R Notebook

Used Car Regressions

Load Tools for Project

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ lubridate 1.9.2     ✓ tibble    3.2.1
## ✓ purrr     1.0.1     ✓ tidyr     1.3.0
## ✓ readr     2.1.4
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(fastDummies)
```

```
## Thank you for using fastDummies!
## To acknowledge our work, please cite the package:
## Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variable
s. Version 1.7.1. URL: https://github.com/jacobkap/fastDummies, https://jacobkap.github.io/fastDummies/.
```

```r
library(caret)
```

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

```r
library(AER)
```

```
## Loading required package: car
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:purrr':
##
##     some
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## Loading required package: lmtest
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
```

```
library(estimatr)
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.3.2
```

```
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
##
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.2
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.3.2
```

Load Data

```
car <- read.csv('Clean Data_pakwheels.csv')
```

Show Data Frame

```
car
```

X <int>	Company.Name <chr>	Model.Name <chr>	Price <int>	Model.Year <int>	Location <chr>	Mileage <int>	Engine.Type <chr>	Engine.Capacity <int>
0	Toyota	Vitz	2385000	2017	Islamabad	9869	Petrol	1000
1	Toyota	Corolla	111000	2019	KPK	11111	Petrol	1300
2	Suzuki	Alto	1530000	2019	KPK	17500	Petrol	660
3	Suzuki	Alto	1650000	2019	Punjab	9600	Petrol	660
4	Toyota	Corolla	1435000	2010	Islamabad	120000	Petrol	1300
5	Honda	Civic	3850000	2017	Punjab	22000	Petrol	1500
6	Suzuki	Wagon	1440000	2017	Punjab	31000	Petrol	1000
7	Mitsubishi	Mirage	1425000	2012	Punjab	101000	Petrol	1000
8	Toyota	Prado	2650000	1998	Punjab	110000	Diesel	3000
9	Honda	Civic	3350000	2017	Punjab	60000	Petrol	1800

1-10 of 10,000 rows | 1-9 of 14 columns Previous **1** 2 3 4 5 6 ... 1000 Next

Drop NA's

```
car%>%
  mutate(drop_na(car))
```

X <int>	Company.Name <chr>	Model.Name <chr>	Price <int>	Model.Year <int>	Location <chr>	Mileage <int>	Engine.Type <chr>	Engine.Capacity <int>
0	Toyota	Vitz	2385000	2017	Islamabad	9869	Petrol	1000
1	Toyota	Corolla	111000	2019	KPK	11111	Petrol	1300
2	Suzuki	Alto	1530000	2019	KPK	17500	Petrol	660
3	Suzuki	Alto	1650000	2019	Punjab	9600	Petrol	660

X <int>	Company.Name <chr>	Model.Name <chr>	Price <int>	Model.Year <int>	Location <chr>	Mileage <int>	Engine.Type <chr>	Engine.Capacity <int>
4	Toyota	Corolla	1435000	2010	Islamabad	120000	Petrol	1300
5	Honda	Civic	3850000	2017	Punjab	22000	Petrol	1500
6	Suzuki	Wagon	1440000	2017	Punjab	31000	Petrol	1000
7	Mitsubishi	Mirage	1425000	2012	Punjab	101000	Petrol	1000
8	Toyota	Prado	2650000	1998	Punjab	110000	Diesel	3000
9	Honda	Civic	3350000	2017	Punjab	60000	Petrol	1800

1-10 of 10,000 rows | 1-9 of 14 columns Previous 1 2 3 4 5 6 ... 1000 Next

Drop all rows that are not Punjab Region, for simplification of model

```
car_2 <-car%>%
  filter(., Location %in% c('Punjab'))
```

Show Car2 data frame

```
car_2
```

X <int>	Company.Name <chr>	Model.Name <chr>	Price <int>	Model.Year <int>	Location <chr>	Mileage <int>	Engine.Type <chr>	Engine.Capacity <int>
3	Suzuki	Alto	1650000	2019	Punjab	9600	Petrol	660
5	Honda	Civic	3850000	2017	Punjab	22000	Petrol	1500
6	Suzuki	Wagon	1440000	2017	Punjab	31000	Petrol	1000
7	Mitsubishi	Mirage	1425000	2012	Punjab	101000	Petrol	1000
8	Toyota	Prado	2650000	1998	Punjab	110000	Diesel	3000
9	Honda	Civic	3350000	2017	Punjab	60000	Petrol	1800
11	Honda	City	1990000	2017	Punjab	75000	Petrol	1300

X	Company.Name	Model.Name	Price	Model.Year	Location	Mileage	Engine.Type	Engine.Capacity
<int>	<chr>	<chr>	<int>	<int>	<chr>	<int>	<chr>	<int>
12	Honda	N	185000	2016	Punjab	20000	Petrol	660
13	Suzuki	Cultus	920000	2012	Punjab	83000	Petrol	1000
14	Toyota	Corolla	2750000	2018	Punjab	51240	Petrol	1300

1-10 of 10,000 rows | 1-9 of 14 columns Previous **1** 2 3 4 5 6 ... 1000 Next

Mutate car2 to car3 for mutations of data

