Car Price Prediction Project

Sweta Swarupa

05/04/22

Contents

1 Introduction:

The aim of this project is to build algorithms to predict selling price of cars. The dataset is taken from Kaggle. Data Source: https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho?select=Car+details+v3.csv

I will use three different models, linear regression, random forest and gradient boosting to predict selling prices of cars and compare the results.

1.1 Attribute Information:

name - Name of the cars

year - Year of the car when it was bought

selling_price - Price at which the car is being sold

km_driven - Number of Kilometers the car is driven

fuel - Fuel type of car (petrol / diesel / CNG / LPG / electric)

seller_type - Tells if a Seller is Individual or a Dealer

transmission - Gear transmission of the car (Automatic/Manual)

Owner - Number of previous owners of the car.

```
#Reading the data
car<- read.csv("https://raw.githubusercontent.com/swetaswarupa/Car-Price-Prediction/main/Car%20details%20v
str(car)
## 'data.frame':    8128 obs. of  13 variables:
##  $ name         : chr  "Maruti Swift Dzire VDI" "Skoda Rapid 1.5 TDI Ambition" "Honda City 2017-2020 EX
##  $ year         : int  2014 2014 2006 2010 2007 2017 2007 2001 2011 2013 ...
##  $ selling_price: int  450000 370000 158000 225000 130000 440000 96000 45000 350000 200000 ...
##  $ km_driven    : int  145500 120000 140000 127000 120000 45000 175000 5000 90000 169000 ...
##  $ fuel         : chr  "Diesel" "Diesel" "Petrol" "Diesel" ...
##  $ seller_type  : chr  "Individual" "Individual" "Individual" "Individual" ...
##  $ transmission : chr  "Manual" "Manual" "Manual" "Manual" ...
##  $ owner        : chr  "First Owner" "Second Owner" "Third Owner" "First Owner" ...
##  $ mileage      : chr  "23.4 kmpl" "21.14 kmpl" "17.7 kmpl" "23.0 kmpl" ...
##  $ engine       : chr  "1248 CC" "1498 CC" "1497 CC" "1396 CC" ...
##  $ max_power    : chr  "74 bhp" "103.52 bhp" "78 bhp" "90 bhp" ...
##  $ torque       : chr  "190Nm@ 2000rpm" "250Nm@ 1500-2500rpm" "12.7@ 2,700(kgm@ rpm)" "22.4 kgm at 1750
##  $ seats        : int  5 5 5 5 5 5 5 4 5 5 ...
```

A portion of the car data set is shown below:

Table 1: Car Data

name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine
Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC
Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC
Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC
Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC
Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC

There are 8128 rows and 13 variables. Our target variable is the selling_price, which signifies the price of the car. We will use other variables to predict selling_price.

2 Data Exploration, Data Cleaning and Data Transformation

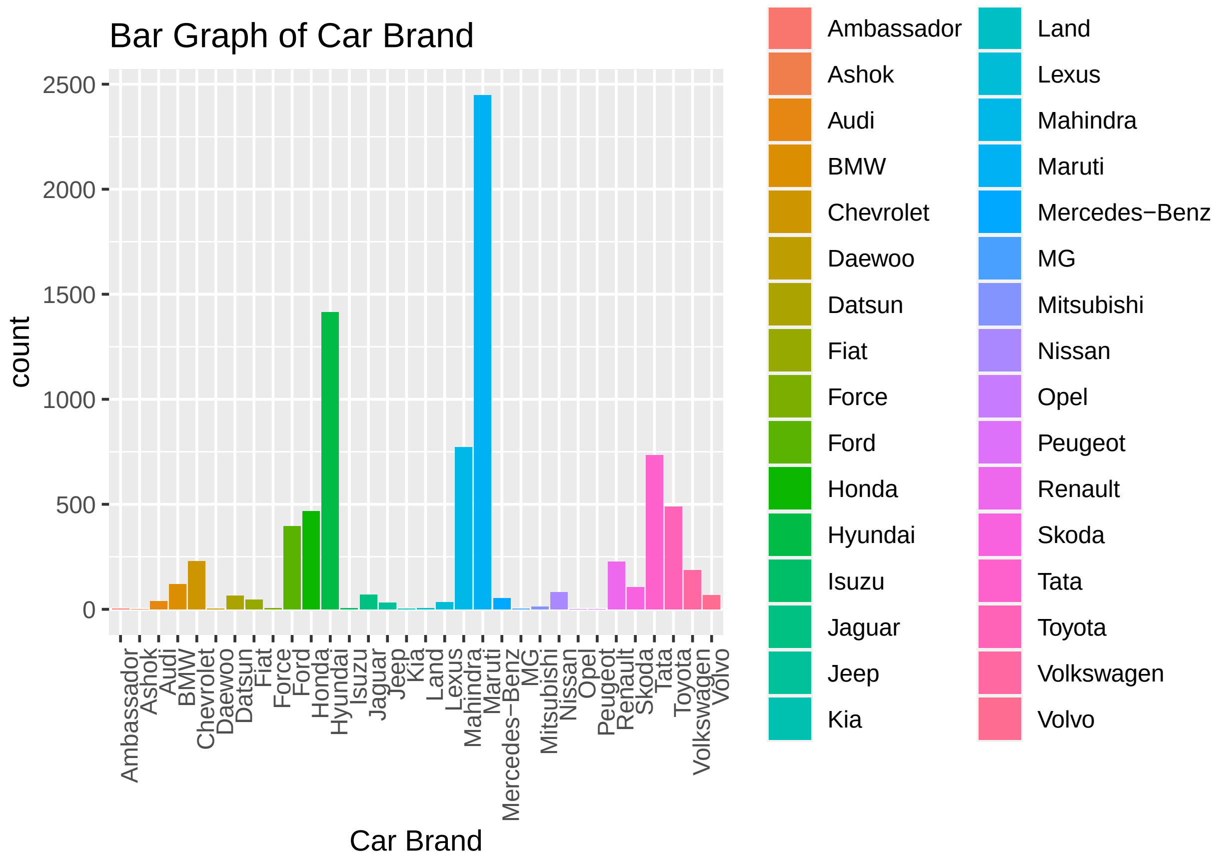
2.1 Car name variable

```r
#Extracting brand name from car name

car$name <- sapply(strsplit(car$name, " "), `[`, 1)

#Plotting car name to check the distribution

ggplot(data = car, aes(x=name, fill = name)) +
  geom_bar() + labs(x='Car Brand') + labs(title = "Bar Graph of Car Brand") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
#Converting car name into Ordinal Encoder

car$name <- str_replace(car$name, 'Maruti', '0')
car$name <- str_replace(car$name, 'Skoda', '1')
car$name <- str_replace(car$name, 'Honda', '2')
car$name <- str_replace(car$name, 'Hyundai', '3')
car$name <- str_replace(car$name, 'Toyota', '4')
car$name <- str_replace(car$name, 'Ford', '5')
car$name <- str_replace(car$name, 'Renault', '6')
car$name <- str_replace(car$name, 'Mahindra', '7')
car$name <- str_replace(car$name, 'Tata', '8')
car$name <- str_replace(car$name, 'Chevrolet', '9')
car$name <- str_replace(car$name, 'Fiat', '10')
car$name <- str_replace(car$name, 'Datsun', '11')
car$name <- str_replace(car$name, 'Jeep', '12')
car$name <- str_replace(car$name, 'Mercedes-Benz', '13')
car$name <- str_replace(car$name, 'Mitsubishi', '14')
car$name <- str_replace(car$name, 'Audi', '15')
car$name <- str_replace(car$name, 'Volkswagen', '16')
car$name <- str_replace(car$name, 'BMW', '17')
car$name <- str_replace(car$name, 'Nissan', '18')
car$name <- str_replace(car$name, 'Lexus', '19')
car$name <- str_replace(car$name, 'Jaguar', '20')
car$name <- str_replace(car$name, 'Land', '21')
car$name <- str_replace(car$name, 'MG', '22')
car$name <- str_replace(car$name, 'Volvo', '23')
car$name <- str_replace(car$name, 'Daewoo', '24')
car$name <- str_replace(car$name, 'Kia', '25')
```

```r
car$name <- str_replace(car$name, 'Force', '26')
car$name <- str_replace(car$name, 'Ambassador', '27')
car$name <- str_replace(car$name, 'Ashok', '28')
car$name <- str_replace(car$name, 'Isuzu', '29')
car$name <- str_replace(car$name, 'Opel', '30')
car$name <- str_replace(car$name, 'Peugeot', '31')

#Converting car name from categorical to numerical value

car$name <- as.numeric(car$name)
table(car$name)
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 2448  105  467 1415  488  397  228  772  734  230   47   65   31   54   14   40
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31
##  186  120   81   34   71    6    3   67    3    4    6    4    1    5    1    1
```

Highest numbers of cars fall into Maruti brand followed by Hyundai, Mahindra and Tata

2.2 Substituting blank with NA for columns mileage, engine, max_power

```r
car$mileage[car$mileage == ""] <- NA
car$engine[car$engine == ""] <- NA
car$max_power[car$max_power == ""] <- NA
```

2.3 Checking for missing values

```r
# Checking for missing values
sapply(car, function(x) sum(is.na(x)))
##          name          year selling_price     km_driven          fuel
##             0             0             0             0             0
##   seller_type  transmission         owner       mileage        engine
##             0             0             0           221           221
##     max_power         seats
##           215           221
```

There are 221 missing values for mileage, engine, seats and 215 missing values for max_power

