Used Car Modeling and Analysis

Emma Lait, Nikitha Patel, Chang Sun, Krishna Agarwal, Lu Li

20/02/2022

Introduction

Selling a car second hand can be a complex process, and there are multiple factors that affect the selling price of a vehicle. A car's fuel type, year, mileage, and transmission type can all have an impact on the price of a car, but price also depends on who is selling the car and how many previous owners it has had. The goal of this project is to examine the factors that affect a car's selling price with various modeling techniques. We also want to use modeling techniques to predict factors like fuel type and transmission type based on the factors provided in our dataset.

Dataset

The dataset used for our analysis is from Kaggle, published with an open licence. The dataset has 8127 rows and 13 columns. After removing rows with empty and N/A values, the dataset has 7905 rows. Below is a list of the variables in the dataset and an explanation.

1. Name: make and model of the car.
2. Year: year that the car was manufactured
3. Selling_price: selling price of the vehicle in dollars
4. Km_driven: total number of kilometers driven
5. Fuel: fuel type
    - 0: CNG (compressed natural gas)
    - 1: Diesel
    - 2: LPG (liquefied petroleum gas)
    - 3: Petrol
6. Seller_type:
    - 0: Dealer
    - 1: Individual
    - 2: Trustmark Dealer
7. Transmission:
    - 0: Automatic
    - 1: Manual
8. Owner:
    - 0: First Owner
    - 1: Fourth & Above Owner
    - 2: Second Owner
    - 3: Test Drive Car

    - 4: Third Owner
9. Mileage: fuel efficiency, values are in km/L and km/kg
10. Engine: engine capacity (CC)
11. Max_power: maximum brake horsepower of the engine
12. Torque: rpm
13. Seats: number of seats in the vehicle

Data Preparation

After importing the data, we checked the names and dimension of the dataset, then removed empty and NA values.

```
## Loading required package: readxl

## Warning: package 'truncnorm' was built under R version 4.1.2

## Warning: package 'survey' was built under R version 4.1.2

## Loading required package: grid

## Loading required package: Matrix

## Loading required package: survival

##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##     dotchart

## Warning: package 'sampling' was built under R version 4.1.2

##
## Attaching package: 'sampling'

## The following objects are masked from 'package:survival':
##
##     cluster, strata

## Warning: package 'mlbench' was built under R version 4.1.2

## Warning: package 'ISLR' was built under R version 4.1.2

## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:sampling':
##
##     cluster

## The following object is masked from 'package:survival':
##
##     cluster

## Warning: package 'klaR' was built under R version 4.1.2

## Loading required package: carData

##                            name year selling_price km_driven   fuel seller_type
## 1  Skoda Rapid 1.5 TDI Ambition 2014        370000    120000 Diesel  Individual
## 2      Honda City 2017-2020 EXi 2006        158000    140000 Petrol  Individual
## 3     Hyundai i20 Sportz Diesel 2010        225000    127000 Diesel  Individual
## 4        Maruti Swift VXI BSIII 2007        130000    120000 Petrol  Individual
## 5 Hyundai Xcent 1.2 VTVT E Plus 2017        440000     45000 Petrol  Individual
## 6  Maruti Wagon R LXI DUO BSIII 2007         96000    175000    LPG  Individual
##   transmission        owner    mileage  engine  max_power
## 1       Manual Second Owner 21.14 kmpl 1498 CC 103.52 bhp
## 2       Manual  Third Owner  17.7 kmpl 1497 CC     78 bhp
## 3       Manual  First Owner  23.0 kmpl 1396 CC     90 bhp
## 4       Manual  First Owner  16.1 kmpl 1298 CC   88.2 bhp
## 5       Manual  First Owner 20.14 kmpl 1197 CC  81.86 bhp
## 6       Manual  First Owner 17.3 km/kg 1061 CC   57.5 bhp
##                     torque seats
## 1      250Nm@ 1500-2500rpm     5
## 2    12.7@ 2,700(kgm@ rpm)     5
## 3 22.4 kgm at 1750-2750rpm     5
## 4    11.5@ 4,500(kgm@ rpm)     5
## 5        113.75nm@ 4000rpm     5
## 6     7.8@ 4,500(kgm@ rpm)     5

##  [1] "name"          "year"          "selling_price" "km_driven"
##  [5] "fuel"          "seller_type"   "transmission"  "owner"
##  [9] "mileage"       "engine"        "max_power"     "torque"
## [13] "seats"

## [1] 8127   13

##          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        torque         seats
##           215           222           221

## [1] 7905   13
```

After removing empty rows, we were left with 7905 observations.

Next we converted the categorical variables to factors and split the units from the mileage, engine, and max_power columns so that we were just left with the values.

The torque column has a high multicollinearity issue with the max_power column because horsepower = torque x engine speed. This relationship between the two columns led us to remove the torque column. The names column was also removed because it was not useful for our analysis.

```
##   year selling_price km_driven   fuel seller_type transmission        owner
## 1 2014        370000    120000 Diesel  Individual       Manual Second Owner
## 2 2006        158000    140000 Petrol  Individual       Manual  Third Owner
## 3 2010        225000    127000 Diesel  Individual       Manual  First Owner
## 4 2007        130000    120000 Petrol  Individual       Manual  First Owner
## 5 2017        440000     45000 Petrol  Individual       Manual  First Owner
## 6 2007         96000    175000    LPG  Individual       Manual  First Owner
##   mileage engine max_power seats
## 1   21.14   1498    103.52     5
## 2   17.70   1497     78.00     5
## 3   23.00   1396     90.00     5
## 4   16.10   1298     88.20     5
## 5   20.14   1197     81.86     5
## 6   17.30   1061     57.50     5
```

Exploratory Analysis

In this section, we looked at each variable individually and how it related to selling price. For each chart, selling price was log transformed so that we could get a better understanding of the relationships that the variables had with selling price. The qualitative data will be represented in Figures 1-6 and the quantitative variables will be represented in Figues 7-10.

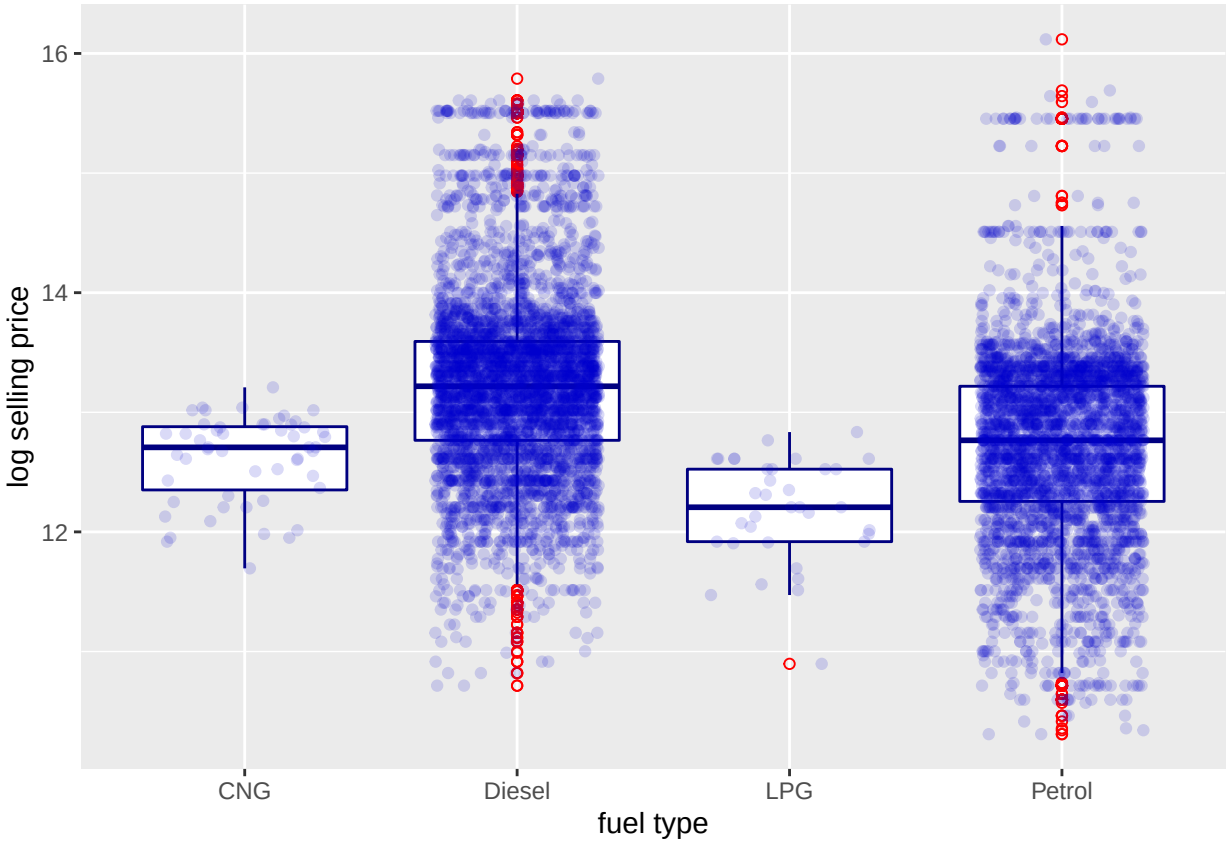


Figure 1: Boxplot of the log selling price and fuel type for used cars. Each blue point represents a car, and red points represent outliers.

From the above boxplot, we can see that most of the cars in this dataset use either Diesel or Petrol, and very few use CNG or LPG. Visually, there does not appear to be a large difference in means between the fuel types.

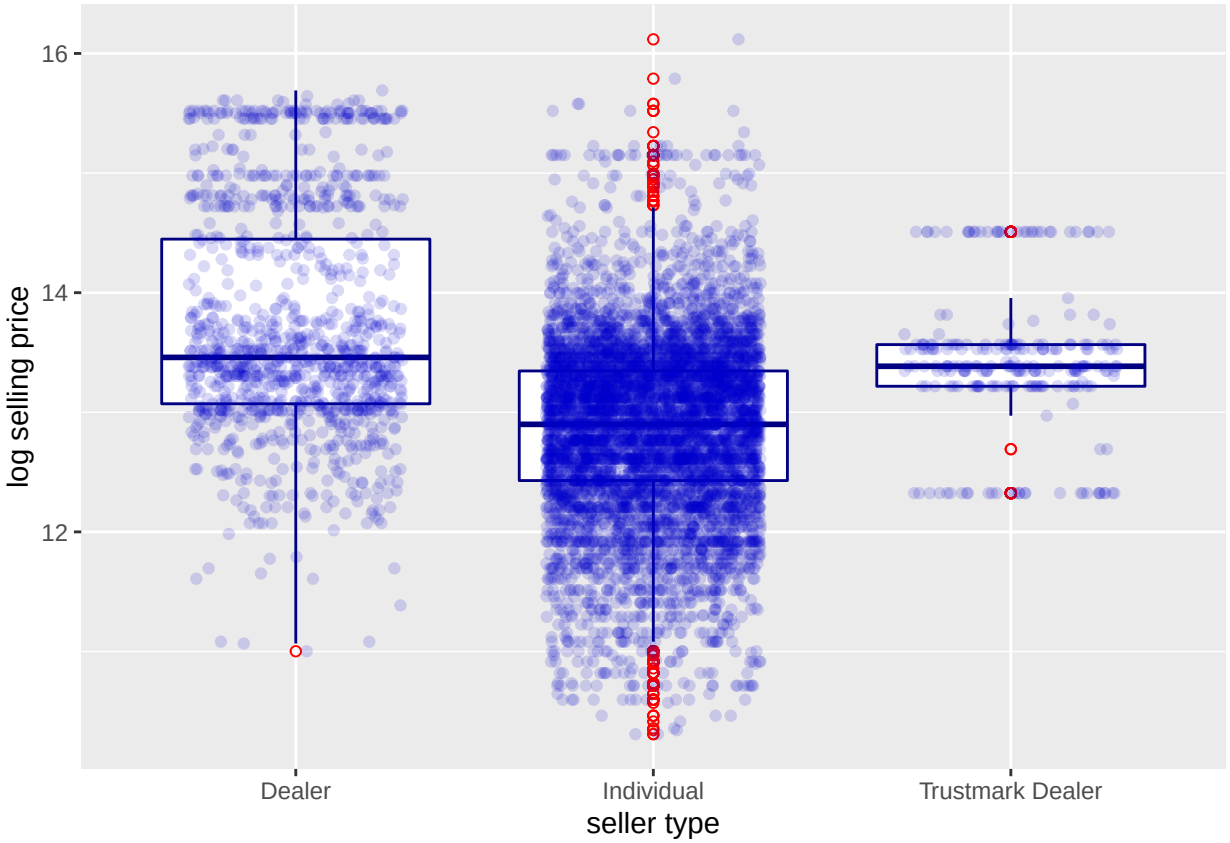


Figure 2: Boxplot of the log selling price and seller type for used cars. Each blue point represents a car, and red points represent outliers.

The above boxplot shows that most of the sellers of used cars are individuals, and there are not as many dealers or trustmark dealers that have sold cars in this dataset.

