Car Usage Prediction

Managerial Report by Chetan Suvarna

Table of Content	Page No.

2 Project Objectives

The objective of the report is to explore the data file Cars.csv in R and generate insights about the data set. This exploration report will consists of the following scenarios:

- Exploratory Data Analysis of the data
- Build appropriate models using the data
- Interpreting the model outputs and performing the necessary modifications wherever eligible
- Check the performance of all the models that you have built
- Determining whether or not an employee will use car as a means of transport
- Significant predictor variables behind the decision

3 Exploratory Data Analysis – Step by step approach

A Typical Data exploration activity consists of the following steps:

1. Environment Set up and Data Import
2. Variable Identification
3. Visualisation Plots

We shall follow these steps in exploring the provided dataset.

3.1 Environment Set up and Data Import

3.1.1 Set up working Directory

Setting a working directory on starting of the R session makes importing and exporting data files and code files easier. Basically, working directory is the location/ folder on the PC where you have the data, codes etc. related to the project.
Please refer Appendix A for Source Code.

3.1.2 Import and Read the Dataset

The given dataset is in .csv format. Hence, the command 'read.csv' is used for importing the file.
Please refer Appendix A for Source Code.

3.2 Variable Identification

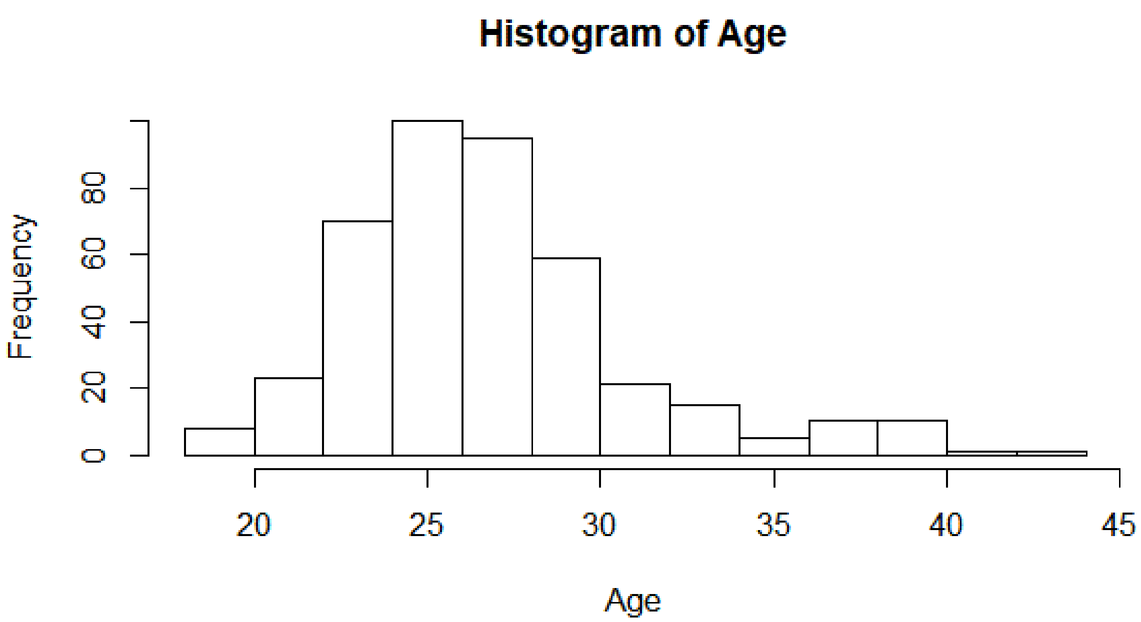
The length and breadth of the data was examined and the names of the data were pulled from the dataset. The data consisted of 9 variables consisting of 418 employees determining whether or not they took car as a medium of transport. The string type of the data was also verified by using the str() function.

3.3 Visualisation Plots

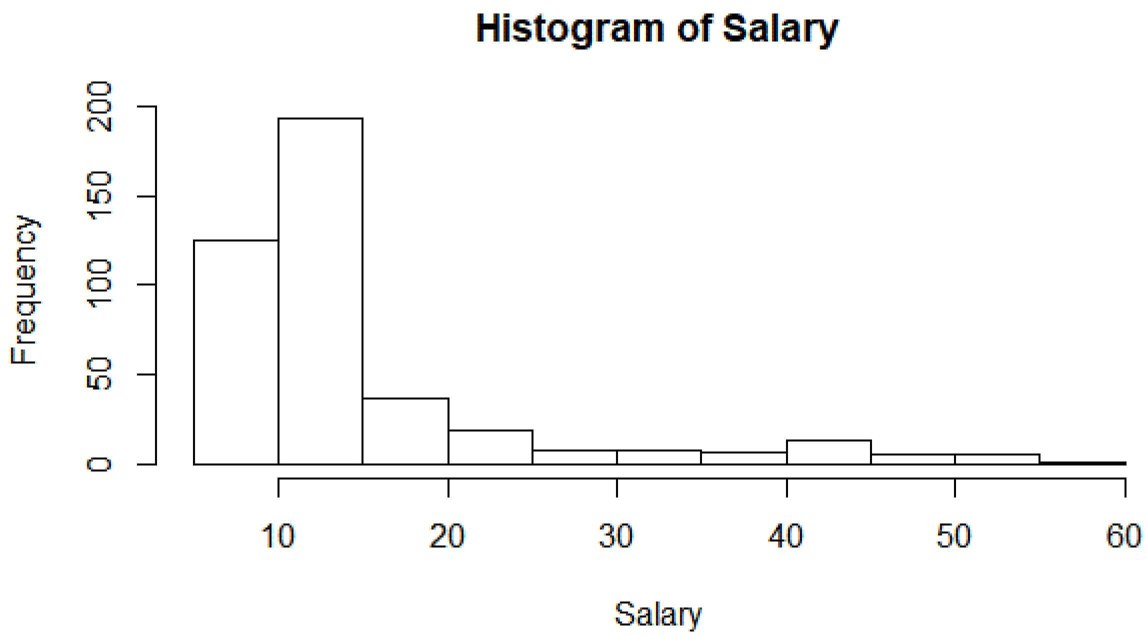
Histogram Plot

Below are the histogram plots for Work experience, Age and Salary from the dataset.

- The above histogram depicts that a major portion of employees had experience between 0-5 years.



- The histogram for Age weeks displays a major proportion of employees ageing within 23-30 which is in line with the work experience.



- A majority of employees earned from 0-15 Lakhs with employees earning between 10-15 Lakhs accounting the maximum.

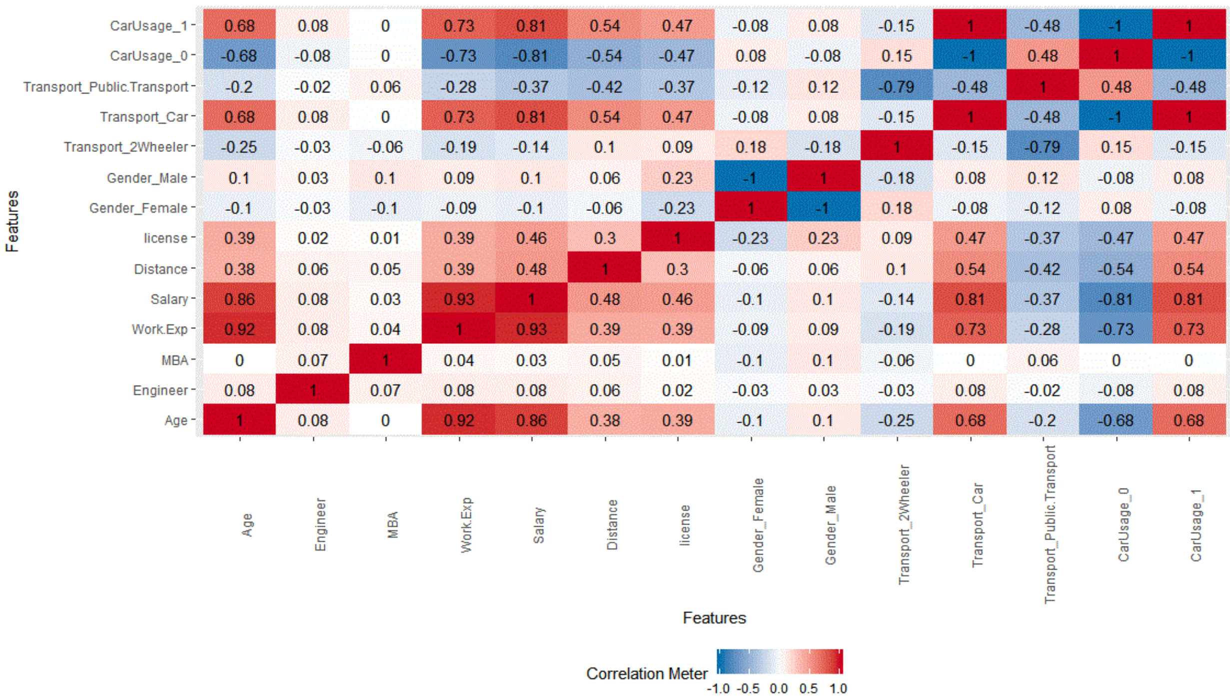
Missing Values

No missing values were found in the data set. The below code was used to determine the missing values:

```
nrow(cars[is.na(cars),])
```

Corrplot

Corrplot was plotted between all the variables to determine the correlation amongst the variables and the output is displayed in the below figure.



From the above corrplot the below observations can be made:

- Salary and Work Exp are highly correlated.
- Work Exp is also related to Age.

4. Refining the dataset

- A new variable CarUsage has been created. Employees using car as Transport has been marked as 1, whereas 2-wheeler and bus has been marked 0.