2.4 Transforming mileage, engine, max_power and seat from categorical to numerical value and replacing missing values with their mean values

```r
#Removing unit from mileage, converting it to numeric value and replacing the missing values
car$mileage <- str_replace(car$mileage, 'kmpl', '')
car$mileage <- str_replace(car$mileage, 'km/kg', '')
car$mileage <- as.numeric(car$mileage)
car$mileage[is.na(car$mileage)]<-mean(car$mileage,na.rm=TRUE)
```

```r
#Removing unit from engine, converting it to numeric value and replacing the missing values

car$engine <- str_replace(car$engine, 'CC', '')
car$engine <- as.numeric(car$engine)
car$engine[is.na(car$engine)]<-mean(car$engine,na.rm=TRUE)
```

```r
#Removing unit from max_power, converting it to numeric value and replacing the missing values
car$max_power <- str_replace(car$max_power, 'bhp', '')
car$max_power <- as.numeric(car$max_power)
car$max_power[is.na(car$max_power)]<-mean(car$max_power,na.rm=TRUE)
```

```r
#Converting seats to numeric value and replacing the missing values
car$seats <- as.numeric(car$seats)
car$seats[is.na(car$seats)]<-median(car$seats,na.rm=TRUE)
```

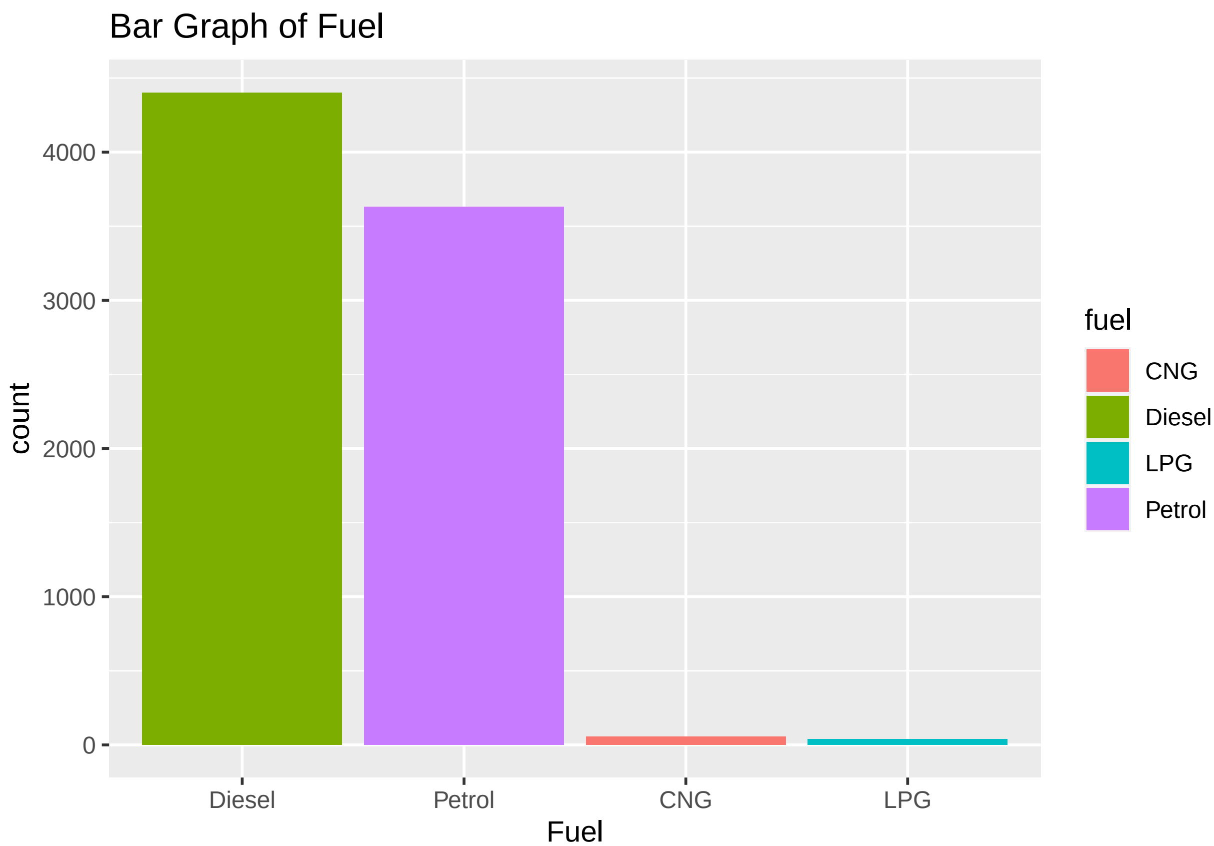
Let's check for missing values after treating missing values

```r
# Checking for missing values once again
sapply(car, function(x) sum(is.na(x)))
##          name          year selling_price     km_driven          fuel 
##             0             0             0             0             0 
##   seller_type  transmission         owner       mileage        engine 
##             0             0             0             0             0 
##     max_power         seats 
##             0             0
```

There are no missing vales any more.

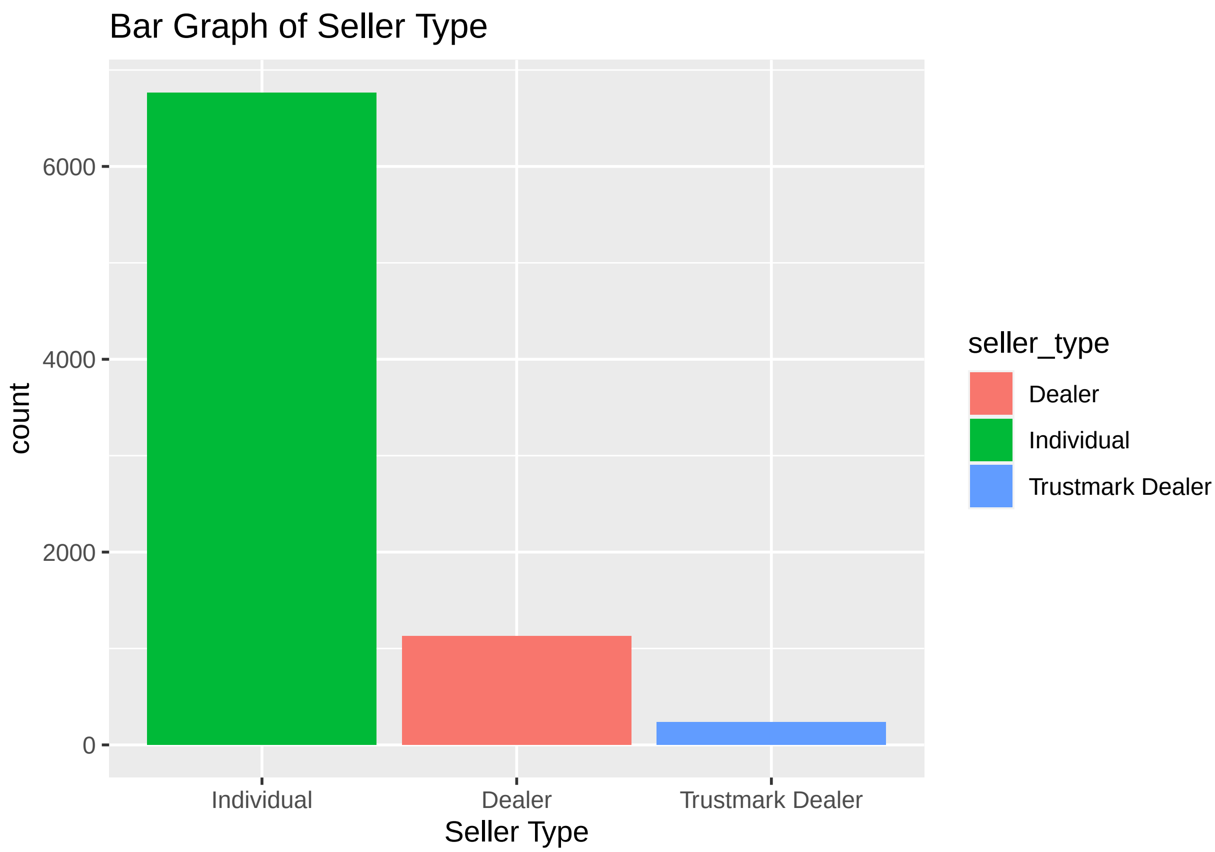
2.5 Plotting categorical Values and checking for distribution

```r
# Bar graph of Fuel
ggplot(data = car, aes(x=reorder(fuel, fuel, function(x)-length(x)), fill = fuel)) +
  geom_bar() + labs(x='Fuel') + labs(title = "Bar Graph of Fuel")
```



