Fisheries data set

1-Introduction: -

1.1- Background and the purpose of the analysis: -

As per the 2017 annual economic report by Scientific, technical and economic committee for fisheries (STECF) a comprehensive overview of the latest information available on the structure and economic performance of the EU member states(MS) fishing fleets. It is not possible to pinpoint precisely the determining factor(s) behind the performance of the EU fishing fleet, it's trend and economic result. However, it is likely that several features together have contributed to the economic result of many commercially important stocks and to the overall economic development of the EU fleet. The main purpose of the analysis is to find the dependency of the Net Profit/Loss of the various vessel, on the given features.

1.2 - Structure of the Data sets.

The data is divided into two csv files. The first file, economic dataset depicts the Net Profit/Loss of various vessel based on their Vessel ID. The data is collected during the course of 8 years i.e. from 2008-2016. Overall there are 1432 observations with 40 variables. The main features of the data set are: -

1-Vessel ID
2-Year
3- Segment
4- Total Engine Power(kW) for Segment Size
5-Total Capacity (GT) or Segment Size
6-Total Jobs
7-FTE (Full time equivalent)
8-Total Income
9-Total Variable Costs
10-Total Fixed Costs
11-Total costs
12-Gross profit
13-Depreciation
14-Sundry receipts
15-Net Profit/Loss

The importance and the analysis of the various variables will be covered in the later part of the report. Net Profit/Loss is initially considered the target variables and rest are the grouped as independent feature.

The second data set is the fisheries dataset which gives the details of the economic value of various species of the fishes linked with the vessel ID, which is the primary key between the two dataset. The data is collected from year 2004 to 2016. The description of this data set can be found in the separate file. The file links the fish species with the the ID for e.g.

Abalone	S1
Albacore	S2
Anchovy	S3
Black Scabbard fish	S4
Blue Ling	S5
Blue whiting	S6
Blue mouth	S7
Boarfish	S8

The various columns of the data set are: -
1-V.ID (Vessel ID)
2-Year
3-Value.S1
4-Value S10
5-Value S100

The analysis of the variables is done in the later part of the report.

2- Examination of the Data and Initial Data reconfiguration: -

2.1- Data cleaning and reconfiguration of economic dataset: -

The data is not cleaned and contains a lot of NA values which have to be removed before analysis. The first task is to convert the factors into numeric which are done using the basic MS Excel formatting. The data also has some garbage values like 10 Dec in segment column which are corrected during the importing the file. It is necessary to import the dataset using read.csv2 function with utf-8 encoding instead of importing the dataset directly from the R-Studio. The financial feature of the dataset has a Euro symbol attached with the value in form U+20AC which can either be removed using the substitute function in R or with the MS Excel. In this research latter is used.
Note- If the factors are converted directly to numeric value using the *as.numeric()* function the value changes, thus it is first converted into character using *sub("," ,"",as.character())* function then converted to numeric.

2.2-Data cleaning and initial reconfiguration fisheries dataset: -

The fisheries dataset is clean and has no NA values and thus doesn't require any initial reconfiguration.

2.3- Merging the two dataset: -

The two data set are merged using the V.ID. as the primary key by running a python script attached in the appendix. All the further exploratory analysis is done on this merged file. There are 756 matched features in the final CSV file. After removing the features with more than 80% NA values the final CSV has 68 variables.

2.4- Handling NA values: -

There are total 756 NA values in 45 variables. As the data represent the financial NA values can be converted into 0's without hindering the analysis or down-sampling the data set.

3- Data Pre-processing and feature selection: -

3.1- Variable importance of initial data frame: -

During the initial analysis of the data it was found that when analysing the variable importance while predicting the Net Profit/Loss, the importance of Gross profit, Depreciation, Total income is very high which creates the bias in the model.

Feature	Importance
Capacity GT	0.158915
Capacity Index	0.158915
Depreciation	1.299835
Fishing Income	0.768044
Gross Profit	2.303017
Legal Fees	0.159930
Non Fishing Income	0.635315
Size Index	0.158915
Sundry receipts	0.300903
Total Fixed Costs	0.158915
Total Income	1.218557
Total Variable Costs	0.483077
Value S45	0.539889
vid	0.276566
Wages	0.176012

Table 1 Variable Importance before split

The reason for this bias is because if the dataset is analysed the Net Profit/Loss is linearly related to these variables.

Total Income=Fishing Income + Non-Fishing Income

Total variable income= Wages + Energy Costs + Repair and maintenance + Filters + Provisions + Ice + Dues & Levies + Sundry Variables Costs

Total fixed cost=Insurance + Loan interest + Accountancy + Legal Fees + Sundry fixed cost

Total cost = Total fixed cost + Total variable cost

Gross Profit = Total Income – Total cost

Net Profit/Loss = Gross Profit – Depreciation + Sundry Receipts

Thus if these variables even though being important will give a huge bias in the model and will not be good model.

3.2 Choosing Target Variable: -

By looking at the above relation, in this analysis instead of predicting the Net Profit/Loss I am predicting the Gross profit, Depreciation and Sundry receipts the dependency on non-financial features. If these three variable are predicted, then the Net Profit/Loss can be easily predicted.
The data set is thus divided into 3 different data sets with Gross Profit, Depreciation and Sundry Receipts.

3.3 Variable importance of new Datasets: -

3.3.1- Gross Profit: - To find the importance of the independent feature over the target value, function rpart was used of Caret package.

Feature	Importance
Capacity GT.	0.8461436
Capacity Index	0.7929698
FTE	0.4068596
Length	0.6507358
Segment	0.2060755
Total Engine Power kW for Segment Size	0.7278284
Total Jobs	0.3122556
value.S2	0.9124160
value.S45	0.5873163
value.S51	1.2699220
vid	0.4476998
Year	0.2761130

Table 2 Variable Importance of variables on Gross Profit

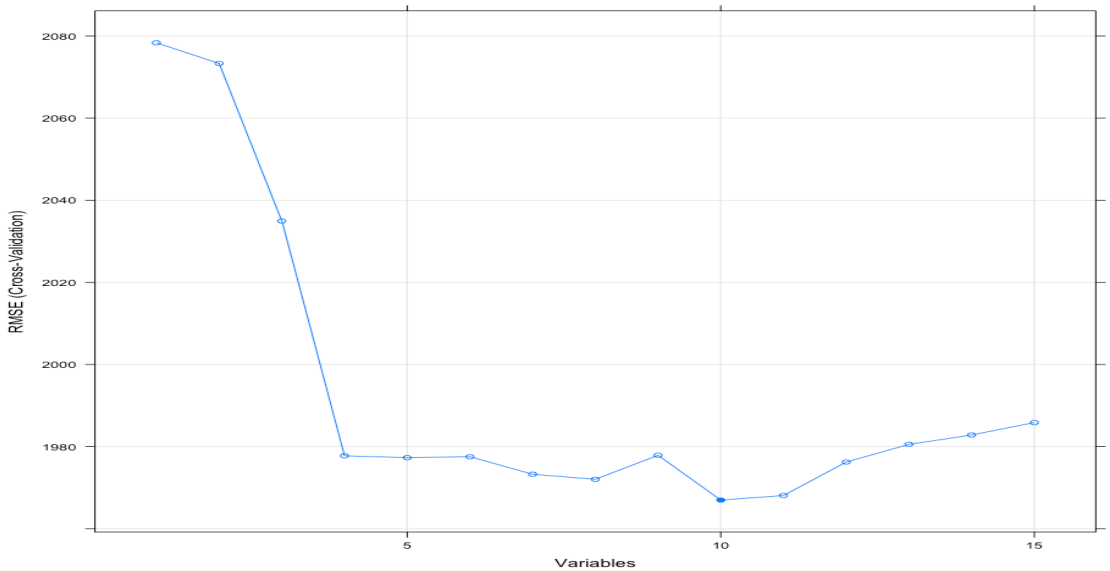

Fig 1 RMSE vs Number of variable

It is clear from the graph that the model RMSE is lowest with 10 variables. The variables chosen are value S2, Capacity GT and capacity index.

3.3.2- Depreciation: - Depreciation is a totally independent variable. To find the importance of the independent feature over the target value, function rpart was used of Caret package.

Feature	Importance
Capacity GT.	1.28054629
Capacity Index	1.27060112
FTE	0.18924672
Length	1.07623811
Segment	0.42294234
Size Index	0.57942894
Total Active Vessels in Segment	0.17355466
Total Engine Power kW for Segment Size	0.14857864
Total Jobs	0.24443831
value.S2	0.09560687
value.S45	0.90642396
value.S51	0.58062327
vid	0.22844808

Table 3 Variable Importance of variables on Gross Profit

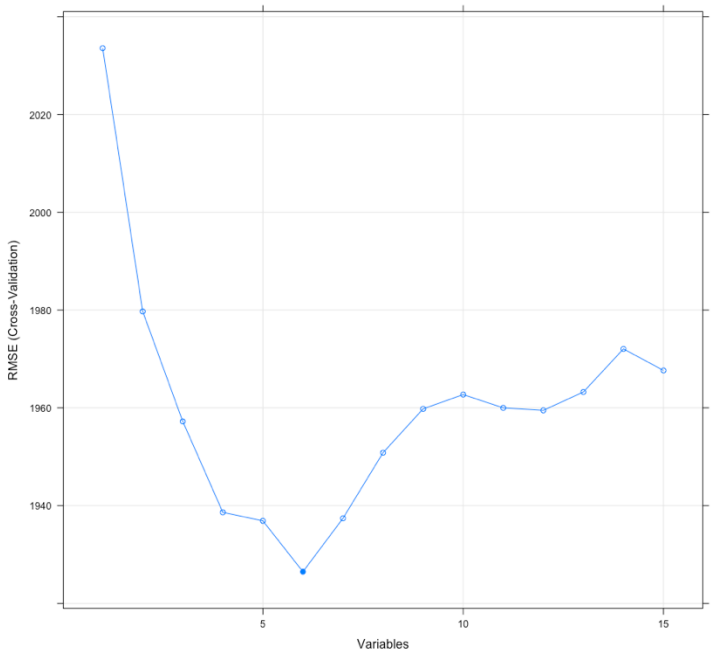

Fig 2 RMSE vs Number of variable

It is clear from the graph that the model RMSE is lowest with 6 variables. The variables chosen are Length, Capacity GT and capacity index.

