**PAC Report: Predictive Analytics Summer I  
Anshita Thakkar**

The PAC competition explored the impact of a number of variables on the selling price of used cars. The final model I used for the PAC competition was a decision tree through which I yielded the lowest RMSE score amongst my attempts and my highest score of 3780.75467. The final code is as follows:

tree10= rpart(price~is\_new+wheelbase\_inches+horsepower+daysonmarket+highway\_fuel\_economy+engine\_displacement+mileage+seller\_rating+owner\_count+year+maximum\_seating+fuel\_tank\_volume\_gallons+highway\_fuel\_economy+city\_fuel\_economy+back\_legroom\_inches+front\_legroom\_inches+length\_inches+width\_inches+height\_inches+has\_BackupCamera+has\_NavigationSystem+power+powerRPM+torque+torqueRPM+has\_BlindSpotMonitoring+has\_ConveniencePackage+has\_AndroidAuto+has\_ParkingSensors+has\_TechnologyPackage+has\_PreferredPackage+has\_HeatPackage+has\_SEPackage+has\_PremiumWheels+has\_SportPackage+has\_ColdWeatherPackage+isCoupe+isHatchback+isMinivan+isPickupTruck+isSedan+isSUV+isWagon,data = train, method = 'anova', cp = 0.000000001)

The prediction was predictions10 <- predict(tree10, newdata = scoringData)

The measure I used to check the RMSE score of the train and test data was as follows: rmse\_train10 = sqrt(mean((predictions10 - train$price)^2))

pred10\_test = predict(tree10, newdata=test)

rmse\_test10 = sqrt(mean((pred10\_test - test$price)^2))

rmse\_train10

rmse\_test10

The train RMSE score was 23726.39, while the test RMSE score was 4497.457

. The low test RSME score was a positive sign that the model would work well beyond the analysis dataset, and that it could be generalized the model had good predictive performance.

I outline the steps I took in choosing to use a decision tree for my final model.

Exploring the data

After reading in the csv file, I used a number of techniques to explore the data. I ran the scoring data = scoringData set as well as the analysis data = usedCars\_decisiontree to understand the data, see what the variables are and identify the data that needed parsing. I used the functions ncol() and nrow() to check that both the scoring data and analysis data has the same number of columns and rows, and to identify the number of datapoints I was dealing with. At an initial glance, I noticed that the columns ‘power’ and ‘torque’ had important numeric values with metrics that required parsing and would be useful numeric data to analyze. I reflected on my own insight of buying a car to identify that power and torque would be important determinants in a buyer’s decision.

*Data tidying – analysis & scoring data*

I used code from the Data Tidying section on classes and the ‘forcats’ package to mutate and parse the data. I created two new columns from power which were ‘power’ and ‘power RPM’ both with separate values; values separated by @. Additionally, I created two new columns from torque which were ‘torque’ and ‘torqueRPM’. This code is referenced at the end of this document. I parsed all four columns to remove the metrics so the numbers would be analyzable. I applied the same code to the scoringData so that there would be no issues when it came to submitting my model. As an inexperienced coder, this step took me a couple of days to execute.

Secondly, I identified ‘major\_options’ as a column that required parsing as it had important features such as a backup camera or navigation system for example which in my opinion would influence a buyer’s car purchase decision. I used the ‘toString’ function to remove the square brackets and quotation marks. I then used the ‘dplyr’ package’s data tidying code like **group\_by()**, **summarise()**, **mutate()**, **arrange()**, and **ungroup()**to identify the distribution of features in the major\_option column in descending order and the frequency of each feature in the total count. I later created new columns for these features.

*Splitting the data*

Before creating new coluns, I split the usedCar\_decisiontree2 dataset into train and test at p = 0.75 and groups = 10. I set seed to 2267. I then printed the train and test data to ensure that there were no errors in running the code and the numbers totaled the original dataset.

*Creating new columns – major\_options and body\_type*

I felt the features of the major\_options and body\_type columns would be important determinants in a buyer’s decision to buy a car.

I used the stringr code to create new columns for major\_options. An example of this code is as follows: train$has\_BackupCamera = ifelse(str\_detect(string = train$major\_options,pattern = 'Backup Camera'),TRUE,FALSE). I ran this code for each feature on the train, test and scoringData datasets separately.

I then made used code to make the new columns/ features into factors so no issues would arise when analyzing the data and using it against the test or scoringData dataset. The code is as follows: train$has\_BackupCamera = factor(train$has\_BackupCamera, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE')). Again, I repeated this step for the test and scoringData datasets.

In hindsight, it would have been better to do a linear regression with categorical variables to identify which of the features in the major\_options column (after parsing) would be the best predictors of price. The columns I chose to create reflected the largest 10-15 features which I was able to deduce when I created distinct features using dplyr features. I now realize that greater occurrence of a feature in the data is not necessarily a higher predictor or price. Please refer to the code on page 6 to see how the features were identified and arranged in ascending order.

Imputing missing data

Initially I tried to do a liner regression but found out that I could not do this when data was missing from columns. Hence, I used the ‘mice’ package to impute missing data. This imputed numerical variables in the data set. I then encoded categorical variables in the dataset by changing all values that had NA to ‘Did Not Respond’.

