Contents

[1. Approach 3](#_Toc15264333)

[2. Colour code 3](#_Toc15264334)

[3. Data attributes 3](#_Toc15264335)

[4. Exploratory Analysis, Data Cleaning & Pre-processing 4](#_Toc15264336)

[1. Check Data Structure 4](#_Toc15264337)

[2. Check Missing Values 5](#_Toc15264338)

[3. Convert Transport into binary 5](#_Toc15264339)

[4. Plot data to see the distribution 6](#_Toc15264340)

[5. Check for multi-colinearity 10](#_Toc15264341)

[6. Check for Outliers 11](#_Toc15264342)

[7. Remove the outliers capping it 95% 12](#_Toc15264343)

[8. Check if the target variable Transport is balanced 13](#_Toc15264344)

[1. From the below table is shows that Cars has a very low count 13](#_Toc15264345)

[2. Generate data using SMOTE 13](#_Toc15264346)

[3. Add the SMOTE data to the original data 13](#_Toc15264347)

[9. Split the data into 70% for training and 30% for test 14](#_Toc15264348)

[5. Modelling 14](#_Toc15264349)

[1. Logistic Regression Model 14](#_Toc15264350)

[1. Model Building 14](#_Toc15264351)

[2. Predict model using test data 14](#_Toc15264352)

[2. Naïve Bayes Classifier 15](#_Toc15264353)

[1. Model Building 15](#_Toc15264354)

[2. Predict test data using Naïve Bayes model 15](#_Toc15264355)

[3. Train KNN model using Caret 16](#_Toc15264356)

[1. Model Building 16](#_Toc15264357)

[2. Predict KNN model 17](#_Toc15264358)

[4. Model Tuning 17](#_Toc15264359)

[1. Applying Bagging (Bootstrap Aggregation) 17](#_Toc15264360)

[2. Applying Boosting 17](#_Toc15264361)

[**6.** Validation 18](#_Toc15264362)

[1. Confusion Matrix of Logistic Regression model 18](#_Toc15264363)

[2. Confusion Matrix of Naïve Bayes model 18](#_Toc15264364)

[3. Confusion Matrix of KNN model 19](#_Toc15264365)

[4. Confusion Matrix for Bagging 19](#_Toc15264366)

[5.Confusion Matrix for Boosting 20](#_Toc15264367)

[7. Model Comparison 21](#_Toc15264368)

[8. Conclusion 21](#_Toc15264369)

[9. Recommendation 21](#_Toc15264370)

Problem Statement – Cars Assessment

A Cars.csv consists of data of a number of Employees. There are multiple variables in this dataset, which give info about the employee like age, salary, work exp etc. We need to predict whether the employee will use Cars as a mode of transport.

# Approach

1. Load Data in R
2. Exploratory Analysis of the Data
3. Data Cleaning
4. Data pre-processing
5. Modelling
6. Validation
7. Model Comparison

# Colour code

R Command Output

# Data attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No. | Variable Name | Description | Type | Remark |
| 1 | Age | Age of employee | Continuous | Independent Variable |
| 2 | Gender | Gender of employee | Categorical | Independent Variable |
| 3 | Engineer | 1 if employee is Engineer and 0 if he is not | Categorical | Independent Variable |
| 4 | MBA | 1 if employee is MBA and 0 if he is not | Categorical | Independent Variable |
| 5 | Work Exp | Work Experience of Employee | Continuous | Independent Variable |
| 6 | Salary | Salary of Employee | Continuous | Independent Variable |
| 7 | Distance | Distance of employee’s home from office | Continuous | Independent Variable |
| 8 | License | 1 if employee has license and 0 if he is not | Categorical | Independent Variable |
| 9 | Transport | Type of transport employee uses (Public Transport / Two wheeler / Car) | Categorical | Dependent Variable |

# Exploratory Analysis, Data Cleaning & Pre-processing

## Check Data Structure

> str(Cars)

Below results show that there are 9 variables in the data out of which 2 are categorical variable with data class as character and the remaining are numeric with data class as integer.

