DATA ANALYTICS ASSIGNMENT

COURSE CODE: CS7DS1-A-Y-201718
STUDENT NUMBER: 17311993
STUDENT NAME: Sethuram Ramalinga Reddy

Table of Contents

Table of Figures

1. Fisheries Dataset

1.1 Introduction:

The Fish Dataset deals with the Fisheries in Ireland. It gives information about the Net profit/loss obtained from each type of boat. This has three sub-datasets where each gives unique information. The Species dataset gives information about the names of the fishes caught in Ireland. There are totally 100 varieties of them. The Fish Dataset tells the monetary value of each of these 100 species for every vessel in a year. The Economic dataset gives the information about attributes involved in the fishing like the details of the boat, jobs, income and expenditure and finally the net profit/loss.

1.2 Examination of Data:

As soon as you see the dataset, we can find that the data is not clean and cannot be used straight away for the analysis. So, the pre-processing of data is necessary in order to proceed with the analysis.

1.2.1 Data Cleaning:

- The column names of the data are not uniform and they include some special characters. So, by using the janitor package these column names are brought into a uniform format.
- Now let's consider each column individually. The "Size Category" column has been imputed with irrelevant data like **18-Dec and 12-Oct**. This doesn't make any sense to the size category so this has to be replaced with NA or NULL value. Then, This variable has to be converted into factors since this is a categorical variable.
- All the columns that represent continuous variable are not in proper numerical format. They have commas and euro symbols within them which will make the analytical tool to treat them as character field instead of numerical field. So these symbols and commas are removed and converted them into numerical predictor variable which would make them suitable for analysis.
- Finally, the columns are converted to continuous or categorical based on what they represent.

1.2.2 Missing Values:

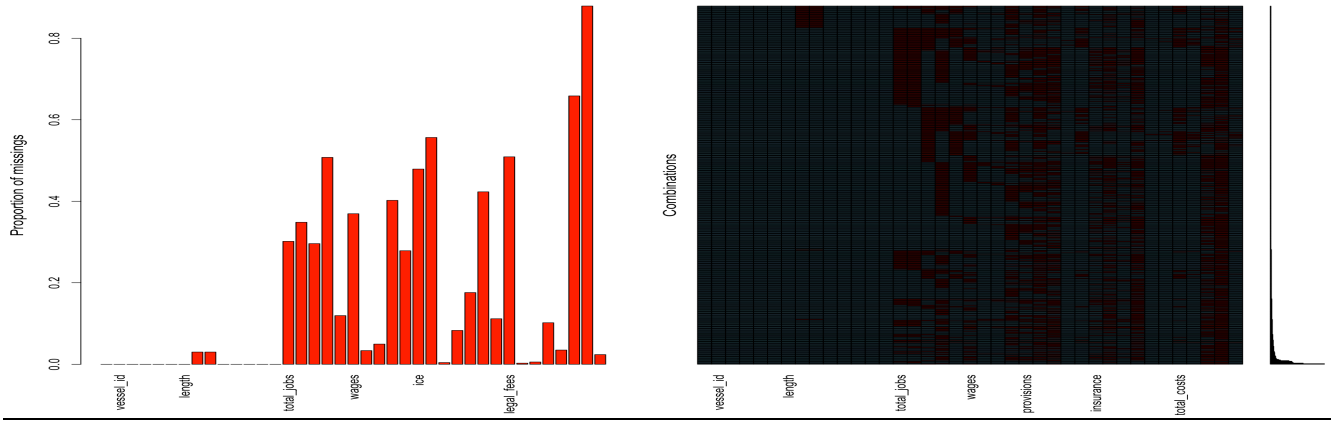


Figure 1.1 Aggregation Plot for Missing Values

As you can see from the plot, this dataset has a large number of missing values. The columns having a high percentage of missing values are given below:

i. Total Jobs
ii. FTE
iii. Non Fishing Income
iv. Wages
v. Filters Lube Oil
vi. Ice
vii. Dues Levies
viii. Loan Interest
ix. Legal Fees
x. Depreciation
xi. Sundry Receipts

From these variables, if we put up the percentage for the number of missing values in each column to the total number of rows, we can get the missing value percentage. The columns mentioned above had a missing value percentage of more than 30%. So, if we take a cutoff of 40%, then the following columns will be removed.

i. Non Fishing Income
ii. Ice
iii. Dues Levies
iv. Loan Interest
v. Legal Fees
vi. Depreciation
vii. Sundry Receipts

From these missing variables, we can make some assumptions as mentioned below:

- Most of them don't have income other than fishing or else they are not willing to provide their non-fishing incomes.
- More than half of them have their own arrangements for ice and they do sell ice for others.
- More than 40 percent of the people have not taken any loan or else they have taken the loan and not paid.

This is done because when about 40% of the data is missing then the most important information from the data will be lost. Assuming the variable to be normally distributed, then the missing value percentage of 40 means that there might be a loss of data in the inter-quartile range i.e. in the 25-75% region of the distribution. As a result, the information might be misrepresented.

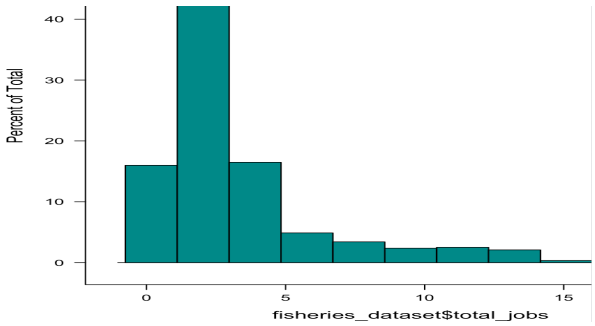
The reference number column is used only for uniquely identifying the variables, so this is not related to our target and hence it is removed from the dataset.

The Gross profit is the profit without any deductions and we also have expense details along with total cost. This variable overshadows the influence caused by the other predictor variables and model is centered only around this variable. So, this variable is removed.1.2.3 Missing Value Imputations:

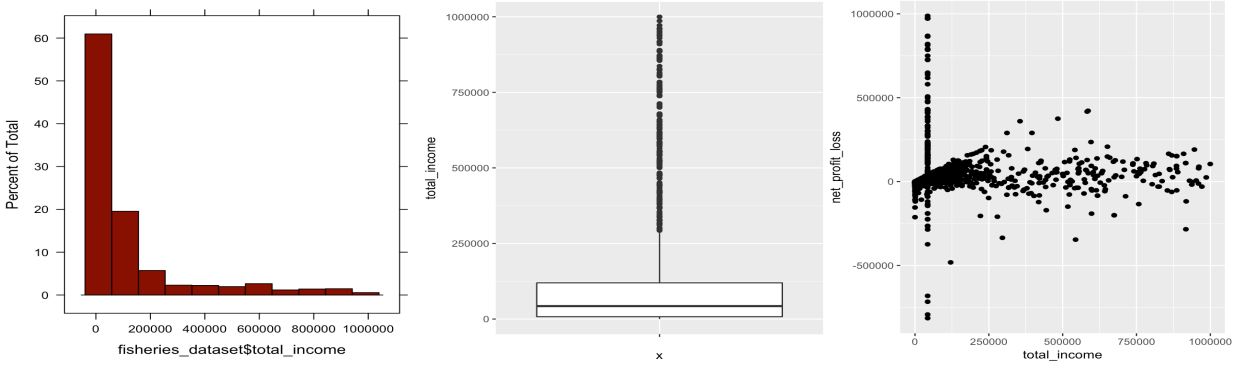
The predictor variables whose missing value percentage is less than 40% must be imputed with a suitable value to fill those missing values. For this purpose, the rough-fix strategy in R is used.

For continuous variables, the mean of the data gets imputed and for categorical variables, the most repeated instance will get imputed. This might introduce bias in the dataset but it is low so this can be used for prediction of our target variables.

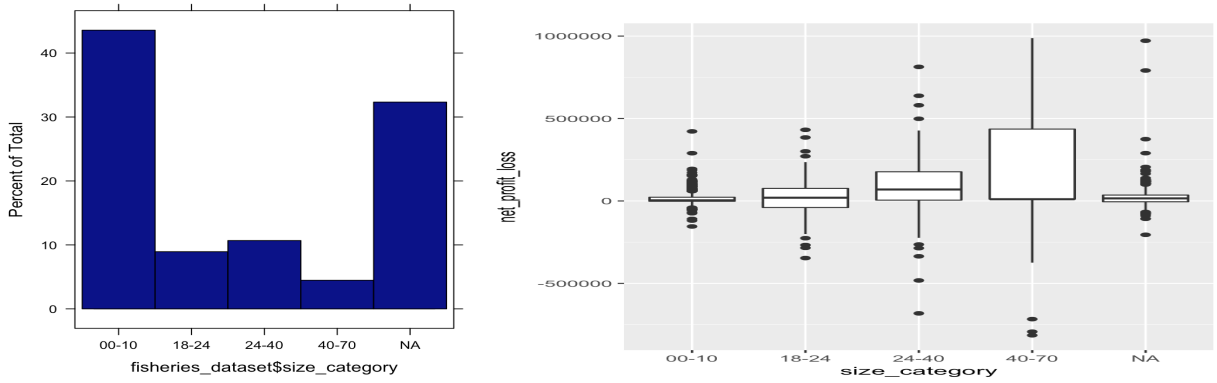
1.2.4 Variable Analysis:



The predictor variable has less positively skewed histogram which indicates the presence of few outliers. This variable is not so dependent on target variable because the points don't follow a linear pattern. We can say in most of the cases when the number of jobs is less, then the net profit is high.



The predictor variable "total_income" has a positively skewed Distribution and the mean is shifted towards the right and this is due to the influence of a large number of outliers in that variable. The relationship between total income variable and net profit/loss is not linear as you can see in the scatter plot. Even though there is a large total income we can say that the net profit/loss is same, this is because of the expenses. In another word, larger the income, larger the expenses.



In predictor variable "size category", we can see that 40% of the data is present in 00-10 category and very few percentage in other categories. There is about 30% of missing data. If we

consider the relationship between size category and net-Profit/Loss, there is a slight increase in median value as we increase levels of size category without considering the NA category. We can also say that in 40-70 category, most of the vessels have high net profit. From the above observations, we can conclude that the larger the size category, larger number of vessels will have larger net profit.

1.2.5 Correlation:

Now we have to find the correlation between the variables which is used to identify the strength of the relationship between the variables along with its direction of the relationship. Here we use Pearson correlation method to determine the linear relationship between two variables and the result is plotted using correlation plot.

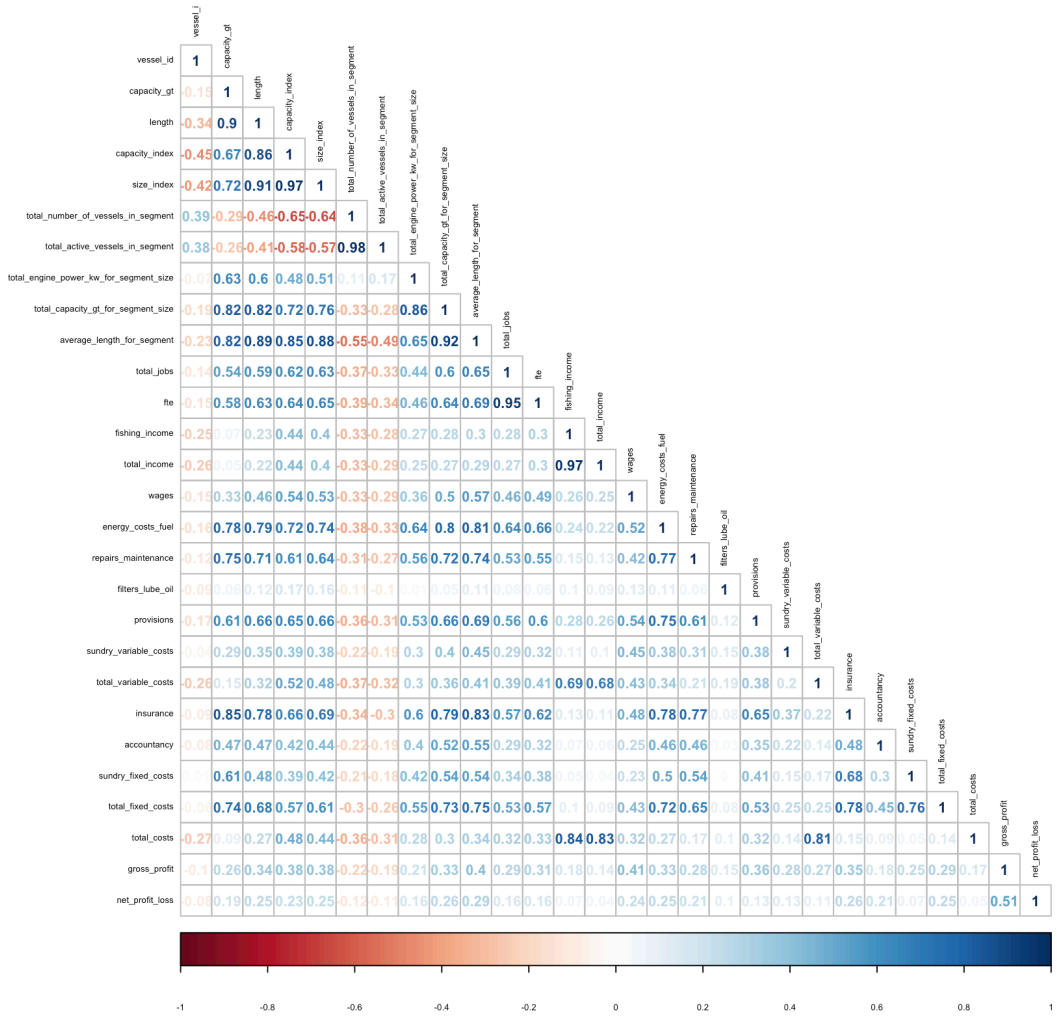


Figure 1.2 Correlation Plot representing Correlation Matrix

Based on the correlation matrix plotted above, the correlation of magnitude greater than 0.6 are highly correlated and we can select any one from these correlated pairs which will avoid

the repetition of exhibiting same behavior within the predictor variables. By doing this, the performance of the model will be improved. The highly correlated variables are mentioned below:

Average Length for Segment, Size Index, Capacity Index, Length, Total Capacity in GT for Segment Size, Energy Costs Fuel, Insurance, Capacity, Provisions, FTE, Repairs Maintenance, Total Fixed Costs, Total number of vessels in the segment, Total variable costs, Total Cost and Fishing Income.