Imputing the data was highly effective in that I was able to conduct a linear regression with multiple variables and identify the data that was significantly and strongly correlated with price.

model7 = m(price~fuel\_tank\_volume\_gallons+highway\_fuel\_economy+city\_fuel\_economy+city\_fuel\_economy+wheelbase\_inches+back\_legroom\_inches+front\_legroom\_inches+length\_inches+width\_inches+height\_inches+engine\_displacement+horsepower+daysonmarket+maximum\_seating+year+mileage+owner\_count+seller\_rating+body\_type+power+powerRPM+torque+torqueRPM+transmission,data=train)

The variables that had strong and significant correlations with price informed my decision to include them in tree10 / the final decision tree model I used which yielded my personal best score.

I also followed the code in the feature selection section of classes to identify which of the numeric variables were correlated with price and which variables were strongly predicted with each other. Please refer to the code on page 19. It was surprising to find that wheelbase\_inches+back\_legroom\_inches+front\_legroom\_inches+length\_inches+width\_inches+height\_inches appeared to have an impact on the price of a car. Perhaps this is unconscious to buyers and is merely a coincidence associated with the body\_type of a car. I hence decided to include body\_type variables into my final decision tree.

Linear regression

It was my aim to do a linear regression with the following variables to get the lowest RMSE score:

model7 = lm(price~fuel\_tank\_volume\_gallons+highway\_fuel\_economy+city\_fuel\_economy+city\_fuel\_economy+wheelbase\_inches+back\_legroom\_inches+front\_legroom\_inches+length\_inches+width\_inches+height\_inches+engine\_displacement+horsepower+daysonmarket+maximum\_seating+year+mileage+owner\_count+seller\_rating+body\_type+fuel\_type+power+ powerRPM+torque+torqueRPM+transmission+trim\_name+make\_name+model\_name,data=train)

However, due to lack of expertise, experience and time it was not feasible for the PAC competition and for this class. A summary of the above model showed a multiple r-squared value of 0.964 which in my view would have yielded good results. Unfortunately, I was not able to predict or submit this model due to multiple errors associated with the factor levels. An example of the error I got was Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = attr(object, :

factor power has new levels 298, 357, 386, 464, 530, 536, 542, 543, 555, 556, 601, 616, 626, which I was not able to resolve with my current knowledge and experience. On a positive note, I was able to identify variables that would potentially be good predictors of price.

Decision Tree

I decided to use a decision tree for my final model which yielded the best results. I submitted about 18 entries of decision tree – mainly trials and errors involving multiple variables. I included in the decision tree the variables factor selection and linear regression had enabled me to identify as potential effective predictors of price. This yielded the lowest RMSE score for me. I played around with the variables and predict(treex), summary(treex) code and treex$variable.importance which I found extremely useful in evaluating my model. I also experimented with different cp level until I got the lowest result.

Personally, the PAC Competition was an immensely challenging yet rewarding experience for me. I am completely new to coding, and this class was my first exposure to R and coding. While I understood the theory in class, this competition enabled me to put into practice the code we had learnt in class and apply it to a completely new dataset. This enabled me to gain confidence in using this code beyond completing assignments in class.

I made many errors throughout the journey which are learnings for my coding journey. I faced challenges submitting the data correctly. There was a lot of back and forth involving modifying the code to best suit the purpose and lower the RMSE score. The process required persistence and patience. I am glad I partook in it because I’m now able to confidently say that I know how to code and if given a dataset in a workplace, I’ll be able to work my way around it, much faster this time.

**Code**

**Reading in the data**

setwd('/Users/anshitathakkar/Documents/Predictive Analytics/PAC')

usedCars\_decisiontree = read.csv('/Users/anshitathakkar/Documents/Predictive Analytics/PAC/analysisData.csv')

scoringData = read.csv('/Users/anshitathakkar/Documents/Predictive Analytics/PAC/scoringData.csv')

**Explore**

summary(usedCars\_decisiontree)

head(usedCars\_decisiontree)

nrow(usedCars\_decisiontree)

ncol(usedCars\_decisiontree)

nrow(scoringData)

ncol(scoringData)

**Data tidying/wrangling – analysis and scoring data**

***library(dplyr); library(readr); library(forcats); library(tidyr)***

usedCars\_decsiontree2 =

usedCars\_decisiontree%>%

separate(col = power, into = c('power', 'powerRPM'), sep = '@')%>%

mutate(power = parse\_number(power))%>%

mutate(powerRPM = parse\_number(powerRPM))%>%

separate(col = torque, into = c('torque', 'torqueRPM'), sep = '@')%>%

mutate(torque = parse\_number(torque))%>%

mutate(torqueRPM = parse\_number(torqueRPM))

usedCars\_decsiontree2

***Tidying major\_options column***

major\_options = toString(usedCars\_decsiontree2$major\_options)

updated = gsub("\\[|\\]", "", major\_options)

updated2 = gsub("\\'|\\'", "", updated)

major\_options = data.frame(strsplit(updated2, split=", "),stringsAsFactors = TRUE)

major\_options

head(major\_options)