3.3.3-Sundry Receipts: - Sundry Receipts is a totally independent variable. To find the importance of the independent feature over the target value, function rpart was used of Caret package.

Feature	Importance
Capacity GT.	0.04417342
FTE	0.12985156
Length	0.03681243
Total Active Vessels in Segment	0.12544564
Total Engine Power kW for Segment Size	0.09407904
Total Jobs	0.11582950
value.S44	0.26900967
value.S45	0.06647982
value.S51	0.07106498
vid	0.05131070

Table 4 Variable Importance of variables on Sundry Receipts

It is clear that importance of the variable is quite less indicating that the model might not be able to predict the target variable very well which will be discussed in the later part of the report.

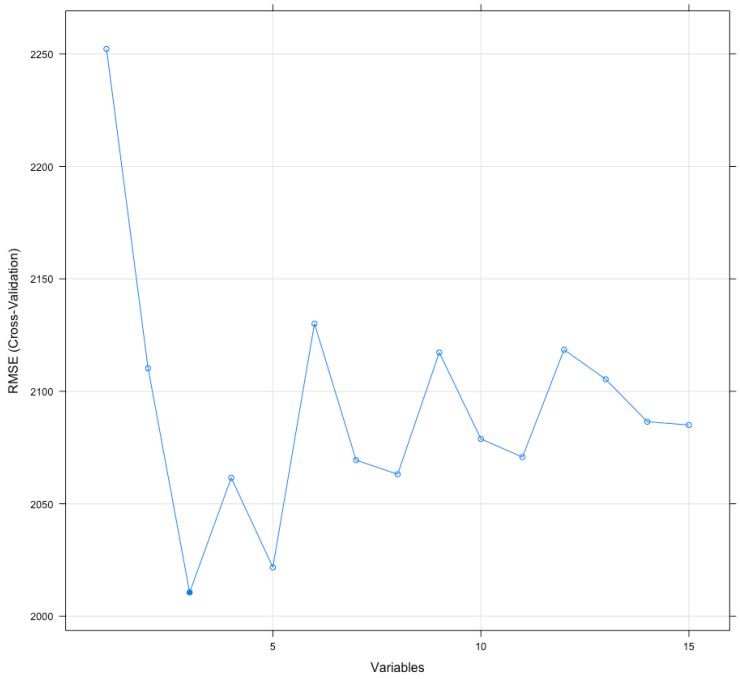

Fig 3 RMSE vs Number of variable

3.4-Variables visualization: -

All the plots used in the analysis are plotted using the library *ggplot()*
Multiplot function is used to generate these plots. (Code attached in Appendix)

3.4.1- Gross Profit: -

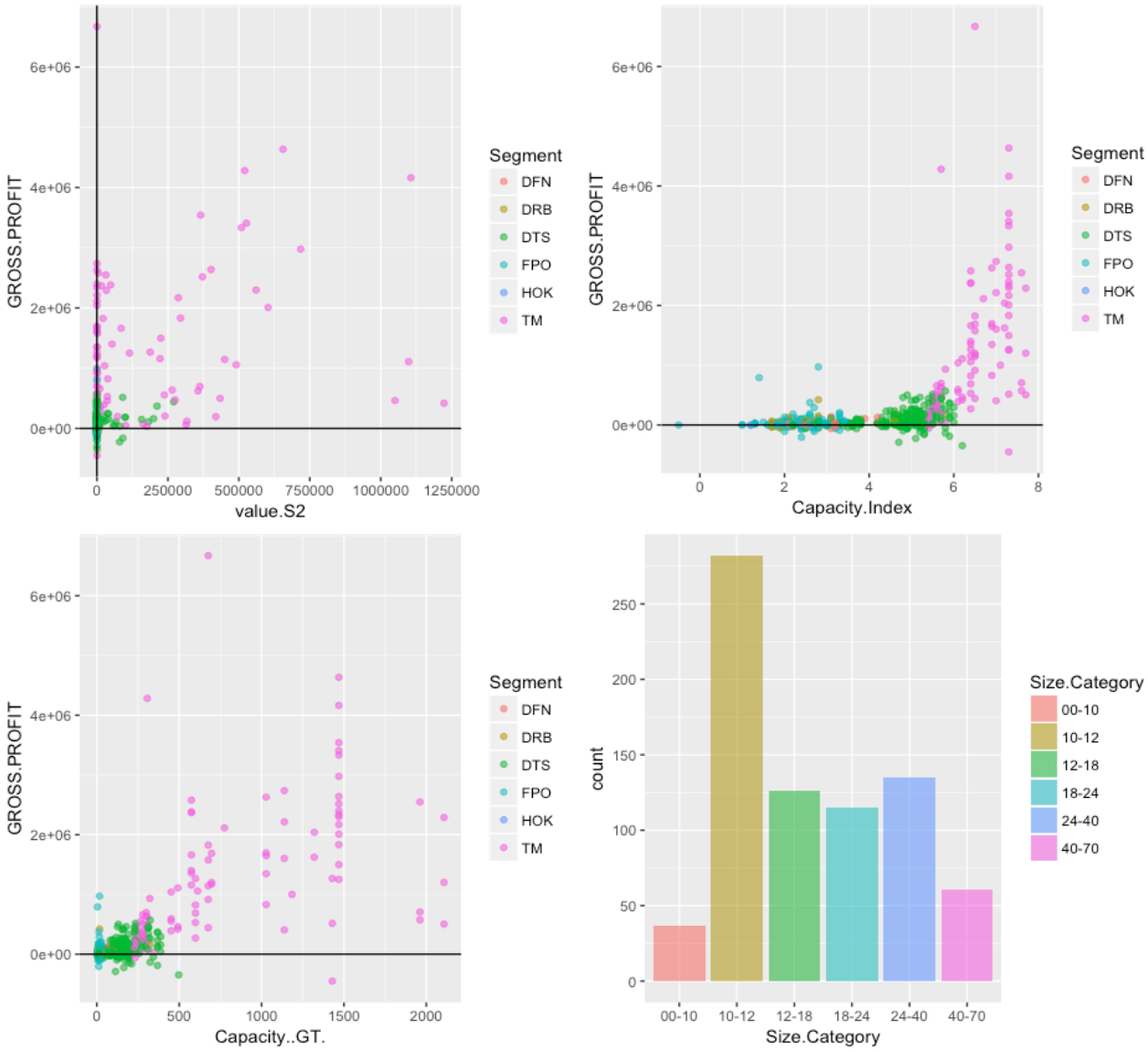

Fig 4 Gross profit relation with important variables

The first graph shows the spread of Gross Profit with the feature value.S2 plotted segment wise. The relation is clearly not linear and with a lot outlier especially when Value S2 is near 0 or is tending towards max value.
The second graph shows the spread of Gross Profit with Capacity Index. The spread is quite concentrated in the beginning and shows somehow exponential relation with few outliers at maximum value of Capacity Index.

The third graph shows the spread of Gross profit with capacity GT. Again the spread is quite concentrated in the beginning. With increment of Capacity value, the graphs are spreaded with a lot of outliers in otherwise almost linear relation.

