Final Project: Cars Price Prediction

MATH 40028/50028: Statistical Learning

Paritosh Gandre

INTRODUCTION

The dataset is about Indian Car market with more than 8,000 observations and 13 variables which contain Company name, Year,selling price, kilometers driven, fuel type, seller type, transmission, owner, mileage, engine, maximum power, torque, and number of seats.
In this project, our target variable i.e. our response variable will be **"selling_price"** of the vehicle. We will predict selling price of a vehicle based on other variable inputs.
We need to clean the dataset as it has null values and change types of few columns and then we can use this dataset.
We will first split the dataset into training and testing split. Using these splits we will train a Linear, Random forest and Gradient boosting models and evaluate them based on RMSE. There are few combinations of splitting like 70:30, 75:25, 80:20 splits, but we will use 70:30 split that is 70% of the data will be used as a training set and remaining 30% will be used as a testing set. Once we find a optimal model, we will do our prediction on unseen data and will evaluate that too.

```
data = read.csv("cars_price.csv")
dim(data)
```

```
## [1] 8128   13
```

```
head(data,5)
```

```
##                           name year selling_price km_driven   fuel seller_type
## 1       Maruti Swift Dzire VDI 2014        450000    145500 Diesel  Individual
## 2 Skoda Rapid 1.5 TDI Ambition 2014        370000    120000 Diesel  Individual
## 3     Honda City 2017-2020 EXi 2006        158000    140000 Petrol  Individual
## 4    Hyundai i20 Sportz Diesel 2010        225000    127000 Diesel  Individual
## 5       Maruti Swift VXI BSIII 2007        130000    120000 Petrol  Individual
##   transmission        owner    mileage  engine  max_power
## 1       Manual  First Owner  23.4 kmpl 1248 CC     74 bhp
## 2       Manual Second Owner 21.14 kmpl 1498 CC 103.52 bhp
## 3       Manual  Third Owner  17.7 kmpl 1497 CC     78 bhp
## 4       Manual  First Owner  23.0 kmpl 1396 CC     90 bhp
## 5       Manual  First Owner  16.1 kmpl 1298 CC   88.2 bhp
##                     torque seats
## 1           190Nm@ 2000rpm     5
```

```
## 2       250Nm@ 1500-2500rpm     5
## 3     12.7@ 2,700(kgm@ rpm)     5
## 4 22.4 kgm at 1750-2750rpm     5
## 5     11.5@ 4,500(kgm@ rpm)     5
```

```
sum(is.na(data))
```

```
## [1] 221
```

The dataset is about Indian Car market with more than **8,000** observations and **13** variables which contain Company name, Year,selling price, kilometers driven, fuel type, seller type, transmission, owner, mileage, engine, maximum power, torque, and number of seats with **221** null values.
In this project, our target variable i.e. our response variable will be selling price of the vehicle. We will predict selling price of a vehicle based on other variable inputs.
We need to clean the dataset as it has null values and change types of few columns and then we can use this dataset.
We will first split the dataset into training and testing split. Using these splits we will train a few model and evaluate them based on few metrics. There are few combinations of splitting like 70:30, 75:25, 80:20 splits, but we will use 70:30 split that is 70% of the data will be used as a training set and remaining 30% will be used as a testing set. Once we find a optimal model, we will do our prediction on unseen data and will evaluate that too.

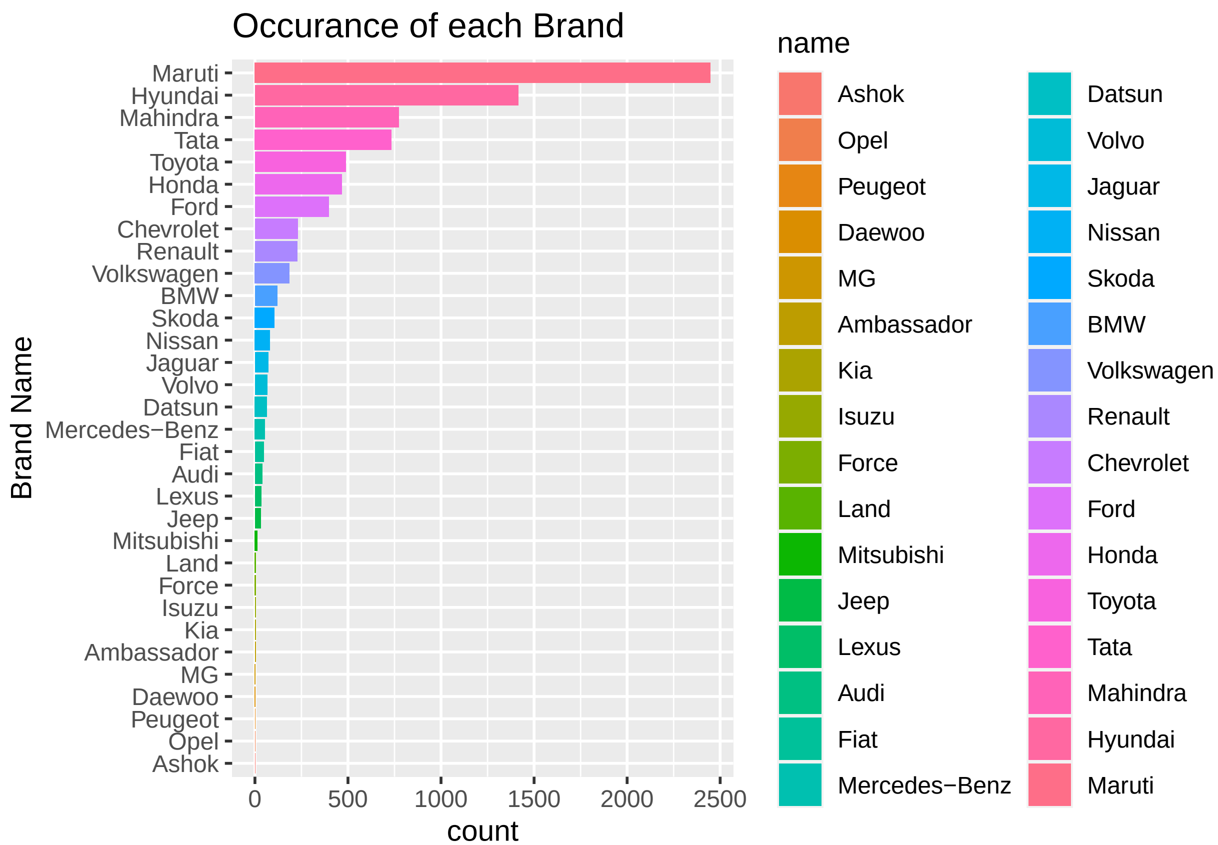
Statistical learning strategies and methods

Exploratory Data Analysis

splitting company names and storing first word

```
data$name = sapply(strsplit(data$name," "),`[`,1)
data = subset(data, select = -c(12))
```