#all\_genre

colnames(major\_options) = c('feature')

distinct\_features = major\_options |>

group\_by(feature) |>

summarise(count = n()) |>

mutate(Freq = count/sum(count)) |>

arrange(desc(count)) |>

ungroup()

distinct\_features

***Parsing scoring data***

***library(dplyr)***

***library(forcats); library(stringr); library(tidyr); library(readr)***

scoringData =

scoringData%>%

separate(col = power, into = c('power','powerRPM'), sep = '@')%>%

mutate(power = parse\_number(power))%>%

mutate(powerRPM = parse\_number(powerRPM))%>%

separate(col = torque, into = c('torque', 'torqueRPM'), sep = '@')%>%

mutate(torque = parse\_number(torque))%>%

mutate(torqueRPM = parse\_number(torqueRPM))

scoringData

**Splitting the data**

***library(caret)***

set.seed(2792)

split = createDataPartition(y = usedCars\_decsiontree2$price, p = 0.75, list = F,groups = 10)

train = usedCars\_decsiontree2[split,]

test = usedCars\_decsiontree2[-split,]

mean(train$price)

mean(test$price)

***Adding columns for analysis***

**Train data**

***library(stringr)***

train$has\_BackupCamera = ifelse(str\_detect(string = train$major\_options,pattern = 'Backup Camera'),TRUE,FALSE)

train$has\_Bluetooth = ifelse(str\_detect(string = train$major\_options,pattern = 'Bluetooth'),TRUE,FALSE)

train$has\_AlloyWheels = ifelse(str\_detect(string = train$major\_options,pattern = 'Alloy Wheels'),TRUE,FALSE)

train$has\_HeatedSeats = ifelse(str\_detect(string = train$major\_options,pattern = 'Heated Seats'),TRUE,FALSE)

train$has\_NavigationSystem = ifelse(str\_detect(string = train$major\_options,pattern = 'Navigation System'),TRUE,FALSE)

train$has\_Sunroof\_Moonroof = ifelse(str\_detect(string = train$major\_options,pattern = 'Sunroof/Moonroof'),TRUE,FALSE)

train$has\_RemoteStart = ifelse(str\_detect(string = train$major\_options,pattern = 'Remote Start'),TRUE,FALSE)

train$has\_CarPlay = ifelse(str\_detect(string = train$major\_options,pattern = 'CarPlay'),TRUE,FALSE)

train$has\_BlindSpotMonitoring = ifelse(str\_detect(string = train$major\_options,pattern = 'Blind Spot Monitoring'),TRUE,FALSE)

train$has\_LeatherSeats = ifelse(str\_detect(string = train$major\_options,pattern = 'Leather Seats'),TRUE,FALSE)

train$has\_AndroidAuto = ifelse(str\_detect(string = train$major\_options,pattern = 'Android Auto'),TRUE,FALSE)

train$has\_ParkingSensors = ifelse(str\_detect(string = train$major\_options,pattern = 'Parking Sensors'),TRUE,FALSE)

train$has\_ThirdRowSeating = ifelse(str\_detect(string = train$major\_options,pattern = 'Third Row Seating'),TRUE,FALSE)

train$has\_SteelWheels = ifelse(str\_detect(string = train$major\_options,pattern = 'Steel Wheels'),TRUE,FALSE)

train$has\_QuickOrderPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Quick Order Package'),TRUE,FALSE)

train$has\_PremiumPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Premium Package'),TRUE,FALSE)

train$has\_ConveniencePackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Convenience Package'),TRUE,FALSE)

train$has\_PowerPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Power Package'),TRUE,FALSE)

train$has\_TowPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Tow Package'),TRUE,FALSE)

train$has\_MultiZoneClimateControl = ifelse(str\_detect(string = train$major\_options,pattern = 'Multi Zone Climate Control'),TRUE,FALSE)

train$has\_AppearancePackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Appearance Package'),TRUE,FALSE)

train$has\_TechnologyPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Technology Package'),TRUE,FALSE)

train$has\_PreferredPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Preferred Package'),TRUE,FALSE)

train$has\_HeatPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Heat Package'),TRUE,FALSE)

train$has\_SEPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'SE Package'),TRUE,FALSE)

train$has\_PremiumWheels = ifelse(str\_detect(string = train$major\_options,pattern = 'Premium Wheels'),TRUE,FALSE)

train$has\_SportPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Sport Package'),TRUE,FALSE)

train$has\_ColdWeatherPackage = ifelse(str\_detect(string = train$major\_options,pattern = 'Cold Weather Package'),TRUE,FALSE)

