**Linear Model**

> summary(model\_1)

Call:

lm(formula = price ~ highwaympg + citympg + peakrpm + horsepower +

compressionratio + stroke + bore + enginesize + curbweight +

height + width + length + wheelbase)

Residuals:

Min 1Q Median 3Q Max

-11674.3 -1620.6 4.4 1543.8 13809.5

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -63633.132 16432.763 -3.872 0.000151 \*\*\*

highwaympg 287.178 164.689 1.744 0.082918 .

citympg -305.769 182.951 -1.671 0.096407 .

peakrpm 2.117 0.677 3.127 0.002063 \*\*

horsepower 37.280 18.211 2.047 0.042109 \*

compressionratio 247.104 86.596 2.854 0.004834 \*\*

stroke -2962.854 797.179 -3.717 0.000270 \*\*\*

bore -838.703 1217.519 -0.689 0.491802

enginesize 127.760 15.208 8.401 1.33e-14 \*\*\*

curbweight 1.639 1.739 0.942 0.347275

height 326.301 143.014 2.282 0.023691 \*

width 632.908 258.021 2.453 0.015128 \*

length -84.382 57.919 -1.457 0.146894

wheelbase 60.048 104.790 0.573 0.567342

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3168 on 179 degrees of freedom

Multiple R-squared: 0.857, Adjusted R-squared: 0.8466

F-statistic: 82.52 on 13 and 179 DF, p-value: < 2.2e-16

**#it can be seen that p-value of the F-statistics is < 2.26-16, which is highly significant. This means that, at least, one of the 13**

**explanatory variables is significantly related to the responsive variable**

**#Multiple R-squared is 0.857, which indicate a pretty well fit to the data**

**#based on the coefficients table we can see that**

**the coefficient with \*\*\* are statistically significant**

**the coefficient with \*\* are somewhat statistically significant**

**the coefficient with \* are not very statistically significant**

**the coefficient with are not statistically significant**

**in summary, changing in engine size and stroke are significantly associated to changes in price, changing in compression ratio and peak rpm are somewhat associate to changes in price while changing in rest variables are not significantly associated with price.**

> model2 <- lm(price ~ stroke + enginesize, data = df)

> summary(model2)

Call:

lm(formula = price ~ stroke + enginesize, data = df)

Residuals:

Min 1Q Median 3Q Max

-13553.2 -1770.5 -260.7 1128.9 14074.0

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1126.508 2730.551 -0.413 0.68040

stroke -2544.482 853.327 -2.982 0.00324 \*\*

enginesize 177.001 6.472 27.350 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3643 on 190 degrees of freedom

Multiple R-squared: 0.7993, Adjusted R-squared: 0.7972

F-statistic: 378.4 on 2 and 190 DF, p-value: < 2.2e-16

**#based on this model, we can tell that engine size have more significantly effect on the price**

> model3 <- lm(price ~ enginesize, data = df)

> summary(model3)

Call:

lm(formula = price ~ enginesize, data = df)

Residuals:

Min 1Q Median 3Q Max

-11490 -2031 -193 1460 14050

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -8862.79 868.66 -10.2 <2e-16 \*\*\*

