Capstone: Regression Approaches for MSRP Estimation

Andres Eduardo Osuna

6/21/2020

Contents

1. Executive Summary

E-commerce has taken a key role in today's supply chain and customer-employee interaction has been decreasing over the years. Nowadays, these type of business are expanding to many different new industries. Now, online car sales are growing fast. Less people want to go buy something in a store when they can do it from the comfort of their home. These businesses are able to price tag their cars, used and new, automatically. Furthermore, many customers may want to get a car quote online before going to the car dealer to buy it. They don't want to spend an hour of their time engaging on negotiations with the car salesman.

Price estimation is an important aspect of daily business. It is important to benchmark your prices against other competitors. In addition, offering good customer service with an e-commerce mindset and structure is highly competitive and challenging, but extremely rewarding. By gathering data from around the industry you can estimate optimal prices to gain market share, customer satisfaction, increase profit, and set your business as e-commerce success. This project used a dataset with manufacturer's suggested retail prices (MSRP) of over ten thousand cars of different brand, model, year, transmission type, horsepower, and more. All of these variables provide a way to estimate the MSRP for that particular car.

Different machine learning models were used to better estimate MSRP. This could be used to automatically calculate the MSRP, and give quotes depending on the car's characteristics and market segment. These models are called K-Nearest Neighbour Regression, Regression Trees and Random Forest. All three models gave above expectation results with an R^2 score of 0.89, 0.94 and 0.96, respectively. To set that into perspective, the maximum value is 1. Moreover, the median percent errors were 8.3%, 7% and 6.6%, respectively.These are outstanding results for the amount and varibaility of the observations gathered in the data. Other measurements like Mean Absolute value are also provided. This report consists in a data exploratory section, data visualization insights, data pre-processing, feature engineering, models development, results and conclusions.

2. Dataset Exploratory Analysis

Let's start by picking at the data and see what are some of the variables and levels that are contained within it:

	Make	Model	Transmission.Type	Number.of.Doors	Vehicle.Style	MSRP
1	BMW	1 Series M	MANUAL	2	Coupe	46135
263	Nissan	350Z	AUTOMATIC	2	Coupe	30600
583	FIAT	500L	AUTOMATIC	4	Wagon	22995
1005	Volvo	960	AUTOMATIC	4	Sedan	2000
4008	GMC	Envoy	AUTOMATIC	4	4dr SUV	26760
6325	Mitsubishi	Lancer Sportback	AUTOMATIC	4	4dr Hatchback	18395
8430	Acura	RDX	AUTOMATIC	4	4dr SUV	42020
10326	Ford	Tempo	MANUAL	4	Sedan	2000

This dataset contains two different types of variables: factor and integer class. As seen in the output, the dataset contains a total of 16 variables. In the output code below, the dataset variables are shown:

```
##  [1] "city.mpg"          "Driven_Wheels"     "Engine.Cylinders"
##  [4] "Engine.Fuel.Type"  "Engine.HP"         "highway.MPG"
##  [7] "Make"              "Market.Category"   "Model"
## [10] "MSRP"              "Number.of.Doors"   "Popularity"
## [13] "Transmission.Type" "Vehicle.Size"      "Vehicle.Style"
## [16] "Year"
```

By exploring the dataset, it was important to notice a group of observations with N/As. For simplicity, these few observations were removed.

```r
sum(is.na(data))
```

```
## [1] 105
```

Also, the code output below shows the new number of of rows and columns of the dataset after removing these.

```
data<- data %>% drop_na() #drop all NAs
nrow(data) #number of observations
```

```
## [1] 11815
```

```
ncol(data) #number of variables
```

```
## [1] 16
```

The output code below shows when the dataset was created. It has observations from 1990 to 2017.

```
min(data$Year) #data start
```

```
## [1] 1990
```

```
max(data$Year) #this dataset is from 2017
```

```
## [1] 2017
```

Some of the original features can be used to calculate further variables that simplify and summarize the information. The code below creates 2 new variables, fuel efficiency and car age. The new columns can be seen in the table below:

```
#So if we want to calculate an age variable we can do the following:
data_new<- data %>% mutate(age= max(data$Year)-Year)
#Fuel economy feature= weighted average between city mpg and highway mpg
data_new<- data_new %>% mutate(Fuel_eff= (city.mpg*0.55+ highway.MPG*0.45))
#This shows the two new variables
data_new[,17:18] %>% head()
```

```
##   age Fuel_eff
## 1   6    22.15
## 2   6    23.05
## 3   6    23.60
## 4   6    22.50
## 5   6    22.50
## 6   5    22.50
```

Additionally, all outliers were removed using the general rule of interquartiles. These data points could be considered as data noise, or in this case, as cars that are extremely rare. Therefore, there were no similar observations to use for MSRP predictions of these cars. Since some of the columns have so many different levels within it, the data was partitioned using an 80-20 approach to allow the test set to include a good amount of similar variability as the training set.

After removing outliers and partition the data into sets, this is the amount of observations inside each one:

Set	Rows	Columns
Train	8665	18
Test	2168	18

The focus will be now on the training set. The training data summary is shown below:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2000   20025   28410   28617   38200   74000
```

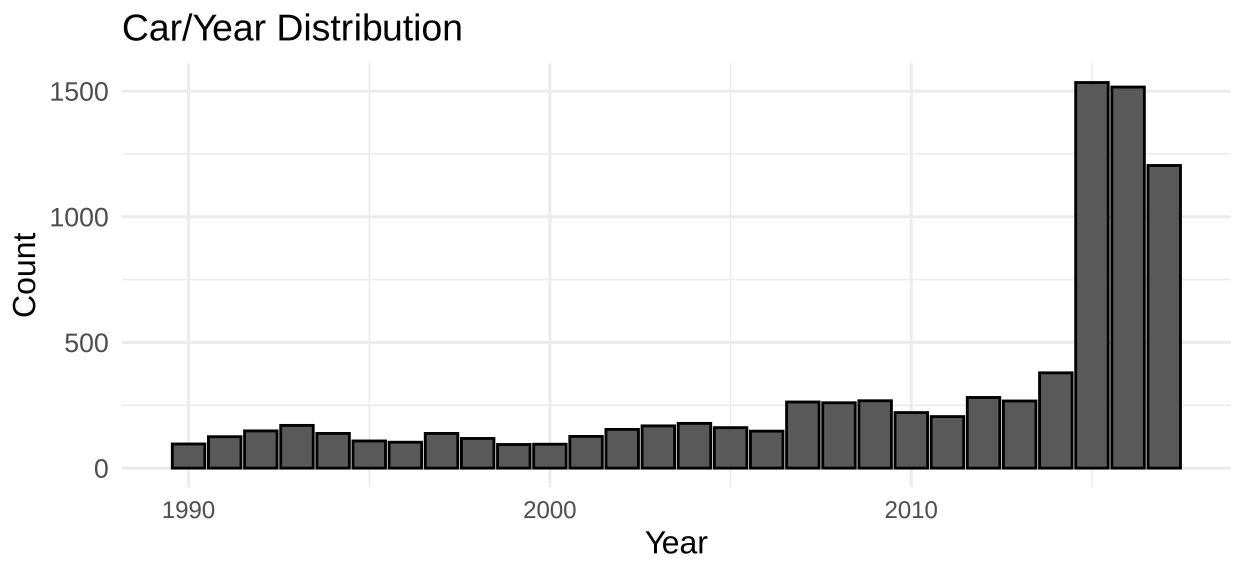
```
## [1] "There are a total of 38 unique car brands in our dataset"
```

Before moving on to the next phase of the report, it is important to make sure our data is clean and ready for use:

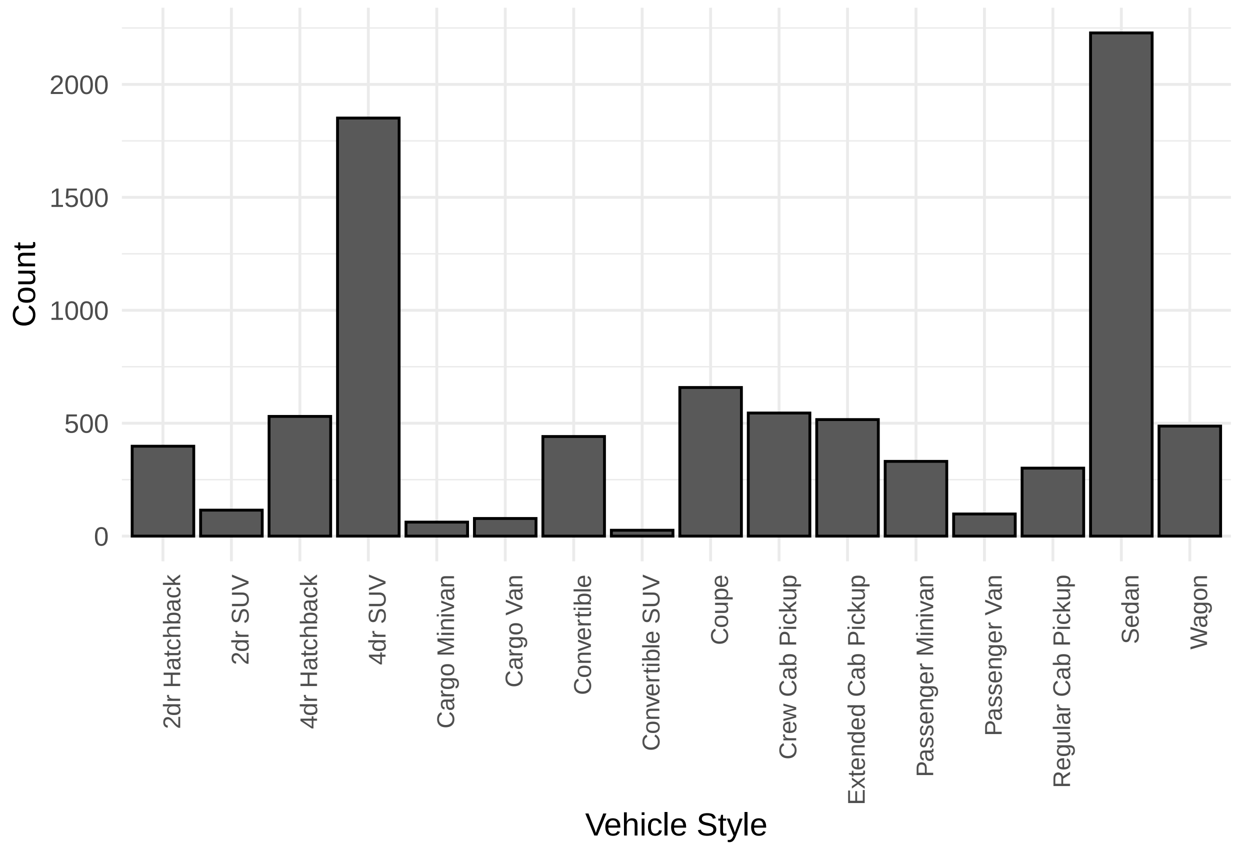
Variable	NAs
Make	0
Model	0
Year	0
Engine.Fuel.Type	0
Engine.HP	0
Engine.Cylinders	0
Transmission.Type	0
Driven_Wheels	0
Number.of.Doors	0
Market.Category	0
Vehicle.Size	0
Vehicle.Style	0
highway.MPG	0
city.mpg	0
Popularity	0
MSRP	0
age	0
Fuel_eff	0

3. Data Visualization Analysis

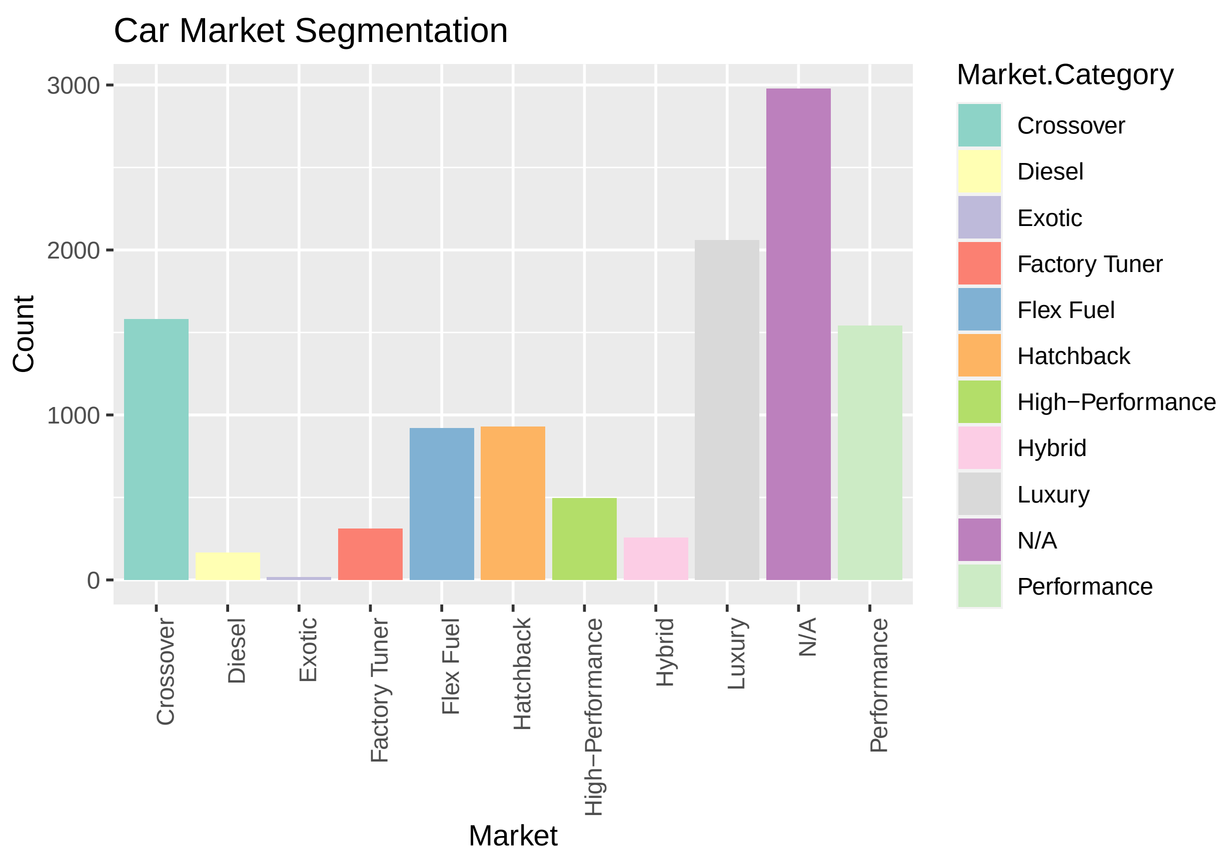
As mentioned in the section before, the dataset includes cars from all over the years (1990-2017). This graph shows the distribution of our data points with respect to how old they are. It is clear that most of the observations could be considered as recent cars.