1.2.6 Variable Importance:

The above method weeds out the mutually correlated variables and there is another method which selects based on the importance of that variable. Here the variable importance is calculated by excluding that predictor variable and measuring the increase in mean square error or decrease in purity. Based on this feature the variable importance is plotted and its shown below.

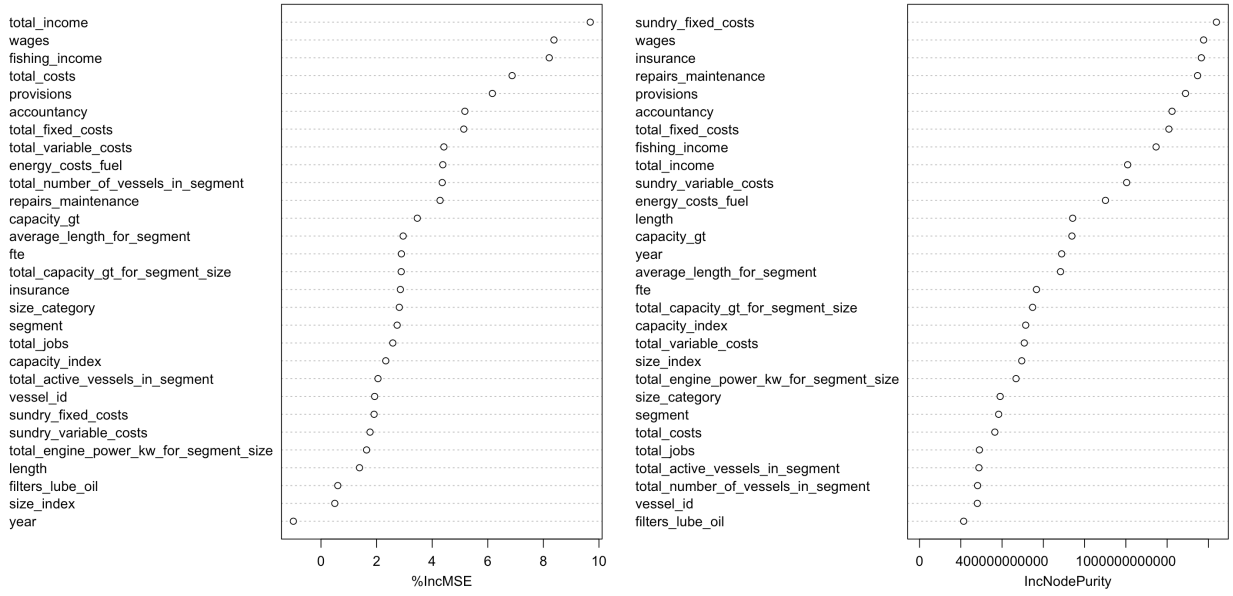


Figure 1.3 Variable Importance

In this plot, it is ranked from most important to least important and we can see that if we remove the total income and train our model then this has the highest increase in mean squared error and similarly when this is included, the increase in purity is very high. There is no variable which has a very high difference in importance and we can see that the top five important variables have very less difference in importance between them. This makes sure that our model will not be based on one single variable overshadowing the effects of other predictor variables.

As we have done earlier, we split the data into a training dataset and testing dataset in the ratio of 70:30. The first set of data (70%) will be used for training and the remaining 30 % of the data will be used for evaluation. This will make sure that our model is not exposed to our testing dataset and makes our prediction more accurate.

1.3 Single Tree Model:

Now the data is ready to build the Decision tree model. The main principle behind the decision tree is to divide the sample space into many small rectangles. The target variable is Net Profit/Loss. Since this is a continuous variable the decision tree will act as a regression tree. In the regression tree, at each node, it splits the data into two branches and the split is chosen in such a way that it reduces the mean squared error. This is repeated until the size of the tree reaches the user-specified size and finally, this becomes the terminal node. In regression trees, the stopping criterion is given by the user.

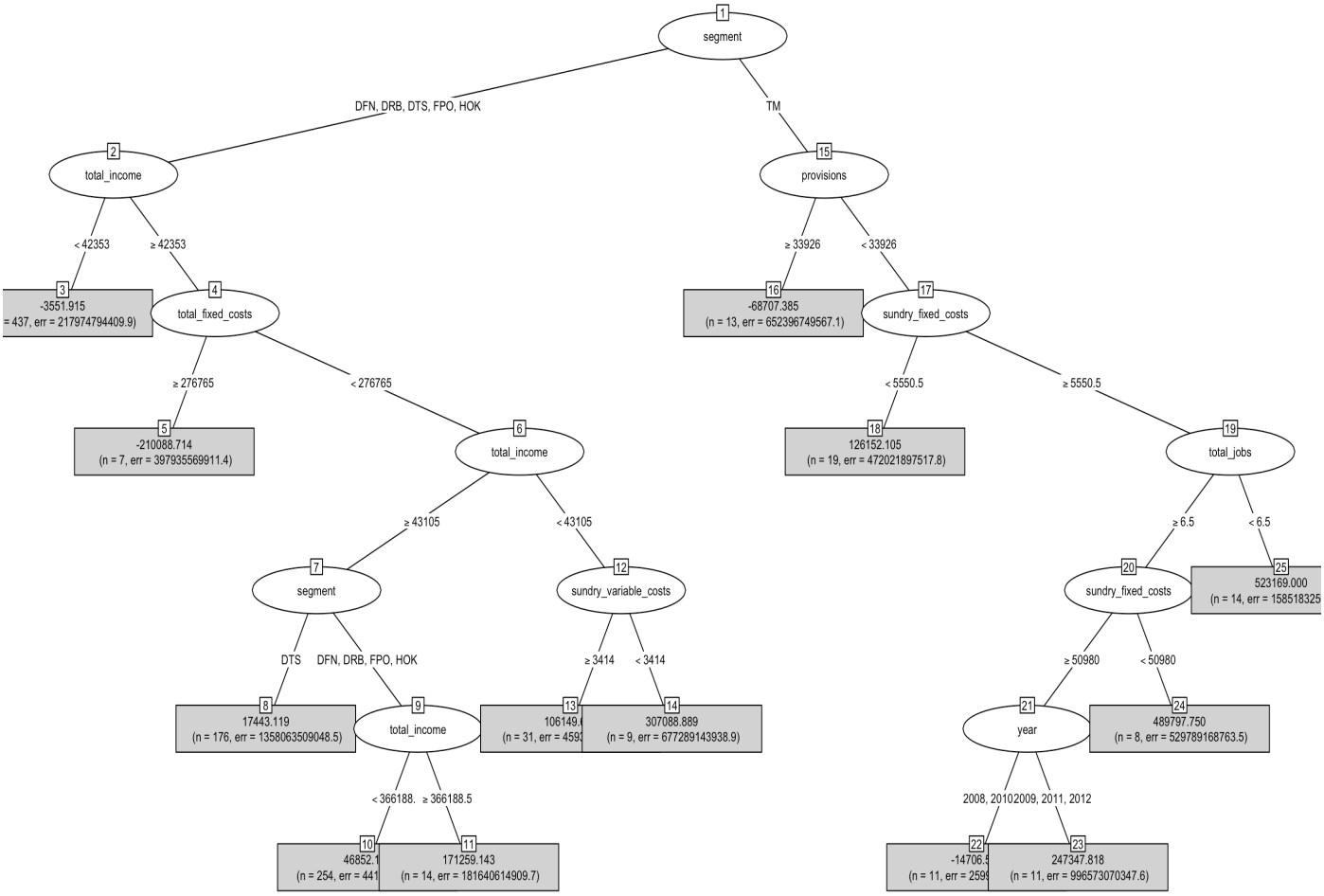


Figure 1.4 Decision tree before Pruning

Now the Decision tree is built by using rpart in R and Gini is used as a splitting Criterion which is the purity measure of the data. Initially, the complexity parameter has the default value and the complete tree is formed which would ideally be a large tree. Now we have to find the best split which would be helpful to prune the tree.

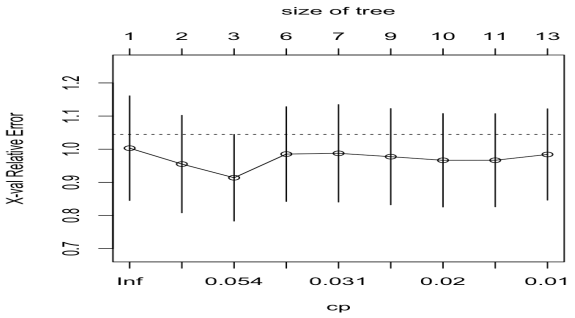


Figure 1.5 Initial CP value

We have found the best CP value as 0.02835051 by finding the minimum of the cross-validation error and standard deviation. Now with the help of the best complexity parameter, we can prune the trees so that all the weak classifiers are removed and improve our prediction accuracy. After the tree has been pruned, the regression tree is plotted and it is shown below:

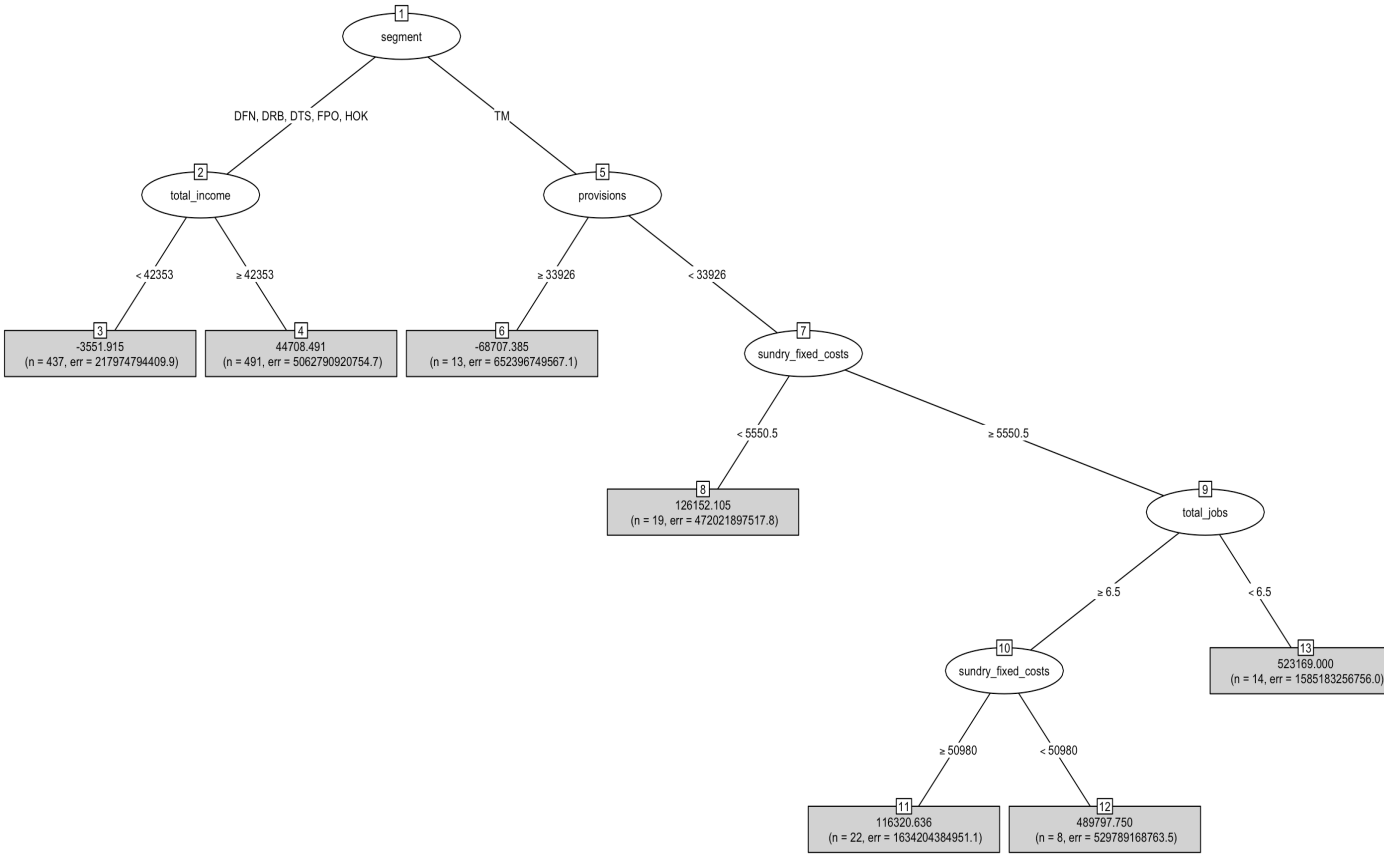


Figure 1.6 Decision Tree after Pruning

In the above tree, the root node shows the mean of predicted values and number of cases (n)in that terminal node and error rate represented as err.