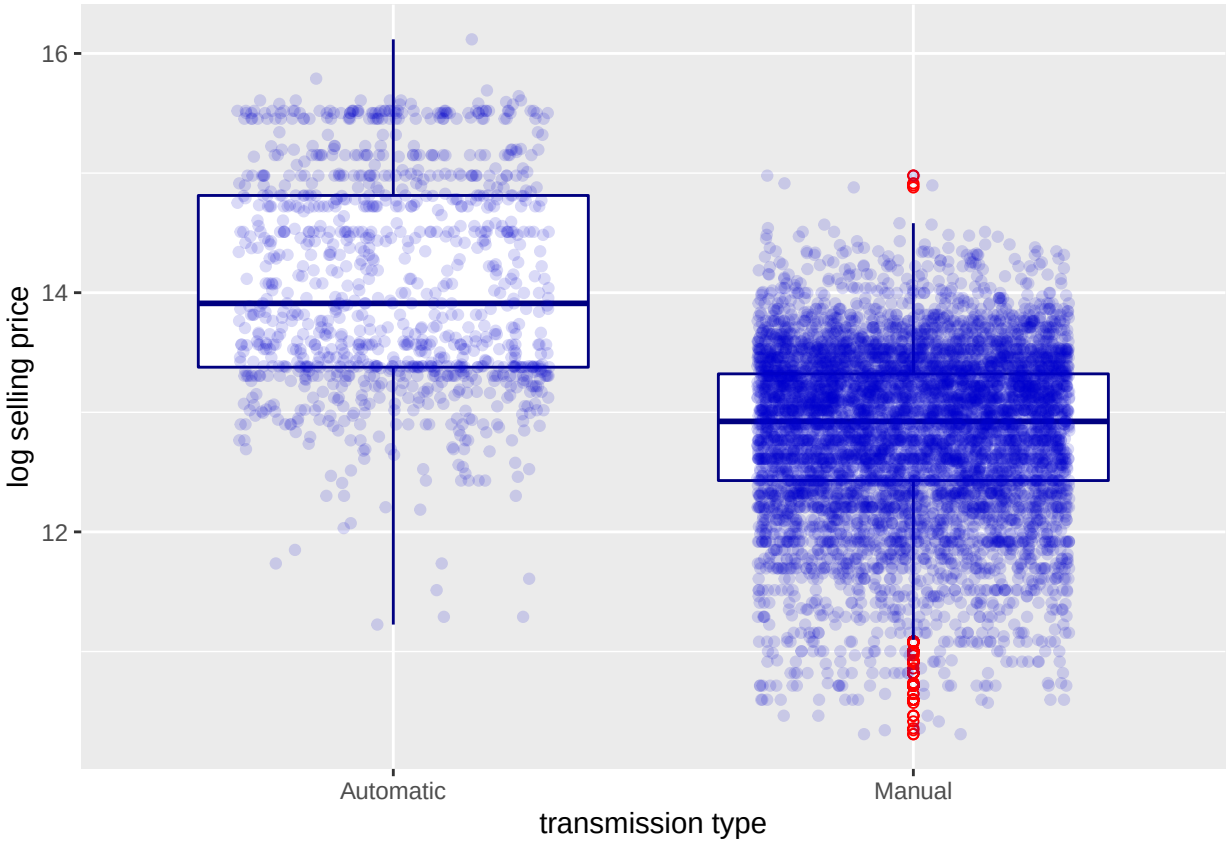


Figure 3: Boxplot of the log selling price and transmission type for used cars. Each blue point represents a car, and red points represent outliers.

Most of the cars sold in this dataset have a manual transmission, and it appears that cars sold with an automatic transmission are sold for a higher price, on average.

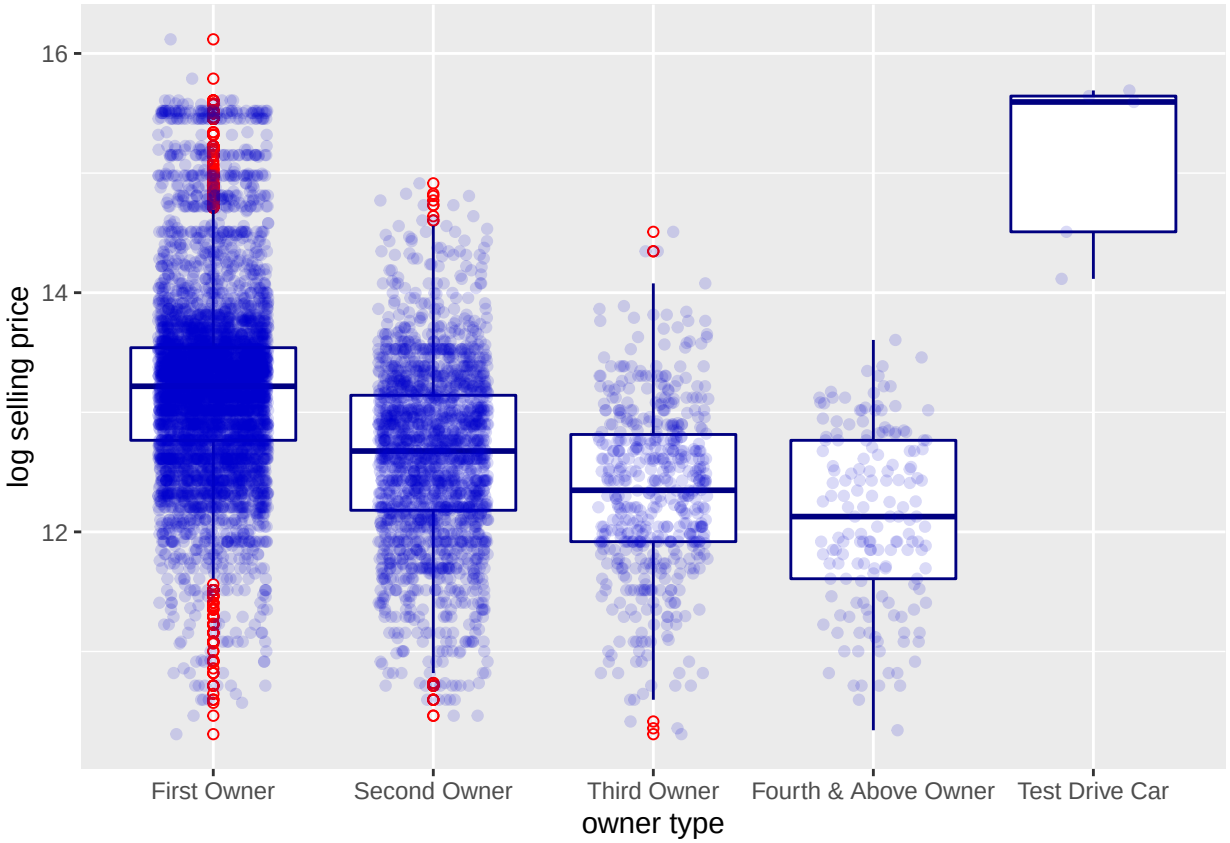


Figure 4: Boxplot of the log selling price and owner type for used cars. Each blue point represents a car, and red points represent outliers.

The above plot shows that of the owner types, most cars in the dataset are sold by the first owner. The number of cars sold decreases with the second, third, and fourth owners, and very few cars are sold as test drive cars with no previous owner. The selling price appears to decrease with each subsequent owner, and the test drive cars sell for a much higher price than those with previous owners.

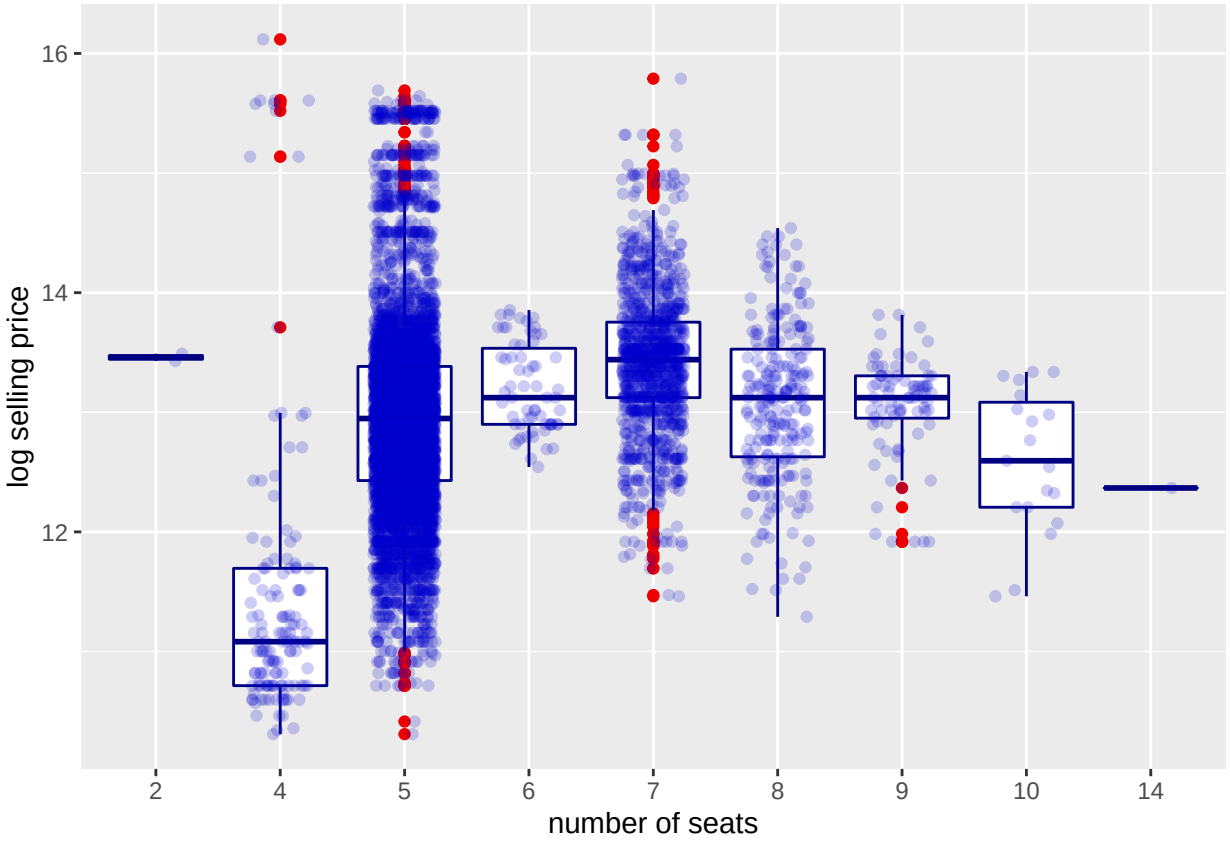


Figure 6: Boxplot of the log selling price and number of seats for used cars. Each blue point represents a car, and red points represent outliers.

We can see with this plot that most used cars have 5 or 7 seats, and there are far fewer cars sold with a different number of seats.

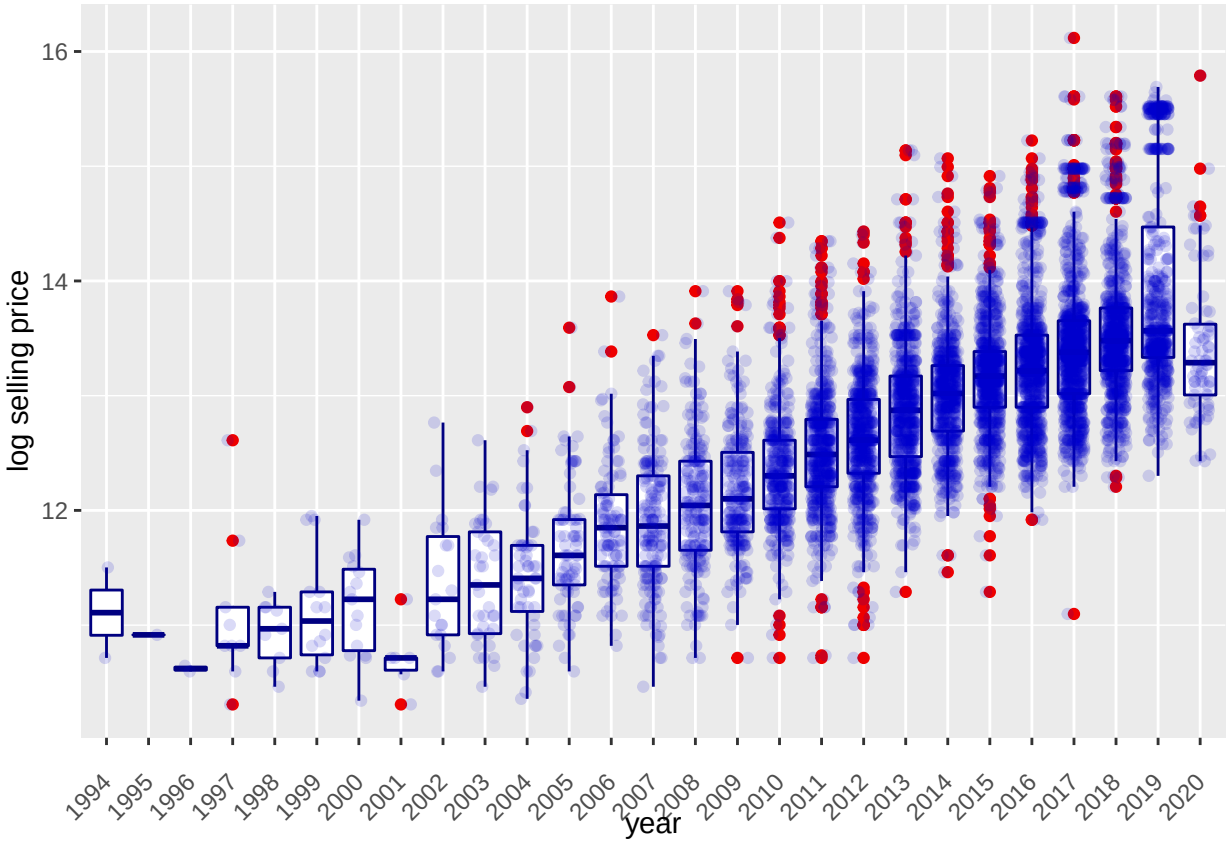


Figure 6: Boxplot of the log selling price and fuel type for used cars. Each blue point represents a car, and red points represent outliers.

