Predicting Fuel Efficiency - Regression Modelling

Dahye Kim

1/15/2021

This program predicts `fuel efficiency` of a car (in km/L) based on its characteristics. Since Australia is one of the largest automobile markets in the world, correctly predicting the fuel efficiency is necessary to control emission rate to the environment.

Following datasets are used for the predictive analysis:

- **RegressionTrain.csv**: contains observations and predictors obtained from many retailers for car models available for sale from 2017 to 2020. The **target variable** is the `fuel efficiency` in km/L. The higher this value, the better the fuel efficiency of the car.

```
# read in the file
train <- read.csv('RegressionTrain.csv')
test <- read.csv('RegressionTest.csv')
```

Part 1. Building a Model

Explore the Data

Before building a model, I first inspected the characteristics of each attribute in the data set. I first identified the data types of each attribute, and inspected its statistical summary and its correlation to other attribute. The data set included six numerical attribute, including the target variable `Comb.FE`, and four categorical variables.

```
# categorise the columns based on the attribute types
numeric.col <- c('Model.Year','Eng.Displacement','No.Gears','No.Cylinders',
                 'Max.Ethanol','Comb.FE')
factor.col <- names(train)[!names(train)%in%numeric.col]
```

Inspecting Predictors with Continuous Variables

```
# inspecting the data type of each attribute
sapply(train, class)
```

```
##              Model.Year        Eng.Displacement            No.Cylinders
##               "integer"               "numeric"               "integer"
##              Aspiration                No.Gears Lockup.Torque.Converter
##             "character"               "integer"             "character"
##               Drive.Sys             Max.Ethanol               Fuel.Type
##             "character"               "integer"             "character"
##                  Comb.FE
##               "numeric"
```

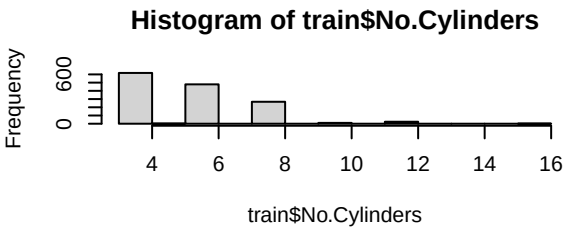
```
# categorise the attributes based on data type for easing the visualisation process
numeric.col <- c('Model.Year','Eng.Displacement','No.Gears','No.Cylinders',
```

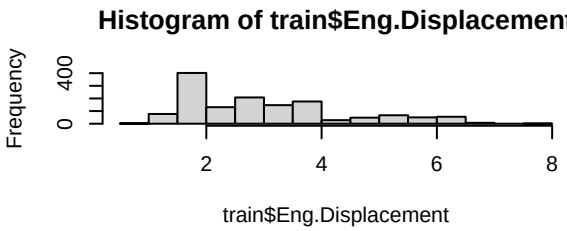
```
                'Max.Ethanol','Comb.FE')
factor.col <- names(train)[!names(train)%in%numeric.col]
```

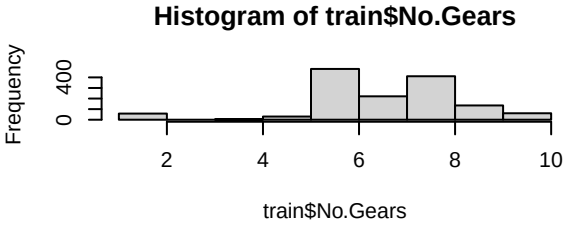
```
# statistical summary of predictors with continuous variables
summary(train[numeric.col])
```

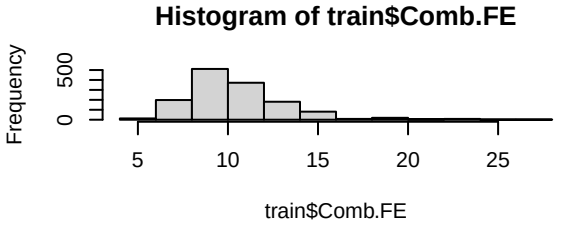
```
##    Model.Year   Eng.Displacement    No.Gears       No.Cylinders
##  Min.   :2017   Min.   :0.900    Min.   : 1.000   Min.   : 3.000
##  1st Qu.:2017   1st Qu.:2.000    1st Qu.: 6.000   1st Qu.: 4.000
##  Median :2018   Median :3.000    Median : 7.000   Median : 6.000
##  Mean   :2018   Mean   :3.144    Mean   : 6.974   Mean   : 5.622
##  3rd Qu.:2019   3rd Qu.:3.625    3rd Qu.: 8.000   3rd Qu.: 6.000
##  Max.   :2020   Max.   :8.000    Max.   :10.000   Max.   :16.000
##   Max.Ethanol       Comb.FE
##  Min.   :10.00   Min.   : 4.974
##  1st Qu.:10.00   1st Qu.: 8.591
##  Median :10.00   Median : 9.947
##  Mean   :15.11   Mean   :10.447
##  3rd Qu.:15.00   3rd Qu.:11.756
##  Max.   :85.00   Max.   :26.224
```