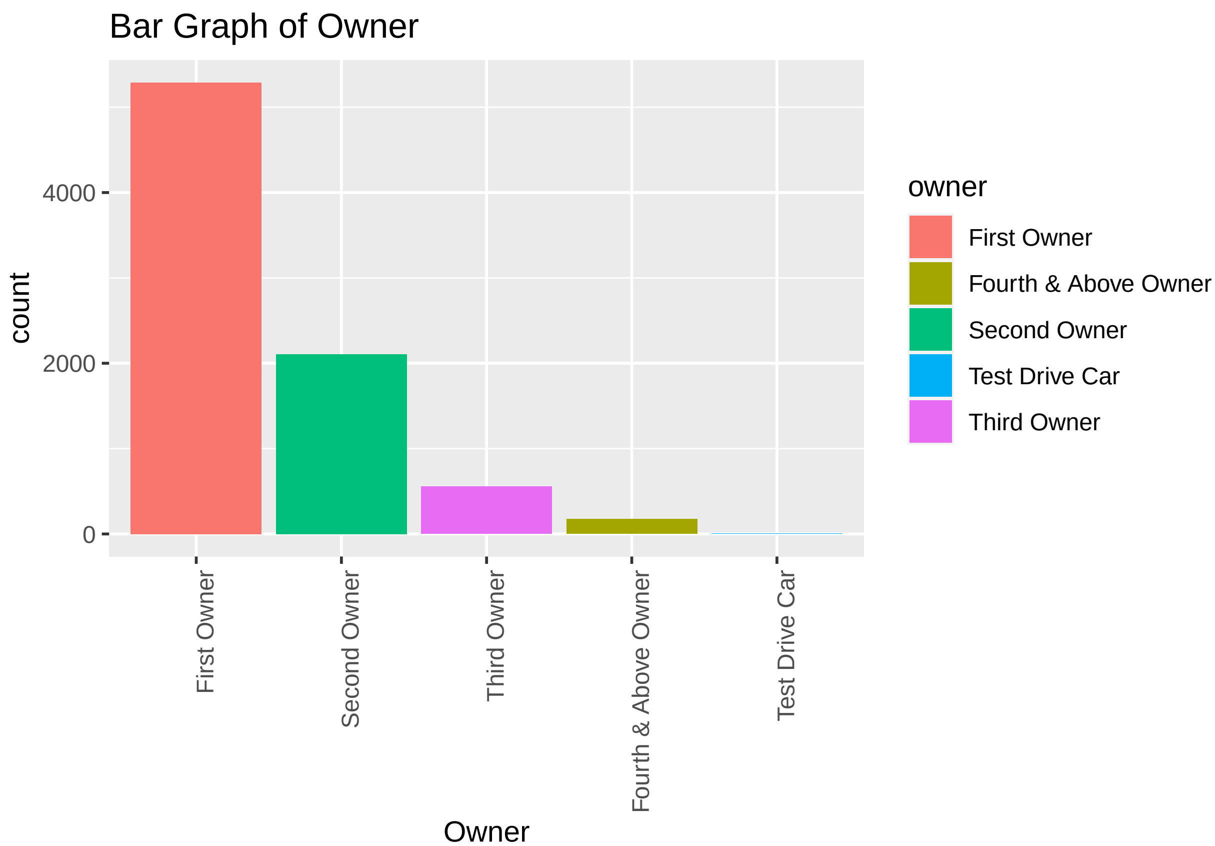
Most of the cars fall into Diesel category followed by Petrol. Very few cars fall into CNG and LPG category.

```
#Bar graph of Seller Typs
ggplot(data = car, aes(x=reorder(seller_type, seller_type, function(x)-length(x)), fill = seller_type)) +
  geom_bar() + labs(x='Seller Type') + labs(title = "Bar Graph of Seller Type")
```



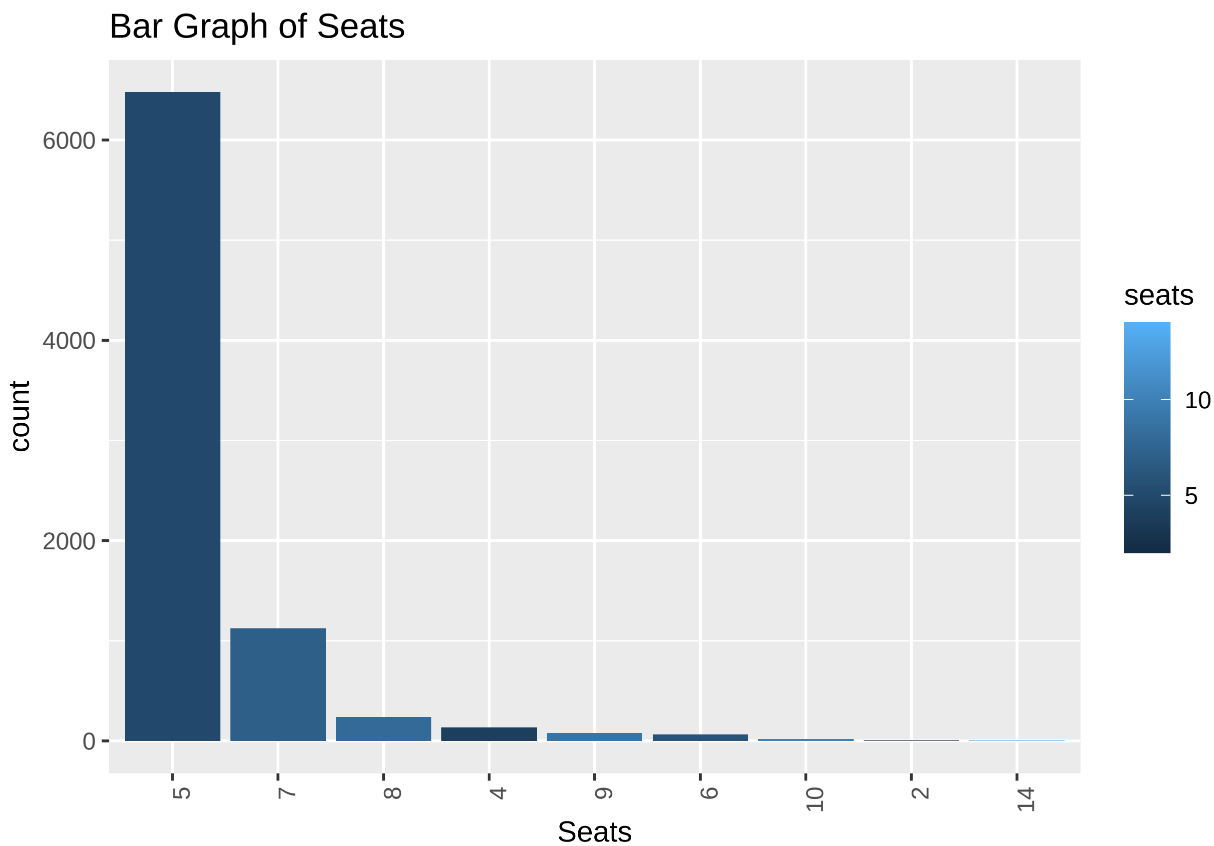
Huge number of cars are owned by individual owners followed by Dealer and Trustmark Dealers.

```
# Bar graph of Owner
ggplot(data = car, aes(x=reorder(owner, owner, function(x)-length(x)), fill = owner)) +
  geom_bar() + labs(x='Owner') + labs(title = "Bar Graph of Owner") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Most of the cars are owned by first owners.

```
# Bar graph of seats
ggplot(data = car, aes(x=reorder(seats, seats, function(x)-length(x)), fill = seats)) +
  geom_bar() + labs(x='Seats') + labs(title = "Bar Graph of Seats") +theme(axis.text.x = element_text(angl
```



Most of the cars are 5 seater.

2.6 Converting transmission, owner, seller type and fuel into ordinal encoder

```
#Converting transmission column into binary 0 if Manual and 1 if Automatic
car$transmission <- str_replace(car$transmission, 'Manual', "0")
car$transmission <- str_replace(car$transmission, 'Automatic', "1")
car$transmission <- as.numeric(car$transmission)
table(car$transmission)
##
##    0    1
## 7078 1050
```

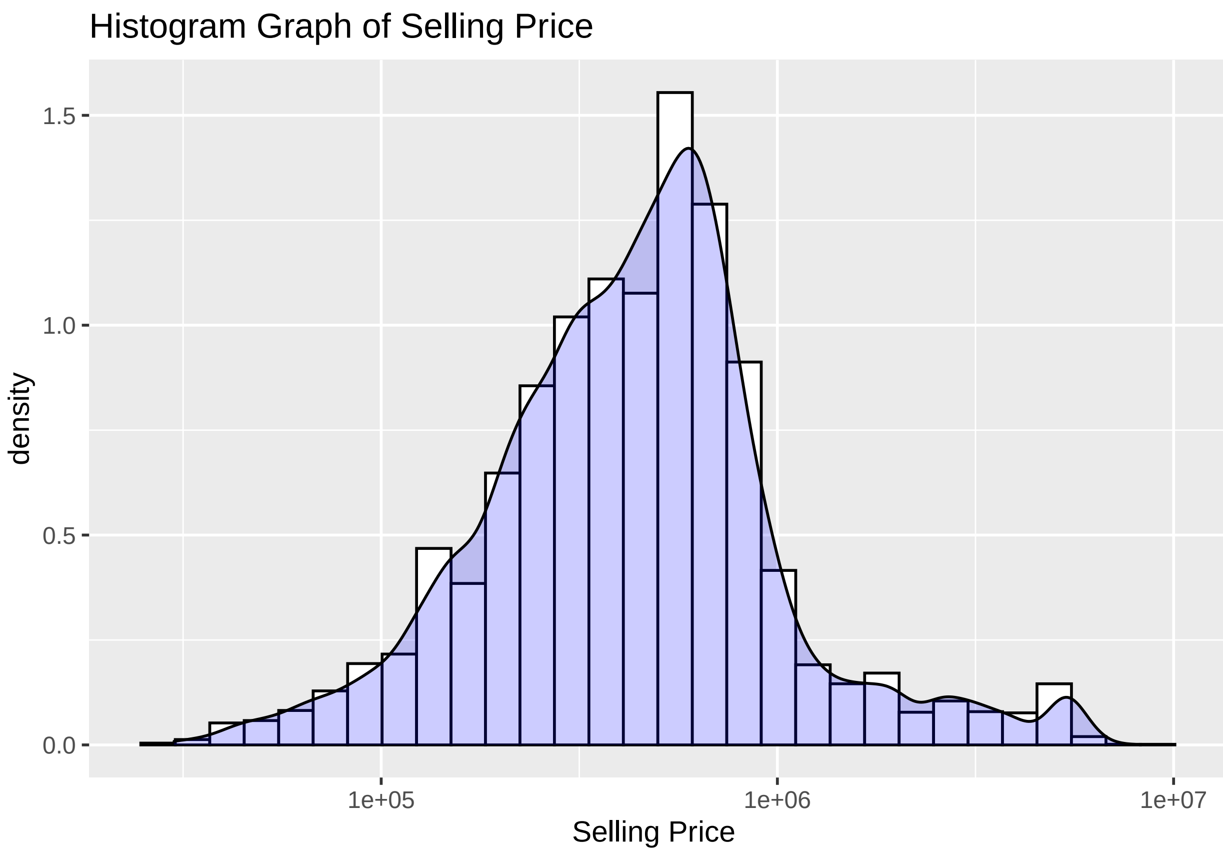
```
#Converting owner into Ordinal Encoder
car$owner <- str_replace(car$owner, 'First Owner', "0")
car$owner <- str_replace(car$owner, 'Second Owner', "1")
car$owner <- str_replace(car$owner, 'Third Owner', "2")
car$owner <- str_replace(car$owner, 'Fourth & Above Owner', "3")
car$owner <- str_replace(car$owner, 'Test Drive Car', "4")
car$owner <- as.numeric(car$owner)
table(car$owner)
##
##    0    1    2    3    4
## 5289 2105  555  174    5
```