1.3.1 Conclusion from Decision Tree:

- The Net Profit/Loss is High when all the following conditions are satisfied:
  - segment is TM
  - provision is less than 33926 euro
  - sundry fixed cost is greater than 5550 euro
  - total jobs is less than 6.5
- The Net Profit/Loss is Low when any of the following conditions satisfied:
  - total income is less than 4200 euro
  - amount spent for provisions greater than 3390 euros

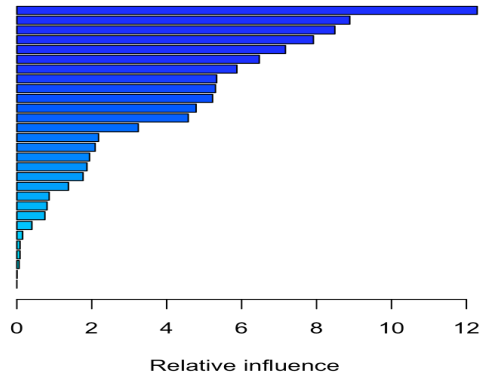
1.3.2 Evaluation:

In Regression Tree, the terminal nodes hold the mean value of all the values of the predictor variables in that partitioned region. Now we will use this model to do prediction on the test dataset. Once the target value has been predicted for the test dataset, we can evaluate the model using Root Mean Squared Error(RMSE). RMSE is the average of the distance of data points from the fitted line measured along the vertical line. The RMSE obtained is 138767. The RMSE is so high because the predicted values are the mean value of that region so the predicted values are nowhere close to the actual values. As a result, we are getting very high RMSE values.

1.4 Ensemble Method 1 - Boosting:

Boosting is a process of transforming weak learners into strong learners. The weak learner is a learner that has an error rate of less than 50%. In this technique, our aim is to reduce the high variance present in the learners by averaging the outputs from lots of models built from these bootstrapped training data in order to avoid over-fitting. In order to achieve this, the prediction outputs from the base learners where the output would be a weak rule or it might have prediction error and the observation which was found to be having an error is weighted and given as input to another weak learner. This process is done repeatedly until we obtain a high accuracy. This ensemble technique is different from the Bagging which we have used earlier. In Bagging all the models are built with equal weight and their model outputs are not given as input to the other models. In this way, we can describe Boosting as a sequential process.

In this process, we use decision tree as a weak learner. This is because the weak learners should always be simple and not complex models like neural nets. Here, the Gradient Boosting algorithm is used which is very much similar to the boosting. It tries to correct the loss between the first learner and actual value by providing this loss as an input to the next model along with the predictor output. In each step, the cost function is minimized. This is repeated until some cutoff is reached or when there is no change in the cost function. In this way, we improve the accuracy successively. Here the target variable is Net-Profit/Loss and the model is trained using training dataset. The parameters selected for the model are the Distribution which is Gaussian and the number of trees which is 1000.



	var	rel.inf
repairs_maintenance	repairs_maintenance	12.288240210
total_fixed_costs	total_fixed_costs	8.888132214
insurance	insurance	8.489679800
sundry_fixed_costs	sundry_fixed_costs	7.917581211
length	length	7.170499617
provisions	provisions	6.471558200
wages	wages	5.873198939
accountancy	accountancy	5.332938795
segment	segment	5.302292990
total_capacity_gt_for_segment_size	total_capacity_gt_for_segment_size	5.224926052

Figure 1.7 Relative Influence of each variable

The above plot gives variable importance details. The repair maintenance is ranked First and this has the relative influence of 12.2. on target variable (Net –Profit/Loss). From the above plots and values, we can say that the Net-Profit/Loss depends majorly on top 5 important variables. This can be proved by using the partial Dependency plot given below. This shows that repair maintenance is directly related to the Net Profit/Loss and Total Fixed Cost is negatively correlated with the target variable.

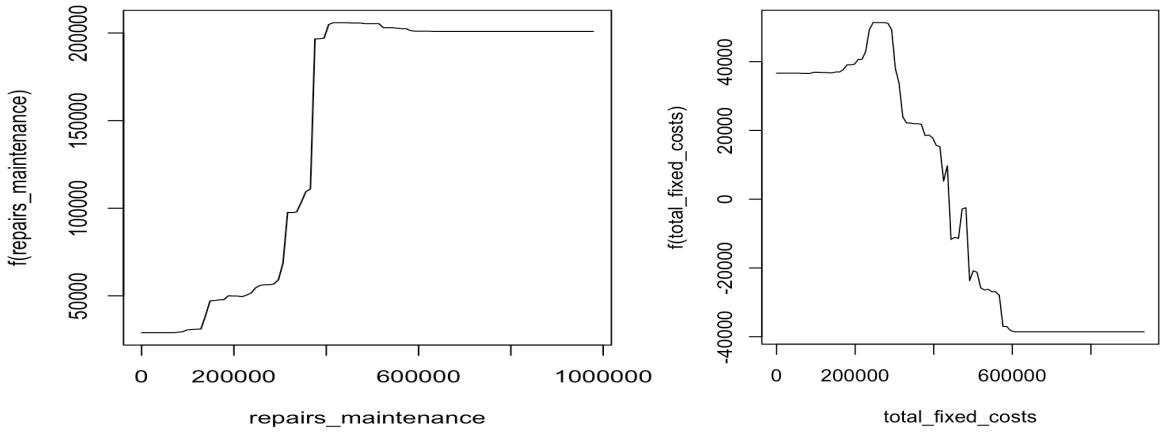


Figure 1.8 Partial Dependency Plots of repairs Maintenance and total fixed costs

Now we can evaluate the performance of boosting the test dataset. For this, we have to calculate the mean squared error which is the difference between the observed and predicted for each case. Here, the Boosting is done by increasing the number of trees from 100 to 10,000 over a step rate of 100. So, we can find the MSE for every 100's of trees and from this, we can infer the performance.

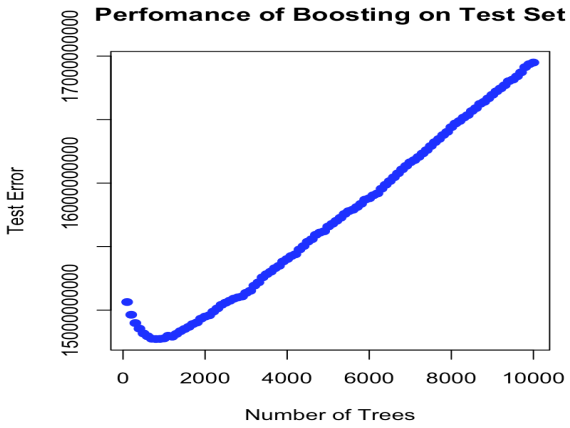


Figure 1.9 Performance of Boosting

From the plot, we can say that the MSE values initially decrease to 2000 trees and after that the MSE increases which indicate the presence of overfitting.

1.5 Ensemble Method 2- Stacking:

Stacking is a general procedure where individual learners are combined to form strong learners. Here there are two levels of learners. First Level learners are the individual learners and the second level learners are called as meta-learners that are used to combine first level learners. In First level learners, a new dataset is created and this is given as an input to the second –level learners. In first level learners, original training data is used to create the output and we should make sure that the training data which is used to train the first level learners must not be included to predict the output from the first learners. For this purpose, K-Fold cross-validation is used. For example, if we consider dividing the data into 10 folds, then one-fold can be used as a validation dataset and this is done with each fold successively. By this way, the initial learners will predict more accurately without any exposure to bias in the dataset. The first level classifiers can be of any kind but the performance improves if they are less alike. In another word, stacking ensembles are mostly heterogeneous and they can also be constructed using homogeneous ensembles. The second level learners always need not be a machine learning algorithm or model, it can also be a simple mathematical rule like average and also voting. Before giving the first level learner output to the second level learner as input we have to put up a correlation matrix and check the correlation between the outputs. If there is a high correlation we can select one of them in order to avoid redundancy of information. If they are very less correlated, then we can get good performance from the meta-learners.

There is a new package called “caretEnsemble” for performing stacking procedure in R. Here the algorithms, chosen to be first level learners are linear regression, decision tree and support vector machines (SVM). The first level learners chosen are somewhat homogeneous and all the learners can be used for regression problems. This package has three main methods CaretList, CaretEnsemble, and CaretStack. CaretList is used to build the models specified by the user for same training data. CaretEnsemble uses the logistic regression model to combine the outputs from the First level learners whereas the CaretStack uses user-specified models to combine them. The parameters given for the CaretList Function are the algorithm list and the control arguments which include cross-validation factors and Boolean input to save prediction and class probabilities. The first level learners are formed using tuning parameter as 10-Fold cross-validation. Resample function is used to analyze and visualize the output from CaretList.

```
Call:
summary.resamples(object = results)

Models: rpart, lm, svmRadial
Number of resamples: 30

MAE
             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
rpart     35829.22 50758.98 56584.75 55035.24 61196.40 69079.43    0
lm        28945.03 42568.16 48394.81 49266.03 56427.60 68334.79    0
svmRadial 34884.50 46523.16 53732.98 52394.51 57919.85 63630.06    0

RMSE
             Min.   1st Qu.   Median     Mean  3rd Qu.     Max. NA's
rpart     62629.20  98191.31 124835.7 116277.7 134339.2 154891.7    0
lm        49042.76  89222.94 117117.6 116506.0 140989.6 171837.7    0
svmRadial 61862.76 107570.64 127308.5 122014.9 138997.5 165798.4    0

Rsquared
                Min.    1st Qu.     Median       Mean    3rd Qu.      Max. NA's
rpart     0.00500222936 0.11529654 0.17255083 0.22077849 0.29476049 0.5388238    0
lm        0.00006860764 0.06287281 0.28143167 0.26388174 0.44357084 0.5759671    0
svmRadial 0.02232528144 0.05072474 0.05957657 0.07378305 0.08767522 0.1895211    0
```

From the above output, we can find the Root Mean Squared Error (RMSE) and Coefficient of Determination (R2) for every individual model i.e. for first level learners. We can find that the mean R2 for linear regression model is 0.26 which moderately fits the data. So, all these outputs from the first level learners must be less correlated in order to produce a good performance. The correlation matrix is given below:

```
                rpart        lm svmRadial
rpart      1.0000000 0.6961796 0.5380819
lm         0.6961796 1.0000000 0.7411117
svmRadial  0.5380819 0.7411117 1.0000000
```

We can find that all the models are highly correlated and we can also use scatter plot matrix to prove the relation between them.

