Capstone Project

Flight cost prediction

Notes-2

K.Vineet *PATNAIK*

Table of contents

1. **Model Building and Interpretation**
2. **Tuning and Testing the models**

   Multi-linear regression,CART,Random Forest

3. **Interpretation of the best model**

Model Building and Interpretation

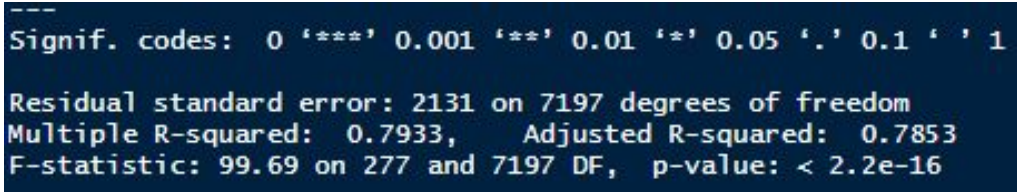
Since this is a regression problem we will use methods such as Multi linear regression, CART & Random forest .To make the data usage simple and fast execution we can manipulate some of the variables in the given data like convert different time periods to intervals like morning, noon etc and convert the dates to days of the month to also understand when the maximum passengers are flying by and performance metrics such as rmse,R-squared value are taken

*Multi Linear Regression Model *:

```
## Dividing the data into 70,30
set.seed(250)
smp_size = floor(0.70*nrow(dataset))
train_ind <- sample(seq_len(nrow(dataset)), size = smp_size)
g_train = dataset[train_ind, ]
g_test =  dataset[-train_ind,]
```

```
nrow(g_train)        #[1] 7478                                   nrow(g_test)
#[1] 3205
```

```
glm_model = lm(Price~ Airline+Duration+Arrival_Time+Total_Stops,data = g_test)
summary(glm_model)
```



```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2131 on 7197 degrees of freedom
Multiple R-squared:  0.7933,	Adjusted R-squared:  0.7853
F-statistic: 99.69 on 277 and 7197 DF,  p-value: < 2.2e-16
```

Here the model looks good because of a high R-squared value of 0.7624 when not considering all the variables and while considering all the variables there is even higher value of 0.7933 which is very good So we have tested with various variables and removed some which had very less impact on the output like dep_time,Source,Date_of_journey etc.

```
library(Metrics)
rmse(g_test$Price,predict(glm_model,g_test))
```

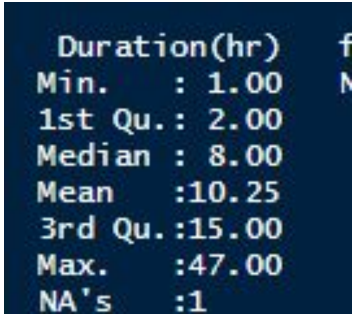
Rmse value = 1239.304 which is a pretty good value

CART & Random Forest

We have created some new variables in order to reduce the complexity of the data that will be used to create trees. Some of the variables created are Arr_time, Day, Duration(min),Flight range and cleared some of the clutter and some graphs have been produced in tableau to better understand the variables New sheet has been added to perform in R for CART and random forest .(FlightPrice train 5.xlsv)

The different libraries used are :
library(readxl),library(ggplot2),library(caTools),library(caret),library(rpart), library(ipred),library(randomForest),library(rpart.plot),library(tidyverse),library(Metrics), library(rpart.plot), library(RColorBrewer), library(rattle)

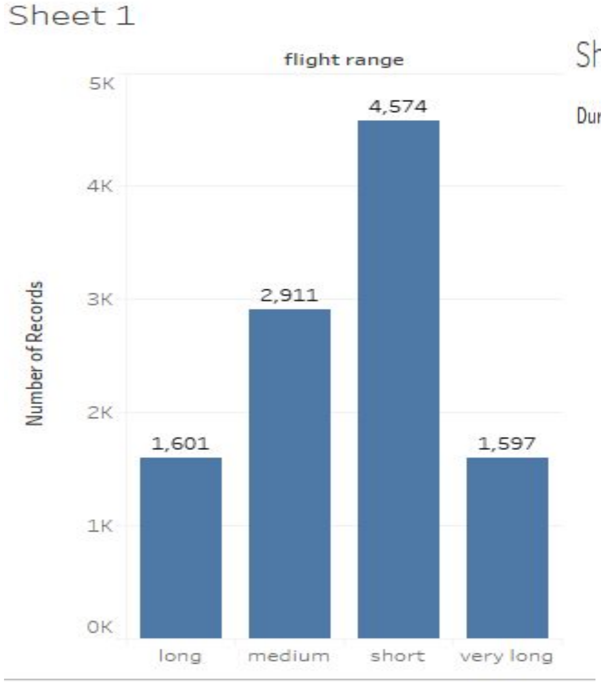
summary(duration(hr))