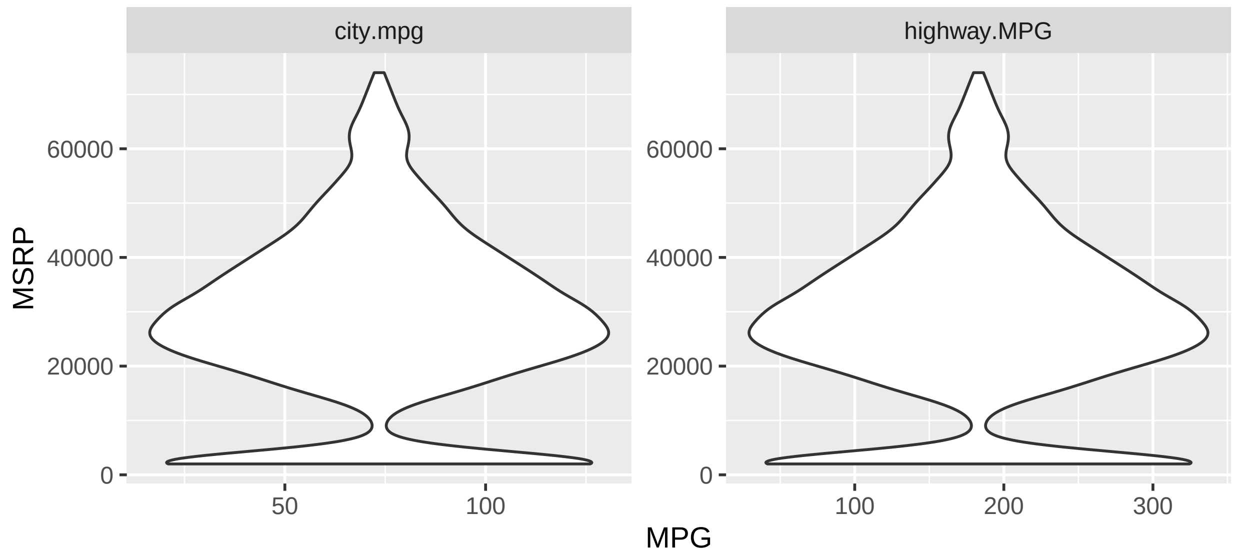
Within the dataset, there are different styles of vehicle. The majority of the cars in the dataset are considered to be sedans or 4-door SUVs. This makes sense considering that those are the two most common vechicle types in the market.



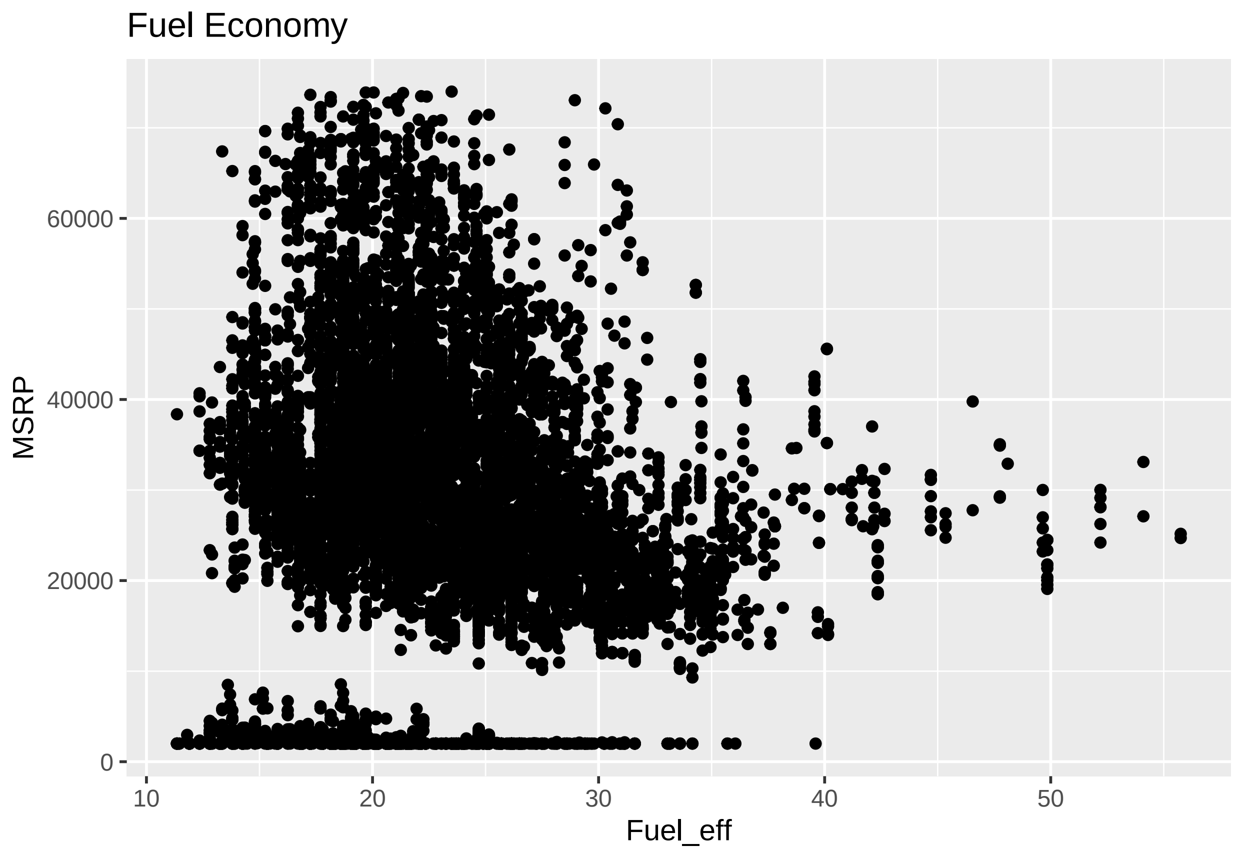
These vehicles compete in different types of markets. As seen in the figure below, the data suggests that there are 11 type of markets with the most common market being called "N/A". If you drill down into this group, the data shows that this market is actually composed by the average car, like for example, the Honda Civic. The second most prominent group is the Luxury style car.



The next graph below shows how the distribution of city.mpg (City-Miles per gallon) and highway.mpg (Highway-Miles per gallon) are pretty much the same with respect to