3.4.2- Depreciation: -

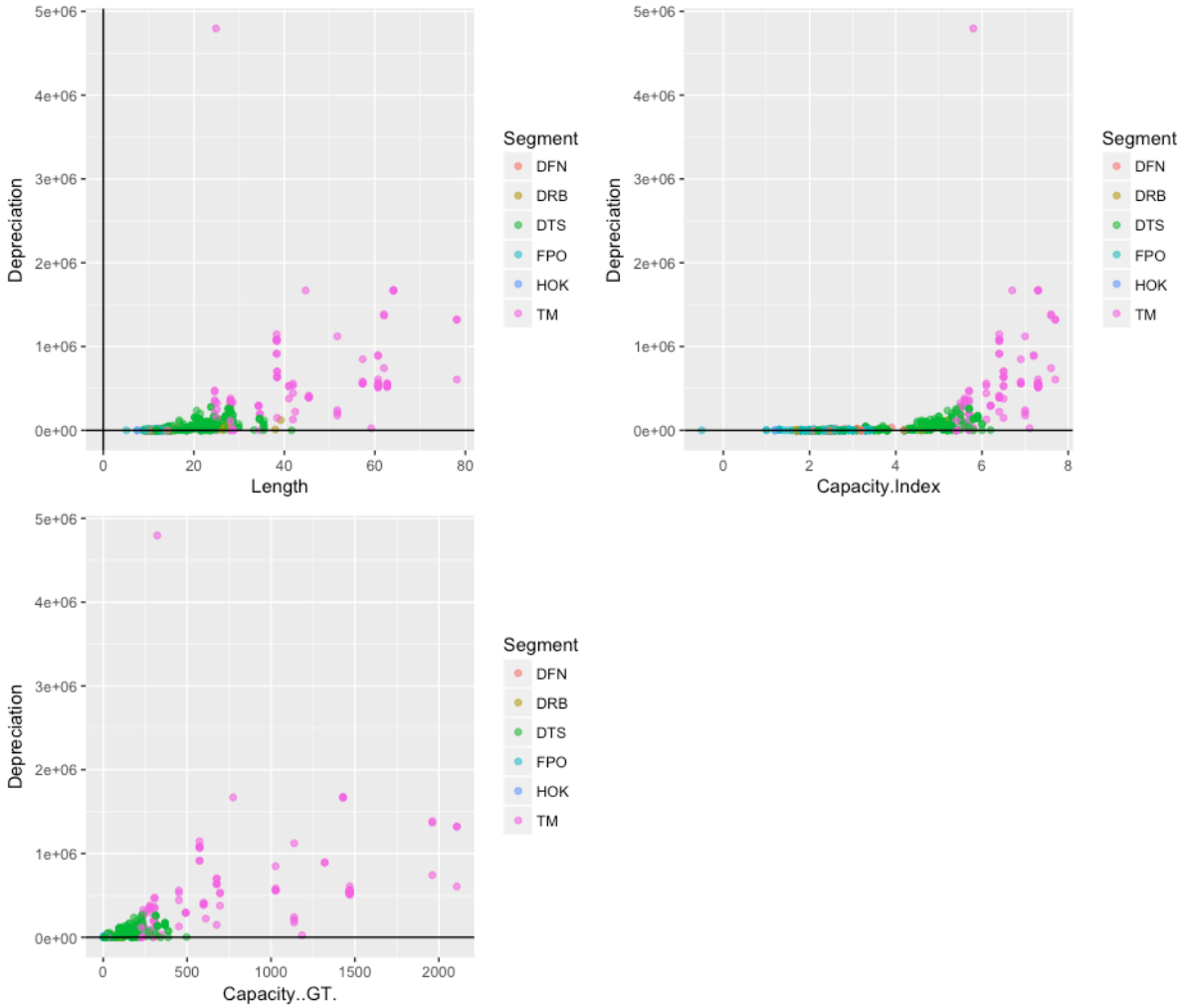

Fig 5 Depreciation relation with important variables

All the graphs have concentrated spread with very few outliers indicating a good model based on the variables chosen.

3.4.3: - Sundry Receipts: -

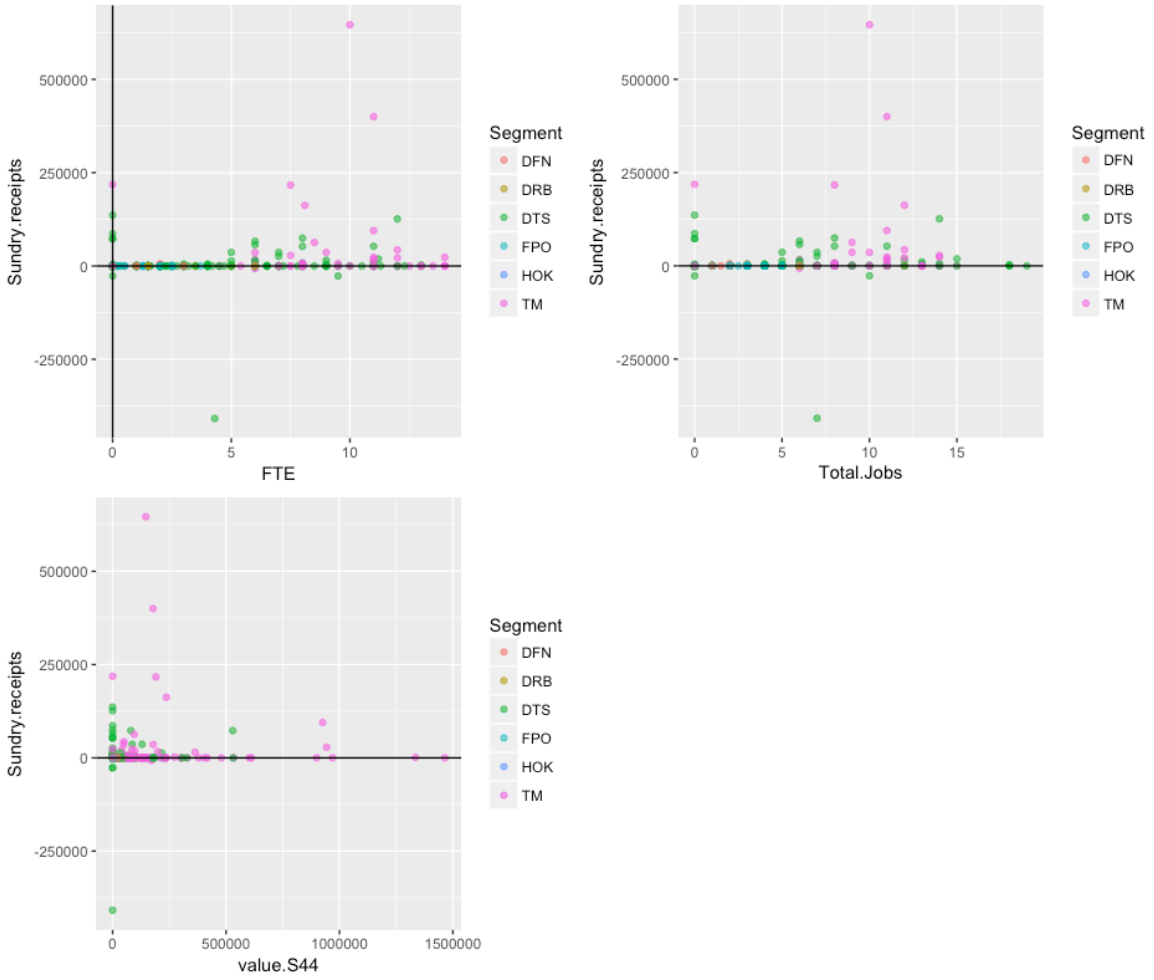

Fig 6 Sundry Receipts relation with important variables

Although most important variables are chosen but it is visible in the graphs that the spread is quite random and value of sundry receipts is near to zero. This can be explained by understanding that sundry receipts are pretty random and mostly contain penalties on fees, thus it could not be predicted very well. For the sake of this analysis sundry receipts can be ignored as of now. It won't affect the model much given more than 91% NA values.

4- Regression Tree Model: -

4.1- Gross Profit: -

It can be seen from the single regression tree below that the first split is done on Capacity GT and the subsequent splits are done on value S51 and S45. The tree is pruned and 8 terminal nodes.

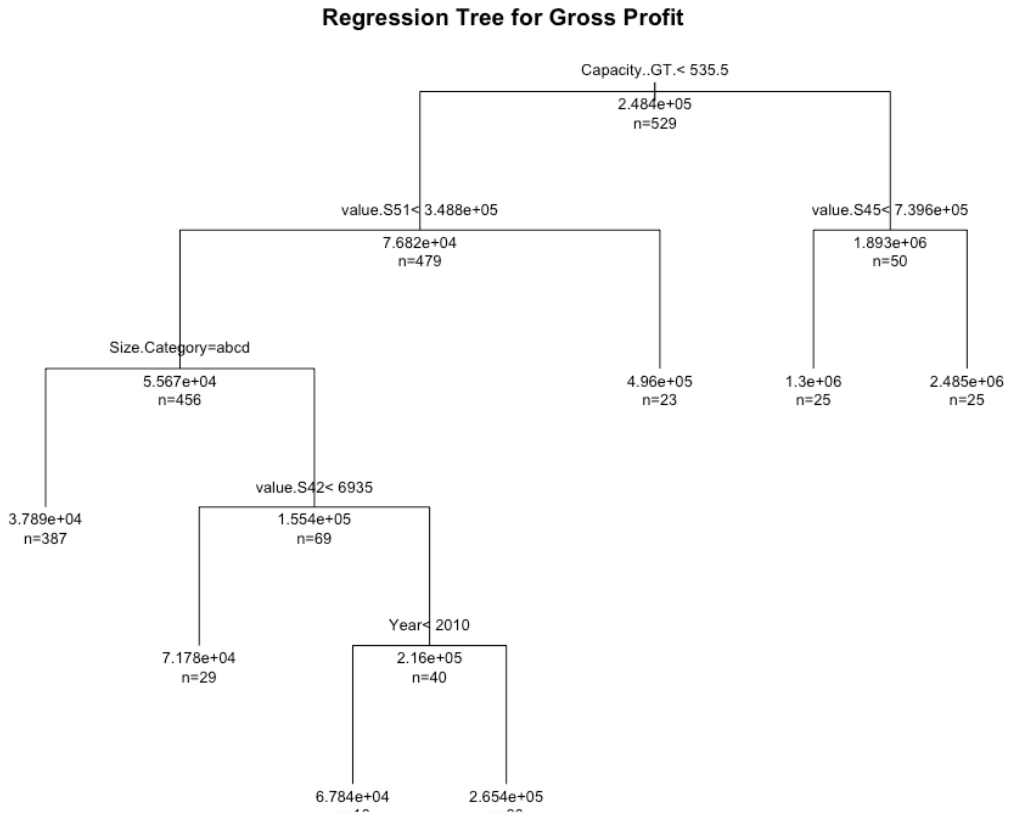

Fig 7 Single Regression tree of Gross Profit

The squared root of MSE achieved is **347880.7** and the achieved accuracy is **39.65%.**
The accuracy is low and clearly indicates that decision tree is not a good model to predict the Gross profit thus Ensemble model are to be used which are covered in the later part of the analysis.

