Used Car Price Prediction

Geedhu

2023-08-05

Basic Set Up

```
# Clear plots
if(!is.null(dev.list())) dev.off()
```

```
## null device
##           1
```

```
# Clear console
cat("\014")
```

```r
# Clean workspace
rm(list=ls())
```

```r
#If the library is not already downloaded, download it
#if(!require(lattice)){install.packages("lattice")}
library("lattice")
#if(!require(pastecs)){install.packages("pastecs")}
library("pastecs")
#if(!require(corrgram)){install.packages("corrgram")}
library("corrgram")
```

```
##
## Attaching package: 'corrgram'

## The following object is masked from 'package:lattice':
##
##     panel.fill
```

```r
#if(!require(cowplot)){install.packages("cowplot")}
library("cowplot")
#if(!require(tidyverse)){install.packages("tidyverse")}
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks pastecs::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks pastecs::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks pastecs::last()
## x lubridate::stamp() masks cowplot::stamp()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
#if(!require(caret)){install.packages("caret")}
library("caret")
```

```
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
#if(!require(datarium)){install.packages("datarium")}
library("datarium")
```

Procuring Data Sets

Data set is obtained from websit Kaggle https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho?select=car+data.csv

Reading the train data set

```
dataset <- read.csv("C:/Users/Geedhu/Documents/Maths _ Data Analysys/Project/CAR DETAILS FROM CAR DEKHO
head(dataset,3)
```

```
##                         name year selling_price km_driven   fuel seller_type
## 1           Maruti 800 AC 2007         60000     70000 Petrol  Individual
## 2 Maruti Wagon R LXI Minor 2007        135000     50000 Petrol  Individual
## 3     Hyundai Verna 1.6 SX 2012        600000    100000 Diesel  Individual
##   transmission       owner
## 1       Manual First Owner
## 2       Manual First Owner
## 3       Manual First Owner
```

changing the column names

```
# Viewing the first few rows of data set
colnames(dataset) <- paste(colnames(dataset),"JDG",sep="_")
head(dataset,3)
```

```
##                     name_JDG year_JDG selling_price_JDG km_driven_JDG fuel_JDG
## 1           Maruti 800 AC     2007             60000         70000   Petrol
## 2 Maruti Wagon R LXI Minor     2007            135000         50000   Petrol
## 3     Hyundai Verna 1.6 SX     2012            600000        100000   Diesel
##   seller_type_JDG transmission_JDG   owner_JDG
## 1      Individual           Manual First Owner
## 2      Individual           Manual First Owner
## 3      Individual           Manual First Owner
```

Structure of the dataset

```
str(dataset)
```

```
## 'data.frame':    4340 obs. of  8 variables:
##  $ name_JDG         : chr  "Maruti 800 AC" "Maruti Wagon R LXI Minor" "Hyundai Verna 1.6 SX" "Datsun
##  $ year_JDG         : int  2007 2007 2012 2017 2014 2007 2016 2014 2015 2017 ...
##  $ selling_price_JDG: int  60000 135000 600000 250000 450000 140000 550000 240000 850000 365000 ...
##  $ km_driven_JDG    : int  70000 50000 100000 46000 141000 125000 25000 60000 25000 78000 ...
##  $ fuel_JDG         : chr  "Petrol" "Petrol" "Diesel" "Petrol" ...
##  $ seller_type_JDG  : chr  "Individual" "Individual" "Individual" "Individual" ...
##  $ transmission_JDG : chr  "Manual" "Manual" "Manual" "Manual" ...
##  $ owner_JDG        : chr  "First Owner" "First Owner" "First Owner" "First Owner" ...
```

Transform character variables to factor variables.

```
dataset$name_JDG = as.factor(dataset$name_JDG)
dataset$fuel_JDG= as.factor(dataset$fuel_JDG)
dataset$seller_type_JDG= as.factor(dataset$seller_type_JDG)
dataset$transmission_JDG= as.factor(dataset$transmission_JDG)
dataset$owner_JDG= as.factor(dataset$owner_JDG)
str(dataset)
```

```
## 'data.frame':    4340 obs. of  8 variables:
##  $ name_JDG         : Factor w/ 1491 levels "Ambassador CLASSIC 1500 DSL AC",..: 774 1040 566 120 278
##  $ year_JDG         : int  2007 2007 2012 2017 2014 2007 2016 2014 2015 2017 ...
##  $ selling_price_JDG: int  60000 135000 600000 250000 450000 140000 550000 240000 850000 365000 ...
##  $ km_driven_JDG    : int  70000 50000 100000 46000 141000 125000 25000 60000 25000 78000 ...
##  $ fuel_JDG         : Factor w/ 5 levels "CNG","Diesel",..: 5 5 2 5 2 5 5 5 5 1 ...
##  $ seller_type_JDG  : Factor w/ 3 levels "Dealer","Individual",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ transmission_JDG : Factor w/ 2 levels "Automatic","Manual": 2 2 2 2 2 2 2 2 2 2 ...
##  $ owner_JDG        : Factor w/ 5 levels "First Owner",..: 1 1 1 1 3 1 1 3 1 1 ...
```

Removing null values

```
dataset[dataset == ""]
```

```
## character(0)
```

```
dataset[dataset == ""] <- NA
ratio = sum(is.na(dataset))/nrow(dataset)
ratio
```

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

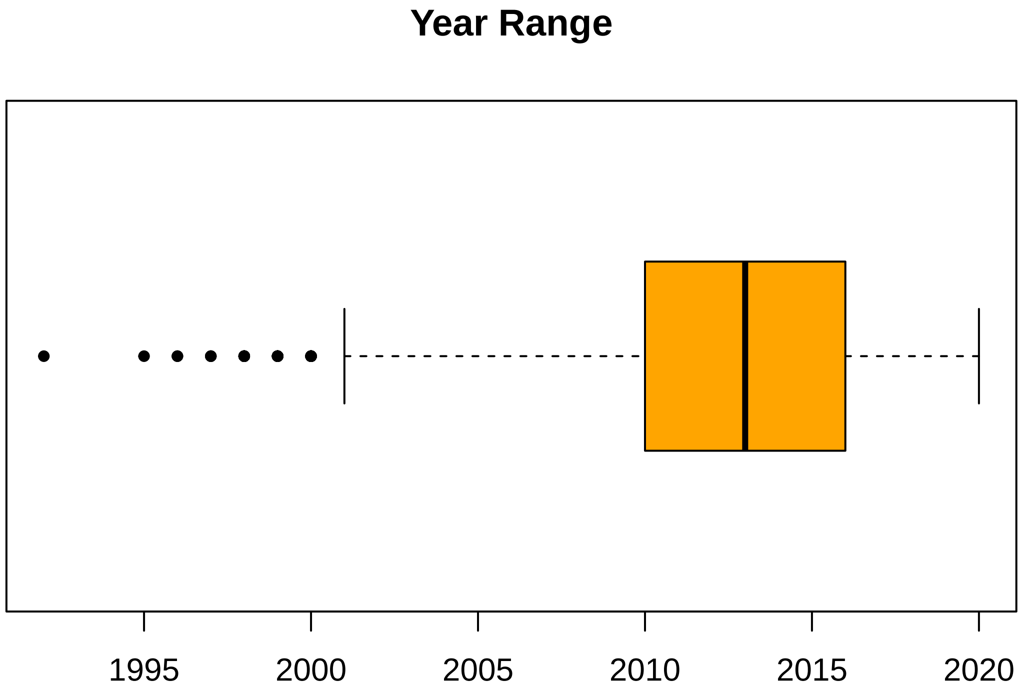
remove duplicates

```
dataset <- unique(dataset)
str(dataset)
```

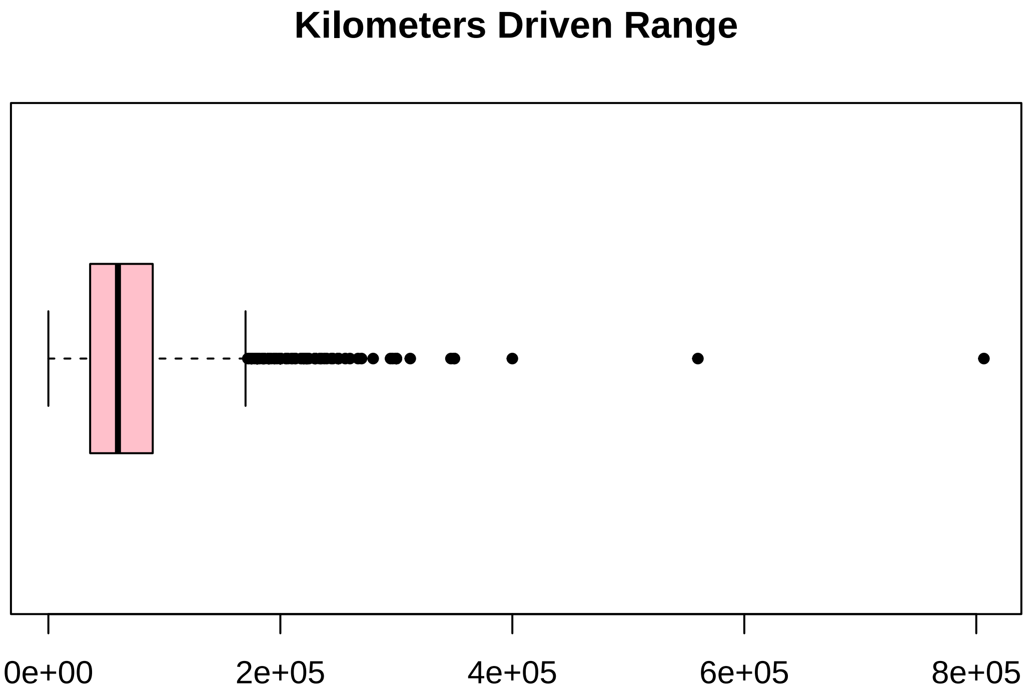
```
## 'data.frame':    3577 obs. of  8 variables:
##  $ name_JDG         : Factor w/ 1491 levels "Ambassador CLASSIC 1500 DSL AC",..: 774 1040 566 120 278
##  $ year_JDG         : int  2007 2007 2012 2017 2014 2007 2016 2014 2015 2017 ...
##  $ selling_price_JDG: int  60000 135000 600000 250000 450000 140000 550000 240000 850000 365000 ...
##  $ km_driven_JDG    : int  70000 50000 100000 46000 141000 125000 25000 60000 25000 78000 ...
##  $ fuel_JDG         : Factor w/ 5 levels "CNG","Diesel",..: 5 5 2 5 2 5 5 5 5 1 ...
##  $ seller_type_JDG  : Factor w/ 3 levels "Dealer","Individual",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ transmission_JDG : Factor w/ 2 levels "Automatic","Manual": 2 2 2 2 2 2 2 2 2 2 ...
##  $ owner_JDG        : Factor w/ 5 levels "First Owner",..: 1 1 1 1 3 1 1 3 1 1 ...
```