The above graph shows that most of the cars in this dataset were manufactured between 2009 and 2019, although there are cars in the dataset sold outside of those years. We can see a relationship with price and year, where newer cars sell for a higher price on average.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

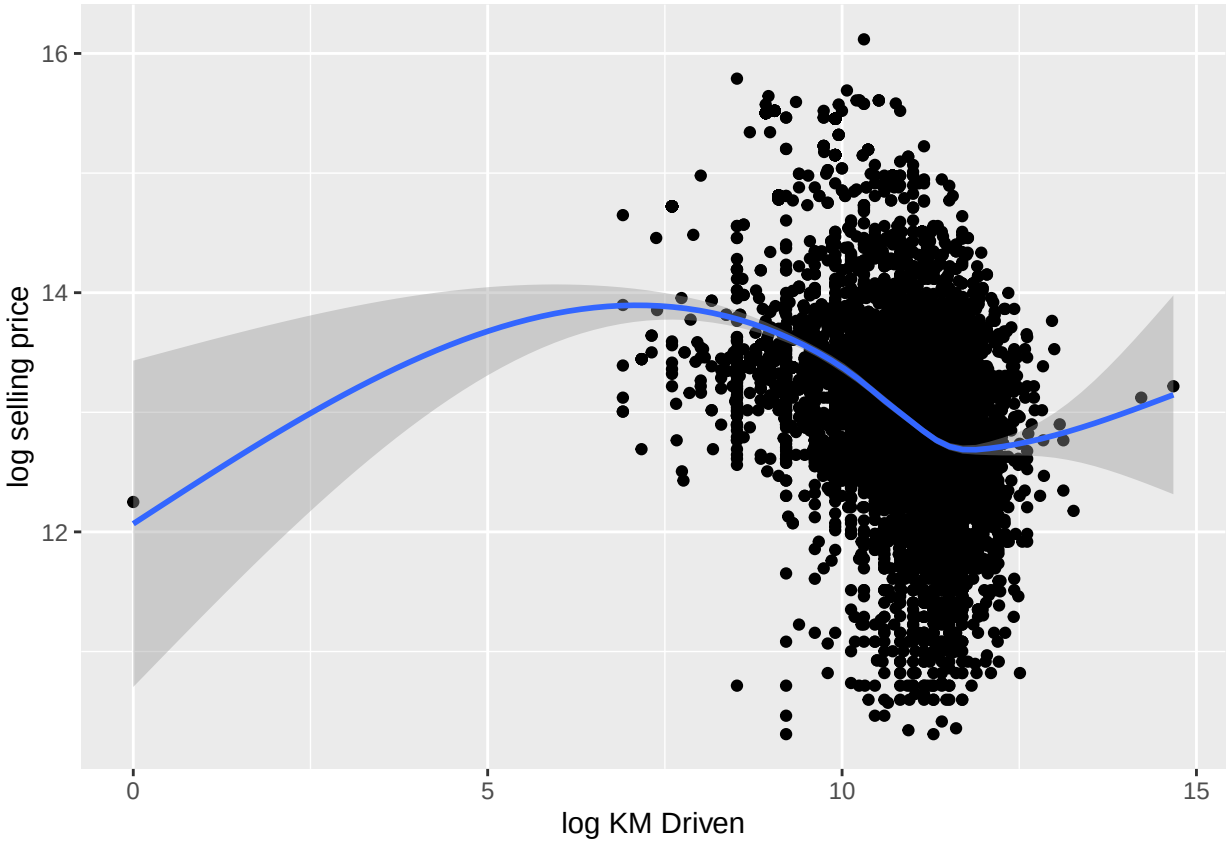


Figure 7: Scatterplot of the log selling price and log km driven for used cars.

From the above plot comparing selling price and km driven, no significant conclusions can be drawn about relationships between these two variables. There does appear to be a slight negative relationship between these two variables, where selling price decreases with higher km driven.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

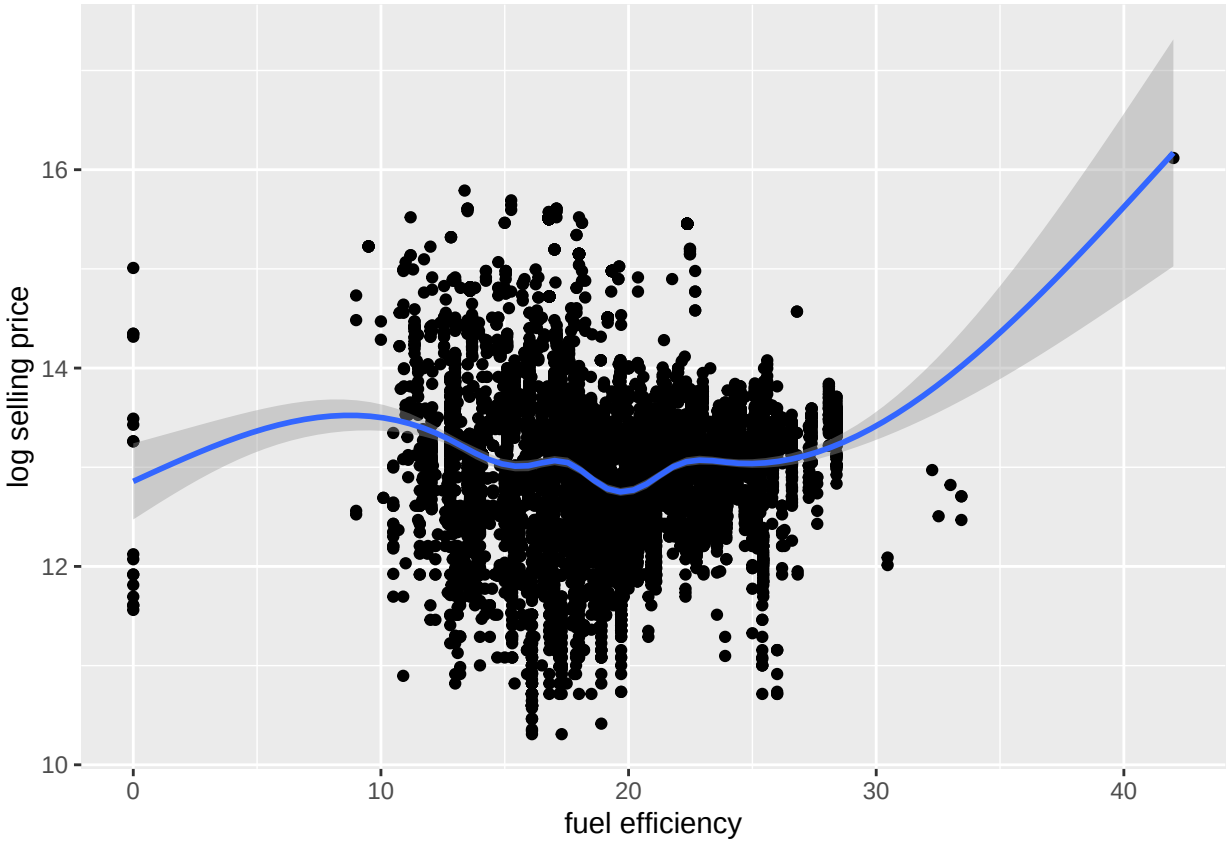


Figure 8: Scatterplot of the log selling price and fuel efficiency for used cars.

The above scatterplot does not show much of a relationship between selling price and fuel efficiency.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

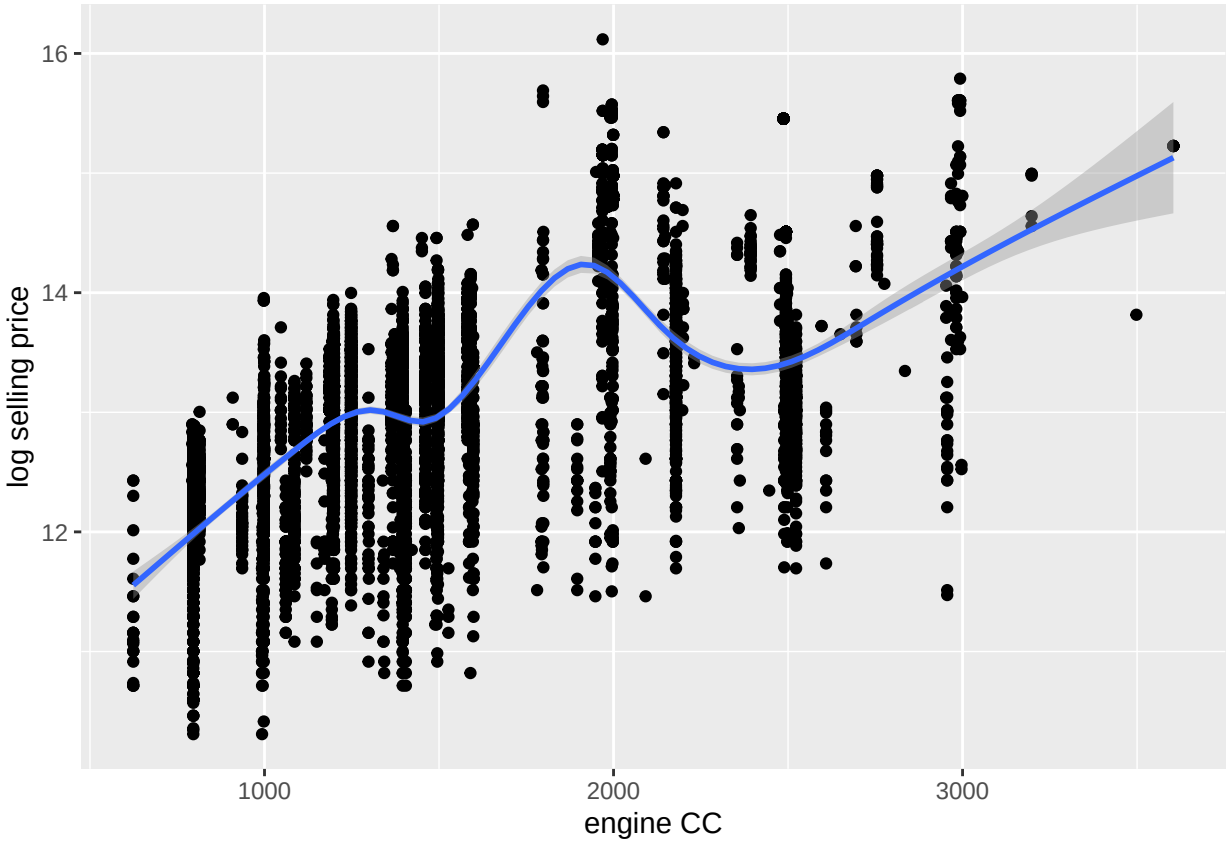


Figure 7: Scatterplot of the log selling price and engine CC for used cars.