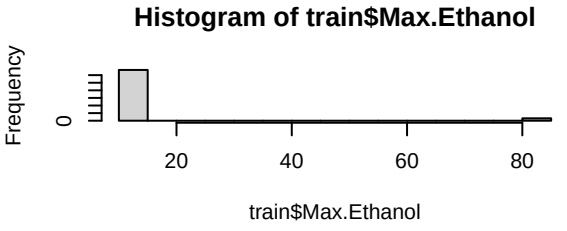
```
# inspect the distribution of numeric predictors
# check the normality of the variables
par(mfrow=c(3,2))
hist(train$No.Cylinders)
hist(train$Eng.Displacement)
hist(train$No.Gears)
hist(train$Comb.FE)
hist(train$Max.Ethanol)
hist(train$Model.Year)
```

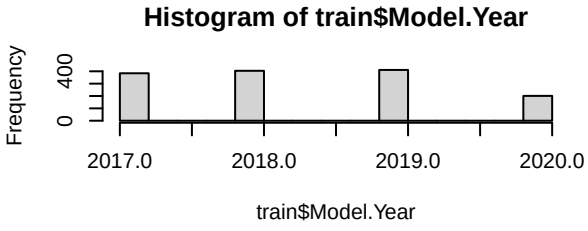













```
# inspect the correlation of all the numerical variables
round(cor(train[numeric.col]),4)
```

```
##                 Model.Year Eng.Displacement No.Gears No.Cylinders Max.Ethanol
## Model.Year          1.0000          -0.0046   0.1326      -0.0146     -0.0728
## Eng.Displacement   -0.0046           1.0000   0.2715       0.9203      0.1300
## No.Gears            0.1326           0.2715   1.0000       0.2777     -0.0662
## No.Cylinders       -0.0146           0.9203   0.2777       1.0000      0.0844
## Max.Ethanol        -0.0728           0.1300  -0.0662       0.0844      1.0000
## Comb.FE             0.0036          -0.7106  -0.4110      -0.6866     -0.0916
##                 Comb.FE
## Model.Year       0.0036
## Eng.Displacement -0.7106
## No.Gears         -0.4110
## No.Cylinders     -0.6866
## Max.Ethanol      -0.0916
## Comb.FE           1.0000
```

Inspecting Predictors with Categorical Variables

```
# statistical summary of predictors with continuous variables
summary(train[factor.col])
```

```
##   Aspiration        Lockup.Torque.Converter  Drive.Sys
##  Length:1400        Length:1400             Length:1400
##  Class :character   Class :character        Class :character
##  Mode  :character   Mode  :character        Mode  :character
##   Fuel.Type
##  Length:1400
```

```
##  Class :character
##  Mode  :character
```

After visualising the distribution of each numerical predictor variables, we can observe that neither all the numerical predictor variables and nor `Comb.FE` are perfectly normally distributed. `No.Gears` and `Comb.FE` are both highly skewed with outliers, and `Eng.Displacement` has a bimodal distribution.

According to the histogram of all the numerical variables in the data set, I realised that `Model.Year` and `Max.Ethanol` could also be regarded as categorical variables. Therfore, when visualising the relationship between `Comb.FE` and other predictor variables, I also looked at the distribution of `Comb.FE` after converting these two attributes into `factor` attribute.