removing outliers

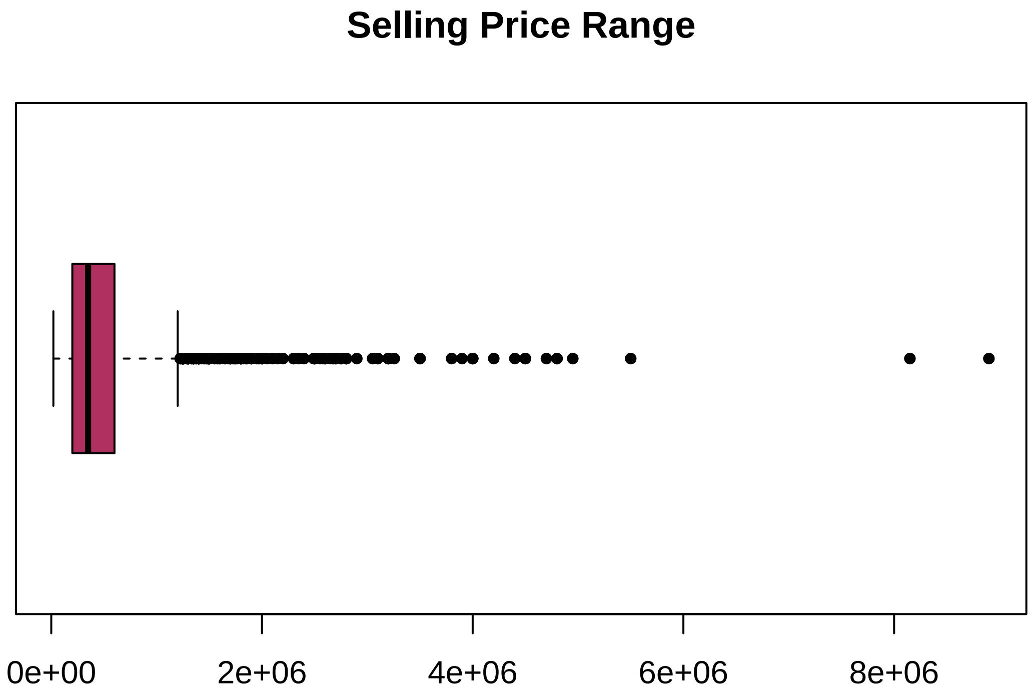
```
#Finding Outliers
# Boxplots can be plotted only for numerical data
boxplot(dataset$year, horizontal=TRUE, pch=20,main = "Year Range",col="Orange")
```



```
boxplot(dataset$km_driven, horizontal=TRUE, pch=20,main = "Kilometers Driven Range",col="Pink")
```

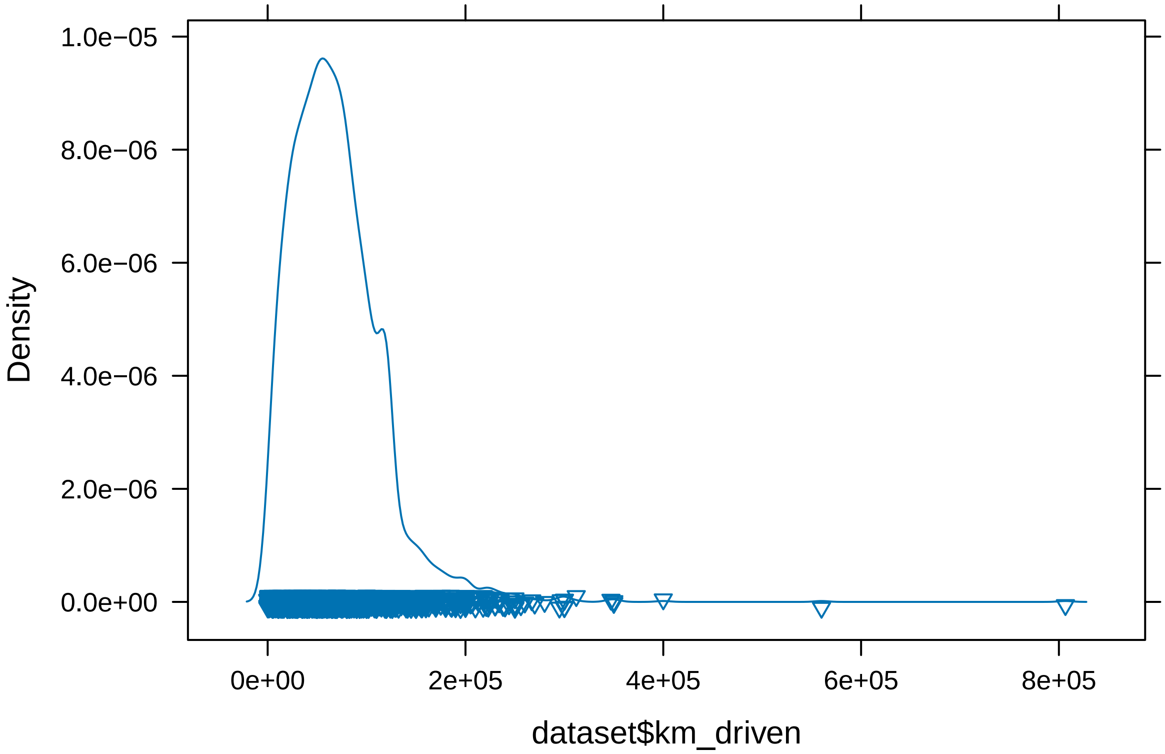


```
boxplot(dataset$selling_price, horizontal=TRUE, pch=20, main = "Selling Price Range",col="Maroon")
```

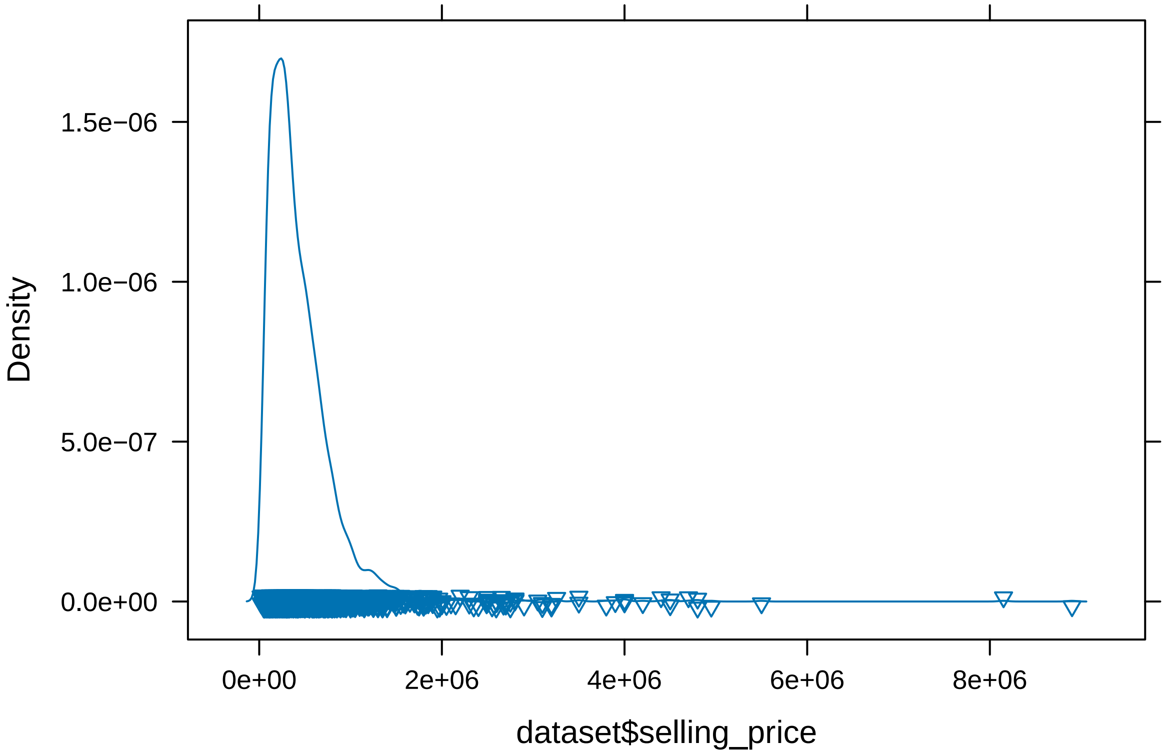


#Density Plots for further details

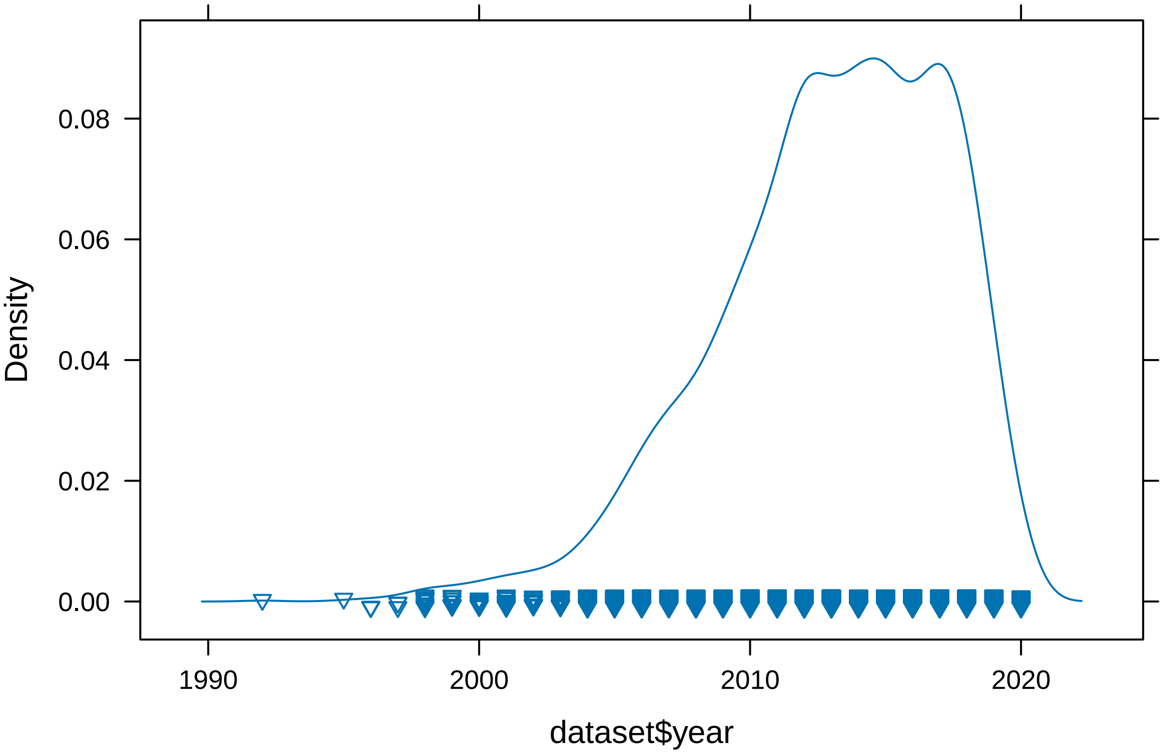
```
densityplot( ~ dataset$km_driven, pch=6)
```



```
densityplot( ~ dataset$selling_price, pch=6)
```