When comparing selling price and engine CC, we can see a slight positive relationship between these two variables, with selling price increasinng as engine CC increases.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

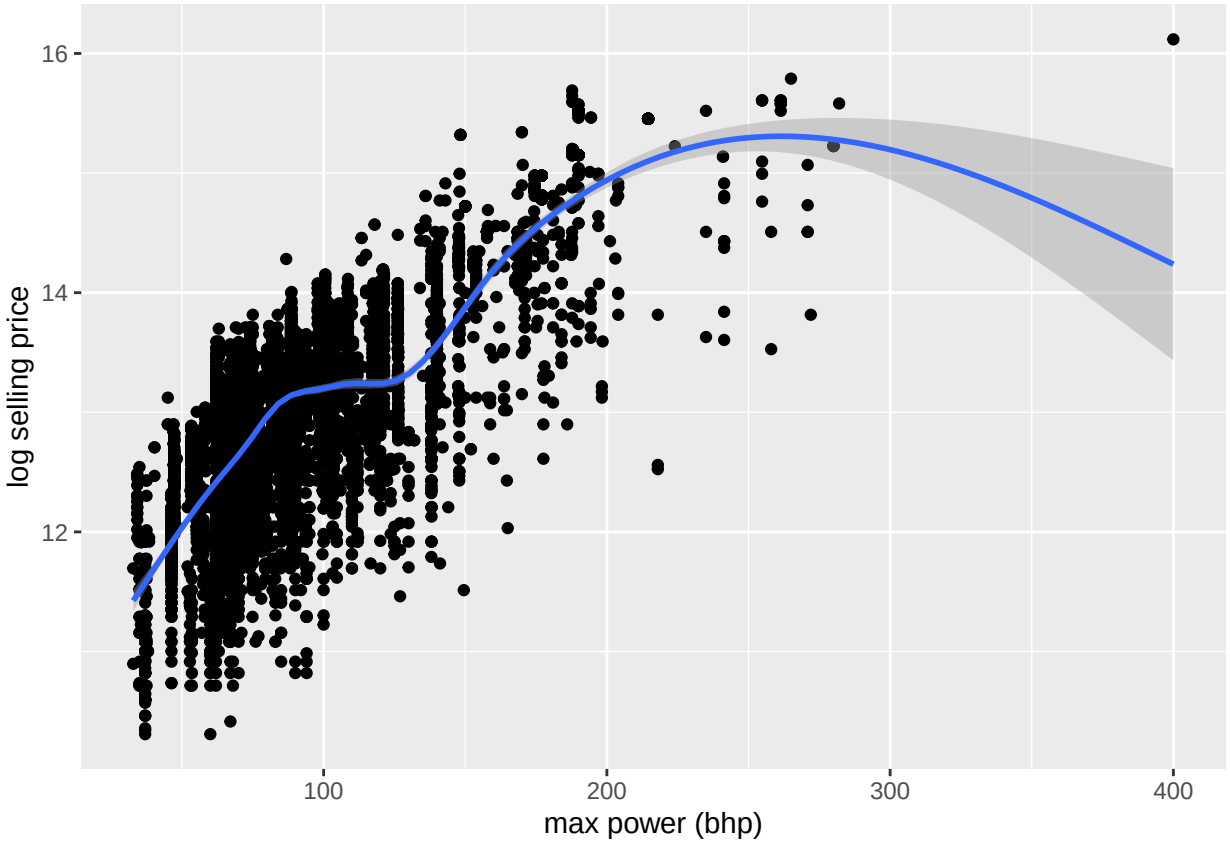


Figure 7: Scatterplot of the log selling price and max power for used cars.

The relationship between selling price and max power also appears to have a positive linear relationship, showing that selling price increases with max engine power.

Price Modeling and Prediction Using Regression Tree and Linear Regression

In this section, we will be performing a regression tree analysis and linear regression analysis to examine the relationship between price and the other variables in our dataset.

To examine the ditribution of price, a price histogram is plotted:

```
hist(car$selling_price,nclass=50,main = "",xlab = "Selling price")
```

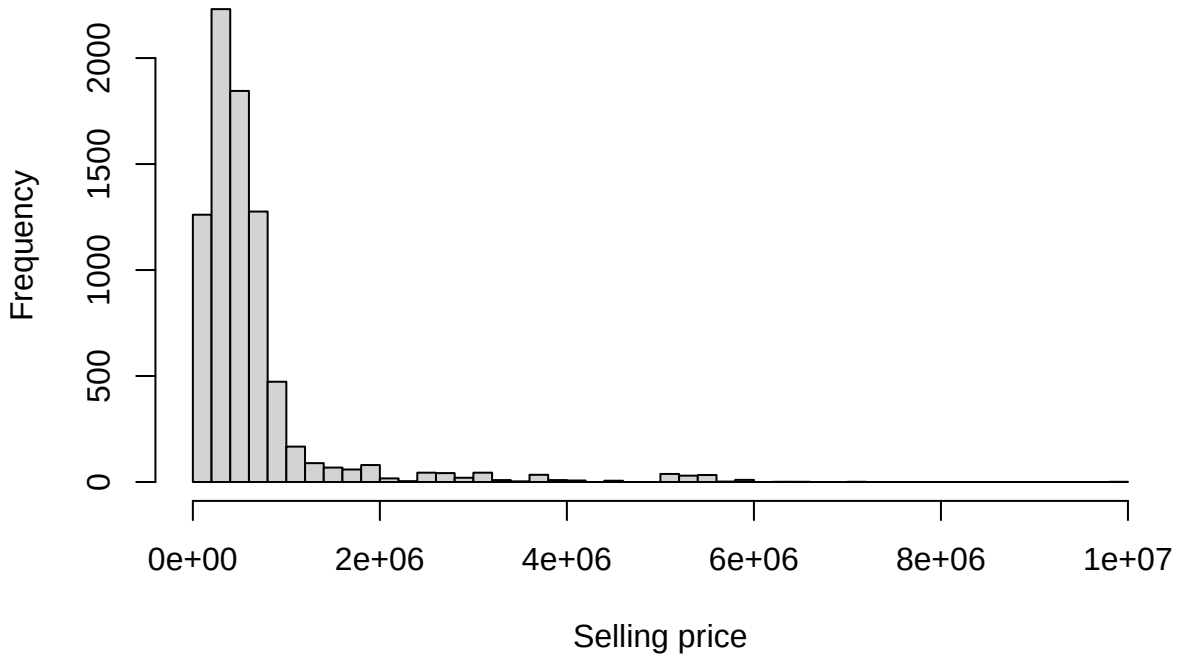


Figure 8: Histogram of selling price values in the dataset.

We can see from this histogram that the price distribution is heavily right skewed. In addition, the scale of selling price is large. In order to decrease the scale and transform the distribution towards normal distribution, we performed a logarithmic transformation on selling price and checked the distribution using histogram again. The following regression methods on price are conducted on log selling price.

```
car1 <- car
car1$LogPrice=log(car1$selling_price)
car1 <- car1[,-2]
head(car1)
```

```
##   year km_driven   fuel seller_type transmission        owner mileage engine
## 1 2014    120000 Diesel  Individual       Manual Second Owner   21.14   1498
## 2 2006    140000 Petrol  Individual       Manual  Third Owner   17.70   1497
## 3 2010    127000 Diesel  Individual       Manual  First Owner   23.00   1396
```

```
## 4 2007    120000 Petrol  Individual       Manual  First Owner   16.10   1298
## 5 2017     45000 Petrol  Individual       Manual  First Owner   20.14   1197
## 6 2007    175000    LPG  Individual       Manual  First Owner   17.30   1061
##   max_power seats LogPrice
## 1    103.52     5 12.82126
## 2     78.00     5 11.97035
## 3     90.00     5 12.32386
## 4     88.20     5 11.77529
## 5     81.86     5 12.99453
## 6     57.50     5 11.47210
```

```
hist(car1$LogPrice,nclass=50,main = "",xlab = "Log Selling price")
```

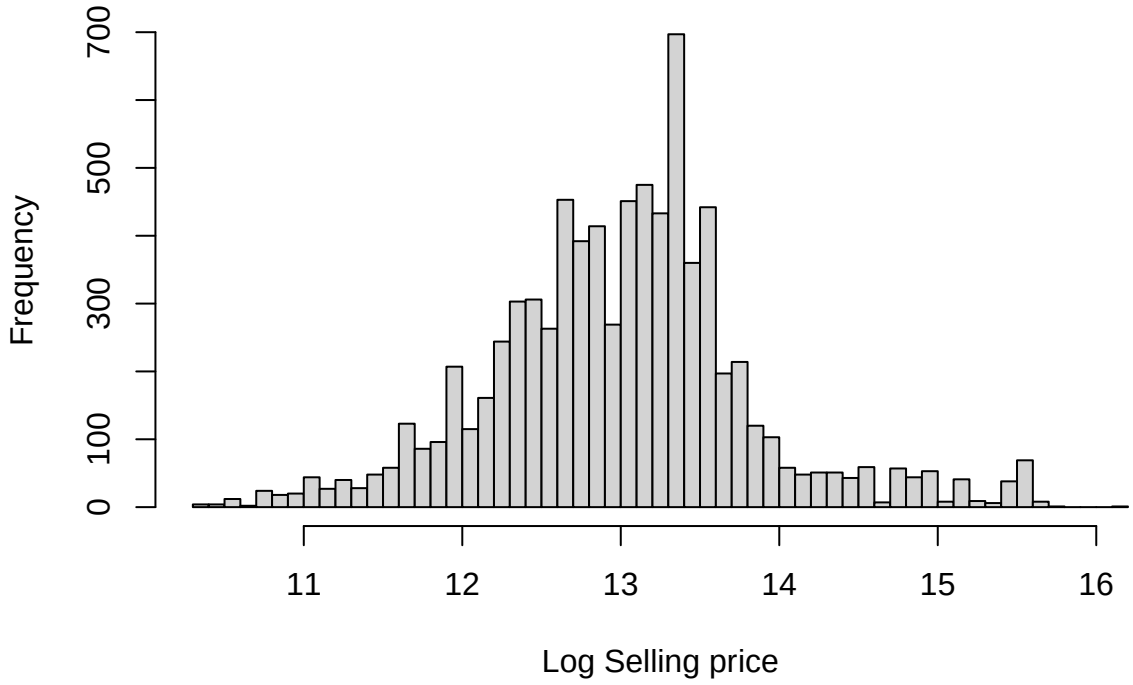


Figure 9: Histogram of the log-transformed selling price values.

```
## Warning: package 'tree' was built under R version 4.1.2
```

We first randomly sampled the data 10 times and fit a tree each time to see what variables will be used. Each sample has 70% of total units in the dataset.

```
set.seed(2022)
for(i in 1:10) {
  train <- sample(1:nrow(car1), 0.7*nrow(car1))
  tree_model <- tree(LogPrice~
                     year+km_driven+seller_type+
                     transmission+owner+mileage+engine+
```

```
                  max_power+seats, car1, subset=train)
  print(as.character(summary(tree_model)$used))
}
```

```
## [1] "year"      "max_power" "engine"
## [1] "year"      "max_power" "engine"
## [1] "year"      "max_power" "engine"
## [1] "year"      "max_power" "engine"
## [1] "year"      "max_power"
## [1] "year"      "max_power"
## [1] "year"      "max_power" "engine"
## [1] "year"      "max_power" "engine"
## [1] "year"      "max_power" "engine"
## [1] "year"      "max_power" "engine"
```

Although they all are from the same population, different samples result in different variable lists. Variables year and max_power are always in the list. Validation approach is then used to set aside 30% of data as test data, with the rest designated as training data. A regression tree is built using the training data.

```
set.seed(2022)
ind <- sample(1:nrow(car1), 0.7*nrow(car1))
train <- car1[ind,]
test <- car1[-ind,]
tree_model <- tree(LogPrice~
                  year+seller_type+
                  transmission+owner+mileage+engine+
                  max_power+seats, data=train)
summary(tree_model)
```

```
##
## Regression tree:
## tree(formula = LogPrice ~ year + seller_type + transmission +
##     owner + mileage + engine + max_power + seats, data = train)
## Variables actually used in tree construction:
## [1] "year"      "max_power" "engine"
## Number of terminal nodes:  10
## Residual mean deviance:  0.145 = 800.9 / 5523
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.64800 -0.23360  0.01302  0.00000  0.23940  1.49300
```

```
plot(tree_model)
text(tree_model)
title(main='Unpruned Tree')
```

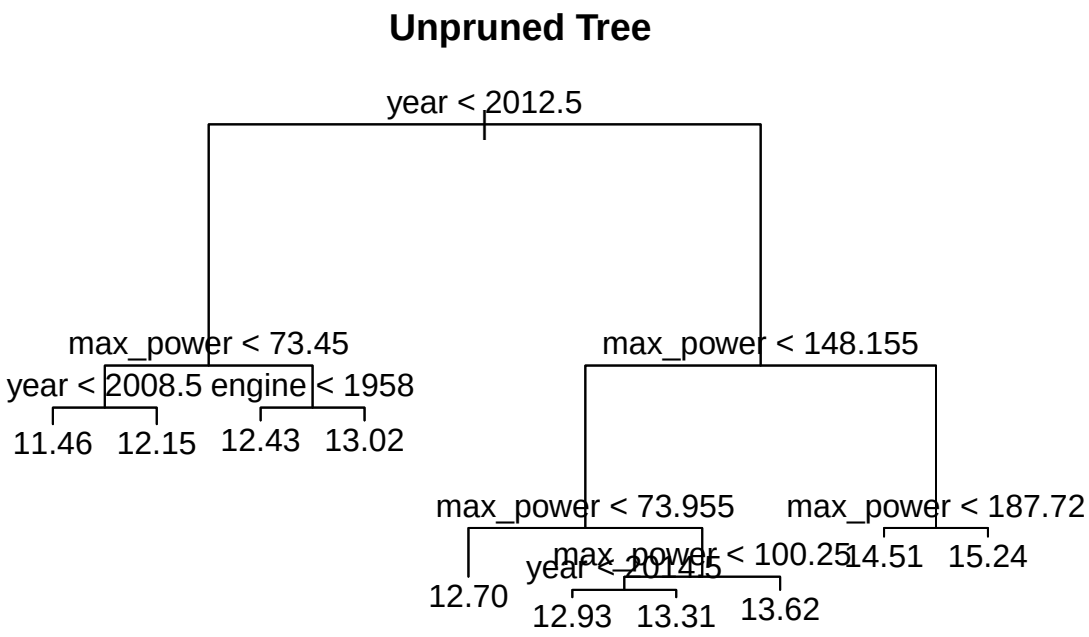


Figure 10: Plot of the unpruned regression tree model.