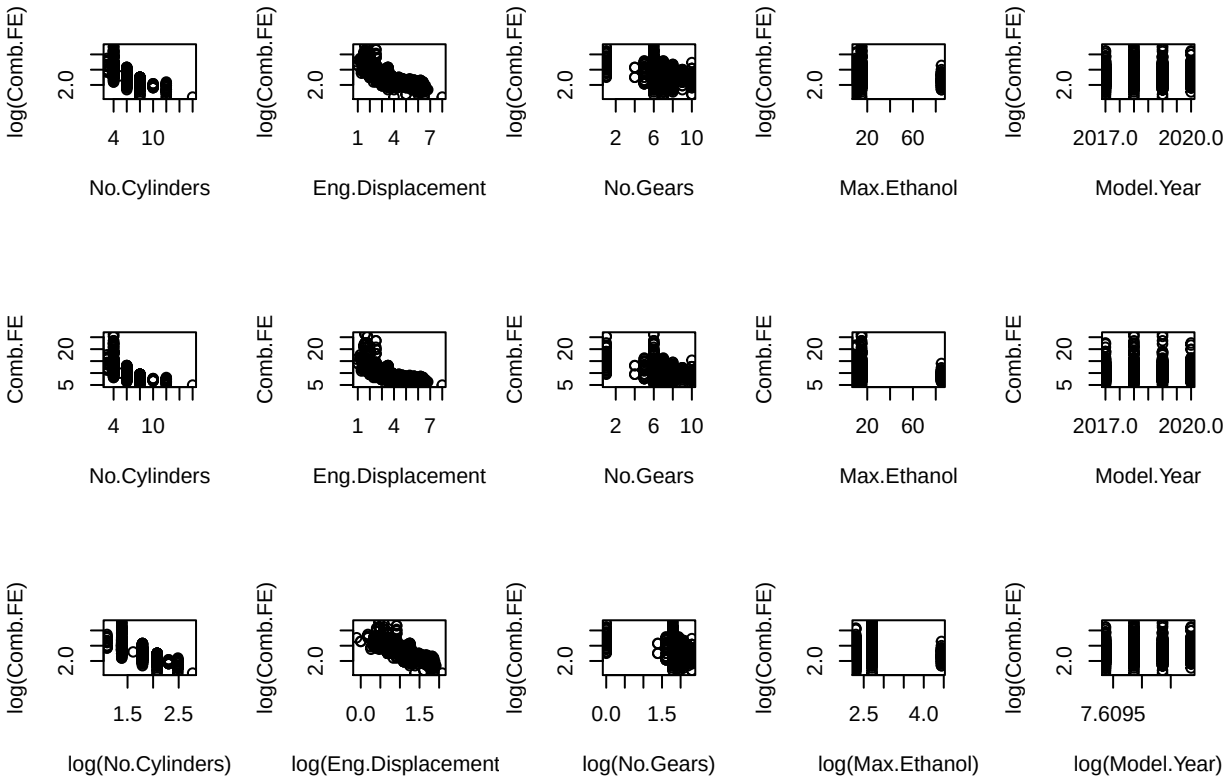
Based on the correlation matrix generated by `cor()` function, `Eng.Displacement` and `No.Cylinders` have a very high positive correlation of 0.9203. In fact, this observation could violate the assumption of multicollinearity when building a multiple regression - independent/predictor variables should be independent of another, and this high correlation between `Eng.Displacement` and `No.Cylinders` could violate the assumption. Therefore, a multiple regression model cannot be an optimal model.

Visualising the Predictors - Linearity and Normality

To observe the relationship and critical features of the data set, I visualised the attributes to the target variable `Comb.FE` using scatter plot and boxplot. For numerical predictor variables, I used scatter plot and also applied log-lin and log-log transformation to the scatter plots since `Comb.FE` and all features were non-linear.