***Making train data - new columns into factor***

```{r}

train$has\_BackupCamera = factor(train$has\_BackupCamera, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_Bluetooth = factor(train$has\_Bluetooth, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_AlloyWheels = factor(train$has\_AlloyWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_HeatedSeats = factor(train$has\_HeatedSeats, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_NavigationSystem = factor(train$has\_NavigationSystem, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_Sunroof\_Moonroof = factor(train$has\_Sunroof\_Moonroof, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_RemoteStart = factor(train$has\_RemoteStart, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_CarPlay = factor(train$has\_CarPlay, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_BlindSpotMonitoring = factor(train$has\_BlindSpotMonitoring, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_LeatherSeats = factor(train$has\_LeatherSeats, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_AndroidAuto = factor(train$has\_AndroidAuto, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_ParkingSensors = factor(train$has\_ParkingSensors, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_ThirdRowSeating = factor(train$has\_ThirdRowSeating, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_SteelWheels = factor(train$has\_SteelWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_QuickOrderPackage = factor(train$has\_QuickOrderPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_PremiumPackage = factor(train$has\_PremiumPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_ConveniencePackage = factor(train$has\_ConveniencePackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_PowerPackage = factor(train$has\_PowerPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_TowPackage = factor(train$has\_TowPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_MultiZoneClimateControl = factor(train$has\_MultiZoneClimateControl, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_AppearancePackage = factor(train$has\_AppearancePackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_TechnologyPackage = factor(train$has\_TechnologyPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_PreferredPackage = factor(train$has\_PreferredPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_HeatPackage = factor(train$has\_HeatPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_SEPackage = factor(train$has\_SEPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_PremiumWheels = factor(train$has\_PremiumWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_SportPackage = factor(train$has\_SportPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$has\_ColdWeatherPackage = factor(train$has\_ColdWeatherPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

***Repeating step with test data***

test$has\_BackupCamera = ifelse(str\_detect(string = test$major\_options,pattern = 'Backup Camera'),TRUE,FALSE)

test$has\_Bluetooth = ifelse(str\_detect(string = test$major\_options,pattern = 'Bluetooth'),TRUE,FALSE)

test$has\_AlloyWheels = ifelse(str\_detect(string = test$major\_options,pattern = 'Alloy Wheels'),TRUE,FALSE)

test$has\_HeatedSeats = ifelse(str\_detect(string = test$major\_options,pattern = 'Heated Seats'),TRUE,FALSE)

test$has\_NavigationSystem = ifelse(str\_detect(string = test$major\_options,pattern = 'Navigation System'),TRUE,FALSE)

test$has\_Sunroof\_Moonroof = ifelse(str\_detect(string = test$major\_options,pattern = 'Sunroof/Moonroof'),TRUE,FALSE)

test$has\_RemoteStart = ifelse(str\_detect(string = test$major\_options,pattern = 'Remote Start'),TRUE,FALSE)

test$has\_CarPlay = ifelse(str\_detect(string = test$major\_options,pattern = 'CarPlay'),TRUE,FALSE)

test$has\_BlindSpotMonitoring = ifelse(str\_detect(string = test$major\_options,pattern = 'Blind Spot Monitoring'),TRUE,FALSE)

test$has\_LeatherSeats = ifelse(str\_detect(string = test$major\_options,pattern = 'Leather Seats'),TRUE,FALSE)

test$has\_AndroidAuto = ifelse(str\_detect(string = test$major\_options,pattern = 'Android Auto'),TRUE,FALSE)

test$has\_ParkingSensors = ifelse(str\_detect(string = test$major\_options,pattern = 'Parking Sensors'),TRUE,FALSE)

test$has\_ThirdRowSeating = ifelse(str\_detect(string = test$major\_options,pattern = 'Third Row Seating'),TRUE,FALSE)

test$has\_SteelWheels = ifelse(str\_detect(string = test$major\_options,pattern = 'Steel Wheels'),TRUE,FALSE)

test$has\_QuickOrderPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Quick Order Package'),TRUE,FALSE)

test$has\_PremiumPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Premium Package'),TRUE,FALSE)

test$has\_ConveniencePackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Convenience Package'),TRUE,FALSE)

test$has\_PowerPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Power Package'),TRUE,FALSE)

test$has\_TowPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Tow Package'),TRUE,FALSE)

test$has\_MultiZoneClimateControl = ifelse(str\_detect(string = test$major\_options,pattern = 'Multi Zone Climate Control'),TRUE,FALSE)

test$has\_AppearancePackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Appearance Package'),TRUE,FALSE)

test$has\_TechnologyPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Technology Package'),TRUE,FALSE)

test$has\_PreferredPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Preferred Package'),TRUE,FALSE)

test$has\_HeatPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Heat Package'),TRUE,FALSE)

test$has\_SEPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'SE Package'),TRUE,FALSE)

test$has\_PremiumWheels = ifelse(str\_detect(string = test$major\_options,pattern = 'Premium Wheels'),TRUE,FALSE)

test$has\_SportPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Sport Package'),TRUE,FALSE)

test$has\_ColdWeatherPackage = ifelse(str\_detect(string = test$major\_options,pattern = 'Cold Weather Package'),TRUE,FALSE)

test

***Test data factors***