Classes ‘tbl\_df’, ‘tbl’ and 'data.frame': 444 obs. of 9 variables:

$ Age : int 28 23 29 28 27 26 28 26 22 27 ...

$ Gender : chr "Male" "Female" "Male" "Female" ...

$ Engineer : int 0 1 1 1 1 1 1 1 1 1 ...

$ MBA : int 0 0 0 1 0 0 0 0 0 0 ...

$ Work Exp : int 4 4 7 5 4 4 5 3 1 4 ...

$ Salary : num 14.3 8.3 13.4 13.4 13.4 12.3 14.4 10.5 7.5 13.5 ...

$ Distance : num 3.2 3.3 4.1 4.5 4.6 4.8 5.1 5.1 5.1 5.2 ...

$ license : int 0 0 0 0 0 1 0 0 0 0 ...

$ Transport: chr "Public Transport" "Public Transport" "Public Transport" "Public Transport" ...

- attr(\*, "spec")=List of 2

..$ cols :List of 9

.. ..$ Age : list()

.. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"

.. ..$ Gender : list()

.. .. ..- attr(\*, "class")= chr "collector\_character" "collector"

.. ..$ Engineer : list()

.. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"

.. ..$ MBA : list()

.. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"

.. ..$ Work Exp : list()

.. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"

.. ..$ Salary : list()

.. .. ..- attr(\*, "class")= chr "collector\_double" "collector"

.. ..$ Distance : list()

.. .. ..- attr(\*, "class")= chr "collector\_double" "collector"

.. ..$ license : list()

.. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"

.. ..$ Transport: list()

.. .. ..- attr(\*, "class")= chr "collector\_character" "collector"

..$ default: list()

.. ..- attr(\*, "class")= chr "collector\_guess" "collector"

..- attr(\*, "class")= chr "col\_spec"

From the below looking at the median and the max numbers it gives an idea that Age, Salary and Distance are skewed. We will plot the data to see further.

> summary(Cars)

Age Gender Engineer MBA Work Exp

Min. :18.00 Length:444 Min. :0.0000 Min. :0.0000 Min. : 0.0

1st Qu.:25.00 Class :character 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.: 3.0

Median :27.00 Mode :character Median :1.0000 Median :0.0000 Median : 5.0

Mean :27.75 Mean :0.7545 Mean :0.2528 Mean : 6.3

3rd Qu.:30.00 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.: 8.0

Max. :43.00 Max. :1.0000 Max. :1.0000 Max. :24.0

NA's :1

Salary Distance license Transport

Min. : 6.50 Min. : 3.20 Min. :0.0000 Length:444

1st Qu.: 9.80 1st Qu.: 8.80 1st Qu.:0.0000 Class :character

Median :13.60 Median :11.00 Median :0.0000 Mode :character

Mean :16.24 Mean :11.32 Mean :0.2342

3rd Qu.:15.72 3rd Qu.:13.43 3rd Qu.:0.0000

Max. :57.00 Max. :23.40 Max. :1.0000

## Check Missing Values

> sum(is.na(Cars))

[1] 1

We found out only 1 missing value and hence we remove it.

> Cars<- na.omit(Cars)

## Convert Transport into binary

Assign variable having value Car = 1, Two wheeler & Public transport=0 as we want to predict whether the employee will use Cars as a mode of transport or not.