```
car_3 <- car_2 %>%

  #group data by decade

  mutate(decade_1 = case_when( Model.Year >= '1990' & Model.Year <= '2000' ~ 1,Model.Year >= '2001' ~ 0 ))%>%
  mutate(decade_2 = case_when( Model.Year >= '2001' & Model.Year <= '2010' ~ 1,Model.Year >= '2011'|Model.Year < '2001'~0))%
>%
  mutate(decade_3 = case_when( Model.Year >='2011' & Model.Year <= '2019' ~ 1, Model.Year < '2011'|Model.Year > '2019'~0))%
>%



  #group data by manufacturing location

  mutate(East_asia = case_when(Company.Name == 'Toyota'
                              |Company.Name == 'Honda'
                              |Company.Name == 'Daihatsu'
                              |Company.Name == 'Nissan'
                              |Company.Name =='Mitsubishi'
                              |Company.Name == 'Hyundai'
                              |Company.Name == 'FAW'
                              |Company.Name == 'Suzuki' ~ 1,

                              Company.Name != 'Toyota'
                              |Company.Name !='Honda'
                              |Company.Name !='Daihatsu'
                              |Company.Name !='Nissan'
                              |Company.Name !='Hyundai'
                              |Company.Name !='Suzuki'
                              |Company.Name != 'Mitsubishi' ~0 ))%>%

  mutate(german = case_when(Company.Name == 'Audi'
                              |Company.Name=='Mercedes'
                              |Company.Name=='BMW'~ 1,

                              Company.Name !='Audi'
                              |Company.Name !='Mercedes'
                              |Company.Name !='BMW' ~0))%>%
```

```
#create a dummy for transmission type

mutate(tran_dum = case_when(Transmission.Type =="Manual" ~ 0, Transmission.Type =="Automatic" ~ 1))%>%



#create dummies for engine type, hybrid and diesel

mutate(Engine_num = case_when(Engine.Type== 'Petrol'| Engine.Type=='Hybrid'~ 1, Engine.Type=='Diesel'~ 0))%>%



#Local vehicles serve as the baseline

mutate(Assembly_num =case_when(Assembly == 'Local' ~ 0, Assembly == 'Imported' ~ 1))%>%



#Control for body type

mutate(dummy_cols(., select_columns ='Body.Type'))%>%



#Control for color, separated in three categories

mutate(.,Color_num = case_when(Color=='Black'~ 'Black', Color =='White'~ 'White', Color!='Black'|Color!='White' ~ 'Othe
r'))%>%

mutate((dummy_cols(.,select_columns = 'Color_num')))%>%



#Convert price into USD for context (this might change according to the audience)
#Conversion on November 25th 2023 is 83.31 rupees to 1 dollar

mutate(USD = Price/83.31)%>%
```

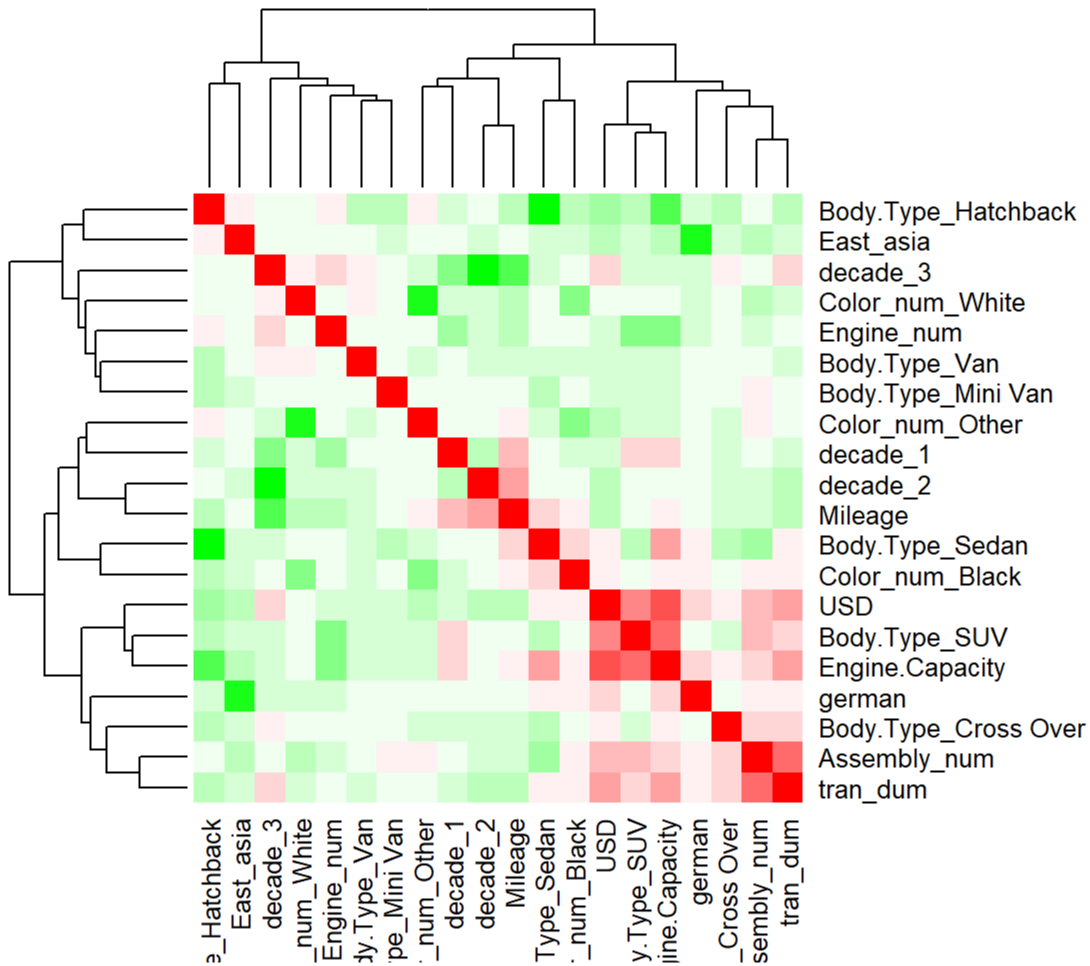
```
  #Select your variables that you want