MSRP. These variables do not seem to have a big effect on MSRP estimation. Although, maybe fuel efficiency, which was a computed variable dependent them, could have an impact on the target variable.

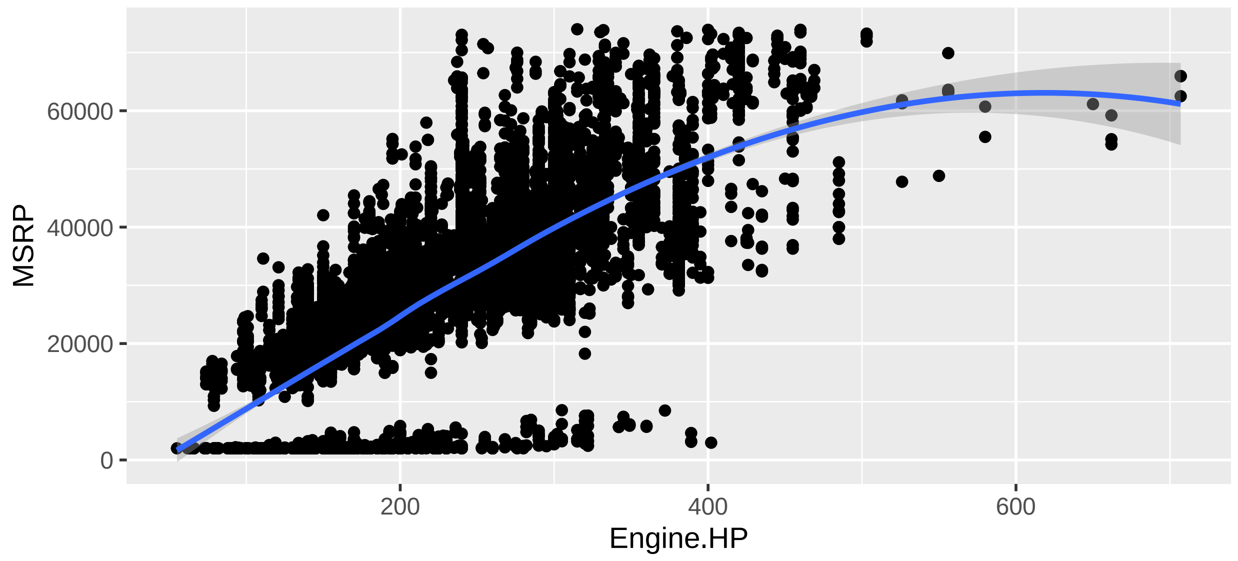


The graph below shows a very particular trend. Fuel efficiency seems to have some sort of relationship with price. It is not a clear and definite relationship, but further testing could confirm this. It seems that more expensive cars have less fuel economy efficieny levels. This makes sense with what it is known about the avgerage car industry.

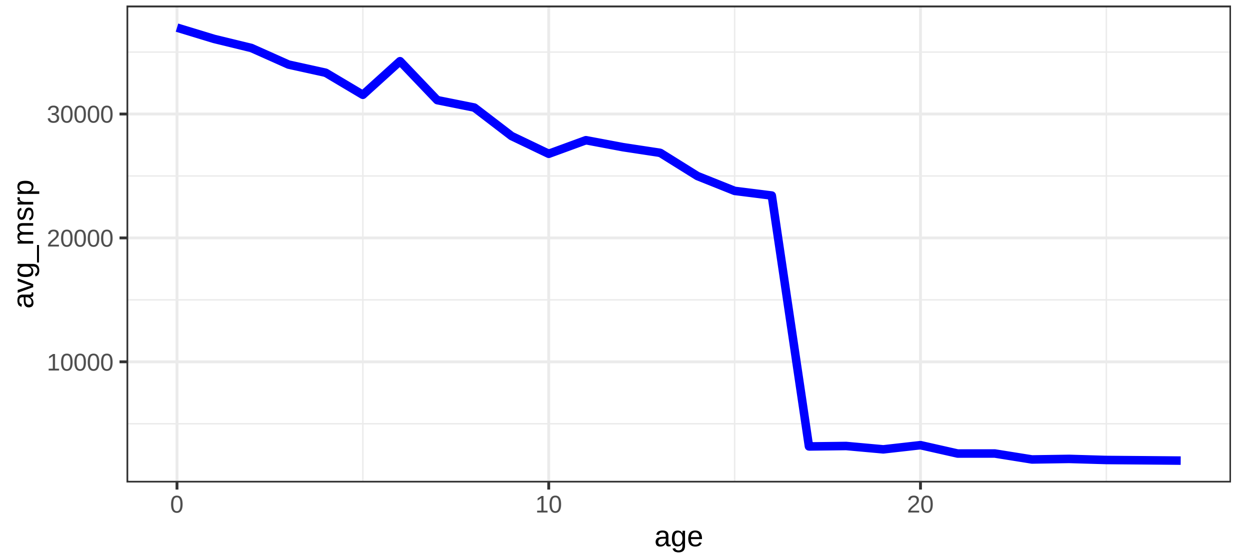


Horsepower seems to have a direct implication on the MSRP. The higher the the horsepower capacity, the higher the price. This also makes sense with what we know about the automobile industry. Sport cars, which tend to be faster, are always more expensive.

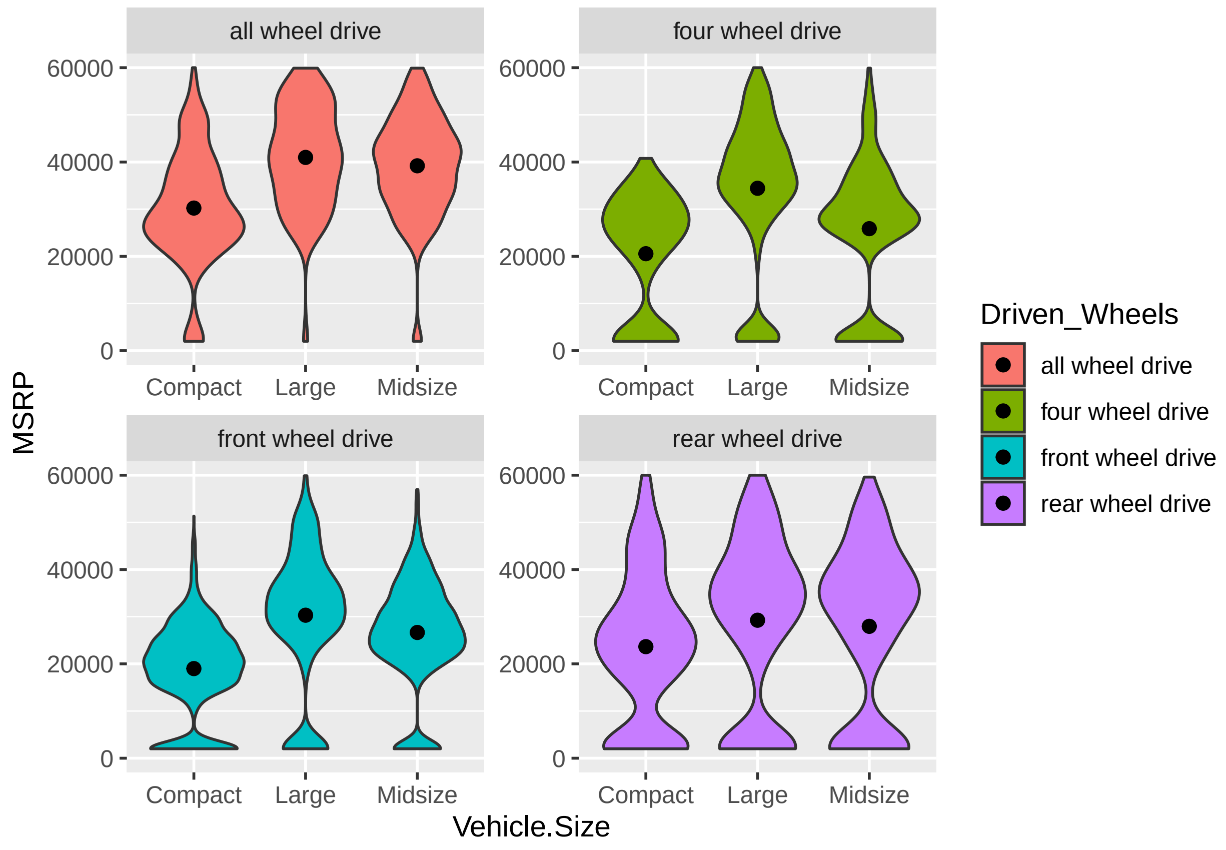
```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