- The output of car/non-car users has been mentioned below

```
> table(cars$CarUsage)

  0   1
383  35
```

- Only 8.37% of the entire dataset has been using cars as the means of Transport.
- All the variables were converted to numeric using as.numeric() function in order to determine Ordinary Least Square.
- Ordinary Least Square was found using lm function
- VIF was calculated for the dataset mentioned below:

```
> vif(OLS.full)
```

Age	Gender	Engineer	MBA	Work.Exp	Salary	Distance	license
7.045897	1.070569	1.01335	1.031029	14.575246	9.179929	1.346537	1.364719

- Dimensionality reduction was performed and Work.Exp was removed from the dataset due to **multicollinearity**
- The VIF values found after dimensionality reduction was found to be good as mentioned below.

```
> vif(OLS.full1)
```

Age	Gender	Engineer	MBA	Salary	Distance	license
3.823939	1.069003	1.013267	1.019841	4.481698	1.321141	1.340242

- Cars2 is taken as the final dataset which has been split into train and test in 70:30 ratio.
- The data has been split based on CarUsage variable.

5. Model algorithms with Performance Measures

1. Logistic Regression

- Loading the data
- Found generalised linear model using glm() function
- % of sample in train, test and full data was checked and found to be similar
-

Performance Measures

- **Confusion matrix**

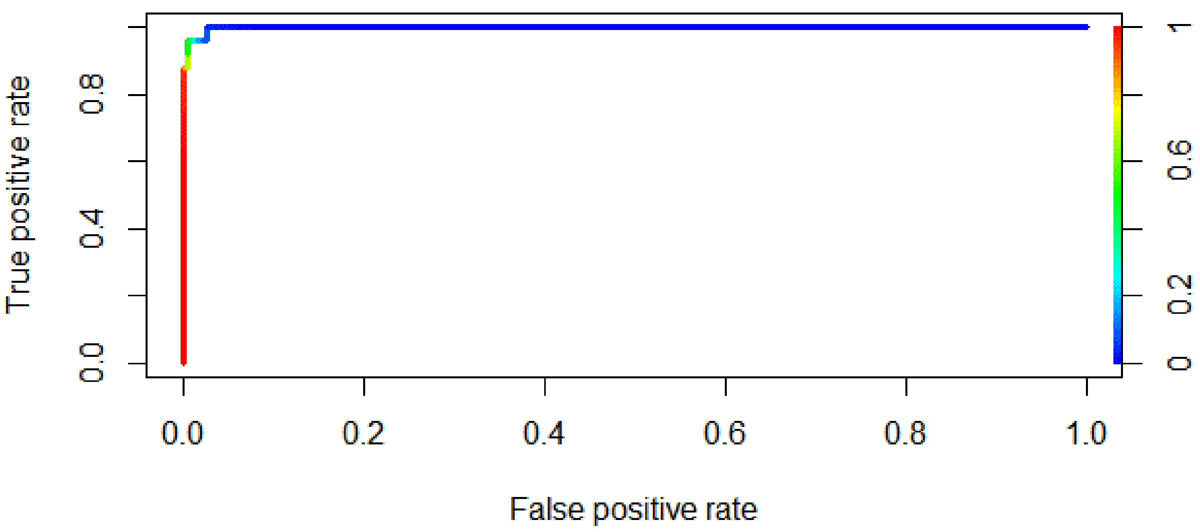
```
> conf_mat

  FALSE  TRUE
0   114     0
1     1     9
```

- **ROC curve**

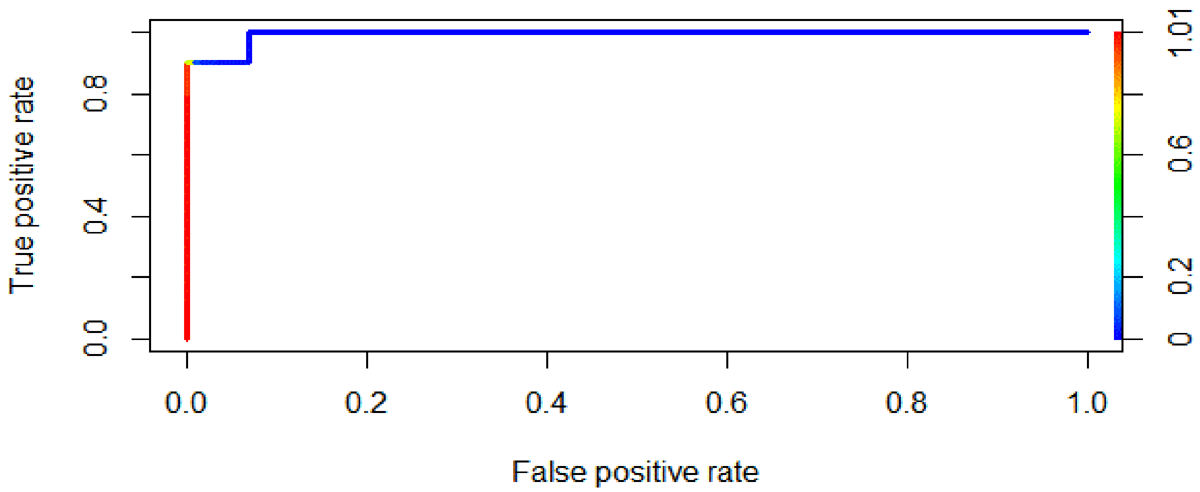
The graphs for all the performance measures have been plotted below.

Train



Accuracy: 0.9986617

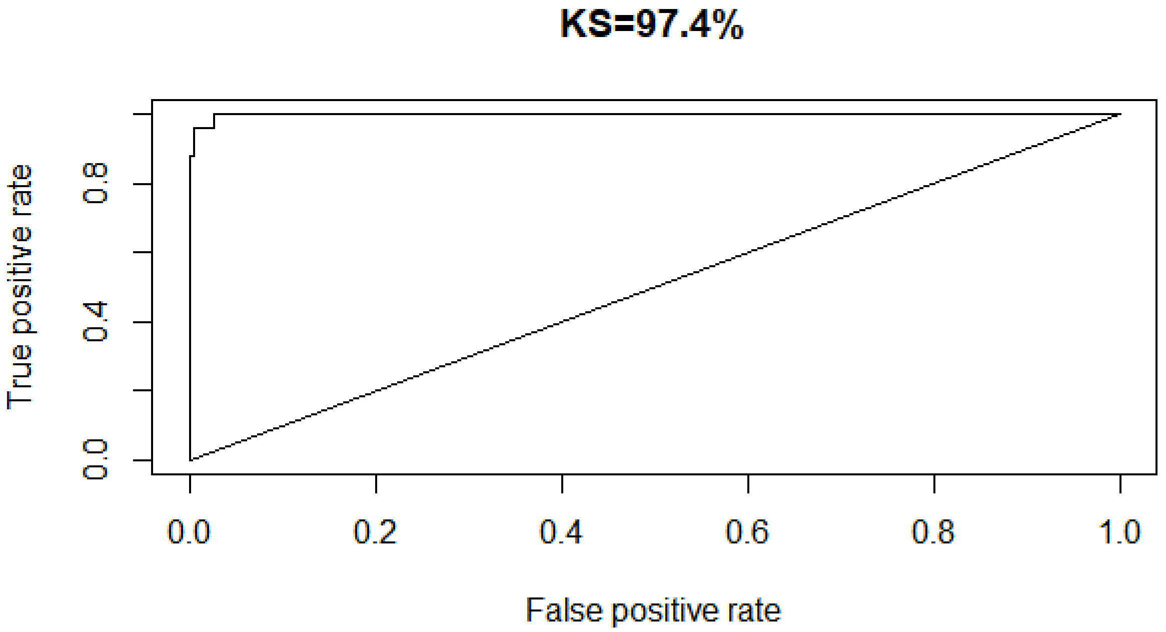
Test



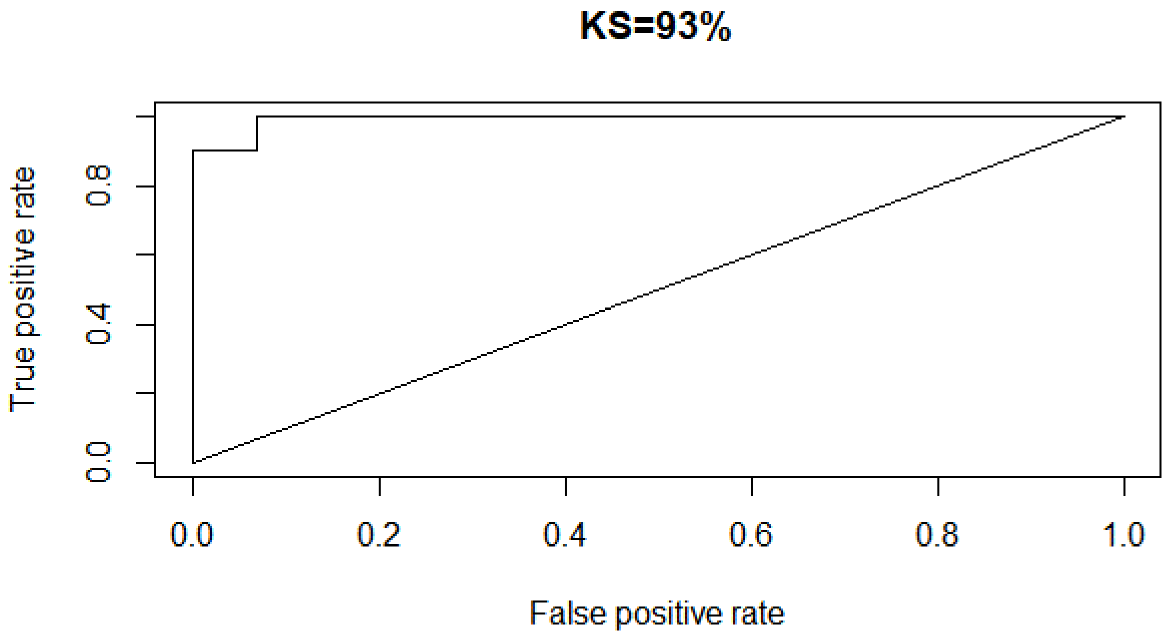
Accuracy: 0.9929825

- **Kolmogorov-Smirnov**

Train



Test



2. KNN

- Converted variables to numerical string to be used for KNN
- Converted the data to data frame and then to factor
- Tried different values of k to find the best KNN model which could minimise the False Negatives
- At K= 17 we found the best matrix with minimal False negatives

```
knn_fit
      0   1
  0 114   0
  1   2   8
```

Accuracy: 0.983871
Total Loss: 0.01637931

3. Naïve Bayes

```
  pred_nb
     0   1
0  111   3
1    1   9
```

Accuracy: 0.9677
Total loss: 0. 0.032258

Naïve Bayes is not preferred because the algorithm makes a very strong assumption about the data having features independent of each other while in reality, they may be dependent in some way. In other words, it assumes that the presence of one feature in a class is completely unrelated to the presence of all other features. If this assumption of independence holds, Naive Bayes performs extremely well and often better than other models. **Naive Bayes can also be used with continuous features but is more suited to categorical variables. If all the input features are categorical, Naive Bayes is recommended. However, in case of numeric features, it makes another strong assumption which is that the numerical variable is normally distributed which was not in the current case.**

4. Bagging

- Bagging was determined using bagging() and by adjusting the maximum depth and minimum split values.
- At maxdepth=6, minsplit=20 we found the below confusion matrix

FALSE TRUE
0 114 0
1 2 8

- Accuracy : 0.9839
- At maxdepth=5, minsplit=15, we found the below confusion matrix

FALSE TRUE
0 114 0
1 3 7

- Accuracy : 0.9758
- **Bagging with max depth=6;minsplit=20 is better**
-

5. Boosting

- A general boosting was performed on the dataset and below matrix was found with 98.39% accuracy.