```{r}

test$has\_BackupCamera = factor(test$has\_BackupCamera, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_Bluetooth = factor(test$has\_Bluetooth, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_AlloyWheels = factor(test$has\_AlloyWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_HeatedSeats = factor(test$has\_HeatedSeats, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_NavigationSystem = factor(test$has\_NavigationSystem, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_Sunroof\_Moonroof = factor(test$has\_Sunroof\_Moonroof, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_RemoteStart = factor(test$has\_RemoteStart, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_CarPlay = factor(test$has\_CarPlay, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_BlindSpotMonitoring = factor(test$has\_BlindSpotMonitoring, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_LeatherSeats = factor(test$has\_LeatherSeats, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_AndroidAuto = factor(test$has\_AndroidAuto, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_ParkingSensors = factor(test$has\_ParkingSensors, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_ThirdRowSeating = factor(test$has\_ThirdRowSeating, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_SteelWheels = factor(test$has\_SteelWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_QuickOrderPackage = factor(test$has\_QuickOrderPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_PremiumPackage = factor(test$has\_PremiumPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_ConveniencePackage = factor(test$has\_ConveniencePackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_PowerPackage = factor(test$has\_PowerPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_TowPackage = factor(test$has\_TowPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_MultiZoneClimateControl = factor(test$has\_MultiZoneClimateControl, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_AppearancePackage = factor(test$has\_AppearancePackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_TechnologyPackage = factor(test$has\_TechnologyPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_PreferredPackage = factor(test$has\_PreferredPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_HeatPackage = factor(test$has\_HeatPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_SEPackage = factor(test$has\_SEPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_PremiumWheels = factor(test$has\_PremiumWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_SportPackage = factor(test$has\_SportPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$has\_ColdWeatherPackage = factor(test$has\_ColdWeatherPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

***Creating columns in scoring data***

```{r}

scoringData$has\_BackupCamera = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Backup Camera'),TRUE,FALSE)

scoringData$has\_Bluetooth = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Bluetooth'),TRUE,FALSE)

scoringData$has\_AlloyWheels = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Alloy Wheels'),TRUE,FALSE)

scoringData$has\_HeatedSeats = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Heated Seats'),TRUE,FALSE)

scoringData$has\_NavigationSystem = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Navigation System'),TRUE,FALSE)

scoringData$has\_Sunroof\_Moonroof = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Sunroof/Moonroof'),TRUE,FALSE)

scoringData$has\_RemoteStart = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Remote Start'),TRUE,FALSE)

scoringData$has\_CarPlay = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'CarPlay'),TRUE,FALSE)

scoringData$has\_BlindSpotMonitoring = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Blind Spot Monitoring'),TRUE,FALSE)

scoringData$has\_LeatherSeats = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Leather Seats'),TRUE,FALSE)

scoringData$has\_AndroidAuto = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Android Auto'),TRUE,FALSE)

scoringData$has\_ParkingSensors = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Parking Sensors'),TRUE,FALSE)

scoringData$has\_ThirdRowSeating = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Third Row Seating'),TRUE,FALSE)

scoringData$has\_SteelWheels = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Steel Wheels'),TRUE,FALSE)

scoringData$has\_QuickOrderPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Quick Order Package'),TRUE,FALSE)

scoringData$has\_PremiumPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Premium Package'),TRUE,FALSE)

scoringData$has\_ConveniencePackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Convenience Package'),TRUE,FALSE)

scoringData$has\_PowerPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Power Package'),TRUE,FALSE)

scoringData$has\_TowPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Tow Package'),TRUE,FALSE)

scoringData$has\_MultiZoneClimateControl = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Multi Zone Climate Control'),TRUE,FALSE)

scoringData$has\_AppearancePackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Appearance Package'),TRUE,FALSE)

scoringData$has\_TechnologyPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Technology Package'),TRUE,FALSE)

scoringData$has\_PreferredPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Preferred Package'),TRUE,FALSE)

scoringData$has\_HeatPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Heat Package'),TRUE,FALSE)

scoringData$has\_SEPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'SE Package'),TRUE,FALSE)

scoringData$has\_PremiumWheels = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Premium Wheels'),TRUE,FALSE)

scoringData$has\_SportPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Sport Package'),TRUE,FALSE)

scoringData$has\_ColdWeatherPackage = ifelse(str\_detect(string = scoringData$major\_options,pattern = 'Cold Weather Package'),TRUE,FALSE)

scoringData

***ScoringData --- create factors***

```{r}

scoringData$has\_BackupCamera = factor(scoringData$has\_BackupCamera, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_Bluetooth = factor(scoringData$has\_Bluetooth, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_AlloyWheels = factor(scoringData$has\_AlloyWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_HeatedSeats = factor(scoringData$has\_HeatedSeats, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_NavigationSystem = factor(scoringData$has\_NavigationSystem, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_Sunroof\_Moonroof = factor(scoringData$has\_Sunroof\_Moonroof, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_RemoteStart = factor(scoringData$has\_RemoteStart, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_CarPlay = factor(scoringData$has\_CarPlay, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_BlindSpotMonitoring = factor(scoringData$has\_BlindSpotMonitoring, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_LeatherSeats = factor(scoringData$has\_LeatherSeats, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_AndroidAuto = factor(scoringData$has\_AndroidAuto, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_ParkingSensors = factor(scoringData$has\_ParkingSensors, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_ThirdRowSeating = factor(scoringData$has\_ThirdRowSeating, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_SteelWheels = factor(scoringData$has\_SteelWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_QuickOrderPackage = factor(scoringData$has\_QuickOrderPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_PremiumPackage = factor(scoringData$has\_PremiumPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_ConveniencePackage = factor(scoringData$has\_ConveniencePackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_PowerPackage = factor(scoringData$has\_PowerPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_TowPackage = factor(scoringData$has\_TowPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_MultiZoneClimateControl = factor(scoringData$has\_MultiZoneClimateControl, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_AppearancePackage = factor(scoringData$has\_AppearancePackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_TechnologyPackage = factor(scoringData$has\_TechnologyPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_PreferredPackage = factor(scoringData$has\_PreferredPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_HeatPackage = factor(scoringData$has\_HeatPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_SEPackage = factor(scoringData$has\_SEPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_PremiumWheels = factor(scoringData$has\_PremiumWheels, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_SportPackage = factor(scoringData$has\_SportPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$has\_ColdWeatherPackage = factor(scoringData$has\_ColdWeatherPackage, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

```

***Creating body\_type columns in train, test and scoringData***