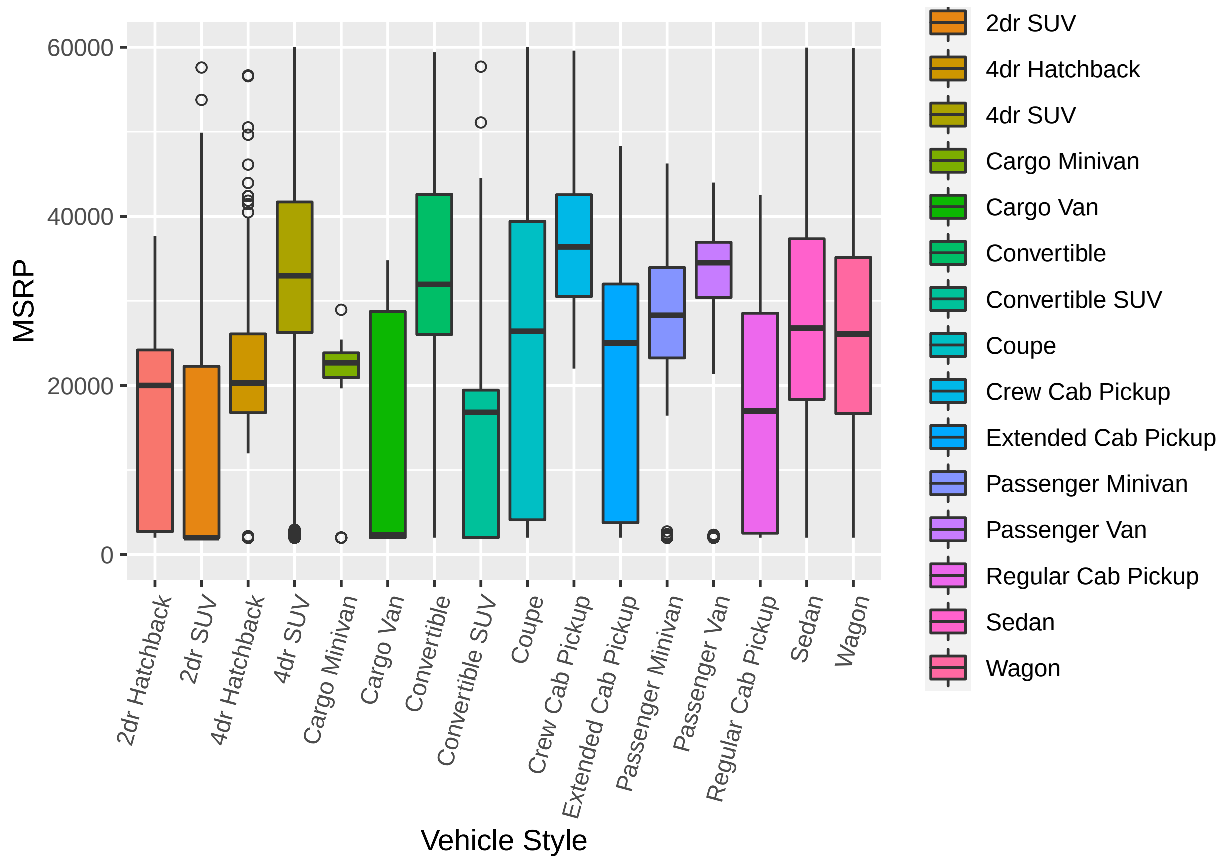
As mentioned before, the dataset contains cars from all over the years. It makes sense to think that the older the car, the lower value it has. As seen in the figure below, the expected negative trend between age and average MSRP is confirmed. It is important to highlight the strong drop in price after approximately 15 years.