enginesize 172.86 6.45 26.8 <2e-16 \*\*\*

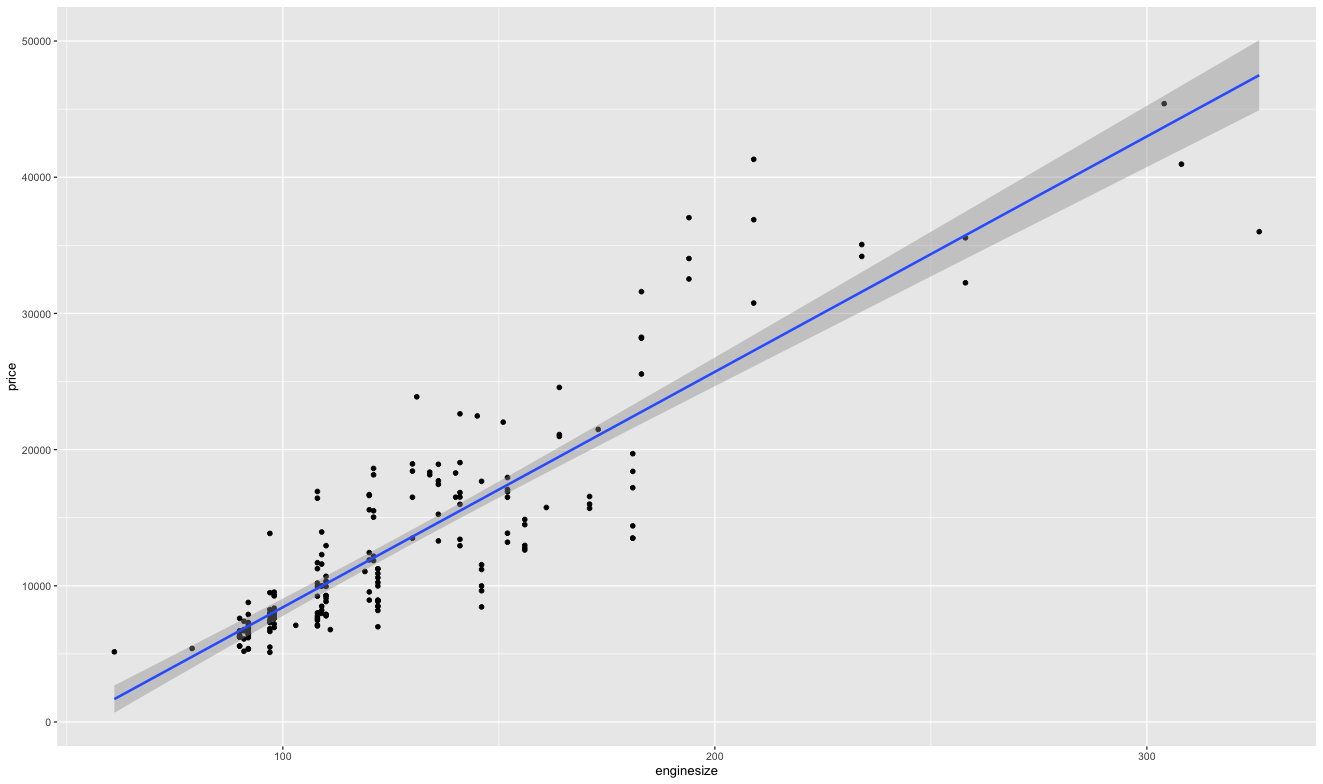
---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3717 on 191 degrees of freedom

Multiple R-squared: 0.7899, Adjusted R-squared: 0.7888

F-statistic: 718.2 on 1 and 191 DF, p-value: < 2.2e-16



**Linear Model Bootstrap Procedure**

The bootstrap method is used to quantify the uncertainty associated with given statistical estimator or with a predictive model

The bootstrap procedure consists of randomly selecting a sample of n observations from the original data set. The subset, called bootstrap data set is then sued to evaluate the model.

The bootstrap process, repeated large number of times and the standard error of the bootstrap estimate is then calculated. The results provide an indiation of the variance of the model's performance

> boot.control <- trainControl(method = "boot", number = 500)

> model\_1\_boot <- train(price ~., data = df, method = "lm", trControl

= boot.control)

> print(model\_1\_boot)

Linear Regression

193 samples

13 predictor

No pre-processing

Resampling: Bootstrapped (500 reps)

Summary of sample sizes: 193, 193, 193, 193, 193, 193, ...

Resampling results:

RMSE Rsquared MAE

3513.489 0.821008 2458.846

Tuning parameter 'intercept' was held constant at a value of TRUE

**The output shows the average model\_1 performance across 500 resamples**

**RMSE(root mean squared error) measures the model prediction error,**

**the lower the better the model.**

**MAE(Mean Absolute Error) measures the model prediction error, the**

**lower the better the model.**

**The R squared represent the proportion of variation in the outcome**

**explained by the predictor variables included in the model. The**

**higher the better.**

> boot.control <- trainControl(method = "boot", number = 500)

> model\_3\_boot <- train(price ~ enginesize, data = df, method = "lm",

trControl = boot.control)

> print(model\_2\_boot)

Linear Regression

193 samples

2 predictor

No pre-processing

Resampling: Bootstrapped (500 reps)

Summary of sample sizes: 193, 193, 193, 193, 193, 193, ...