This unpruned tree has 10 leaves and uses the max_power, year, and engine variables. To avoid over fitting, we used cross validation and Cost of Complexity Penalty to find the best tree size:

```
set.seed(2022)
cv <- cv.tree(tree_model)
plot(cv$size,cv$dev,xlab = "Size", ylab = "SSE")
```

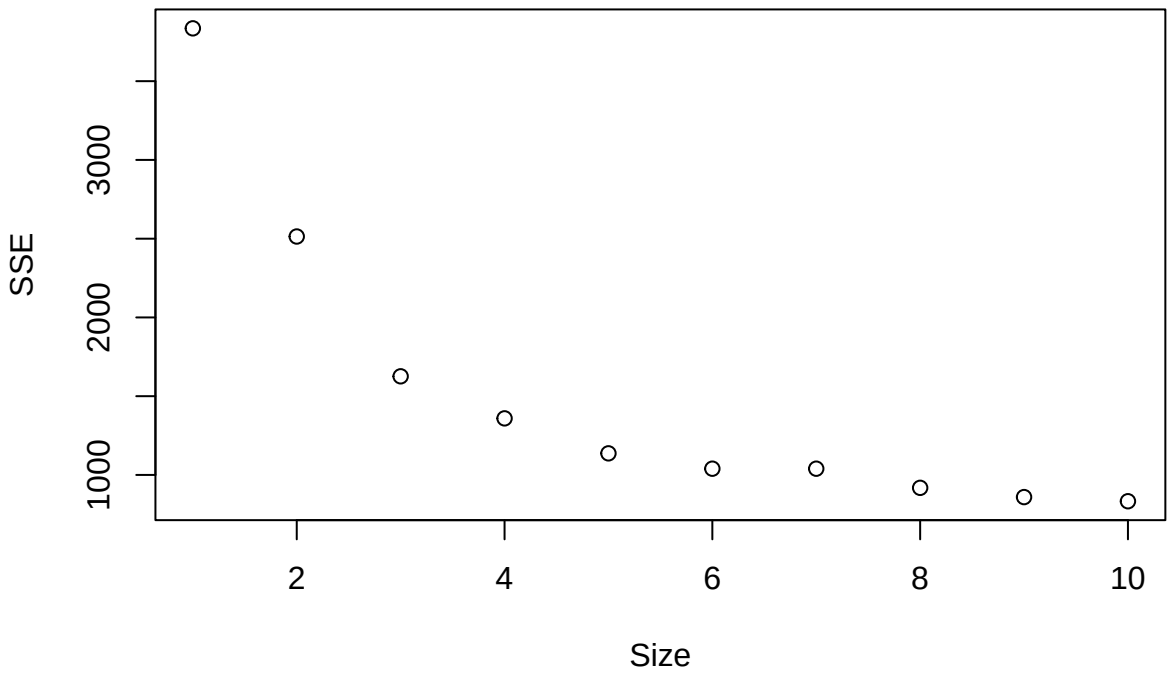


Figure 11: Plot of the sum of square error with tree size, used to determine the optimal size of the regression tree when pruning.

The sum of square error starts to plateau at a tree size of 6, minimizing the trade off between error and model complexity. Therefore, we pruned the tree to size 6:

```
tree_pruned <- prune.tree(tree_model,best=6)
plot(tree_pruned)
text(tree_pruned)
title(main='Pruned Tree')
```

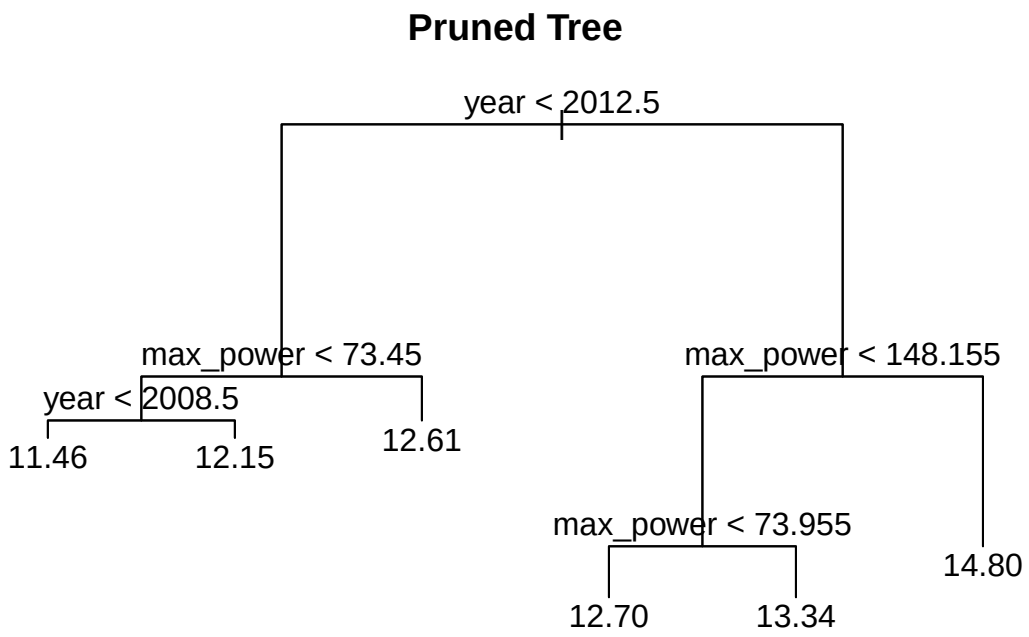


Figure 12: Plot of the pruned regression tree model.

This pruned tree removed the split on the engine variable that was in the unpruned tree. Next, we used root of mean square error as metric to evaluate this pruned tree by comparing predicted price and true price of test data.

```
test_predicted <- predict(tree_pruned,test)
sqrt(mean((exp(test_predicted)-exp(test$LogPrice))^2))
```

```
## [1] 462904.5
```

RMSE is 458200

```
plot(exp(test$LogPrice),exp(test_predicted), xlab = "Actual Price", ylab = "Predicted Price")
abline(0, 1)
```

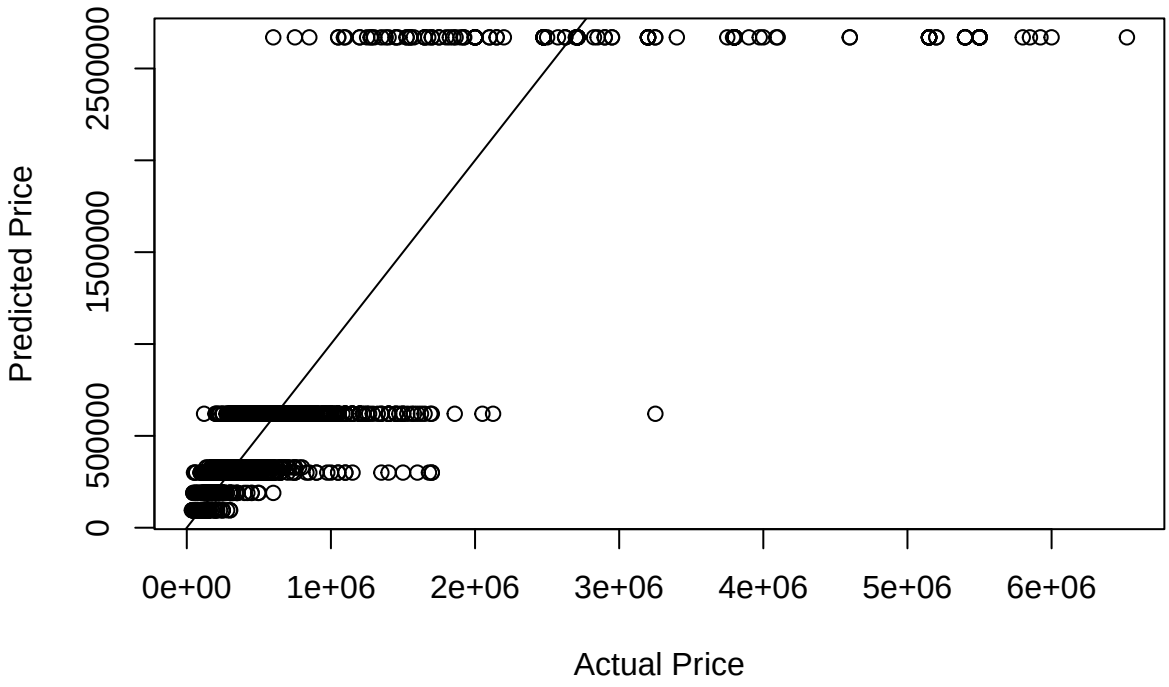


Figure 13: Predicted price versus actual price from the test data. The line represents Predicted Price = Actual Price.

Linear Regression

For comparison, we built a linear regression model and predicted price using the same set of training and test data.

```
lr_model <- glm(LogPrice~
                  year+seller_type+
                  transmission+owner+mileage+engine+
                  max_power+seats,data=train)
summary(lr_model)
```

```
##
## Call:
## glm(formula = LogPrice ~ year + seller_type + transmission +
##     owner + mileage + engine + max_power + seats, data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.32782  -0.17810   0.01948   0.20033   1.73957
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)                 -2.108e+02  2.856e+00 -73.822  < 2e-16 ***
## year                         1.103e-01  1.427e-03  77.299  < 2e-16 ***
## seller_typeIndividual       -1.280e-01  1.303e-02  -9.824  < 2e-16 ***
## seller_typeTrustmark Dealer -5.589e-02  2.603e-02  -2.147 0.031845 *
## transmissionManual          -1.868e-01  1.562e-02 -11.962  < 2e-16 ***
## ownerFourth & Above Owner   -1.139e-01  3.056e-02  -3.727 0.000196 ***
## ownerSecond Owner           -6.397e-02  1.051e-02  -6.086 1.24e-09 ***
## ownerTest Drive Car          4.008e-01  1.762e-01   2.274 0.022991 *
## ownerThird Owner            -1.003e-01  1.825e-02  -5.496 4.07e-08 ***
## mileage                      2.191e-02  1.415e-03  15.482  < 2e-16 ***
## engine                       3.374e-04  1.630e-05  20.697  < 2e-16 ***
## max_power                    1.005e-02  2.032e-04  49.448  < 2e-16 ***
## seats                        4.012e-02  6.188e-03   6.484 9.70e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.09270443)
##
##     Null deviance: 3834.10  on 5532  degrees of freedom
## Residual deviance:  511.73  on 5520  degrees of freedom
## AIC: 2557.6
##
## Number of Fisher Scoring iterations: 2
```

Different from the tree model, the linear regression uses all variables except fuel type and Km_driven to model the price. Fuel type is removed from the analysis because of multicollinearity issues. Km_driven was removed due to the P-value insignificance.

```
lr_price_pred <- predict(lr_model,test)
sqrt(mean((exp(lr_price_pred)-exp(test$LogPrice))^2))
```

```
## [1] 341375.3
```

```
plot(exp(test$LogPrice),exp(lr_price_pred), xlab = "Actual Price", ylab = "Predicted Price")
abline(0, 1)
```

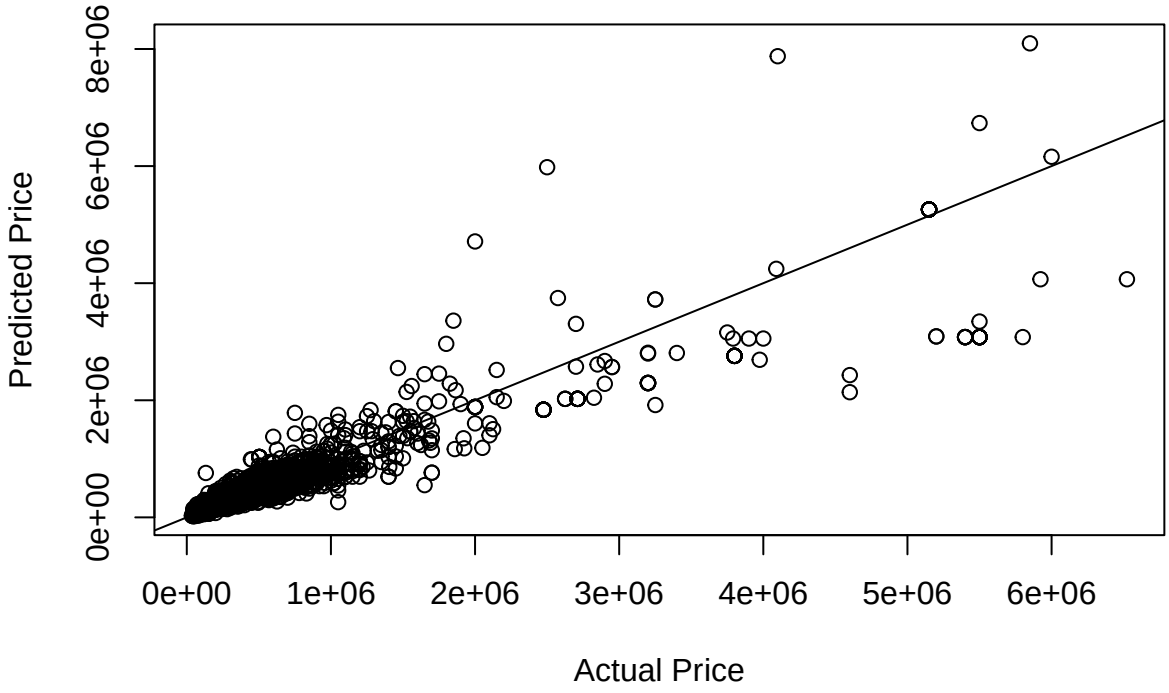


Figure 14: Predicted price versus actual price from the test data for the linear regression model. The line represents Predicted Price = Actual Price.