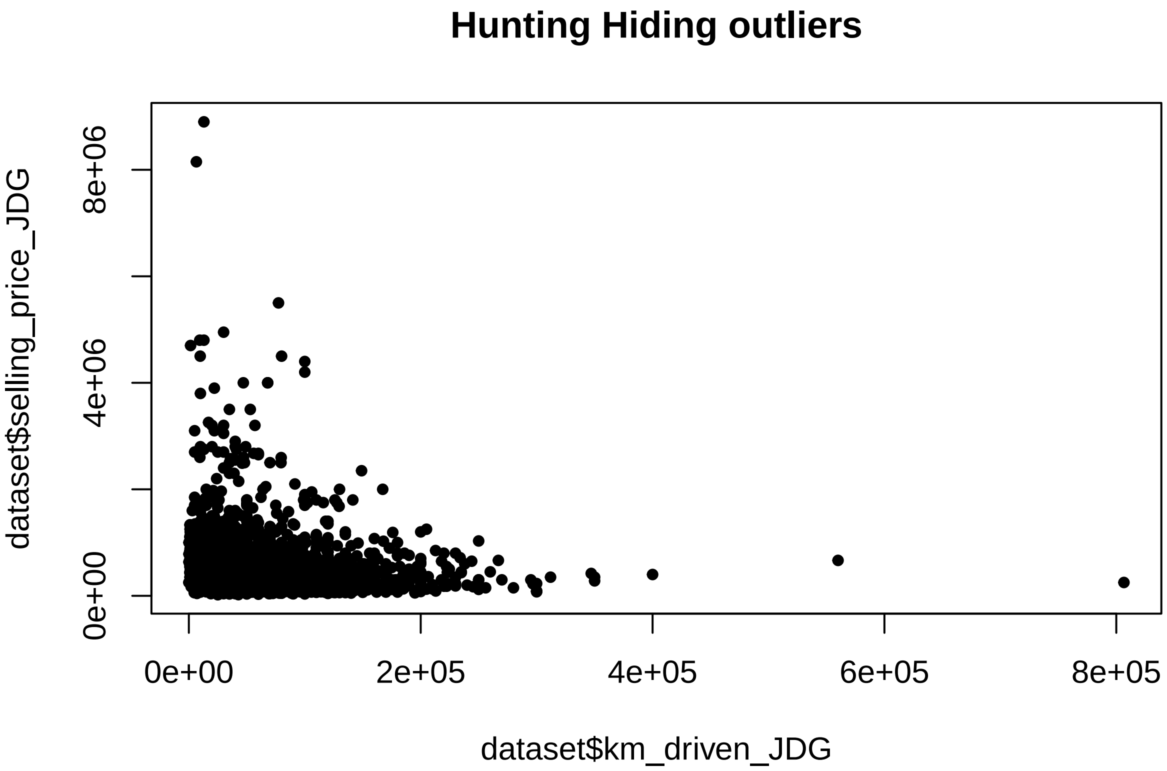


```
densityplot( ~ dataset$year, pch=6)
```

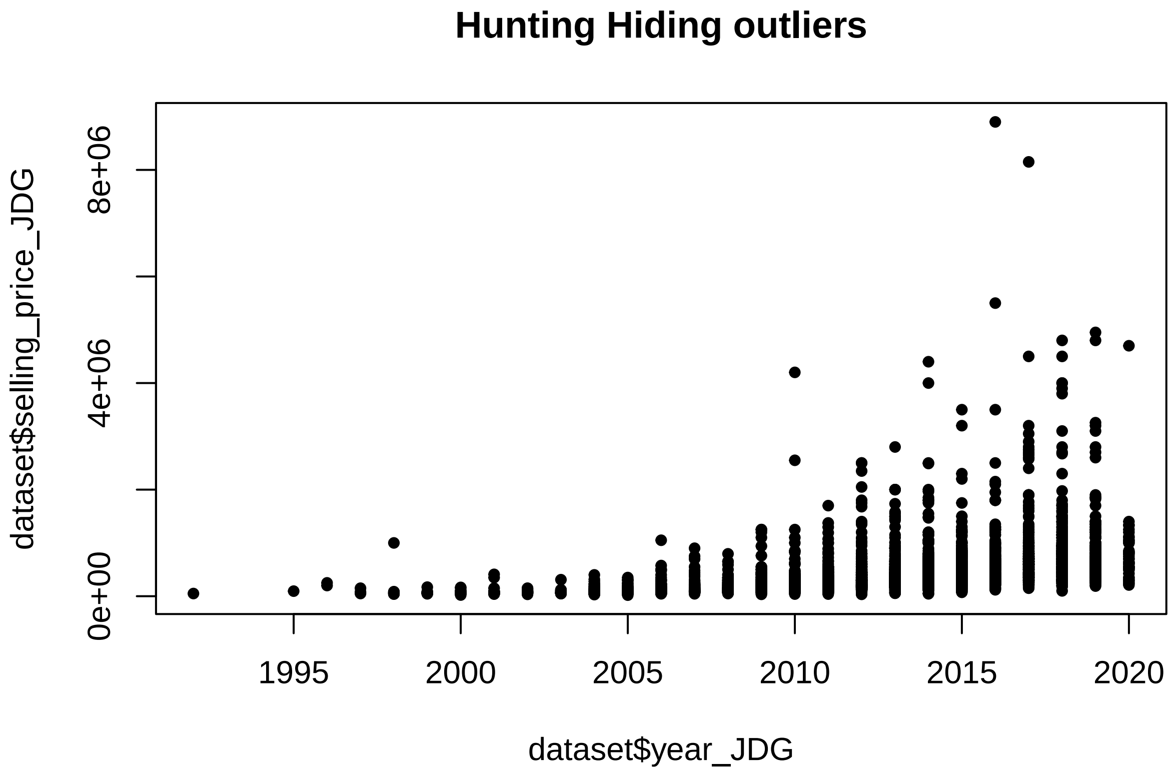


# scatter plot to find hidden outlier

```
plot(dataset$km_driven_JDG,dataset$selling_price_JDG, main='Hunting Hiding outliers',pch=20)
```

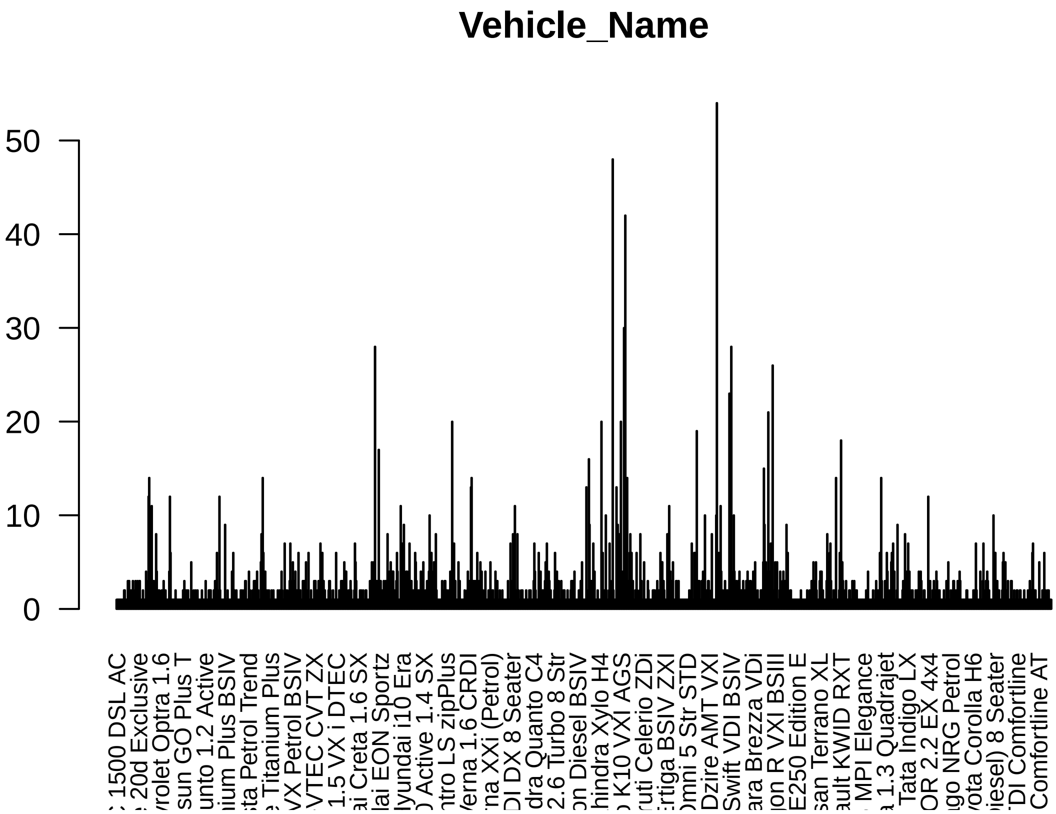


```
plot(dataset$year_JDG,dataset$selling_price_JDG, main='Hunting Hiding outliers',pch=20)
```

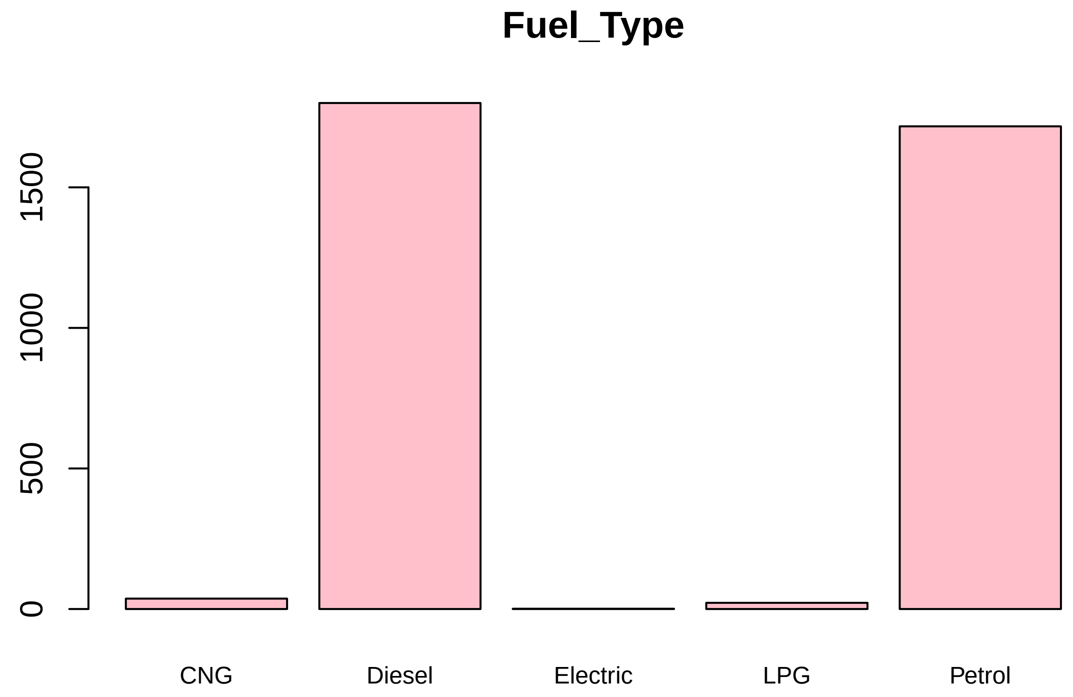


# Bar chart to identify outliers for factor variable

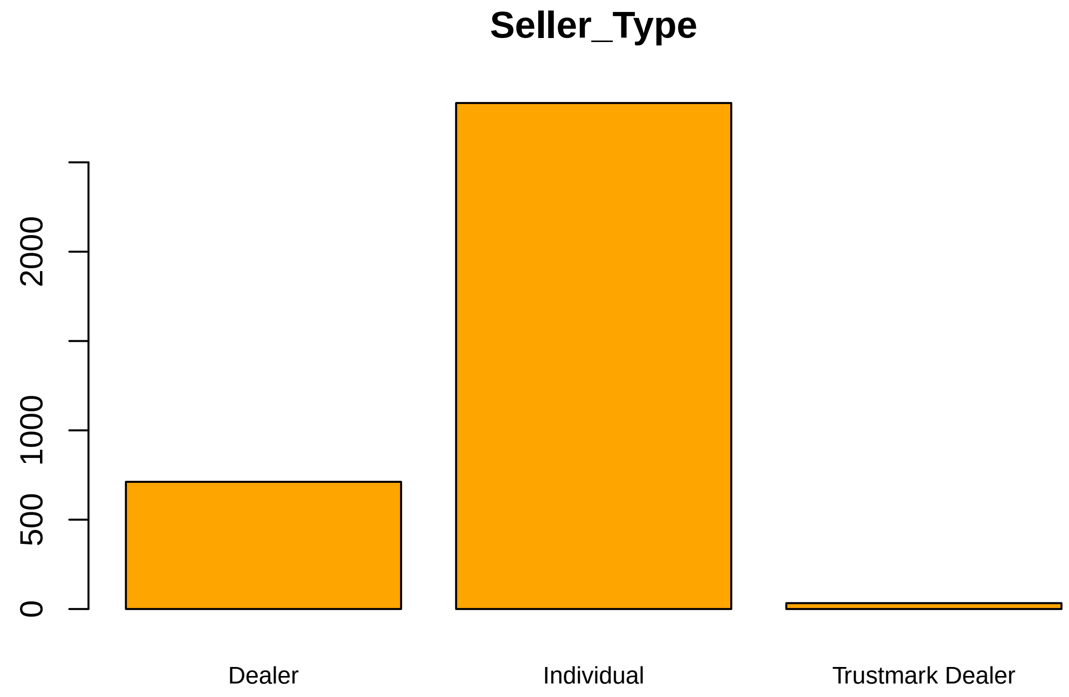
```
barplot(table(dataset$name_JDG ), cex.names=.75,col="lightgreen",las=2,main="Vehicle_Name")
```