	Reference	
Prediction	0	1
0	114	2
1	0	8

- XGBOOST was performed by adjusting the eta, max depth and no. of rounds
- The best values were found to be below:

```
+    max_depth = 5,
+    nrounds = 50,
+    nfold = 5,
```

- The confusion matrix was formed as below:

```
FALSE  TRUE
  0    114     0
  1      2     8
```

6. SMOTE

- Partitioned dataset in the ratio 70:30 has been taken
- SMOTE has been tried with different perc.over and perc.under values

- At `perc.over = 250,perc.under = 150`

```
FALSE  TRUE
  0    111     3
  1      1     9
```

- At `perc.over = 275,perc.under = 150`

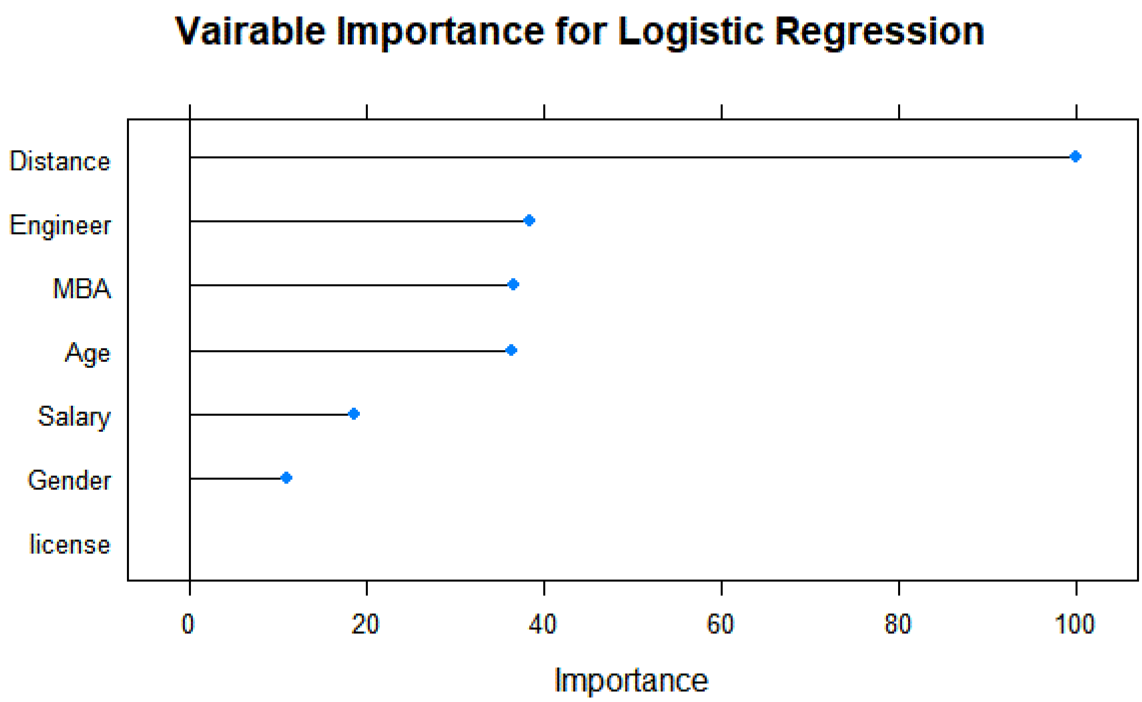
```
  FALSE TRUE
0   114    0
1     1    9
```

- At `perc.over = 250,perc.under = 100`

```
  FALSE TRUE
0   114    0
1     0   10
```

- At perc.over = 250,perc.under = 100, we got the best confusion matrix for SMOTE.

➢ Variable Importance for the dataset



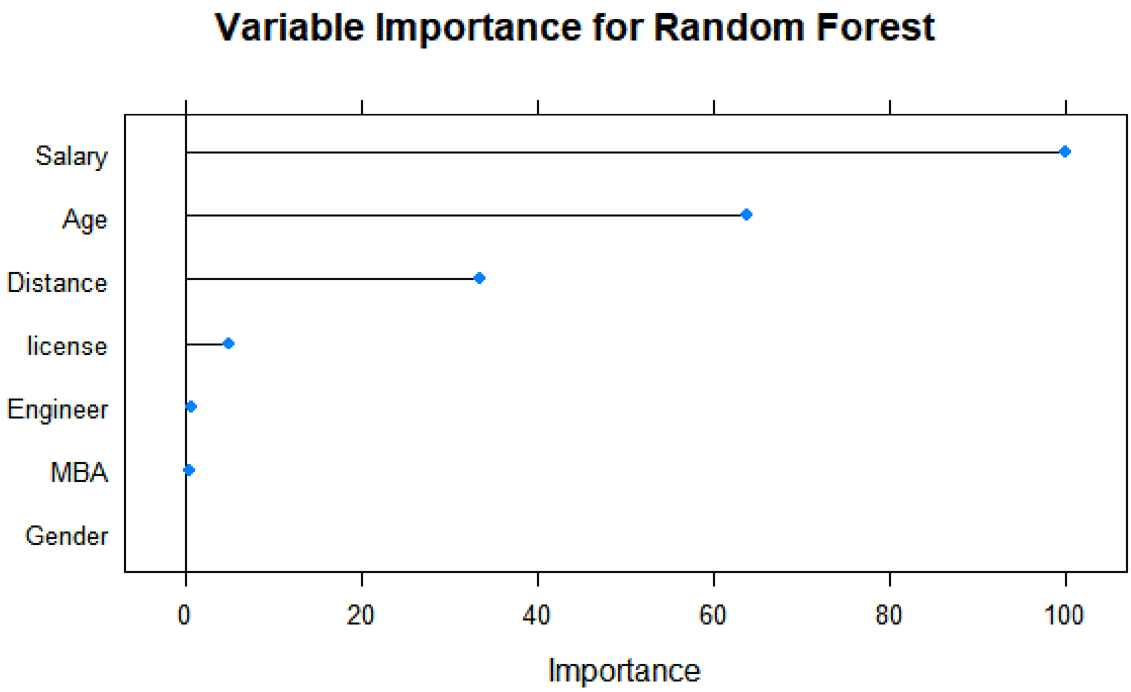
- **Distance and Engineer turn out to be the most important variables in the model**

7. Random Forest

- Confusion Matrix and Statistics

```
          Reference
Prediction   0   1
         0 114   2
         1   0   8
```

- OOB estimate of error rate: 1.7%
- Accuracy : 0.9839



- `Salary and age are most significant variables in the dataset.`

6. Conclusion

Performance measures were obtained for various model algorithms with a data split of 70:30 for train and test from the dataset. Based on the confusion matrixes and plots obtained we can conclude that the model performs best with a combination of learning rate, number of rounds and maximum depth of 0.7, 50 and 5 respectively for XGBOOST when applied on SMOTE with an over and under sampling of 250 and 100 respectively. Variable Work Exp. induced multicolllinearity with Salary and Age and hence was removed. The response variable CarUsage is affected by Salary, Age, Distance, Licence, Engineer, MBA and Gender with Distance and Engineer turning out to be the most important variables in making the decision of using cars as the means of transport. The importance of the variable can be identified by the legend of the correlated coefficients.