> Cars[Cars=="2Wheeler"]<- 0

> Cars[Cars=="Public Transport"]<- 0

> Cars[Cars=="Car"]<- 1

> Cars$Transport <- as.numeric(Cars$Transport)

Convert Gender into binary (Female = 1 & Male = 0)

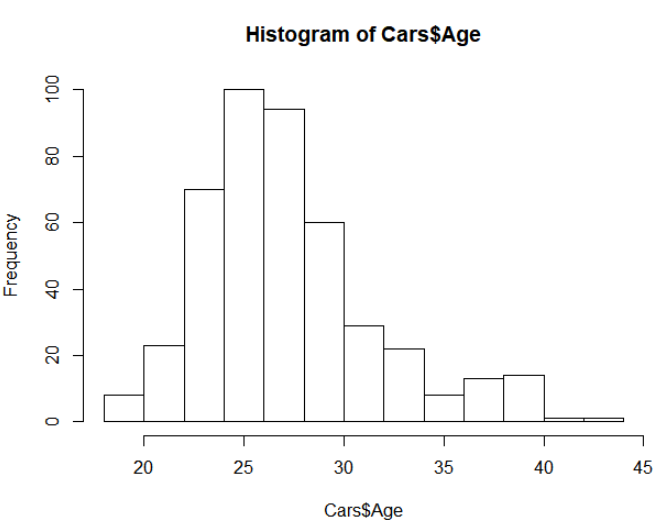
> Cars[Cars=="Female"]<- 1

> Cars[Cars=="Male"]<- 0

> Cars$Gender <- as.numeric(Cars$Gender)

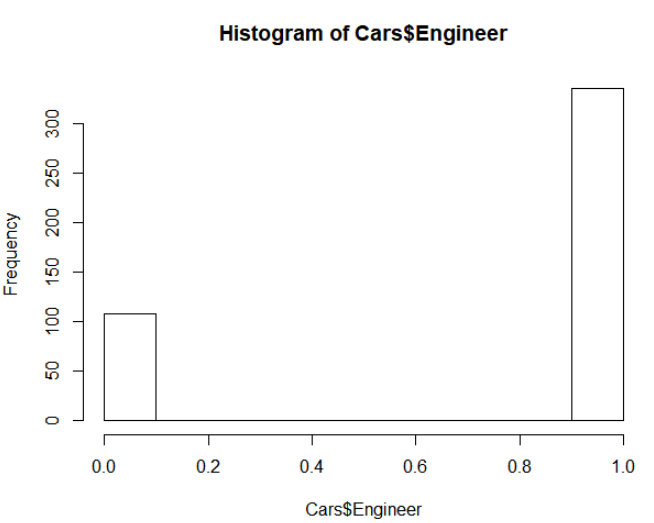
## Plot data to see the distribution

> hist(Cars$Age)



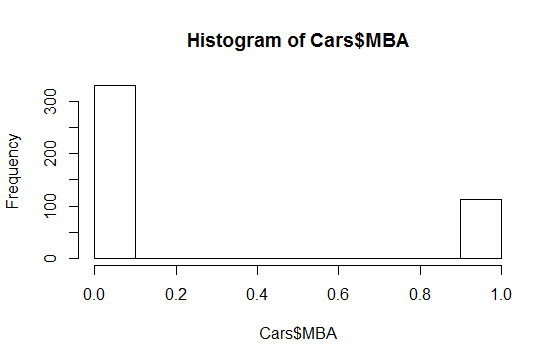
Age is right- skewed tailing at the end, giving hint of outliers.

> hist(Cars$Engineer)



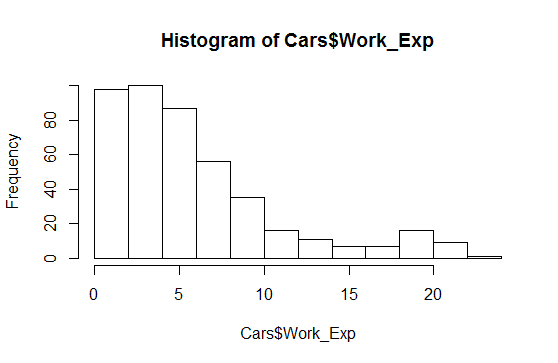
Engineer is a binary variable

> hist(Cars$MBA)