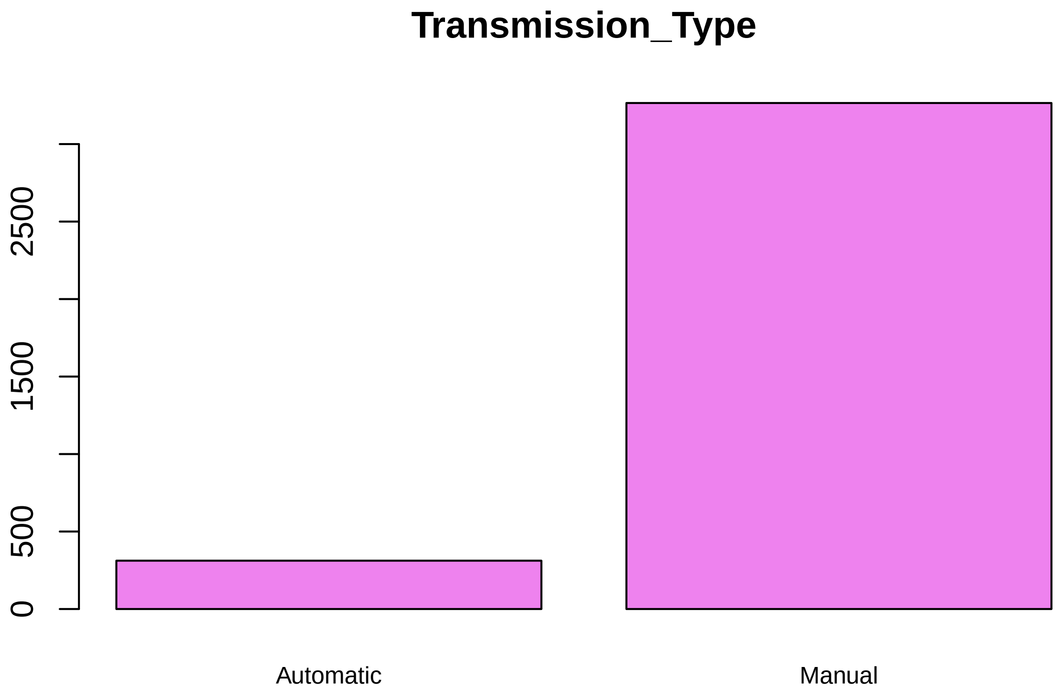
```
barplot(table(dataset$fuel_JDG ), cex.names=.75,col="pink",main="Fuel_Type")
```



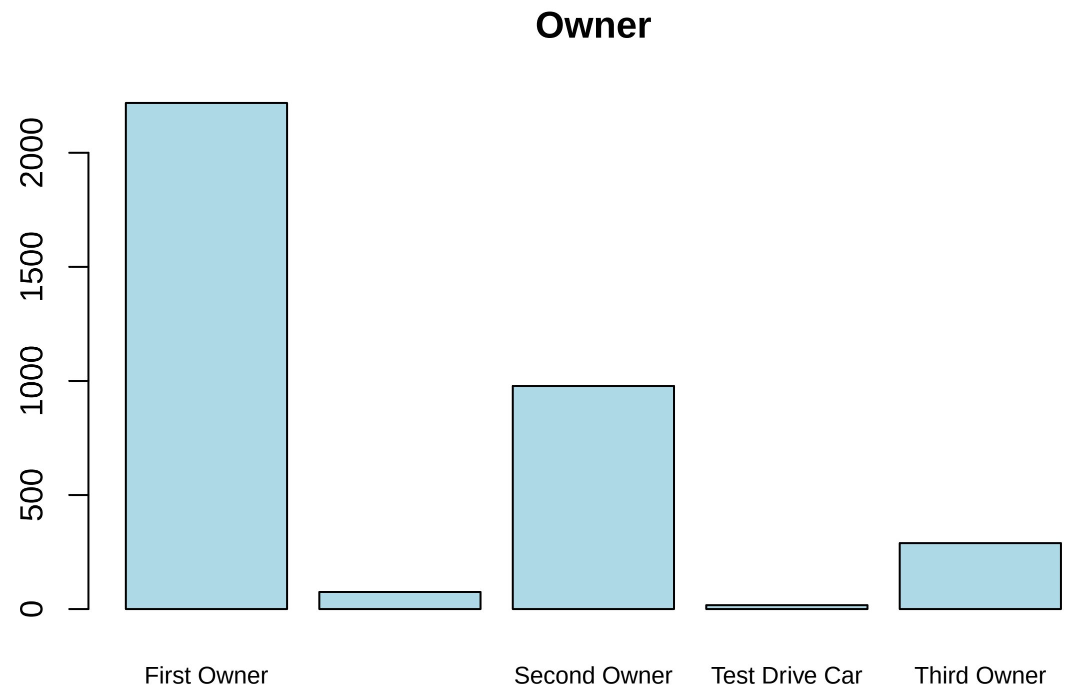
```
barplot(table(dataset$seller_type_JDG ), cex.names=.75,col="orange",main="Seller_Type")
```



```
barplot(table(dataset$transmission_JDG ), cex.names=.75,col="violet",main ="Transmission_Type" )
```



```
barplot(table(dataset$owner_JDG ), cex.names=.75,col="lightblue",main="Owner")
```



```
## removing vehicle name
dataset <- dataset[,-c(1)]
str(dataset)
```

```
## 'data.frame':    3577 obs. of  7 variables:
##  $ year_JDG          : int  2007 2007 2012 2017 2014 2007 2016 2014 2015 2017 ...
##  $ selling_price_JDG: int  60000 135000 600000 250000 450000 140000 550000 240000 850000 365000 ...
##  $ km_driven_JDG    : int  70000 50000 100000 46000 141000 125000 25000 60000 25000 78000 ...
##  $ fuel_JDG         : Factor w/ 5 levels "CNG","Diesel",..: 5 5 2 5 2 5 5 5 5 1 ...
##  $ seller_type_JDG  : Factor w/ 3 levels "Dealer","Individual",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ transmission_JDG : Factor w/ 2 levels "Automatic","Manual": 2 2 2 2 2 2 2 2 2 2 ...
##  $ owner_JDG        : Factor w/ 5 levels "First Owner",..: 1 1 1 1 3 1 1 3 1 1 ...
```

removing year < 2000

```
dataset <- dataset[dataset$year_JDG > 1999,]
str(dataset)
```

```
## 'data.frame':    3552 obs. of  7 variables:
##  $ year_JDG          : int  2007 2007 2012 2017 2014 2007 2016 2014 2015 2017 ...
##  $ selling_price_JDG: int  60000 135000 600000 250000 450000 140000 550000 240000 850000 365000 ...
##  $ km_driven_JDG    : int  70000 50000 100000 46000 141000 125000 25000 60000 25000 78000 ...
##  $ fuel_JDG         : Factor w/ 5 levels "CNG","Diesel",..: 5 5 2 5 2 5 5 5 5 1 ...
##  $ seller_type_JDG  : Factor w/ 3 levels "Dealer","Individual",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ transmission_JDG : Factor w/ 2 levels "Automatic","Manual": 2 2 2 2 2 2 2 2 2 2 ...
##  $ owner_JDG        : Factor w/ 5 levels "First Owner",..: 1 1 1 1 3 1 1 3 1 1 ...
```

FEATURE SELECTION

Filter Feature Selection

1. Identify Low Variance

```
library(pastecs)
summary_df <- stat.desc(dataset)
summary_df
```

```
##                  year_JDG selling_price_JDG km_driven_JDG fuel_JDG
## nbr.val      3.552000e+03      3.552000e+03  3.552000e+03       NA
## nbr.null     0.000000e+00      0.000000e+00  0.000000e+00       NA
## nbr.na       0.000000e+00      0.000000e+00  0.000000e+00       NA
## min          2.000000e+03      2.000000e+04  1.000000e+00       NA
## max          2.020000e+03      8.900000e+06  8.065990e+05       NA
## range        2.000000e+01      8.880000e+06  8.065980e+05       NA
## sum          7.150424e+06      1.692160e+09  2.459072e+08       NA
## median       2.014000e+03      3.500000e+05  6.000000e+04       NA
## mean         2.013070e+03      4.763964e+05  6.923063e+04       NA
## SE.mean      6.823829e-02      8.557010e+03  7.999355e+02       NA
## CI.mean.0.95 1.337902e-01      1.677715e+04  1.568379e+03       NA
## var          1.653976e+01      2.600860e+11  2.272913e+09       NA
## std.dev      4.066910e+00      5.099863e+05  4.767508e+04       NA
## coef.var     2.020253e-03      1.070508e+00  6.886415e-01       NA
##              seller_type_JDG transmission_JDG owner_JDG
## nbr.val                   NA               NA        NA
## nbr.null                  NA               NA        NA
## nbr.na                    NA               NA        NA
## min                       NA               NA        NA
## max                       NA               NA        NA
## range                     NA               NA        NA
## sum                       NA               NA        NA
## median                    NA               NA        NA
## mean                      NA               NA        NA
## SE.mean                   NA               NA        NA
## CI.mean.0.95              NA               NA        NA
## var                       NA               NA        NA
## std.dev                   NA               NA        NA
## coef.var                  NA               NA        NA
```

```
# year_JDG has very low coefficient of variation, which indicates minimal internal variation, and so it

table(dataset$year_JDG)
```

```
##
```

```
## 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015
##   12   16   18   22   38   61   93  117  129  169  210  247  338  298  321  332
## 2016 2017 2018 2019 2020
##  283  346  295  162   45
```