Resampling results:

RMSE Rsquared MAE

3740.456 0.7973457 2582.375

Tuning parameter 'intercept' was held constant at a value of TRUE

boot.control <- trainControl(method = "boot", number = 500)

model\_3\_boot <- train(price ~ enginesize, data = df, method = "lm",

trControl = boot.control)

print(model\_3\_boot)

> print(model\_3\_boot)

Linear Regression

193 samples

1 predictor

No pre-processing

Resampling: Bootstrapped (500 reps)

Summary of sample sizes: 193, 193, 193, 193, 193, 193, ...

Resampling results:

RMSE Rsquared MAE

3786.08 0.7873644 2671.926

Tuning parameter 'intercept' was held constant at a value of TRUE

**Appendix**

library(readxl)

Car <- read\_excel("Documents/4-Lasso／Ridge Regression／Random Decision Forest-1850/Car.xlsx")

View(Car)

price <- Car$price

head(price)

highwaympg <- Car$highwayMpg

head(highwaympg)

citympg <- Car$cityMpg

head(citympg)

peakrpm <- Car$peakRpm

head(peakrpm)

horsepower <- Car$horsepower

head(horsepower)

compressionratio <- Car$compressionRatio

head(compressionratio)

stroke <- Car$stroke

head(stroke)

bore <- Car$bore

head(bore)

enginesize <- Car$engineSize

head(enginesize)

curbweight <- Car$curbWeight

head(curbweight)

height <- Car$height

head(height)

width <- Car$width

head(width)

length <- Car$length

head(length)

wheelbase <- Car$wheelBase

head(wheelbase)

#We first study the regression model between the price and the 13 explanatory variables by a linear regression:

model\_1 <- lm(price ~ highwaympg + citympg + peakrpm + horsepower + compressionratio + stroke + bore + enginesize + curbweight + height + width + length + wheelbase)

summary(model\_1)

#build a model with only statistically significant variables

model2 <- lm(price ~ stroke + enginesize, data = df)

summary(model2)

#build a model with only engine size explanatory variable

model3 <- lm(price ~ enginesize, data = df)

summary(model3)

#plot this model3

ggplot(model3, aes(enginesize, price)) +

geom\_point()+

stat\_smooth(method = lm, formula = y ~ x )

library(boot)

library(tidyverse)

library(caret)

#bootstrap procedure for model\_1 multiple linear regression, we use a bootstrap with 500 samples to test this model

boot.control <- trainControl(method = "boot", number = 500)

model\_1\_boot <- train(price ~., data = df, method = "lm", trControl = boot.control)

print(model\_1\_boot)

#bootstrap procedure for model\_2 multiple linear regression with 2 variables, we use a bootstrap with 500 samples to test this model

boot.control <- trainControl(method = "boot", number = 500)

model\_2\_boot <- train(price ~ enginesize + stroke , data = df, method = "lm", trControl = boot.control)

print(model\_2\_boot)

#bootstrap procedure for model\_3 simple linear regression with only 1 variables, used 500 samples to test this model.

boot.control <- trainControl(method = "boot", number = 500)

model\_3\_boot <- train(price ~ enginesize, data = df, method = "lm", trControl = boot.control)

print(model\_3\_boot)

**Polynomial Model**

> model4 <- lm(price ~ poly(enginesize, 2, raw = TRUE), data = df)

> summary(model4)

Call:

lm(formula = price ~ poly(enginesize, 2, raw = TRUE), data = df)

Residuals:

Min 1Q Median 3Q Max

-9402.6 -1980.5 -49.2 1462.8 13778.1

Coefficients:

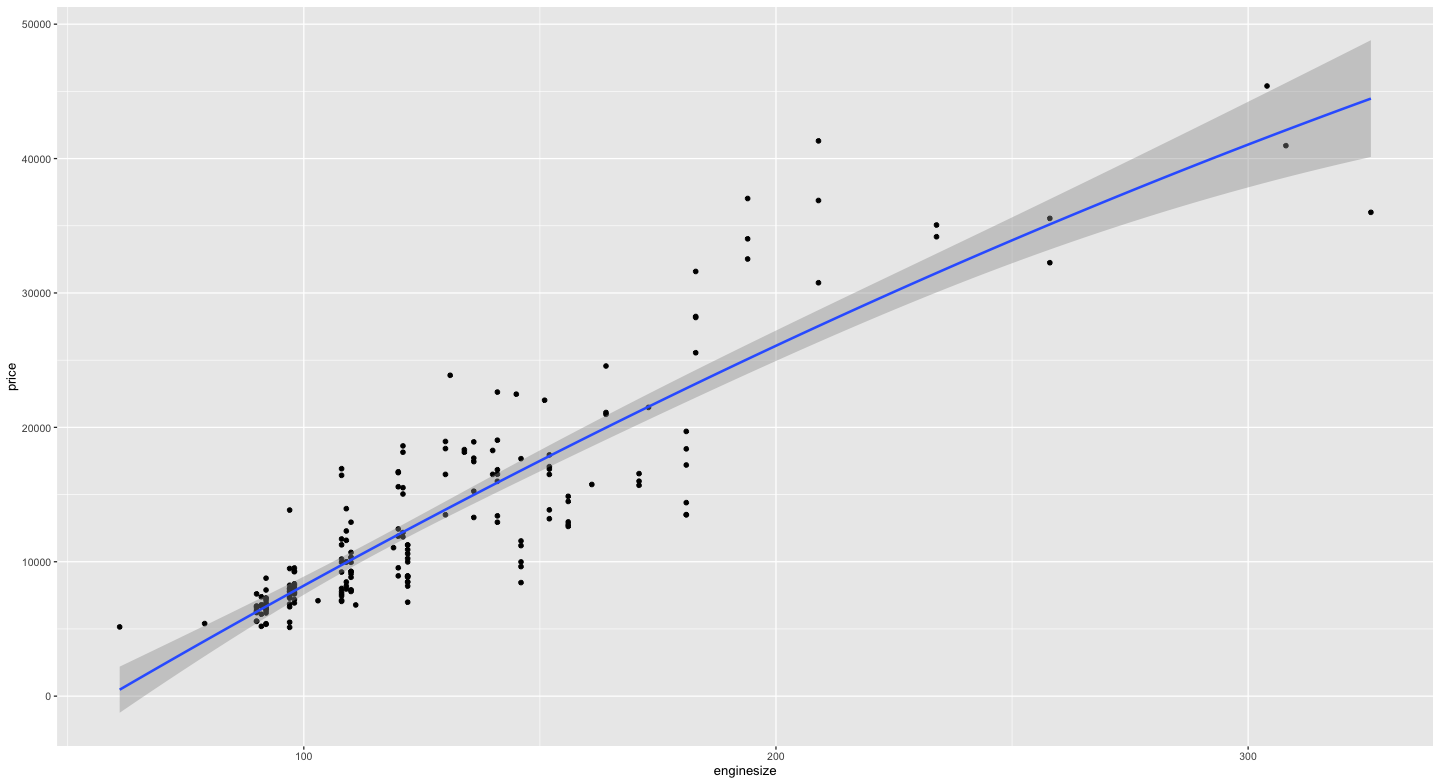
Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.251e+04 2.312e+03 -5.410 1.88e-07 \*\*\*

poly(enginesize, 2, raw = TRUE)1 2.217e+02 2.943e+01 7.532 1.97e-12 \*\*\*

poly(enginesize, 2, raw = TRUE)2 -1.439e-01 8.469e-02 -1.700 0.0908 .

**# price = price = b0 + b1\*engine size + b2\*engine size2**



> model5 <- lm(price ~ log(enginesize), data = df)

> summary(model5)

Call:

lm(formula = price ~ log(enginesize), data = df)

Residuals:

Min 1Q Median 3Q Max

-9694 -2225 115 1795 14442

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -109803 4923 -22.30 <2e-16 \*\*\*