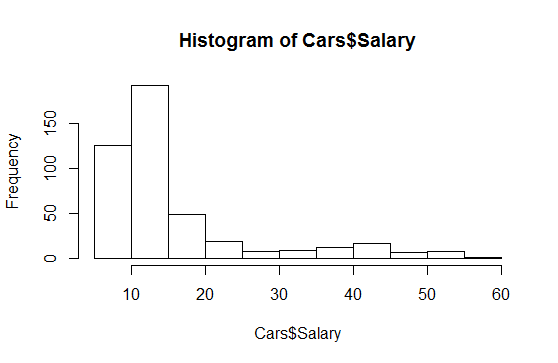
MBA is also a binary variable

> hist(Cars$Work\_Exp)



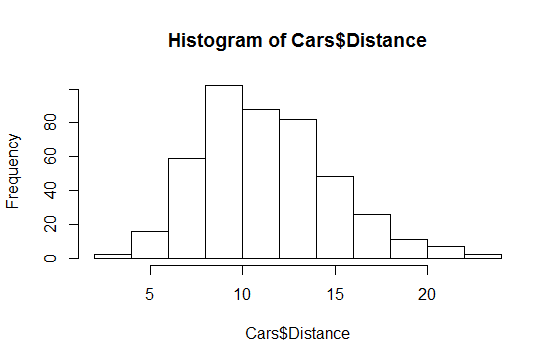
Work experience is right-skewed and tailed as the right

> hist(Cars$Salary)



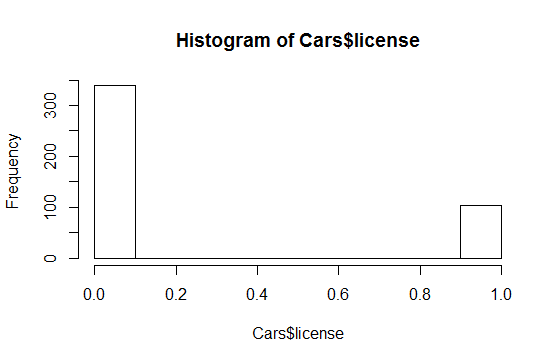
Salary is very unevenly distributed

> hist(Cars$Distance)



Distance is also right-skewed, but the distribution looks good.

> hist(Cars$license)

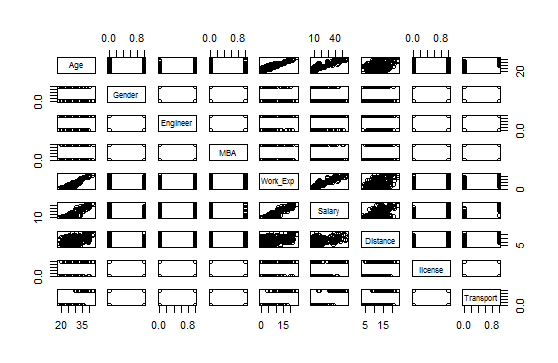


License is also a binary variable

## Check for multi-colinearity

Below image shows that there is relationship between the variables.

> plot(Cars)



Let’s look at the VIF values.

We apply vifcor function to exclude highly correlated variables from the set.

> vifcor(Cars[-9])

1 variables from the 8 input variables have collinearity problem:

Work\_Exp

After excluding the collinear variables, the linear correlation coefficients ranges between:

min correlation ( Salary ~ MBA ): -0.007592236

max correlation ( Salary ~ Age ): 0.8607652

---------- VIFs of the remained variables --------

Variables VIF

1 Age 3.896910

2 Gender 1.070855

3 Engineer 1.014883

4 MBA 1.019907

5 Salary 4.457554

6 Distance 1.260307

7 license 1.430460

So, we need to drop Work\_Exp as the function states it has collinearity problem.