From the graph below, the data shows how each type of Driven_Wheels variable affect the distribution of price by vehicle size. Notice that compact cars always have a lower mean than the other two vehicle sizes.

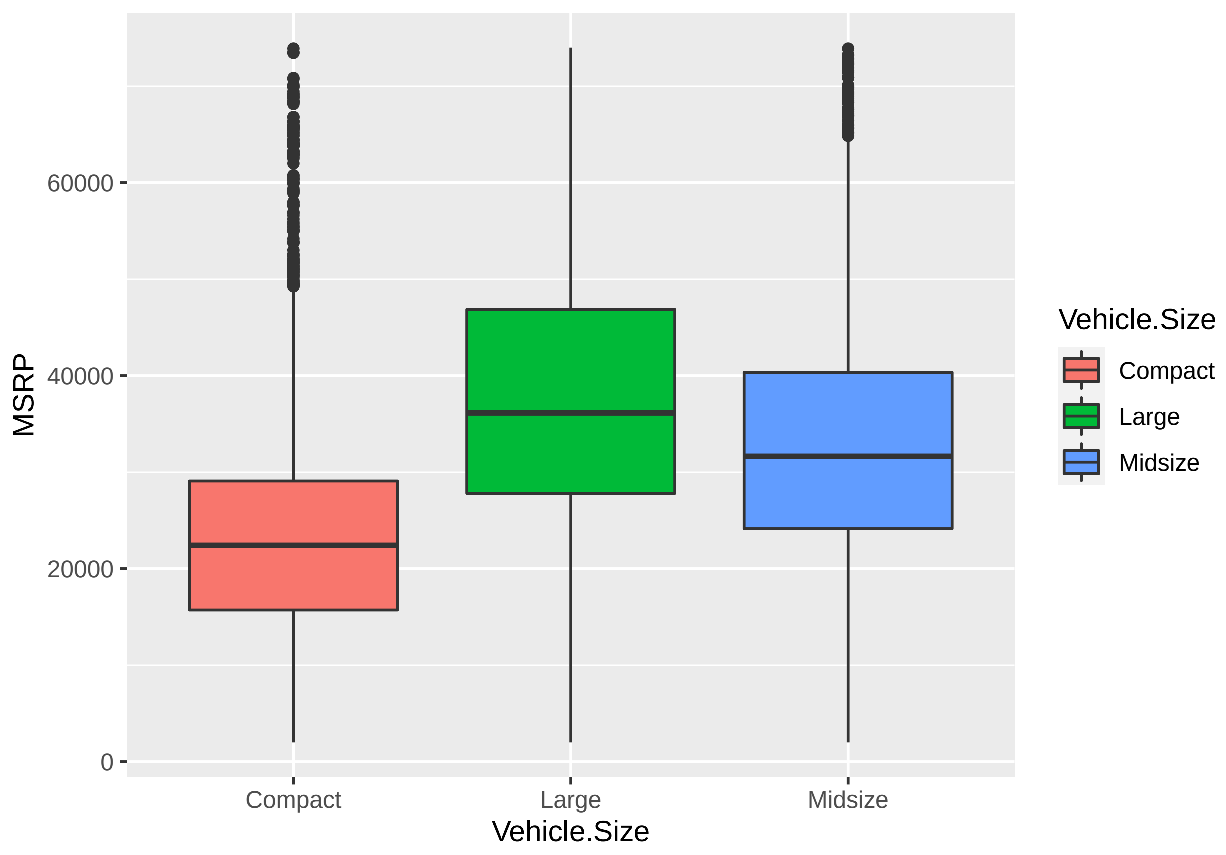


As observed in the graph below, the distribution of MSRP within vehicle style varies a lot. Take a look at the medians and how much variability this graph detects. It is important to notice that these styles are very descriptive. Some of them already actually include how many doors does the car has. This is extremely important because this shows that some of the other variables are confounded within this one.