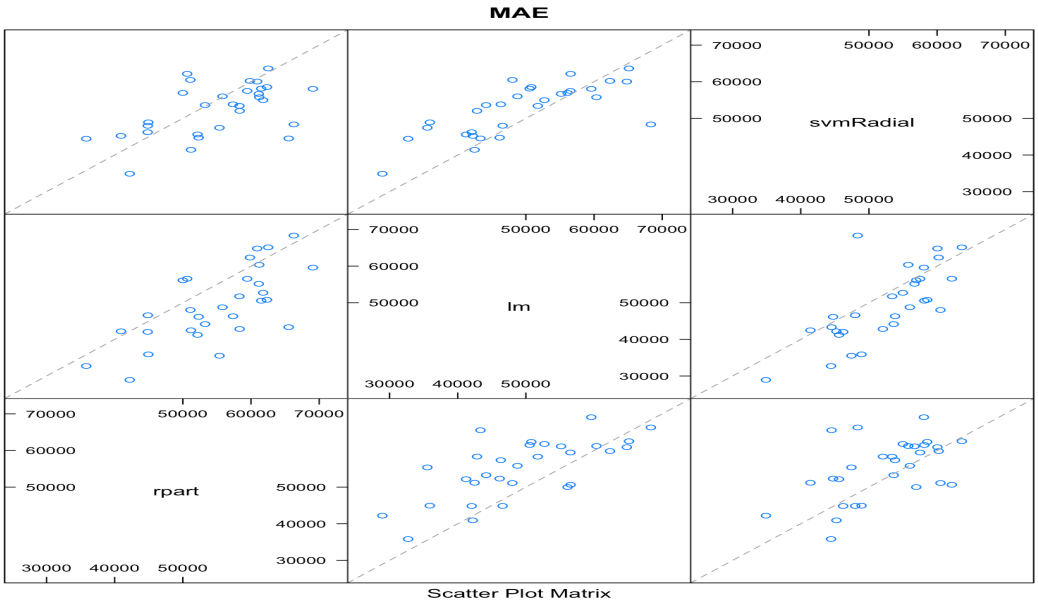


Figure 1.10 Scatter Plot Matrix to find Correlation

Now we can combine these models using CaretStack function with linear regression model and the performance is given below:

RMSE	Rsquared	MAE
109442.5	0.2605424	45235.63

Hence this ensemble model doesn't fit the data very well and this model is not suitable for performing regression.

1.6 Comparison of Models:

By comparing the value of RMSE for all the tree models, we can say that the performance of stacking is better than both the models. In Boosting, the performance depends on the number of trees. From the evaluation of Boosting, we can find that the RMSE is very low when the number of trees was less than 1000 and higher the number of trees, higher the RMSE value.
The decision tree has the highest RMSE value so when compared to these three models this gave the poorer prediction.

My assumption based on the above evaluation is that the above-used models are not suitable for this particular dataset. This is because these models don't give the exact predicted value for each case whereas it gives only the mean value that is derived based on the training dataset. If we have used multi linear regression then this particular dataset would be fitted very well and the prediction accuracy would have been more. But in Multi Linear Regression, Unlike Trees, it is difficult to find the important predictor variables and their cutoff values used in splits This makes out interpretation little bit difficult. If we use both Regression trees and Multilinear regression, then we can form a good model that suits our data.

2. Employee Attrition Dataset

2.1 Introduction:

The Employee attrition dataset gives professional details about the person and also his employment details. Our Objective is to find whether the employee is going to leave the company or not. We have to find the factors which make him leave the company so that, the efforts can be taken to retain him. This is done by careful examination of the Dataset provided and creating a model based on the available data and by the implementation of various available algorithms using R.

2.2 Examination of Data:

First, the dataset is read from the excel sheets in order to analyze them. As you can see the dataset provided is a very clean data and it does not have any encoding like UTF-8, latin1, etc. Now we use janitor package to make the format of the column names uniform. Now we have to do descriptive statistics on the data in order to find its structure. Once you execute the summary command on the dataset, you can find that many predictor variables would have a character class. So, we have to convert these variables into factor class so that they represent a factor class and if required you can initialize suitable levels based on the requirement. When the dataset is read into R, though some variables are factors, they are read as numeric values. So, we have to convert them back to factors. The below-given variables are the additional factor variables:

i. Education
ii. Environmental Satisfaction
iii. Job Involvement
iv. Job Satisfaction
v. Performance Rating
vi. Relationship Satisfaction
vii. Work-Life Balance

In this dataset, the target variable is a categorical variable, with two classes Yes (Attrition = "Yes") and No (Attrition = "No"). So, we have converted this into a factor such that the algorithm which we are going to implement will consider that we are working on a classification problem or else it might take the problem as regression or unsupervised scenario. Now the data is ready for the analysis.

In the dataset, we can see that there are certain predictor variables which makes no sense i.e. the predictor variables don't contribute to the target variable. In another word, they can be used as the predictor variable for analysis according to statistical perspective but in the business point of view, they do not have any impact on the target. Consider the variables mentioned below:

i. Employee Number - This is only used to uniquely identify the employee.
ii. Standard Hours - The Standard working hours will be same for all the employees within an organization.
iii. Employee Count - This is same for all the employees irrespective of department or job designation.
iv. Over 18 – All the employees in an organization should have an age above 18.

All the variables mentioned above have their own purpose but they don't influence on the attrition rate in the business sense. We can use this in our analysis but this will only increase our predictor variable count more and decrease the performance of the model. For better performance, these variables are dropped from the dataset.

Now we have to find the presence of any missing values in each column since missing value may affect the performance of the model. It will introduce bias in the prediction. If the missing value percentage for each column is high (more than 40%) then we can drop the column from the dataset. This will reduce the set of available predictor variables. In this dataset, there are no missing values in each column and this is proved by using an aggregate function in R and it shows the count of missing values in each column which is zero for all the predictor variables. If there were any missing values then we can impute values by using mean, median or any other rough fixes. For categorical variables, the most repeated variables are imputed to these missing values. The result of the aggregate function implemented is given below.

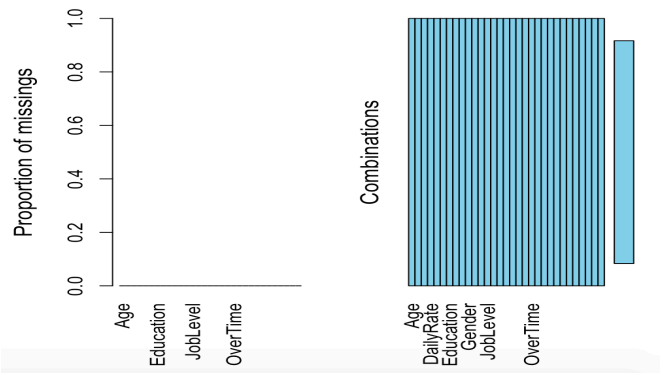


Figure 2.1 Missing values

2.2.1 Variable Analysis:

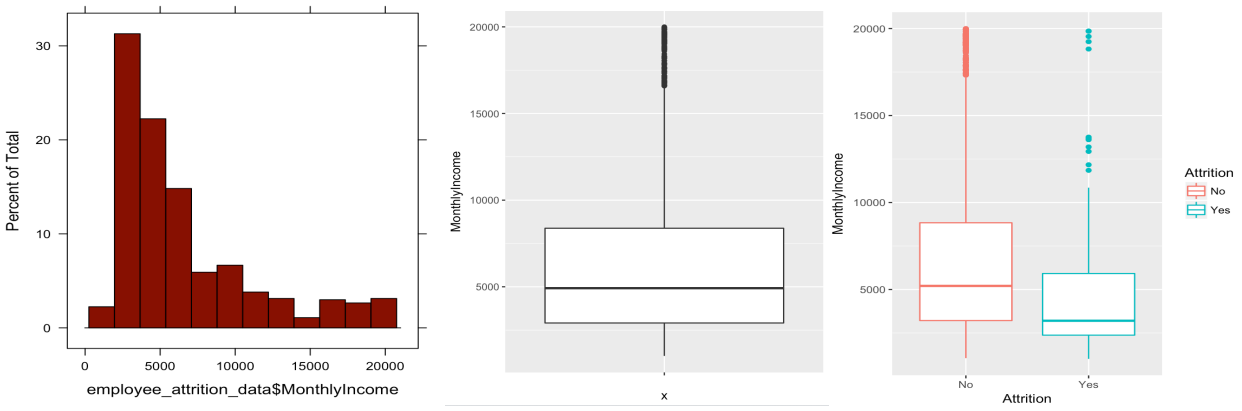


Figure 2.2 Variable Analysis for Monthly income

From the above histogram, we can say that the monthly income variable follows a positively skewed distribution. This indicates that mean of the variable is shifted towards positive direction because of the presence of outliers in the dataset. The presence of outliers can be found out by plotting a boxplot. The variance is high on the positive side of the mean than the negative side. There is more number of variables in the 75-100% part of the distribution and very few

number of variables in 0-25%. Now we can find the relationship between the monthly income and each category of attrition. For class “No”, most of the employees are earning an average monthly income of 5000 euros and only fewer employees in the 25-50% of distribution earn less than the average. For class “Yes”, the average itself is very low when compared to the “No” category. We can say that the around 50 % of the employees in class Yes left the job because they were not satisfied with the given salary.

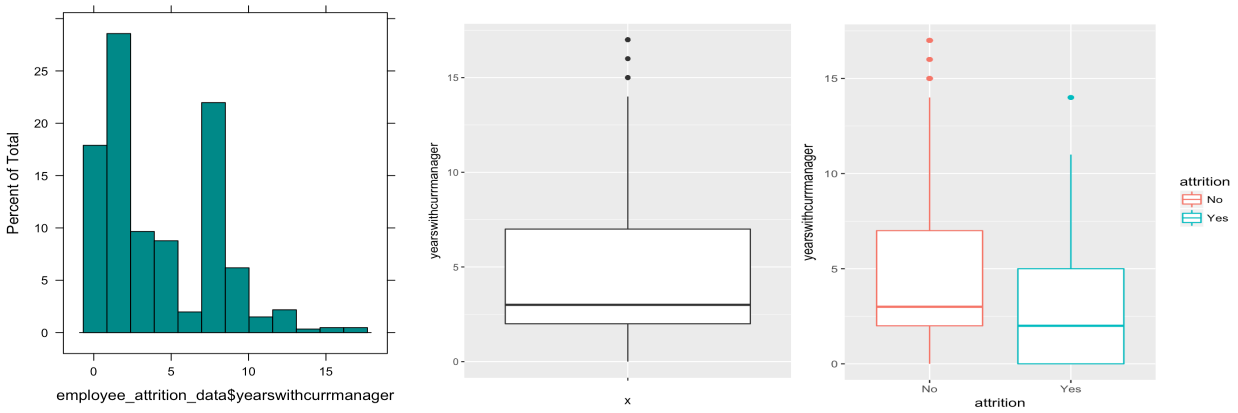


Figure 2.2 Variable Analysis for Years with Current Manager

From the above histogram, we can say that the tail got extended indicating the presence of skewness. There is large number of employees who have worked with the current manager for 1-2 years and 6-7 years. The boxplot brings out the outliers clearly where three of the values i.e. years with the current manager is greater than 15. Most of the employees have an experience greater than 3. For class “No”, the employees are with the current manager for 3-7 years. For class “Yes”, the years with the current manager is between 0-5 more most of the employees. So, we can say that the people who are working with the current manager for less than 5 years have high chances of leaving the company.

ATTRITION

JOB INVOLEVEMENT	No	Yes
1	55	28
2	304	71
3	743	125
4	131	13

Figure 2.3 Variable Analysis for Job Involvement

From the table, we can say that the employees with very high involvement have less chance of leaving the company. The employees whose job involvement is low has high chances of leaving the company. The employees with job involvement medium and high have only less than 20% chances of leaving the company. From the above values, we can say that the job involvement is not a strong reason for leaving the company.

Now we have to examine the correlation between the predictor variables. This has to be examined because a high correlation between the predictor variables may lead to the repetitive

behavior of the same variable. In another word, if the correlation between two predictors is more than 0.6 then we can select any one of the variables instead of both, this will contribute to the increase in accuracy and also, we are reducing the number of available predictor variables for building the model. By reducing the number of variables, we are improving the performance of the model by reducing the complex and high computations. The correlation is computed using Pearson's method and correlation plot is given below.