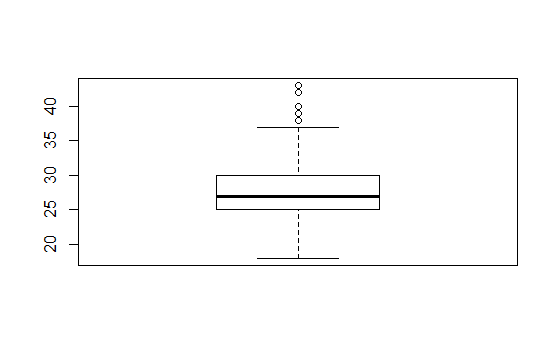
Removing Work\_Exp from data

Cars <- Cars[-5]

## Check for Outliers

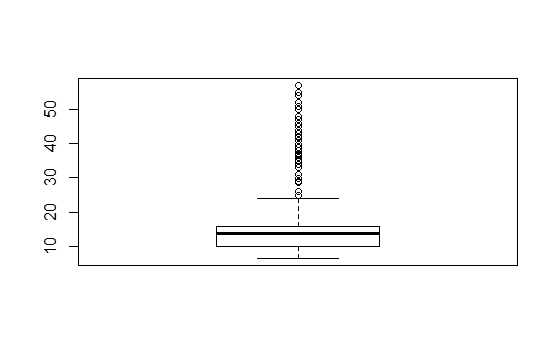
We need to check the outliers in data as data outliers can spoil and mislead the training process resulting in longer training times, less accurate models and ultimately poorer results.

> boxplot(Cars$Age)



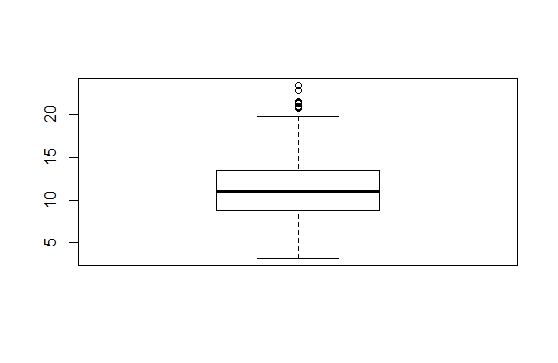
Age has outliers

> boxplot(Cars$Salary)



Salary has outliers

> boxplot(Cars$Distance)



Distance has outliers

## Remove the outliers capping it 95%

Basically there are three way to deal with outliers

1. Univariate method
2. Multivariate method
3. Minkowski erro

Here we will be applying univariate method which looks for data points with extreme values on one variable. We have kept cap of 95% to remove the outliers.

a) Removing outliers with respect to variable Age

Finding value for 95% capping for Age, which comes out to be 38

> quantile(Cars$Age, c(0.95))

95%

38

Removing the data points which are above cut-off point

> Cars$Age[which(Cars$Age>38)]<- 38

b) Removing outliers with respect to variable Salary

Finding value for 95% capping for Salary, which comes out to be 43

> quantile(Cars$Salary,c(0.95))

95%

43

Removing the data points which are above cut-off point

> Cars$Salary[which(Cars$Salary>43)] <- 43

c) Removing outliers with respect to variable Distance

Finding value for 95% capping for Distance, which comes out to be 17.89

> quantile(Cars$Distance,c(0.95))

95%

17.89

Removing the data points which are above cut-off point

> Cars$Distance[which(Cars$Distance> 17.89)] <- 17.89

## Check if the target variable Transport is balanced

> table(Cars$Transport)

### From the below table is shows that Cars has a very low count

|  |  |
| --- | --- |
| 2 Wheeler + Public Transport | Car |
| 382 | 61 |

In this problem, we need to predict whether the employee will use Car (13.76%) as a mode of transport. We can see as the data is imbalance and the variable that we want to predict is minority class. In such cases, overall accuracy of the model may be good but prediction of the minority class is not good. In real life, we face such cases like predicting bad loan where the minority class is less like in our example of bad loans where most of the time bad loan percentage are less than 5% but predicting whether a case will default or not is more important that overall accuracy. One of the way to improve prediction of minority class is oversampling.