log(enginesize) 25584 1022 25.04 <2e-16 \*\*\*

---

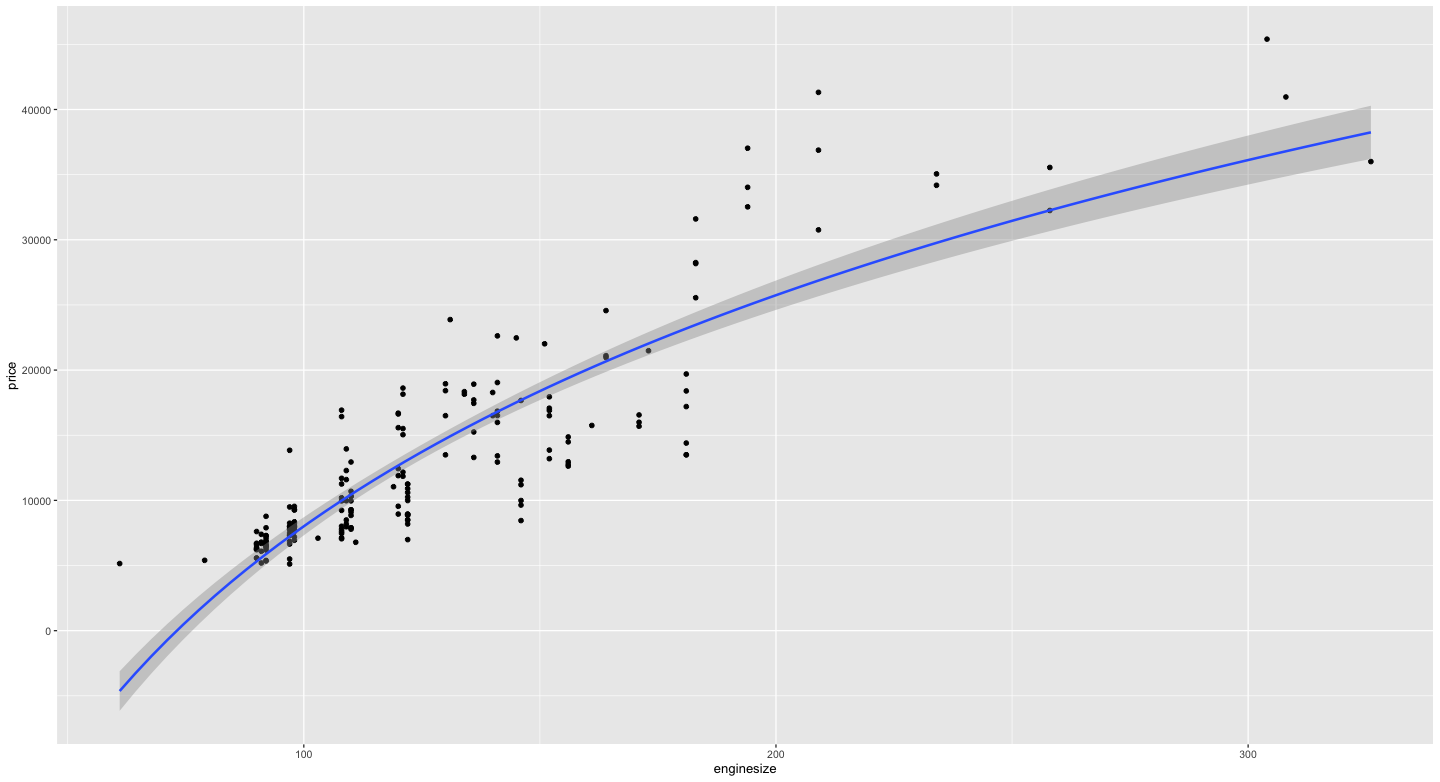
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3919 on 191 degrees of freedom

Multiple R-squared: 0.7665, Adjusted R-squared: 0.7653

F-statistic: 627.1 on 1 and 191 DF, p-value: < 2.2e-16

**# price = b0 + b1\*log(engine size)**



Appendix

# we are going to build polynomial regression between responsive variable "price" and explanatory variable"engine size"

model4 <- lm(price ~ poly(enginesize, 2, raw = TRUE), data = df)

summary(model4)

# plot model 4

ggplot(model4, aes(enginesize, price)) +

geom\_point()+

stat\_smooth(method = lm, formula = y ~ poly(x, 2, raw = TRUE))

# build model 5 based on log(enginesize)

model5 <- lm(price ~ log(enginesize), data = df)

summary(model5)

#plot model 5

ggplot(model5, aes(enginesize, price))+

geom\_point()+

stat\_smooth(method = lm, formula = y ~ log(x))

**Lasso/Ridge Model**

The standard linear model (or the ordinary least squares method) performs poorly in a situation, where you have a large multivariate data set containing a number of variables superior to the number of samples.