```{r}

train$isCoupe = ifelse(str\_detect(string = train$body\_type,pattern = 'Coupe'),TRUE,FALSE)

train$isHatchback = ifelse(str\_detect(string = train$body\_type,pattern = 'Hatchback'),TRUE,FALSE)

train$isMinivan = ifelse(str\_detect(string = train$body\_type,pattern = 'Minivan'),TRUE,FALSE)

train$isPickupTruck = ifelse(str\_detect(string = train$body\_type,pattern = 'Pickup Truck'),TRUE,FALSE)

train$isSedan = ifelse(str\_detect(string = train$body\_type,pattern = 'Sedan'),TRUE,FALSE)

train$isSUV = ifelse(str\_detect(string = train$body\_type,pattern = 'SUV / Crossover'),TRUE,FALSE)

train$isVan = ifelse(str\_detect(string = train$body\_type,pattern = 'Van'),TRUE,FALSE)

train$isWagon = ifelse(str\_detect(string = train$body\_type,pattern = 'Wagon'),TRUE,FALSE)

train$isCoupe = factor(train$isCoupe, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$isHatchback = factor(train$isHatchback, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$isMinivan = factor(train$isMinivan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$isPickupTruck = factor(train$isPickupTruck, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$isSedan = factor(train$isSedan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$isSUV = factor(train$isSUV, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$isVan = factor(train$isVan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train$isWagon = factor(train$isWagon, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

train

test$isCoupe = ifelse(str\_detect(string = test$body\_type,pattern = 'Coupe'),TRUE,FALSE)

test$isHatchback = ifelse(str\_detect(string = test$body\_type,pattern = 'Hatchback'),TRUE,FALSE)

test$isMinivan = ifelse(str\_detect(string = test$body\_type,pattern = 'Minivan'),TRUE,FALSE)

test$isPickupTruck = ifelse(str\_detect(string = test$body\_type,pattern = 'Pickup Truck'),TRUE,FALSE)

test$isSedan = ifelse(str\_detect(string = test$body\_type,pattern = 'Sedan'),TRUE,FALSE)

test$isSUV = ifelse(str\_detect(string = test$body\_type,pattern = 'SUV / Crossover'),TRUE,FALSE)

test$isVan = ifelse(str\_detect(string = test$body\_type,pattern = 'Van'),TRUE,FALSE)

test$isWagon = ifelse(str\_detect(string = test$body\_type,pattern = 'Wagon'),TRUE,FALSE)

test$isCoupe = factor(test$isCoupe, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$isHatchback = factor(test$isHatchback, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$isMinivan = factor(test$isMinivan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$isPickupTruck = factor(test$isPickupTruck, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$isSedan = factor(test$isSedan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$isSUV = factor(test$isSUV, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$isVan = factor(test$isVan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test$isWagon = factor(test$isWagon, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

test

scoringData$isCoupe = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'Coupe'),TRUE,FALSE)

scoringData$isHatchback = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'Hatchback'),TRUE,FALSE)

scoringData$isMinivan = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'Minivan'),TRUE,FALSE)

scoringData$isPickupTruck = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'Pickup Truck'),TRUE,FALSE)

scoringData$isSedan = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'Sedan'),TRUE,FALSE)

scoringData$isSUV = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'SUV / Crossover'),TRUE,FALSE)

scoringData$isVan = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'Van'),TRUE,FALSE)

scoringData$isWagon = ifelse(str\_detect(string = scoringData$body\_type,pattern = 'Wagon'),TRUE,FALSE)

scoringData$isCoupe = factor(scoringData$isCoupe, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$isHatchback = factor(scoringData$isHatchback, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$isMinivan = factor(scoringData$isMinivan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$isPickupTruck = factor(scoringData$isPickupTruck, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$isSedan = factor(scoringData$isSedan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$isSUV = factor(scoringData$isSUV, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringDataisVan = factor(scoringData$isVan, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData$isWagon = factor(scoringData$isWagon, levels = c(FALSE,TRUE), labels = c('FALSE','TRUE'))

scoringData

***Skimming the data***

library(skimr)

skim(train)

summary(train)