```r
#Converting seller_type into Ordinal Encoder
car$seller_type <- str_replace(car$seller_type, "Trustmark Dealer", "0")
car$seller_type <- str_replace(car$seller_type, "Dealer", "1")
car$seller_type <- str_replace(car$seller_type, "Individual", "2")
car$seller_type <- as.numeric(car$seller_type)
table(car$seller_type)
##
##    0    1    2
##  236 1126 6766
```

```r
#Converting fuel into Ordinal Encoder
car$fuel <- str_replace(car$fuel, 'Diesel', "0")
car$fuel <- str_replace(car$fuel, 'Petrol', "1")
car$fuel <- str_replace(car$fuel, 'CNG', "2")
car$fuel <- str_replace(car$fuel, 'LPG', "3")
car$fuel <- as.numeric(car$fuel)
table(car$fuel)
##
##    0    1    2    3
## 4402 3631   57   38
```

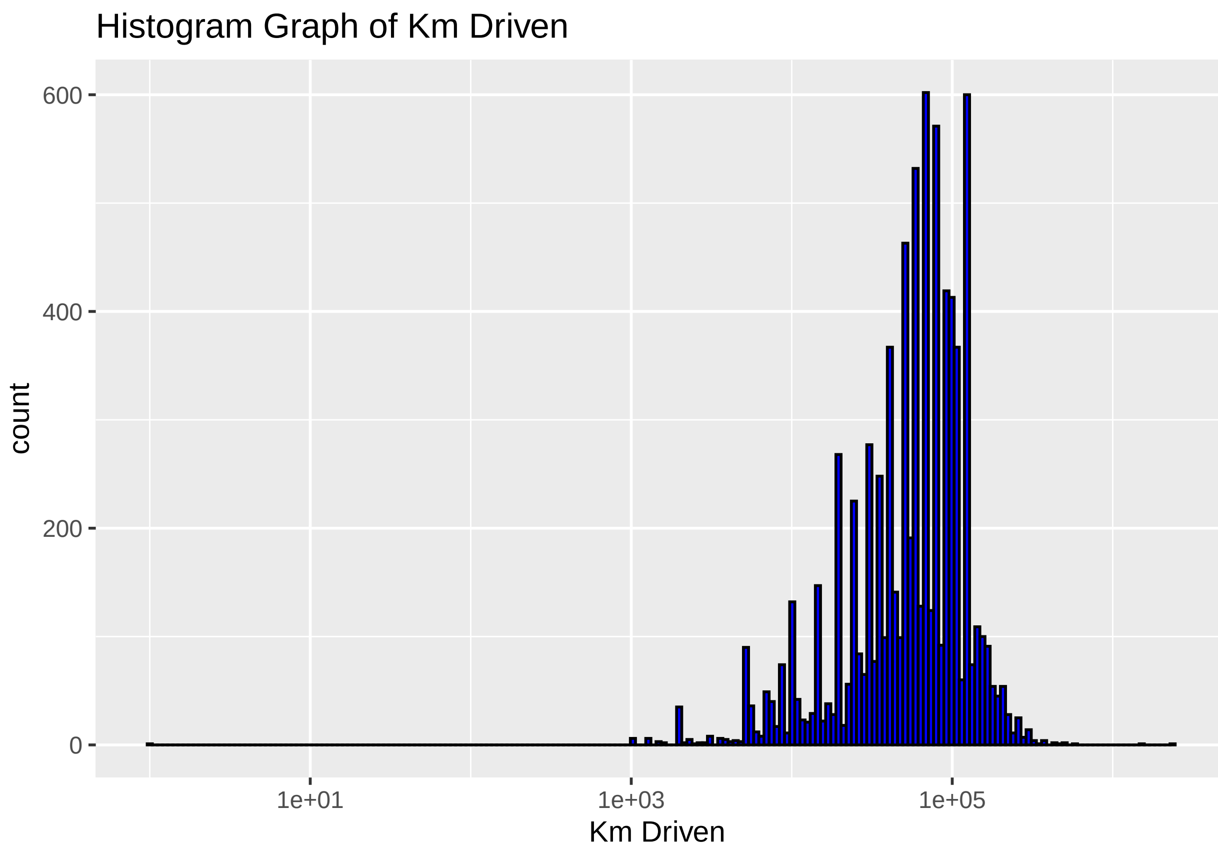
2.7 Plotting histogram of selling price, km driven to check the distribution

```r
#Histogram of Selling Price
ggplot(car, aes(x=selling_price)) +
  geom_histogram(aes(y=..density..), colour="black", fill="white")+
  geom_density(alpha=.2, fill="blue")+
  labs(x='Selling Price ') + labs(title = "Histogram Graph of Selling Price") +
  scale_x_continuous(trans='log10')
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



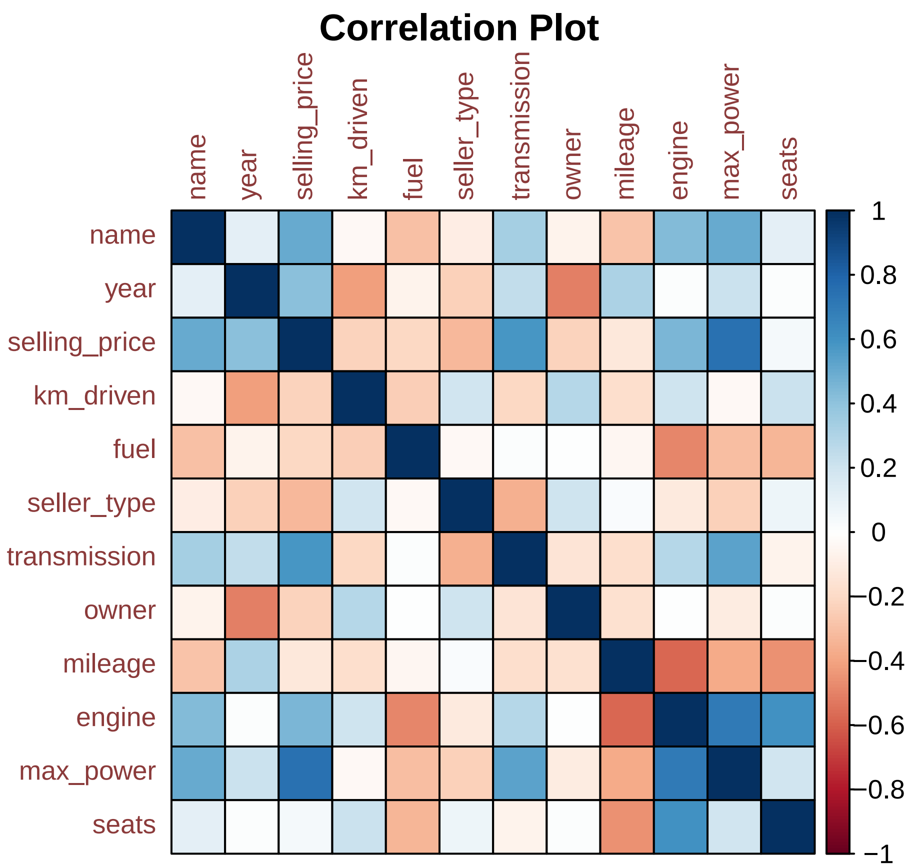
We can see that selling price is heavily skewed.

```
#Histogram of Km Driven
ggplot(car, aes(x=km_driven)) +
  geom_histogram(color="black", fill="blue", bins = 200)+
  labs(x='Km Driven ') + labs(title = "Histogram Graph of Km Driven") +
  scale_x_continuous(trans='log10')
```