Here we are using SMOTE algorithm (Synthetic Minority Oversampling Technique) to up sample the observations with mode of transport Car. At a high level, SMOTE creates synthetic observation of minority class by finding k-nearest neighbours of minority class n and also getting k-nearest neighbours of new data, after randomly picking the new data and finding k-nearest neighbours but randomly tweaked to get new observation.

### Generate data using SMOTE

Apply SMOTE

> library(DMwR)

> sdf <- SMOTE(Transport ~ ., Cars)

> table(sdf$Transport)

|  |  |
| --- | --- |
| 2 Wheeler + Public Transport | Car |
| 244 | 183 |

### Add the SMOTE data to the original data

> Carsdf <- rbind(Cars,sdf)

> table(Carsdf$Transport)

|  |  |
| --- | --- |
| 2 Wheeler + Public Transport | Car |
| 626 | 244 |

After adding SMOTE data, our minority class that need to be predict (Cars) has increased from 13.76% to 28%.

## Split the data into 70% for training and 30% for test

> intrain<-createDataPartition(y=Carsdf$Transport,p=0.7,list=FALSE)

> traindf<-Carsdf[intrain,]

> testdf<-Carsdf[-intrain,]

> table(traindf$Transport)

|  |  |
| --- | --- |
| 2 Wheeler + Public Transport | Car |
| 439 | 171 |

> table(testdf$Transport)

|  |  |
| --- | --- |
| 2 Wheeler + Public Transport | Car |
| 187 | 73 |

We see almost equal representation (28%) in both training and testing set for dependent variable.

# 5. Modelling

## Logistic Regression Model

### Model Building

> lgmodel <- glm(formula= Transport ~.,traindf, family=binomial)

> lgmodel

Call: glm(formula = Transport ~ ., family = binomial, data = traindf)

Coefficients:

(Intercept) Age Gender Engineer MBA Salary Distance license

-47.850873 1.224876 1.206434 0.870857 -3.182128 -0.001695 0.645775 1.908440

Degrees of Freedom: 609 Total (i.e. Null); 602 Residual

Null Deviance: 723.8

Residual Deviance: 108.9 AIC: 124.9

### Predict model using test data

> lg\_predictions <- predict(lgmodel,testdf,type="response")

## Naïve Bayes Classifier

The e1071 package holds the naiveBayes function. It allows continuous and categorical features to be used in the naive bayes model. It is count-based classifier i.e. only thing it does is – count how often each variable’s distinct values occur for each class.

### Model Building

> nbmodel <- naiveBayes(Transport ~., data=traindf)

> nbmodel

Naive Bayes Classifier for Discrete Predictors

Call:

naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:

Y

0 1

0.7196721 0.2803279

Conditional probabilities:

Age

Y [,1] [,2]

0 26.38269 3.044461

1 35.34648 2.800506

Gender

Y [,1] [,2]

0 0.2847380 0.4518045

1 0.1713784 0.3750265

Engineer

Y [,1] [,2]

0 0.7448747 0.4364290

1 0.8735961 0.3252259

MBA

Y [,1] [,2]

0 0.2687927 0.4438377

1 0.1852705 0.3862524

Salary

Y [,1] [,2]

0 13.10319 5.28899

1 34.44519 10.82715

Distance

Y [,1] [,2]

0 10.61312 3.065808

1 14.63015 2.790821

license

Y [,1] [,2]

0 0.1457859 0.3532940

1 0.8094774 0.3882705

### Predict test data using Naïve Bayes model

> nb\_predictions <- predict(nbmodel,testdf)

## Train KNN model using Caret

KNN is supervised classifier, which uses neighbor data points’ information to predict outcome variable. Neighbors are identified using distance measures such as Euclidean distance.

### Model Building

Caret package has train() method for training our data for various algorithms. We just need to pass different parameter values for desired algorithms.

We will first use trainControl() method to control the computational nuances of the train() method.

* “method” parameter refers to resampling method. Let’s try to use CV i.e., cross-validation
* “number” parameter implies number of resampling iterations

It automatically iterates through different values of “k” and identifies the optimal value.