7 Appendix A – Source Code

Here is the code produced in RStudio for doing an analysis on Product Service Management.

```
> #Machine Learning assignment on cars dataset
>
> #Loading libraries
> library(car)
> library(caret)
> library(class)
> library(devtools)
> library(e1071)
> library(ggord)
> library(ggplot2)
> library(Hmisc)
> library(klaR)
> library(MASS)
> library(nnet)
> library(plyr)
> library(pROC)
> library(psych)
> library(scatterplot3d)
> library(SDMTools)
> library(dplyr)
> library(ElemStatLearn)
> library(rpart)
> library(rpart.plot)
> library(randomForest)
> library(neuralnet)
> library(caret)
> library(car)
> library(DMwR)
> library(rattle)
> cars <- read.csv("Cars.csv")
> head(cars)
  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
> View(cars)
> attach(cars)
The following objects are masked from carsdatatrain:

    Age, Distance, Engineer, Gender, license, MBA, Salary

The following objects are masked from cars (pos = 11):

    Age, Distance, Engineer, Gender, license, MBA, Salary, Transport, Work.Exp
>summary(cars)
      Age            Gender       Engineer          MBA              Work.Exp
 Min.   :18.00   Female:121   Min.   :0.0000   Min.   :0.0000   Min.   : 0.000
 1st Qu.:25.00   Male  :297   1st Qu.:0.2500   1st Qu.:0.0000   1st Qu.: 3.000
 Median :27.00                Median :1.0000   Median :0.0000   Median : 5.000
 Mean   :27.33                Mean   :0.7488   Mean   :0.2608   Mean   : 5.873
```

```
 3rd Qu.:29.00                    3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 8.000
 Max.   :43.00                    Max.   :1.0000   Max.   :1.0000   Max.   :24.000
     Salary          Distance        license                    Transport
 Min.   : 6.500   Min.   : 3.20   Min.   :0.0000   2Wheeler        : 83
 1st Qu.: 9.625   1st Qu.: 8.60   1st Qu.:0.0000   Car             : 35
 Median :13.000   Median :10.90   Median :0.0000   Public Transport:300
 Mean   :15.418   Mean   :11.29   Mean   :0.2033
 3rd Qu.:14.900   3rd Qu.:13.57   3rd Qu.:0.0000
 Max.   :57.000   Max.   :23.40   Max.   :1.0000


> summary(Transport)
        2Wheeler              Car Public Transport
              83               35              300
> #to check any  missing values
> nrow(cars[is.na(cars),])
[1] 0
> str(cars)
'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 ...
>
> cars$CarUsage=ifelse(cars$Transport=='Car',1,0)
> table(cars$CarUsage)

  0   1 
383  35 
> sum(cars$CarUsage)/nrow(cars)
[1] 0.08373206

> cars$CarUsage=as.factor(cars$CarUsage)
> hist(Work.Exp)
> hist(Age)
> hist(Salary)
> #displays that the major portion of the group falls in the bracket of 22-30
yrs
> library(DataExplorer)
> plot_correlation(cars)
> #displays that age, salary and work experience are related and we are exclud
ing
> #variable work exp  from the dataset
> str(cars)
'data.frame':	418 obs. of  10 variables:
 $ Age      : int  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 2 2 2 2 ...
 $ Engineer : int  1 1 1 0 0 0 1 0 1 1 ...
 $ MBA      : int  0 0 0 0 0 0 1 0 0 0 ...
 $ Work.Exp : int  5 6 9 1 3 3 3 0 4 6 ...
 $ Salary   : num  14.4 10.6 15.5 7.6 9.6 9.5 11.7 6.5 8.5 13.7 ...
 $ Distance : num  5.1 6.1 6.1 6.3 6.7 7.1 7.2 7.3 7.5 7.5 ...
 $ license  : int  0 0 0 0 0 0 0 0 0 1 ...
 $ Transport: Factor w/ 3 levels "2Wheeler","Car",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ CarUsage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> cars$Age<-as.numeric(cars$Age)
> cars$Gender<-as.numeric(cars$Gender)
```

```
> cars$Engineer<-as.numeric(cars$Engineer)
> cars$MBA<-as.numeric(cars$MBA)
> cars$Work.Exp<-as.numeric(cars$Work.Exp)
> cars$license<-as.numeric(cars$license)
> cars$license<-as.numeric(cars$license)
> cars$CarUsage=as.numeric(cars$CarUsage)
> str(cars)
'data.frame':	418 obs. of  10 variables:
 $ Age      : num  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender   : num  2 2 1 2 1 2 2 2 2 2 ...
 $ 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  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  : num  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 ...
 $ CarUsage : num  1 1 1 1 1 1 1 1 1 1 ...
> OLS.full<-lm(CarUsage~Age+Gender+Engineer+MBA+Work.Exp+Salary+Distance+
+                license, data=cars)
> summary(OLS.full)

Call:
lm(formula = CarUsage ~ Age + Gender + Engineer + MBA + Work.Exp +
    Salary + Distance + license, data = cars)

Residuals:
     Min       1Q   Median       3Q      Max
-0.57133 -0.05632 -0.00245  0.05985  0.88198

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.590751   0.114665   5.152 4.02e-07 ***
Age          0.001968   0.004815   0.409 0.683058
Gender      -0.015574   0.017169  -0.907 0.364905
Engineer     0.007616   0.017467   0.436 0.663046
MBA         -0.017263   0.017404  -0.992 0.321836
Work.Exp    -0.006090   0.005973  -1.020 0.308473
Salary       0.021571   0.002363   9.128  < 2e-16 ***
Distance     0.013548   0.002364   5.730 1.95e-08 ***
license      0.072934   0.021843   3.339 0.000918 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1539 on 409 degrees of freedom
Multiple R-squared:  0.6981,	Adjusted R-squared:  0.6922
F-statistic: 118.2 on 8 and 409 DF,  p-value: < 2.2e-16

> vif(OLS.full)
      Age    Gender  Engineer       MBA  Work.Exp    Salary  Distance   licens
e
 7.045897  1.070569  1.013350  1.031029 14.575246  9.179929  1.346537  1.36471
9
> #removing work exp from the dataset as its showing multicollinearity
> #with age and salary using corrplot
>
> OLS.full1<-lm(CarUsage~Age+Gender+Engineer+MBA+Salary+Distance+
+                 license, data=cars)
> summary(OLS.full1)

Call:
lm(formula = CarUsage ~ Age + Gender + Engineer + MBA + Salary +
```

```
    Distance + license, data = cars)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.56936 -0.05939 -0.00414  0.06582  0.87279 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.667197   0.086767   7.690 1.11e-13 ***
Age         -0.001353   0.003548  -0.381 0.703130    
Gender      -0.014904   0.017158  -0.869 0.385541    
Engineer     0.007777   0.017468   0.445 0.656378    
MBA         -0.019112   0.017311  -1.104 0.270211    
Salary       0.019847   0.001651  12.019  < 2e-16 ***
Distance     0.013879   0.002342   5.926 6.59e-09 ***
license      0.075917   0.021647   3.507 0.000503 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1539 on 410 degrees of freedom
Multiple R-squared:  0.6973,	Adjusted R-squared:  0.6921 
F-statistic: 134.9 on 7 and 410 DF,  p-value: < 2.2e-16

> vif(OLS.full1)
     Age   Gender Engineer      MBA   Salary Distance  license 
3.823939 1.069003 1.013267 1.019841 4.481698 1.321141 1.340242 
> pred.reg<-predict(OLS.full1,newdata=cars, interval="predict")
> pred.reg
          fit       lwr      upr
1   0.9638654 0.6592429 1.268488
2   0.9077379 0.6033684 1.212107
3   1.0158336 0.7109390 1.320728
4   0.8418423 0.5370267 1.146658
5   0.9019923 0.5968036 1.207181
6   0.8960672 0.5899560 1.202178
7   0.9270779 0.6217402 1.232416
8   0.8365960 0.5318525 1.141339
9   0.8854903 0.5816425 1.189338
10  1.0592002 0.7534074 1.364993
11  0.9563799 0.6517275 1.261032
12  0.9071145 0.5999948 1.214234
13  0.9114280 0.6065439 1.216312
14  0.9855032 0.6814009 1.289606
15  1.0162752 0.7107443 1.321806
16  1.1303113 0.8249261 1.435696
17  0.9923096 0.6861621 1.298457
18  1.0305143 0.7252979 1.335731
19  1.2087562 0.9042253 1.513287
20  0.9117472 0.6059921 1.217502
21  1.0122126 0.7081997 1.316225
22  0.9966538 0.6925993 1.300708
23  1.5326721 1.2255142 1.839830
24  1.1206855 0.8155768 1.425794
25  0.9679038 0.6638346 1.271973
26  0.9999101 0.6945065 1.305314
27  1.0106594 0.7062964 1.315022
28  1.1077220 0.8017714 1.413673
29  0.9710552 0.6648864 1.277224
30  1.0276394 0.7236726 1.331606
31  0.9496042 0.6454975 1.253711
32  1.0000619 0.6941811 1.305943
33  1.0349814 0.7310093 1.338954
```

34	0.9332680	0.6280284	1.238508
35	1.2892125	0.9844393	1.593986
36	1.1453752	0.8381889	1.452562
37	0.9826519	0.6785598	1.286744
38	0.9133816	0.6094606	1.217303
39	1.0271415	0.7231795	1.331104
40	0.9292593	0.6253369	1.233182
41	1.0413594	0.7373334	1.345385
42	1.3981421	1.0921537	1.704130
43	1.1284029	0.8232122	1.433594
44	1.1706841	0.8655214	1.475847
45	0.9887230	0.6845184	1.292928
46	1.1887987	0.8846164	1.492981
47	1.1446365	0.8387464	1.450527
48	1.0339387	0.7254426	1.342435
49	0.9560369	0.6506475	1.261426
50	1.1276548	0.8215312	1.433778
51	1.5693554	1.2629132	1.875798
52	0.9136658	0.6086400	1.218692
53	0.9264430	0.6187185	1.234167
54	0.9663143	0.6606824	1.271946
55	0.9783157	0.6731102	1.283521
56	1.0543662	0.7493446	1.359388
57	0.9355537	0.6310870	1.240020
58	1.2256805	0.9205072	1.530854
59	0.9417023	0.6371612	1.246243
60	1.1436125	0.8368144	1.450411
61	0.9333632	0.6281966	1.238530
62	0.9957737	0.6911896	1.300358
63	0.9955328	0.6900503	1.301015
64	1.2957634	0.9902557	1.601271
65	1.0158016	0.7108389	1.320764
66	1.0294677	0.7216836	1.337252
67	1.0118672	0.7068863	1.316848
68	0.9542173	0.6485703	1.259864
69	1.1933309	0.8872393	1.499422
70	1.1913462	0.8852234	1.497469
71	1.2081471	0.9021342	1.514160
72	1.0215478	0.7163168	1.326779
73	0.9935070	0.6884718	1.298542
74	1.0364809	0.7293858	1.343576
75	1.0665668	0.7585059	1.374628
76	1.3076096	1.0014211	1.613798
77	1.1119990	0.8054299	1.418568
78	1.0360590	0.7304331	1.341685
79	1.0068246	0.7013803	1.312269
80	1.0369050	0.7296342	1.344176
81	1.2163725	0.9073605	1.525384
82	1.2090146	0.9019178	1.516111
83	1.2284717	0.9174331	1.539510
84	1.8180284	1.5102119	2.125845
85	1.6798341	1.3728025	1.986866
86	1.8748636	1.5663579	2.183369
87	1.7653570	1.4580681	2.072646
88	1.5538044	1.2479498	1.859659
89	1.1272054	0.8201182	1.434293
90	1.1898032	0.8801351	1.499471
91	1.7777786	1.4695219	2.086035
92	1.6725207	1.3638864	1.981155
93	1.6863230	1.3797194	1.992927
94	1.6303790	1.3228393	1.937919
95	1.4590594	1.1532813	1.764837