  select(.,East_asia, german, Mileage, decade_1,decade_2,decade_3, Engine.Capacity,tran_dum:Body.Type_Van,Color_num_Black:US
D)
```

See relationship and check for multi-collinearity & Relationship with Target Variable

```
cor_check <- cor(car_3)
#cor_check
```

```
palette = colorRampPalette(c("green", "white", "red")) (20)
heatmap(x = cor_check, col = palette, symm = TRUE)
```

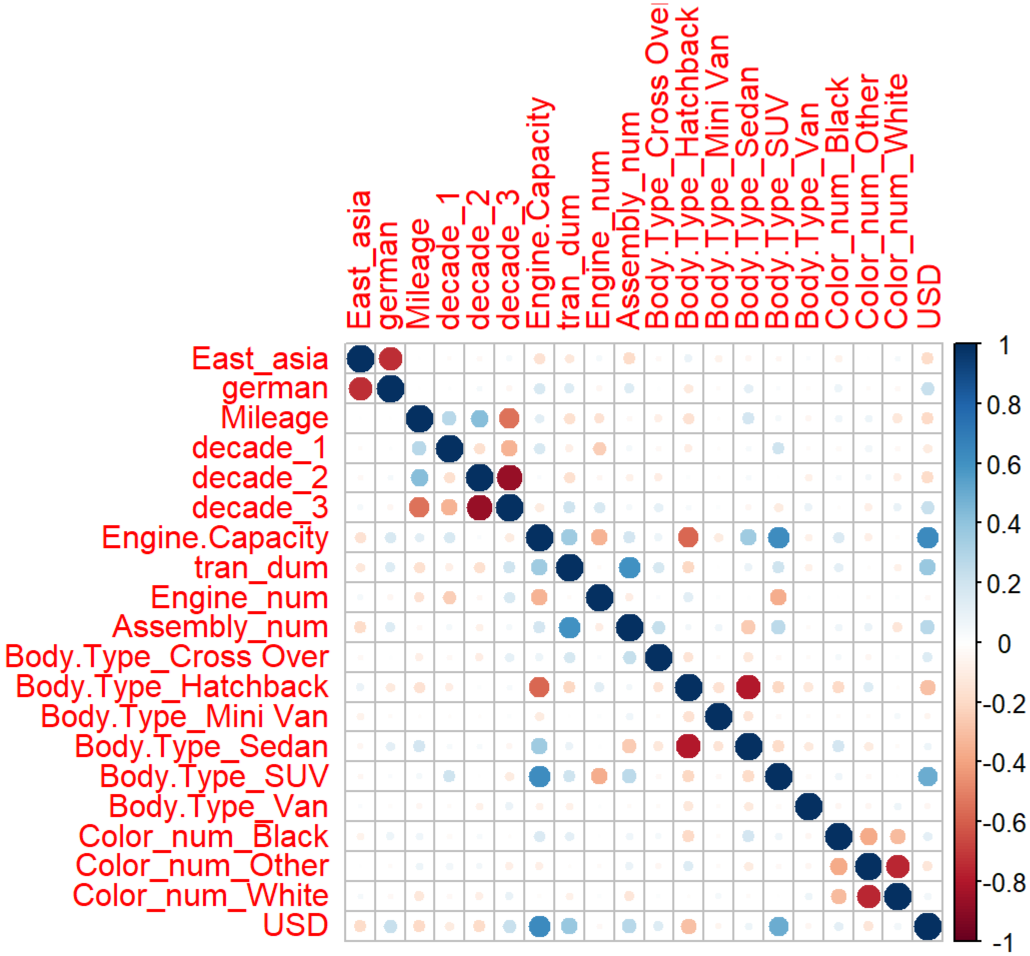



```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.2
```

```
## corrplot 0.92 loaded
```

```r
corrplot(cor_check)
```



Run a simple model of price on the variable with the highest correlation