3 Checking correlation between variables

```
library(corrplot)
corrplot(cor(car), type="full",
         method ="color", title = "Correlation Plot",
         mar=c(0,0,1,0), tl.cex= 0.8, outline= T, tl.col="indianred4")
```




```
round(cor(car),2)
##                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 max_power then transmission and name.

4 Splitting the Data into training and test data sets

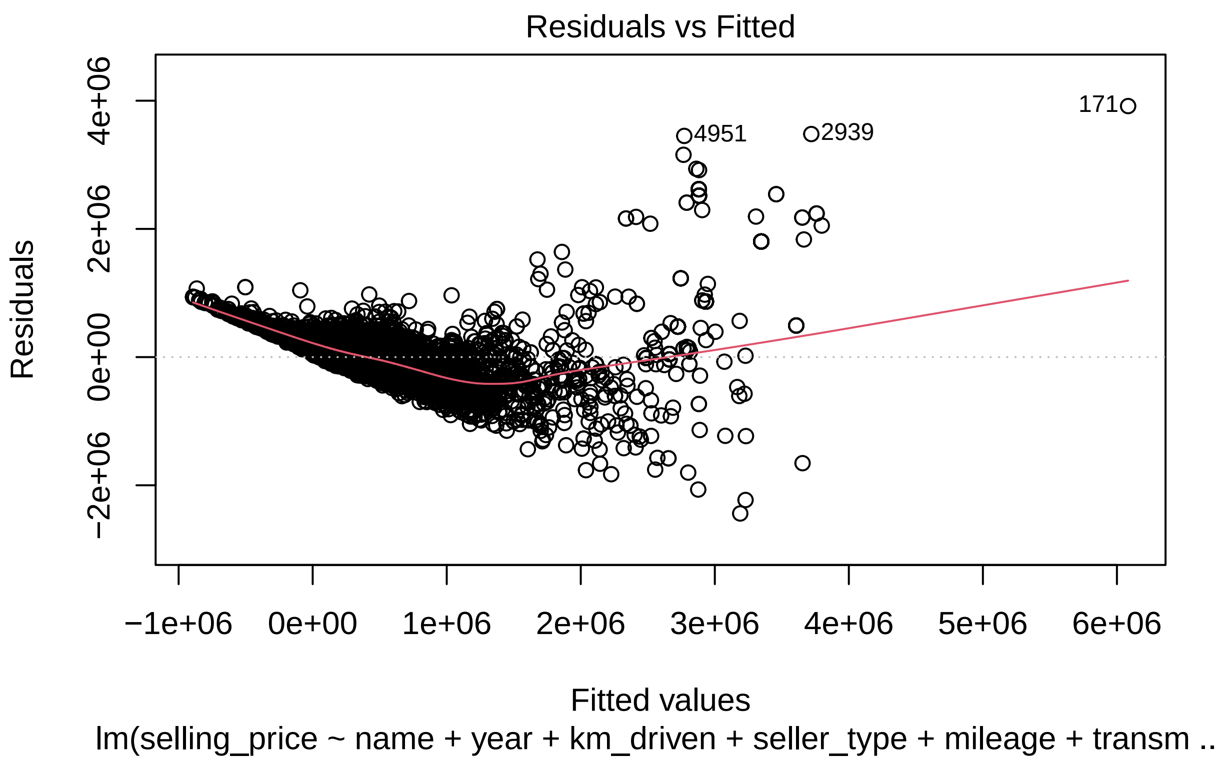
```
set.seed(5)
trainIndex <- createDataPartition(car$selling_price, p = .7,
                                  list = FALSE,
                                  times = 1)
Train <- car[ trainIndex,]
Test <- car[-trainIndex,]
```

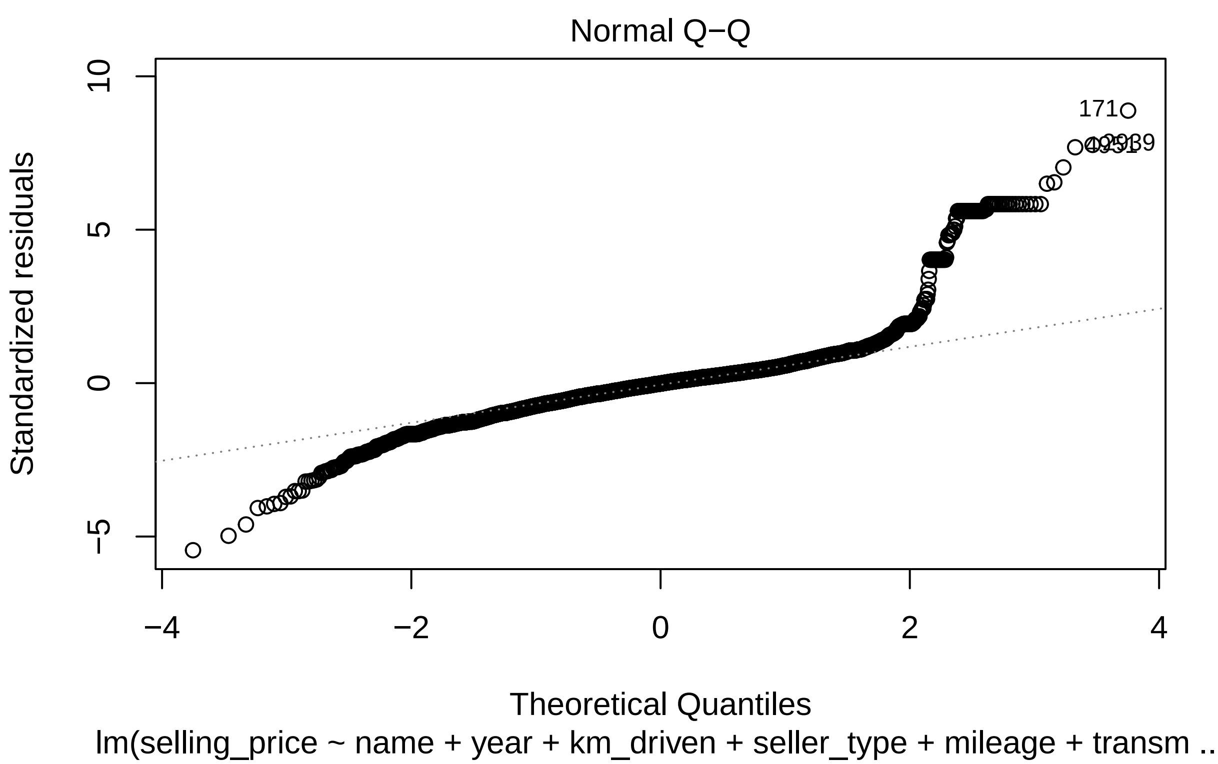
Splitting data into 70% Training and 30% Test.

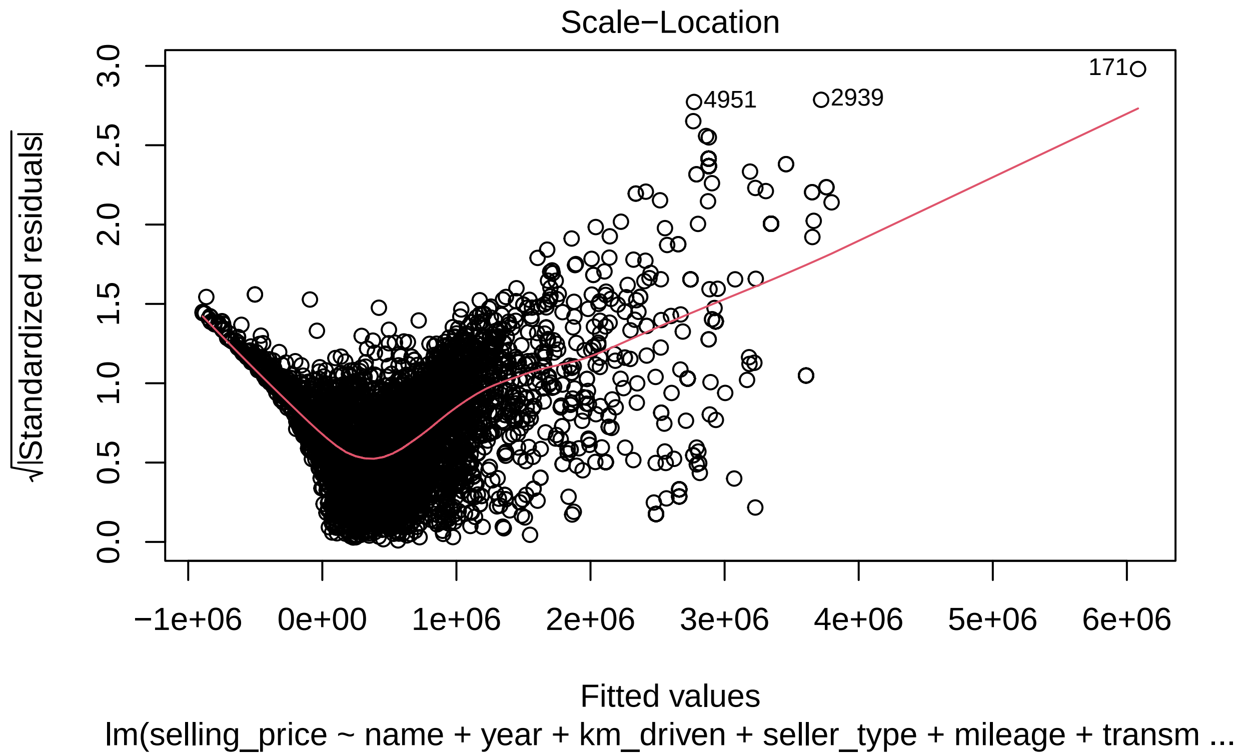
5 Model 1 - Linear Regression

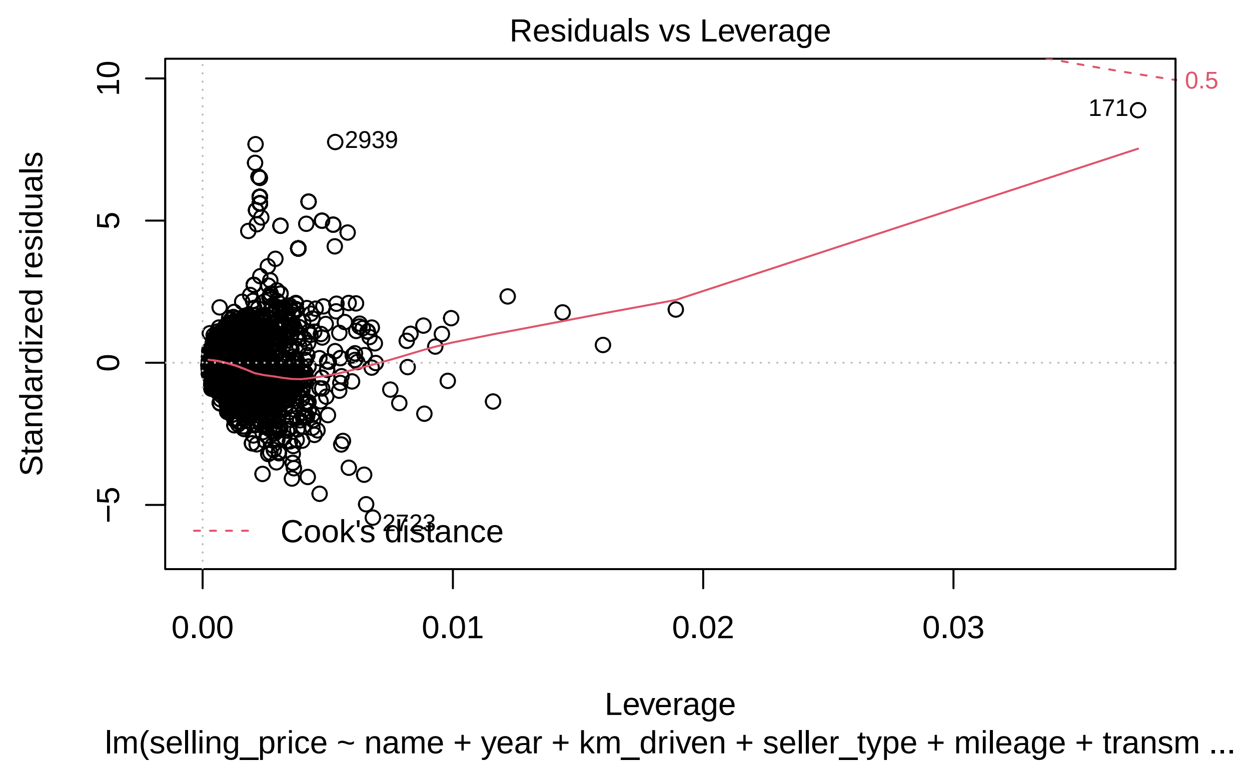
5.1 Building Model