96	1.7140058	1.4065166	2.021495
97	1.7739477	1.4662633	2.081632
98	1.4955846	1.1898053	1.801364
99	1.6175176	1.3114164	1.923619
100	1.5733701	1.2665293	1.880211
101	1.6912978	1.3842395	1.998356
102	1.8324991	1.5256420	2.139356
103	1.8140050	1.5073444	2.120666
104	1.8722281	1.5649267	2.179530
105	1.6100221	1.3025315	1.917513
106	1.9308967	1.6213825	2.240411
107	1.9428029	1.6342914	2.251314
108	2.0195550	1.7104531	2.328657
109	1.1839028	0.8752863	1.492519
110	1.8348941	1.5277362	2.142052
111	1.7841260	1.4761568	2.092095
112	1.9645319	1.6549348	2.274129
113	1.7249017	1.4182255	2.031578
114	2.0952657	1.7847830	2.405748
115	1.8413480	1.5341442	2.148552
116	1.8625480	1.5551104	2.169986
117	1.9125080	1.6028294	2.222187
118	1.9663362	1.6568974	2.275775
119	0.9277324	0.6213956	1.234069
120	0.8394844	0.5335065	1.145462
121	0.9287859	0.6235460	1.234026
122	0.9314827	0.6247929	1.238173
123	0.9384316	0.6337029	1.243160
124	0.9966458	0.6898302	1.303461
125	0.9040719	0.5991186	1.209025
126	0.8350386	0.5300558	1.140021
127	0.9487439	0.6442497	1.253238
128	0.9266599	0.6215966	1.231723
129	1.0260490	0.7195297	1.332568
130	0.8563435	0.5519754	1.160712
131	1.0268402	0.7204034	1.333277
132	0.9858370	0.6799135	1.291760
133	0.8981955	0.5930364	1.203355
134	1.0085299	0.7013648	1.315695
135	0.9509002	0.6461447	1.255656
136	0.8674121	0.5629907	1.171833
137	0.8616194	0.5564443	1.166795
138	0.9209096	0.6168134	1.225006
139	0.9712309	0.6655228	1.276939
140	0.8596697	0.5547671	1.164572
141	0.8917149	0.5863620	1.197068
142	0.9455310	0.6404872	1.250575
143	0.9342689	0.6270163	1.241521
144	0.9966883	0.6913855	1.301991
145	0.8931029	0.5877752	1.198431
146	0.8530956	0.5482391	1.157952
147	1.0269251	0.7225173	1.331333
148	0.9604563	0.6558475	1.265065
149	0.9994642	0.6942210	1.304707
150	0.9489248	0.6448994	1.252950
151	0.9431583	0.6361284	1.250188
152	0.8851385	0.5804002	1.189877
153	0.8764959	0.5718558	1.181136
154	1.0488878	0.7429014	1.354874
155	0.9516429	0.6471456	1.256140
156	0.8839625	0.5780492	1.189876
157	0.9182797	0.6144038	1.222156

158	1.4936036	1.1844102	1.802797
159	0.8750989	0.5693232	1.180875
160	0.9928877	0.6883979	1.297377
161	0.9849963	0.6782219	1.291771
162	0.8586245	0.5529299	1.164319
163	1.0052103	0.7001310	1.310290
164	0.9648048	0.6582385	1.271371
165	1.0063803	0.7006016	1.312159
166	1.1541877	0.8487799	1.459595
167	0.9607846	0.6569698	1.264599
168	0.8844818	0.5799299	1.189034
169	0.8847341	0.5809607	1.188507
170	0.8669962	0.5629848	1.171008
171	0.9962025	0.6917603	1.300645
172	0.8647942	0.5600650	1.169523
173	1.0929752	0.7872231	1.398727
174	1.0183215	0.7135970	1.323046
175	0.8830703	0.5784412	1.187699
176	1.0011804	0.6971441	1.305217
177	0.9424637	0.6379297	1.246998
178	0.8691541	0.5645908	1.173717
179	1.0096951	0.7042542	1.315136
180	0.9868501	0.6830483	1.290652
181	0.9749700	0.6680522	1.281888
182	0.8844687	0.5764984	1.192439
183	0.8924300	0.5886609	1.196199
184	0.9341438	0.6304520	1.237836
185	0.9346128	0.6289342	1.240291
186	1.2079134	0.9005726	1.515254
187	0.9749386	0.6694039	1.280473
188	1.0003428	0.6954599	1.305226
189	0.8956399	0.5889593	1.202320
190	1.0216712	0.7168049	1.326538
191	1.0101048	0.7061732	1.314036
192	0.9633546	0.6587994	1.267910
193	1.0098525	0.7056311	1.314074
194	0.9994937	0.6944866	1.304501
195	0.9803817	0.6739687	1.286795
196	1.0139289	0.7076448	1.320213
197	0.8810705	0.5738305	1.188310
198	0.9567667	0.6530435	1.260490
199	0.9964657	0.6896685	1.303263
200	1.0102097	0.7048278	1.315592
201	0.9224300	0.6188044	1.226056
202	1.0300568	0.7248961	1.335217
203	0.8941739	0.5896477	1.198700
204	1.0135474	0.7088425	1.318252
205	1.0281070	0.7221764	1.334038
206	1.0292077	0.7244207	1.333995
207	0.9483975	0.6405998	1.256195
208	0.8993697	0.5956619	1.203078
209	1.0164128	0.7127468	1.320079
210	0.9002874	0.5955432	1.205032
211	1.0036925	0.6986135	1.308772
212	1.1094710	0.8037044	1.415238
213	0.9927578	0.6881531	1.297363
214	0.9894288	0.6815239	1.297334
215	0.9795978	0.6752590	1.283937
216	0.9861699	0.6825855	1.289754
217	0.9587635	0.6546481	1.262879
218	1.0867581	0.7819971	1.391519
219	0.9627679	0.6586968	1.266839