#2. Identify High Correlation

```
library(dplyr)
corr = cor(select(dataset,year_JDG,selling_price_JDG,km_driven_JDG),method="pearson")
corr
```

```
##                     year_JDG selling_price_JDG km_driven_JDG
## year_JDG           1.0000000         0.4277128    -0.4367229
## selling_price_JDG  0.4277128         1.0000000    -0.1872496
## km_driven_JDG     -0.4367229        -0.1872496     1.0000000
```

wrapper feature selection

```
FitStart = lm(selling_price_JDG ~ 1, data=dataset) # lm developed using one feature

FitAll = lm(selling_price_JDG ~., data=dataset) #lm developed using all features


summary(FitAll)
```

```
##
## Call:
## lm(formula = selling_price_JDG ~ ., data = dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1084042  -153233   -25658   103888  7637482
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -7.092e+07  4.018e+06 -17.648  < 2e-16 ***
## year_JDG                         3.580e+04  1.991e+03  17.979  < 2e-16 ***
## km_driven_JDG                   -7.796e-01  1.655e-01  -4.712 2.55e-06 ***
## fuel_JDGDiesel                   2.703e+05  6.390e+04   4.230 2.39e-05 ***
## fuel_JDGElectric                -5.001e+05  3.898e+05  -1.283  0.19963
## fuel_JDGLPG                      2.882e+04  1.049e+05   0.275  0.78361
## fuel_JDGPetrol                   9.490e+03  6.394e+04   0.148  0.88203
## seller_type_JDGIndividual       -5.932e+04  1.692e+04  -3.506  0.00046 ***
## seller_type_JDGTrustmark Dealer  8.778e+04  6.843e+04   1.283  0.19965
## transmission_JDGManual          -7.742e+05  2.328e+04 -33.248  < 2e-16 ***
## owner_JDGFourth & Above Owner   -4.093e+04  4.810e+04  -0.851  0.39483
## owner_JDGSecond Owner           -4.668e+04  1.630e+04  -2.864  0.00420 **
## owner_JDGTest Drive Car          1.797e+05  9.478e+04   1.896  0.05798 .
## owner_JDGThird Owner            -4.772e+04  2.617e+04  -1.824  0.06826 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 383500 on 3538 degrees of freedom
## Multiple R-squared:  0.4367, Adjusted R-squared:  0.4346
## F-statistic:   211 on 13 and 3538 DF,  p-value: < 2.2e-16
```

```
summary(FitStart)
```

```
##
## Call:
## lm(formula = selling_price_JDG ~ 1, data = dataset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -456396 -276396 -126396  123604 8423604
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   476396       8557   55.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 510000 on 3551 degrees of freedom
```

```
step(FitStart,direction = "forward", scope=formula(FitAll))
```

```
## Start:  AIC=93362.76
## selling_price_JDG ~ 1
##
##                     Df  Sum of Sq        RSS   AIC
## + transmission_JDG  1 2.1917e+14 7.0440e+14 92403
## + year_JDG          1 1.6896e+14 7.5461e+14 92647
## + fuel_JDG          4 7.0408e+13 8.5316e+14 93089
## + owner_JDG         4 5.6491e+13 8.6707e+14 93147
## + seller_type_JDG   2 3.2974e+13 8.9059e+14 93238
## + km_driven_JDG     1 3.2382e+13 8.9118e+14 93238
## <none>                           9.2357e+14 93363
##
## Step:  AIC=92402.51
## selling_price_JDG ~ transmission_JDG
##
##                   Df  Sum of Sq        RSS   AIC
## + year_JDG         1 1.2708e+14 5.7731e+14 91698
## + fuel_JDG         4 6.4237e+13 6.4016e+14 92071
## + owner_JDG        4 4.3461e+13 6.6093e+14 92184
## + km_driven_JDG    1 1.7777e+13 6.8662e+14 92314
## + seller_type_JDG  2 1.2696e+13 6.9170e+14 92342
## <none>                          7.0440e+14 92403
##
## Step:  AIC=91697.84
## selling_price_JDG ~ transmission_JDG + year_JDG
##
##                   Df  Sum of Sq        RSS   AIC
## + fuel_JDG         4 4.7188e+13 5.3013e+14 91403
## + seller_type_JDG  2 4.0308e+12 5.7328e+14 91677
## + owner_JDG        4 2.1959e+12 5.7512e+14 91692
## + km_driven_JDG    1 4.8463e+11 5.7683e+14 91697
## <none>                          5.7731e+14 91698
##
## Step:  AIC=91402.95
## selling_price_JDG ~ transmission_JDG + year_JDG + fuel_JDG
##
##                   Df  Sum of Sq        RSS   AIC
## + km_driven_JDG    1 4.5816e+12 5.2555e+14 91374
## + seller_type_JDG  2 4.1347e+12 5.2599e+14 91379
## + owner_JDG        4 3.9236e+12 5.2620e+14 91385
## <none>                          5.3013e+14 91403
##
## Step:  AIC=91374.12
## selling_price_JDG ~ transmission_JDG + year_JDG + fuel_JDG +
##     km_driven_JDG
##
##                   Df  Sum of Sq        RSS   AIC
## + seller_type_JDG  2 3.4537e+12 5.2209e+14 91355
## + owner_JDG        4 2.9294e+12 5.2262e+14 91362
## <none>                          5.2555e+14 91374
##
## Step:  AIC=91354.7
## selling_price_JDG ~ transmission_JDG + year_JDG + fuel_JDG +
##     km_driven_JDG + seller_type_JDG
##
##             Df  Sum of Sq        RSS   AIC
## + owner_JDG  4 1.8357e+12 5.2026e+14 91350
## <none>                    5.2209e+14 91355
##
## Step:  AIC=91350.19
## selling_price_JDG ~ transmission_JDG + year_JDG + fuel_JDG +
##     km_driven_JDG + seller_type_JDG + owner_JDG
```

```
##
## Call:
## lm(formula = selling_price_JDG ~ transmission_JDG + year_JDG +
##     fuel_JDG + km_driven_JDG + seller_type_JDG + owner_JDG, data = dataset)
##
## Coefficients:
##                     (Intercept)           transmission_JDGManual
##                      -7.092e+07                       -7.742e+05
##                        year_JDG                   fuel_JDGDiesel
##                       3.580e+04                        2.703e+05
##                fuel_JDGElectric                      fuel_JDGLPG
##                      -5.001e+05                        2.882e+04
##                  fuel_JDGPetrol                    km_driven_JDG
##                       9.490e+03                       -7.796e-01
##       seller_type_JDGIndividual  seller_type_JDGTrustmark Dealer
##                      -5.932e+04                        8.778e+04
##   owner_JDGFourth & Above Owner            owner_JDGSecond Owner
##                      -4.093e+04                       -4.668e+04
##         owner_JDGTest Drive Car             owner_JDGThird Owner
##                       1.797e+05                       -4.772e+04
```

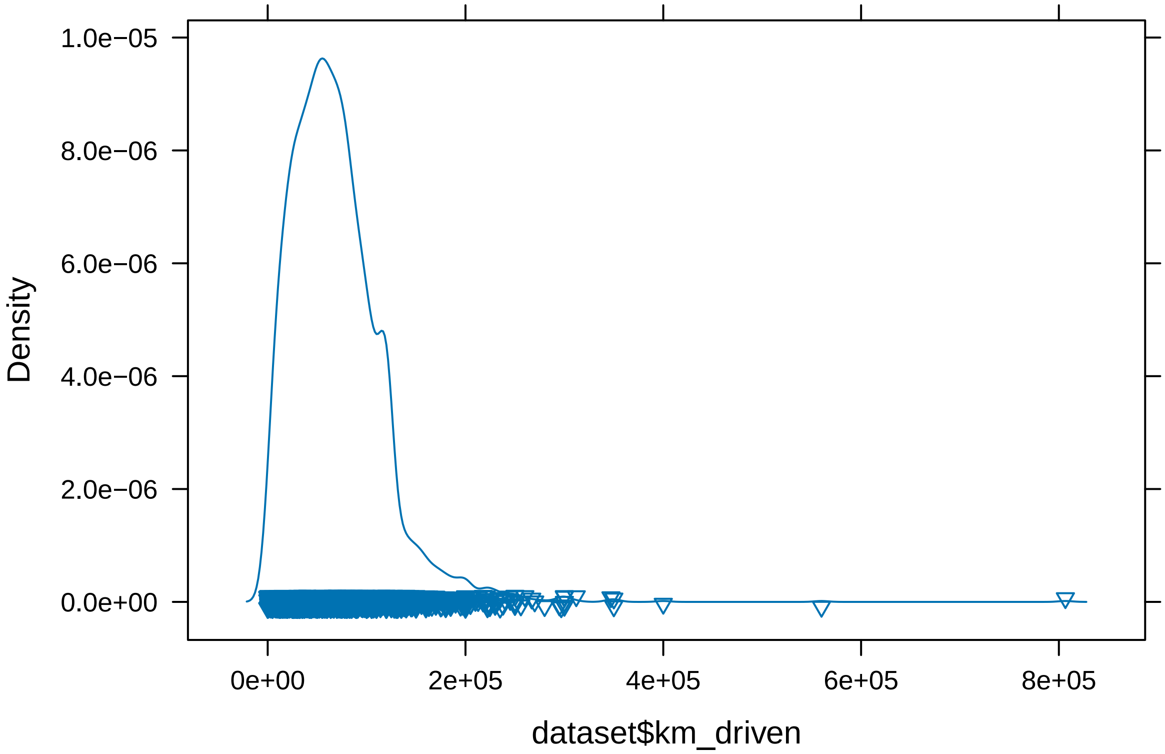
```
step(FitAll,direction = "backward")
```

```
## Start:  AIC=91350.19
## selling_price_JDG ~ year_JDG + km_driven_JDG + fuel_JDG + seller_type_JDG + 
##     transmission_JDG + owner_JDG
## 
##                     Df  Sum of Sq        RSS   AIC
## <none>                           5.2026e+14 91350
## - owner_JDG          4 1.8357e+12 5.2209e+14 91355
## - seller_type_JDG    2 2.3599e+12 5.2262e+14 91362
## - km_driven_JDG      1 3.2644e+12 5.2352e+14 91370
## - year_JDG           1 4.7535e+13 5.6779e+14 91659
## - fuel_JDG           4 5.0881e+13 5.7114e+14 91674
## - transmission_JDG  1 1.6255e+14 6.8280e+14 92314
```