After seeing how the vehicle style has an impact on price, let's look at the size of the vehicle now. It was mentioned before that the compact vehicle always had a mean lower than the rest. This confirms it, but also shows how the median is lower. The price seems to be increasing by car size, which also makes sense. Notice the amount of outliers within the compact vehicle categorization. Some other numerical features directly related to the vehicle size might be useful for the the KNN regression approach.

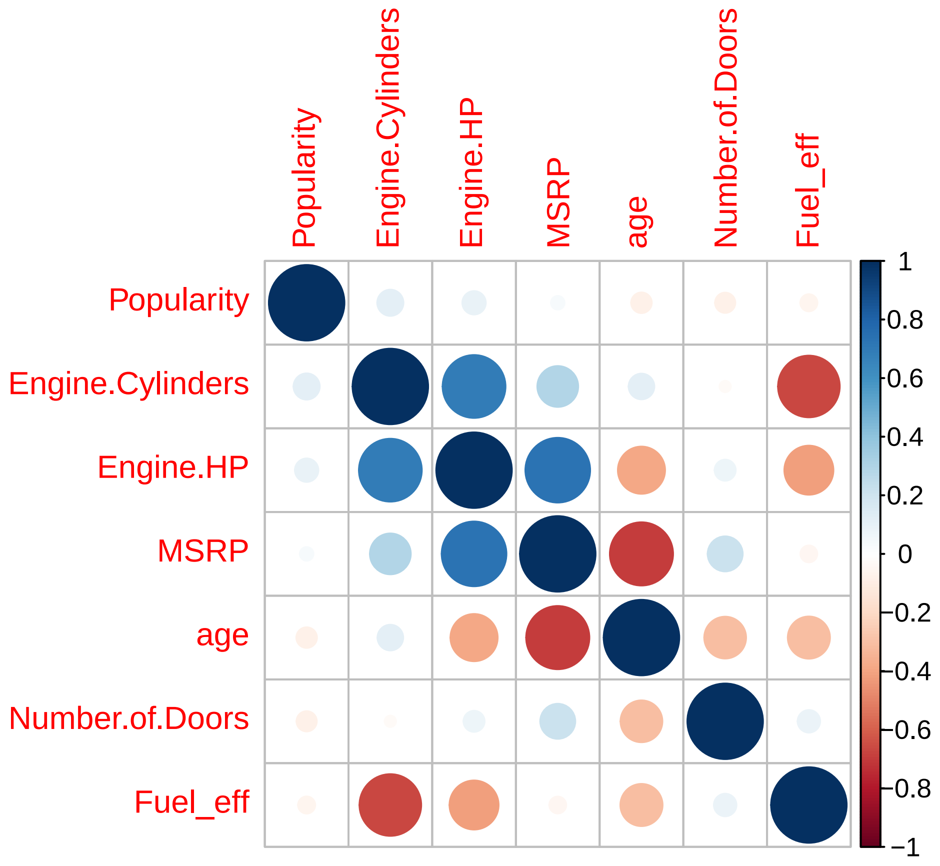
```
#In the next graph we can compare the boxplots for each vehicle type
train_set %>% ggplot(aes(Vehicle.Size, MSRP, fill= Vehicle.Size)) +
  geom_boxplot()
```



4. Feature Engineering & Selection Analysis

A critical step before jumping into the model development process is to determine the optimal features to use in them. Including all variables is not an option since it increases computational time drastically. In addition, this could lead to the model learning the data too well and then failing to perform at a diferent dataset. It could lead to overtraining. Therefore, this section provides a correlation and feature significance analysis.

Let's start by testing the correlation between all numerical variables:



In the figure above, high positive correlation between MSRP and horsepower is confirmed. In the data visualization section, the positive trend between these two variables was also shown. Age has a negative correlation with MSRP, wich was also expected. In addition, it is important to notice that horsepower and number of engine cylinders is also highly correlated.

Furthemore, the dataset seems to contain a variable column with market categories for each car. Each entry seems to have several markets, which means that using this information as is, is not very useful since there are many different combinations of markets and this represents a feature of many dimensions.

```
##                         Market.Category
## 1 Factory Tuner,Luxury,High-Performance
## 3                Luxury,High-Performance
## 4                     Luxury,Performance
## 5                                 Luxury
## 6                     Luxury,Performance
```

By separating the “Market.Category” variable we can expand the number of available features and have 11 additional binary variables. This is extremely helpful because it will create new feature combination possibilities while capturing more true characteristics of the car. This process is applied to both datasets. The new dataset names are “train_set_clean” and “test_set_clean”.

```
#This process expands the original matrix by separating
#the category market feature into 11 different new binary variables
names<- unique(train_set %>%
                 separate_rows(Market.Category, sep = "\\,") %>% select(Market.Category))
```

```r
train_set_clean<-train_set %>%
  separate(Market.Category, into= c(names$Market.Category), sep = "\\,") %>%
  bind_cols(train_set$Market.Category)
```

```
## New names:
## * NA -> ...29
```

```r
train_set_clean$...29<- as.character(train_set_clean$...29)

#This loop will assign the binary values to the new variables:
for (i in 1:nrow(train_set_clean)){
  for (j in 10:20){ #one j for each unique type of category
    string<- data.frame(n= i, string= c(str_split(train_set_clean$...29[i],pattern = ",")))
    colnames(string)<-c("n","string")
    string$string<-as.character(string$string)
    value<-ifelse(colnames(train_set_clean[j]) %in% string$string, 1, 0)
   train_set_clean[i,j]<- value
  }
}
```

```
## New names:
## * NA -> ...29
```

Then, the new “N/A” market category variable is now called “Regular” for better understanding. In addition, all these features get converted to numeric class for correlation testing.

```r
#Lets rename the N/A version of car market category
names(train_set_clean)[names(train_set_clean) == "N/A"] <- "Regular"
names(test_set_clean)[names(test_set_clean) == "N/A"] <- "Regular"
```