220	1.1819694	0.8775514	1.486387
221	0.9595197	0.6552366	1.263803
222	0.9496857	0.6440808	1.255291
223	1.0027143	0.6990701	1.306358
224	0.8938552	0.5877161	1.199994
225	0.9641558	0.6600946	1.268217
226	1.0628783	0.7574322	1.368324
227	0.9539424	0.6501268	1.257758
228	1.1161421	0.8100638	1.422220
229	1.0368687	0.7327173	1.341020
230	0.9484164	0.6432113	1.253622
231	0.8766771	0.5722891	1.181065
232	0.9660470	0.6615568	1.270537
233	0.9752493	0.6708115	1.279687
234	1.2271591	0.9213267	1.532991
235	0.8536169	0.5491020	1.158132
236	1.3957149	1.0882997	1.703130
237	1.2847854	0.9787610	1.590810
238	1.0062813	0.7026750	1.309888
239	0.9858026	0.6822843	1.289321
240	0.8852131	0.5802152	1.190211
241	0.9480932	0.6444629	1.251723
242	1.0200985	0.7132833	1.326914
243	1.0167808	0.7117911	1.321770
244	0.9124930	0.6089305	1.216055
245	0.9273971	0.6233102	1.231484
246	0.9291638	0.6247547	1.233573
247	1.0036091	0.6987741	1.308444
248	0.9356914	0.6314245	1.239958
249	0.9323750	0.6288219	1.235928
250	0.9774052	0.6731404	1.281670
251	0.9833737	0.6768009	1.289947
252	1.0150113	0.7114356	1.318587
253	0.8762748	0.5719010	1.180649
254	0.9992582	0.6957512	1.302765
255	0.9317782	0.6282211	1.235335
256	0.9536100	0.6501130	1.257107
257	1.0165375	0.7122254	1.320850
258	1.0289304	0.7233653	1.334495
259	0.9166568	0.6131019	1.220212
260	1.0877503	0.7823973	1.393103
261	1.0265330	0.7200099	1.333056
262	1.0992232	0.7931965	1.405250
263	1.0143355	0.7092043	1.319467
264	1.0143355	0.7092043	1.319467
265	1.0512554	0.7461256	1.356385
266	0.9221524	0.6178970	1.226408
267	0.9450535	0.6401833	1.249924
268	0.9023265	0.5986900	1.205963
269	0.9355068	0.6310983	1.239915
270	0.9856979	0.6814951	1.289901
271	1.0062487	0.7000748	1.312422
272	0.9964146	0.6910480	1.301781
273	0.8986569	0.5932536	1.204060
274	1.0398484	0.7361499	1.343547
275	1.0227073	0.7192524	1.326162
276	1.0281016	0.7234129	1.332790
277	1.3238965	1.0179642	1.629829
278	0.9255812	0.6220301	1.229132
279	1.0940720	0.7877745	1.400369
280	0.9051024	0.6014620	1.208743
281	1.0391182	0.7332310	1.345005

```
282 1.3672204 1.0601494 1.674291
283 1.0037896 0.6996015 1.307978
284 1.0342502 0.7295750 1.338925
285 1.0255319 0.7206610 1.330403
286 1.1225164 0.8180000 1.427033
287 0.8961083 0.5917515 1.200465
288 1.0437949 0.7397307 1.347859
289 1.0549819 0.7503475 1.359616
290 1.0594546 0.7548563 1.364053
291 1.0223536 0.7174387 1.327269
292 1.0266888 0.7227118 1.330666
293 1.1410901 0.8350234 1.447157
294 1.0238893 0.7194064 1.328372
295 1.0666701 0.7627622 1.370578
296 0.8968995 0.5925317 1.201267
297 0.9736030 0.6700749 1.277131
298 1.0097209 0.7046579 1.314784
299 1.1137693 0.8072008 1.420338
300 1.4253536 1.1183602 1.732347
301 1.0647553 0.7597557 1.369755
302 1.0349972 0.7284202 1.341574
303 0.9413542 0.6369990 1.245709
304 1.0092642 0.7034046 1.315124
305 1.0350672 0.7274286 1.342706
306 1.0063459 0.7019754 1.310716
307 1.0571004 0.7534177 1.360783
308 1.0004035 0.6961840 1.304623
309 1.0509290 0.7468423 1.355016
310 1.0565631 0.7501647 1.362961
311 0.9791596 0.6724104 1.285909
312 1.0366916 0.7324566 1.340927
313 1.0360599 0.7322315 1.339888
314 1.2449948 0.9390540 1.550936
315 1.0203067 0.7168127 1.323801
316 1.0878148 0.7818158 1.393814
317 1.1666813 0.8613975 1.471965
318 1.0250323 0.7215385 1.328526
319 1.0823190 0.7768747 1.387763
320 1.0756611 0.7716051 1.379717
321 0.9854765 0.6812539 1.289699
322 1.0769596 0.7727052 1.381214
323 0.9982926 0.6940783 1.302507
324 1.0453658 0.7400476 1.350684
325 0.9579046 0.6522458 1.263563
326 1.0242532 0.7202254 1.328281
327 1.1572579 0.8518575 1.462658
328 1.0942601 0.7897375 1.398783
329 1.0252405 0.7208843 1.329597
330 1.0648663 0.7607615 1.368971
331 1.0148992 0.7093002 1.320498
332 1.0319858 0.7276334 1.336338
333 1.3581673 1.0515710 1.664764
 [ reached getOption("max.print") -- omitted 85 rows ]
> cars1=cars[,c(1:4,6:10)]
> str(cars1)
'data.frame': 418 obs. of  9 variables:
 $ Age      : num  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender   : num  2 2 1 2 1 2 2 2 2 2 ...
 $ Engineer : num  1 1 1 0 0 0 1 0 1 1 ...
 $ MBA      : num  0 0 0 0 0 0 1 0 0 0 ...
 $ 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  : num  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 ...
 $ CarUsage : num  1 1 1 1 1 1 1 1 1 1 ...

>
> #model building and data split
>
> str(cars1)
'data.frame':	418 obs. of  9 variables:
 $ Age      : num  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender   : num  2 2 1 2 1 2 2 2 2 2 ...
 $ Engineer : num  1 1 1 0 0 0 1 0 1 1 ...
 $ MBA      : num  0 0 0 0 0 0 1 0 0 0 ...
 $ 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  : num  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 ...
 $ CarUsage : num  1 1 1 1 1 1 1 1 1 1 ...
> #convert to a data frame
> cars2<-as.data.frame(cars1)
> cars2$CarUsage=cars2$CarUsage -1
>
> cars2$CarUsage<-as.factor(cars2$CarUsage)
> cars2<-cars2[,c(9,1,2,3,4,5,6,7)]
> str(cars2)
'data.frame':	418 obs. of  8 variables:
 $ CarUsage: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Age     : num  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender  : num  2 2 1 2 1 2 2 2 2 2 ...
 $ Engineer: num  1 1 1 0 0 0 1 0 1 1 ...
 $ MBA     : num  0 0 0 0 0 0 1 0 0 0 ...
 $ 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 : num  0 0 0 0 0 0 0 0 0 1 ...
>
> set.seed(123)
> #splitting the data based on carusage
>
> carindex<-createDataPartition(cars2$CarUsage, p=0.7,list = FALSE,t
imes = 1)
>
> carsdatatrain<-cars2[carindex,]
> carsdatatest<-cars2[-carindex,]
>
> prop.table(table(carsdatatrain$CarUsage))

         0          1 
0.91496599 0.08503401 
> prop.table(table(carsdatatest$CarUsage))

         0          1 
0.91935484 0.08064516 
>
> View(carsdatatrain)
> carsdatatrain<-carsdatatrain[,c(1:8)]
> carsdatatest<-carsdatatest[,c(1:8)]
> ## The train and test data have almost same percentage of cars usa
ge as the base data
>
>
```

```
> #we can check the ratios of the minority class to majority class
> table(carsdatatrain$CarUsage)

  0   1 
269  25 
> table(carsdatatest$CarUsage)

  0   1 
114  10 
> #build model
> #logistic regression
> cars_logistic <- glm(CarUsage~., data=carsdatatrain, family=binomi
al(link="logit"))
> carsdatatest$log.pred<-predict(cars_logistic, carsdatatest[1:8], t
ype="response")
> table(carsdatatest$CarUsage,carsdatatest$log.pred>0.5)

    FALSE TRUE
  0   114    0
  1     1    9
> ###predict the response of the model using the train data
> 
> predTrain=predict(cars_logistic,newdata=carsdatatrain,type='respon
se')
> 
> #plot the ROC curve for calculating AUC
> library(ROCR)
> ROCRpred=prediction(predTrain,carsdatatrain$CarUsage)
> as.numeric(performance(ROCRpred,'auc')@y.values)
[1] 0.9986617
> perf_train=performance(ROCRpred,'tpr','fpr')
> plot(perf_train,col='black',lty=2,lwd=2)
> plot(perf_train,lwd=3,colorize=TRUE)
> ks_train <- max(perf_train@y.values[[1]]- perf_train@x.values[[1]]
)
> plot(perf_train,main=paste0('KS=',round(ks_train*100,1),'%'))
> lines(x = c(0,1),y=c(0,1))
> ###predict the response of the model using the test data
> 
> predTest=predict(cars_logistic,newdata=carsdatatest,type='response
')
> #build confusion matrix; >0.5=true else false
> conf_mat=table(carsdatatest$CarUsage,predTest>0.5)
> conf_mat

    FALSE TRUE
  0   114    0
  1     1    9
> #plot the ROC curve for calculating AUC
> library(ROCR)
> ROCRpred=prediction(predTest,carsdatatest$CarUsage)
> as.numeric(performance(ROCRpred,'auc')@y.values)
[1] 0.9929825
> perf_test=performance(ROCRpred,'tpr','fpr')
> plot(perf_test,col='black',lty=2,lwd=2)
> plot(perf_test,lwd=3,colorize=TRUE)
> ks_test <- max(perf_test@y.values[[1]]- perf_test@x.values[[1]])
> plot(perf_test,main=paste0('KS=',round(ks_test*100,1),'%'))
> lines(x = c(0,1),y=c(0,1))
> #knn
> str(cars2)
'data.frame': 418 obs. of  8 variables:
```

```
 $ CarUsage: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Age     : num  28 24 27 25 25 21 23 23 24 28 ...
 $ Gender  : num  2 2 1 2 1 2 2 2 2 2 ...
 $ Engineer: num  1 1 1 0 0 0 1 0 1 1 ...
 $ MBA     : num  0 0 0 0 0 0 1 0 0 0 ...
 $ 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 : num  0 0 0 0 0 0 0 0 0 1 ...
> library(class)
> #convert variables to num in order to use knn
> tcnorm<-scale(cars2[,-1])
> tcnorm<-cbind(cars2[,1],tcnorm)
> colnames(tcnorm)[1]<-'CarUsage'
> View(tcnorm)
> #convert to a data frame
>
> df_tcnorm<-as.data.frame(tcnorm)
> df_tcnorm$CarUsage<-as.factor(df_tcnorm$CarUsage)
>
> #check number values
> table(df_tcnorm$CarUsage)

  1   2 
383  35 
>
> #partition the data
> library(caTools)
> #train using knn
>
> sqrt(nrow(carsdatatrain))
[1] 17.14643
> View(carsdatatrain)
> knn_fit<- knn(carsdatatrain[,2:7], carsdatatest[,2:7],
+                cl= carsdatatrain[,1],k = 17,prob=TRUE)
> #check confusion matrix
> table.knn=table(carsdatatest[,1],knn_fit)
> table.knn
   knn_fit
      0   1
  0 114   0
  1   2   8
> #check accuracy
> sum(diag(table.knn)/sum(table.knn))
[1] 0.983871
> #ch loss
> loss.knn<-table.knn[2,1]/(table.knn[2,1]+table.knn[1,1])
> loss.knn
[1] 0.01724138
> opp.loss.knn<-table.knn[1,2]/(table.knn[1,2]+table.knn[2,2])
> opp.loss.knn
[1] 0
> tot.loss.knn<-0.95*loss.knn+0.05*opp.loss.knn
> tot.loss.knn
[1] 0.01637931
```