Plotting number of occurance of each brand



Converting name column to numerical columns by assigning numbers to each brand

Replacing blanks with NA values

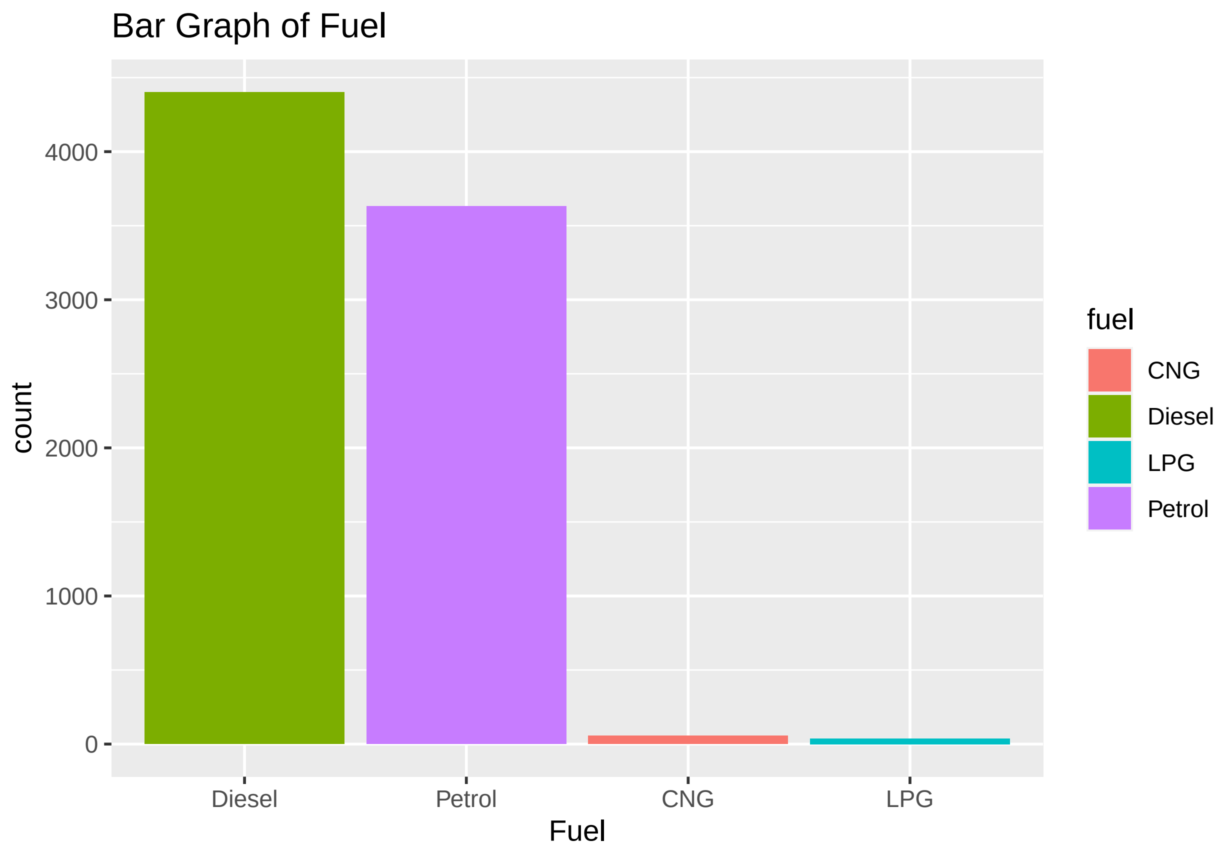
```
data$mileage[data$mileage == ""] = NA
data$engine[data$engine == ""] = NA
data$max_power[data$max_power == ""] = NA
```

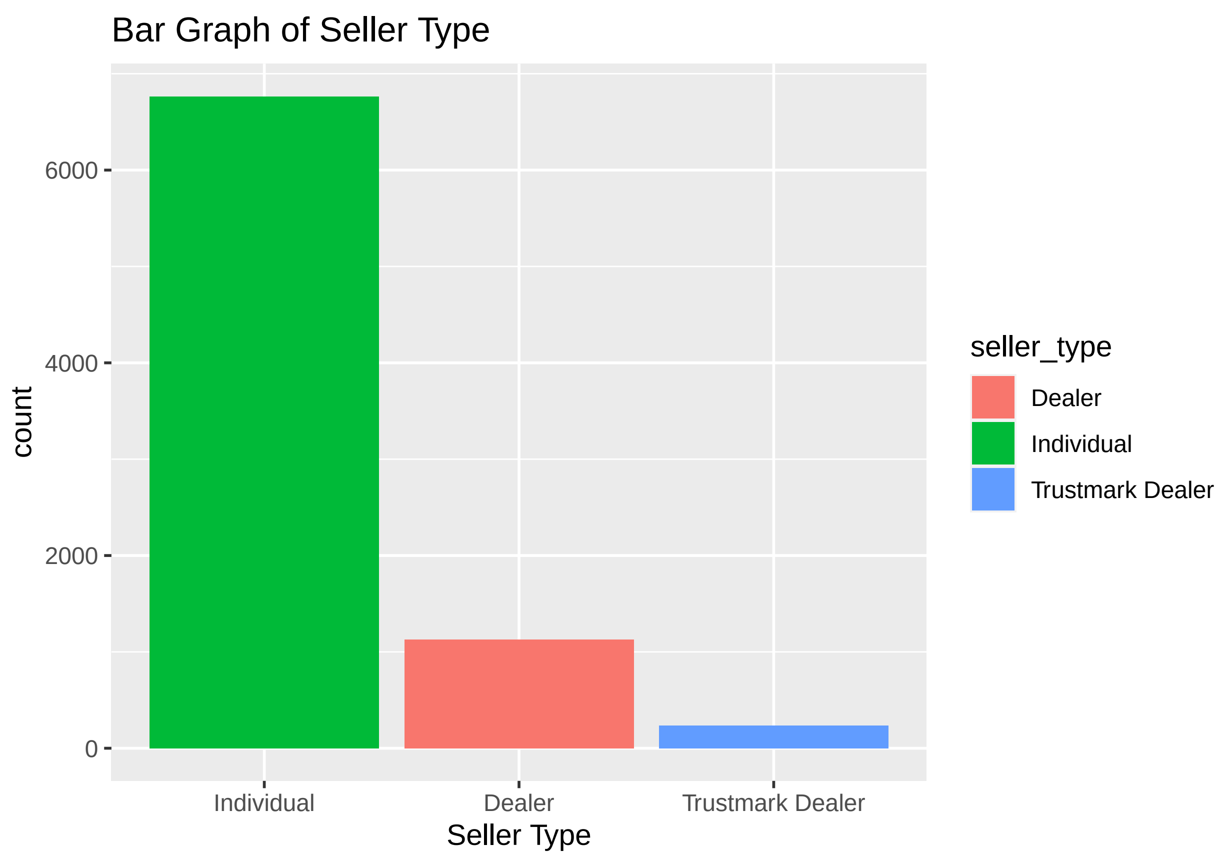
Cleaning and Converting categorical columns to numerical columns