```
base <- lm(USD ~ Engine.Capacity, car_3)
summary(base)
```

```
##
## Call:
## lm(formula = USD ~ Engine.Capacity, data = car_3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -182603   -6921    -436    5273  572794
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -18378.25     351.86  -52.23   <2e-16 ***
## Engine.Capacity     32.07       0.25  128.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22610 on 24730 degrees of freedom
## Multiple R-squared:  0.3995, Adjusted R-squared:  0.3995
## F-statistic: 1.645e+04 on 1 and 24730 DF,  p-value: < 2.2e-16
```

Run Regression based on selected features (baseline model)

```
mult_reg <- lm(USD~ Engine.Capacity + Mileage + tran_dum + Engine_num + Color_num_Other + Color_num_White + Body.Type_SUV +
Body.Type_Hatchback + `Body.Type_Cross Over`+ Assembly_num, car_3)

summary(mult_reg)
```

```
##
## Call:
## lm(formula = USD ~ Engine.Capacity + Mileage + tran_dum + Engine_num +
##     Color_num_Other + Color_num_White + Body.Type_SUV + Body.Type_Hatchback +
##     `Body.Type_Cross Over` + Assembly_num, data = car_3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -137792   -4901    -202    4111  537075
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -5.431e+04  1.284e+03 -42.295  < 2e-16 ***
## Engine.Capacity         3.059e+01  3.615e-01  84.627  < 2e-16 ***
## Mileage                -9.823e-02  2.091e-03 -46.983  < 2e-16 ***
## tran_dum                2.414e+03  3.558e+02   6.784 1.19e-11 ***
## Engine_num              4.350e+04  1.005e+03  43.298  < 2e-16 ***
## Color_num_Other        -3.572e+03  3.940e+02  -9.065  < 2e-16 ***
## Color_num_White         4.692e+02  4.026e+02   1.165    0.244
## Body.Type_SUV           2.635e+04  8.359e+02  31.521  < 2e-16 ***
## Body.Type_Hatchback     3.124e+03  3.322e+02   9.401  < 2e-16 ***
## `Body.Type_Cross Over`  1.093e+04  9.016e+02  12.119  < 2e-16 ***
## Assembly_num            5.752e+03  3.888e+02  14.794  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19610 on 24721 degrees of freedom
## Multiple R-squared:  0.5483, Adjusted R-squared:  0.5481
## F-statistic:  3001 on 10 and 24721 DF,  p-value: < 2.2e-16
```

Lasso for feature selection

```
model_lasso <- train(USD ~ .,
               data = car_3,
               method = "glmnet",
               tuneGrid = data.frame(alpha=1,
                                     lambda=seq(0.0000,1)))
model_lasso
```

```
## glmnet
##
## 24732 samples
##    19 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 24732, 24732, 24732, 24732, 24732, 24732, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE      Rsquared   MAE
##   0       18584.18  0.6006657  7876.787
##   1       18584.18  0.6006657  7876.787
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 1.
```

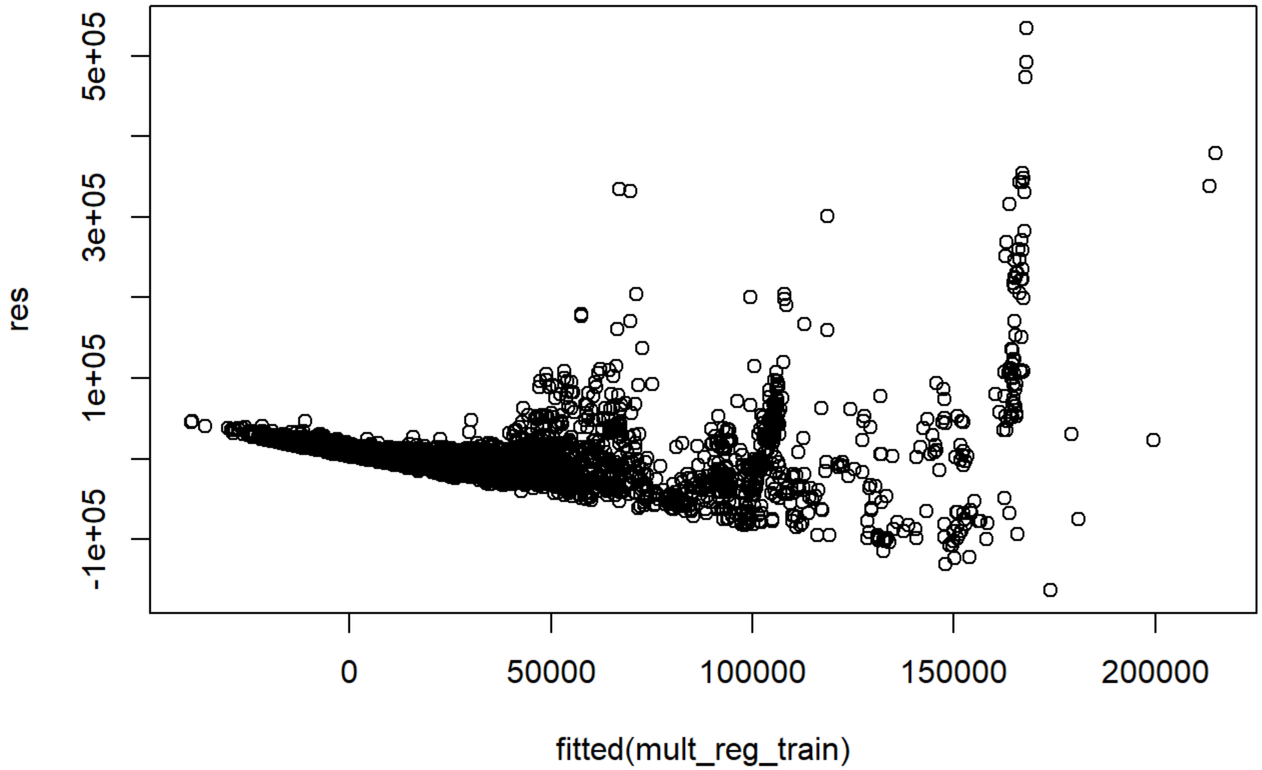
Predicting car price based on our model Train and Testing Split for generalization