```
m1_lr <- lm(selling_price ~ name+year+km_driven+seller_type+mileage+transmission+max_power, data = Train)
summary(m1_lr)
##
## Call:
## lm(formula = selling_price ~ name + year + km_driven + seller_type +
##     mileage + transmission + max_power, data = Train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2439581  -212038    -5929   162978  3916432
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.961e+07  3.768e+06 -15.818  < 2e-16 ***
## name          2.471e+04  1.374e+03  17.981  < 2e-16 ***
## year          2.919e+04  1.877e+03  15.550  < 2e-16 ***
## km_driven    -1.496e+00  1.471e-01 -10.163  < 2e-16 ***
## seller_type  -1.031e+05  1.393e+04  -7.402 1.54e-13 ***
## mileage       2.033e+04  1.816e+03  11.192  < 2e-16 ***
## transmission  4.344e+05  2.259e+04  19.228  < 2e-16 ***
## max_power     1.303e+04  2.356e+02  55.287  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 449500 on 5683 degrees of freedom
## Multiple R-squared:  0.6954, Adjusted R-squared:  0.695
## F-statistic:  1853 on 7 and 5683 DF,  p-value: < 2.2e-16
plot(m1_lr)
```


Residuals vs Fitted
Residuals
4e+06
2e+06
0e+00
−2e+06
4951
2939
171
−1e+06
0e+00
1e+06
2e+06
3e+06
4e+06
5e+06
6e+06
Fitted values
lm(selling_price ~ name + year + km_driven + seller_type + mileage + transm ...


Normal Q−Q
Standardized residuals
10
5
0
−5
171
2939
−4
−2
0
2
4
Theoretical Quantiles
lm(selling_price ~ name + year + km_driven + seller_type + mileage + transm ...


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
−1e+06
0e+00
1e+06
2e+06
3e+06
4e+06
5e+06
6e+06
4951
2939
171
Fitted values
lm(selling_price ~ name + year + km_driven + seller_type + mileage + transm ...

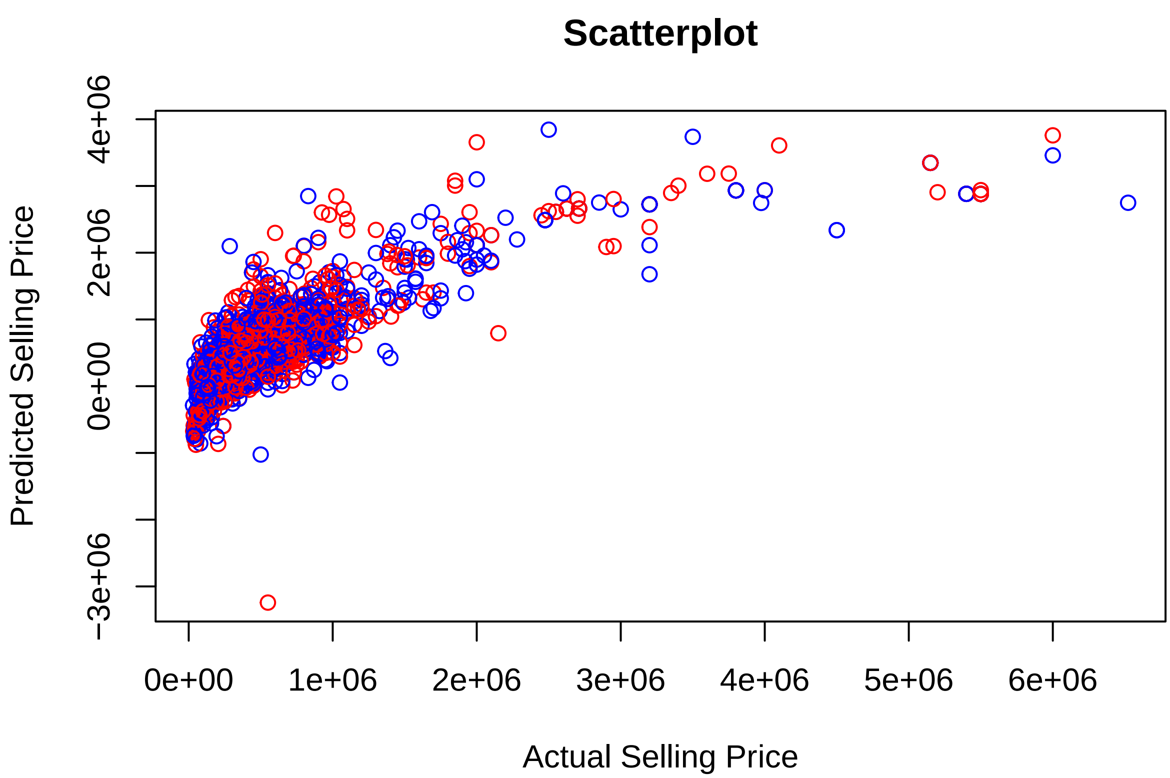

Residuals vs Leverage
Standardized residuals
−5
0
5
10
0.00
0.01
0.02
0.03
2939
171
2723
0.5
Cook's distance
Leverage
lm(selling_price ~ name + year + km_driven + seller_type + mileage + transm ...

5.2 Using the model to predict selling price in the Test dataset

```
pred_lr <- predict(m1_lr, newdata = Test)
error_lr <- Test$selling_price - pred_lr
RMSE_lr <- sqrt(mean(error_lr^2))
RMSE_lr
## [1] 457916.9
```

5.3 Plotting predicted vs. actual values