A better alternative is the penalized regression allowing to create a linear regression model that is penalized, for having too many variables in the model, by adding a constraint in the equation (James et al. 2014,P. Bruce and Bruce (2017)). This is also known as shrinkage or regularization methods.

The most commonly used penalized regression methods, including ridge regression, lasso regression and elastic net regression.

> coef(model\_ridge)

15 x 1 sparse Matrix of class "dgCMatrix"

s0

(Intercept) -52521.413997

(Intercept) .

highwaympg 33.695113

citympg -55.411996

peakrpm 1.433091

horsepower 50.342830

compressionratio 202.774111

stroke -2179.591382

bore -504.192996

enginesize 90.287572

curbweight 2.073589

height 152.271556

width 525.062090

length -18.370088

wheelbase 39.219352

**#we can see that compared to other variables, “peakrpm”, “curbweight” were shrink(roughly) to zero**

**# Ridge regression shrinks the regression coefficients, so that variables, with minor contribution to the outcome, have their coefficients close to zero.**

> coef(model\_lasso)

15 x 1 sparse Matrix of class "dgCMatrix"

s0

(Intercept) -56268.890897

(Intercept) .

highwaympg .

citympg .

peakrpm 1.593027

horsepower 38.898365

compressionratio 184.045835

stroke -2095.917630

bore .

enginesize 120.525882

curbweight 0.581188

height 208.862026

width 518.480814

length .

wheelbase .

**#we can see that only 8 variables have non-zero coefficients. The coefficients of all other variables have been set to zero by the lasso algorithm, reducing the complexity of the model**

**#Lasso stands for Least Absolute Shrinkage and Selection Operator. It shrinks the regression coefficients toward zero by penalizing the regression model with a penalty term called L1-norm, which is the sum of the absolute coefficients**

**Appendix**

library(tidyverse)

library(dplyr)

library(caret)

library(glmnet)

df <- data.frame(price, highwaympg, citympg, peakrpm , horsepower , compressionratio , stroke , bore , enginesize , curbweight , height , width , length , wheelbase)

y <- price

head(y)

x <- model.matrix(price ~ highwaympg + citympg + peakrpm + horsepower + compressionratio + stroke + bore + enginesize + curbweight + height + width + length + wheelbase)

#computing ridge regression

#find the best lambda using cross-validation, lambda = a number value defining the amount of shrinkage

set.seed(123)

cv <- cv.glmnet(x, y, alpha = 0)

#display the best lambda value

cv$lambda.min

#fit the final model

model\_ridge <- glmnet(x, y, alpha = 0, lambda = cv$lambda.min)

#display regression coefficients

coef(model\_ridge)

#computing lasso regression

#find the best lambda using cross-validation

set.seed(123)

cv <- cv.glmnet(x, y, alpha = 1)

#display the best lambda value

cv$lambda.min

#fit the final model

model\_lasso <- glmnet(x, y, alpha = 1,lambda = cv$lambda.min)

#display regression coefficients

coef(model\_lasso)

**Random Forest Model**

Call:

randomForest(formula = price ~ ., data = df)