```
set.seed(12L)
trainIndex <- createDataPartition(car_3$USD,
                                  p = 0.8,
                                  list = FALSE,
                                  times = 1)
car_3_train <- car_3[trainIndex, ]
car_3_test <- car_3[-trainIndex, ]
```

```
mult_reg_train <- lm(USD~ Engine.Capacity + Mileage + tran_dum + Engine_num
+Color_num_Black + Color_num_White + Body.Type_SUV + Body.Type_Hatchback + Body.Type_Sedan+
+ Body.Type_Van + East_asia + decade_2 + decade_3+ Assembly_num, data=car_3_train)
summary(mult_reg_train)
```

```
##
## Call:
## lm(formula = USD ~ Engine.Capacity + Mileage + tran_dum + Engine_num +
##     Color_num_Black + Color_num_White + Body.Type_SUV + Body.Type_Hatchback +
##     Body.Type_Sedan + +Body.Type_Van + East_asia + decade_2 +
##     decade_3 + Assembly_num, data = car_3_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -163421   -5001    -364    4298  534109
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -5.521e+04  1.728e+03 -31.953  < 2e-16 ***
## Engine.Capacity      3.106e+01  4.077e-01  76.193  < 2e-16 ***
## Mileage             -4.714e-02  2.637e-03 -17.874  < 2e-16 ***
## tran_dum             2.247e+02  3.947e+02   0.569   0.5692
## Engine_num           3.823e+04  1.105e+03  34.603  < 2e-16 ***
## Color_num_Black      2.343e+03  4.272e+02   5.485 4.18e-08 ***
## Color_num_White      3.450e+03  2.989e+02  11.540  < 2e-16 ***
## Body.Type_SUV        2.476e+04  1.133e+03  21.848  < 2e-16 ***
## Body.Type_Hatchback -1.535e+03  6.635e+02  -2.313   0.0207 *
## Body.Type_Sedan     -5.287e+03  6.938e+02  -7.620 2.64e-14 ***
## Body.Type_Van       -6.649e+03  1.200e+03  -5.542 3.02e-08 ***
## East_asia           -1.623e+04  8.283e+02 -19.599  < 2e-16 ***
## decade_2             1.301e+04  6.280e+02  20.710  < 2e-16 ***
## decade_3             2.407e+04  6.541e+02  36.797  < 2e-16 ***
## Assembly_num         5.664e+03  4.344e+02  13.037  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19000 on 19772 degrees of freedom
## Multiple R-squared:  0.5823, Adjusted R-squared:  0.582
## F-statistic:  1969 on 14 and 19772 DF,  p-value: < 2.2e-16
```

```
#check residuals
res <- resid(mult_reg_train)
plot(fitted(mult_reg_train), res)
```




```
##predicting with multiple regression

pred_mult <- predict(mult_reg_train, car_3_test)
#pred_mult
postResample(pred = pred_mult, car_3_test$USD)
```

```
##         RMSE     Rsquared          MAE
## 1.709639e+04 6.345992e-01 7.888478e+03
```

```
#lets improve with a lasso regression
model_lasso <- train(USD ~ .,
               data = car_3_train,
               method = "glmnet",

               tuneGrid = data.frame(alpha=1,
                                     lambda=seq(0.0001,1)))
model_lasso
```

```
## glmnet
##
## 19787 samples
##    19 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 19787, 19787, 19787, 19787, 19787, 19787, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   18550.92  0.5954659  7903.787
##
## Tuning parameter 'alpha' was held constant at a value of 1
## Tuning
##  parameter 'lambda' was held constant at a value of 1e-04
```

```
#summary(model_lasso)
#Summary doesn't work for advanced models

#model_2 <-train(USD~., car_3_train)
```

Now predict outcomes in test set

```
p <- predict(model_lasso, car_3_test, type = 'raw')
postResample(pred=p, obs= car_3_test$USD)
```

```
##         RMSE     Rsquared          MAE
## 1.697333e+04 6.398262e-01 7.679397e+03
```

```
#RMSE(p)
paste("MSE: ", mean((p - car_3_test$USD)^2))
```

```
## [1] "MSE:  288094060.736897"
```

```
paste("RMSE: ", sqrt(mean((p - car_3_test$USD)^2)))
```

```
## [1] "RMSE:  16973.333813276"
```

```
# add predictions to initial dataset
#c_test$pred_churn <- p
```

Preparing data for Decision Trees and Random Forests

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.2
```