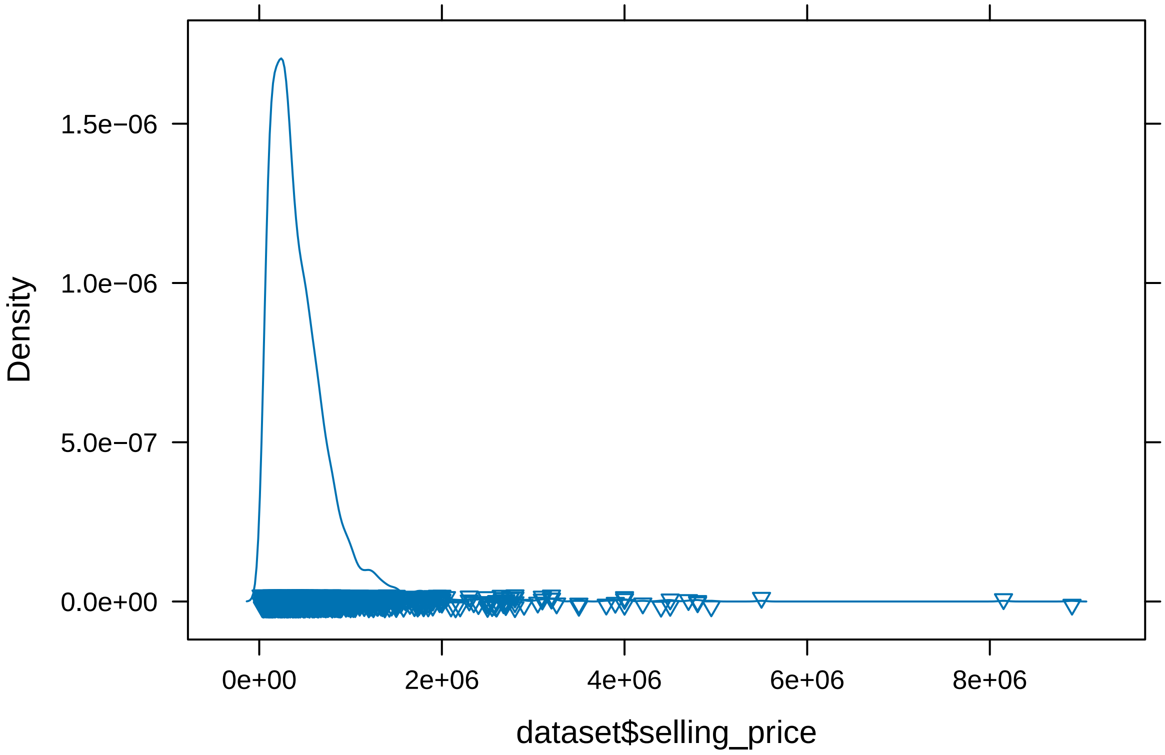
```
## 
## Call:
## lm(formula = selling_price_JDG ~ year_JDG + km_driven_JDG + fuel_JDG + 
##     seller_type_JDG + transmission_JDG + owner_JDG, data = dataset)
## 
## Coefficients:
##                     (Intercept)                         year_JDG  
##                      -7.092e+07                        3.580e+04  
##                   km_driven_JDG                   fuel_JDGDiesel  
##                      -7.796e-01                        2.703e+05  
##                fuel_JDGElectric                      fuel_JDGLPG  
##                      -5.001e+05                        2.882e+04  
##                  fuel_JDGPetrol        seller_type_JDGIndividual  
##                       9.490e+03                       -5.932e+04  
## seller_type_JDGTrustmark Dealer           transmission_JDGManual  
##                       8.778e+04                       -7.742e+05  
##   owner_JDGFourth & Above Owner            owner_JDGSecond Owner  
##                      -4.093e+04                       -4.668e+04  
##         owner_JDGTest Drive Car             owner_JDGThird Owner  
##                       1.797e+05                       -4.772e+04
```

#Density Plots for further details

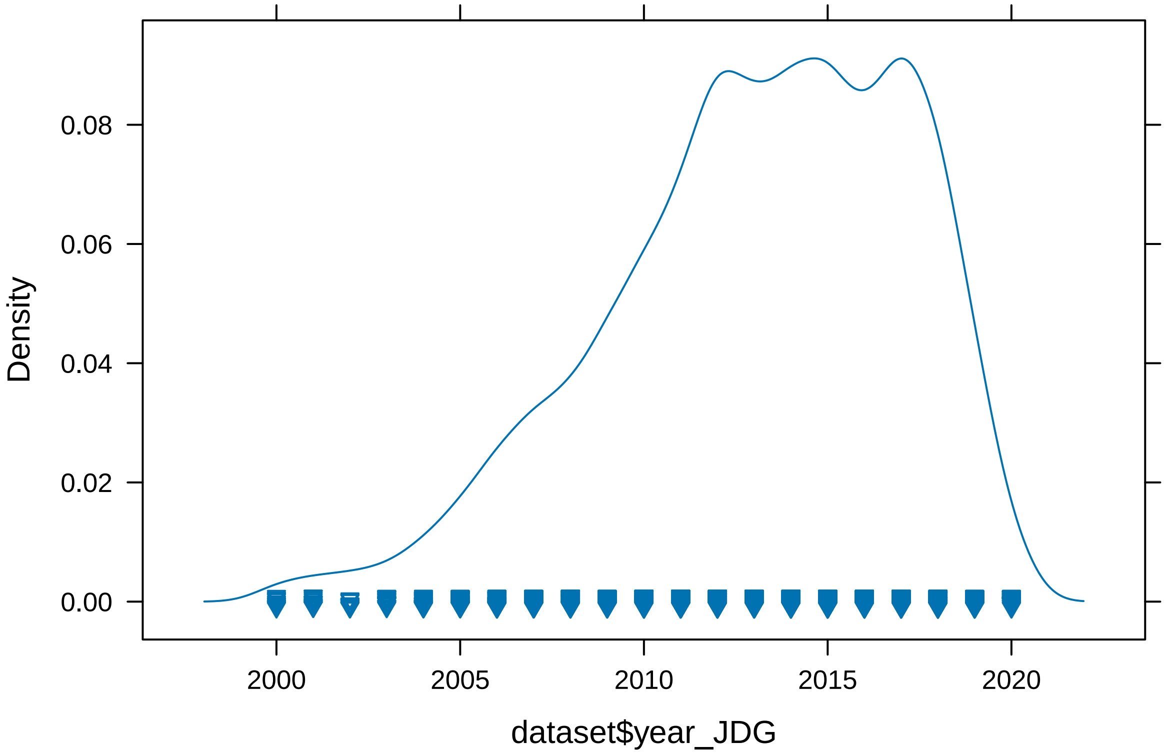
```
densityplot( ~ dataset$km_driven, pch=6)
```



```
densityplot( ~ dataset$selling_price, pch=6)
```



```
densityplot( ~ dataset$year_JDG, pch=6)
```



## Data Scaling

We could see from density plot that numerical data of dataset is not normally distributed (Km and selling price is right skewed and year is left skewed).Without scaling the machine learning models will try to disregard coefficients of features that has low values because their impact will be very small as compared to the higher values. In this project, we use Tukeys ladder method find the power transformation that makes the transformed data as close to normal as possible (normality) and stabilizes the variance (homoscedasticity)

```
if(!require(rcompanion)){install.packages("rcompanion")}
```

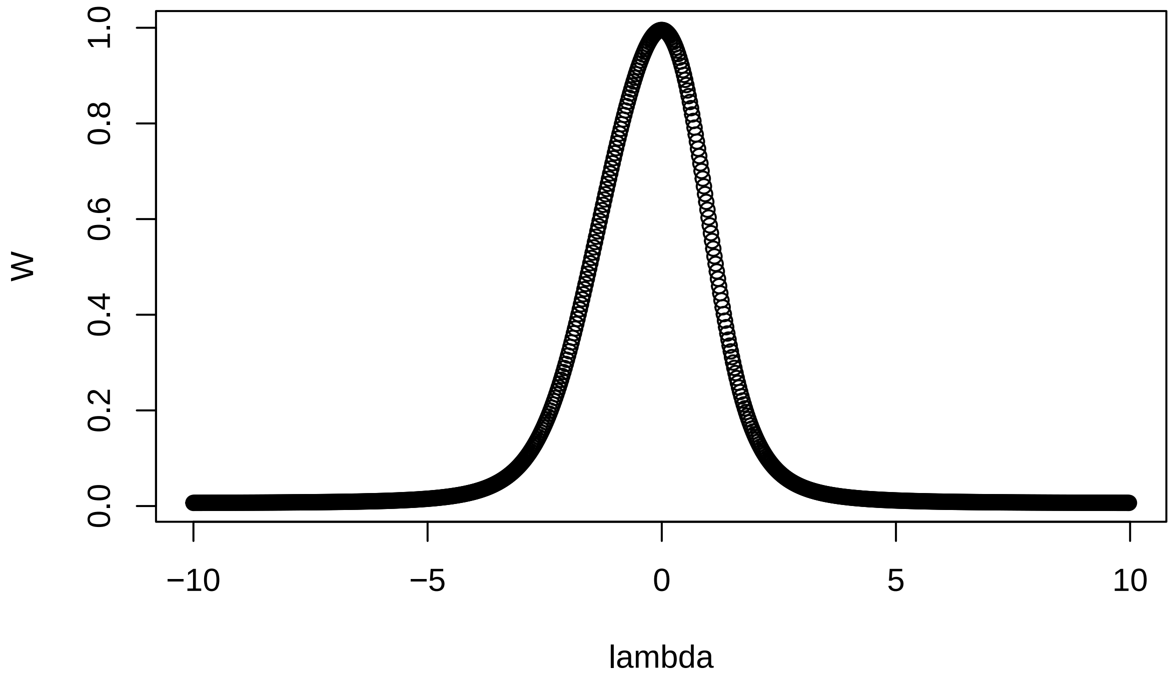
```
## Loading required package: rcompanion
```

```
library("rcompanion")#Q-Q plot
if(!require(psych)){install.packages("psych")}
```

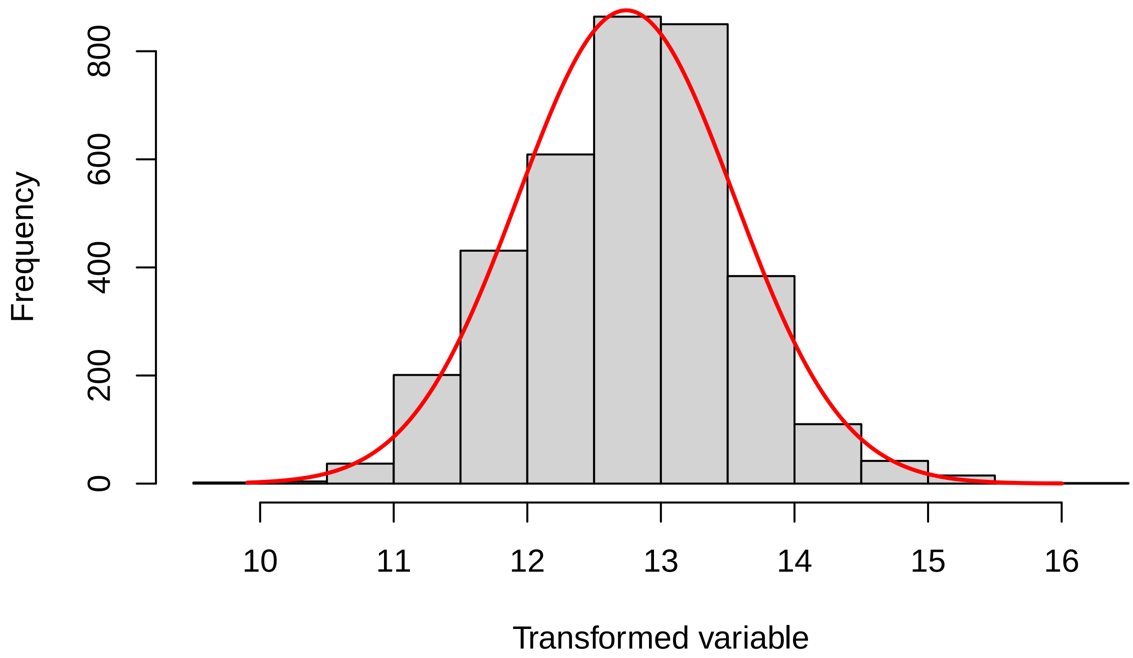
```
## Loading required package: psych

##
## Attaching package: 'psych'

## The following object is masked from 'package:rcompanion':
##
##     phi

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

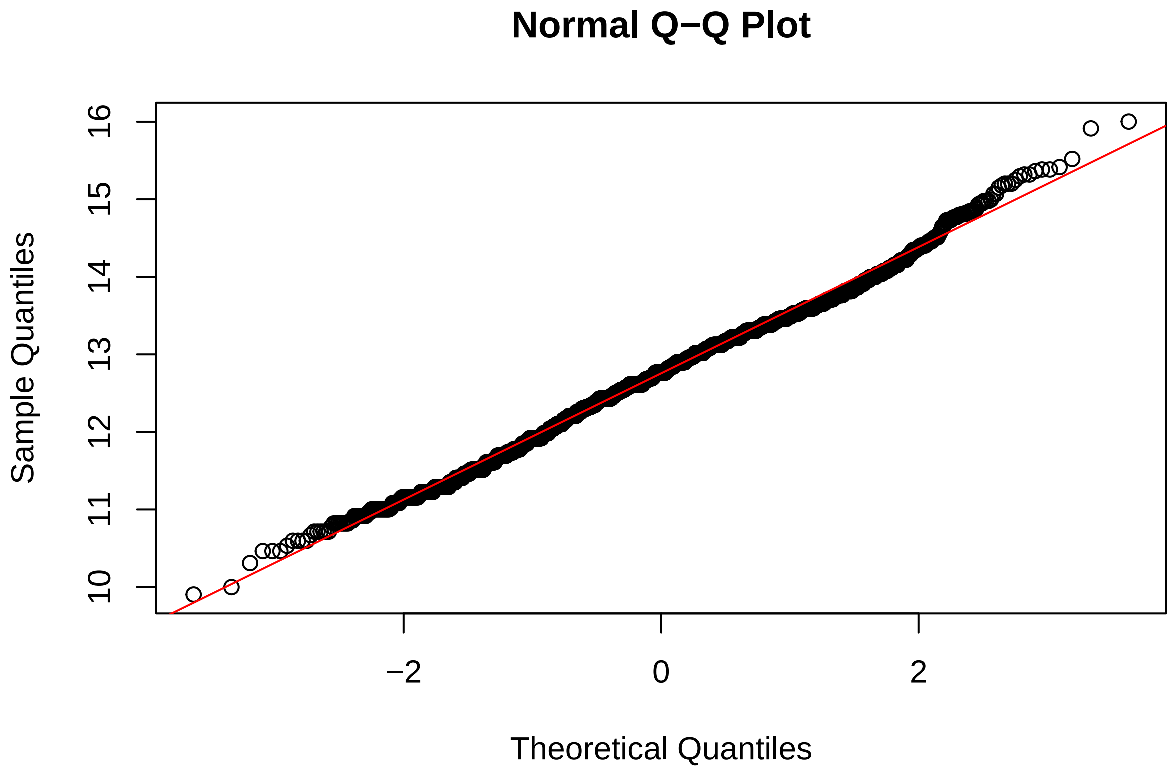
```
library("psych")
```

