accuracy of 97.83% which is much better than our last model with 91.3%

### 3.7.3 GBM model 3

# Let us now keep the n.trees as 1000 ,the same as last model   
# Now we tweak the learning rate to 0.01 keeping the rest same as our last model  
# Let us do our 3rd gbm fit on the training set  
   
gbm.fit3 <- gbm(  
 formula = Transport\_Car ~ .,  
 distribution = "bernoulli",  
 data = cartrain,  
 n.trees = 1000,   
 interaction.depth = 1,  
 shrinkage = 0.01,  
 cv.folds = 5,  
 n.cores = NULL,   
 verbose = FALSE  
)  
  
# Let us test it on our validation set  
cartest$pred <- predict(gbm.fit3, cartest, type = "response")

## Using 566 trees...

# Accuracy - confusion matrix  
table(cartest$Transport\_Car,cartest$pred>0.5)

##   
## FALSE TRUE  
## 0 126 0  
## 1 1 11

(126+11)/nrow(cartest)

## [1] 0.9927536

# This is the best fit tht could be for the data, as this gives an accuracy of 99.28%  
# Firstly, it has correctly predicted all the 0s which is the employees who do not use car as their commute  
# Secondly, this model has almost predicted all the employees who choose to use car as their commute, leaving out just one

# 4. Actionable Insights and Recommendations

From all the above conducted models, we can say that there has been a huge improvement on the model effeciency. While before constructing a model, we can assume that even with no model built, we can be 91% accuracte of our findings because we were presentee with an unbalanced data, that gave more than 90% information about the employees who chose not to travel by car. While our problem statement required us to predict the employees who will use car as their mode of transport.

* The first problem we had was that the data was **unbalanced** with only less than 10% of the target class. Hence we used SMOTE for our data, which undersampled our majority class to roughly 72% and our minority class (target) But we did not see a huge improvement in the logistic regression model on the smoted data. The accuracies almost remained the same for the logistic model on the imbalanced and the smote data. Hence that did not help.
* We conducted models using 3 algorithms. Logistic regression, K Nearest Neighbour and Naive Bayes. Out of which, logistic regression performed well followed by KNN. But the Naive Bayes model failed in its purpose becasue it did not predict the employees who actually use cars as their mode of commute. Though it had a very good accuracy of correctly predicting the employees who did not use cars
* Upon EDA, we found that there was multicollinearity as variables such as age, work experience and salary were very much related to each other and had a positive correlation. We also found that most of the people with a higher age, experience and salary chose to travel by cars. This may be because, the growth is positively correlated to affordability. It may also be that the seniority has some status issues and the reason why the senior people chose to travel by cars. This also explains why a lot of young employees (in work experience as well) chose two wheelers or public transport
* We also figured a reason why most of them chose to travel by public transport as most of them did not own a license. And we can also say it was women with lesser percentage of licence (roughly 6%) than the men (26.3%)
* The data in itself is a small one and a larger data set would help. This is because the target was not well achieved as the model could not be well trained with just limited amount of data on the target (predicting the number of employees who will use car needs data that actually has enough number of entries that have people using cars to facilitate in training the model).