Type of random forest: regression

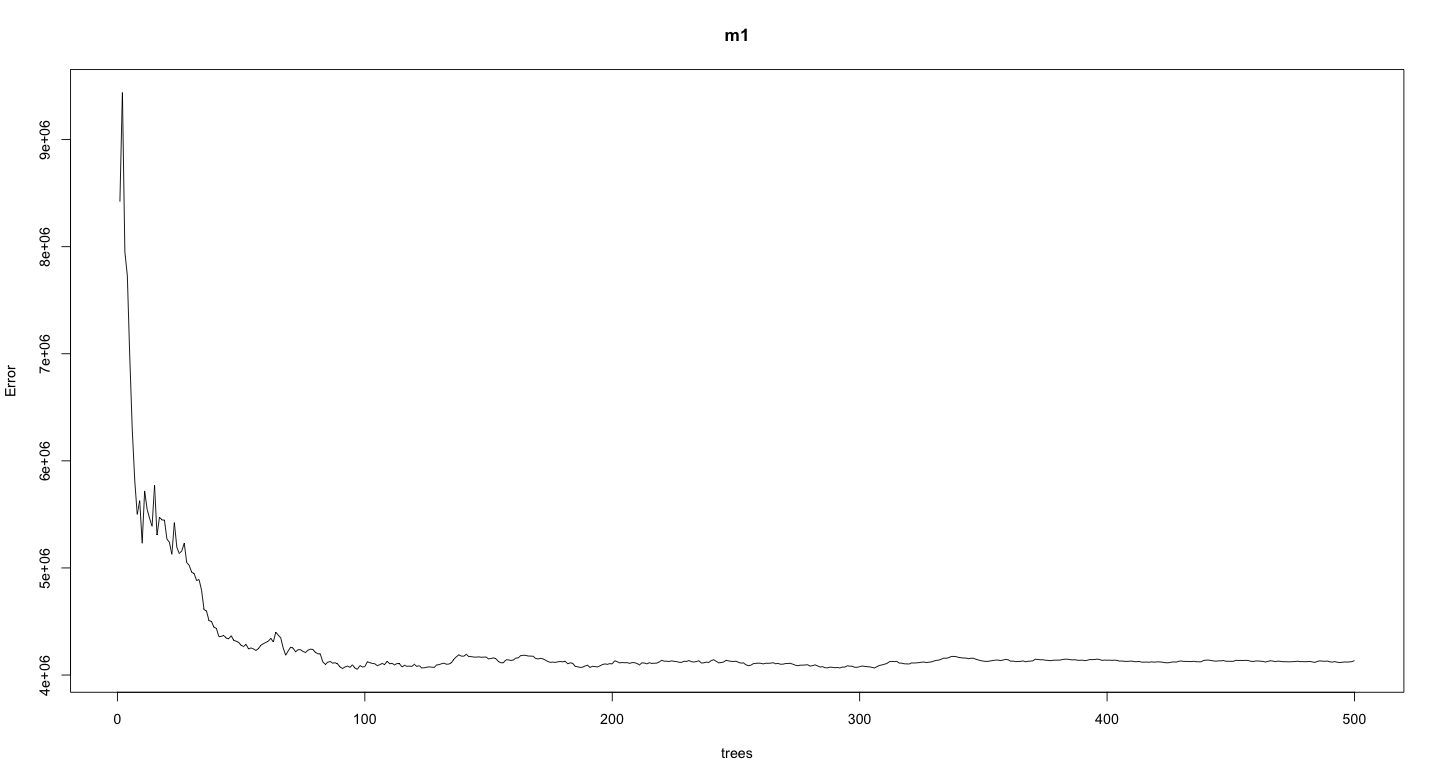
Number of trees: 500

No. of variables tried at each split: 4

Mean of squared residuals: 4134536

% Var explained: 93.65

> plot(m1)



> which.min(m1$mse) [1] 97

> sqrt(m1$mse[which.min(m1$mse)]) [1] 2013.531

**#we can find which number of tress providing the lowest error rate and the result is 97 trees providing an average car sales price error of $2013.53**

model\_randomforest <- train(price~., data = df, method = "rf",

trControl = trainControl("cv", number = 13), importance = TRUE)

model\_randomforest$bestTune

mtry

27 model\_randomforest$finalModel Call:

randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 7

Mean of squared residuals: 4177836

% Var explained: 93.58

**#in this method, we used the caret workflow, which invoked the randomforest() function to automatically select the optimal**

**number(mtry) [which is 7]of predictor variables randomly sampled as candidates at each split**

> varImp(m1)