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

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

```
##
## Attaching package: 'randomForest'
```

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

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

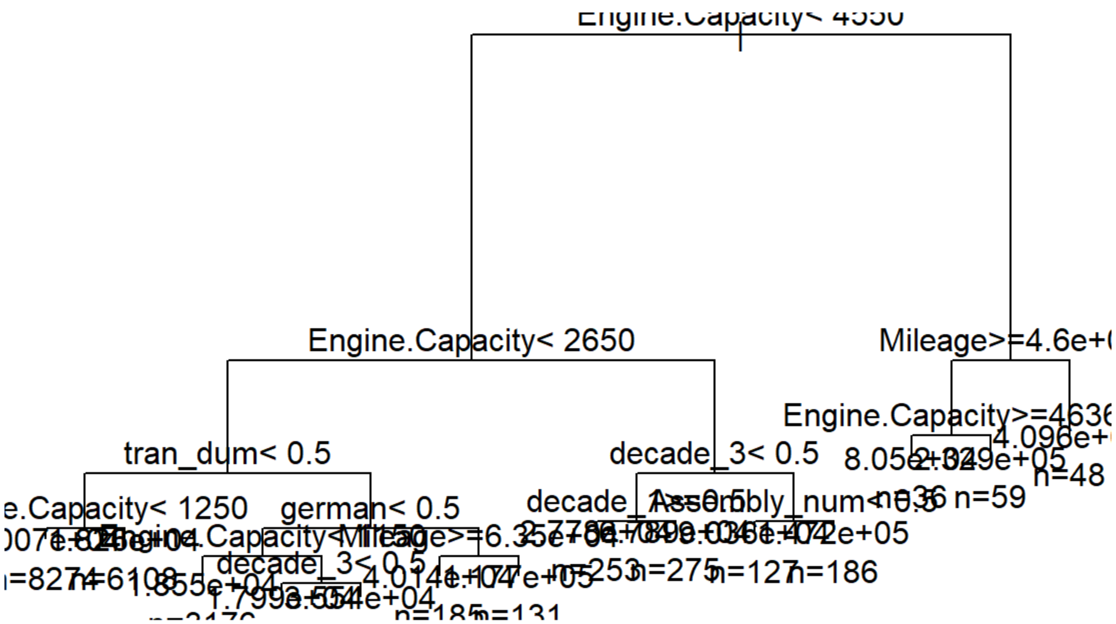
```
#print(car_3)

car_3 <- car_3 %>% rename(Body.Type_CrossOver = 'Body.Type_Cross Over')
car_3 <- car_3 %>% rename(Body.Type_MiniVan = 'Body.Type_Mini Van')
car_3 <- car_3 %>% rename(Price = USD)
```

Decision Trees

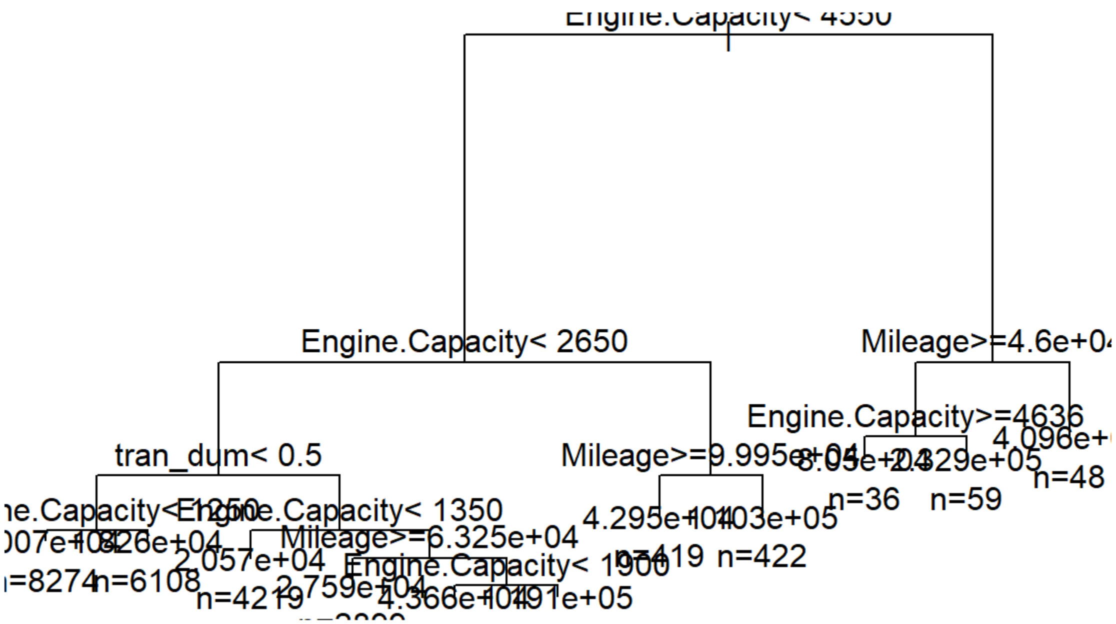
```
#Decision tree with every feature. A little messy
decision_tree_model <- rpart(Price ~ ., data = car_3, method = "anova")

plot(decision_tree_model)
text(decision_tree_model, use.n = TRUE)
```




```
#This tree only is using three main features
decision_tree_model <- rpart(Price ~ Engine.Capacity + Mileage + tran_dum, data = car_3, method = "anova")

plot(decision_tree_model)
text(decision_tree_model, use.n = TRUE)
```