The Complexity parameter graph is drawn below. The number of splits of the trees can be decided by analysing the CP plot. It is clear that after 4 split there is no decrement of CP is insignificant. Thus there is no point of growing the tree any further.

Further if the the accuracy is analysed there are a lot of outliers with the predicted line which result in the significant decrement in the accuracy. The predicted line in the graph is almost a straight line which is not the actual case.

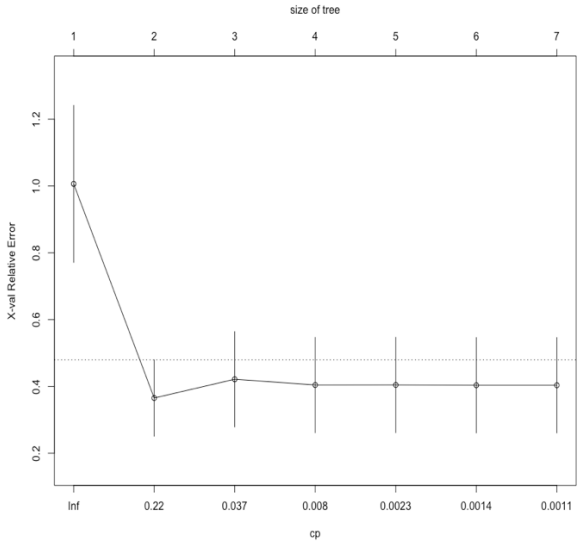

Fig 8 Complexity Parameter of tree on Gross Profit

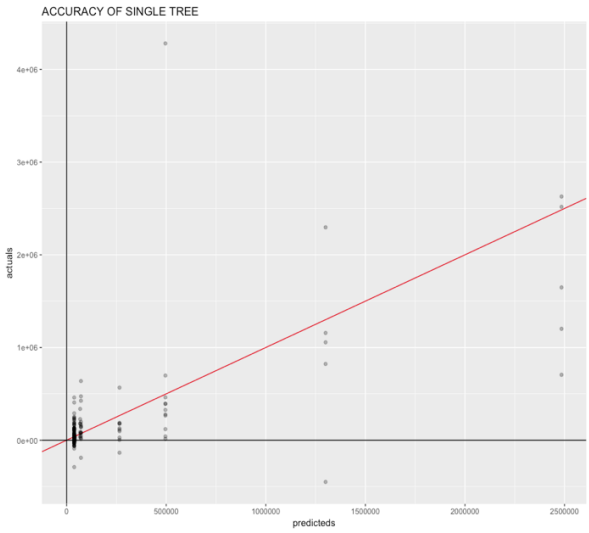

Fig 9 Accuracy of single regression tree on Gross Profit

4.2 - Depreciation: -

The Singe regression tree is shown below. The first split is done on the basis of S45. There are total 8 terminal node. All the chosen variable in the analysis are used for the split. The pruned tree is using more variable than 3 thus for the sake of better analysis, more than 3 variable are chosen for better prediction in this analysis.

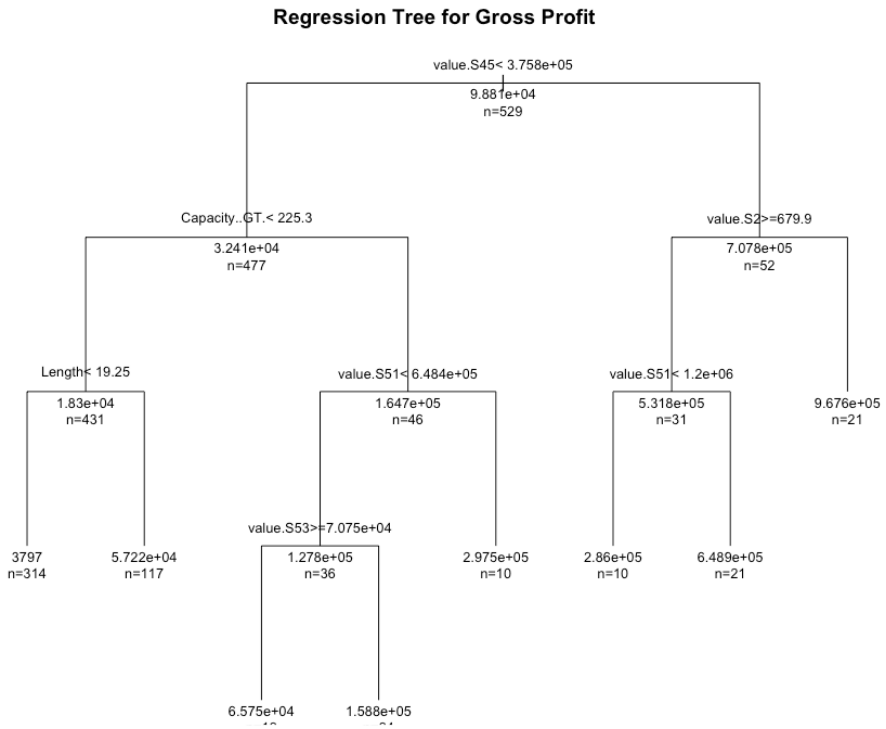

Fig 10 Single regression tree on Depreciation

The complexity plot is drawn below. It is pretty clear from the plot that after the 6 splits there is no significant decrement of complexity thus there is no point of growing the tree beyond this point.

The accuracy graph is also drawn below.
The achieved accuracy in this case is coming to be 73.74% and root squared value of MSE is 120464.6. The accuracy achieved is pretty good as compared to Gross Profit. It's good sign in favour of the hypothesis taken in the starting of the analysis that Depreciation can be predicted on the basis of non-financial based variables.