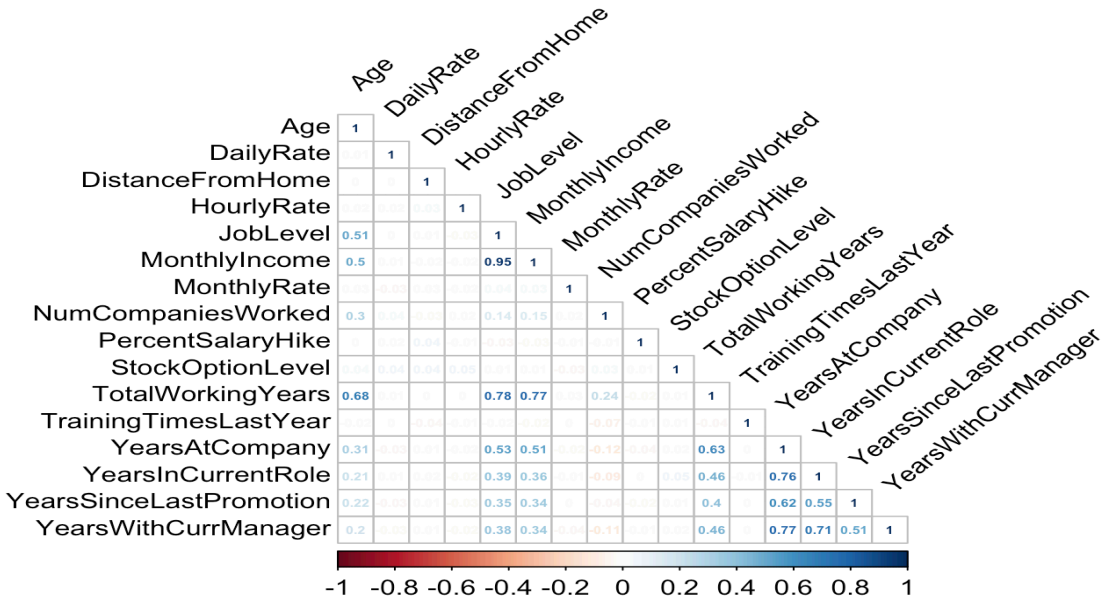


Figure 2.4 Correlation Plot

In the above figure color of the numbers represents the direction of the correlation (Red represents negative correlation and Blue represents positive correlation). The negative correlation means one variable is inversely proportional to the other and vice versa. The positive correlation means one variable is directly proportional to the other and vice versa. The number represents the magnitude (Strength) of the correlation. If the magnitude is 0.5 to 0.6, then it has moderate correlation and if the value is greater than 0.6 then it has a strong correlation. Based on the above figure, the highly correlated variables are given below:

i. Total Working Years
ii. Years at Company
iii. Job Level
iv. Years in current role
v. Years with current manager

By selecting the correlation value to be greater than 0.75, we pick only one of them between these strongly correlated pairs.

The above-stated method was based on correlation and there is another method of selecting features and is called as Variable Importance. This tells the significance of a variable for classification. This method is inbuilt in some of the models like Decision trees, random forest etc. Variable Importance is calculated using the decrease in Gini (impurity). Here the variables are ranked using the amount of decrease in Gini index at each split. The variable that decreases the

impurity to the largest ranks the highest in order. The variables having the highest decrease in Gini will contribute to the highest homogeneity. In this way, it is calculated and a graph is shown below indicating the importance of the variable.

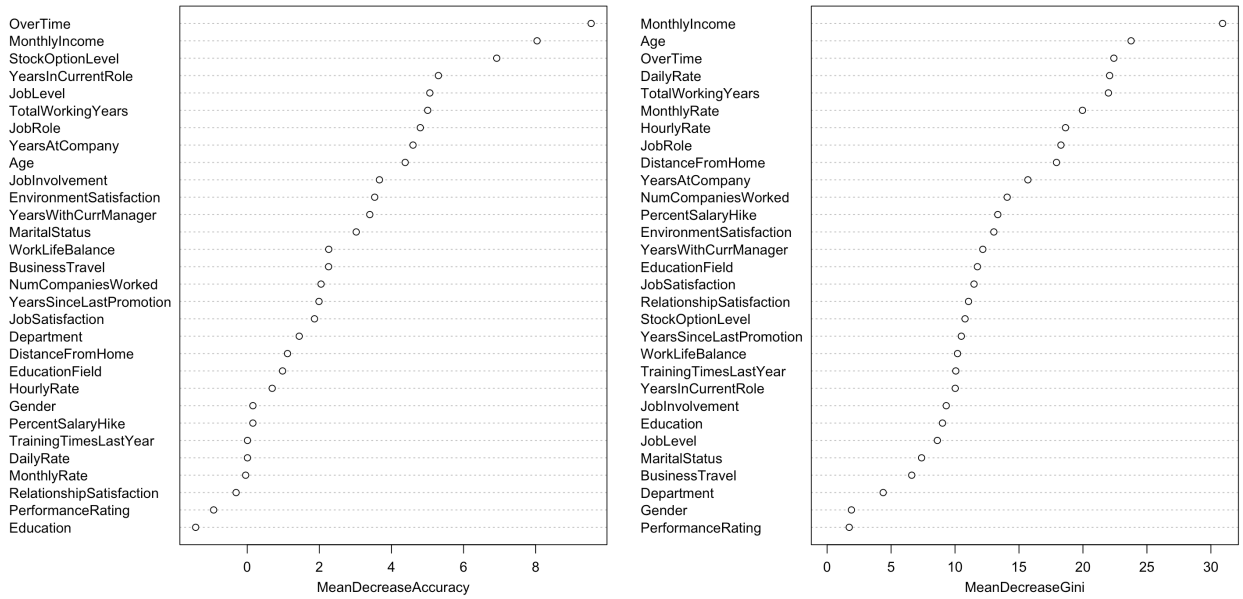


Figure 2.5 Variable Importance Plots

The above plot shows all the predictors on the Y-axis. It is ranked from the top to bottom as most important to least important. The most important variables are on the top and their corresponding importance is represented using a dot on the X-axis. It gives both mean decrease in accuracy and mean decrease in Gini. The mean decrease in accuracy is calculated during out of bag validation. If the accuracy of the model decreases due to the removal of that predictor variable, then the variable is said to be more important. Therefore, larger the decrease more the variable is important. In a similar way mean decrease in Gini is also calculated. Here there is a high difference in mean decrease accuracy and mean decrease gini between first two highly ranked variables. As a result, we can see these variables to see the best split in more branches.

Split the available data into the training dataset and testing dataset. This is done for every analysis irrespective of the model. This is done to check the predictive capability of the model. Here the testing data set is removed from the entire dataset because in this way we are not allowing bias in the model which means we make sure that that the trained model doesn't know anything about the testing data. As a result, the accuracy obtained from the model is much thrust worthy.

2.3 Single Tree Model:

Decision tree is implemented by specifying the split method as Gini value and complexity parameter to be the default value which is the penalty imposed on the misclassification error. We can also choose Entropy as the split method. Now the model is trained by using the training dataset. Since we don't have any missing values there is no need to specify the number of surrogate splits exclusively. Now we have to find the best complexity parameter value. Initially, when we train the model with default cp value we get the graph as given below:

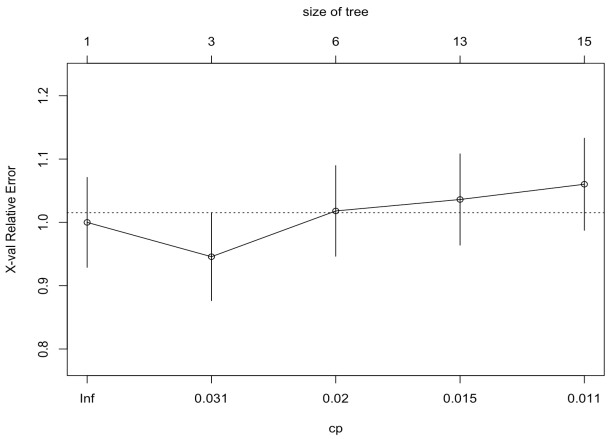


Figure 2.6 Initial CP value

The dotted line represents the minimum of relative error added with 1 standard deviation. Our goal is to keep the CP value less than the dotted line. We can see from the graph that CP value drops to 0.017 and again gets increased. So, we need to prune the model in order to get the tree only with minimum CP value. In order to do that we have to sum the minimum of Xerror and Xsd and find the difference with each value in the CP table and from that, we find the minimum CP value.

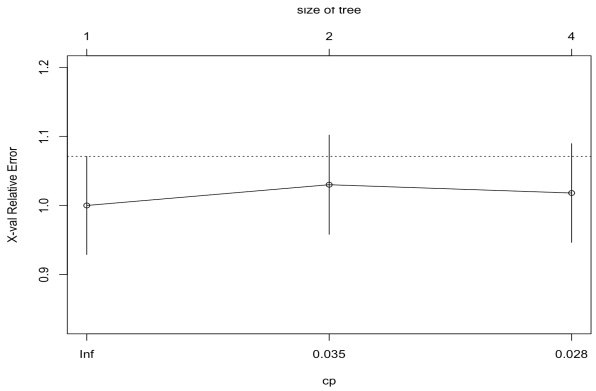


Figure 2.7 Minimum CP value

Now by using this minimum complexity parameter value and split criterion as "Gini", we will build the tree on the training dataset and the tree is shown below:

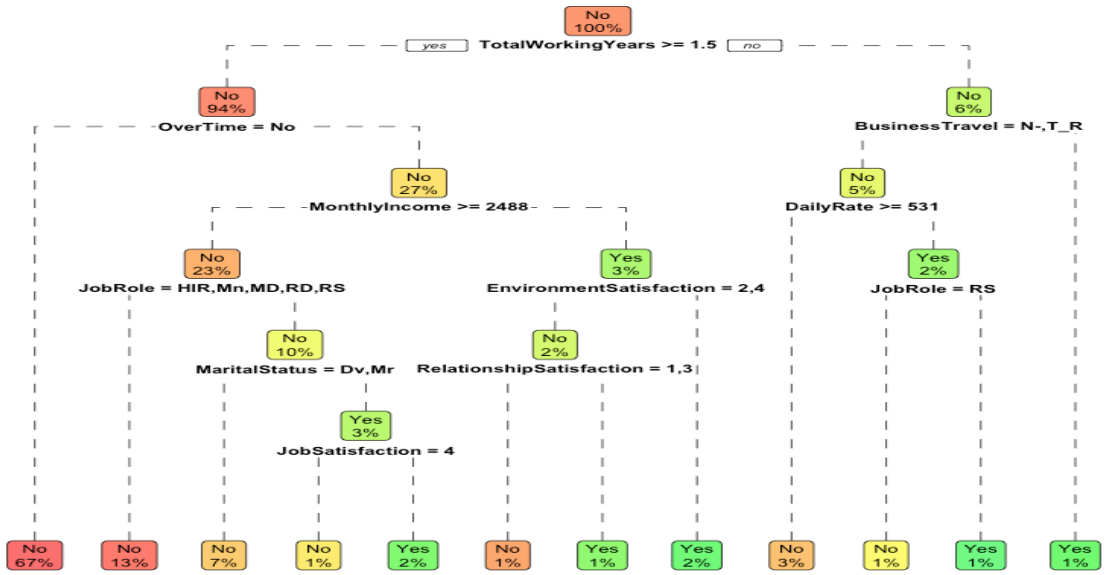


Figure 2.8 Decision Tree

Now we have constructed our model and by using this model we can form various conclusions and they are given below:

Conditions	Attrition
Total Working Years >= 1.5 and over time =No	No
Total Working Years >= 1.5 and Over time = Yes and Monthly Income < 248 and Environmental Satisfaction not in {medium, very high}	Yes
Total Working Years >= 1.5 and Over time = Yes and Monthly Income >2488 and Job Role in {HIR, MN, MD, RD, RS}	No
Total Working Years >= 1.5 and Over time = Yes and Monthly Income > 2488 and Job Role not in {HIR, MN, MD, RD, RS} and Martial Status in {Dv, Mr}	No
Total Working Years >= 1.5 and Over time = Yes and Monthly Income > 2488 and Job Role not in {HIR, MN, MD, RD, RS} and Martial Status not in {Dv, Mr} and Job Satisfaction is very High	No
Total Working Years >= 1.5 and Over time = Yes and Monthly Income > 2488 and Job Role not in {HIR, MN, MD, RD, RS} and Martial Status not in {Dv, Mr} and Job Satisfaction is not very High	Yes
Total Working Years >= 1.5 and Over time = Yes and Monthly Income < 2488 and Environmental Satisfaction in {medium, very high} and Relationship Satisfaction in {low, high}	No
Total Working Years >= 1.5 and Over time = Yes and Monthly Income < 2488 and Environmental Satisfaction in {medium, very high} and Relationship Satisfaction in {low, high}	Yes
Total Working Years < 1.5 and Business Travel Frequently	Yes
Total Working Years < 1.5 and Business Travel Rarely and Daily Rate > 531	No
Total Working Years < 1.5 and Business Travel Rarely and Daily Rate < 531 and Job Role is RS	No
Total Working Years < 1.5 and Business Travel Rarely and Daily Rate < 531 and Job Role is not RS	Yes

Now we can use this model to do prediction on our test dataset. After Prediction is done, we can examine the predicted probabilities using box plots as shown below.