> trControl <- trainControl(method = "cv", number = 10)

> knnmod <- caret::train(Transport ~ .,

+ method = "knn",

+ tuneGrid = expand.grid(k = 2:20),

+ trControl = trControl,

+ metric = "Accuracy",

+ preProcess = c("center","scale"),

+ data = traindf)

> knnmod

k-Nearest Neighbors

610 samples

7 predictor

2 classes: '0', '1'

Pre-processing: centered (7), scaled (7)

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 549, 549, 549, 549, 549, 549, ...

Resampling results across tuning parameters:

k Accuracy Kappa

2 0.9672131 0.9199466

3 0.9754098 0.9390375

4 0.9573770 0.8937419

5 0.9524590 0.8797580

6 0.9573770 0.8918066

7 0.9524590 0.8774636

8 0.9491803 0.8674899

9 0.9491803 0.8674728

10 0.9475410 0.8630993

11 0.9491803 0.8688093

12 0.9459016 0.8591393

13 0.9459016 0.8596482

14 0.9508197 0.8723858

15 0.9524590 0.8765382

16 0.9524590 0.8770644

17 0.9524590 0.8759968

18 0.9491803 0.8692622

19 0.9540984 0.8816032

20 0.9491803 0.8690743

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 3.

### Predict KNN model

knn\_predictions <- predict(knnmod,testdf)

## Model Tuning

### Applying Bagging (Bootstrap Aggregation)

Bagging is a way to decrease the variance in the prediction by generating additional data for training from dataset using combinations with repetitions to produce multi-sets of the original data.

#### Model Building

> mod.bagging <- bagging(Transport ~.,

+ data=traindf,

+ control=rpart.control(maxdepth=5, minsplit=4))

#### Predicting Bagging model

> bag.pred <- predict(mod.bagging, testdf)

### Applying Boosting

Boosting is an iterative technique which adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation. Boosting in general builds strong predictive models.

#### Model Building

> mod.boost <- gbm(Transport ~ .,data=traindf, distribution=

"bernoulli",n.trees =5000 , interaction.depth =4, shrinkage=0.01)

#### Predicting Boosting model

> boost.pred <- predict(mod.boost, testdf,n.trees =5000, type="response")

# 6. Validation

## Confusion Matrix of Logistic Regression model

> y\_pred\_numl <- ifelse(lg\_predictions > 0.5, 1, 0)

> y\_predl <- factor(y\_pred\_numl, levels=c(0, 1))

> confusionMatrix(y\_predl ,testdf$Transport,Positive=”1”)

In the below we can see the model has 95% accuracy

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 180 6

1 7 67

Accuracy : 0.95

95% CI : (0.916, 0.9731)

No Information Rate : 0.7192

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8767

Mcnemar's Test P-Value : 1

Sensitivity : 0.9178

Specificity : 0.9626

Pos Pred Value : 0.9054

Neg Pred Value : 0.9677

Prevalence : 0.2808

Detection Rate : 0.2577

Detection Prevalence : 0.2846

Balanced Accuracy : 0.9402

'Positive' Class : 1

## Confusion Matrix of Naïve Bayes model

> confusionMatrix(nb\_predictions,testdf$Transport)

In the below we can see the model has 93.85% accuracy

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 180 9

1 7 64

Accuracy : 0.9385

95% CI : (0.902, 0.9644)

No Information Rate : 0.7192

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8463

Mcnemar's Test P-Value : 0.8026

Sensitivity : 0.8767

Specificity : 0.9626

Pos Pred Value : 0.9014

Neg Pred Value : 0.9524

Prevalence : 0.2808

Detection Rate : 0.2462

Detection Prevalence : 0.2731

Balanced Accuracy : 0.9196

'Positive' Class : 1

## Confusion Matrix of KNN model

> confusionMatrix(knn\_predictions,testdf$Transport)