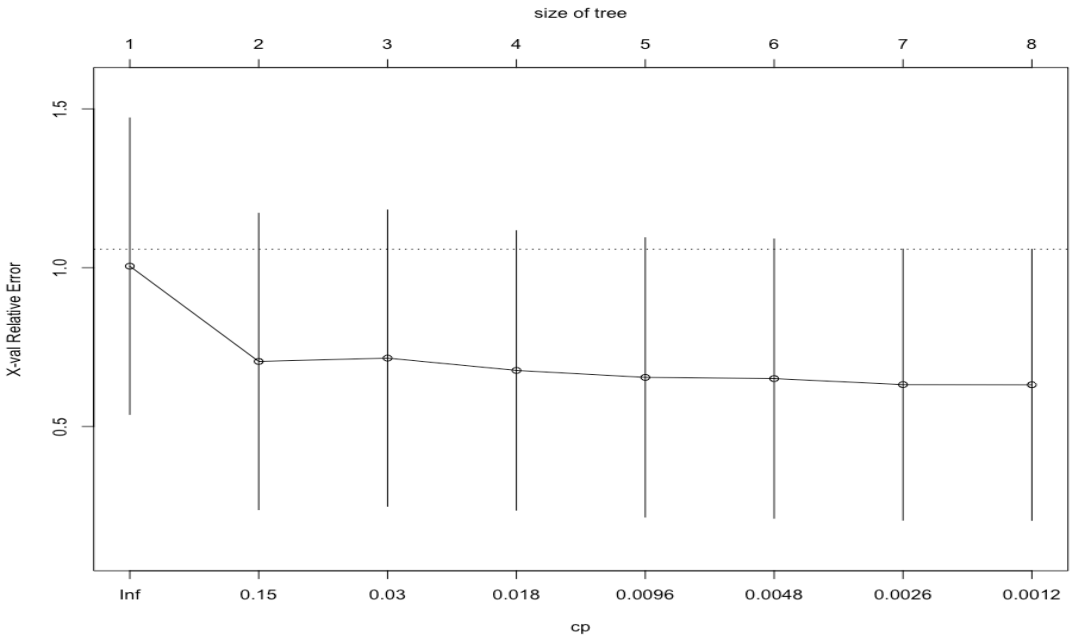

Fig 11 Complexity plot graph on Depreciation

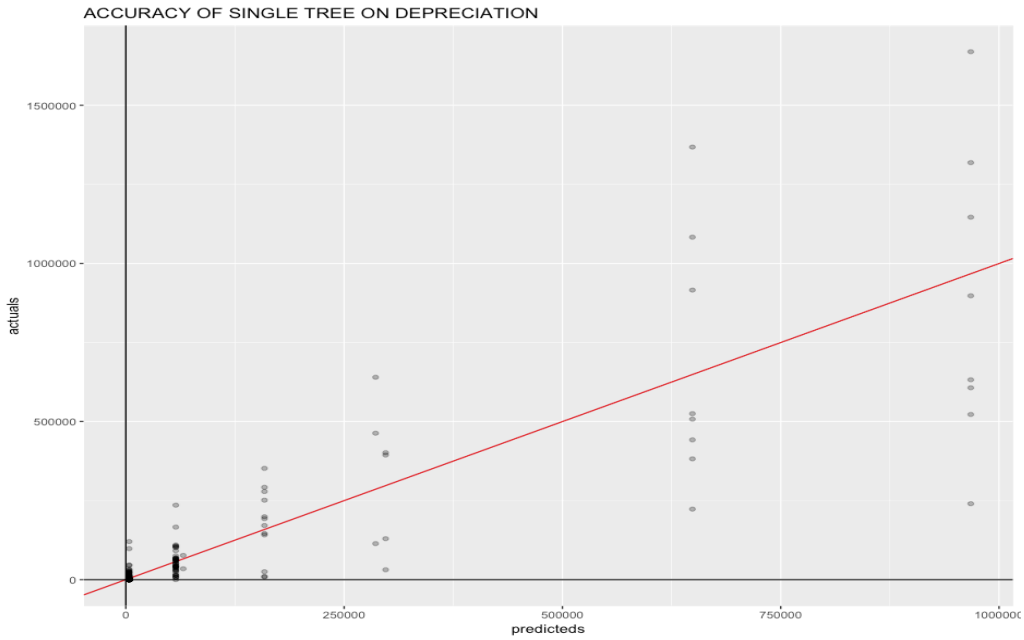

Fig 12 Accuracy graph on Depreciation

5- Ensemble Models: -

For model building the given data which is already divided into 3 data frames. But for this analysis only two are considered i.e. the one with Gross Profit as the target variable and the other one as Depreciation as the target variable are chosen.
Both these data frame were divided into train and test cases with 70/30 split. Function *rpart()* was used Caret library for splitting the data set. There are total 756 number of instances with 44 variables. After model training on training data set, predictions are made on the test data frame. The control function has cross validation function with 3 folds only reducing the computational cost.
For metric evaluation r squared value and square root of MSE are used. Further details are covered in the late part of the report in the evaluation section.

5.1- Random Forest: -

The random forest is an ensemble approach that can also be thought of as a form of nearest neighbour predictor. he random forest starts with a standard machine learning technique called a "decision tree" which, in ensemble terms, corresponds to our weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets. The random forest takes this notion to the next level by combining trees with the notion of an ensemble. Thus, in ensemble terms, the trees are weak learners and the random forest is a strong learner.

The reason random forest is chosen is evident from the fact that a single tree was able to predict the depreciation significantly. So if I tried to fit random forest on the data set to see its working and it worked quite well. Evidently it works satisfactory on predicting the Gross Profit as well for which I was skeptic in the start.

5.1.1 - Random Forest for predicting Gross Profit: -

Random forest was grown for predicting the Gross Profit because single regression tree is not able to predict satisfactory.
The model used 529 samples and 43 predictors. For resampling cross validation is used with 3 folds. The summary of the sample size is 353,353,352. The resampling across the tuning parameter are given in the below table.

mtry	RMSE	Rsquared	MAE
2	395026.6	0.6584316	149468.2
26	368088.1	0.7022247	137390.8
51	371302.1	371302.1	138971.6

Table 5 Tuning parameter of the random forest for Gross Profit

The final value of the used for the model was mtry = 26. The RMSE graph shows the number of randomly selected predictors. There are 25 randomly selected predictors in the model.

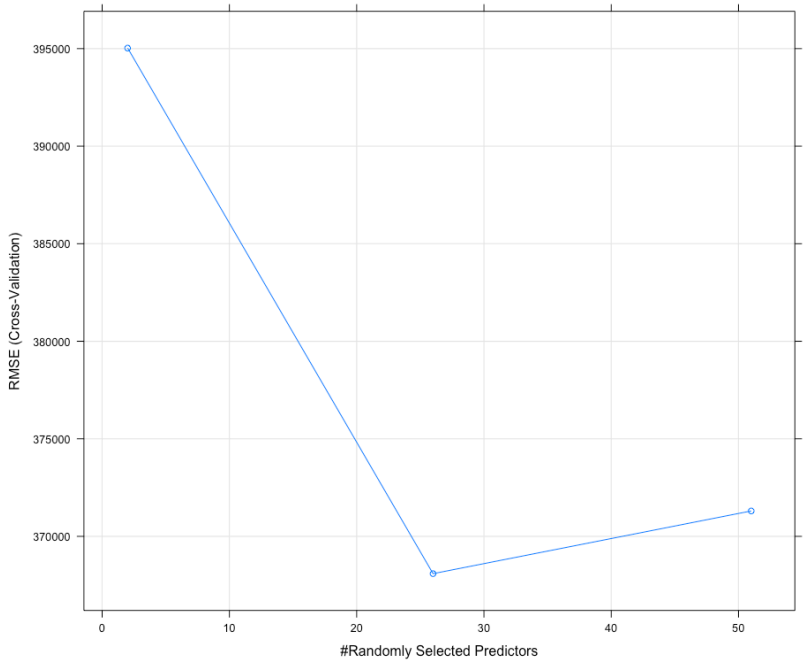

Fig 13 RMSE variance with randomly selected predictor

The final accuracy of the model is coming out to be 65.98% with the square root of MSE 243507. The accuracy has increased significantly when compare to single decision tree. The blue curve shows the fitting of the model.

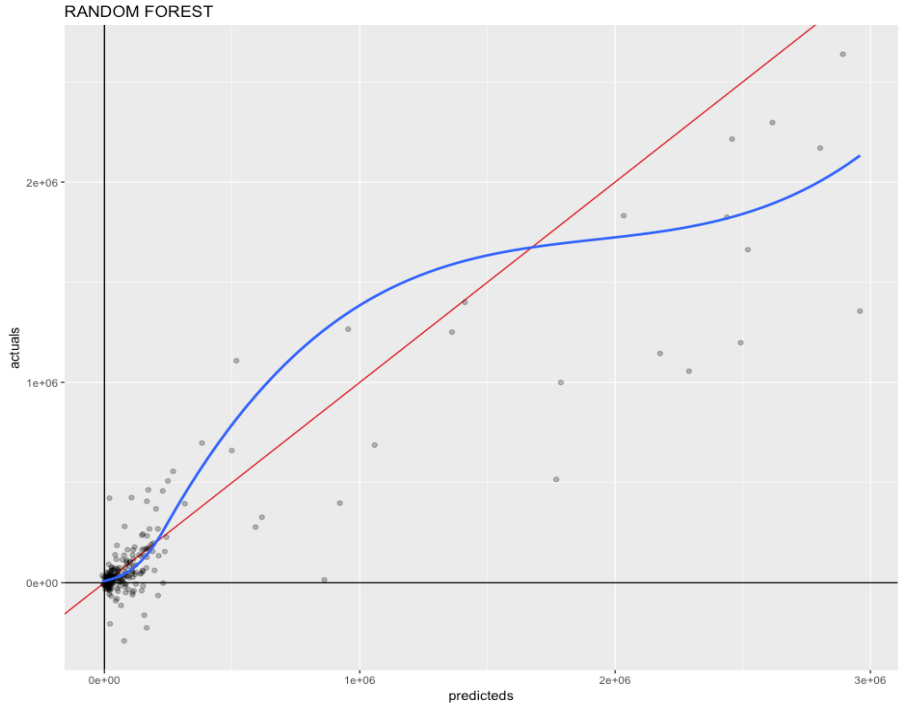

Fig 14 Accuracy curve of the random forest for Gross Profit

5.1.2- Random Forest for predicting Depreciation: -

Random for grown for predicting the Depreciation because single regression tree is not able to predict satisfactory.
The model used 529 samples and 43 predictors. For resampling cross validation is used with 3 folds. The summary of the sample size is 353,353,352. The resampling across the tuning parameter are given in the below table.

mtry	RMSE	Rsquared	MAE
2	202417.3	0.5816261	54929.59
26	219117.4	0.5562502	55119.71
51	240419.3	0.5087427	58708.88

Table 7 Tuning parameter of the random forest for Depreciation

RMSE was used to select the optimal model using the smallest value. The final value used for the model was mtry = 2. The RMSE The RMSE graph shows the number of randomly selected predictors. There are 25 randomly selected predictors in the model.