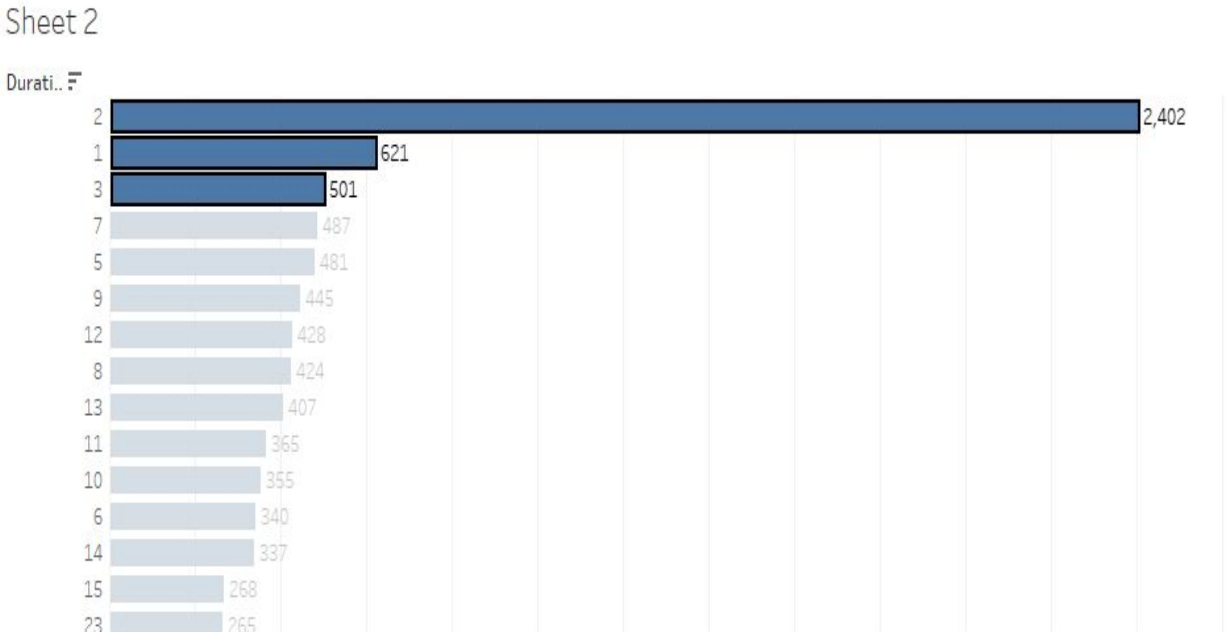


```
  Duration(hr)    f
 Min.    : 1.00   N
 1st Qu.: 2.00
 Median : 8.00
 Mean   :10.25
 3rd Qu.:15.00
 Max.   :47.00
 NA's   :1
```

Data from TABLEAU

Some of the info has been collected from Tableau about new variables





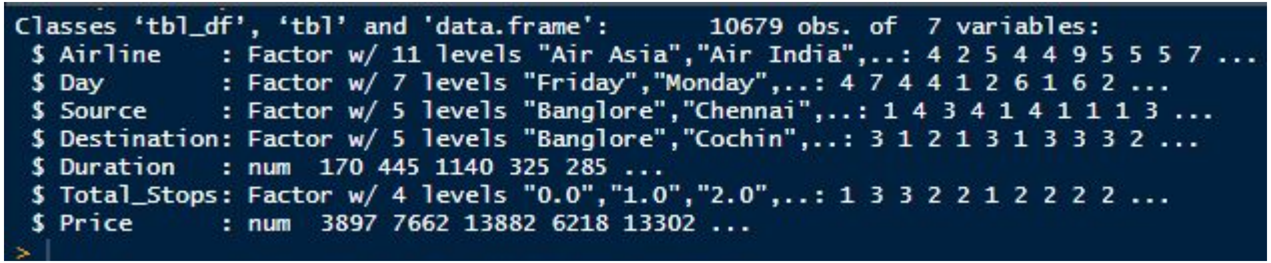
Flight range data

clearly people prefer short range flight and a

maximum prefer 2-5 hour flights.

For the CART model all the variables are converted to factors and numeric so there would not be any coercion while analysing using R

str(dataset1)



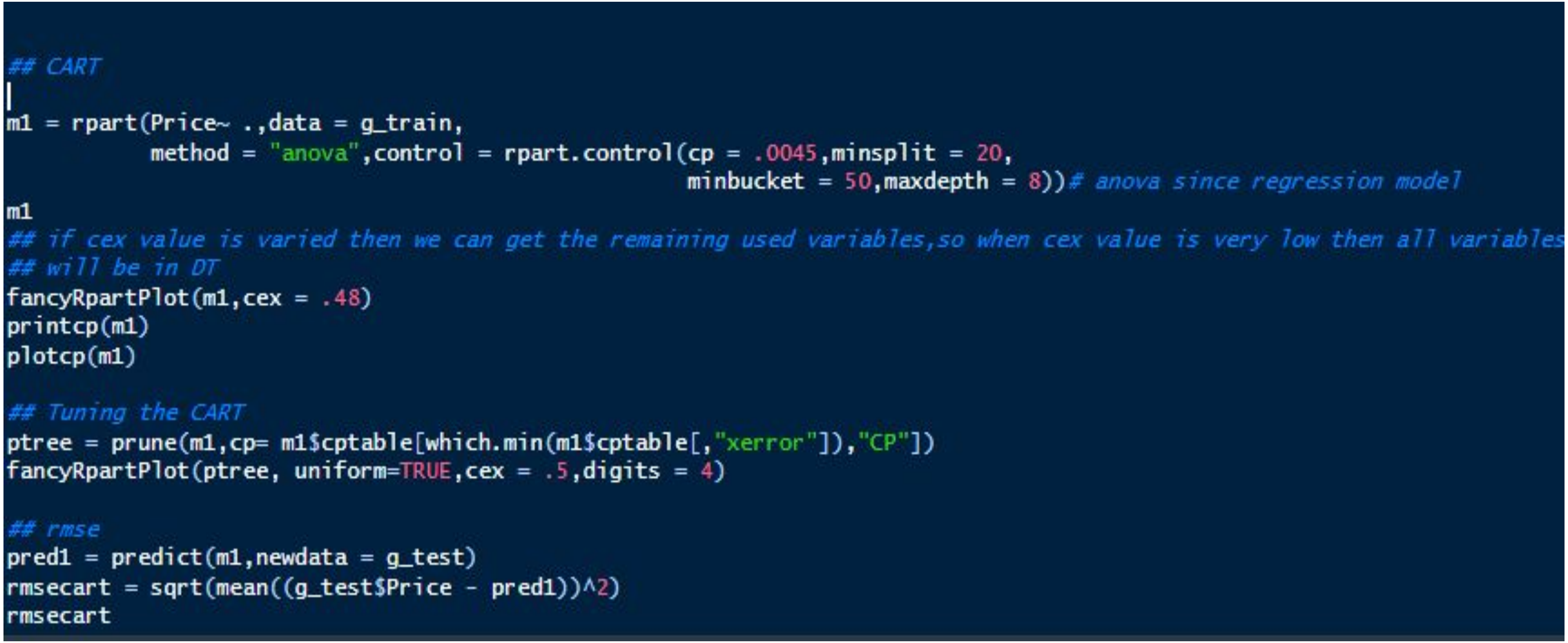
```
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':	10679 obs. of  7 variables:
 $ Airline    : Factor w/ 11 levels "Air Asia","Air India",..: 4 2 5 4 4 9 5 5 5 7 ...
 $ Day        : Factor w/ 7 levels "Friday","Monday",..: 4 7 4 4 1 2 6 1 6 2 ...
 $ Source     : Factor w/ 5 levels "Banglore","Chennai",..: 1 4 3 4 1 4 1 1 1 3 ...
 $ Destination: Factor w/ 5 levels "Banglore","Cochin",..: 3 1 2 1 3 1 3 3 3 2 ...
 $ Duration   : num  170 445 1140 325 285 ...
 $ Total_Stops: Factor w/ 4 levels "0.0","1.0","2.0",..: 1 3 3 2 2 1 2 2 2 2 ...
 $ Price      : num  3897 7662 13882 6218 13302 ...
>
```

```
set.seed(250)
smp_size = floor(0.70*nrow(dataset1))
train_ind <- sample(seq_len(nrow(dataset1)), size = smp_size)
g_train = dataset1[train_ind, ]
g_test =  dataset1[-train_ind,]
```