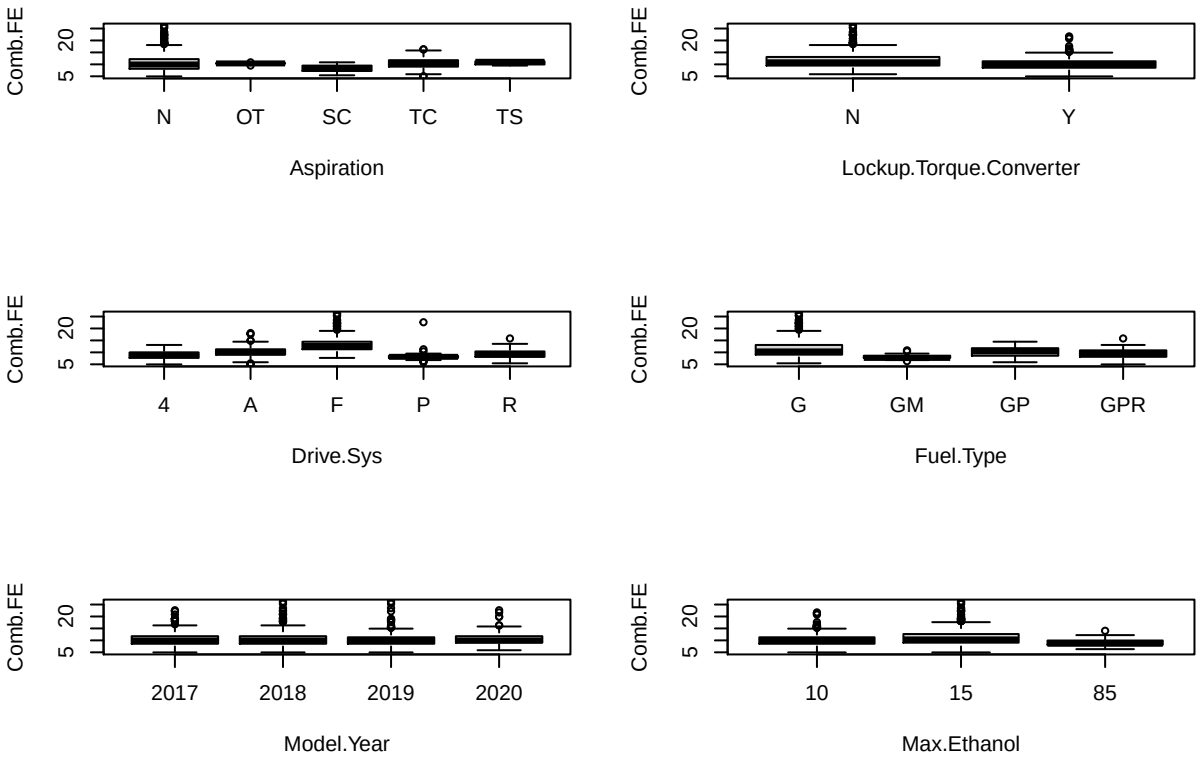
When plotting `Comb.FE~Model.Year` and `Comb.FE~Max.Ethanol` using scatter plot, it was very hard to observe distribution of `Comb.FE` since these two attribute only have less than 5 discrete observations. Therefore, I converted these two variables to `factor` and created boxplots as well.

```
par(mfrow=c(3,5))
plot(log(Comb.FE)~No.Cylinders*Eng.Displacement*No.Gears*Max.Ethanol*Model.Year, train)
# log-lin plot on all the continuous features to log(Comb.FE)

plot(Comb.FE~No.Cylinders*Eng.Displacement*No.Gears*Max.Ethanol*Model.Year, train)
# lin-lin plot on all the continuous features to Comb.FE

plot(log(Comb.FE)~log(No.Cylinders)*log(Eng.Displacement)*log(No.Gears)*log(Max.Ethanol)*
     log(Model.Year), train)
```




```
# log-log plot on all the continuous features to Comb.FE after applying logarithm to both axes
```

```
par(mfrow=c(3,2))
boxplot(Comb.FE~Aspiration,train)
boxplot(Comb.FE~Lockup.Torque.Converter,train)
boxplot(Comb.FE~Drive.Sys,train)
boxplot(Comb.FE~Fuel.Type,train)
# visualising the distribution of Comb.FE based on each level of categorical predictors

train$Model.Year<- as.factor(train$Model.Year)
# convert Model.Year into a categorical variable
boxplot(Comb.FE~Model.Year, train)

train$Max.Ethanol<- as.factor(train$Max.Ethanol)
# convert Max.Ethanol into a categorical variable
boxplot(Comb.FE~Max.Ethanol, train)
```



For the numerical predictor variables, I used scatter plot to visualise the relationship between `Comb.FE` and all the predictor variables using scatter plot. Based on all the scatter plots, `Comb.FE` doesn't seem to linearly dependent on any numerical variables. However, after applying logarithm to both axes, `Eng.Displacement` and `No.Cylinders` seemed to be linearly dependent to log(Comb.FE). The log-log plot suggest that these two variables could have exponential relationship with `Comb.FE`. Therefore, when building the model, I also considered transforming `Comb.FE` by applying logarithm function.

I also visualised the relationship between `Comb.FE` and all the categorical variables using boxplot. With every level of each categorical predictor variable, we can observe outliers in `Comb.FE`. Additionally, the distribution of `Comb.FE` changes significantly at `Drive.Sys`, `Aspiration`, and `Fuel.Type`. This could imply that these three categorical predictor variables are associated with the changes in `Comb.FE`. `Lockup.Torque.Converter` and `Max.Ethanol` also seemed to impact the distribution of `Comb.FE`, but not as significant as three aforementioned predictors.

Assumption and Models

After exploring the relationship between each predictor variables, we realised that `Eng.Displacement` and `No.Cylinders` are highly correlated to another. This could violate the multicolinearity assumption for multiple regression model, and thus multiple regression is a sub-optimal choice for our prediction.

Our data set is composed of both categorical and numerical variables, in which two of the predictors are possibly dependent on another. Additionally, the distribution of all the numerical variables are not normal, and some of them contain outliers. To overcome these problems I decided to use regression tree to predict the target variable. However, since the decision trees are non-robust, I chose random forest algorithm instead of one decision tree for predicting `Comb.FE`.

```
library('randomForest')
```

```
## randomForest 4.6-14
```

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

```r
# RMSE for evaluating the predictability of the model

rmse <- function(pred.label, truth.label){
    return(sqrt(mean((pred.label - truth.label)^2)))
}
```

For `Model.Year` and `Max.Ethanol`, these features only contained three and four discrete values respectively. Therefore, I built two seperate models - one in which forementioned predictors are treated as categorical variables, and another as numerical varaibles.