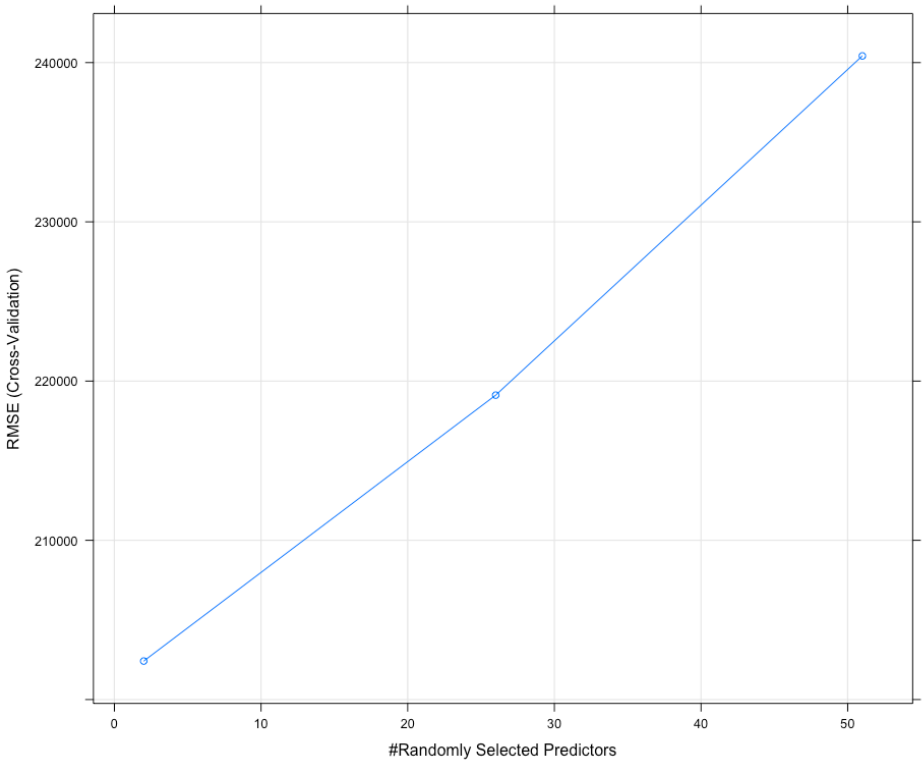

Fig 15 RMSE variance with randomly selected predictor

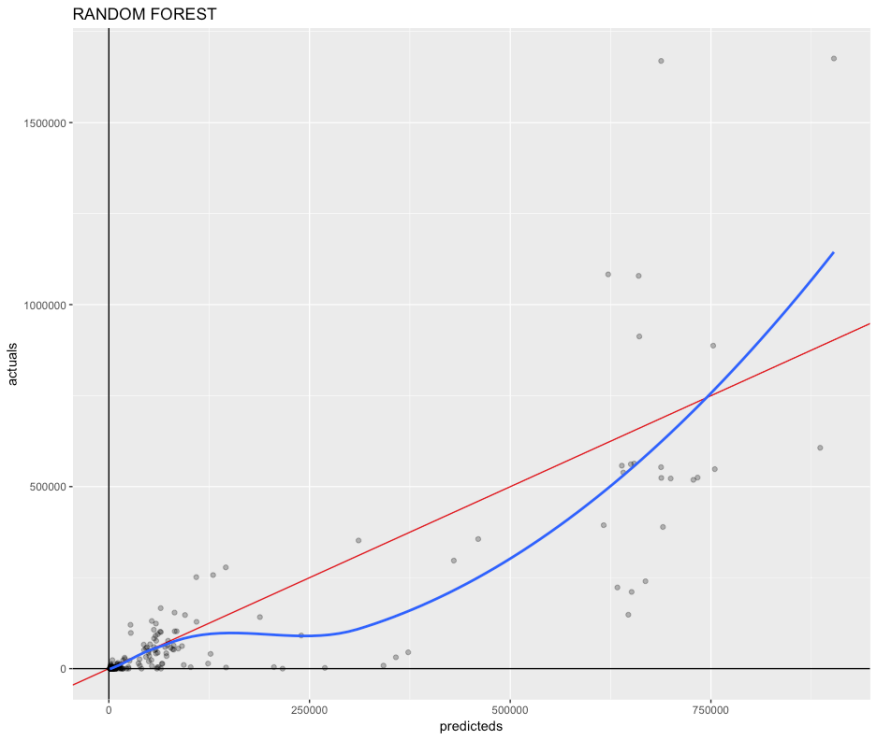

Fig 16 Accuracy curve of the random forest for Depreciation

The final accuracy of the model is coming out to be 67.31% with the square root of MSE 133161.8. The accuracy has increased significantly when compare to single decision tree. The blue curve shows the fitting of the model.

5.2 - Gradient Boosting Method: -

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Fitting the training set too closely can lead to degradation of the model's generalization ability. Several so-called regularization techniques reduce this overfitting effect by constraining the fitting procedure.

One natural regularization parameter is the number of gradient boosting iterations M (i.e. the number of trees in the model when the base learner is a decision tree). Increasing M reduces the error on training set, but setting it too high may lead to overfitting. An optimal value of M is often selected by monitoring prediction error on a separate validation data set. Besides controlling M, several other regularization techniques are used.

5.2.1 – GBM for predicting the Gross Profit: -

GBM method was used for predicting the Gross profit based on the independent variables and the accuracy of the model is analysed.

The model used 529 samples with 43 predictors. For resampling cross-validation was used with 3 folds. The summary size of the is 353,353,352. The resampling results across tuning parameters is given in the table below.

Intrecation Depth	n.trees	RMSE	Rsquared	MAE
1	50	409840.1	0.6419506	160850.6
1	100	421287.6	0.6221092	166372.5
1	150	423915.8	0.6291458	166264.4
2	50	394742.1	0.6646479	148055.9
2	100	395160	0.6644788	152680.6
2	150	406865.1	0.6470154	157806.4
3	50	404775.7	0.6550131	154070.6
3	100	410143.1	0.6431657	158262.1
3	150	410043.4	0.6374619	162330.5

Table 8 Tuning parameter of the GBM for Gross Profit

Tuning parameter 'shrinkage' was held constant at a value of 0.1 Tuning parameter 'n.minobsinnode' was held constant at a value of 10

The final values used for the model were n.trees = 50, interaction.depth = 2, shrinkage = 0.1 and n.minobsinnode = 10. The accuracy graph shows the model fitting and outliers.

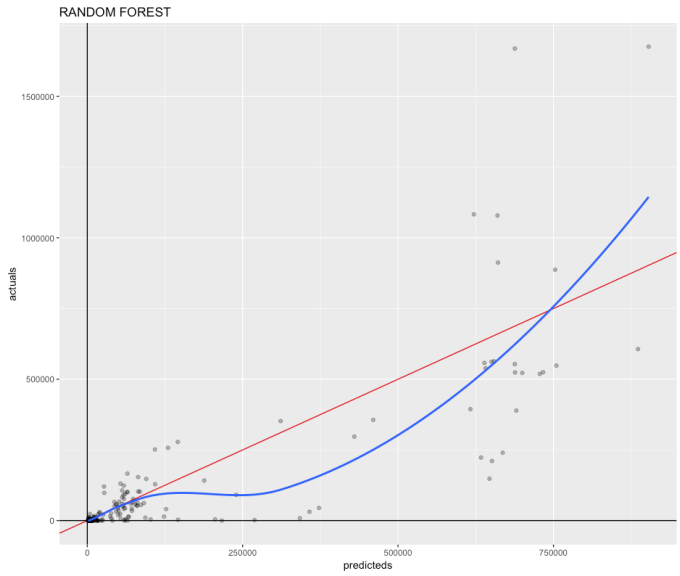

Fig 17 Accuracy curve of the GBM for Gross Profit

The accuracy achieved by this model 73.67% with the MSE 222988.8.

5.2.1 – GBM for predicting the Depreciation: -

GBM method was used for predicting the Gross profit based on the independent variables and the accuracy of the model is analysed.

The model used 529 samples with 43 predictors. For resampling cross-validation was used with 3 folds. The summary size of the is 353,353,352. The resampling results across tuning parameters is given in the table below.

Intrecation Depth	n.trees	RMSE	Rsquared	MAE
1	50	211659	0.5232602	66713.64
1	100	218079.5	0.5011932	72594.23
1	150	218366.4	0.496752	71626.18
2	50	214450.2	0.5161655	62080.84
2	100	216069.7	0.5073452	62974.95
2	150	224726.8	0.4708357	68223.22
3	50	209494.3	0.5329373	61533.95
3	100	215830.1	0.5039485	62876.04
3	150	221734.4	0.4792195	65405.62

Table 9 Tuning parameter of the GBM for Depreciation

Tuning parameter 'shrinkage' was held constant at a value of 0.1Tuning parameter 'n.minobsinnode' was held constant at a value of 10. RMSE was used to select the optimal model using the smallest value.

The final values used for the model were n.trees = 50, interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

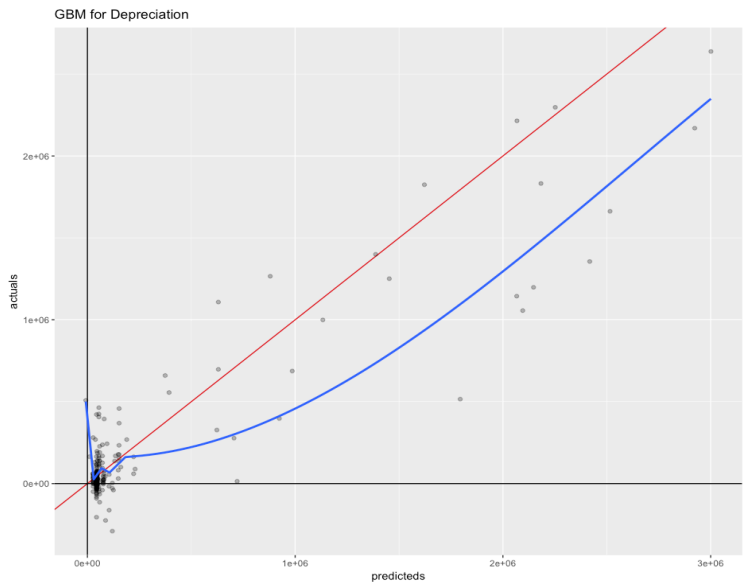

Fig 18 Accuracy curve of the GBM for Depreciation

The accuracy achieved is 75.0889% which is very satisfactory for the real world data. The square root of MSE 102853.1.