The training and testing is created to analyze the CART and RandomForest model.

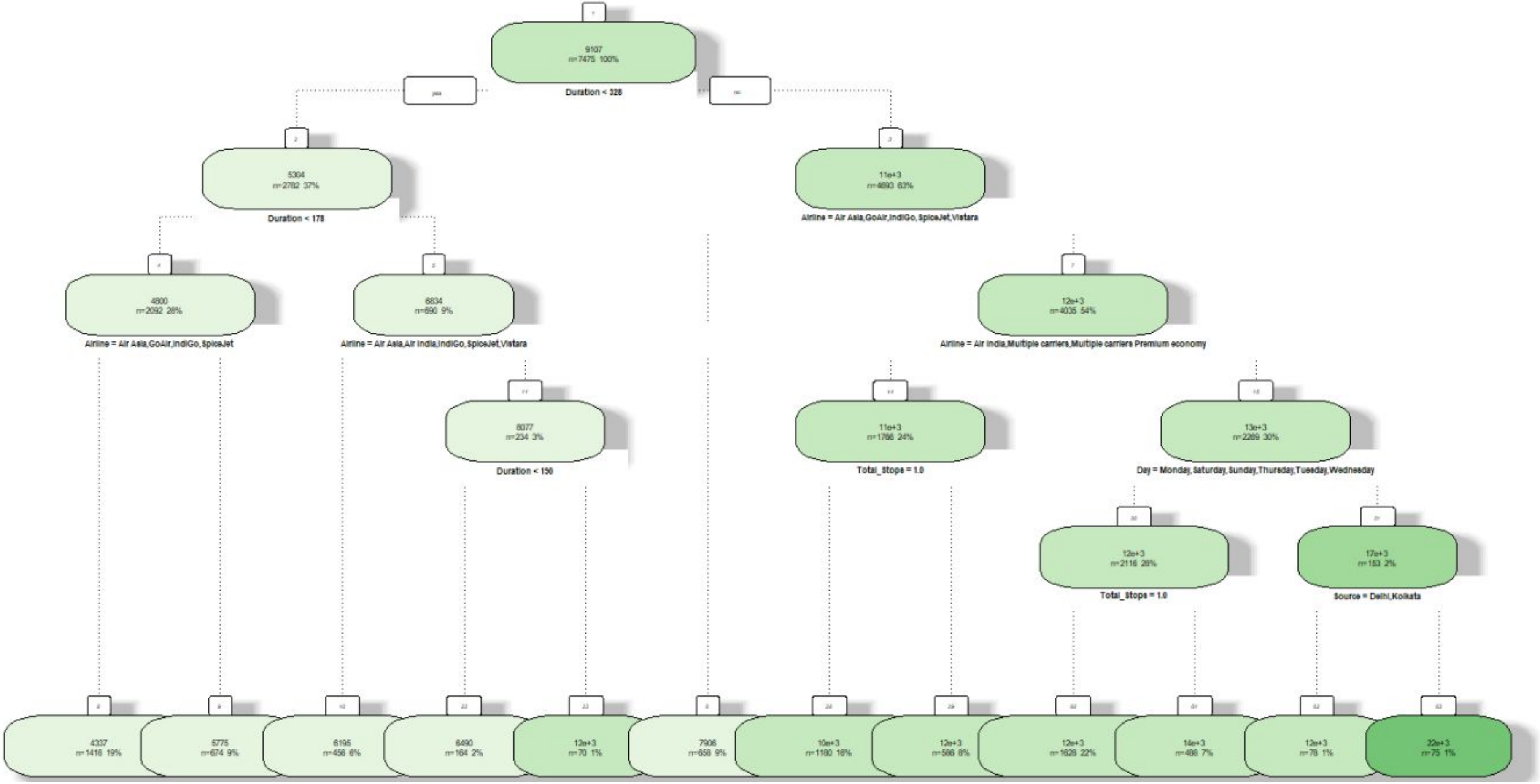
The Cart model was created and tree was built and tuning was done and the root mean square error(rmse) is found,tuning is done so that the error value will be less and the prediction(Price) is more accurate .