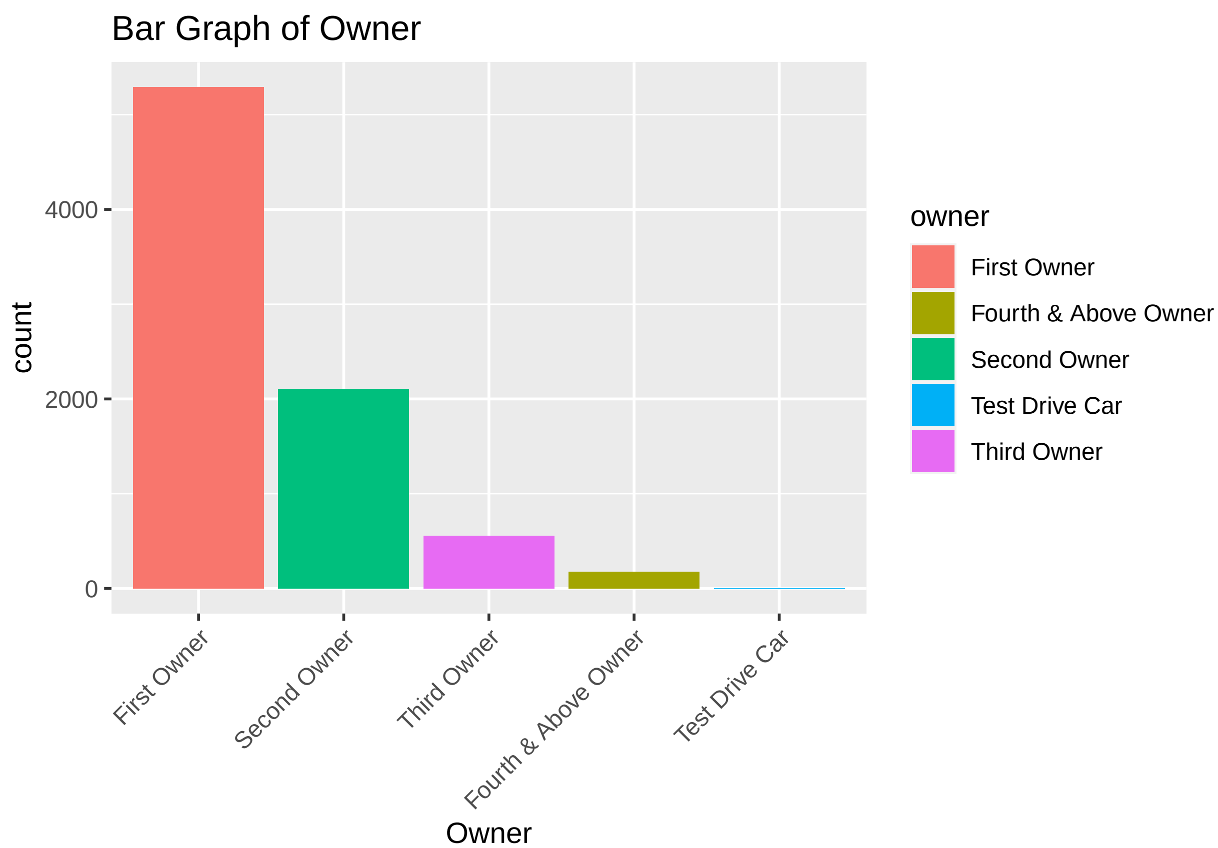
```
sum(is.na(data))
```

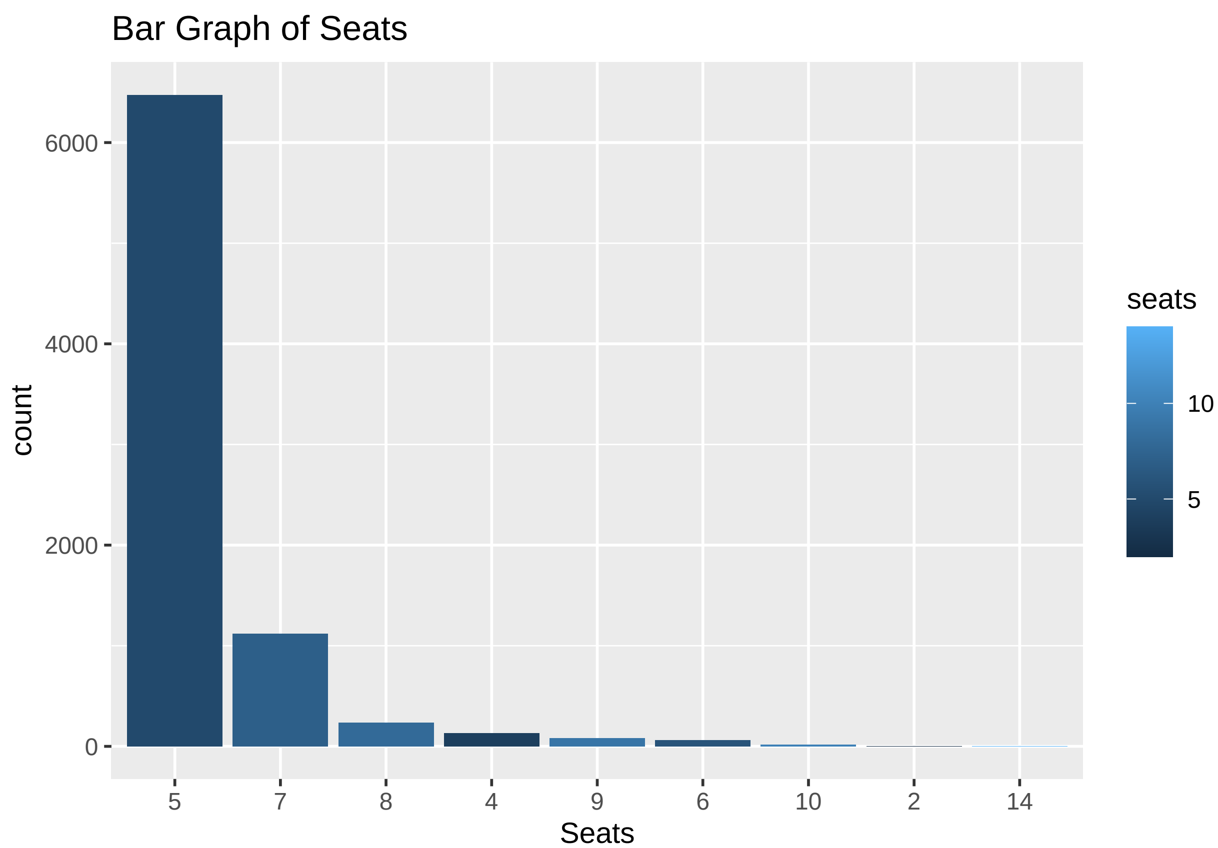
```
## [1] 0
```

Plotting distribution of vehicles by fuel type, seller types, number of seats, and transmission






Bar Graph of Owner
count
4000
2000
0
First Owner
Second Owner
Third Owner
Fourth & Above Owner
Test Drive Car
Owner
owner
First Owner
Fourth & Above Owner
Second Owner
Test Drive Car
Third Owner



Converting transmission, owner, seller type and fuel to 0's and 1's

```
table(data$transmission)
```

```
##
##    0    1
## 7078 1050
```

```
table(data$owner)
```

```
##
##    0    1    2    3    4
## 5289 2105  555  174    5
```

```
table(data$seller_type)
```

```
##
##    0    1    2
##  236 1126 6766
```

```
table(data$fuel)
```