6- Comparison of models and conclusion: -

Total three models were used for predicting the target variables in this analysis: -

1- Single regression tree
2- Random Forest Ensemble
3- Gradient Boosting Method

The accuracy achieved in the single regression tree is very low giving the unpredictable nature and the hypothesis taken in the beginning that the target variable can be predicted without any financial attribute variable. Below tables shows the achieved accuracy of the model: -

Model	Gross Profit	Depreciation
Single regression tree	39.65%	73.74%
Random Forest	65.98%	67.31%
GBM	73.67%	75.08%

Table 10 Accuracy comparison of different models

The accuracy is increased significantly in the GBM Ensemble especially the depreciation

which is also predicted quite nicely with the single regression tree as well.
The hypothesis can be verified by the final results.

Sundry Receipts are ignored in the prediction given total 1389 NA values out of 1436 variables. Thus if the Gross Profit and Depreciation are predicted the Net Profit/Loss can be predicted.
The final model that should be selected for the prediction as per this analysis should be GBM.
The data has a lot of NA values which should be be removed and variables are removed for analysis to prevent any wrong training of the model.

Appendix i- Source Code: -

```
library(caret)
library(gbm)
library(mlbench)
library(rpart)
library(rpart.plot)
library(ggplot2)
library(psych)
library(corrplot)
library(randomForest)
library(party)
library(grid)
library(plyr)

set.seed(1234)
#file loading
setwd("~/Desktop")
eco_df<-read.csv2("merged_data.csv",sep=",",stringsAsFactors = TRUE)
str(eco_df)
#Data Cleaning and converting to numeric
head(eco_df$Capacity..GT.)
eco_df$Capacity..GT. <-  sub("," ,"",as.character(eco_df$Capacity..GT.))
eco_df$Capacity..GT. <- as.numeric(as.character(eco_df$Capacity..GT.))
eco_df$Average.Length.for.Segment <-  sub("," ,"",as.character(eco_df$Average.Length.for.Segment))
eco_df$Average.Length.for.Segment<-as.numeric(eco_df$Average.Length.for.Segment)
eco_df$Capacity.Index<-  sub("," ,"",as.character(eco_df$Capacity.Index))
eco_df$Capacity.Index<-as.numeric(eco_df$Capacity.Index)
eco_df$Size.Index<-  sub("," ,"",as.character(eco_df$Size.Index))
eco_df$Size.Index<-as.numeric(eco_df$Size.Index)
eco_df$Length<-  sub("," ,"",as.character(eco_df$Length))
eco_df$Length<-as.numeric(eco_df$Length)

#converting factor to numeric
eco_df[,15:45] <- apply(eco_df[,15:45] ,2 ,function(x){(gsub(",","",x))})
eco_df[,15:45] <- apply(eco_df[,15:45] ,2 ,function(x){(gsub(",","",x))})
eco_df[,15:45] <- apply(eco_df[,15:45] ,2 ,as.numeric)

# data cleaning to remove garbage value
eco_df$Size.Category<-as.character(eco_df$Size.Category)
eco_df$Size.Category[eco_df$Size.Category== '43079']<-'10-12'
eco_df$Size.Category[eco_df$Size.Category== '43435']<-'12-18'
table(eco_df$Size.Category)
eco_df$Size.Category<-as.factor(eco_df$Size.Category)

#Removing NA values
eco_df[is.na(eco_df)]<-0
str(eco_df)

#feature cleaning
na_count <-sapply(eco_df, function(y) sum(length(which(is.na(y)))))
na_count <- data.frame(na_count)
print(na_count)
table(eco_df$.=='0')
eco_df<-subset(eco_df,select = -c(1))
```

```
#redundant feature removal with cost attriribute
eco_df<-eco_df[,c(2:44,64,65,66)]
str(eco_df)
eco_df$Total.Jobs<-as.numeric(eco_df$Total.Jobs)
eco_df$FTE<-as.numeric(eco_df$FTE)

#Dividing the data frame into different target variables
cost1<-eco_df[,c(1:43,44)]
cost2<-eco_df[,c(1:43,45)]
cost3<-eco_df[,c(1:43,46)]
str(cost1)
str(cost2)
str(cost3)

#variable importance and feature removal for gross profit
fit1<-rpart(GROSS.PROFIT~.,data=cost1)
fit2<-rpart(Depreciation~.,data=cost2)
fit3<-rpart(Sundry.receipts~.,data=cost3)
varimp1<-data.frame(varImp(fit1))
varimp2<-data.frame(varImp(fit2))
varimp3<-data.frame(varImp(fit3))
print(varimp1)
print(varimp2)
print(varimp3)
control <- rfeControl(functions=rfFuncs, method="cv", number=2)

# run the RFE algorithm
results1 <- rfe(cost1[,c(1:15)], cost1[,c(16)], sizes=c(1:16), rfeControl=control)
results2 <- rfe(cost2[,c(1:15)], cost2[,c(16)], sizes=c(1:16), rfeControl=control)
results3 <- rfe(cost3[,c(1:15)], cost3[,c(16)], sizes=c(1:16), rfeControl=control)

# summarize the results
print(results1)
print(results2)
print(results3)

# list the chosen features
predictors(results1)
predictors(results2)
predictors(results3)

# plot the results
plot(results1, type=c("g", "o"))
plot(results2, type=c("g", "o"))
plot(results3, type=c("g", "o"))

#random forest data splitting
split1 <- sample(nrow(cost1), floor(0.7*nrow(cost1)))
split2 <- sample(nrow(cost2), floor(0.7*nrow(cost1)))
split3 <- sample(nrow(cost3), floor(0.7*nrow(cost1)))
traindf1 <- cost1[split1,]
testdf1 <-  cost1[-split1,]
traindf2 <- cost2[split2,]
testdf2 <-  cost2[-split2,]
traindf3 <- cost3[split3,]
testdf3 <-  cost3[-split3,]
```

```
# Model Defination
forest.tree1 <- train(GROSS.PROFIT~ ., method = "rf",
                data = traindf1, importance = T,
                trControl = trainControl(method = "cv", number = 3))

forest.tree2 <- train(Depreciation~ ., method = "rf",
                data = traindf2, importance = T,
                trControl = trainControl(method = "cv", number = 3))

forest.tree3 <- train(Sundry.receipts~ ., method = "rf",
                data = traindf3, importance = T,
                trControl = trainControl(method = "cv", number = 3))

forest.pred1 <- predict(forest.tree1, testdf1)
forest.pred2 <- predict(forest.tree2, testdf2)
forest.pred3 <- predict(forest.tree3, testdf3)
print(forest.tree1)
print(forest.tree2)
print(forest.tree3)
plot(forest.tree1, type=c("g","o"))
plot(forest.tree2, type=c("g","o"))

rf_preds_df1 <- data.frame(cbind(actuals=testdf1$GROSS.PROFIT , predicteds=forest.pred1))
str(rf_preds_df1)

#confusionMatrix(cost1$GROSS.PROFIT, predicteds ,positive = "Yes")
rf_rmse1 <- (mean((testdf1$GROSS.PROFIT- forest.pred1)^2))**0.5
rf_sse1 <- sum( (rf_preds_df1$predicteds - rf_preds_df1$actuals)^2 )
print(mean(rf_preds_df1$actuals))
rf_sst1 <- sum( (mean(cost1$GROSS.PROFIT) - rf_preds_df1$actuals)^2)
print(rf_sst1)
rf_r21 <- 1- rf_sse1/rf_sst1
cat('FOR RANDOM FOREST \n')
cat('Squar Root of MSE',rf_rmse1,'\n')
cat('R squared value:', rf_r21*100, '%\n' )
rf1<- ggplot(rf_preds_df1, aes(predicteds, actuals))+ geom_abline(color='#E41A1C')+ geom_point(alpha=0.3) +
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0)+geom_smooth(method = "loess",
se=FALSE)+ggtitle('RANDOM FOREST')
print(rf1)
head(rf_preds_df1)

rf_preds_df2 <- data.frame(cbind(actuals=testdf2$Depreciation, predicteds=forest.pred2))

#confusionMatrix(cost1$GROSS.PROFIT, predicteds ,positive = "Yes")
rf_rmse2 <- (mean((testdf2$Depreciation- forest.pred2)^2))**0.5
rf_sse2 <- sum( (rf_preds_df2$predicteds - rf_preds_df2$actuals)^2 )
rf_sst2 <- sum( (mean(cost2$Depreciation) - rf_preds_df2$actuals)^2)
rf_r22 <- 1- rf_sse2/rf_sst2
cat('FOR RANDOM FOREST \n')
cat('Squar Root of MSE',rf_rmse2,'\n')
cat('R squared value:', rf_r22*100, '%\n' )
head(rf_preds_df2)
rf2<- ggplot(rf_preds_df2, aes(predicteds, actuals))+ geom_abline(color='#E41A1C')+ geom_point(alpha=0.3) +
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0)+geom_smooth(method = "loess",
se=FALSE)+ggtitle('RANDOM FOREST')
```

```
print(rf2)

rf_preds_df3 <- data.frame(cbind(actuals=testdf3$Sundry.receipts, predicteds=forest.pred3))
rf_rmse3 <- (mean((testdf3$Sundry.receipts - forest.pred3)^2))**0.5
rf_sse3 <- sum( (rf_preds_df3$predicteds - rf_preds_df3$actuals)^2 )