R Code

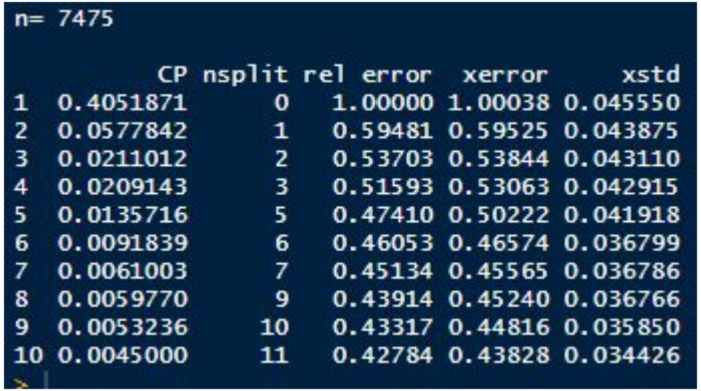


```
## CART

m1 = rpart(Price~ .,data = g_train,
           method = "anova",control = rpart.control(cp = .0045,minsplit = 20,
                                                    minbucket = 50,maxdepth = 8))# anova since regression model
m1
## if cex value is varied then we can get the remaining used variables,so when cex value is very low then all variables
## will be in DT
fancyRpartPlot(m1,cex = .48)
printcp(m1)
plotcp(m1)

## Tuning the CART
ptree = prune(m1,cp= m1$cptable[which.min(m1$cptable[,"xerror"]),"CP"])
fancyRpartPlot(ptree, uniform=TRUE,cex = .5,digits = 4)

## rmse
pred1 = predict(m1,newdata = g_test)
rmsecart = sqrt(mean((g_test$Price - pred1))^2)
rmsecart
```

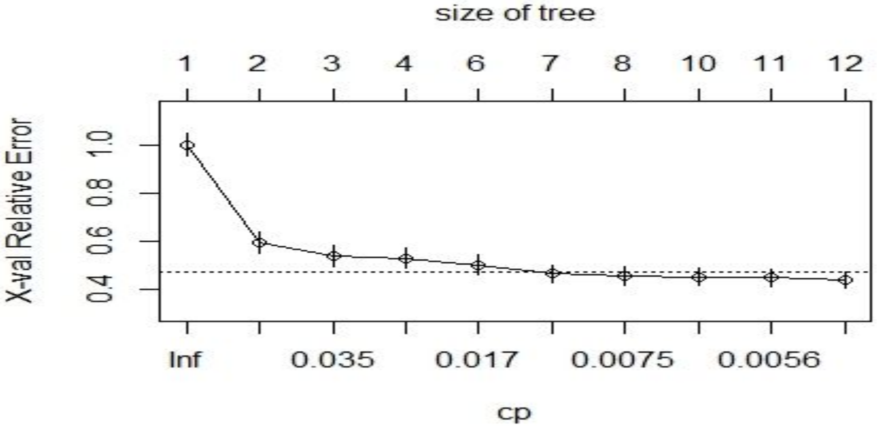
The long method to building tree is given



Printing and plotting cp to fine tune the model and deciding cp value for tuning

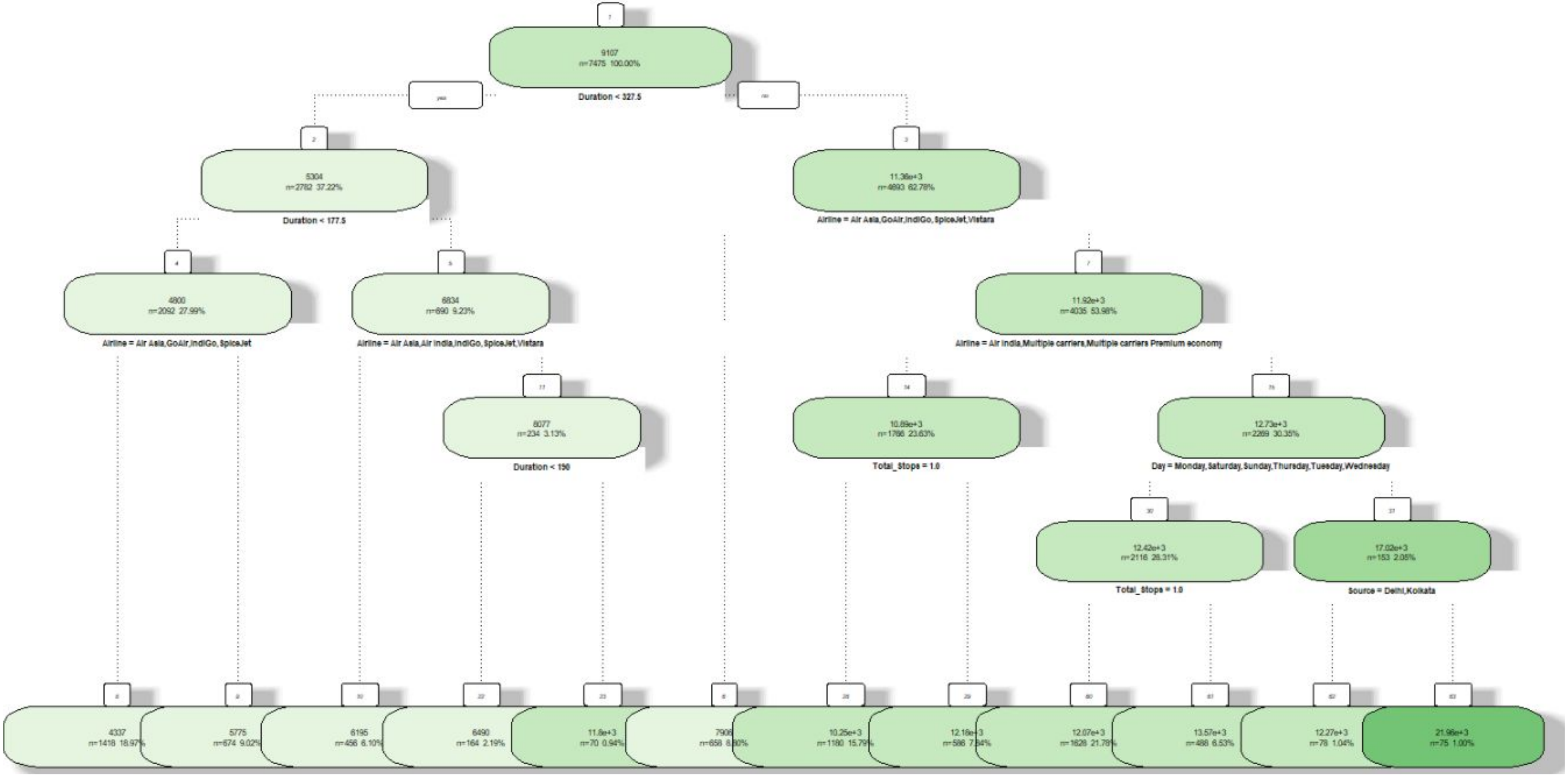


```
n= 7475

          CP nsplit rel error  xerror     xstd
1  0.4051871      0   1.00000 1.00038 0.045550
2  0.0577842      1   0.59481 0.59525 0.043875
3  0.0211012      2   0.53703 0.53844 0.043110
4  0.0209143      3   0.51593 0.53063 0.042915
5  0.0135716      5   0.47410 0.50222 0.041918
6  0.0091839      6   0.46053 0.46574 0.036799
7  0.0061003      7   0.45134 0.45565 0.036786
8  0.0059770      9   0.43914 0.45240 0.036766
9  0.0053236     10   0.43317 0.44816 0.035850
10 0.0045000     11   0.42784 0.43828 0.034426
>
```