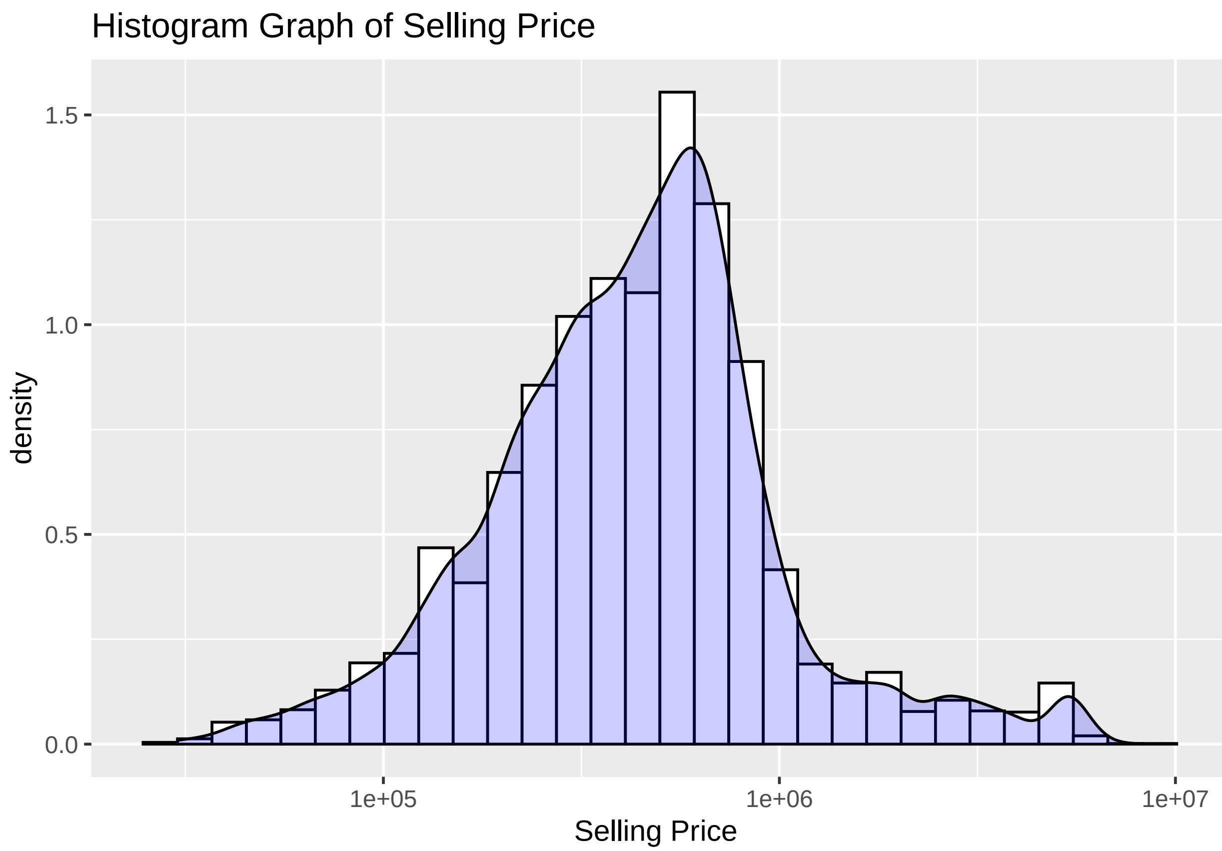
```
##
##    0    1    2    3
```

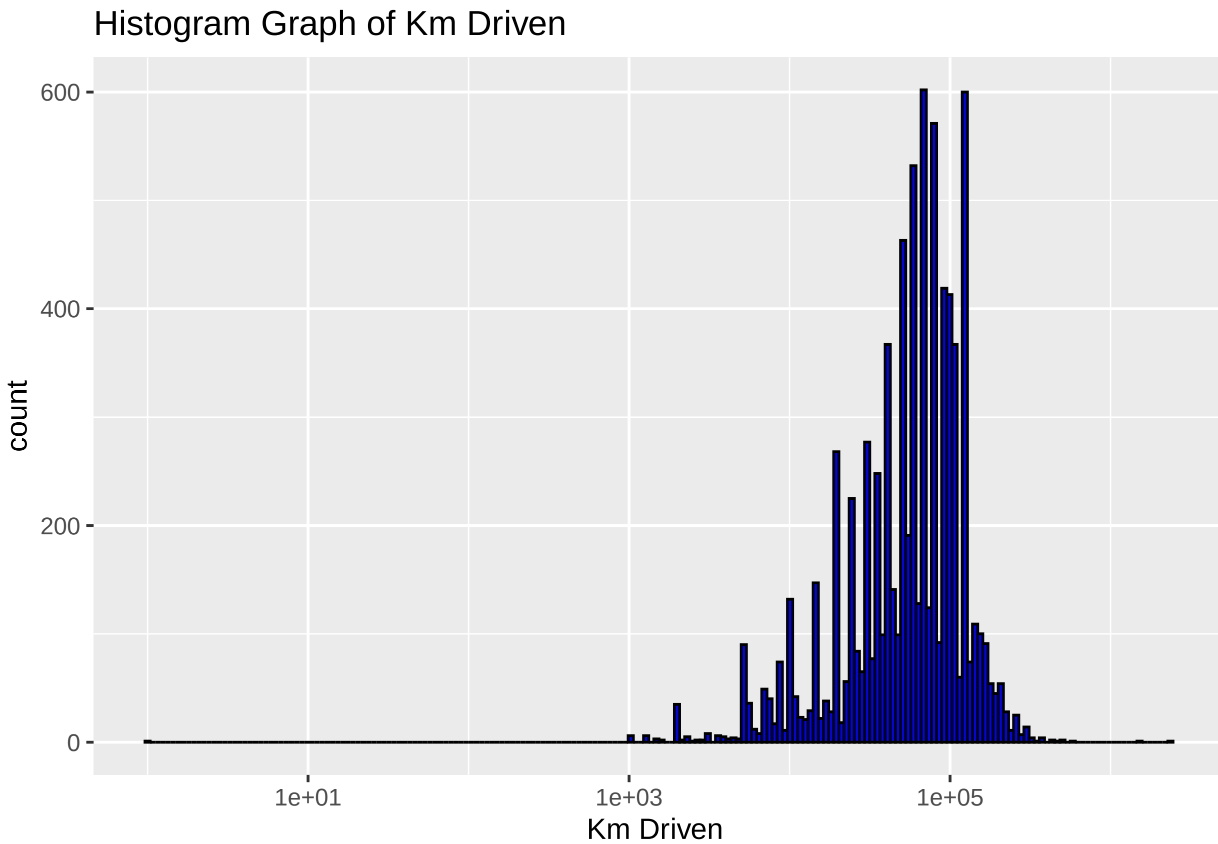
```
## 4402 3631   57   38
```

Distribution of Selling price, and Kilometers Driven

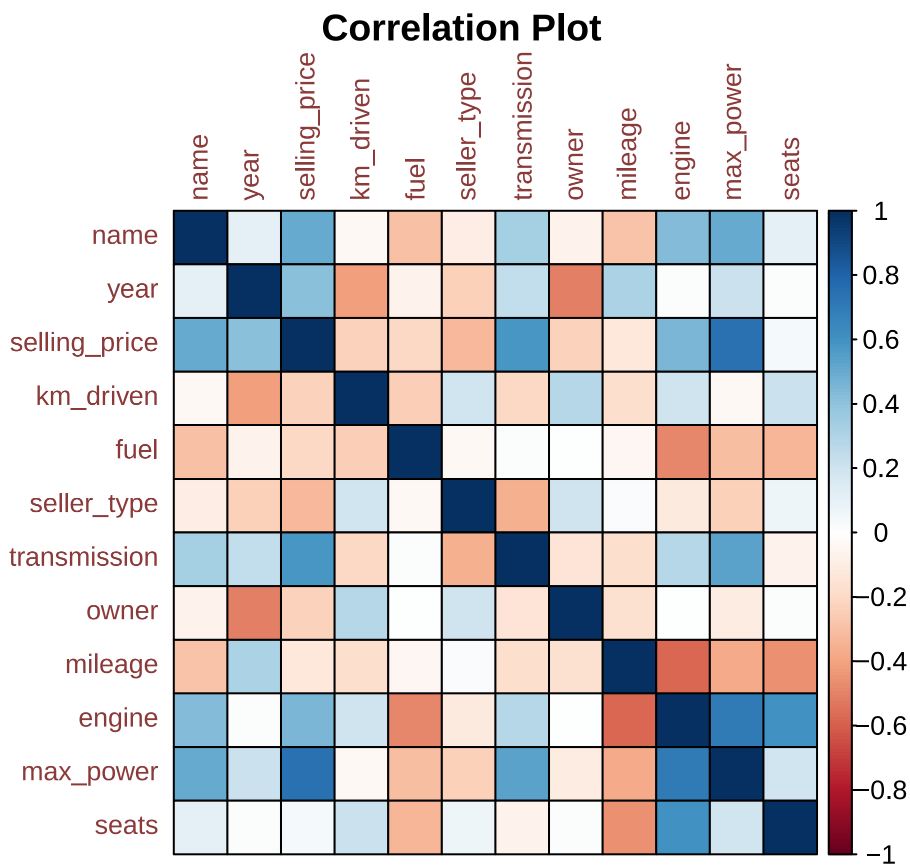
```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




Histogram Graph of Km Driven
600
400
200
0
count
1e+01
1e+03
1e+05
Km Driven

Correlation between variables with CORRPLOT library