rf_sst3 <- sum( (mean(cost3$Sundry.receipts) - rf_preds_df3$actuals)^2)
rf_r23 <- 1- rf_sse3/rf_sst3
cat('FOR RANDOM FOREST \n')
cat('Squar Root of MSE',rf_rmse3,'\n')
cat('R squared value:', rf_r23*100, '%\n' )
head(rf_preds_df3)
tail(cost3$Sundry.receipts)

#due to large number of zeros sundry receipt column dropped
colSums(cost3 ==0)
str(cost1)


#graph plotting

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {

 library(grid)



 # Make a list from the ... arguments and plotlist

 plots <- c(list(...), plotlist)



 numPlots = length(plots)



 # If layout is NULL, then use 'cols' to determine layout

 if (is.null(layout)) {

  # Make the panel

  # ncol: Number of columns of plots

  # nrow: Number of rows needed, calculated from # of cols

  layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),

                   ncol = cols, nrow = ceiling(numPlots/cols))

 }



 if (numPlots==1) {

  print(plots[[1]])
```

```
  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}

y1 <- ggplot(cost1, aes(value.S2, GROSS.PROFIT))+
  geom_point(aes(color=Segment), alpha = 0.6) + geom_vline(xintercept = 0) + geom_hline(yintercept = 0)
y2 <- ggplot(cost1, aes(Capacity..GT., GROSS.PROFIT))+
  geom_point(aes(color=Segment), alpha = 0.6) + geom_hline(yintercept = 0)
y3 <- ggplot(cost1, aes(Capacity.Index, GROSS.PROFIT))+
  geom_point(aes(color=Segment), alpha = 0.6) + geom_hline(yintercept = 0)
y4 <- ggplot(cost1, aes(Size.Category))+ geom_bar(aes(fill=Size.Category), alpha = 0.6)+
scale_y_continuous(breaks = seq(0,700, by=50))
multiplot(y1,y2,y3,y4 , cols=2)
z1 <- ggplot(cost2, aes(Length, Depreciation))+
  geom_point(aes(color=Segment), alpha = 0.6) + geom_vline(xintercept = 0) + geom_hline(yintercept = 0)
z2 <- ggplot(cost2, aes(Capacity..GT., Depreciation))+
```

```
  geom_point(aes(color=Segment), alpha = 0.6) + geom_hline(yintercept = 0)

z3 <- ggplot(cost2, aes(Capacity.Index, Depreciation))+

  geom_point(aes(color=Segment), alpha = 0.6) + geom_hline(yintercept = 0)


multiplot(z1,z2,z3, cols=2)

zz1 <- ggplot(cost3, aes(FTE, Sundry.receipts))+

  geom_point(aes(color=Segment), alpha = 0.6) + geom_vline(xintercept = 0) + geom_hline(yintercept = 0)

zz2 <- ggplot(cost3, aes(value.S44, Sundry.receipts))+

  geom_point(aes(color=Segment), alpha = 0.6) + geom_hline(yintercept = 0)

zz3 <- ggplot(cost3, aes(Total.Jobs, Sundry.receipts))+

  geom_point(aes(color=Segment), alpha = 0.6) + geom_hline(yintercept = 0)


multiplot(zz1,zz2,zz3, cols=2)

#single regression tree for gross profit

tree1 <- rpart(traindf1$GROSS.PROFIT ~ . ,method = "anova", data= traindf1, control = rpart.control(minsplit =
30, cp=0.001))
tree.pred1 <- predict(object = tree1, newdata= testdf1)

plot(tree1, uniform=TRUE,
    main="Regression Tree for Gross Profit ")
text(tree1, use.n=TRUE, all=TRUE, cex=.8)

tree_preds_df1 <- data.frame(cbind(actuals=testdf1$GROSS.PROFIT, predicteds=tree.pred1))
tree_rmse1 <- (mean((testdf1$GROSS.PROFIT- tree.pred1)^2))**0.5
tree_sse1 <- sum( (tree_preds_df1$predicted - tree_preds_df1$actuals)^2 )
tree_sst1 <- sum( (mean(eco_df$GROSS.PROFIT) - tree_preds_df1$actuals)^2)
tree_r21 <- 1- tree_sse1/tree_sst1

cat('FOR SINGLE REGRESSION TREE OF GROSS PROFIT \n')
cat('Squar Root of MSE',tree_rmse1,'\n')
cat('R squared value:', tree_r21*100, '%\n' )

head(tree_preds_df1)
printcp(tree1)
plotcp(tree1)
rsq.rpart(tree1)

x1<- ggplot(tree_preds_df1, aes(predicteds, actuals))+ geom_abline(color='#E41A1C')+ geom_point(alpha=0.3)
+
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0) +ggtitle('ACCURACY OF SINGLE TREE')
print(x1)

# regresssion tree for depriciation
```

```
tree2 <- rpart(traindf2$Depreciation~ . ,method = "anova", data= traindf2, control = rpart.control(minsplit =
30, cp=0.001))
tree.pred2 <- predict(object = tree2, newdata= testdf2)

plot(tree2, uniform=TRUE,
    main="Regression Tree for Gross Profit ")
text(tree2, use.n=TRUE, all=TRUE, cex=.8)

tree_preds_df2 <- data.frame(cbind(actuals=testdf2$Depreciation, predicteds=tree.pred2))
tree_rmse2 <- (mean((testdf2$Depreciation - tree.pred2)^2))**0.5
tree_sse2 <- sum( (tree_preds_df2$predicted - tree_preds_df2$actuals)^2 )
tree_sst2 <- sum( (mean(eco_df$Depreciation) - tree_preds_df2$actuals)^2)
tree_r22 <- 1- tree_sse2/tree_sst2

cat('FOR SINGLE TREE ON DEPRECIATION \n')
cat('Squar Root of MSE',tree_rmse2,'\n')
cat('R squared value:', tree_r22*100, '%\n' )

head(tree_preds_df2)
printcp(tree2)
plotcp(tree2)
rsq.rpart(tree2)

x2<- ggplot(tree_preds_df2, aes(predicteds, actuals))+ geom_abline(color='#E41A1C')+ geom_point(alpha=0.3)
+
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0) +ggtitle('ACCURACY OF SINGLE TREE ON
DEPRECIATION')
print(x2)

#GBM model for Gross profit
boost.model1 <- train(GROSS.PROFIT~ ., method = "gbm",data = traindf1, verbose = F,
                trControl = trainControl(method = "cv", number = 3))

boost.pred1 <- predict(boost.model1, testdf1)
print(boost.pred1)
print(boost.model1)
#plot(varImp(boost.model), top = 10)

boost_preds_df1 <- data.frame(cbind(actuals=testdf1$GROSS.PROFIT, predicteds=boost.pred1))
plot(boost_preds_df1, type=c("g","o"))
gbm1<- ggplot(boost_preds_df1, aes(predicteds, actuals))+ geom_abline(color='#E41A1C')+
geom_point(alpha=0.3) +
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0)+geom_smooth(method = "loess",
se=FALSE)+ggtitle('GBM for Gross Profit')
print(rf2)

boost_rmse1 <- (mean((testdf1$GROSS.PROFIT - boost.pred1)^2))**0.5
boost_sse1 <- sum( (boost_preds_df1$predicteds - boost_preds_df1$actuals)^2 )
boost_sst1 <- sum( (mean(eco_df$GROSS.PROFIT) - boost_preds_df1$actuals)^2)
boost_r21 <- 1- boost_sse1/boost_sst1

cat('FOR GBM \n')
cat('Squar Root of MSE',boost_rmse1,'\n')
cat('R squared value:', boost_r21*100, '%\n' )

#GBM model for Depreciation
boost.model2 <- train(Depreciation~ ., method = "gbm",data = traindf2, verbose = F,
```

```
                trControl = trainControl(method = "cv", number = 3))

boost.pred2 <- predict(boost.model2, testdf2)


boost_preds_df2 <- data.frame(cbind(actuals=testdf2$Depreciation, predicteds=boost.pred2))

boost_rmse2 <- (mean((testdf2$Depreciation - boost.pred2)^2))**0.5
boost_sse2 <- sum( (boost_preds_df2$predicteds - boost_preds_df2$actuals)^2 )
boost_sst2 <- sum( (mean(eco_df$Depreciation) - boost_preds_df2$actuals)^2)
boost_r22 <- 1- boost_sse2/boost_sst2
gbm2<- ggplot(boost_preds_df1, aes(predicteds, actuals))+ geom_abline(color='#E41A1C')+
geom_point(alpha=0.3) +
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0)+geom_smooth(method = "loess",
se=FALSE)+ggtitle('GBM for Depreciation')
print(gbm2)
print(boost.model2)
cat('FOR GBM \n')
cat('Squar Root of MSE',boost_rmse2,'\n')
cat('R squared value:', boost_r22*100, '%\n' )
```

Appendix ii- Python Script for merging the data: -

```
import pandas as pd
eco_df = pd.read_csv('economic.csv')
eco_df.rename(columns={'Vessel  ID': 'vid'}, inplace=True)
print(eco_df)
fish_df = pd.read_csv('fish2.csv')
# print(fish_df)

# result = pd.concat([eco_df, fish_df], axis=1, join_axes=[eco_df.index])
# result = pd.DataFrame()
# result = pd.concat([eco_df, fish_df], axis=1, join='inner')
# print(result)

result = pd.merge(eco_df, fish_df, on=['vid', 'Year'])
print(result)

dup = eco_df.duplicated(['vid', 'Year']).tolist()
print("number of duplicates = ", sum(dup))

true_index_list = []
for i, b in enumerate(dup):
   if b is True:
      true_index_list.append(i)

print("row numbers of duplicates = ", true_index_list)

print(result.duplicated(['vid', 'Year']))


result.to_csv('merged_data.csv')
```