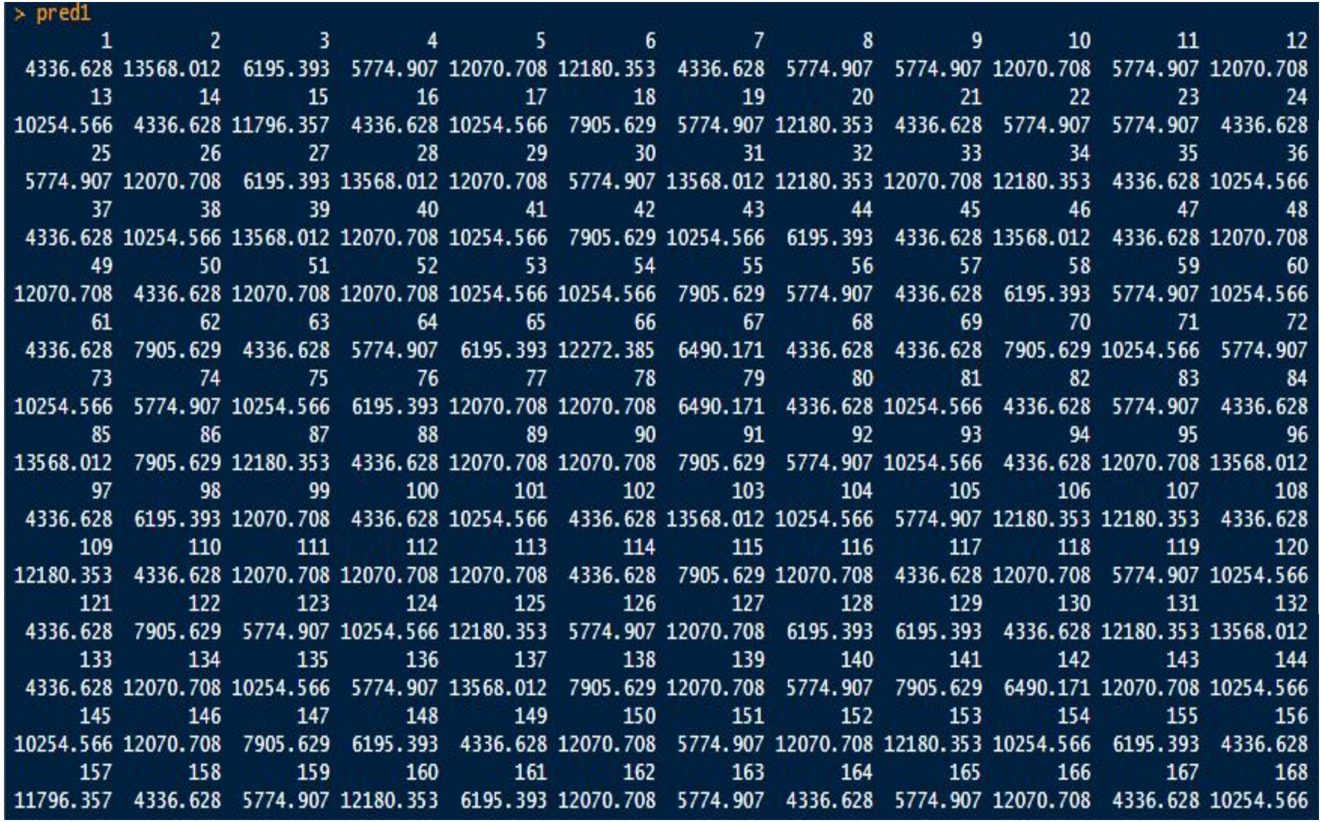


‘The rmse value for this model is 2021.95 and mape value is 0.21’ which not very good since the error value is more .From the tree we know that Duration,Airline were most important variables and then Total_stops and Days were variables considered.So from the tree a lot of people consider Duration as one of the important factor, also airline preference is clearly seen and also from Tableau we can can conclude a lot of people prefer morning or day_time flights .