>##Naive Bayes

```
> nb_gd<-naiveBayes(x=carsdatatrain[,2:8], y=as.factor(carsdatatrain
[,1]))
> pred_nb<-predict(nb_gd,newdata = carsdatatest[,2:8])
> table(carsdatatest[,1],pred_nb)
   pred_nb
```

```
     0   1
  0 111   3
  1   1   9

> library(gbm)          # basic implementation using AdaBoost
> library(xgboost)      # a faster implementation of a gbm
> library(caret)  # an aggregator package for performing many machin
e learning models
>
> ## Bagging
> library(ipred)
> library(rpart)
> cars.bagging <- bagging(CarUsage ~.,
+                             data=carsdatatrain,
+                             control=rpart.control(maxdepth=6, minspl
it=20))
> carsdatatest$pred.class <- predict(cars.bagging, carsdatatest)
> str(carsdatatest)
'data.frame':	124 obs. of  10 variables:
 $ CarUsage  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Age       : num  24 25 25 21 23 23 28 26 21 22 ...
 $ Gender    : num  2 2 1 2 2 2 2 2 2 1 ...
 $ Engineer  : num  1 0 0 0 1 0 1 0 0 1 ...
 $ MBA       : num  0 0 0 0 1 0 0 0 1 0 ...
 $ Salary    : num  10.6 7.6 9.6 9.5 11.7 6.5 13.7 12.6 10.6 8.5 ...
 $ Distance  : num  6.1 6.3 6.7 7.1 7.2 7.3 7.5 7.5 7.7 8.1 ...
 $ license   : num  0 0 0 0 0 0 1 0 0 0 ...
 $ log.pred  : num  4.62e-08 1.02e-07 1.41e-07 4.02e-08 2.00e-08 ...
 $ pred.class: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> ###since pred class is in factor hence converting it to numeric to
determine confusion matrix
> carsdatatest$pred.class=as.numeric(as.character(carsdatatest$pred.
clas))
> carsdatatest$pred.class<- ifelse(carsdatatest$pred.class>0.5,1,0)
> confusionMatrix(data=factor(carsdatatest$pred.class),
+                 reference=factor(carsdatatest$CarUsage),
+                 positive='1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 114   2
         1   0   8

               Accuracy : 0.9839
                 95% CI : (0.943, 0.998)
    No Information Rate : 0.9194
    P-Value [Acc > NIR] : 0.002091

                  Kappa : 0.8803

 Mcnemar's Test P-Value : 0.479500

            Sensitivity : 0.80000
            Specificity : 1.00000
         Pos Pred Value : 1.00000
         Neg Pred Value : 0.98276
             Prevalence : 0.08065
         Detection Rate : 0.06452
   Detection Prevalence : 0.06452
      Balanced Accuracy : 0.90000
```

```
       'Positive' Class : 1

> ###
> table(carsdatatest$CarUsage,carsdatatest$pred.class>0.5)
   
    FALSE TRUE
  0   114    0
  1     2    8
> ##trying with a different value for maxdepth and minsplit
> cars.bagging <- bagging(CarUsage ~.,
+                         data=carsdatatrain,
+                         control=rpart.control(maxdepth=5, minsplit
=15))
> carsdatatest$pred.class <- predict(cars.bagging, carsdatatest)
> str(carsdatatest)
'data.frame':	124 obs. of  10 variables:
 $ CarUsage  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Age       : num  24 25 25 21 23 23 28 26 21 22 ...
 $ Gender    : num  2 2 1 2 2 2 2 2 2 1 ...
 $ Engineer  : num  1 0 0 0 1 0 1 0 0 1 ...
 $ MBA       : num  0 0 0 0 1 0 0 0 1 0 ...
 $ Salary    : num  10.6 7.6 9.6 9.5 11.7 6.5 13.7 12.6 10.6 8.5 ...
 $ Distance  : num  6.1 6.3 6.7 7.1 7.2 7.3 7.5 7.5 7.7 8.1 ...
 $ license   : num  0 0 0 0 0 0 1 0 0 0 ...
 $ log.pred  : num  4.62e-08 1.02e-07 1.41e-07 4.02e-08 2.00e-08 ...
 $ pred.class: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> ###since pred class is in factor hence converting it to numeric to
determine confusion matrix
> carsdatatest$pred.class=as.numeric(as.character(carsdatatest$pred.
clas))
> carsdatatest$pred.class<- ifelse(carsdatatest$pred.class>0.5,1,0)
> 
> 
> confusionMatrix(data=factor(carsdatatest$pred.class),
+                 reference=factor(carsdatatest$CarUsage),
+                 positive='1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 114   3
         1   0   7
                                          
               Accuracy : 0.9758          
                 95% CI : (0.9309, 0.995) 
    No Information Rate : 0.9194          
    P-Value [Acc > NIR] : 0.008296        
                                          
                  Kappa : 0.811           
                                          
 Mcnemar's Test P-Value : 0.248213        
                                          
            Sensitivity : 0.70000         
            Specificity : 1.00000         
         Pos Pred Value : 1.00000         
         Neg Pred Value : 0.97436         
             Prevalence : 0.08065         
         Detection Rate : 0.05645         
   Detection Prevalence : 0.05645         
      Balanced Accuracy : 0.85000         
                                          
       'Positive' Class : 1
```

```
> ###
> table(carsdatatest$CarUsage,carsdatatest$pred.class>0.5)

    FALSE TRUE
  0   114    0
  1     3    7
> ##bagging with max depth=6;minsplit=20 is better
> #Boosting
> boostcontrol <- trainControl(number=10)
>
> xgbGrid <- expand.grid(
+   eta = 0.3,
+   max_depth = 1,
+   nrounds = 50,
+   gamma = 0,
+   colsample_bytree = 0.6,
+   min_child_weight = 1, subsample = 1
+ )
> carsxgb <-  train(CarUsage ~ .,carsdatatrain,trControl = boostcontrol,tuneGri
d = xgbGrid,metric = "Accuracy",method = "xgbTree")
> carsxgb$finalModel
##### xgb.Booster
raw: 13.1 Kb
call:
  xgboost::xgb.train(params = list(eta = param$eta, max_depth = param$max_depth
,
    gamma = param$gamma, colsample_bytree = param$colsample_bytree,
    min_child_weight = param$min_child_weight, subsample = param$subsample),
    data = x, nrounds = param$nrounds, objective = "binary:logistic")
params (as set within xgb.train):
  eta = "0.3", max_depth = "1", gamma = "0", colsample_bytree = "0.6", min_chil
d_weight = "1", subsample = "1", objective = "binary:logistic", silent = "1"
xgb.attributes:
  niter
callbacks:
  cb.print.evaluation(period = print_every_n)
# of features: 7
niter: 50
nfeatures : 7
xNames : Age Gender Engineer MBA Salary Distance license
problemType : Classification
tuneValue :
         nrounds max_depth eta gamma colsample_bytree min_child_weight subsampl
e
1      50         1 0.3     0              0.6                1         1
obsLevels : 0 1
param :
        list()
> ##predict using test dataset
> predictions_xgb=predict(carsxgb,carsdatatest)
> confusionMatrix(predictions_xgb,carsdatatest$CarUsage)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 114   2
         1   0   8

               Accuracy : 0.9839
                 95% CI : (0.943, 0.998)
    No Information Rate : 0.9194
```

```
    P-Value [Acc > NIR] : 0.002091

                  Kappa : 0.8803

 Mcnemar's Test P-Value : 0.479500

            Sensitivity : 1.0000
            Specificity : 0.8000
         Pos Pred Value : 0.9828
         Neg Pred Value : 1.0000
             Prevalence : 0.9194
         Detection Rate : 0.9194
   Detection Prevalence : 0.9355
      Balanced Accuracy : 0.9000

       'Positive' Class : 0

> str(carsdatatrain)
'data.frame':	294 obs. of  8 variables:
 $ CarUsage: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Age     : num  28 27 24 28 23 29 29 25 34 28 ...
 $ Gender  : num  2 1 2 1 2 1 2 1 2 2 ...
 $ Engineer: num  1 1 1 0 1 0 1 1 1 1 ...
 $ MBA     : num  0 0 0 0 0 0 0 0 1 0 ...
 $ Salary  : num  14.4 15.5 8.5 19.7 8.8 14.6 23.8 11.6 36.9 14.7 ...
 $ Distance: num  5.1 6.1 7.5 9 9.2 9.2 9.4 10.1 10.4 10.5 ...
 $ license : num  0 0 0 0 1 0 0 0 1 1 ...
> #####XGB BOOST###
> # XGBoost works with matrices that contain all numeric variables
>
> View(carsdatatrain)
> str(carsdatatrain)
'data.frame':	294 obs. of  8 variables:
 $ CarUsage: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Age     : num  28 27 24 28 23 29 29 25 34 28 ...
 $ Gender  : num  2 1 2 1 2 1 2 1 2 2 ...
 $ Engineer: num  1 1 1 0 1 0 1 1 1 1 ...
 $ MBA     : num  0 0 0 0 0 0 0 0 1 0 ...
 $ Salary  : num  14.4 15.5 8.5 19.7 8.8 14.6 23.8 11.6 36.9 14.7 ...
 $ Distance: num  5.1 6.1 7.5 9 9.2 9.2 9.4 10.1 10.4 10.5 ...
 $ license : num  0 0 0 0 1 0 0 0 1 1 ...
> cars_features_train<-as.matrix(carsdatatrain[,2:8])
> cars_label_train<-as.matrix(carsdatatrain[,1])
> cars_features_test<-as.matrix(carsdatatest[,2:8])
>
> xgb.fit <- xgboost(
+   data = cars_features_train,
+   label = cars_label_train,
+   eta = 0.001,
+   max_depth = 3,
+   min_child_weight = 3,
+   nrounds = 10000,
+   nfold = 5,
+   objective = "binary:logistic",  # for regression models
+   verbose = 0,                    # silent,
+   early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tree
s
+ )
> carsdatatest$xgb.pred.class <- predict(xgb.fit, cars_features_test)
>
> table(carsdatatest$CarUsage,carsdatatest$xgb.pred.class>0.5)
```

```
    FALSE TRUE
  0   112    2
  1     2    8
>
> #or simply the total correct of the minority class
> sum(carsdatatest$CarUsage==1 & carsdatatest$xgb.pred.class>=0.5)
[1] 8
> #adjusting lr,md and nr
>
> #adjusting nr
>
> tp_xgb<-vector()
> lr <- c(0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1)
> md<-c(1,3,5,7,9,15)
> nr<-c(2, 50, 100, 1000, 10000)
> for (i in nr) {
+
+   xgb.fit <- xgboost(
+     data = cars_features_train,
+     label = cars_label_train,
+     eta = 0.7,
+     max_depth = 5,
+     nrounds = i,
+     nfold = 5,
+     objective = "binary:logistic",  # for regression models
+     verbose = 0,               # silent,
+     early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tr
ees
+   )
+
+   carsdatatest$xgb.pred.class <- predict(xgb.fit, cars_features_test)
+
+   tp_xgb<-cbind(tp_xgb,sum(carsdatatest$CarUsage==1 & carsdatatest$xgb.pred.c
lass>=0.5))
+
+ }
>
> tp_xgb
     [,1] [,2] [,3] [,4] [,5]
[1,]    8    8    8    8    8
> #Stopping. Best iteration after 13 rounds at nr=0.003401
>
> #adjusting lr or eta
>
> tp_xgb<-vector()
> lr <- c(0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1)
> md<-c(1,3,5,7,9,15)
> nr<-c(2, 50, 100, 1000, 10000)
> for (i in lr) {
+
+   xgb.fit <- xgboost(
+     data = cars_features_train,