```
### Tukey's Ladder for selling Price
selling_price_JDG = transformTukey(dataset$selling_price_JDG,plotit=TRUE)
```

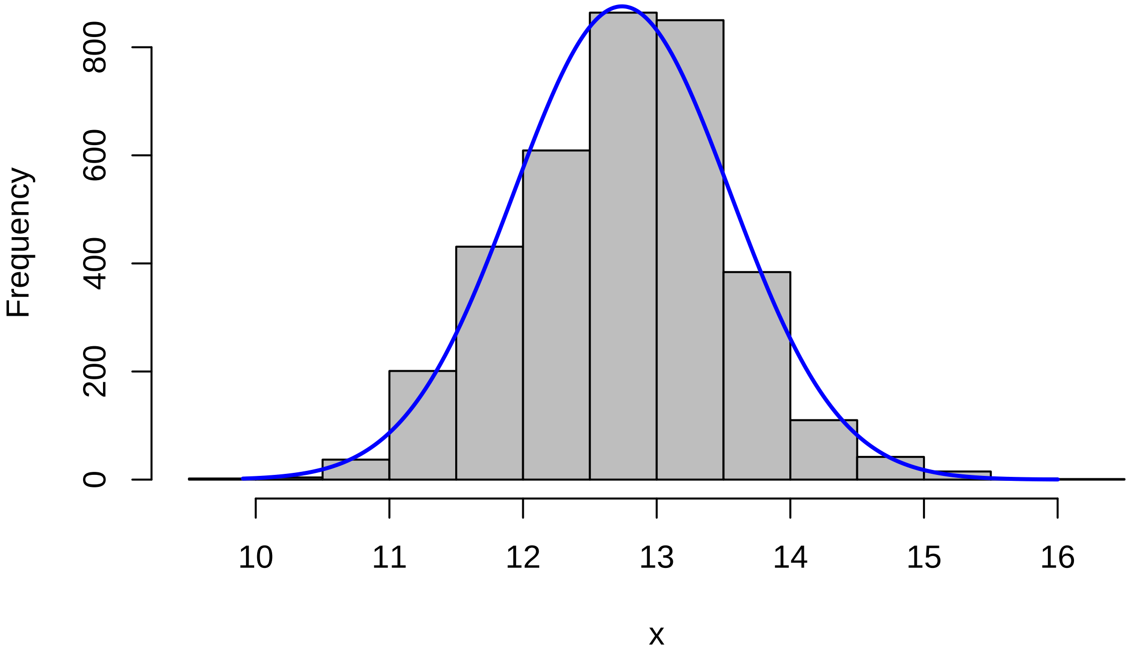


```
##
##     lambda      W Shapiro.p.value
## 401      0 0.9954       4.696e-09
##
## if (lambda >  0){TRANS = x ^ lambda}
## if (lambda == 0){TRANS = log(x)}
## if (lambda <  0){TRANS = -1 * x ^ lambda}
```


Frequency
0
200
400
600
800
10
11
12
13
14
15
16
Transformed variable



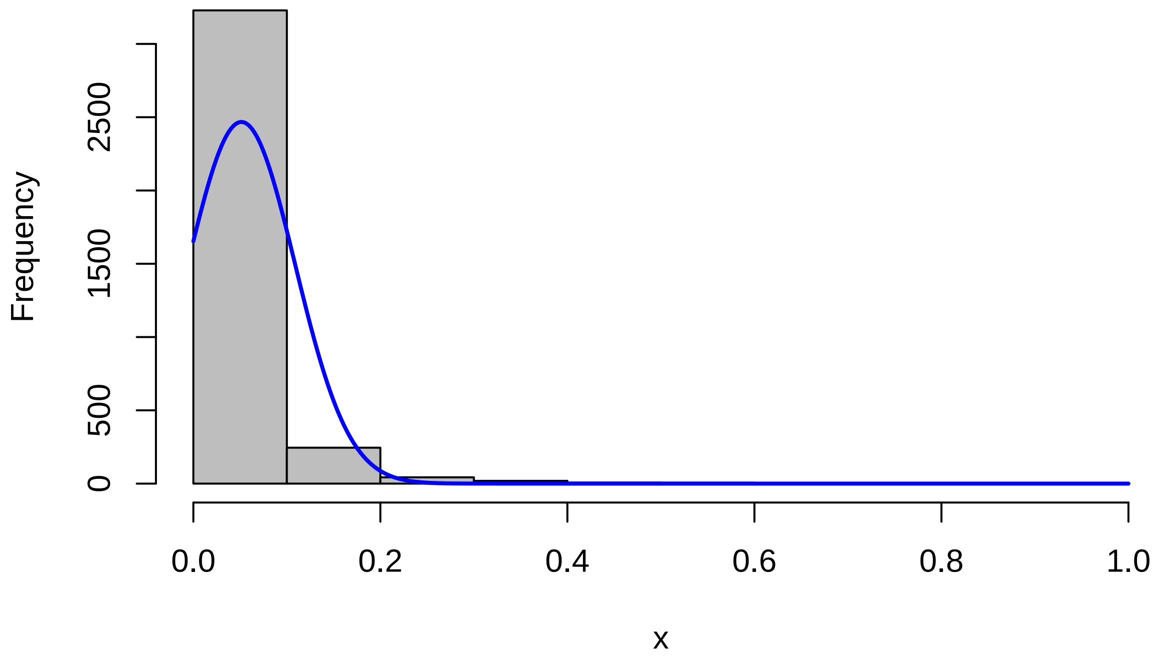
```
plotNormalHistogram(selling_price_JDG)
```



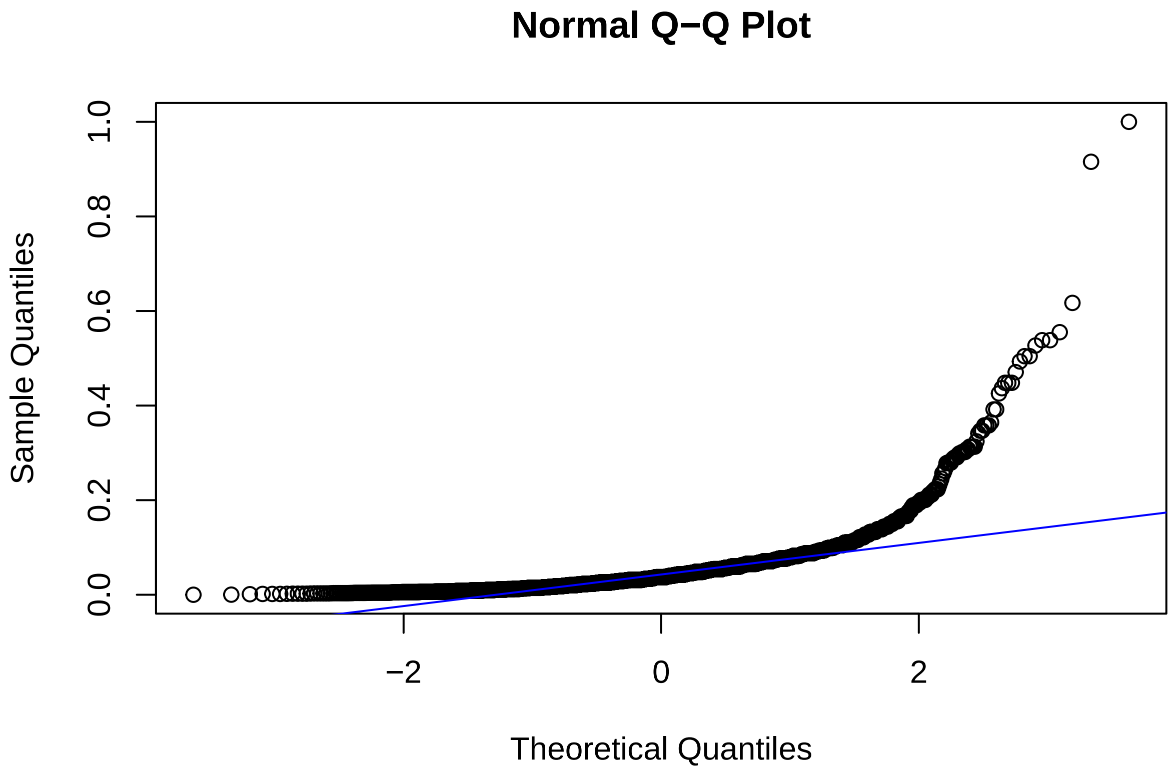
```
#Q-Q Plot
##qqnorm(dataset$selling_price_JDG) #the result shows it is not normal distributed.
##qqline(dataset$selling_price_JDG,col="blue")
```

Using Min-Max scaling

```
# Calculate the minimum and maximum values
min_value <- min(dataset$selling_price_JDG)
max_value <- max(dataset$selling_price_JDG)