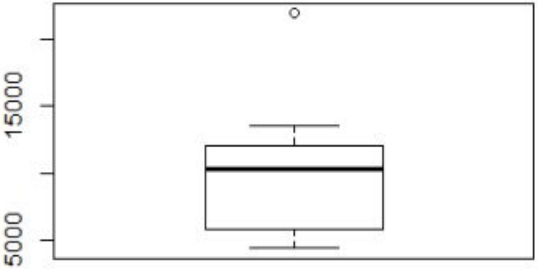
Model tree after tuning



Some of the predicted values and the boxplot for CART model

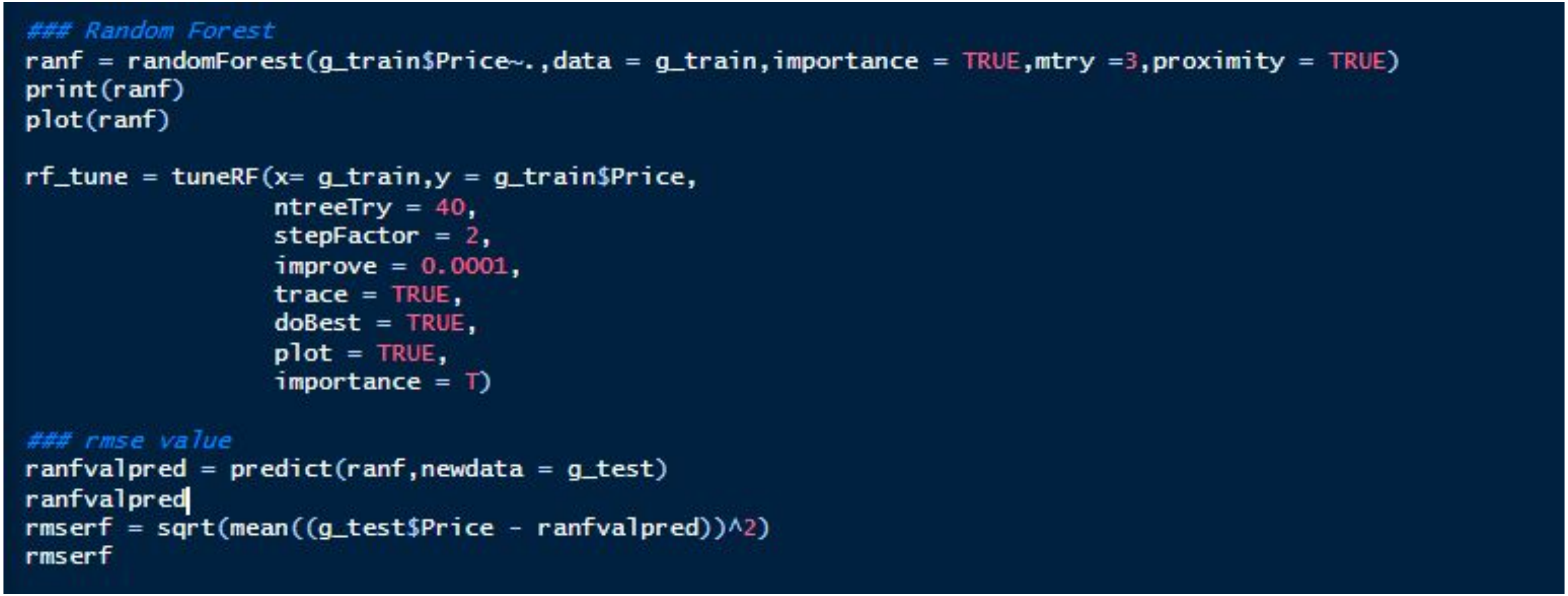


```
> pred1
        1         2         3         4         5         6         7         8         9        10        11        12 
 4336.628 13568.012  6195.393  5774.907 12070.708 12180.353  4336.628  5774.907  5774.907 12070.708  5774.907 12070.708 
       13        14        15        16        17        18        19        20        21        22        23        24 
10254.566  4336.628 11796.357  4336.628 10254.566  7905.629  5774.907 12180.353  4336.628  5774.907  5774.907  4336.628 
       25        26        27        28        29        30        31        32        33        34        35        36 
 5774.907 12070.708  6195.393 13568.012 12070.708  5774.907 13568.012 12180.353 12070.708 12180.353  4336.628 10254.566 
       37        38        39        40        41        42        43        44        45        46        47        48 
 4336.628 10254.566 13568.012 12070.708 10254.566  7905.629 10254.566  6195.393  4336.628 13568.012  4336.628 12070.708 
       49        50        51        52        53        54        55        56        57        58        59        60 
12070.708  4336.628 12070.708 12070.708 10254.566 10254.566  7905.629  5774.907  4336.628  6195.393  5774.907 10254.566 
       61        62        63        64        65        66        67        68        69        70        71        72 
 4336.628  7905.629  4336.628  5774.907  6195.393 12272.385  6490.171  4336.628  4336.628  7905.629 10254.566  5774.907 
       73        74        75        76        77        78        79        80        81        82        83        84 
10254.566  5774.907 10254.566  6195.393 12070.708 12070.708  6490.171  4336.628 10254.566  4336.628  5774.907  4336.628 
       85        86        87        88        89        90        91        92        93        94        95        96 
13568.012  7905.629 12180.353  4336.628 12070.708 12070.708  7905.629  5774.907 10254.566  4336.628 12070.708 13568.012 
       97        98        99       100       101       102       103       104       105       106       107       108 
 4336.628  6195.393 12070.708  4336.628 10254.566  4336.628 13568.012 10254.566  5774.907 12180.353 12180.353  4336.628 
      109       110       111       112       113       114       115       116       117       118       119       120 
12180.353  4336.628 12070.708 12070.708 12070.708  4336.628  7905.629 12070.708  4336.628 12070.708  5774.907 10254.566 
      121       122       123       124       125       126       127       128       129       130       131       132 
 4336.628  7905.629  5774.907 10254.566 12180.353  5774.907 12070.708  6195.393  6195.393  4336.628 12180.353 13568.012 
      133       134       135       136       137       138       139       140       141       142       143       144 
 4336.628 12070.708 10254.566  5774.907 13568.012  7905.629 12070.708  5774.907  7905.629  6490.171 12070.708 10254.566 
      145       146       147       148       149       150       151       152       153       154       155       156 
10254.566 12070.708  7905.629  6195.393  4336.628 12070.708  5774.907 12070.708 12180.353 10254.566  6195.393  4336.628 
      157       158       159       160       161       162       163       164       165       166       167       168 
11796.357  4336.628  5774.907 12180.353  6195.393 12070.708  5774.907  4336.628  5774.907 12070.708  4336.628 10254.566 
```