Overall

highwaympg 1155914809

citympg 1698936572

peakrpm 143867512

horsepower 1777509056

compressionratio 150172622

stroke 82049090

bore 143176747

enginesize 3300931389

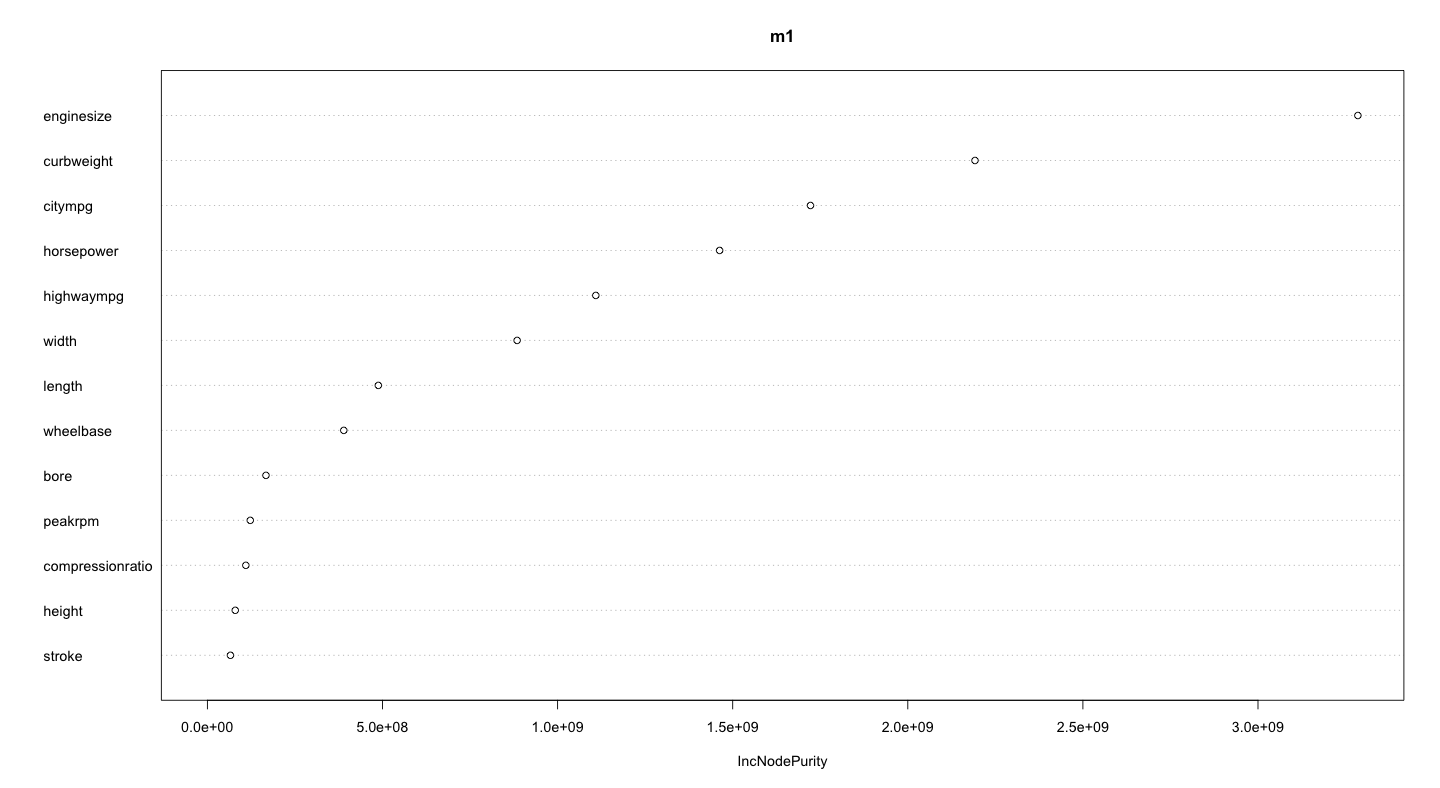
curbweight 2381692550

height 85127338

width 550259688

length 472596984

wheelbase 345065255



Appendix

library(rsample)

library(randomForest)

library(ranger)

library(caret)

library(h2o)

#split data into training(80%) and test set(20%)

training.samples <- df$price %>% createDataPartition(p=0.8, list = FALSE)

train.data <- df[training.samples,]

test.data <- df[-training.samples,]

#default Random Forest model

m1 <- randomForest(formula = price ~., data = df)

m1

plot(m1)

#plotting the model m1 illustrate the error rate drops significantly around 30 tress and stabalizes around 100 trees

#we can find which number of tress providing the lowest error rate and the result is 97 trees providing an average car sales price error of $2013.53

which.min(m1$mse)

sqrt(m1$mse[which.min(m1$mse)])

#use another method to fit the model

model\_randomforest <- train(price~., data = df, method = "rf", trControl = trainControl("cv", number = 13), importance = TRUE)

model\_randomforest$bestTune

#ranking variable importance

importance\_m1 <- m1$importance

importance\_m1

#obtain variable importance

varImp(m1)

#rank the variable importance by MeanDecreaseGini

varImpPlot(m1, type = 2)