In addition, through the visualisation above, I could observe a linear relationship between `Comb.FE` and other numerical predictors after log-log transformation. Therefore, I also built two models - one in which the target variable is applied with logarithm.

```r
train$Model.Year <- as.numeric(train$Model.Year)
train$Max.Ethanol <- as.numeric(train$Max.Ethanol)
# convert Model.Year and Max.Ethanol to numerical variables for two models below
```

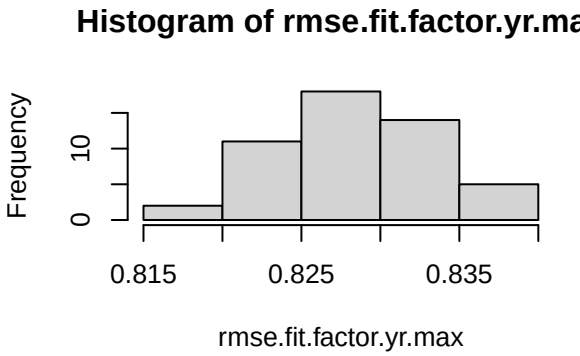
```r
rmse.log.numeric.yr.max<-c()
# record the rmse of random forest model generated from each simulation

for (i in 1:50){
    # running 50 simulations for randomForest model

    fit.log.yr.max <- randomForest(log(Comb.FE)~.*.+
                       log(Eng.Displacement)+log(No.Cylinders)+
                       log(No.Gears)+log(Max.Ethanol)+log(Model.Year)+log(Eng.Displacement*No.Gears)+
                       I(Eng.Displacement^2)+I(No.Cylinders^2)+
                       I(No.Gears^2)+I(Max.Ethanol^2)+I(Model.Year^2),
                       train, ntree = 100)
    # applying logarithm to the target
    # generate 100 decision trees and select the best out of 100

    predict.label <- exp(predict(fit.log.yr.max, train[1:9]))
    # exponential function should be applied to the final predicted value
    # create predicted value out of the training data set

    rmse.log.numeric.yr.max<-c(rmse.log.numeric.yr.max,
                               rmse(predict.label, train$Comb.FE))
    # record the rmse of each model to rmse.log.numeric.yr.max vector
}
```

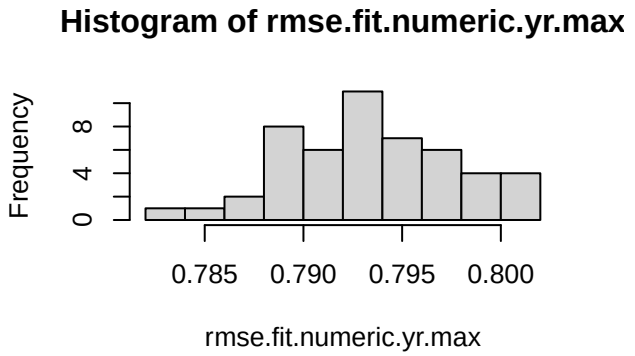
```r
rmse.fit.numeric.yr.max<-c()
for (i in 1:50){
    fit.numeric.yr.max <- randomForest(Comb.FE~.*.+
                       log(Eng.Displacement)+log(No.Cylinders)+
                       log(No.Gears)+log(Max.Ethanol)+log(Model.Year)+
                       I(Eng.Displacement^2)+I(No.Cylinders^2)+log(Eng.Displacement*No.Gears)+
                       I(No.Gears^2)+I(Max.Ethanol^2)+I(Model.Year^2),
                       train, ntree = 100)

    predict.label <- predict(fit.numeric.yr.max, train[1:9])
    rmse.fit.numeric.yr.max<-c(rmse.fit.numeric.yr.max,
                               rmse(predict.label, train$Comb.FE))}
```

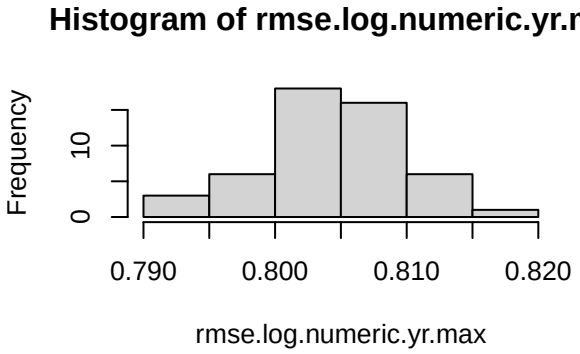
```r
train$Model.Year<- factor(train$Model.Year)
train$Max.Ethanol<- factor(train$Max.Ethanol)
# convert Model.Year and Max.Ethanol to categorical variables

rmse.log.factor.yr.max<-c()
for (i in 1:50){
    log.factor.yr.max <- randomForest(log(Comb.FE)~.*.+
                        log(Eng.Displacement)+log(No.Cylinders)+
                        log(No.Gears)+log(Eng.Displacement*No.Gears)+
                        I(Eng.Displacement^2)+I(No.Cylinders^2)+
                        I(No.Gears^2),
                        train, ntree = 100)

    predict.label <- exp(predict(log.factor.yr.max, train[1:9]))
    rmse.log.factor.yr.max<-c(rmse.log.factor.yr.max,
                              rmse(predict.label, train$Comb.FE))}
```