***DETERMINING WHICH VARIABLES TO EXPLORE***

This step took place before splitting the data into train and test

#Step 1 impute data for linear regression

usedCars\_missing = usedCars\_16

for(i in 1:45){

set.seed(i)

x = sample(1:nrow(usedCars), 1)

y = sample(1:ncol(usedCars), 1)

usedCars\_missing[x, y] = NA

}

usedCars\_missing

usedCars\_missing = scoringData

for(i in 1:45){

set.seed(i)

x = sample(1:nrow(usedCars), 1)

y = sample(1:ncol(usedCars), 1)

usedCars\_missing[x, y] = NA

}

scoringData

usedCars\_missing\_bi = usedCars\_missing

usedCars\_missing\_bi[!is.na(usedCars\_missing)] = 1

usedCars\_missing\_bi[is.na(usedCars\_missing)] = 0

scoringData\_bi = scoringData

scoringData\_bi[!is.na(scoringData)] = 1

scoringData\_bi[is.na(scoringData)] = 0

apply(usedCars\_missing,

MARGIN = 2,

FUN = function(x) 100\*sum(is.na(x))/(sum(is.na(x))+ sum(!is.na(x))))

apply(scoringData,

MARGIN = 2,

FUN = function(x) 100\*sum(is.na(x))/(sum(is.na(x))+ sum(!is.na(x))))

#mice package

library(mice)

usedCars16\_imputed = mice::complete(mice(usedCars\_16, method = "mean", seed = 2267))

# Perform mean imputation on scoringdata using mice package

scoringData = mice::complete(mice(scoringData, method = "mean", seed = 2267))

**# Specify the columns with missing data**

columns\_to\_fill <- c("make\_name", "model\_name", "trim\_name", "body\_type", "fuel\_type", "power", "powerRPM", "torque", "torqueRPM", "transmission",

"transmission\_display", "wheel\_system", "wheel\_system\_display", "engine\_type", "description",

"exterior\_color", "interior\_color", "major\_options", "fleet", "frame\_damaged", "franchise\_dealer",

"franchise\_make", "has\_accidents", "isCab", "is\_cpo", "is\_new", "listed\_date", "listing\_color", "salvage")

**# Fill empty values with 'Did Not Respond' for each column**

for (col in columns\_to\_fill) {

usedCars16\_imputed[[col]][usedCars16\_imputed[[col]] == ""] <- 'Did Not Respond'}

columns\_to\_fill <- c("make\_name", "model\_name", "trim\_name", "body\_type", "fuel\_type", "power", "powerRPM", "torque", "torqueRPM", "transmission",

"transmission\_display", "wheel\_system", "wheel\_system\_display", "engine\_type", "description",

"exterior\_color", "interior\_color", "major\_options", "fleet", "frame\_damaged", "franchise\_dealer",

"franchise\_make", "has\_accidents", "isCab", "is\_cpo", "is\_new", "listed\_date", "listing\_color", "salvage")

**# Fill empty values with 'Did Not Respond' for each column**

for (col in columns\_to\_fill) {

scoringData[[col]][scoringData[[col]] == ""] <- 'Did Not Respond'

}

for (col in columns\_to\_fill) {

scoringData[[col]][scoringData[[col]] == ""] <- 'Did Not Respond'

}

skim(scoringData)

***Linear regression:***

model7 = lm(price~fuel\_tank\_volume\_gallons+highway\_fuel\_economy+city\_fuel\_economy+city\_fuel\_economy+wheelbase\_inches+back\_legroom\_inches+front\_legroom\_inches+length\_inches+width\_inches+height\_inches+engine\_displacement+horsepower+daysonmarket+maximum\_seating+year+mileage+owner\_count+seller\_rating+body\_type+power+ powerRPM+torque+torqueRPM+transmission,data=train)

summary(model7)

pred7 = predict(model7)

sse7 = sum((pred7 - train$price)^2)

sst7 = sum((mean(train$price)-train$price)^2)

model7\_r2 = 1 - sse7/sst7; model7\_r2

rmse7 = sqrt(mean((pred7-train$price)^2)); rmse7

predtest\_7 = predict(model7, newdata=test). #this is where I got an error and I could not move forward. However, with a linear regression I was able to identify the vairables that strongly correlated with the selling price of a used car

sse7\_test = sum((predtest\_7 - test$price)^2)

sst7\_test = sum((mean(train$price)-test$price)^2)

model7\_r2\_test = 1 - sse7\_test/sst7\_test; model7\_r2\_test

rmse7\_test = sqrt(mean((predtest\_7-test$price)^2)); rmse7\_test

***Feature Selection***

I also used Feature selection to identify variables associated with the selling price of a used car.

str(train). # I identified the numeric variables

cor(train[, c(6, 8, 9, 16:21, 23:25, 30:31, 42:43, 45)])

round(cor(train[, c(6, 8, 9, 16:21, 23:25, 30:31, 42:43, 45)]), 2)\*100

library(tidyr); library(dplyr); library(ggplot2)

corMatrix = as.data.frame(cor(train[, c(6, 8, 9, 16:21, 23:25, 30:31, 42:43, 45)]))

corMatrix$var1 = rownames(corMatrix)

corMatrix %>%

gather(key=var2,value=r,1:11)%>%

arrange(var1,desc(var2))%>%

ggplot(aes(x=var1,y=reorder(var2, order(var2,decreasing=F)),fill=r))+

geom\_tile()+

geom\_text(aes(label=round(r,2)),size=3)+

scale\_fill\_gradientn(colours = c('#d7191c','#fdae61','#ffffbf','#a6d96a','#1a9641'))+

theme(axis.text.x=element\_text(angle=75,hjust = 1))+xlab('')+ylab('')

library(ggcorrplot)

ggcorrplot(cor(train[, c(6, 8, 9, 16:21, 23:25, 30:31, 42:43, 45)]),

method = 'square',

type = 'lower',

show.diag = F,

colors = c('#e9a3c9', '#f7f7f7', '#a1d76a'))

#Looking at the above visual models, I was able to identify which variables were strongly correlated and potential predictors of price.