Random Forest Model

R Code



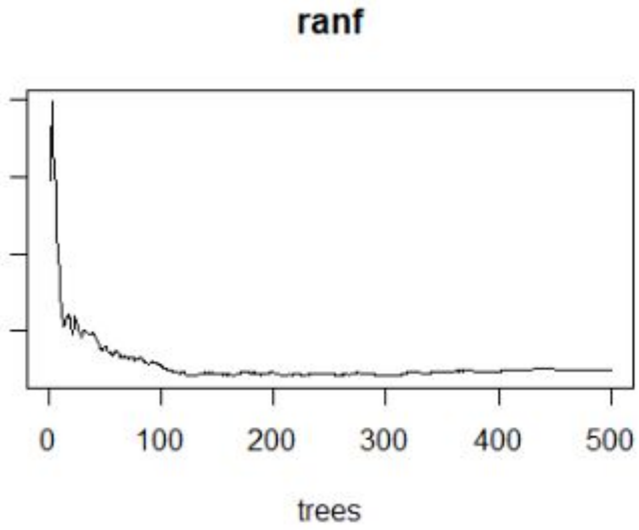
```
### Random Forest
ranf = randomForest(g_train$Price~.,data = g_train,importance = TRUE,mtry =3,proximity = TRUE)
print(ranf)
plot(ranf)

rf_tune = tuneRF(x= g_train,y = g_train$Price,
                 ntreeTry = 40,
                 stepFactor = 2,
                 improve = 0.0001,
                 trace = TRUE,
                 doBest = TRUE,
                 plot = TRUE,
                 importance = T)

### rmse value
ranfvalpred = predict(ranf,newdata = g_test)
ranfvalpred
rmserf = sqrt(mean((g_test$Price - ranfvalpred))^2)
rmserf
```

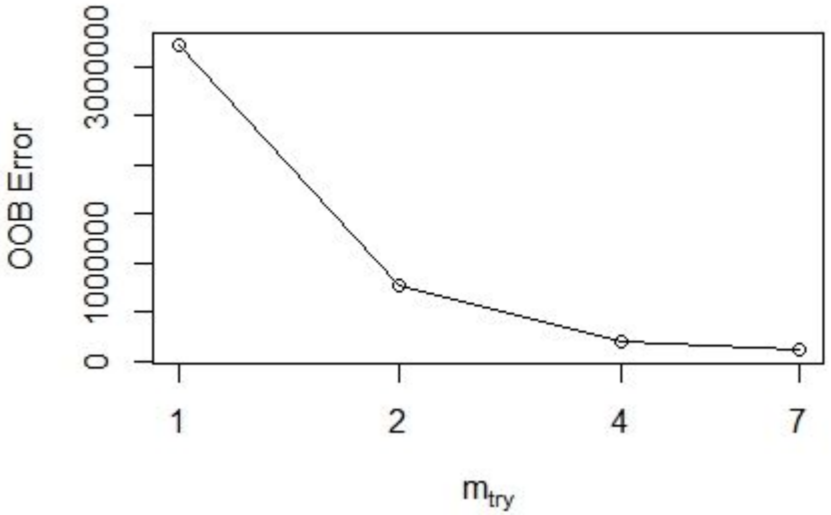
Similarly like the Cart model we build a Random forest model and tune it and then take the rmse value and then interpret the value of the model accordingly .

Error vs no. of trees

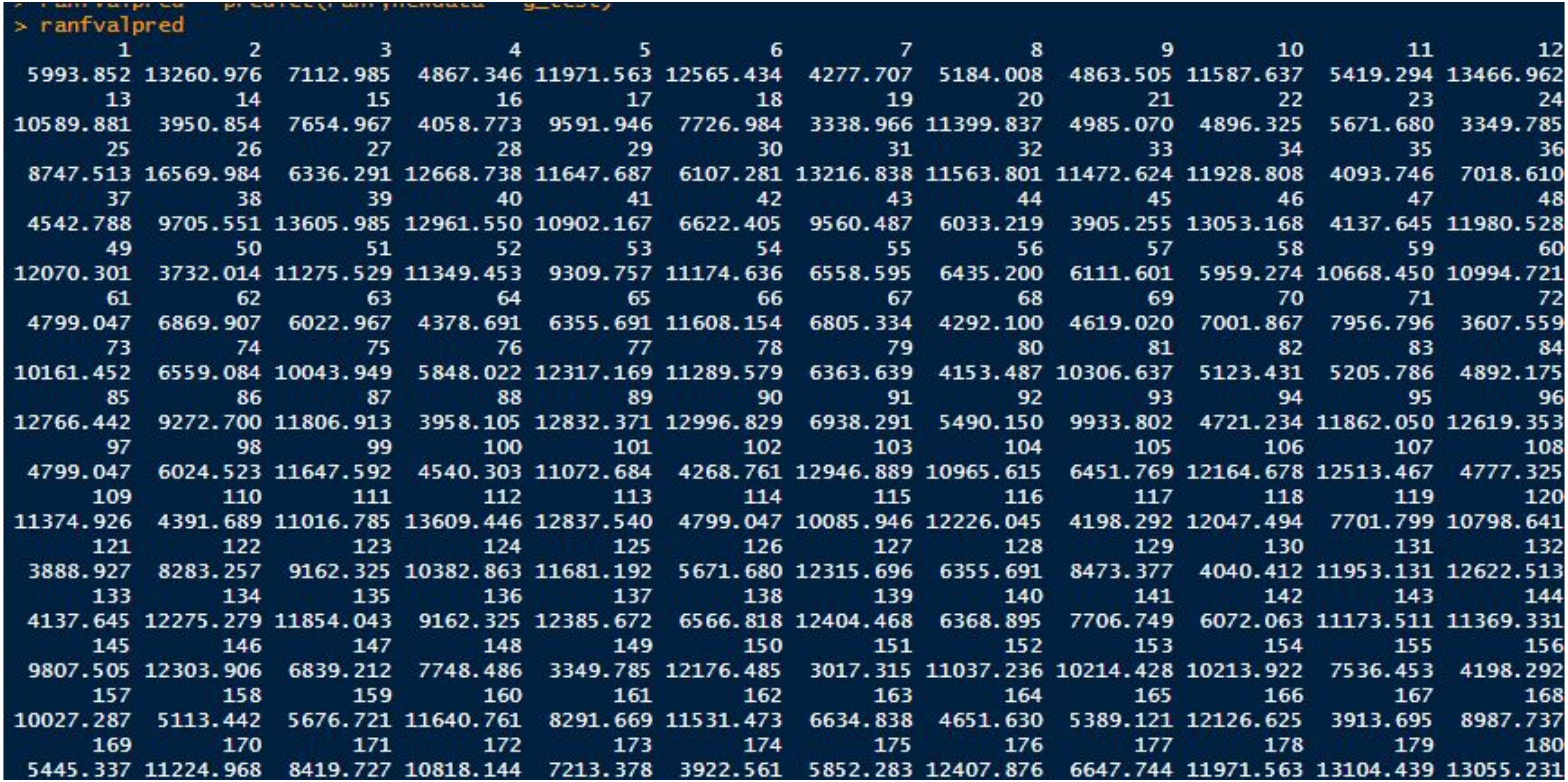
And after the tuning the OOB error gets reduced and