```
##                name  year selling_price km_driven  fuel seller_type
## name           1.00  0.12          0.50     -0.03 -0.29       -0.10
## year           0.12  1.00          0.41     -0.42 -0.06       -0.23
## selling_price  0.50  0.41          1.00     -0.23 -0.21       -0.32
## km_driven     -0.03 -0.42         -0.23      1.00 -0.24        0.19
## fuel          -0.29 -0.06         -0.21     -0.24  1.00       -0.03
## seller_type   -0.10 -0.23         -0.32      0.19 -0.03        1.00
## transmission   0.34  0.24          0.59     -0.20  0.01       -0.36
## owner         -0.06 -0.50         -0.22      0.28  0.00        0.20
## mileage       -0.28  0.31         -0.13     -0.17 -0.04        0.02
## engine         0.43  0.02          0.45      0.20 -0.48       -0.12
## max_power      0.51  0.21          0.74     -0.04 -0.30       -0.24
## seats          0.11  0.01          0.05      0.22 -0.34        0.07
##               transmission owner mileage engine max_power seats
## name                  0.34 -0.06   -0.28   0.43      0.51  0.11
## year                  0.24 -0.50    0.31   0.02      0.21  0.01
## selling_price         0.59 -0.22   -0.13   0.45      0.74  0.05
## km_driven            -0.20  0.28   -0.17   0.20     -0.04  0.22
## fuel                  0.01  0.00   -0.04  -0.48     -0.30 -0.34
## seller_type          -0.36  0.20    0.02  -0.12     -0.24  0.07
```

```
## transmission          1.00 -0.14   -0.18   0.28      0.54 -0.07
## owner                -0.14  1.00   -0.17   0.01     -0.10  0.02
## mileage              -0.18 -0.17    1.00  -0.58     -0.37 -0.45
## engine                0.28  0.01   -0.58   1.00      0.70  0.61
## max_power             0.54 -0.10   -0.37   0.70      1.00  0.19
## seats                -0.07  0.02   -0.45   0.61      0.19  1.00
```

We can see that selling price is highly correlated to engine, max_power, name, and transmission, with year as well.

Splitting of Dataset into 70 and 30 split

```
set.seed(5)
trainIndex = createDataPartition(data$selling_price, p = .7,
                                  list = FALSE,
                                  times = 1)
train_data = data[ trainIndex,]
test_data = data[-trainIndex,]
```

Linear Regression

```
##
## Call:
## lm(formula = selling_price ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2464480  -211652    -5224   167360  3952046
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5.854e+07  4.256e+06 -13.753  < 2e-16 ***
## name           2.444e+04  1.426e+03  17.134  < 2e-16 ***
## year           2.862e+04  2.127e+03  13.455  < 2e-16 ***
## km_driven     -1.585e+00  1.608e-01  -9.855  < 2e-16 ***
## fuel           1.777e+04  1.532e+04   1.160 0.246009
## seller_type   -9.833e+04  1.407e+04  -6.990 3.07e-12 ***
## transmission   4.348e+05  2.307e+04  18.844  < 2e-16 ***
## owner         -3.004e+03  9.705e+03  -0.310 0.756893
## mileage        2.360e+04  2.422e+03   9.744  < 2e-16 ***
## engine         9.283e+01  2.682e+01   3.461 0.000542 ***
## max_power      1.243e+04  3.001e+02  41.427  < 2e-16 ***
## seats         -1.293e+04  9.401e+03  -1.376 0.169011
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 449100 on 5679 degrees of freedom
## Multiple R-squared:  0.696,  Adjusted R-squared:  0.6954
## F-statistic:  1182 on 11 and 5679 DF,  p-value: < 2.2e-16
```

We can get rid of Fuel, Owner and Seats as they are least significant for the model. Now we train our model with name,year,km_driven,seller_type,mileage,transmission,max_power and evaluate the model

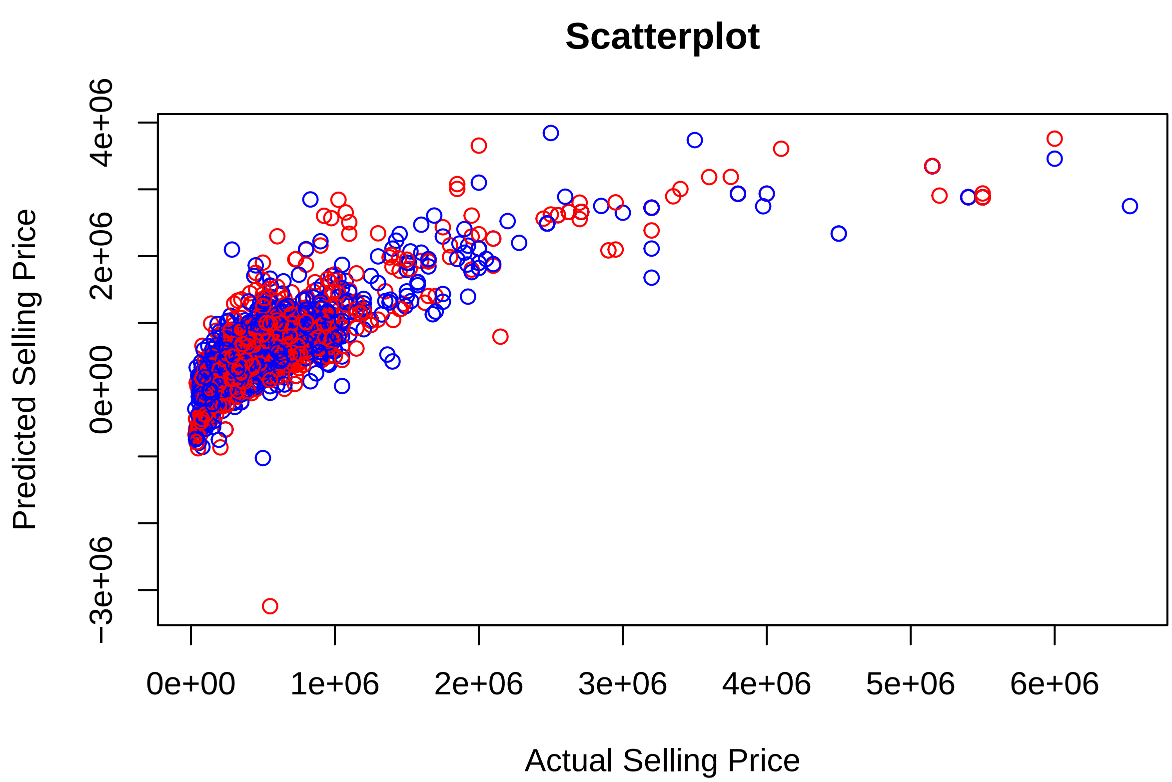
```
lm1_data = lm(selling_price ~ name+year+km_driven+seller_type+mileage+transmission+max_power, da
```

using this Linear Regression model to predict