In the below we can see the model has 95% accuracy

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 181 7

1 6 66

Accuracy : 0.95

95% CI : (0.916, 0.9731)

No Information Rate : 0.7192

P-Value [Acc > NIR] : <2e-16

Kappa : 0.8757

Mcnemar's Test P-Value : 1

Sensitivity : 0.9041

Specificity : 0.9679

Pos Pred Value : 0.9167

Neg Pred Value : 0.9628

Prevalence : 0.2808

Detection Rate : 0.2538

Detection Prevalence : 0.2769

Balanced Accuracy : 0.9360

'Positive' Class : 1

## 4. Confusion Matrix for Bagging

> confusionMatrix(bag.pred,testdf$Transport)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 182 3

1 5 70

Accuracy : 0.9692

95% CI : (0.9403, 0.9866)

No Information Rate : 0.7192

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9244

Mcnemar's Test P-Value : 0.7237

Sensitivity : 0.9589

Specificity : 0.9733

Pos Pred Value : 0.9333

Neg Pred Value : 0.9838

Prevalence : 0.2808

Detection Rate : 0.2692

Detection Prevalence : 0.2885

Balanced Accuracy : 0.9661

'Positive' Class : 1

## 

## 5.Confusion Matrix for Boosting

> y\_pred\_num <- ifelse(boost.pred > 0.5, 1, 0)

> y\_pred <- factor(y\_pred\_num, levels=c(0, 1))

> confusionMatrix(y\_pred ,testdf$Transport)

In the below we can see the model has 99.62% accuracy

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 186 0

1 1 73

Accuracy : 0.9962

95% CI : (0.9788, 0.9999)

No Information Rate : 0.7192

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9905

Mcnemar's Test P-Value : 1

Sensitivity : 1.0000

Specificity : 0.9947

Pos Pred Value : 0.9865

Neg Pred Value : 1.0000

Prevalence : 0.2808

Detection Rate : 0.2808

Detection Prevalence : 0.2846

Balanced Accuracy : 0.9973

'Positive' Class : 1

# Model Comparison

In this problem, we are keen to identify positives very accurately. Hence, we will not just evaluate models based on accuracy on test data as discussed before; we will also use sensitivity as metric to compare model performances. (Assuming we want to find ability of our model to predict employee who uses car)

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Accuracy | Sensitivity | Specificity |
| Logistic Regression | 95 | 91.78 | 96.26 |
| Naïve Bayes classifier | 93.85 | 87.67 | 96.26 |
| K Nearest Neighbors classifier | 95 | 90.41 | 96.79 |
| Bagging | 96.92 | 95.89 | 97.33 |
| Boosting | 99.62 | 100 | 99.47 |

In different problem, we use different parameter to compare models. For example

1. To find quality of disease (Cancer) detecting machine where a person suffering from disease is positive in the model. We would be interested to know out of people who are suffering from cancer, how many have been detected by the machine. Thus sensitivity or recall will be used to choose model.
2. A bank problem, while disbursing loan to assure that they do not disburse bad loan, they will be disbursing loan only to people where the model is saying there will be no default. They would be interested in how many from them will not default because of the model efficiency. Here loan default is positive in model. We will be interested in Neg Pred Value.
3. If the same bank wants to compare model on how many good clients are they loosing because of model rejecting them, than they would refer Pos Pred Value.

# Conclusion

We explored the data and completed the required data cleaning and pre-processing. We removed the missing values, capped the outliers, and removed multi-colinearity. We created 5 models namely Logistic Regresion, Naïve Bayes, KNN, Bagging and Boosting. From the validation scores of confusion matrix we can see that Boosting has the highest accuracy followed by Bagging. With respect to Sensitivity also boosting is the best model.

# Recommendation

Further validations to check over-fitting of the model should be done before deploying the model into production. Data transformations like log transformations can also be done on the skewed variables and the impact to the model can be validated. Techniques of over-sampling & under sampling can be tried for the imbalanced data problem.