Random Forests:

The first RF model we ran was a regular one with all the features included. I tried to run it with certain features omitted, but it would return a higher RMSE each time.

```
#Regular RF
set.seed(123)  # reproducibility
# Splitting the data into training and test sets
train_indices <- sample(1:nrow(car_3), 0.8 * nrow(car_3))
train_data <- car_3[train_indices, ]
test_data <- car_3[-train_indices, ]


# Fit model on training data
fitted_model <- randomForest(Price ~ ., data = train_data, ntree = 500)


# Predict on test data
predictions <- predict(fitted_model, test_data)

# Calculate MSE
results <- data.frame(predictions, test_data$Price)
results$Difference = abs(results$predictions - results$test_data.Price)
print("Predictions for RF with all variables included")
```

```
## [1] "Predictions for RF with all variables included"
```

```
head(results)
```

	predictions <dbl>	test_data.Price <dbl>	Difference <dbl>
1	15000.02	19805.546	4805.526
6	34974.59	40211.259	5236.665
8	20427.83	2220.622	18207.209
12	12498.34	10923.058	1575.282
24	39878.42	45432.721	5554.306
29	24604.93	28688.033	4083.099

6 rows

```
paste("MSE: ", mean((results$predictions - results$test_data.Price)^2))
```

```
## [1] "MSE:  54137181.7664882"
```

```
paste("RMSE: ", sqrt(mean((results$predictions - results$test_data.Price)^2)))
```

```
## [1] "RMSE:  7357.79734475531"
```

```
paste("R-Squared: ", cor(results$test_data.Price, results$predictions)^2)
```

```
## [1] "R-Squared:  0.932957112939525"
```

```
#Tried running RF with select features, but no combination was nearly as good as including all features
#fitted_model <- randomForest(Price ~ East_asia + german + Mileage +  Engine.Capacity + tran_dum + Engine_num +
#                               Assembly_num + Body.Type_CrossOver + Body.Type_Hatchback + Body.Type_MiniVan + Body.Type_Sed
an + Body.Type_SUV + Body.Type_Van +
#                               Color_num_Black + Color_num_Other + Color_num_White, data = train_data, ntree = 500)


# Predict on test data
#predictions <- predict(fitted_model, test_data)

#results <- data.frame(predictions, test_data$Price)
#print("Predictions for RF with all variables included")
#head(results)
#paste("MSE: ", mean((results$predictions - results$test_data.Price)^2))
#paste("RMSE: ", sqrt(mean((results$predictions - results$test_data.Price)^2)))
```

Next I tried to focus on outliers, which began with limiting the lowest-end cars, however this had no significant change

```
#Removing the lowest end cars
car_4 <- subset(car_3, Price >= 8000)

# Splitting the data into training and test sets
train_indices <- sample(1:nrow(car_4), 0.8 * nrow(car_4))
train_data <- car_4[train_indices, ]
test_data <- car_4[-train_indices, ]


# Fit model on training data
fitted_model <- randomForest(Price ~ ., data = train_data, ntree = 500)

# Predict on test data
predictions <- predict(fitted_model, test_data)

# Calculate MSE
results <- data.frame(predictions, test_data$Price)
print("Predictions for RF with outliers removed")
```

```
## [1] "Predictions for RF with outliers removed"
```

```
head(results)
```

	predictions <dbl>	test_data.Price <dbl>
4	21180.38	17104.79
10	30137.08	33009.24
13	102459.84	103829.07
15	26690.83	25207.06
20	24735.51	23826.67
29	24932.26	28688.03

6 rows

```
paste("MSE: ", mean((results$predictions - results$test_data.Price)^2))
```

```
## [1] "MSE:  100650958.855945"
```

```
paste("RMSE: ", sqrt(mean((results$predictions - results$test_data.Price)^2)))
```

```
## [1] "RMSE:  10032.4951460713"
```

```
paste("R-Squared: ", cor(results$test_data.Price, results$predictions)^2)
```

```
## [1] "R-Squared:  0.90872669929471"
```

When I did the opposite and left out observations with an actual price over $60,000, it significantly improved and I had the best RMSE by far (Around 3400, which represents the model being off by an average of 3400 each time, which isn't bad with it having to do with car prices)

```
#Removing the highest end cars (Best Model)
car_4 <- subset(car_3, Price <= 60000)

# Splitting the data into training and test sets
train_indices <- sample(1:nrow(car_4), 0.8 * nrow(car_4))
train_data <- car_4[train_indices, ]
test_data <- car_4[-train_indices, ]


# Fit model on training data
fitted_model <- randomForest(Price ~ ., data = train_data, ntree = 500)

# Predict on test data
predictions <- predict(fitted_model, test_data)

# Calculate MSE
results <- data.frame(predictions, test_data$Price)
print("Predictions for RF with outliers removed")
```

```
## [1] "Predictions for RF with outliers removed"
```

```
head(results)
```

	predictions <dbl>	test_data.Price <dbl>
7	24004.159	23886.688
9	13547.106	11043.092
12	12505.082	10923.058
19	4872.387	4801.344
26	17413.588	17344.857
29	24709.331	28688.033