```
pred_lr = predict(lm1_data, newdata = test_data)
error_lr = (test_data$selling_price - pred_lr)
RMSE_lr = sqrt(mean(error_lr^2))
print(paste("RMSE LINEAR REGRESSION: ",RMSE_lr))
```

```
## [1] "RMSE LINEAR REGRESSION:  457916.921763458"
```

now we plot the predicted values and actual values



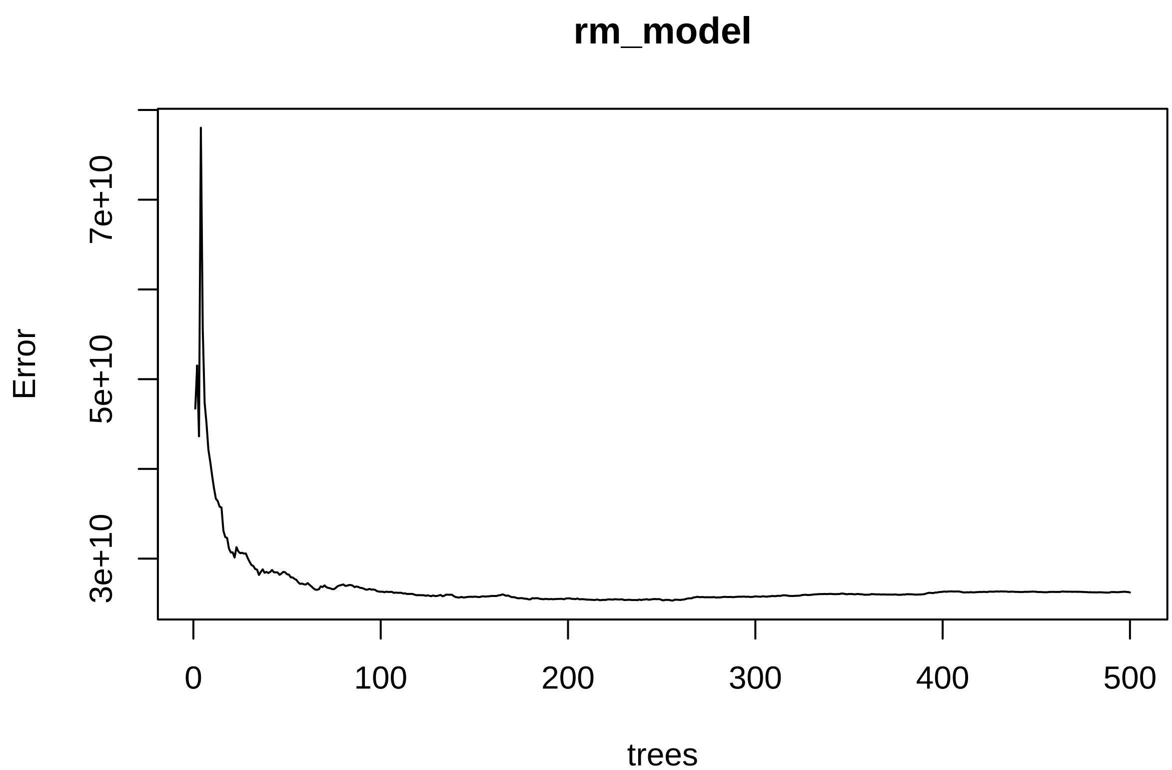
With Linear Regression we got 4.5791692×10^5 of RMSE.

Model 2 : Random Forest

```
rm_model= randomForest(selling_price ~ ., data = train_data)
rm_model
```

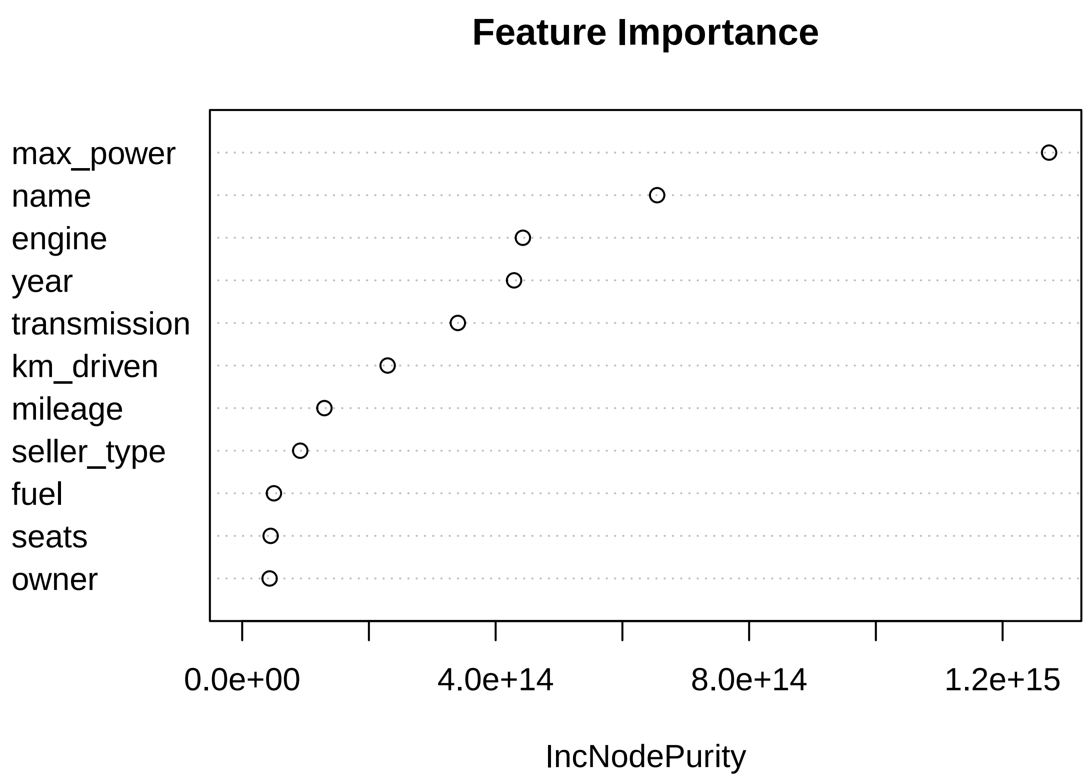
```
##
## Call:
##  randomForest(formula = selling_price ~ ., data = train_data)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 26231797034
##                     % Var explained: 96.04
```