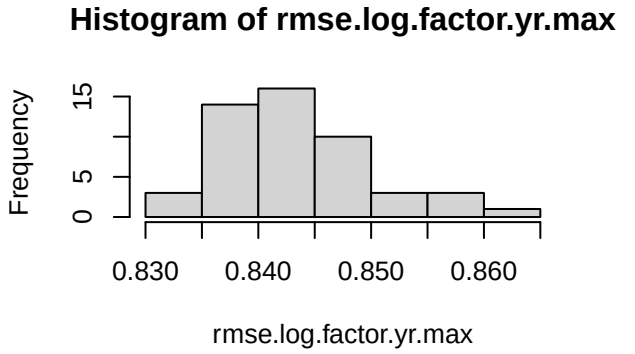
```r
rmse.fit.factor.yr.max<-c()
for (i in 1:50){
    fit.factor.yr.max <- randomForest(Comb.FE~.*.+
                        log(Eng.Displacement)+log(No.Cylinders)+
                        log(No.Gears)+log(Eng.Displacement*No.Gears)+
                        I(Eng.Displacement^2)+I(No.Cylinders^2)+
                        I(No.Gears^2),
                        train, ntree = 100)

    predict.label <- predict(fit.factor.yr.max, train[1:9])
    rmse.fit.factor.yr.max<-c(rmse.fit.factor.yr.max,
                              rmse(predict.label, train$Comb.FE))}
```

```r
par(mfrow=c(2,2))
hist(rmse.fit.factor.yr.max)
hist(rmse.fit.numeric.yr.max)
hist(rmse.log.numeric.yr.max)
hist(rmse.log.factor.yr.max)
```










```
# plot the rmse of each model after all the simulations
```

Since `randomForest` generates decision tree with randomly chosen input preditors and choose the best decision tree out of all the generated trees, the model built at each simulation cannot be identical to another. This implies that the rmse could also differ based on each random forest generated from each simulation. To overcome this problem, I ran 50 simulation of each random forest, and created histogram of rmse from each model.

Based on four histograms above, `rmse.fit.numeric.yr.max` model has the least spread and the smallest mean compared to all other random forest model. Therefore, I chose this model as my final model for prediction.

```
train$Model.Year<- as.numeric(train$Model.Year)
train$No.Cylinders<- as.numeric(train$No.Cylinders)
train$Max.Ethanol<- as.numeric(train$Max.Ethanol)
# convert Model.Year, No.Cylinders, and Max.Ethanol to numerical variables
```

```
# The final model
fin.mod<-randomForest(Comb.FE~.*.+
                log(Eng.Displacement)+log(No.Cylinders)+
                log(No.Gears)+log(Max.Ethanol)+log(Model.Year)+
               log(Eng.Displacement*No.Gears)+
                I(Eng.Displacement^2)+I(No.Cylinders^2)+
                I(No.Gears^2)+I(Max.Ethanol^2)+I(Model.Year^2),
              train, ntree = 150)
```

```
levels(test$Aspiration)<- fin.mod$forest$xlevel$Aspiration
levels(test$Lockup.Torque.Converter)<- fin.mod$forest$xlevel$Lockup.Torque.Converter
levels(test$Drive.Sys)<- fin.mod$forest$xlevel$Drive.Sys
levels(test$Fuel.Type)<- fin.mod$forest$xlevel$Fuel.Type
# fit the levels of categorical features to that of the model
```

```r
# values predicted based on the model
pred.label <- predict(fin.mod, test)
# export data frame after inserting the newly generated values
write.csv(data.frame("RowIndex" = seq(1, length(pred.label)), "Prediction" = pred.label),
          "RegressionPredictLabel.csv", row.names = F)
```

Part 2. Multiple Regression for Fuel Efficiency Prediction

This part of the program predicts the fuel efficiency with a simple multiple regression modelling and step-wise regression modelling.