Linear regression has an RMSE of 341375.3, much less than that of the regression tree. Also the trend follows line Predicted Price = Actual Price pretty well.

Comparison using cross validation

The previous two models are built based on training data that split the dataset only one time. To avoid the randomness brought in by the random split of train and test data and to compare two models' application on this data fairly, the full data set will be used for 10 folds cross validation to check the average RMSE.

Ten folds cross validation on regression tree

```
set.seed(2022)
tree_model_full <- tree(LogPrice~
                  year+seller_type+
                  transmission+owner+mileage+engine+
                  max_power+seats,data=car1)
tree_pruned <- prune.tree(tree_model_full,best=6)
cv_full <- cv.tree(tree_pruned)
plot(cv_full$size,cv_full$dev,ylab = "SSE",xlab = "Tree Size")
```

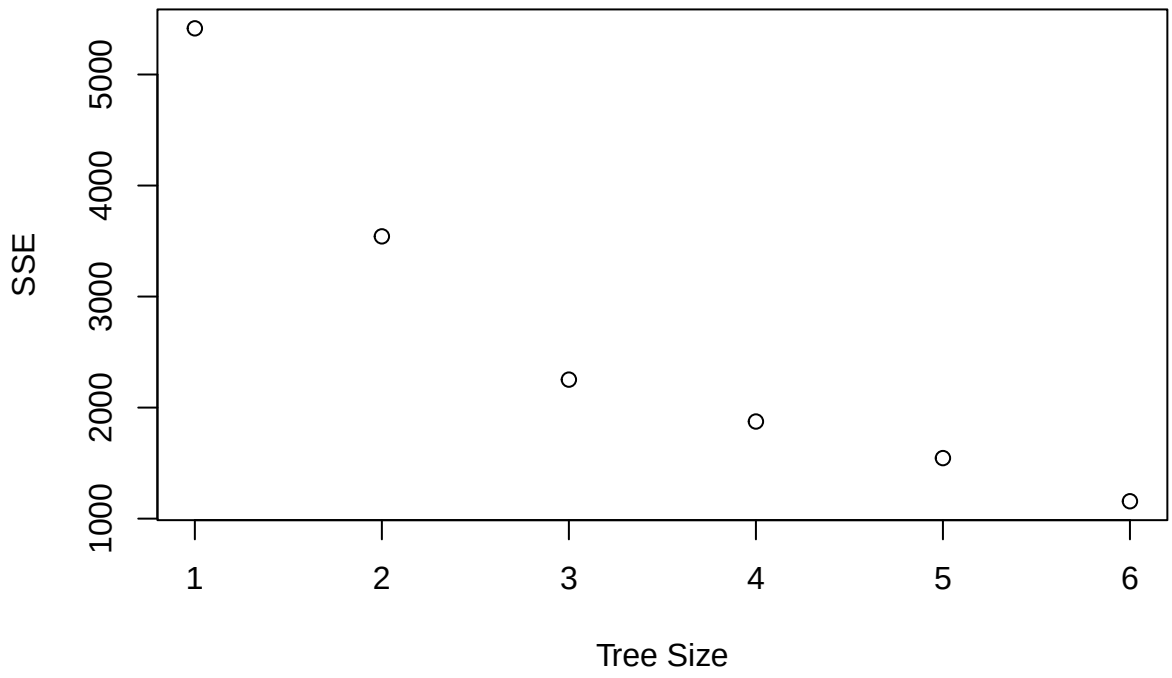
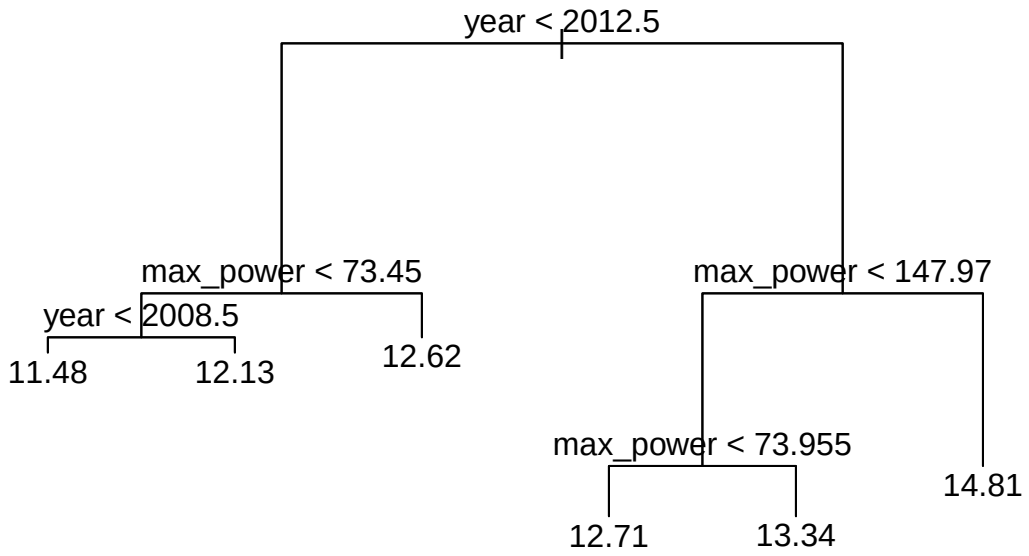


Figure 15: Plot of sum of square error for the pruned regression tree using ten folds cross validation

```
plot(tree_pruned)
text(tree_pruned)
```




```
sqrt(cv_full$dev[cv_full$size==6]/nrow(car1))
```

```
## [1] 0.3825185
```

Figure 16: Plot of the pruned regression tree using ten folds cross validation

RMSE of 10 folds cross validation for tree with 6 leaves is 0.38. This RMSE is calculated based on the log transformation of selling price.

Ten folds cross validation on linear regression

```
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
## The following object is masked from 'package:survival':
##
##     aml
```

```
lr_model <- glm(LogPrice~
                  year+seller_type+
                  transmission+owner+mileage+engine+
                  max_power+seats,data=car1)
set.seed(2022)
cv_lr <- cv.glm(car1,lr_model,K=10)
sqrt(cv_lr$delta[1])
```

```
## [1] 0.3022293
```

RMSE of 10 folds cross validation for the linear regression is 0.3022293, less than that of linear regression with the initial training and test data split. Same with the case of the above regression tree, this RMSE is also calculated based on the log transformation of selling price.

By comparing RMSE, the linear regression performs better on this data. Next, a very important part of linear regression is assumptions validation.

Multi-collinearity check for the Linear Regression Model:

```
library(mctest)
imcdiag(lr_model,method='VIF')
```

```
##
## Call:
## imcdiag(mod = lr_model, method = "VIF")
##
##
##  VIF Multicollinearity Diagnostics
##
##                                VIF detection
## year                        1.8246         0
## seller_typeIndividual       1.4369         0
## seller_typeTrustmark Dealer 1.2108         0
## transmissionManual          1.6486         0
## ownerFourth & Above Owner   1.0993         0
## ownerSecond Owner           1.2650         0
## ownerTest Drive Car         1.0063         0
## ownerThird Owner            1.1954         0
## mileage                     1.9721         0
## engine                      4.1348         0
## max_power                   3.2545         0
## seats                       2.1710         0
##
## NOTE:  VIF Method Failed to detect multicollinearity
##
##
## 0 --> COLLINEARITY is not detected by the test
```

```
##
## ================================
```

There is no multicollinearity present in this model because the fuel variable was removed due to multicollinearity between the Diesel and Petrol fuel types.

Linearity Assumption:

```
plot(fitted(lr_model), residuals(lr_model),xlab="Fitted Values", ylab="Residuals")
abline(h=0,lty=1)
title("Residual vs Fitted")
```

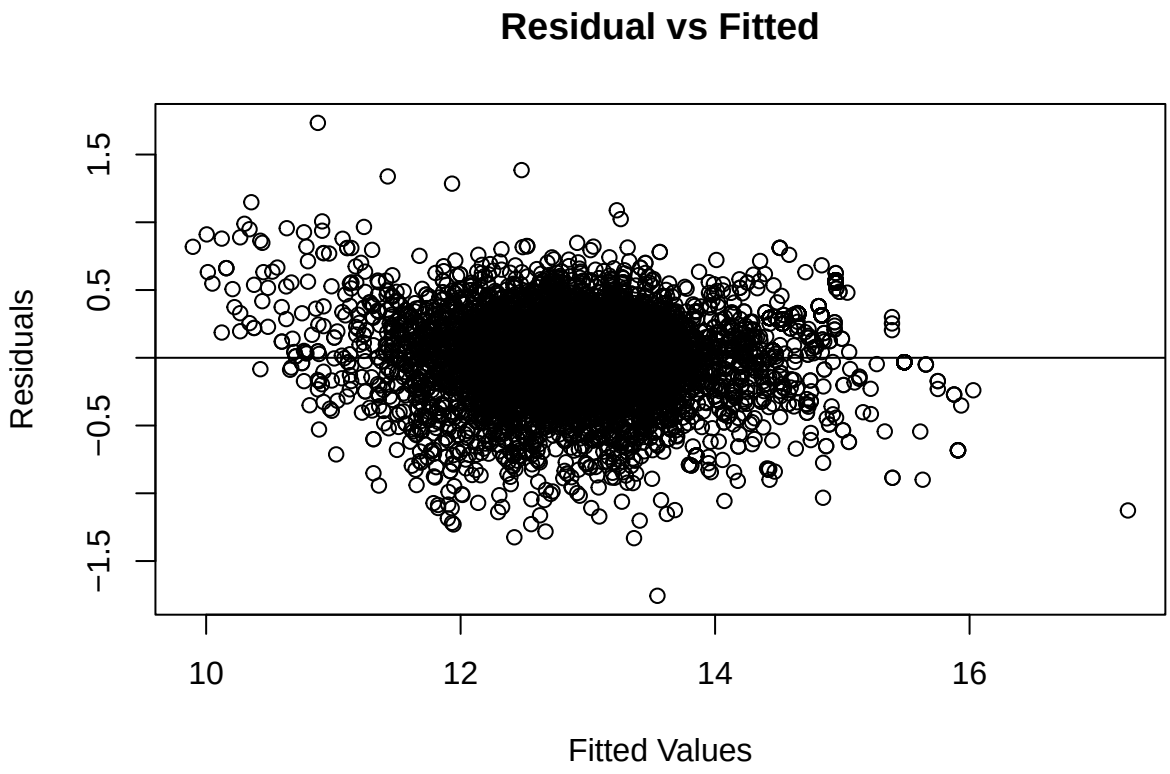


Figure 17: Residual vs. Fitted plot for the final linear regression model

From the above plot, we do not observe any clear patterns in the residuals and we can conclude that the linearity assumption is met.

Independence Assumption:

Independence Assumption is not met because of the presence of the 'year' variable. Since the response variable 'selling_price' is dependent year, a time-series variable, it is implied that the measurements may have issues with independence.