***Creating a decision tree***

***library(rpart); library(rpart.plot)***

tree10 = rpart(price~is\_new+wheelbase\_inches+horsepower+daysonmarket+highway\_fuel\_economy+engine\_displacement+mileage+seller\_rating+owner\_count+year+maximum\_seating+fuel\_tank\_volume\_gallons+highway\_fuel\_economy+city\_fuel\_economy+back\_legroom\_inches+front\_legroom\_inches+length\_inches+width\_inches+height\_inches+has\_BackupCamera+has\_NavigationSystem+power+powerRPM+torque+torqueRPM+has\_BlindSpotMonitoring+has\_ConveniencePackage+has\_AndroidAuto+has\_ParkingSensors+has\_TechnologyPackage+has\_PreferredPackage+has\_HeatPackage+has\_SEPackage+has\_PremiumWheels+has\_SportPackage+has\_ColdWeatherPackage+isCoupe+isHatchback+isMinivan+isPickupTruck+isSedan+isSUV+isWagon,data = train, method = 'anova', cp = 0.000000001)

***Prediction***

predictions10 <- predict(tree10, newdata = scoringData)

summary(tree10)

tree10$variable.importance

horsepower fuel\_tank\_volume\_gallons mileage engine\_displacement wheelbase\_inches year

4.467052e+12 2.489654e+12 2.336568e+12 2.296585e+12 2.076999e+12 1.971369e+12

height\_inches length\_inches power is\_new width\_inches torque

1.732969e+12 1.612721e+12 1.497214e+12 1.386689e+12 1.162121e+12 9.820670e+11

torqueRPM front\_legroom\_inches has\_BackupCamera powerRPM back\_legroom\_inches has\_NavigationSystem

4.319904e+11 2.842283e+11 2.797362e+11 2.774524e+11 2.233783e+11 1.655892e+11

owner\_count maximum\_seating highway\_fuel\_economy city\_fuel\_economy isSUV daysonmarket

1.552461e+11 1.290410e+11 9.906072e+10 8.255709e+10 7.036654e+10 5.765978e+10

seller\_rating has\_PremiumWheels has\_BlindSpotMonitoring isPickupTruck has\_ParkingSensors isSedan

4.158341e+10 4.122827e+10 3.901048e+10 2.357911e+10 1.802673e+10 1.430723e+10

has\_HeatPackage has\_TechnologyPackage isCoupe has\_AndroidAuto has\_ConveniencePackage isMinivan

9.725740e+09 8.695070e+09 8.233654e+09 7.281802e+09 5.321765e+09 3.568947e+09

has\_SportPackage has\_PreferredPackage has\_SEPackage isHatchback isWagon has\_ColdWeatherPackage

2.283266e+09 1.815754e+09 1.732391e+09 1.670778e+09 1.533843e+09 1.189521e+09

RMSE: train and test

rmse\_train10 = sqrt(mean((predictions10 - train$price)^2))

pred10\_test = predict(tree10, newdata=test)

rmse\_test10 = sqrt(mean((pred10\_test - test$price)^2))

rmse\_train10

rmse\_test10

[1] 23726.39

[1] 4497.457 #theRMSE test is better

***Preparing csv for submission***

# ensure analysisData.csv and scoringData.csv are in your working directory

# following code will read data and construct a simple model

data = read.csv('/Users/anshitathakkar/Documents/Predictive Analytics/PAC/analysisData.csv')

model = predict(tree10, newdata = scoringData, type = 'vector')

# read in scoring data and apply model to generate predictions

predictions10 <- predict(tree10, newdata = scoringData)

# construct submission from predictions

submissionFile = data.frame(id = scoringData$id, price = predictions10)

write.csv(submissionFile, 'decisiontree\_multiple5.csv',row.names = F)

getwd()

***Unsuccessful Code***

model7 = lm(price~fuel\_tank\_volume\_gallons+highway\_fuel\_economy+city\_fuel\_economy+city\_fuel\_economy+wheelbase\_inches+back\_legroom\_inches+front\_legroom\_inches+length\_inches+width\_inches+height\_inches+engine\_displacement+horsepower+daysonmarket+maximum\_seating+year+mileage+owner\_count+seller\_rating+body\_type+fuel\_type+power+ powerRPM+torque+torqueRPM+transmission+trim\_name+make\_name+model\_name,data=train)

I was not able to make similar factor levels amongst the analysis and scoring data.