Fitting a Multiple Linear Model

We starts off by fitting a multiple linear model to the `train` dataset and interpret the predictors' significance using 0.05 significant level.

```r
# retrieve column names of the train dataset
names(train)
```

```
##  [1] "Model.Year"              "Eng.Displacement"
##  [3] "No.Cylinders"            "Aspiration"
##  [5] "No.Gears"                "Lockup.Torque.Converter"
##  [7] "Drive.Sys"               "Max.Ethanol"
##  [9] "Fuel.Type"               "Comb.FE"
```

`Comb.FE` is our target variable - the fuel efficiency. We first fit all the predictors to a multiple regression model as below.

```r
# create a multiple regression model with all the predictors to predict Comb.FE
lm.fit <- lm(Comb.FE~., train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Comb.FE ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9311 -0.9994 -0.0668  0.7007 11.3673
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               15.85624    0.32487  48.808  < 2e-16 ***
## Model.Year                 0.10303    0.04257   2.420 0.015632 *
## Eng.Displacement          -1.36781    0.10277 -13.309  < 2e-16 ***
## No.Cylinders               0.04285    0.06798   0.630 0.528583
## AspirationOT              -0.31845    0.63632  -0.500 0.616837
## AspirationSC              -0.92497    0.23058  -4.011 6.36e-05 ***
## AspirationTC              -1.27850    0.12890  -9.918  < 2e-16 ***
## AspirationTS              -1.12557    0.49586  -2.270 0.023365 *
## No.Gears                  -0.12902    0.03000  -4.301 1.82e-05 ***
## Lockup.Torque.ConverterY -0.85118    0.11182  -7.612 4.97e-14 ***
## Drive.SysA                -0.08133    0.15326  -0.531 0.595731
## Drive.SysF                 1.44711    0.17170   8.428  < 2e-16 ***
## Drive.SysP                -0.35446    0.30153  -1.176 0.239990
```

```
## Drive.SysR                0.03447    0.14822   0.233 0.816135
## Max.Ethanol               0.06944    0.08342   0.832 0.405278
## Fuel.TypeGM               0.70372    0.41961   1.677 0.093755 .
## Fuel.TypeGP               0.46454    0.13906   3.341 0.000858 ***
## Fuel.TypeGPR              0.21823    0.14432   1.512 0.130721
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.601 on 1382 degrees of freedom
## Multiple R-squared:  0.6616, Adjusted R-squared:  0.6574
## F-statistic: 158.9 on 17 and 1382 DF,  p-value: < 2.2e-16
```

A Multiple regression model is a parametric model, where we need to make an estimation of the parameters. The parameters for this model are the coefficients of the predictors, in this case all the attributes in the dataset except for the target variable `Comb.FE`.

To identify and evaluate the importance of these predictor variables for predicting the target, we perform a hypothesis testing and check whether the parameters of the model, $\hat{\beta}_i$ equals 0. If the test rejects the null hypothesis ($H_0 : \beta_i = 0$), we have sufficient evidence to claim that the respective parameters are related to the target variable. The lower the p-value, the more evidence we have to reject the null hypothesis, and thus the predictors are more related to the target variable predicted by the model.

With 0.05 significant level for p-value, `Model.Year`, `Eng.Displacement`, all levels of `Aspiration` attribute, `No.Gears`, `Lockup.Torque.ConverterY`, `Drive.SysA`, `Max.Ethanol`, and `Fuel.TypeGP` are possibly associated with expected `Comb.FE` since we have strong evidence to reject the H_0 that the regression coefficients of these predictor variables are equal to 0.

Based on the multiple regression model built above, the `Eng.Displacement`, `AspirationTC`, and `Drive.SysF` has the lowest p-values. The lower the p-value, the stronger our evidence against the null, which implies that the `Eng.Displacement`, `AspirationTC`, and `Drive.SysF` appear to be the strongest predictors of fuel efficiency.

0.0964 is the coefficient of one of the predictive variable `Model.Year`. 0.0964 regression coefficient implies that the expected `fuel efficiency` calculated by our model increases by 0.0964 km/l when the `Model.Year` increases by 1 year, which is 1 unit of the predictor `Model.Year`.

Similarly, -0.1307 is the coefficient of the predictive variable `No.Gears`. This means that the expected `fuel efficiency` calculated by the multi-regression model above decreases by 0.1307 when the `No.Gears` increases by 1 unit. The coefficient is negative and thus the target value decreases when the respective predictor increases by 1 unit.