the model performance is known through rmse value





Predicted values for Price using Random Forest



```
> ranfvalpred <- predict(ranf,newdata = g_test)
> ranfvalpred
        1         2         3         4         5         6         7         8         9        10        11        12 
 5993.852 13260.976  7112.985  4867.346 11971.563 12565.434  4277.707  5184.008  4863.505 11587.637  5419.294 13466.962 
       13        14        15        16        17        18        19        20        21        22        23        24 
10589.881  3950.854  7654.967  4058.773  9591.946  7726.984  3338.966 11399.837  4985.070  4896.325  5671.680  3349.785 
       25        26        27        28        29        30        31        32        33        34        35        36 
 8747.513 16569.984  6336.291 12668.738 11647.687  6107.281 13216.838 11563.801 11472.624 11928.808  4093.746  7018.610 
       37        38        39        40        41        42        43        44        45        46        47        48 
 4542.788  9705.551 13605.985 12961.550 10902.167  6622.405  9560.487  6033.219  3905.255 13053.168  4137.645 11980.528 
       49        50        51        52        53        54        55        56        57        58        59        60 
12070.301  3732.014 11275.529 11349.453  9309.757 11174.636  6558.595  6435.200  6111.601  5959.274 10668.450 10994.721 
       61        62        63        64        65        66        67        68        69        70        71        72 
 4799.047  6869.907  6022.967  4378.691  6355.691 11608.154  6805.334  4292.100  4619.020  7001.867  7956.796  3607.559 
       73        74        75        76        77        78        79        80        81        82        83        84 
10161.452  6559.084 10043.949  5848.022 12317.169 11289.579  6363.639  4153.487 10306.637  5123.431  5205.786  4892.175 
       85        86        87        88        89        90        91        92        93        94        95        96 
12766.442  9272.700 11806.913  3958.105 12832.371 12996.829  6938.291  5490.150  9933.802  4721.234 11862.050 12619.353 
       97        98        99       100       101       102       103       104       105       106       107       108 
 4799.047  6024.523 11647.592  4540.303 11072.684  4268.761 12946.889 10965.615  6451.769 12164.678 12513.467  4777.325 
      109       110       111       112       113       114       115       116       117       118       119       120 
11374.926  4391.689 11016.785 13609.446 12837.540  4799.047 10085.946 12226.045  4198.292 12047.494  7701.799 10798.641 
      121       122       123       124       125       126       127       128       129       130       131       132 
 3888.927  8283.257  9162.325 10382.863 11681.192  5671.680 12315.696  6355.691  8473.377  4040.412 11953.131 12622.513 
      133       134       135       136       137       138       139       140       141       142       143       144 
 4137.645 12275.279 11854.043  9162.325 12385.672  6566.818 12404.468  6368.895  7706.749  6072.063 11173.511 11369.331 
      145       146       147       148       149       150       151       152       153       154       155       156 
 9807.505 12303.906  6839.212  7748.486  3349.785 12176.485  3017.315 11037.236 10214.428 10213.922  7536.453  4198.292 
      157       158       159       160       161       162       163       164       165       166       167       168 
10027.287  5113.442  5676.721 11640.761  8291.669 11531.473  6634.838  4651.630  5389.121 12126.625  3913.695  8987.737 
      169       170       171       172       173       174       175       176       177       178       179       180 
 5445.337 11224.968  8419.727 10818.144  7213.378  3922.561  5852.283 12407.876  6647.744 11971.563 13104.439 13055.231 
```

The rmse value for this model is 146.8 which is really good considering the error rate is very less.since in the tuning we used step-factor method which is an ensemble method to boost the performance .

Interpretation of the best model

According to rmse performance metric Random Forest > linear regression>CART

Since the rmse value for random forest is very less (only 146.8),so since the error value is less, this model is the best model for this particular data and then linear regression model which had an rmse value of 1239 and then CART with highest error value of 2200

According to MAPE performance metric too it is in the same order with Random forest being a very good model and CART model underperformed .the linear regression model gave a prediction of .77 % accuracy