```
plot(rm_model)
```



Plotting feature importance

```
varImpPlot(rm_model, main = "Feature Importance")
```



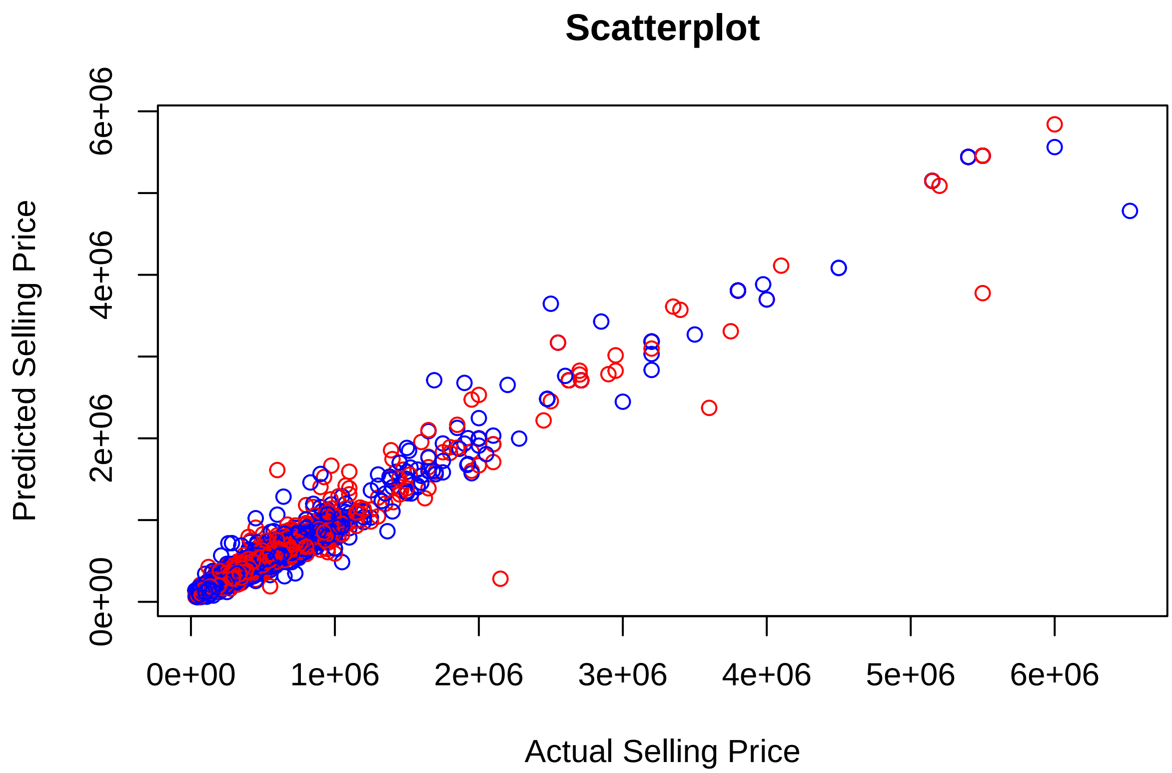
Using Random Forest model on Test Dataset

```
pred_rf = predict(rm_model, test_data)
error_rm = test_data$selling_price - pred_rf
rmse_rm = sqrt(mean(error_rm^2))
print(paste('Random Forest RMSE: ', rmse_rm))
```

```
## [1] "Random Forest RMSE:  128703.952584409"
```

plotting of predicted values from Random forest and actual values

```
plot(test_data$selling_price,pred_rf, main="Scatterplot", col = c("red","blue"), xlab = "Actual
```



With Random Forest we got 1.2870395×10^5 of RMSE.

Model 3 : Gradient Boosting

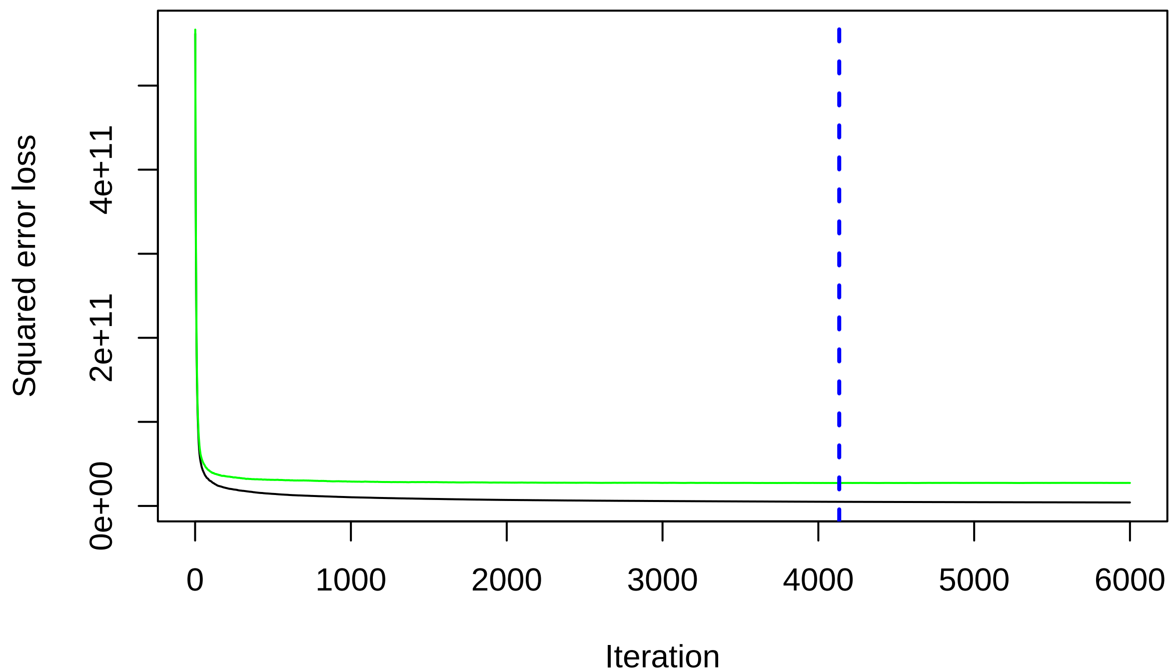
```
gbm_model = gbm(formula = selling_price ~ .,
                distribution = "gaussian",
                data = train_data,
                n.trees = 6000,
                interaction.depth = 3,
                shrinkage = 0.1,
                cv.folds = 5,
                n.cores = NULL, # will use all cores by default
                verbose = FALSE)

gbm_model
```

```
## gbm(formula = selling_price ~ ., distribution = "gaussian", data = train_data,
##     n.trees = 6000, interaction.depth = 3, shrinkage = 0.1, cv.folds = 5,
##     verbose = FALSE, n.cores = NULL)
## A gradient boosted model with gaussian loss function.
## 6000 iterations were performed.
```

```
## The best cross-validation iteration was 4134.
## There were 11 predictors of which 11 had non-zero influence.
```

Plotting loss function as a result of n tress added to the ensemble