```
plot(Test$selling_price,pred_lr, main="Scatterplot", col = c("red","blue"), xlab = "Actual Selling Price",
```



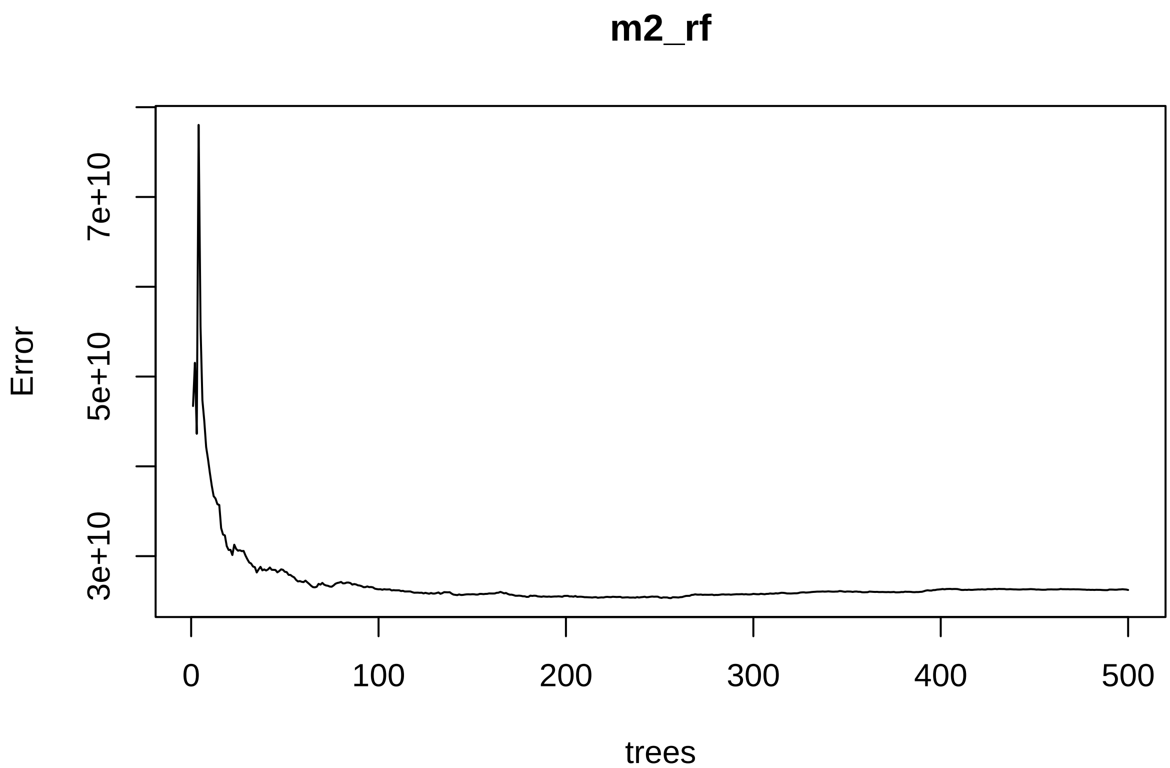
Built Linear Regression models with different variables but kept the model with best RMSE value. RMSE value of 457916.9

6 Model 2 - Random Forest

6.1 Building Model

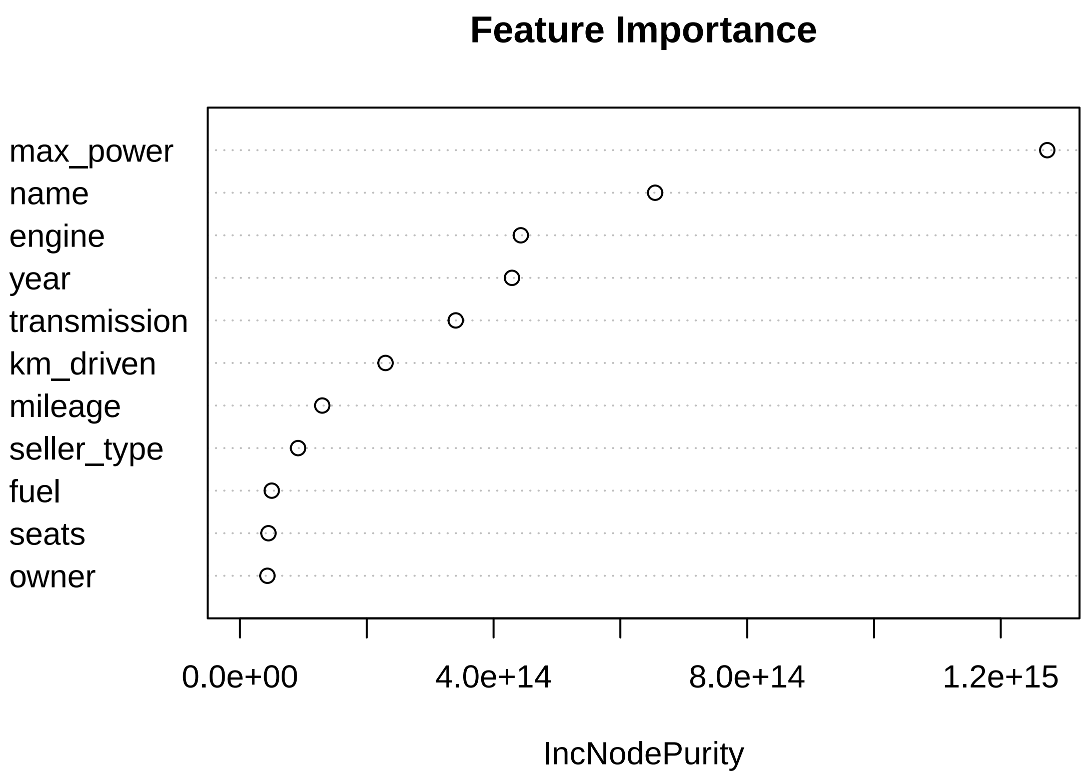
```
m2_rf <- randomForest(selling_price~.,data = Train)
m2_rf
##
```

```
## Call:
##  randomForest(formula = selling_price ~ ., data = Train)
##                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(m2_rf)
```



6.2 Feature Importance Plot

```
varImpPlot(m2_rf, main ='Feature Importance')
```

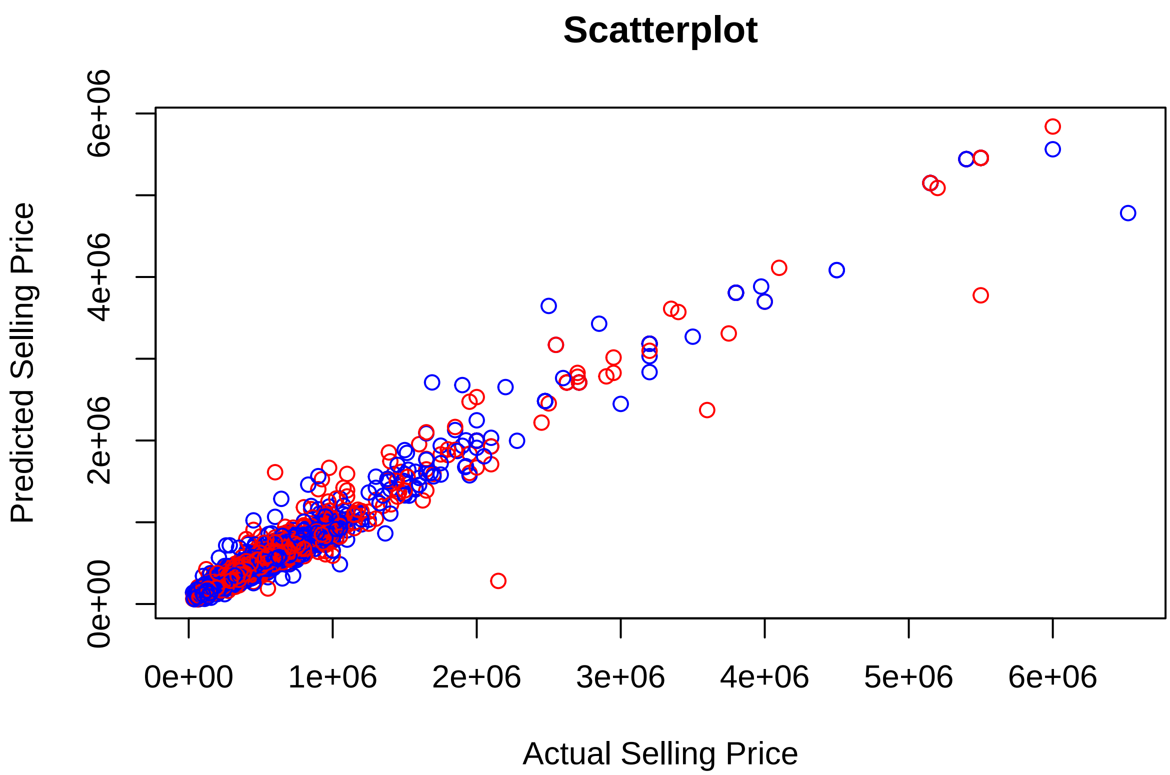


6.3 Using the model to predict selling price in the Test dataset