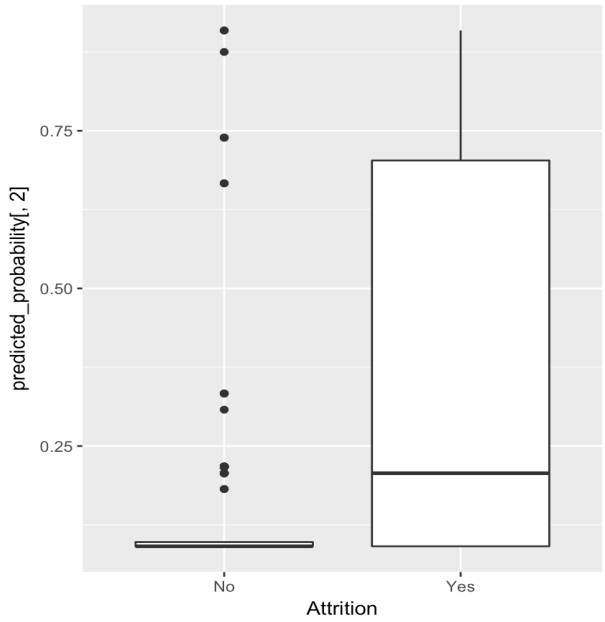


Figure 2.9 Predicted Probability

From the box plot, we can infer that about 50% of the employees wanted to stay back in their present company because, in the plot, the median value itself has a probability value of less than 0.25. The region below the median has low variance so our classification accuracy will be good. But on the other hand, the region above the median has high variance, this indicates that the 25% of the people may or may not leave the company. This is because the predicted probability for the 50-75% quarter has a spread from 0.24 to 0.70 and in this region, about 60% of the employees have predicted probabilities of value less than 0.50. Therefore, we can be sure that about 60% of the employees will not leave the company and remaining 40% have good chance quitting the company.

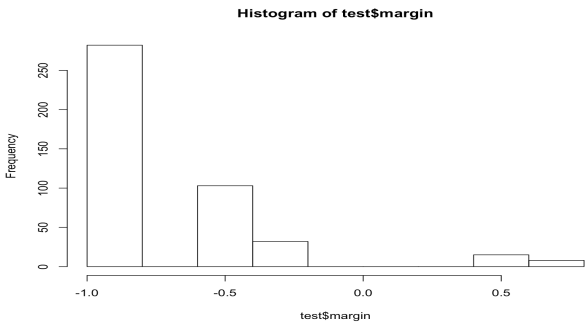


Figure 2.10 Histogram of Margin

From this Histogram, we can find the margin of the classifier. Here margin is large and this helps in proper classification of the target variable. The margin plot also confirms that most of the employees will be classified into the attrition class of "No" and only a few employees have classified into the attrition "Yes" category.

We can evaluate the models in depth using error rates, accuracy and their plots are given below:

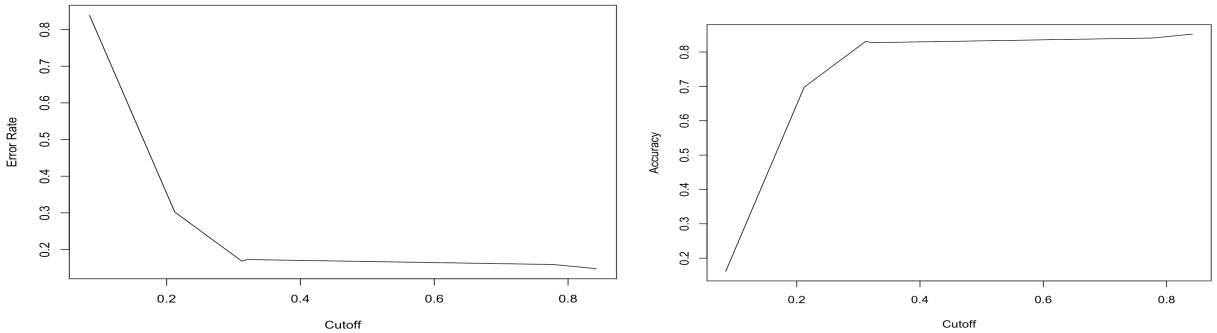


Figure 2.11 Error Rate vs Cut Off & Error Rate vs Accuracy

In the above plot between Error Rate and Cutoff, we can see that Error rate decreases significantly when the cutoff is 0.2. This indicates that there is no need to increase the cutoff beyond 0.4 since the error rate has got saturated after that. Similarly, In the above plot between Accuracy and Cutoff, the accuracy increases abruptly till 0.3 and after that it gets saturated.

Now we can also use ROC curve which is the tradeoff between the sensitivity (True Positive Rate) and 1-specifity (False positive Rate) to evaluate the model and the plot is shown below:

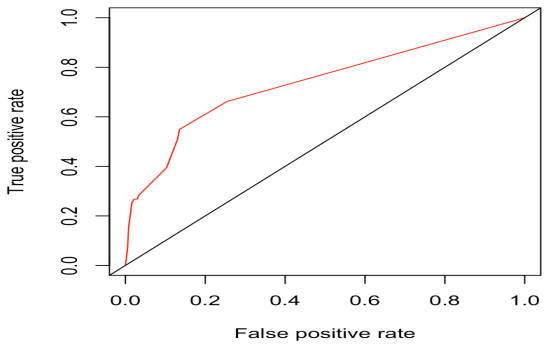


Figure 2.12 ROC for Decision Tree

From the above plot, we can say that the accuracy of the classifier is moderate (above 50%) as the line doesn't get close to the upper left corner where the ideal state exists (100% sensitivity and 100% specificity). The area under curve(AUC) value is **0.7368984** which shows the probability that it will rank the randomly chosen cases to be positive cases higher than the negative cases.

2.4 Ensemble Method 1 - Bagging:

Bagging is an ensemble technique where several trees are created using the bootstrapped samples. Bootstrapping is a technique in which random samples are created from an available sample with replacement. For every sample created decision trees are formed and they are tested with out of bag samples and the misclassification rate is called as out of bag error rate. Bagging is done for reducing the variance between the trees.

In R Bagging is done to the training dataset using ipred package since we use data frame we are going to bagging function in that package. This function has the following additional parameters:

- nbagg - This parameter is used to give the information about the number of bootstrap replications required for building the model and this is chosen to be initially 50
- coob - This is used to tell whether we need to compute the out of bag error or misclassification error while building the model and this property is set TRUE.
- control - With the help of this parameter we can tune/control the building process of an individual tree. Using this the tree is constructed using Gini as Split Criterion, and also providing best complexity parameter value. This CP value is used to prune the fully grown trees. Since there are no missing values, the number of surrogate splits were not mentioned. The maximum depth parameter was not given because we are building a fully grown tree and then prune it back using our best CP value.
- ns - Number of samples to be selected from the training sample. By default it selects the same number of samples but with replacement.

This trained model is applied to the test dataset to predict the attrition dataset. If we form a confusion matrix for Predicted vs Actual, then the result is obtained as given below:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 367  61
         1   2  10

               Accuracy : 0.8568
                 95% CI : (0.8206, 0.8882)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 0.1657

                  Kappa : 0.2038
 Mcnemar's Test P-Value : 0.0000000000002725

            Sensitivity : 0.14085
            Specificity : 0.99458
         Pos Pred Value : 0.83333
         Neg Pred Value : 0.85748
             Prevalence : 0.16136
         Detection Rate : 0.02273
   Detection Prevalence : 0.02727
      Balanced Accuracy : 0.56771

       'Positive' Class : 1
```

From the above-mentioned confusion matrix, we can say that the accuracy obtained is 85%. This accuracy is not the exact accuracy because our data is not balanced which means we don't have an equal number of YES and NO cases in the dataset So the confusion matrix creates a **Balanced accuracy which is 56 %.** This is calculated by averaging the Sensitivity and Specificity. The Sensitivity (True Positive rate) where it has predicted positive cases exactly to positive is 14% and the Specificity (True Negative Rate) where it has predicted negative cases exactly to negative is 99%. Here 61 cases are actually quitting the company whereas they are predicted as not quitting which is False Positive in confusion matrix. This must be considered seriously in the business point of view because they want to retain the good employees from quitting the company. The other case where the employees actually not leaving the company but they are predicted as quitting the company which is the False Negative rate in confusion matrix. This case is not so serious because you are not going to lose any good employees.

Now we can also use ROC curve to examine the model in order to make sure that the accuracy calculated from confusion matrix is correct. The model is examined using the ROC Curve and it is given below:

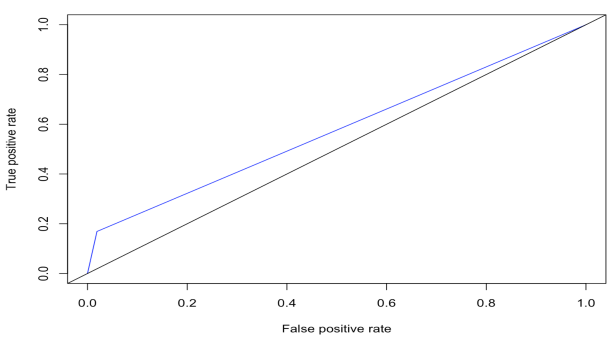


Figure 2.13 ROC for Bagging

In this plot, the ROC curve lies on the upper half of the roc space which indicates that the accuracy is above 50% and it is also shown by the area under curve value (AUC) which is 0.5650219. Hence the accuracy calculated from Confusion Matrix and ROC curve is almost same.

2.5 Ensemble Method 2 - Random Forest:

Random Forest is an ensemble technique where we make use of the bagging technique which reduces the variance among the trees and here we reduce correlation between the trees by reducing the number of variables available during each split by √m where m is the number of predictor variables. The model is built by using the training dataset.

For building the model, we use randomForest function from "randomForest" package in R. This function comes with default parameters that are mentioned below:

- ntree – This parameter is used to specify the number of trees to be grown. This is set to 1000 trees so that we can calculate the error rate for every 100 trees till 1000 trees.
- mtry – This parameter is used to specify the number of predictor variables available at each split. By default this is set to √p where p is the total number of available predictors.
- Importance – This parameter is used to make the function to evaluate the importance of predictor variables.

Now the model is built and this is used for the predicting the attrition of the employees using the test dataset. The output from all the trees are combined using voting for predicting the attrition. This model can be evaluated using Confusion Matrix and ROC curve.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 369  62
         1   0   9

               Accuracy : 0.8591
                 95% CI : (0.823, 0.8902)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 0.1345

                  Kappa : 0.1958
 Mcnemar's Test P-Value : 0.000000000000009408

            Sensitivity : 0.12676
            Specificity : 1.00000
         Pos Pred Value : 1.00000
         Neg Pred Value : 0.85615
             Prevalence : 0.16136
         Detection Rate : 0.02045
   Detection Prevalence : 0.02045
      Balanced Accuracy : 0.56338

       'Positive' Class : 1
```

From the confusion matrix, we can say that the balanced accuracy obtained is 56%. The Sensitivity (True Positive rate) is 12% and the Specificity (True Negative Rate) is 100%. It has correctly predicted all the negative cases to negative class. Now we will use ROC curve to reconfirm our accuracy obtained using confusion matrix and plot is given below:

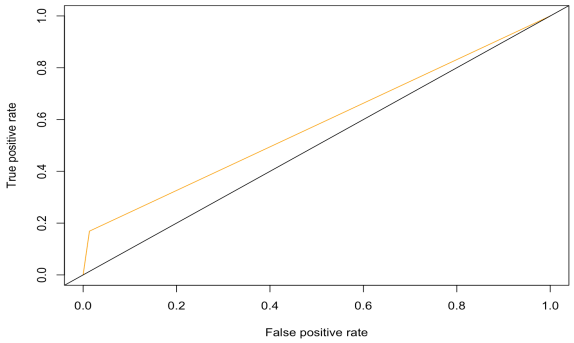


Figure 2.14 ROC for Random Forest

In this plot, the roc curve lies on the upper half of the roc space which indicates that the accuracy is above 50% and it is also shown by the area under curve value (AUC) which is 0.577732. Hence the accuracy obtained is similar in both the evaluation models.

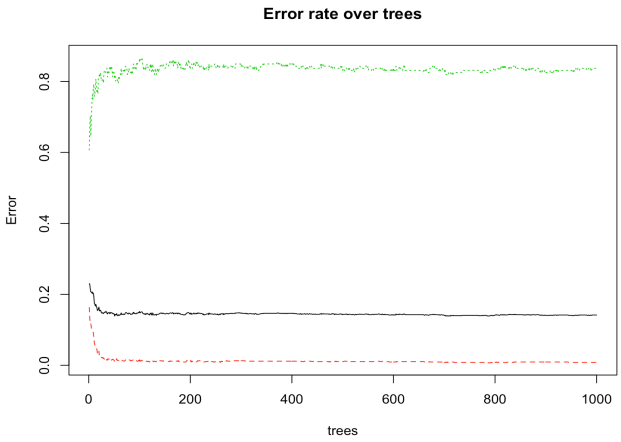


Figure 2.15 Error Rate over Trees

In this plot, we can get error rate over the trees. The black line represents out of bag error rate, red represents error rate for attrition = No and green represents the error rate for attrition = Yes. We can say that the error rate for both the classes becomes almost constant after 200 trees. This implies that we could have trained the model using only 200 trees rather than 1000 trees. The error rate for the Yes class is high as compared to error rate of No class.