Equal Variance Assumption:

```
library(lmtest)
```

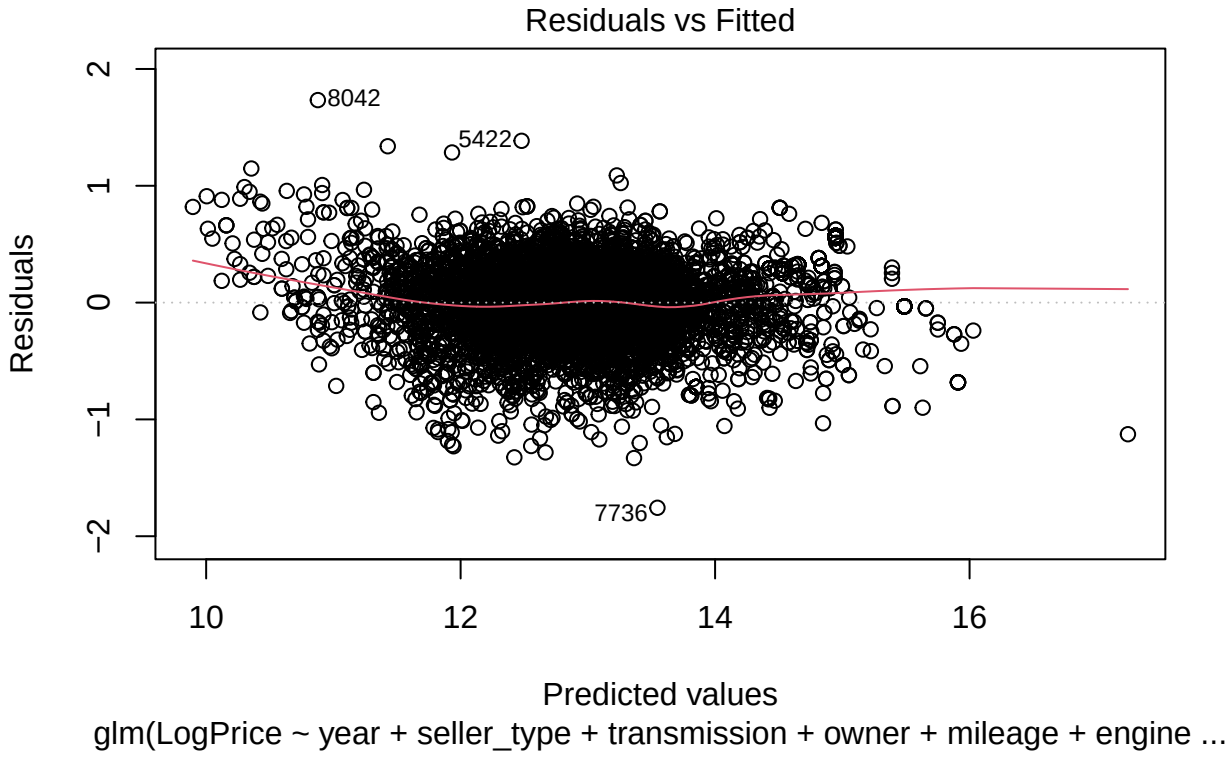
```
## Warning: package 'lmtest' was built under R version 4.1.2
```

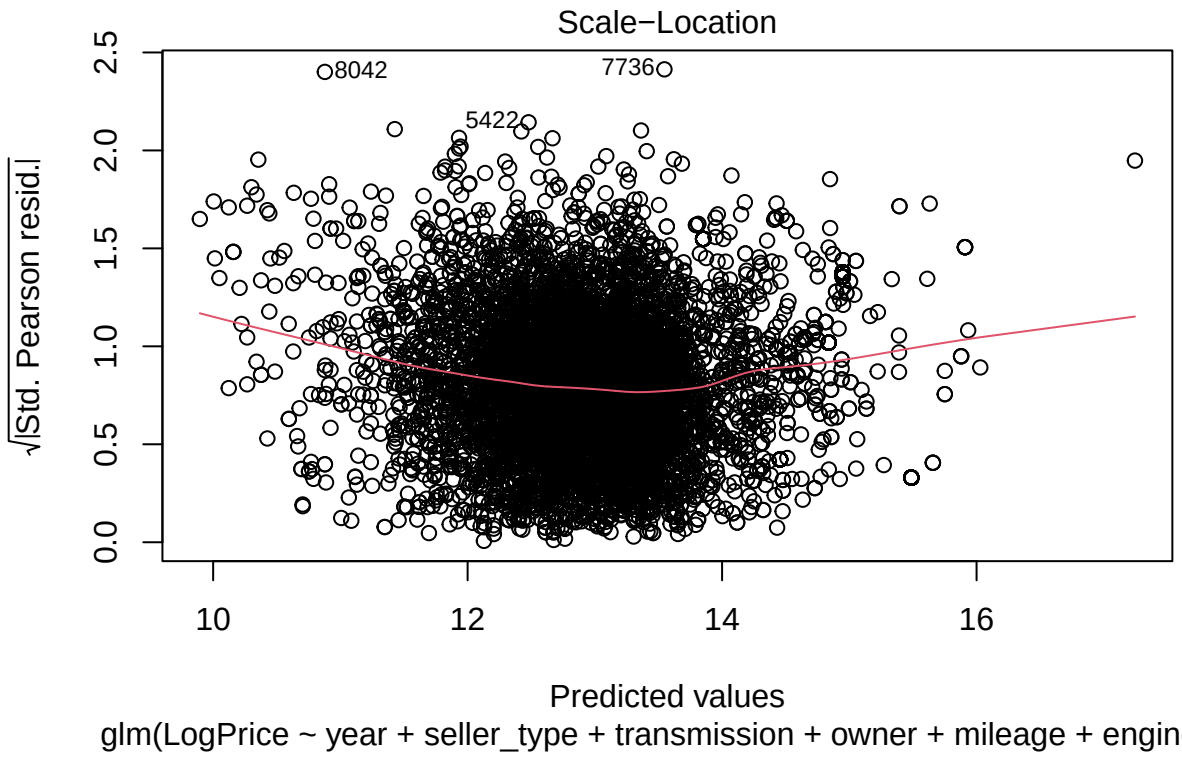
```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
plot(lr_model, which=c(1,3))
```






```
H0 <- "Heteroscedasticity is not present (homoscedasticity)"
Ha <- "Heteroscedasticity is present"

bptest(lr_model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lr_model
## BP = 606.56, df = 12, p-value < 2.2e-16
```

Figure 18: Residuals vs Fitted and Scale-Location graphs for checking homoscedasticity assumption.

Since the p-value from the Breusch-Pagan test is less than 0.05, we reject the null hypothesis and determine that heteroscedasticity is present in the model.

Normality Assumption

```
plot(lr_model, which=2)
```

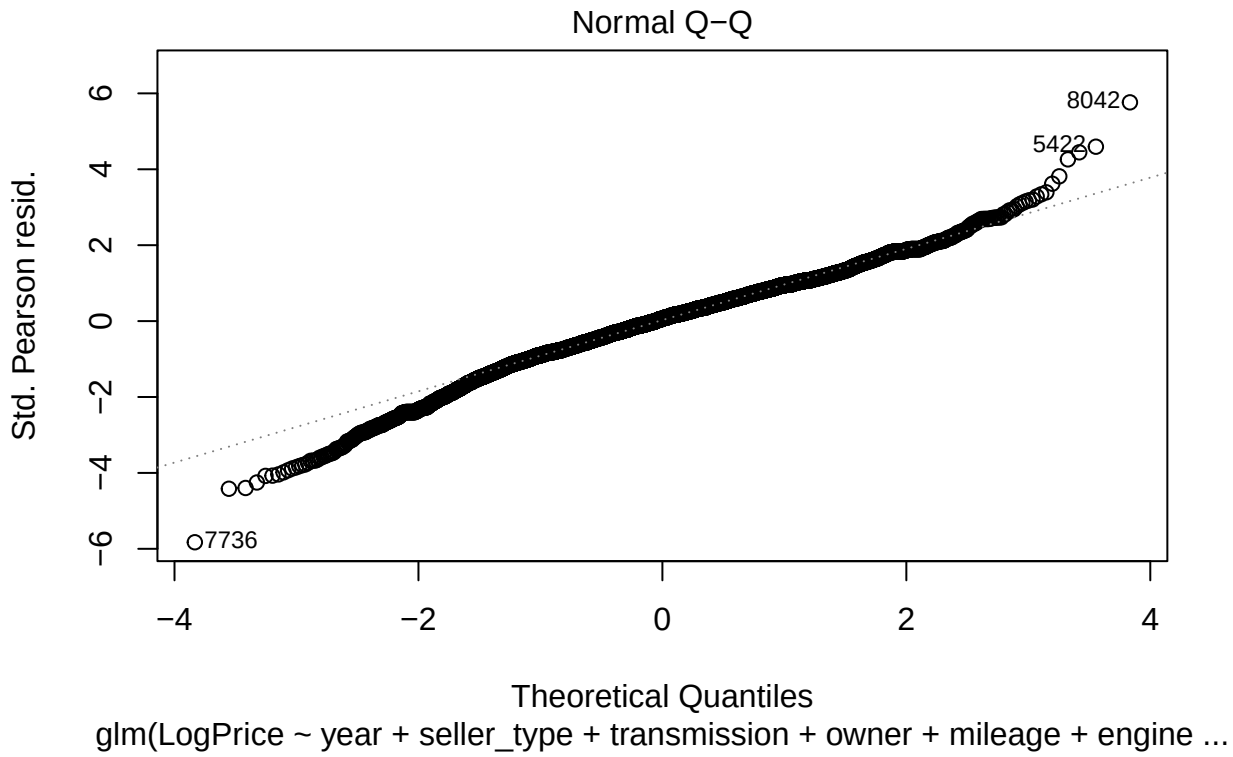


Figure 19: Q-Q normal plot for the final linear regression model

The deviation in the tails from the Q-Q line above signifies that there is non-normality in the residuals of our model.

Classification tree

In order to determine the relationship between the 'fuel' variable and other variables in this dataset, a classification tree is used. First, we split the data into two parts: training and testing sets. The training set contains 75% of the total data and is constructed using simple random sampling, and the testing set has the other 25% of the data. A classification tree will then be constructed using the training part of the dataset and predictions will be made with the constructed model and the test data.

```
set.seed (10)
train=sample(1:nrow(car),3/4*nrow(car))
test=car[-train,]
tree.fuel<-tree(factor(fuel)~. , car, subset=train)
summary(tree.fuel)
```

```
##
## Classification tree:
## tree(formula = factor(fuel) ~ ., data = car, subset = train)
## Variables actually used in tree construction:
## [1] "engine"        "mileage"       "selling_price"
## Number of terminal nodes:  10
```

```
## Residual mean deviance:  0.3278 = 1940 / 5918
## Misclassification error rate: 0.03745 = 222 / 5928
```

The variables used in the tree's construction are "engine", "mileage", and "selling_price".

Next, we can use the unpruned tree to predict which fuel type a car has based on its mileage, engine type, and selling price.

```
plot(tree.fuel)
text(tree.fuel, pretty=0)
title(main='Unpruned Tree')
```

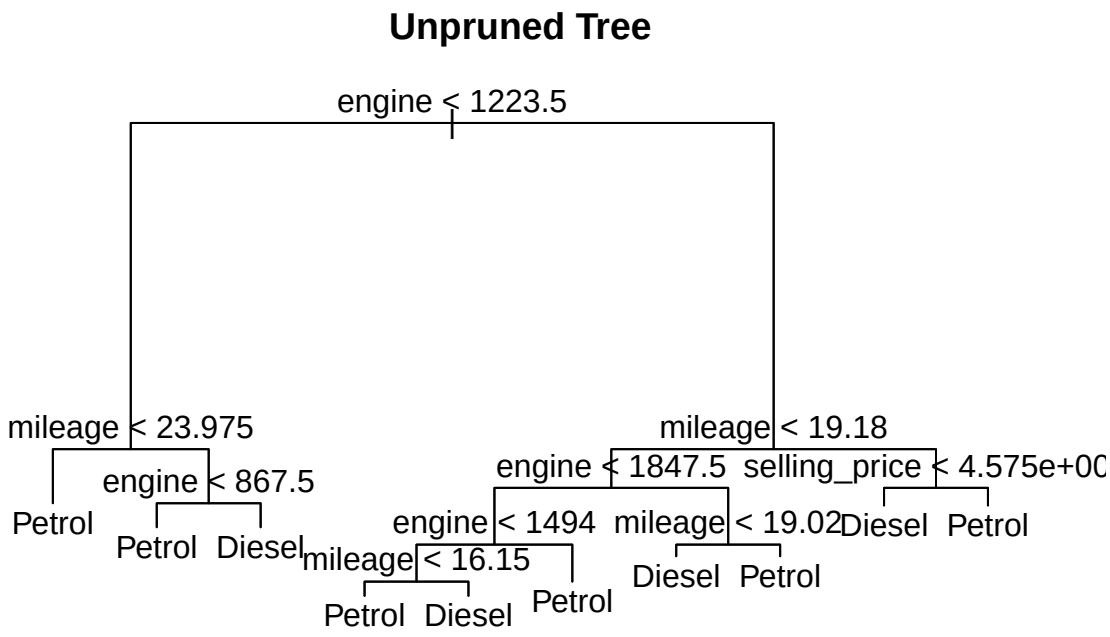


Figure 20: Unpruned classification tree for predicting fuel type

```
tree.pred<-predict(tree.fuel,test,type = "class")
table(tree.pred,test$fuel)
```

```
##
## tree.pred  CNG Diesel  LPG Petrol
##    CNG       0      0    0      0
##    Diesel    7   1031    2     33
##    LPG       0      0    0      0
##    Petrol    9     24    5    866
```

```
mean(tree.pred!=test$fuel)
```

```
## [1] 0.04046535
```

From our prediction, we obtained the above confusion matrix and a misclassification rate of 0.04046535.