# Perform Min-Max scaling
dataset$selling_price_JDG <- (dataset$selling_price_JDG - min_value) / (max_value - min_value)

plotNormalHistogram(dataset$selling_price_JDG)
```



```
#Q-Q Plot
qqnorm(dataset$selling_price_JDG) #the result shows it is not normal distributed.
qqline(dataset$selling_price_JDG,col="blue")
```



## Split the data using “Two way Hold out validation”

```
m=length(dataset$selling_price_JDG)
cat("Dataset size: ",m)
```

```
## Dataset size:  3552
```

```
set.seed(456)
train_index=sample(m,m*0.8)
train_set <- dataset[train_index, ]
number_train_set <- length(train_set$year_JDG)
cat("\nNumber of train set: ",number_train_set)
```

```
##
## Number of train set:  2841
```

```
test_set <-dataset[-train_index,]
number_test_set <- length(test_set$year_JDG)
cat("\nNumber of test set: ",number_test_set)
```

```
##
## Number of test set:  711
```

Performing Wilcox test for numerical values containing attributes to check if both columns have no evidence of statistically significant difference

```r
wilcox.test(train_set$year_JDG, test_set$year_JDG)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  train_set$year_JDG and test_set$year_JDG
## W = 1039189, p-value = 0.2309
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(train_set$selling_price_JDG, test_set$selling_price_JDG)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  train_set$selling_price_JDG and test_set$selling_price_JDG
## W = 995905, p-value = 0.565
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(train_set$km_driven_JDG, test_set$km_driven_JDG)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  train_set$km_driven_JDG and test_set$km_driven_JDG
## W = 1004495, p-value = 0.8226
## alternative hypothesis: true location shift is not equal to 0
```

```r
#wilcox.test(train_set$Number_of_years, test_set$Number_of_years)
```

Since all the wilcox test performed on all the attributes has p-value greater than the significance level 0.05, we do not have enough evidence to reject the null hypothesis. Therefore, we can conclude that there is no significant difference in the distribution of all the attributes between the train_set and test_set data.

Model1: Multi-Linear Regression Model using "Two way Hold Validation"

```r
start_time_mlr <- Sys.time()
mlr_model = lm( selling_price_JDG ~ ., data=train_set, na.action=na.omit)
end_time_mlr <- Sys.time()
mlr_Time <- end_time_mlr - start_time_mlr
cat("Time taken to train the model is ",mlr_Time)
```

```
## Time taken to train the model is  0.005219221
```

```
print("Model Decsription:")
```

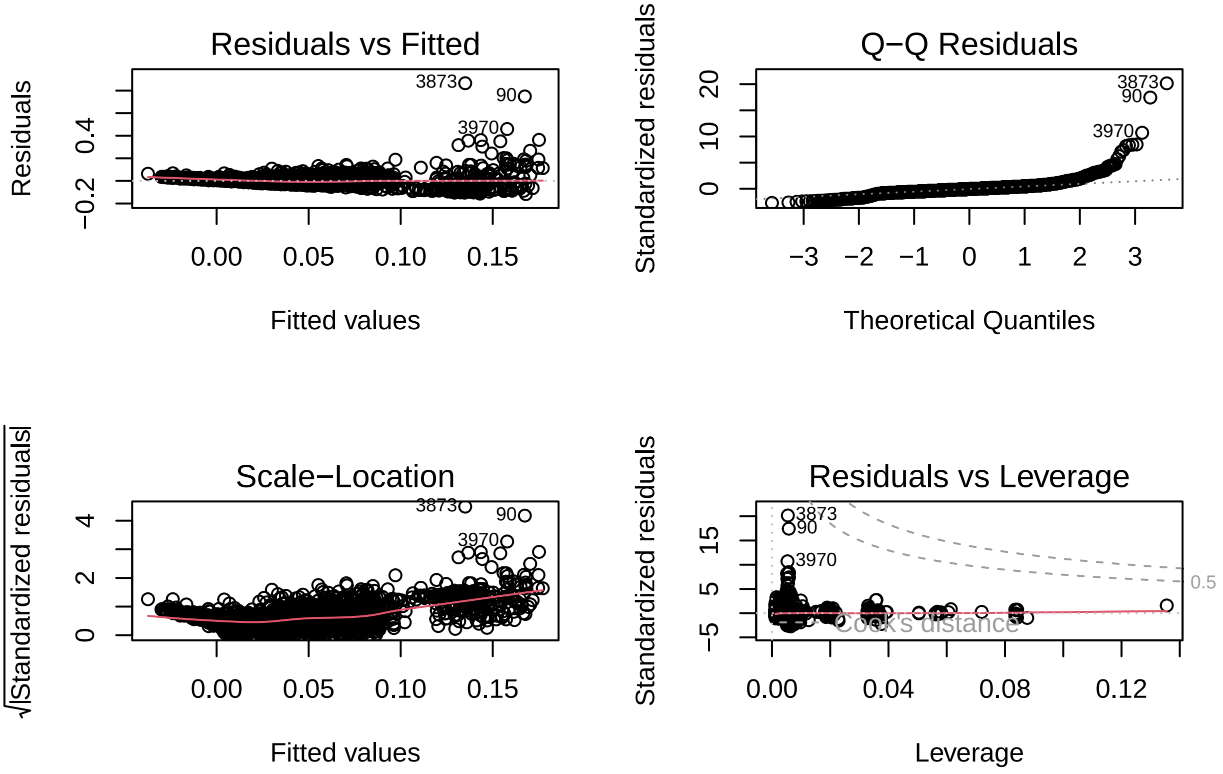
```
## [1] "Model Decsription:"
```

```
summary(mlr_model)
```

```
##
## Call:
## lm(formula = selling_price_JDG ~ ., data = train_set, na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.11677 -0.01737 -0.00296  0.01156  0.86496
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -8.147e+00  5.074e-01 -16.056  < 2e-16 ***
## year_JDG                         4.108e-03  2.515e-04  16.336  < 2e-16 ***
## km_driven_JDG                   -7.747e-08  2.108e-08  -3.674 0.000243 ***
## fuel_JDGDiesel                   2.870e-02  8.090e-03   3.547 0.000395 ***
## fuel_JDGElectric                -5.304e-02  4.392e-02  -1.208 0.227326
## fuel_JDGLPG                      2.418e-03  1.292e-02   0.187 0.851596
## fuel_JDGPetrol                   9.291e-04  8.094e-03   0.115 0.908623
## seller_type_JDGIndividual       -7.615e-03  2.113e-03  -3.604 0.000319 ***
## seller_type_JDGTrustmark Dealer  1.290e-02  7.947e-03   1.623 0.104636
## transmission_JDGManual          -8.210e-02  2.931e-03 -28.013  < 2e-16 ***
## owner_JDGFourth & Above Owner   -2.404e-03  6.038e-03  -0.398 0.690597
## owner_JDGSecond Owner           -3.544e-03  2.049e-03  -1.729 0.083848 .
## owner_JDGTest Drive Car          2.598e-02  1.263e-02   2.057 0.039762 *
## owner_JDGThird Owner            -4.685e-03  3.334e-03  -1.405 0.160047
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.04301 on 2827 degrees of freedom
## Multiple R-squared:  0.4164, Adjusted R-squared:  0.4138
## F-statistic: 155.2 on 13 and 2827 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
plot(mlr_model)
```

```
## Warning: not plotting observations with leverage one:
##   667
```



###RMSE Evaluation

```
### Train Set
pred <- predict(mlr_model, newdata=train_set)
RMSE_trn_full <- sqrt(mean((train_set$selling_price_JDG - pred)^2))
cat("RMSE value of train set: ",round(RMSE_trn_full,2))
```

```
## RMSE value of train set:  0.04
```

```
### Test Set
pred <- predict(mlr_model, newdata=test_set)
RMSE_trn_full <- sqrt(mean((test_set$selling_price_JDG - pred)^2))
cat("RMSE value of test set: ",round(RMSE_trn_full,2))
```

```
## RMSE value of test set:  0.04
```

Split data using K-fold cross validation

```
set.seed(456)
# defining training control as cross-validation and K=10
train_control <- trainControl(method = "cv", number = 10)

# training the model by assigning sales column
# as target variable and rest other column as independent variable
```

```
model_cv <- train(selling_price_JDG ~., data = dataset,
               method = "lm",
               trControl = train_control)
```

```
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
```

```
# printing model performance metrics
# along with other details
print(model_cv)
```

```
## Linear Regression
##
## 3552 samples
##    6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3197, 3197, 3197, 3196, 3194, 3197, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.04236024  0.4565369  0.02330913
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Split data using 10-times-10-fold Cross-Validation

```
set.seed(456)
# defining training control as cross-validation and K=10
train_control <- trainControl(method = "repeatedcv", number = 10, repeats=10)

# training the model by assigning sales column
# as target variable and rest other column as independent variable
model_cv2 <- train(selling_price_JDG ~., data = dataset,
                  method = "lm",
                  trControl = train_control)
```

```
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
```

```r
# printing model performance metrics
# along with other details
print(model_cv2)
```

```
## Linear Regression
##
## 3552 samples
##    6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 3197, 3197, 3197, 3196, 3194, 3197, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.04241863  0.4546043  0.02329786
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Model 2: Support Vector Regression

```r
#install.packages("e1071")
library(e1071)
```

Model 2: Support Vector Regression

```r
start_time_svr <- Sys.time()
svr_model <- svm(selling_price_JDG ~ ., data = train_set, type = 'eps-regression', na.action = na.omit)
end_time_svr <- Sys.time()
svr_Time <- end_time_svr - start_time_svr
cat("Time taken to train the SVR model is ",svr_Time)
```

```
## Time taken to train the SVR model is  0.7942622
```

```
summary(svr_model)
```

```
##
## Call:
## svm(formula = selling_price_JDG ~ ., data = train_set, type = "eps-regression",
##     na.action = na.omit)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.07142857
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1940
```

RMSE Evluation of SVR

```
### Train Set
pred <- predict(svr_model, newdata=train_set)
RMSE_trn_full <- sqrt(mean((train_set$selling_price_JDG - pred)^2))
cat("RMSE value of train_set",round(RMSE_trn_full,2))
```

```
## RMSE value of train_set 0.04
```

```
### Test Set
pred <- predict(svr_model, newdata=test_set)
RMSE_trn_full <- sqrt(mean((test_set$selling_price_JDG - pred)^2))
cat("RMSE value of test_set",round(RMSE_trn_full,2))
```

```
## RMSE value of test_set 0.04
```

Model 3: Random Forest Regression Model

```
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:psych':
##
##     outlier

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
start_time_rf <- Sys.time()
rf_model <- randomForest(selling_price_JDG ~ ., data = train_set, na.action = na.roughfix)
end_time_rf <- Sys.time()
rf_Time <- end_time_rf - start_time_rf
cat("Time taken to train the RF model is ",rf_Time)
```

```
## Time taken to train the RF model is  2.830544
```

```
summary(rf_model)
```

```
##                 Length Class  Mode
## call               4   -none- call
## type               1   -none- character
## predicted       2841   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       2841   -none- numeric
## importance         6   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               2841   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

RMSE Evluation of RF

```
### Train Set
pred <- predict(rf_model, newdata=train_set)
RMSE_trn_full <- sqrt(mean((train_set$selling_price_JDG - pred)^2))
cat("RMSE value of train_set",round(RMSE_trn_full,2))
```

```
## RMSE value of train_set 0.03
```

```r
### Test Set
pred <- predict(rf_model, newdata=test_set)
RMSE_trn_full <- sqrt(mean((test_set$selling_price_JDG - pred)^2))
cat("RMSE value of test_set",round(RMSE_trn_full,2))
```

```
## RMSE value of test_set 0.04
```