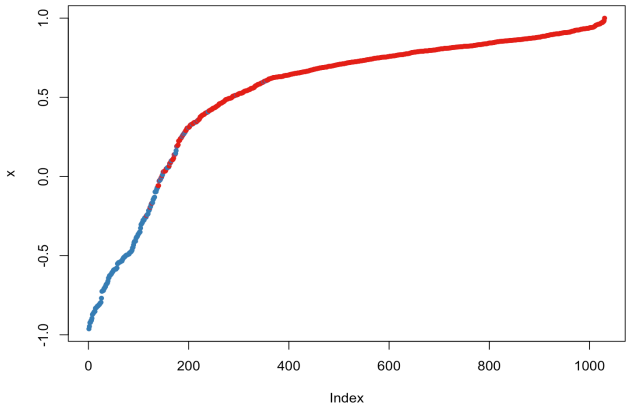


Figure 2.16 Margin of the Classifier

This figure gives information about the margin of the classifier. The margin tells the ratio of votes for the correct class for all the samples. 1 indicates that for one sample all the response values from all the samples pointed to the correct class and 0 indicates that some results were voted for right class and some for the wrong class. Less than 0 means most of them have been classified to the wrong class. The red color indicates that most of the Yes samples have been classified correctly and some of the No samples have been not classified correctly.

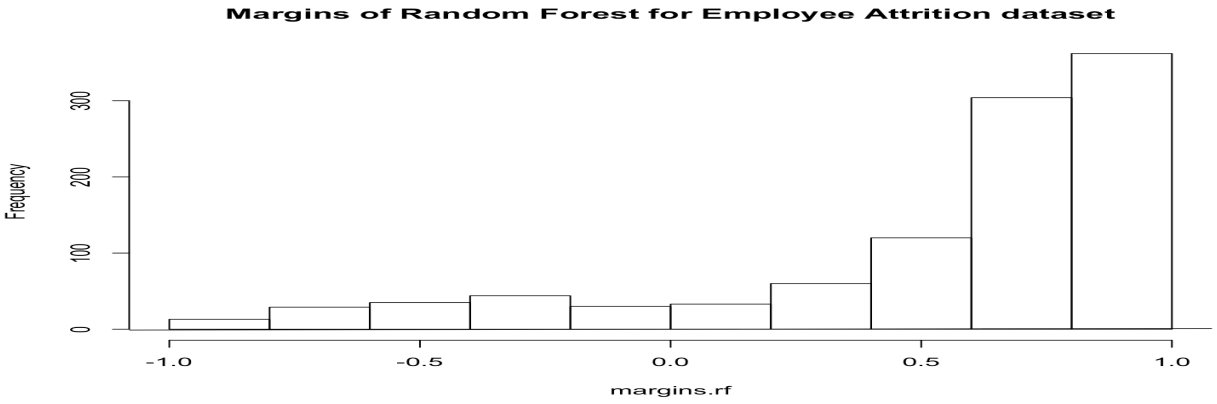


Figure 2.17 Histogram of Margin

The Histogram of margin shows that most of the results from the ensembles have been pointing towards Yes Attrition Class.

2.6 Comparison of Models:

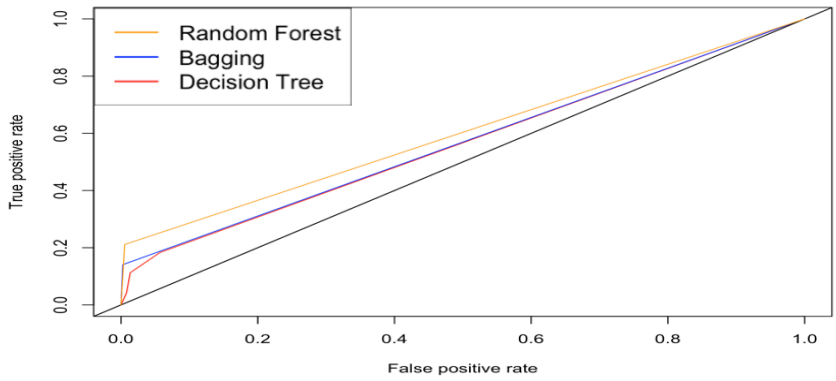


Figure 2.18 ROC for Bagging vs Random Forest vs Decision Tree

From the above figure, it is clear that the performance of random forest model is much better when compared to the other two models. The accuracy can be ranked in the corresponding order **Random Forest < Bagging < Trees**. This is evident that the ensemble models help in the better classification than the trees. Thus, If the company wants to predict the attrition rate of the employees then, they can use Random Forest model and know the employees who have high chances of quitting the company. So that they can take preventive measures to retain good employees.

3. Source Code

3.1 Fisheries:

```
########################################### Libraries ###################################################

library(VIM)
library(dplyr)
library(ggplot2)
library(janitor)
library(caret)
library(rpart)
library(tidyverse)
library(ROCR)
library(SDMTools)
library(boot)
library(ipred)
library(randomForest)
library(pdp)
library(e1071)
library(readxl)
library(corrplot)
library(mlbench)
library(caret)
library(lucr)
library(partykit)
library(gbm)
library(MLmetrics)
library(caretEnsemble)
library(C50)
########################################### Input Data ###################################################

fisheries_dataset<- read.csv("/Users/sethuram/Desktop/Data
Analytics/Lab/economic.csv",fileEncoding="latin1")
fisheries_dataset <-fisheries_dataset %>%
clean_names()%>%
remove_empty_rows() %>%
remove_empty_cols()
glimpse(fisheries_dataset)
ncol(fisheries_dataset)
########################################### Data Cleaning ###################################################

fisheries_dataset$size_category <- as.character(fisheries_dataset$size_category)
filter_date <-function(x){
if(x == "12-18"|x == "10-12" ){
x<- "NA"
#return(x)}
}else{
x<-x
}
}
fisheries_dataset$size_category <-
as.factor((unlist(lapply(fisheries_dataset$size_category,filter_date))))
remove_comma<-function(x){
x<-as.character(x)
x<-sub(",","", x)
}
fisheries_dataset$capacity_gt <-
as.double((unlist(lapply(fisheries_dataset$capacity_gt,remove_comma))))
fisheries_dataset$total_number_of_vessels_in_segment <-
as.double((unlist(lapply(fisheries_dataset$total_number_of_vessels_in_segment,remove_comma))))
fisheries_dataset$total_engine_power_kw_for_segment_size <-
as.double((unlist(lapply(fisheries_dataset$total_engine_power_kw_for_segment_size,remove_comma))))
fisheries_dataset$total_capacity_gt_for_segment_size <-
as.double((unlist(lapply(fisheries_dataset$total_capacity_gt_for_segment_size,remove_comma))))
for(i in 17:39){
fisheries_dataset[,i]<- sub("\u0080","", fisheries_dataset[,i])
fisheries_dataset[,i] <- sub(",","", fisheries_dataset[,i])
fisheries_dataset[,i] <- as.integer(fisheries_dataset[,i])
```

```
}
#nrow(fisheries_dataset)
fisheries_dataset$year <- as.factor(fisheries_dataset$year)
fisheries_dataset$capacity_gt <- as.double(fisheries_dataset$capacity_gt )
fisheries_dataset$total_number_of_vessels_in_segment<-
as.double(fisheries_dataset$total_number_of_vessels_in_segment )
fisheries_dataset$total_engine_power_kw_for_segment_size<-
as.double(fisheries_dataset$total_engine_power_kw_for_segment_size )
fisheries_dataset$total_capacity_gt_for_segment_size<-
as.double(fisheries_dataset$total_capacity_gt_for_segment_size)
########################################### Missing Values ########################################################

oaggr <-aggr(fisheries_dataset)
summary(oaggr)
# Findings:- missing value percentages are high for many columns
# setting missing value column eliminationa cutoff to > 40 %
# Eliminating Following Fields - non_fishing_income, ice, dues_levies,
#
loan_interest,legal_fees,depreciation,sundry_receipts
fisheries_dataset$reference_number<- NULL
fisheries_dataset$non_fishing_income <- NULL
fisheries_dataset$ice<- NULL
fisheries_dataset$dues_levies<- NULL
fisheries_dataset$loan_interest<- NULL
fisheries_dataset$legal_fees<- NULL
fisheries_dataset$depreciation<- NULL
fisheries_dataset$sundry_receipts<- NULL
fisheries_dataset$gross_profit<- NULL
########################################### Missing Value Imputations
#########################################################

column_names<-names(fisheries_dataset)
for(i in 1:length(column_names)){
fisheries_dataset[,i]<- na.roughfix(fisheries_dataset[,i])
}
########################################### Variable Analysis
#########################################################

summary(fisheries_dataset$total_income)
fisheries_dataset$total_income
ggplot(fisheries_dataset,aes(x = "", y = total_income )) + geom_boxplot()
ggplot(fisheries_dataset,aes(x = total_income, y = net_profit_loss )) + geom_point()
histogram(fisheries_dataset$total_income,col="darkred")
summary(fisheries_dataset$total_jobs)
fisheries_dataset$total_jobs
ggplot(fisheries_dataset,aes(x = "", y = total_jobs )) + geom_boxplot()
ggplot(fisheries_dataset,aes(x = total_jobs, y = net_profit_loss )) + geom_point()
histogram(fisheries_dataset$total_jobs,col="darkcyan")
plot(fisheries_dataset$total_jobs,fisheries_dataset$net_profit_loss)
abline(a=0,b=1 ,add = TRUE)
summary(fisheries_dataset$size_category)
summary(fisheries_dataset$net_profit_loss)
ggplot(fisheries_dataset,aes(x = size_category, y = "" )) + geom_boxplot()
ggplot(fisheries_dataset,aes(x = size_category, y = net_profit_loss )) + geom_boxplot()
histogram(fisheries_dataset$size_category,col="darkblue")
########################################### Data Examination
#########################################################

reuired_columns = c()
for (i in 1:length(column_names)){
print(class(fisheries_dataset[[i]]))
if(class(fisheries_dataset[[i]]) != "factor"){
reuired_columns[i] <- column_names[i]
}
}
reuired_columns <- na.omit(reuired_columns)
fisheries_dataset_subset <- select(fisheries_dataset, reuired_columns)
cor(fisheries_dataset_subset)
as.matrix.data.frame(fisheries_dataset_subset)
corrplot(cor(fisheries_dataset_subset),method = "number",type = "lower",tl.cex =0.6, tl.col =
"black",cl.cex = 0.6)
```

```
#--------------feature Selection--
set.seed(7)
# find attributes that are highly corrected (ideally >0.75)
highly_correlated <- findCorrelation(cor(fisheries_dataset_subset), cutoff=0.5)
# print indexes of highly correlated attributes
print(highly_correlated)
names(fisheries_dataset_subset[,highly_correlated])
#--------------Rank Features By Importance--------------
rf=randomForest(net_profit_loss~.,data=fisheries_dataset,ntree=100,importance=TRUE)
importance(rf)
varImpPlot(rf)
############################################## Train Test ##########################################################

set.seed(7)
trainIndex = createDataPartition(fisheries_dataset$net_profit_loss, p=0.7, list=FALSE,times=1)
train = fisheries_dataset[trainIndex,]
bag_train = fisheries_dataset[trainIndex,]
rf_train = fisheries_dataset[trainIndex,]
test = fisheries_dataset[-trainIndex,]
bag_test = fisheries_dataset[-trainIndex,]
rf_test = fisheries_dataset[-trainIndex,]
############################################## Single Tree Model
#########################################################

tree = rpart(net_profit_loss ~ .,data = train ,control=rpart.control(split='Gini'))
plot(as.simpleparty(as.party(tree)),gp = gpar(fontsize = 8))
#rpart.plot::rpart.plot(tree,tweak =1 , extra=100, branch.lty=2, box.palette="RdYlGn",faclen =
2,round = 1,compress = TRUE)
# Plotting Complexity Parameter for choosing the optimum value
printcp(tree)
plotcp(tree)
# Complexity Parameter Selection
min_xerror<-which.min(tree$cptable[,"xerror"])
mininum_sd<-which.min(abs(tree$cptable[,"xerror"] -
(tree$cptable[min_xerror,"xerror"]+tree$cptable[min_xerror,"xstd"])))
best_cp<-tree$cptable[mininum_sd,"CP"]
# Pruning the tree
new_tree = prune(tree,best_cp)
new_tree = rpart(net_profit_loss ~ .,data = train ,control=rpart.control(split='Gini',cp =best_cp
))
printcp(new_tree)
plotcp(new_tree)
#Training the tree using best CP
plot(as.simpleparty(as.party(new_tree)),gp = gpar(fontsize = 8))
test$predicted_probability<-predict(new_tree,test)
summary(test$predicted_probability)
# Evaluating the model
str(tree)
RMSE(test$net_profit_loss,test$predicted_probability)
glimpse(train)
#-------------------------------------------------------------------------------
glimpse(train)
attrition_bag = gbm(net_profit_loss ~.,data=bag_train,distribution = "gaussian",n.trees = 10000 )
n.trees = seq(from=100 ,to=10000, by=100)
bag_test$predicted<-predict(attrition_bag,bag_test,n.trees = n.trees)
bag_test$rf_predicted <- predict(rf,bag_test,n.trees = n.trees)
# Variable importance------------------------------------------------>
summary(attrition_bag)
#Plot of Response variable with lstat variable
plot(attrition_bag,i="repairs_maintenance")
plot(attrition_bag,i="total_fixed_costs")
#Calculating The Mean squared Test Error
test.error<-with(bag_test,apply( (predicted-net_profit_loss)^2,2,mean))
head(test.error) #contains the Mean squared test error for each of the 100 trees averaged
#Plotting the test error vs number of trees
options(scipen=100000)
plot(n.trees , test.error , pch=19,col="blue",xlab="Number of Trees",ylab="Test Error", main =
"Perfomance of Boosting on Test Set")
RMSE(bag_test$predicted,bag_test$net_profit_loss)
#----------------------------------------STACKING-------------------------------------------------
# Example of Stacking algorithms
```

```
# create submodels
control <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE,
classProbs=TRUE)
algorithmList <- c( 'rpart', 'lm', 'svmRadial')
#set.seed(seed)
models <- caretList(net_profit_loss~., data=train, trControl=control, methodList=algorithmList)
results <- resamples(models)
xyplot(resamples(models))
modelCor(resamples(models))
summary(results)
dotplot(results)
modelCor(results)
splom(results)
# stack using glm
stackControl <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE,
classProbs=TRUE)
stack.lm <- caretStack(models, method="lm", metric=c(RMSE,R2), trControl=stackControl)
print(stack.lm)
plot.caretStack(stack.lm)
```