+     label = cars_label_train,
+     eta = i,
+     max_depth = 5,
+     nrounds = 50,
+     nfold = 5,
+     objective = "binary:logistic",  # for regression models
+     verbose = 0,               # silent,
+     early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tr
ees
+   )
```

```
+
+   carsdatatest$xgb.pred.class <- predict(xgb.fit, cars_features_test)
+
+   tp_xgb<-cbind(tp_xgb,sum(carsdatatest$CarUsage==1 & carsdatatest$xgb.pred.c
lass>=0.5))
+
+ }
>
> tp_xgb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    8    8    8    8    8    8    8
> #adjusting md
>
> tp_xgb<-vector()
> lr <- c(0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1)
> md<-c(1,3,5,7,9,15)
> nr<-c(2, 50, 100, 1000, 10000)
> for (i in md) {
+
+   xgb.fit <- xgboost(
+     data = cars_features_train,
+     label = cars_label_train,
+     eta = 0.7,
+     max_depth = i,
+     nrounds = 50,
+     nfold = 5,
+     objective = "binary:logistic",  # for regression models
+     verbose = 0,               # silent,
+     early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tr
ees
+   )
+
+   carsdatatest$xgb.pred.class <- predict(xgb.fit, cars_features_test)
+
+   tp_xgb<-cbind(tp_xgb,sum(carsdatatest$CarUsage==1 & carsdatatest$xgb.pred.c
lass>=0.5))
+
+ }
>
> tp_xgb
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    8    8    8    8    8    8
> #now we put them all into our best fit!
>
> xgb.fit <- xgboost(
+   data = cars_features_train,
+   label = cars_label_train,
+   eta = 0.7,
+   max_depth = 5,
+   nrounds = 50,
+   nfold = 5,
+   objective = "binary:logistic",  # for regression models
+   verbose = 0,               # silent,
+   early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tree
s
+ )
>
> carsdatatest$xgb.pred.class <- predict(xgb.fit, cars_features_test)
>
> sum(carsdatatest$CarUsage==1 & carsdatatest$xgb.pred.class>=0.5)
[1] 8
> table(carsdatatest$CarUsage,carsdatatest$xgb.pred.class>=0.5)
```

```
    FALSE TRUE
  0   114    0
  1     2    8
> library(DMwR)
> #working with SMOTE
> library(DMwR)
>
> set.seed(123)
> carindex<-createDataPartition(cars2$CarUsage, p=0.7,list = FALSE,times = 1)
> carsdatatrain<-cars2[carindex,]
> carsdatatest<-cars2[-carindex,]
> prop.table(table(carsdatatrain$CarUsage))

         0          1
0.91496599 0.08503401
> attach(carsdatatrain)
The following objects are masked from cars (pos = 3):

    Age, Distance, Engineer, Gender, license, MBA, Salary

The following objects are masked from carsdatatrain (pos = 4):

    Age, CarUsage, Distance, Engineer, Gender, license, MBA, Salary

The following objects are masked from cars (pos = 12):

    Age, Distance, Engineer, Gender, license, MBA, Salary


>##Trying with various combinations of perc.over and perc.under

> carsdataSMOTE<-SMOTE(CarUsage~., carsdatatrain, perc.over = 250,perc.under =
150)
> prop.table(table(carsdataSMOTE$CarUsage))

  0   1
0.5 0.5
> ###going with equal ratio in car usage
>
> #model building using xgboost
> #now put our SMOTE data into our best xgboost
>
> smote_features_train<-as.matrix(carsdataSMOTE[,2:8])
> smote_label_train<-as.matrix(carsdataSMOTE$CarUsage)
>
> smote.xgb.fit <- xgboost(
+   data = smote_features_train,
+   label = smote_label_train,
+   eta = 0.7,
+   max_depth = 5,
+   nrounds = 50,
+   nfold = 5,
+   objective = "binary:logistic",  # for regression models
+   verbose = 0,               # silent,
+   early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tree
s
+ )
> smote_features_test<-as.matrix(carsdatatest[,2:8])
> carsdatatest$smote.pred.class <- predict(smote.xgb.fit, smote_features_test)
>
> table(carsdatatest$CarUsage,carsdatatest$smote.pred.class>=0.5)
```

```
    FALSE TRUE
  0   111    3
  1     1    9
> ##we get all the 10 cars as true
> sum(carsdatatest$CarUsage==1 & carsdatatest$xgb.pred.class>=0.5)
[1] 0
> carsdataSMOTE<-SMOTE(CarUsage~., carsdatatrain, perc.over = 275,perc.under =
150)
> prop.table(table(carsdataSMOTE$CarUsage))

  0   1
0.5 0.5
> smote_features_train<-as.matrix(carsdataSMOTE[,2:8])
> smote_label_train<-as.matrix(carsdataSMOTE$CarUsage)
>
> smote.xgb.fit <- xgboost(
+   data = smote_features_train,
+   label = smote_label_train,
+   eta = 0.7,
+   max_depth = 5,
+   nrounds = 50,
+   nfold = 5,
+   objective = "binary:logistic",  # for regression models
+   verbose = 0,                # silent,
+   early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tree
s
+ )
>
> smote_features_test<-as.matrix(carsdatatest[,2:8])
> carsdatatest$smote.pred.class <- predict(smote.xgb.fit, smote_features_test)
>
> table(carsdatatest$CarUsage,carsdatatest$smote.pred.class>=0.5)

    FALSE TRUE
  0   114    0
  1     1    9
> carsdataSMOTE<-SMOTE(CarUsage~., carsdatatrain, perc.over = 250,perc.under =
170)
> prop.table(table(carsdataSMOTE$CarUsage))

      0       1
0.53125 0.46875
> carsdataSMOTE<-SMOTE(CarUsage~., carsdatatrain, perc.over = 250,perc.under =
200)
> prop.table(table(carsdataSMOTE$CarUsage))

        0         1
0.5714286 0.4285714
> carsdataSMOTE<-SMOTE(CarUsage~., carsdatatrain,  perc.over = 250,perc.under =
100)
> prop.table(table(carsdataSMOTE$CarUsage))

  0   1
0.4 0.6
> smote_features_train<-as.matrix(carsdataSMOTE[,2:8])
> smote_label_train<-as.matrix(carsdataSMOTE$CarUsage)
>
> smote.xgb.fit <- xgboost(
+   data = smote_features_train,
+   label = smote_label_train,
+   eta = 0.7,
```

```
+   max_depth = 5,
+   nrounds = 50,
+   nfold = 5,
+   objective = "binary:logistic",  # for regression models
+   verbose = 0,                    # silent,
+   early_stopping_rounds = 10 # stop if no improvement for 10 consecutive tree
s
+ )
>
> smote_features_test<-as.matrix(carsdatatest[,2:8])
> carsdatatest$smote.pred.class <- predict(smote.xgb.fit, smote_features_test)
>
> table(carsdatatest$CarUsage,carsdatatest$smote.pred.class>=0.5)

    FALSE TRUE
  0   114    0
  1     0   10


>
 > ##we get all the 10 cars as true
 > #Let us proceed with building the models
 > ## Model Building We will use the Logistic regression method a model
 > #on the SMOTE data to understand the factors influencing car usage.
 > outcomevar<-'CarUsage'
 > regressors<-c("Age","Salary","Distance","license","Engineer","MBA","Gender")
 > trainctrl<-trainControl(method = 'repeatedcv',number = 10,repeats = 3)
 > carsglm<-train(carsdataSMOTE[,regressors],carsdataSMOTE[,outcomevar],method
 = "glm", family = "binomial",trControl = trainctrl)
 There were 50 or more warnings (use warnings() to see the first 50)
 > summary(carsglm$finalModel)

 Call:
 NULL

 Deviance Residuals:
        Min          1Q      Median          3Q         Max
 -3.098e-05  -2.100e-08   2.100e-08   2.100e-08   4.387e-05

 Coefficients:
               Estimate Std. Error z value Pr(>|z|)
 (Intercept)   -333.100 245515.884  -0.001    0.999
 Age              5.363   8939.547   0.001    1.000
 Salary           1.583   4590.228   0.000    1.000
 Distance        10.040   6645.208   0.002    0.999
 license          6.538  83972.530   0.000    1.000
 Engineer       -40.649  64717.295  -0.001    0.999
 MBA            -23.668  39301.293  -0.001    1.000
 Gender          11.663  49718.430   0.000    1.000

 (Dispersion parameter for binomial family taken to be 1)

     Null deviance: 1.6825e+02  on 124  degrees of freedom
 Residual deviance: 6.5126e-09  on 117  degrees of freedom
 AIC: 16

 Number of Fisher Scoring iterations: 25
 > varImp(object = carsglm)
 glm variable importance

          Overall
 Distance  100.00
```

```
Engineer   38.40
MBA        36.59
Age        36.43
Salary     18.63
Gender     10.94
license     0.00

> plot(varImp(object = carsglm), main="Variable Importance for Logistic Regres
sion")
> #we see that distance and engineer are most significant.
> carusageprediction<-predict.train(object = carsglm,carsdatatest[,regressors]
,type = "raw")
> confusionMatrix(carusageprediction,carsdatatest[,outcomevar], positive='1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 111   1
         1   3   9

               Accuracy : 0.9677
                 95% CI : (0.9195, 0.9911)
    No Information Rate : 0.9194
    P-Value [Acc > NIR] : 0.02476

                  Kappa : 0.8006

 Mcnemar's Test P-Value : 0.61708

            Sensitivity : 0.90000
            Specificity : 0.97368
         Pos Pred Value : 0.75000
         Neg Pred Value : 0.99107
             Prevalence : 0.08065
         Detection Rate : 0.07258
   Detection Prevalence : 0.09677
      Balanced Accuracy : 0.93684

       'Positive' Class : 1

> ##RF MODEL
> rftrcontrol<-control <- trainControl(method="repeatedcv", number=10, repeats
=3)
> mtry<-sqrt(ncol(carsdatatrain))
> tunegridrf <- expand.grid(.mtry=mtry)
> carsrf<-train(CarUsage ~.,carsdatatrain,method = "rf", trControl=rftrcontrol
, tuneGrid = tunegridrf)
> carsrf$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.7%
Confusion matrix:
    0  1 class.error
0 268  1 0.003717472
1   4 21 0.160000000
> plot(varImp(object=carsrf), main = "Variable Importance for Random Forest")
> #OOB estimate of  error rate is  1.7% in training dataset. salary and age ar
```

```
e most significant
> #attempt on test data
> predictions_rf<-predict(carsrf,carsdatatest)
> confusionMatrix(predictions_rf,carsdatatest$CarUsage)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 114   2
         1   0   8

               Accuracy : 0.9839
                 95% CI : (0.943, 0.998)
    No Information Rate : 0.9194
    P-Value [Acc > NIR] : 0.002091

                  Kappa : 0.8803

 Mcnemar's Test P-Value : 0.479500

            Sensitivity : 1.0000
            Specificity : 0.8000
         Pos Pred Value : 0.9828
         Neg Pred Value : 1.0000
             Prevalence : 0.9194
         Detection Rate : 0.9194
   Detection Prevalence : 0.9355
      Balanced Accuracy : 0.9000

       'Positive' Class : 0

>
>
> #98.38% overall accuracy; 80% accuracy for predicting cars

>
```