```r
gbm.perf(gbm_model, method = "cv")
```

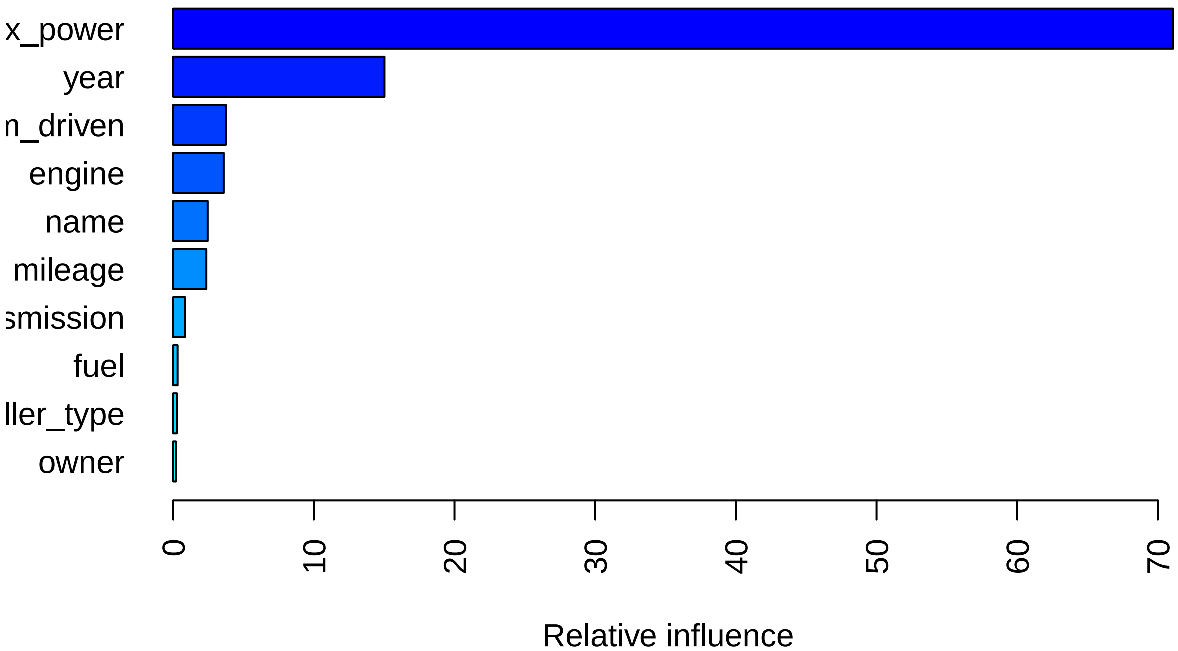



```
## [1] 4134
```

Variable Importance

cBars = 10: This option defines the number of confidence bars to show in the summary plot. Confidence bars are used to show the uncertainty in the estimated values. In this case, it is set to 10, indicating that the summary will include 10 confidence bars.
method = relative.influence: This argument sets the method for calculating variable importance. In this case, it is set to "relative.influence", a strategy typically used in GBM models to quantify predictors' relative importance. It calculates each predictor's influence on the response variable in comparison to the other predictors.
las = 2: This option specifies the orientation of the axis labels in the summary graphic. A value of two indicates that the labels are parallel to the axis.

```r
summary(gbm_model, cBars = 10, method = relative.influence, las =2)
```




```
##                       var    rel.inf
## max_power       max_power 71.0818582
## year                 year 15.0189252
## km_driven       km_driven  3.7381625
## engine             engine  3.5897364
## name                 name  2.4515798
## mileage           mileage  2.3565632
## transmission transmission  0.8353735
## fuel                 fuel  0.3124380
## seller_type   seller_type  0.2620084
## owner               owner  0.1910715
## seats               seats  0.1622834
```

Using the model to predict selling price in the Test dataset

```
pred_gbm = predict(gbm_model, test_data)
```

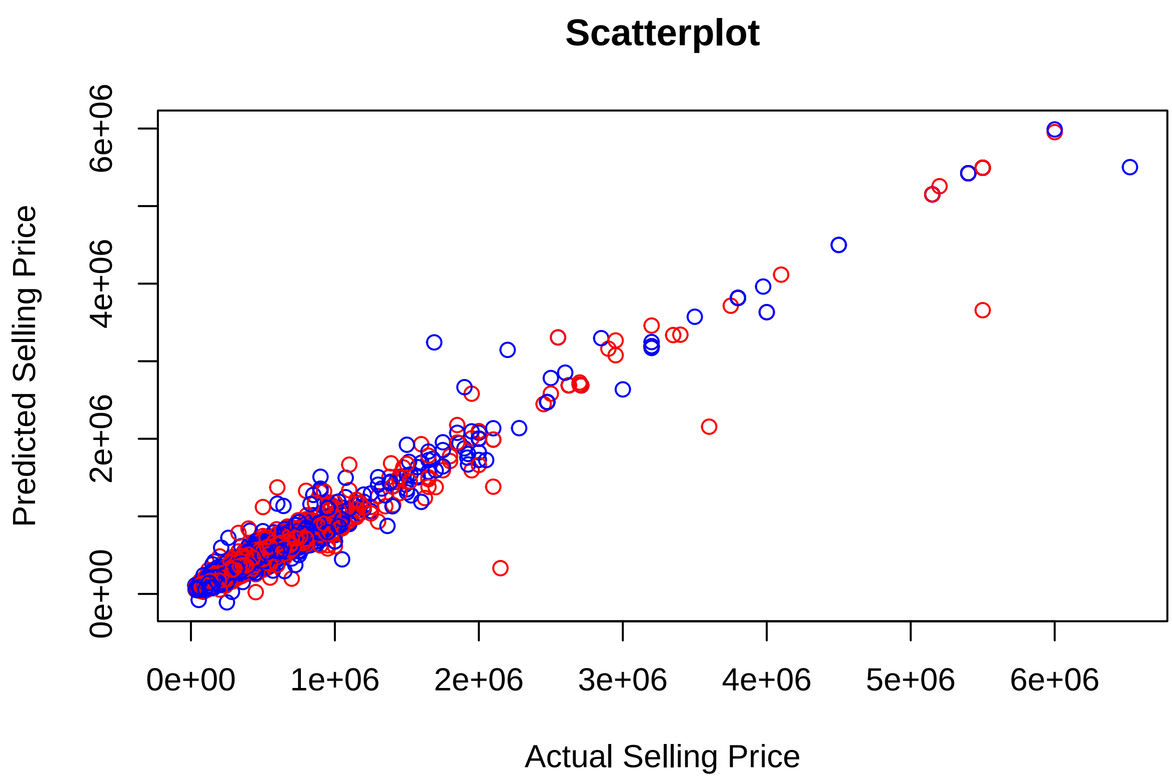
```
## Using 4134 trees...
```

```
error_gbm <- test_data$selling_price - pred_gbm
RMSE_gbm <- sqrt(mean(error_gbm^2))
print(paste("RMSE for Gradient Boosting Model: ", RMSE_gbm))
```

```
## [1] "RMSE for Gradient Boosting Model:  125738.566422292"
```

Plotting predicted and actual values

```
plot(test_data$selling_price,pred_gbm,
     main="Scatterplot",
     col = c("red","blue"),
     xlab = "Actual Selling Price", ylab = "Predicted Selling Price")
```



With Gradient Boosting we got 1.2573857×10^5 of RMSE.

Conclusion

We implemented Linear regression, Random forest, and Gradient Boosting model on the Cars dataset for the prediction of selling price.
After computing all the models we have also evaluated each model based on it's RMSE value. Overall, we can see Gradient Boosting with the least RMSE which indicates Gradient boosting is better than linear regression and random forest at predicting the selling price of a vehicle.
Linear Regression RMSE: 4.5791692×10^5* * **Random Forest RMSE :** 1.2870395×10^5 * * Gradient Boosting RMSE : 1.2573857×10^5

Gradient Boosting surpassed Linear Regression and Random Forest, resulting in the lowest RMSE of **1.2573857 × 10ˆ5**. This shows that Gradient Boosting is a better fit for forecasting vehicle selling prices in this dataset than the other models. Gradient Boosting's RMSE was significantly lower than Linear Regression **(4.5791692 × 10ˆ5)** and Random Forest **(1.2870395 × 10ˆ5)**, showing greater predictive accuracy. As a result, for effectively projecting car prices in this context, Gradient Boosting emerges as the best option among the three models.