3.2 Employee Attrition:

```
############################################ Libraries ###########################################################

library(VIM)
library(dplyr)
library(ggplot2)
library(janitor)
library(caret)
library(rpart)
library(tidyverse)
library(ROCR)
library(SDMTools)
library(boot)
library(ipred)
library(randomForest)
library(pdp)
library(e1071)
#------------------
library(readxl)
library(corrplot)
library(mlbench)
library(caret)
############################################ Input Data ##########################################################

employee_attrition_data <- read_excel("/Users/sethuram/Desktop/Data
Analytics/Lab/Employeeattrition.xlsx", sheet = 1)
# employee_attrition_data <-employee_attrition_data %>%
# clean_names()%>%
# remove_empty_rows() %>%
# remove_empty_cols()
summary(employee_attrition_data)
employee_attrition_data$Attrition <- as.factor(employee_attrition_data$Attrition)
employee_attrition_data$Education <- as.factor(employee_attrition_data$Education)
employee_attrition_data$OverTime <- as.factor(employee_attrition_data$OverTime)
employee_attrition_data$Over18 <- as.factor(employee_attrition_data$Over18)
employee_attrition_data$Gender <- as.factor(employee_attrition_data$Gender)
employee_attrition_data$EducationField <-
as.factor(employee_attrition_data$EducationField)
employee_attrition_data$Department <-
as.factor(employee_attrition_data$Department)
employee_attrition_data$BusinessTravel <-
as.factor(employee_attrition_data$BusinessTravel)
employee_attrition_data$JobRole <- as.factor(employee_attrition_data$JobRole)
employee_attrition_data$MaritalStatus <-
as.factor(employee_attrition_data$MaritalStatus)
employee_attrition_data$WorkLifeBalance <-
```

```
as.factor(employee_attrition_data$WorkLifeBalance)
employee_attrition_data$RelationshipSatisfaction <-
as.factor(employee_attrition_data$RelationshipSatisfaction)
employee_attrition_data$PerformanceRating <-
as.factor(employee_attrition_data$PerformanceRating)
employee_attrition_data$JobSatisfaction <-
as.factor(employee_attrition_data$JobSatisfaction)
employee_attrition_data$JobInvolvement <-
as.factor(employee_attrition_data$JobInvolvement)
employee_attrition_data$EnvironmentSatisfaction <-
as.factor(employee_attrition_data$EnvironmentSatisfaction)
employee_attrition_data$EmployeeNumber<-NULL
employee_attrition_data$StandardHours<-NULL
employee_attrition_data$EmployeeCount<-NULL
employee_attrition_data$Over18<-NULL
column_names<-names(employee_attrition_data)
########################################### Missing Values ###################################################

oaggr <-aggr(employee_attrition_data)
summary(oaggr)
# Result: - No missing values
########################################### Variable Analysis
#####################################################

summary(employee_attrition_data)
employee_attrition_data$MonthlyIncome
ggplot(employee_attrition_data,aes(x = "", y = MonthlyIncome )) + geom_boxplot()
ggplot(employee_attrition_data,aes(x = attrition , y = MonthlyIncome, color =
Attrition )) + geom_boxplot()
histogram(employee_attrition_data$MonthlyIncome,col="darkred")
summary(employee_attrition_data$yearswithcurrmanager)
ggplot(employee_attrition_data,aes(x = "", y = yearswithcurrmanager )) +
geom_boxplot()
ggplot(employee_attrition_data,aes(y = yearswithcurrmanager, x = Attrition ,color
= Attrition )) + geom_boxplot()
histogram(employee_attrition_data$yearswithcurrmanager,col="darkcyan")
plot(employee_attrition_data$yearswithcurrmanager,employee_attrition_data$Attrition)
abline(a=0,b=1 ,add = TRUE)
summary(employee_attrition_data$size_category)
summary(employee_attrition_data$net_profit_loss)
ggplot(employee_attrition_data,aes(x = size_category, y = "" )) + geom_boxplot()
ggplot(employee_attrition_data,aes(x = JobInvolvement, y = Attrition )) +
geom_count()
histogram(employee_attrition_data$size_category,col="darkblue")
table(employee_attrition_data$JobSatisfaction,employee_attrition_data$Attrition)

########################################### Data Examination
#####################################################

reuired_columns = c()
for (i in 1:length(column_names)){
if(typeof(employee_attrition_data[[i]]) == "double"){
reuired_columns[i] <- column_names[i]
}
}
reuired_columns <- na.omit(reuired_columns)
employee_attrition_data_subset <- select(employee_attrition_data, reuired_columns)
glimpse(employee_attrition_data)
cor(employee_attrition_data_subset)
as.matrix.data.frame(employee_attrition_data_subset)
cex.before <- par("cex")
par(cex = 0.5)
corrplot(cor(employee_attrition_data_subset),method = "number",
tl.col = "black",type = "lower",tl.cex = 1/par("cex"),
cl.cex = 1/par("cex"),tl.srt=45)
par(cex = cex.before)
#--------------feature Selection--
set.seed(7)
# find attributes that are highly corrected (ideally >0.75)
highly_correlated <- findCorrelation(cor(employee_attrition_data_subset),
cutoff=0.5)
```

```
# print indexes of highly correlated attributes
print(highly_correlated)
employee_attrition_data_subset[,highly_correlated]
#--------------Rank Features By Importance--------------
rf=randomForest(Attrition~.,data=employee_attrition_data,ntree=100,importance=TRUE)
importance(rf)
varImpPlot(rf)
############################################ Train Test ######################################################

trainIndex = createDataPartition(employee_attrition_data$Attrition, p=0.7,
list=FALSE,times=1)
train = employee_attrition_data[trainIndex,]
bag_train = employee_attrition_data[trainIndex,]
rf_train = employee_attrition_data[trainIndex,]
test = employee_attrition_data[-trainIndex,]
bag_test = employee_attrition_data[-trainIndex,]
rf_test = employee_attrition_data[-trainIndex,]
table(train$Attrition)
table(test$Attrition)
############################################ Single Tree Model
#########################################################

tree = rpart(Attrition ~ .,data = train )
rpart.plot::rpart.plot(tree,tweak =1 , extra=100, branch.lty=2,
box.palette="RdYlGn",faclen = 2,round = 1,compress = TRUE)
# We dont have any missing values so no need to specify explicitly for surrogate
splits
# Plotting Complexity Parameter for choosing the optimum value
printcp(tree)
plotcp(tree)
# Complexity Parameter Selection
min_xerror<-which.min(tree$cptable[,"xerror"])
mininum_sd<-which.min(abs(tree$cptable[,"xerror"] -
(tree$cptable[min_xerror,"xerror"]+tree$cptable[min_xerror,"xstd"])))
best_cp<-tree$cptable[mininum_sd,"CP"]
#Training the tree using best CP
set.seed(7)
new_tree = rpart(Attrition ~ .,data = train ,control=rpart.control(split='Gini',cp
= best_cp ))
printcp(new_tree)
plotcp(new_tree)
rpart.plot::rpart.plot(new_tree,tweak =1 , extra=100, branch.lty=2,
box.palette="RdYlGn",faclen = 2,round = 1,compress = TRUE)
test$predicted_probability<-predict(new_tree,test)
summary(test$predicted_probability[,2])
# Evaluating the model
# Box Plots #
ggplot(data=test,aes(x=Attrition,y=predicted_probability[,1]))+geom_boxplot()
ggplot(data=test,aes(x=Attrition,y=predicted_probability[,2]))+geom_boxplot()
#calculating the margins
train$Attrition<-ifelse(train$Attrition=="No",-1,1)
test$Attrition<-ifelse(test$Attrition=="No",-1,1)
#--------margin calculation--------------------
margin_calc <- function(predict_e,predict_p, target){
if(target == -1){
(predict_e-predict_p)*target
}else{
(predict_p-predict_e)*target
}
}
test$margin<-apply(test[,c('predicted_probability','Attrition')], 1, function(x)
margin_calc(x[1],x[2],x[3]))
hist(test$margin)
#ROC - Curves
predics<-prediction(test$predicted_probability[,2],test$Attrition)
roc_curve_tree<-performance(predics, "tpr","fpr")
abline(a=0,b=1)
area_under_curve_tree<-performance(predics,"auc")@y.values
#Error Rate
perf=performance(predics, "err")
plot(perf)
```

```
plot(performance(predics, "tpr"))
plot(performance(predics, "fpr"))
plot(performance(predics, "fnr"))
plot(performance(predics, "tnr"))
plot(performance(predics, "acc"))
plot(roc_curve_tree,col="red",add = TRUE)
make_prediction <- function(probability_yes, cut_off) {
prediction <- vector(length=length(probability_yes))
for(i in 1:length(prediction)) {
if(probability_yes[i] >cut_off) {
prediction[i] <- "Yes"
} else {
prediction[i] <- "No"
}
}
factor(prediction, levels=c("No", "Yes"))
}
predicted <- make_prediction(test$predicted_probability[,2], 0.7)
table(predicted, test$Attrition, dnn=c("Predicted", "Actual"))
########################################### Bagging ###########################################################

glimpse(train)
attrition_bag <- bagging(Attrition
~.,data=bag_train,control=rpart.control(cp=best_cp),nbagg=50,coob=TRUE )
bag_test$predicted<-predict(attrition_bag,bag_test)
bag_test$predicted<-ifelse(bag_test$predicted=="No",0,1)
bag_test$Attrition<-ifelse(bag_test$Attrition=="No",0,1)
predicts<-prediction(bag_test$predicted,bag_test$Attrition)
confusionMatrix(data=bag_test$predicted,
reference=bag_test$Attrition,
positive='1')
str(predicts)
table()
bag_perf=performance(predicts,"tpr","fpr")
plot(bag_perf,col="blue")
plot(bag_perf,col="blue",add = TRUE)
summary(attrition_bag)
abline(a=0,b=1)
area_under_curve_bag<-performance(predicts,"auc")@y.values
########################################### Random Forest
#########################################################

rf=randomForest(Attrition~.,data=rf_train,ntree=1000,importance=TRUE)
rf_test$predicted<-predict(rf,rf_test)
rf_test$predicted<-ifelse(rf_test$predicted=="No",0,1)
rf_test$Attrition<-ifelse(rf_test$Attrition=="No",0,1)
predicts<-prediction(rf_test$predicted,rf_test$Attrition)
confusionMatrix(data=rf_test$predicted,
reference=rf_test$Attrition,
positive='1')
rf_perf=performance(predicts,"tpr","fpr")
plot(rf_perf,col="orange")
legend(x= "topleft", y=0.92, legend="Random Forest"
,col="orange", lty=1, cex=0.8)
legend(x= "topleft", y=0.92, legend=c("Random Forest", "Bagging", "Decision
Tree"),
col=c("orange", "blue", "red"), lty=1, cex=0.8)
abline(a=0,b=1)
area_under_curve_rf<-performance(predicts,"auc")@y.values
#plot(rf_perf,col="orange",add = TRUE)
#-------------Error plot and margin plots----------------------
plot(rf, main="Error rate over trees")
margins.rf=margin(rf,Attrition)
plot(margins.rf)
hist(margins.rf,main="Margins of Random Forest for Employee Attrition dataset")
boxplot(margins.rf ~employee_attrition_data$Attrition,main="Margins of Random
Forest for Employee Attrition dataset by class")
abline(a=0,b=1)
```