Next we want to prune the tree using cross validation to select the ideal number of nodes.

```
set.seed(10)
cv.fuel<-cv.tree(tree.fuel, FUN = prune.misclass, K=10)
plot(cv.fuel$size, cv.fuel$dev,type="b", xlab = "Tree Size",ylab = "Cross-Validation Error")
```

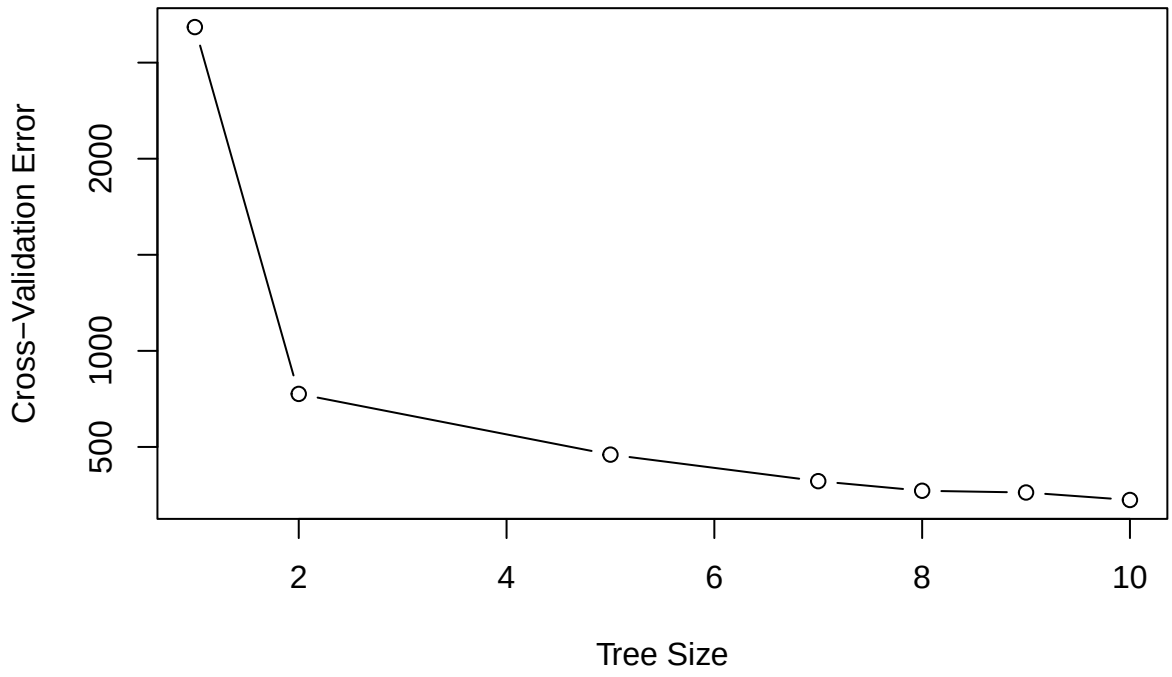


Figure 21: Q-Q normal plot for the final linear regression model

The best number of nodes in this model is 7, as it reduces the cross validation error the most while minimizing the number of terminal nodes to prevent overfitting. The new pruned tree is plotted and we can do the prediction with the new tree.

```
prune.fuel=prune.tree(tree.fuel,best=7)
plot(prune.fuel)
text(prune.fuel,pretty=0)
title(main='Pruned Tree')
```

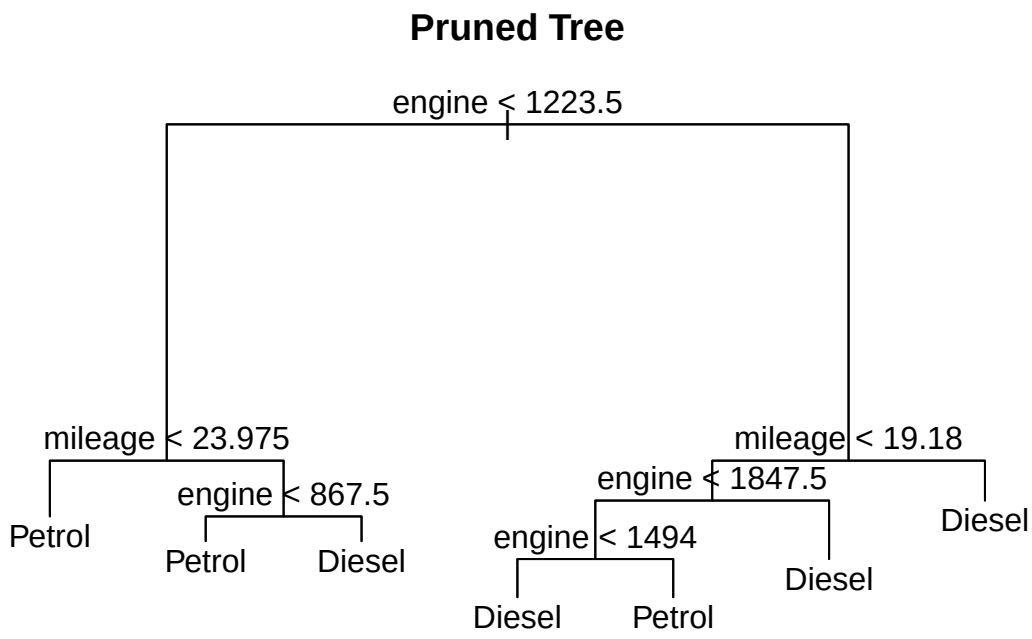


Figure 22: Pruned classification tree for predicting fuel type

```
fuel.prune=predict(prune.fuel,test,type="class")
table(fuel.prune,test$fuel)
```

```
##
## fuel.prune  CNG Diesel  LPG Petrol
##     CNG       0      0    0      0
##     Diesel    7   1036    2     81
##     LPG       0      0    0      0
##     Petrol    9     19    5    818
```

```
mean(fuel.prune!=test$fuel)
```

```
## [1] 0.06221548
```

The misclassification rate with the pruned tree has increased to 0.06221548 from 0.04046535, but this pruned tree we obtained through cross validation is simpler and more readable, so we can keep our pruned tree model in order to predict fuel type. Neither the pruned or unpruned model predicted any values for the CNG or LPG fuel types that are present in this dataset, likely because there are so few observations with those fuel types in this dataset. Therefore this model should only be used to predict whether a car uses diesel or petrol.

Transmission type modelling and predictions

In this section, we are attempting to predict the type of transmission a used car has. We are using logistic regression, linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA), and comparing the effectiveness of the three models.

Logistic Regression modeling for transmission type

For our logistic model, we selected the transmission, year, selling_price, max_power, engine, and fuel variables first. Multicollinearity was detected for the fuel variable, so it was removed from the analysis which improved our models and gave us a lower misclassification_rate.

```
# 1 - Auto
# 2 - Manual

transmission = car2[c("transmission", "year", "selling_price", "max_power", "engine")]

set.seed(10)
idx3=sampling:::strata(transmission, stratanames=c("transmission"), size=c(640, 160), method="srswor")

train3=getdata(transmission, idx3)

idx3=unlist(idx3)
test3=transmission[-idx3,]
```

```
glmModel<-glm(transmission~year+selling_price+max_power+engine, family=binomial, data=train3)
summary(glmModel)
```

```
##
## Call:
## glm(formula = transmission ~ year + selling_price + max_power +
##     engine, family = binomial, data = train3)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.54541   0.08691   0.27315   0.48282   2.43774
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.850e+02  1.156e+02   2.466   0.0137 *
## year          -1.391e-01  5.725e-02  -2.429   0.0151 *
## selling_price -8.811e-07  4.122e-07  -2.138   0.0325 *
## max_power     -6.274e-02  9.283e-03  -6.758 1.40e-11 ***
## engine         2.517e-03  6.014e-04   4.184 2.86e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 800.64  on 799  degrees of freedom
## Residual deviance: 448.02  on 795  degrees of freedom
## AIC: 458.02
```

```
##
## Number of Fisher Scoring iterations: 6
```

```
probPredict<-predict(glmModel, test3, type="response")

Predict<-rep("Neg",dim(test3)[1])
Predict[probPredict>=0.5]="Pos"

Actual<-test3$transmission

table(Predict, Actual)
```

```
##        Actual
## Predict Automatic Manual
##     Neg       459    211
##     Pos       422   6011
```

```
misclassification_rate=(437+152)/8127
misclassification_rate
```

```
## [1] 0.07247447
```

The misclassification rate for Logistic Regression modeling is 0.07247447

Linear Discriminant analysis for transmission type

```
ldaModel<-lda(transmission~year+selling_price+max_power+engine, data=train3)
summary(ldaModel)
```

```
##         Length Class  Mode
## prior   2      -none- numeric
## counts  2      -none- numeric
## means   8      -none- numeric
## scaling 4      -none- numeric
## lev     2      -none- character
## svd     1      -none- numeric
## N       1      -none- numeric
## call    3      -none- call
## terms   3      terms  call
## xlevels 0      -none- list
```

```
class.pred<-predict(ldaModel, test3)
table(class.pred$class, test3$transmission)
```

```
##
##             Automatic Manual
##   Automatic       438    101
##   Manual          443   6121
```

```
misclassification_rate=(458+66)/8127
misclassification_rate
```

```
## [1] 0.06447644
```

The misclassification rate for Linear Discriminant analysis is 0.06447644.

Quadratic Discriminant analysis for transmission type

```
qdaModel <- qda(transmission~year+selling_price+max_power+engine, data=train3)
summary(qdaModel)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means    8     -none- numeric
## scaling 32     -none- numeric
## ldet     2     -none- numeric
## lev      2     -none- character
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
predict <- predict(qdaModel, test3)
table(predict$class, test3$transmission)
```

```
##
##             Automatic Manual
##   Automatic       468    212
##   Manual          413   6010
```

```
misclassification_rate=(425+190)/8127
misclassification_rate
```

```
## [1] 0.07567368
```

The misclassification rate for Quadratic Discriminant analysis is 0.07567368.

Contingency Table:

We use contingency table to establish a relationship between the variables and to check whether they are independent of each other or not.

Here, we consider the two variables, 'transmission' and 'seller_type' for the contingency and check for the test of independence.

```
tab1<-table(car$transmission, car$seller_type)
tab1
```

```
##
##             Dealer Individual Trustmark Dealer
##   Automatic    458        484               99
##   Manual       649       6078              137
```

Probability Table:

```
prop.table(tab1)
```

```
##
##                 Dealer Individual Trustmark Dealer
##   Automatic 0.05793801 0.06122707       0.01252372
##   Manual    0.08209994 0.76888046       0.01733080
```

Computing the margin table:

```
margin.table(tab1,2)
```

```
##
##           Dealer       Individual Trustmark Dealer
##             1107             6562              236
```

Probability table, conditioned with the row variable

```
prop.table(tab1, margin = 1)
```

```
##
##                 Dealer Individual Trustmark Dealer
##   Automatic 0.43996158 0.46493756       0.09510086
##   Manual    0.09455128 0.88548951       0.01995921
```

Pearson's Chi-Square Test for 2X2 table:

```
chisq.test(tab1)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab1
## X-squared = 1133.6, df = 2, p-value < 2.2e-16
```

Based on the chi-square value, we can see that there is high interdependency between the transmission and seller type variables.

For nominal-ordinal data: I have considered the variables 'transmission' and 'seats' to check for the test of independence between them.

```
tr_se_tab<-table(car$transmission, car$seats)
tr_se_tab<-tr_se_tab[,c(1,2,3,4,5,6,7,8,9)]
tr_se_tab
```

```
##
##                2    4    5    6    7    8    9   10   14
##   Automatic    0   16  902    0  106   17    0    0    0
##   Manual       2  117 5351   62 1014  218   80   19    1
```

Chi-Squared Test:

```
chisq.test(table(car$transmission, car$seats))
```

```
## Warning in chisq.test(table(car$transmission, car$seats)): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(car$transmission, car$seats)
## X-squared = 54.334, df = 8, p-value = 5.948e-09
```

Becuase there are lower cell values presemt, the chi-squared test is not fully accurate. We can do the Fishers Exact Test for the table to correct for this.

Fisher's Exact Test:

```
fisher.test(table(car$transmission, car$seats), alternative = "two.sided", simulate.p.value = TRUE)
```

```
##
##  Fisher's Exact Test for Count Data with simulated p-value (based on
##  2000 replicates)
##
## data:  table(car$transmission, car$seats)
## p-value = 0.0004998
## alternative hypothesis: two.sided
```

We can reject the null hypothesis that these variables are independent because the Fisher's test gave us a p-value less than 0.05, so we can conclude that there is a significant relationship between the transmission and seats variables.

Conclusion

The major takeaways from our analysis are:

- In order to predict selling price of the used cars in the dataset, a linear regression is the most effective method. We used 10-fold cross validation to construct the linear model, which gave us a better RMSE than the standard validation approach.
- When predicting fuel type for the used cars using a classification tree, the best variables to use for this prediction after pruning the classification tree are the engine and mileage variables.
- To predict the transmission type, the method with the lowest misclassification rate is logistic regression.
- From the contingency analysis, we can conclude that there is a high correlation between the transmission and seller type variables. We also found a high correlation between the transmission and seats variables through the Fisher's Test.