Applying Step-wise Regression Modelling using BIC penalty

To evaluate the predictive properties of each predictors, we can apply the information criterion and use step-wise regression to 'filter' the predictors one by one.

```r
sw.fit <- step(lm.fit, k = log(nrow(train)), trace = 0, direction = 'both')
# k = log(n) is refers to the information criterion of BIC
# n is the number of observations in the dataset, which is returned by nrow(train)
summary(sw.fit)
```

```
##
## Call:
## lm(formula = Comb.FE ~ Eng.Displacement + Aspiration + No.Gears +
##     Lockup.Torque.Converter + Drive.Sys, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4.0110 -0.9809 -0.0421  0.6207 11.3917
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                16.04607    0.27859  57.597  < 2e-16 ***
## Eng.Displacement           -1.28169    0.04350 -29.465  < 2e-16 ***
## AspirationOT               -0.03176    0.62741  -0.051 0.959634
## AspirationSC               -0.65834    0.21383  -3.079 0.002119 **
## AspirationTC               -1.09908    0.10639 -10.331  < 2e-16 ***
## AspirationTS               -1.07002    0.48235  -2.218 0.026692 *
## No.Gears                   -0.10848    0.02920  -3.715 0.000211 ***
## Lockup.Torque.ConverterY -0.85816    0.10987  -7.810 1.12e-14 ***
## Drive.SysA                  0.05179    0.14587   0.355 0.722622
## Drive.SysF                  1.48823    0.16725   8.898  < 2e-16 ***
## Drive.SysP                 -0.43937    0.29049  -1.513 0.130632
## Drive.SysR                  0.09357    0.14670   0.638 0.523716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.609 on 1388 degrees of freedom
## Multiple R-squared:  0.6566, Adjusted R-squared:  0.6539
## F-statistic: 241.3 on 11 and 1388 DF,  p-value: < 2.2e-16
```

The final multiple regression equation derived by the sample is:

$$E[\text{ Fuel Efficiency }] = 16.20 - 1.28 \text{ Eng.Displacement} - 0.10 \text{ AspirationOT} - 0.70 \text{ AspirationSC} - 1.14 \text{ AspirationTC} - 1.12 \text{ As}$$

Conclusions on Predictors based on the BIC Model

The intercept of the BIC model, 16.1969, is the value of the expected fuel efficiency when all the predictors are 0.

The regression coefficient of each predictor is the amount the predicted value, in this case, the fuel efficiency, changes with one-unit change of the respective predictor. With such statistical interpretation, we can look for the predictors that impacts the expected fuel efficiency the most. The higher the fuel efficiency, the more efficient our new car could be.

One thing to notice here is that `Drive.Sys`, `Aspiration`, and `Lockup.Torque.Converter` are all categorical variables, which implies that all the predictors related to the variables above are Booleans. As an instance, if a car's `Drive.Sys == 'A'`, `Drive.SysA` then equals to 1, while `Drive.SysF`, `Drive.SysP`, and `Drive.SysR` becomes 0. The predictor that increases the expected fuel efficiency with one unit increase of the predictor has positive regression coefficients. These predictors are`Drive.SysF`, `Drive.SysA`, and `Drive.SysR`. The fuel efficiency increases the most at `Drive.SysF` by 1.4802 km/L. Our model thus suggests that the expected fuel efficiency could be the highest when `Drive.Sys == 'A'`.

On the other hand, all `Aspiration` predictors have negative regression coefficients, with `AspirationOT` with the lowest slope at -0.1001. This model implies that the predicted fuel efficiency decreases the least when `Aspiration == 'OT'`, since all other `Aspiration` variables becomes 0 if `Aspiration == 'OT'`.

`Lockup.Torque.ConverterY` also has a negative regression coefficient of -0.8253. This is also a Boolean predictor that becomes 0 when `Lockup.Torque.Converter == 'N'`.

In fact, the predictor that impacts the expected fuel efficiency the most could be `No.Gears` and `Eng.Displacement`, since the former could be a positive integer range between 1 to 10, while the latter has a relatively high but negative regression coefficient but is a continuous numerical value ranging between 0.9 to

8. The increase of one unit of `Eng.Displacement` and `No.Gears` could decrease the expected fuel efficiency by 1.2772km/L and 0.1135 km/L respectively.

In conclusion, this BIC model suggests that a car satisfying the criteria below could increase the expected fuel efficiency:

- all-wheel drive (`Drive.sysA == 1`)
- aspiration type OT (`AspirationOT == 1`)
- NOT transferring rotating power from an internal combustion engine to a rotating driven load (`Lockup.Torque.ConverterY == 0`)
- Lower engine displacement
- Lower number of gears
- Lower level of ethanol in the fuel

95% Confidence Interval using BIC Model

```
predict(sw.fit, test[1,], interval = 'confidence')
```

```
##        fit      lwr      upr
## 1 9.282567 9.047692 9.517442
```

The predicted mean fuel efficiency for this car is 9.2873km/l and the 95% confidence interval for the predicted mean fuel efficiency is between 9.0530km/l and 9.5216km/l.

The new car's predicted mean fuel efficiency is 9.2873km/l and its upper bound is at 9.5216km/l. If, let's say, our current car's fuel efficiency is 9.5km/l, this number is higher than the predicted mean of the fuel efficiency of the first car and is close to the upper bound of 95% confidence interval of mean fuel efficiency. Therefore, it is more likely that our current car is more efficient than this new car.