6 rows

```
paste("MSE: ", mean((results$predictions - results$test_data.Price)^2))
```

```
## [1] "MSE:  11273425.2957469"
```

```
paste("RMSE: ", sqrt(mean((results$predictions - results$test_data.Price)^2)))
```

```
## [1] "RMSE:  3357.59218722985"
```

```
paste("R-Squared: ", cor(results$test_data.Price, results$predictions)^2)
```

```
## [1] "R-Squared:  0.907603848410807"
```

The final alterations I tried were with limiting the number of trees in the model and this had a slight impact

```
# Reducing number of trees to 128
# Splitting the data into training and test sets
train_indices <- sample(1:nrow(car_3), 0.8 * nrow(car_3))
train_data <- car_3[train_indices, ]
test_data <- car_3[-train_indices, ]


# Fit model on training data
fitted_model <- randomForest(Price ~ ., data = train_data, ntree = 128)

# Predict on test data
predictions <- predict(fitted_model, test_data)

# Calculate MSE
results <- data.frame(predictions, test_data$Price)
print("Predictions for RF with all variables included and less trees (128)")
```

```
## [1] "Predictions for RF with all variables included and less trees (128)"
```

```
head(results)
```

	predictions <dbl>	test_data.Price <dbl>
1	15189.40	19805.55
4	21125.13	17104.79
6	35552.29	40211.26
9	13589.93	11043.09
18	17189.38	17824.99
21	26765.29	29348.22

6 rows

```
paste("MSE: ", mean((results$predictions - results$test_data.Price)^2))
```

```
## [1] "MSE:  68082783.0993424"
```

```
paste("RMSE: ", sqrt(mean((results$predictions - results$test_data.Price)^2)))
```

```
## [1] "RMSE:  8251.22918717826"
```

```
paste("R-Squared: ", cor(results$test_data.Price, results$predictions)^2)
```

```
## [1] "R-Squared:  0.939420844952495"
```

```
#Tried many different number of trees, did not make major changes, 128 seemed optimal
# Splitting the data into training and test sets
train_indices <- sample(1:nrow(car_3), 0.8 * nrow(car_3))
train_data <- car_3[train_indices, ]
test_data <- car_3[-train_indices, ]


# Fit model on training data
fitted_model <- randomForest(Price ~ ., data = train_data, ntree = 100)


# Predict on test data
predictions <- predict(fitted_model, test_data)

# Calculate MSE
results <- data.frame(predictions, test_data$Price)
print("Predictions for RF with all variables included")
```

```
## [1] "Predictions for RF with all variables included"
```

```
head(results)
```

	predictions <dbl>	test_data.Price <dbl>
5	27891.521	31808.906
11	16621.496	13803.865
16	14658.224	11583.243
22	22149.355	16324.571
25	17435.199	17885.008
32	7077.242	7502.101

6 rows

```
paste("MSE: ", mean((results$predictions - results$test_data.Price)^2))
```

```
## [1] "MSE:  64327807.0111158"
```

```
paste("RMSE: ", sqrt(mean((results$predictions - results$test_data.Price)^2)))
```

```
## [1] "RMSE:  8020.46177044164"
```

```
paste("R-Squared: ", cor(results$test_data.Price, results$predictions)^2)
```

```
## [1] "R-Squared:  0.920298039622535"
```

```
# Less Trees + select features
# Splitting the data into training and test sets
train_indices <- sample(1:nrow(car_3), 0.8 * nrow(car_3))
train_data <- car_3[train_indices, ]
test_data <- car_3[-train_indices, ]


# Fit model on training data
fitted_model <- randomForest(Price ~ Engine.Capacity + Mileage + tran_dum, data = train_data, ntree = 128)


# Predict on test data
predictions <- predict(fitted_model, test_data)

# Calculate MSE
results <- data.frame(predictions, test_data$Price)
print("Predictions for RF with three variables included (Engine Capacity, Mileage, Transmission) and less trees (128)")
```

```
## [1] "Predictions for RF with three variables included (Engine Capacity, Mileage, Transmission) and less trees (128)"
```

```
head(results)
```

	predictions <dbl>	test_data.Price <dbl>
3	15624.05	17284.84
13	76200.38	103829.07
14	30735.39	19505.46
22	14435.44	16324.57
28	34403.84	21786.10
30	23837.14	16804.71

6 rows

```
paste("MSE: ", mean((results$predictions - results$test_data.Price)^2))
```

```
## [1] "MSE:  226144479.413305"
```

```
paste("RMSE: ", sqrt(mean((results$predictions - results$test_data.Price)^2)))
```

```
## [1] "RMSE:  15038.1009244288"
```

```
paste("R-Squared: ", cor(results$test_data.Price, results$predictions)^2)
```

```
## [1] "R-Squared:  0.815039819609284"
```