```
pred_rf <- predict(m2_rf, Test)
error_rf <- Test$selling_price - pred_rf
RMSE_rf <- sqrt(mean(error_rf^2))
RMSE_rf
## [1] 128704
```

6.4 Plotting predicted vs. actual values

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



We got RMSE value of 129840.9

7 Model 3 - Gradient Boosting

7.1 Building Model

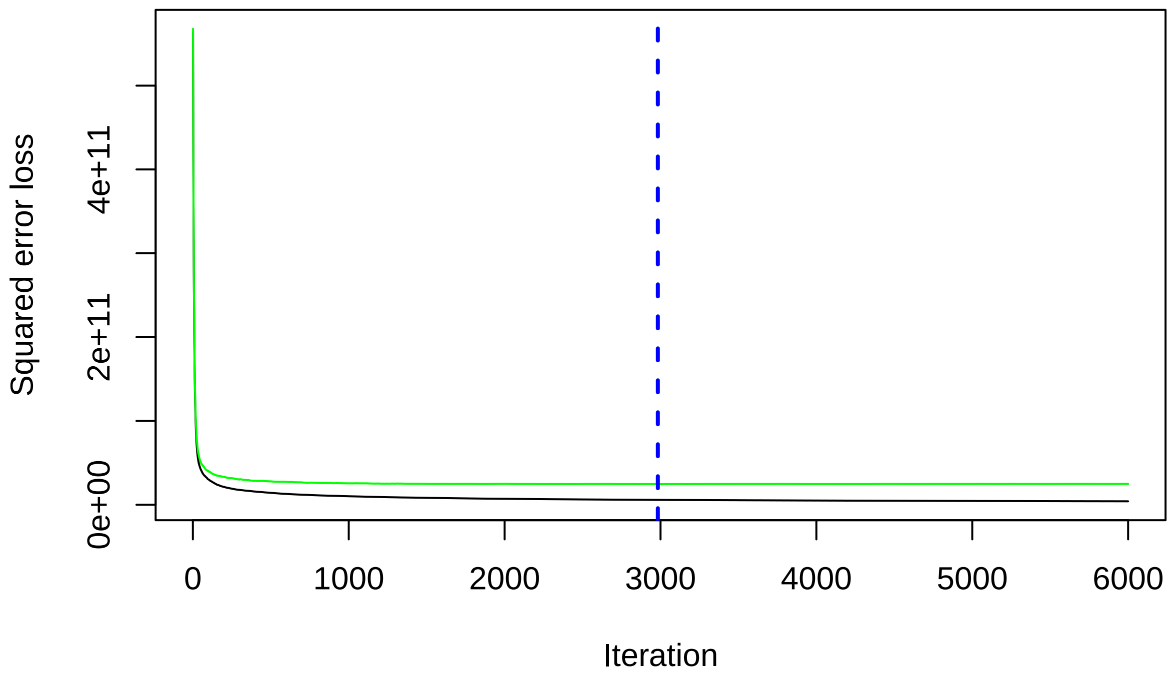
```
library(gbm)
## Loaded gbm 2.1.8
set.seed(123)
m3_gbm <- gbm(
  formula = selling_price ~ .,
  distribution = "gaussian",
  data = Train,
  n.trees = 6000,
  interaction.depth = 3,
  shrinkage = 0.1,
  cv.folds = 5,
  n.cores = NULL, # will use all cores by default
  verbose = FALSE
)

m3_gbm
## gbm(formula = selling_price ~ ., distribution = "gaussian", data = Train,
##     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 2983.
## There were 11 predictors of which 11 had non-zero influence.
```

7.2 plot loss function as a result of n trees added to the ensemble

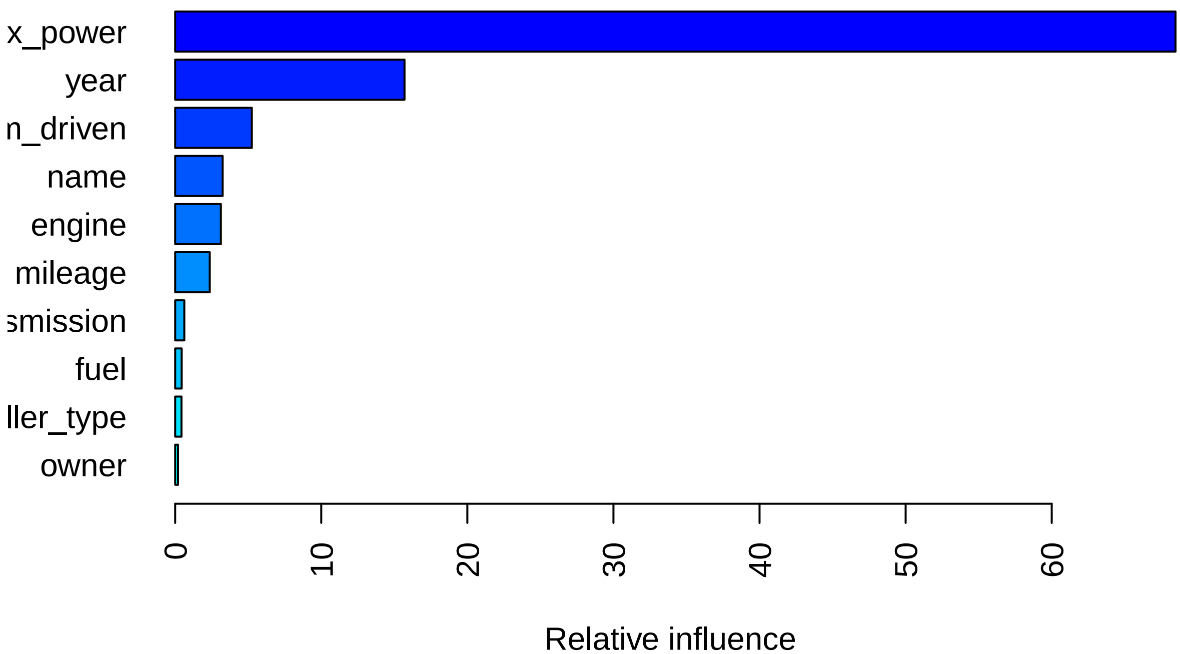
```
gbm.perf(m3_gbm, method = "cv")
```




```
## [1] 2983
```

7.3 Variable importance

```
summary(
  m3_gbm,
  cBars = 10,
  method = relative.influence, las = 2
)
```



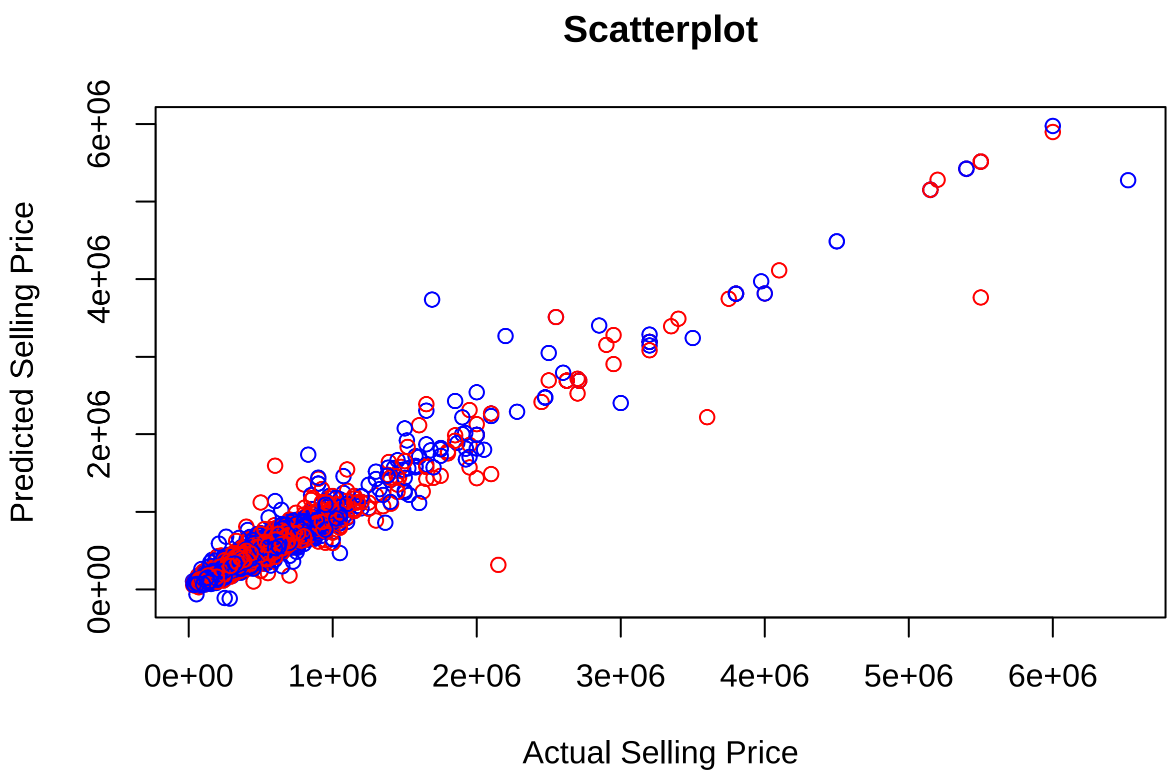

```
##                       var    rel.inf
## max_power       max_power 68.4760718
## year                 year 15.6969182
## km_driven       km_driven  5.2454442
## name                 name  3.2424510
## engine             engine  3.1259345
## mileage           mileage  2.3627410
## transmission transmission  0.6252901
## fuel                 fuel  0.4399373
## seller_type   seller_type  0.4286109
## owner               owner  0.1971400
## seats               seats  0.1594611
```

7.4 Using the model to predict selling price in the Test dataset

```
pred_gbm <- predict(m3_gbm, Test)
## Using 2983 trees...
error_gbm <- Test$selling_price - pred_gbm
RMSE_gbm <- sqrt(mean(error_gbm^2))
RMSE_gbm
## [1] 135282.4
```

7.5 Plotting predicted vs. actual values

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



We got RMSE value of 135282.4

8 Conclusion and Model Comparison

We used linear regression, random forest and gradient boosting models to predict selling price of cars and we see that random forest gives us a better RMSE among the three models. The RMSE comparison for three different models is shown below.

Model	RMSE
Linear Regression	457916.9
Random Forest	129840.9
Gradient Boosting	135282.4

Random Forest explains 96% of the variation. Variables that are useful to describe the variance are max_power, name, engine and year. The accuracy of the model in predicting the car price is measured with RMSE, RMSE of test dataset is 129840.9.In the random forest model we used 500 number of trees and number of variables tried at each split as 3. We can further tune the model to get better RMSE.