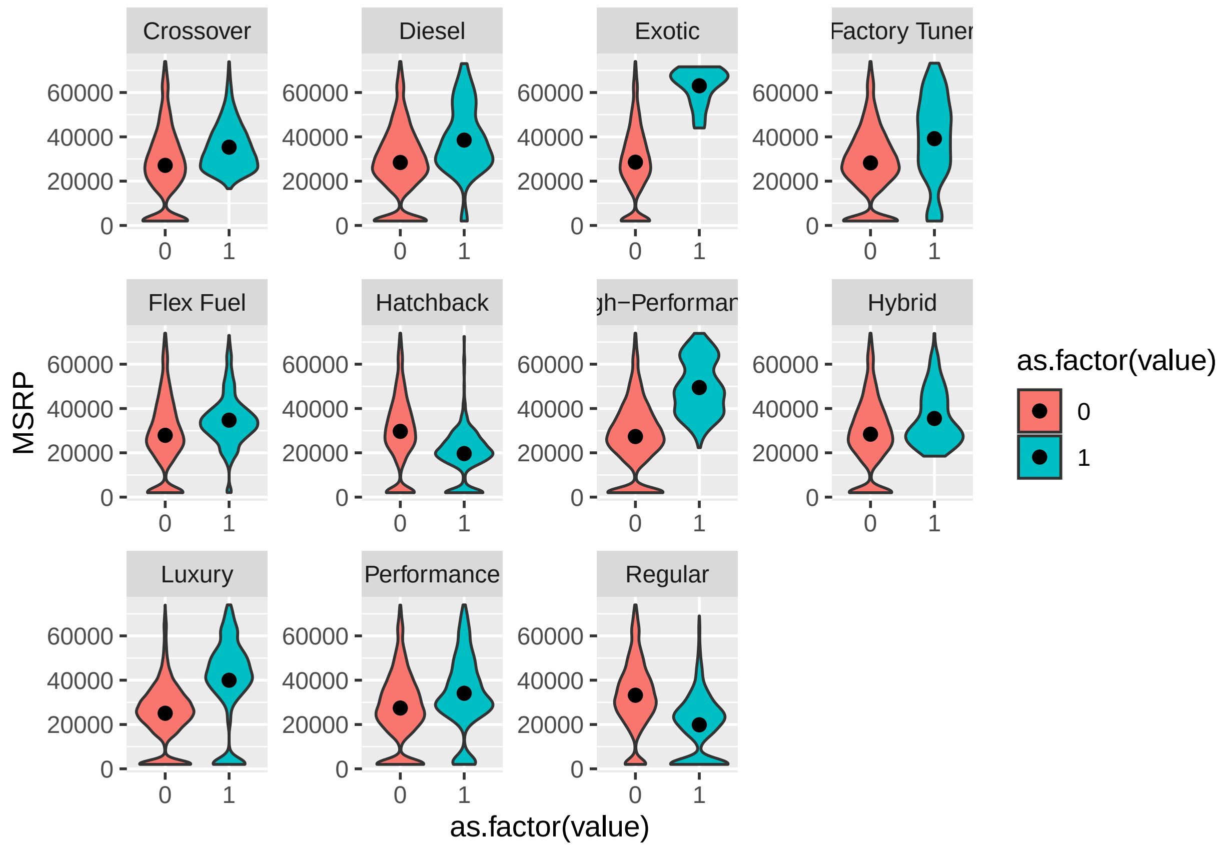
```r
#We can check we convert it the columns into numeric type by typing this code:
class(train_set_clean$Performance)
```

```
## [1] "numeric"
```

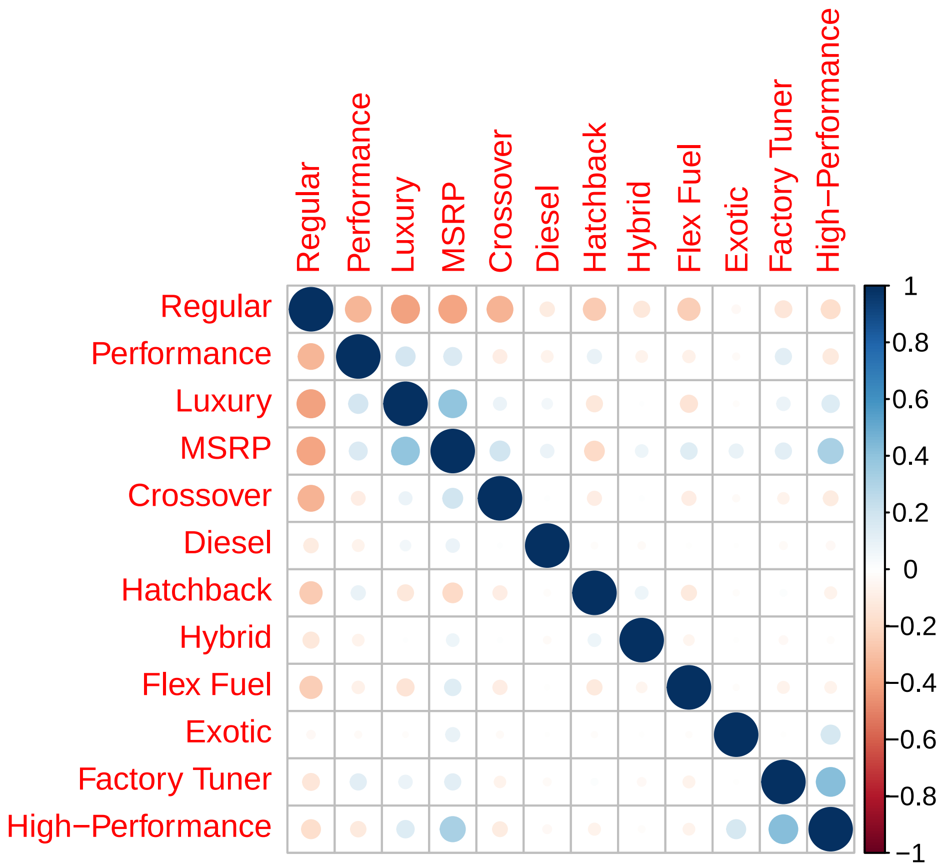
```r
class(test_set_clean$Performance)
```

```
## [1] "numeric"
```

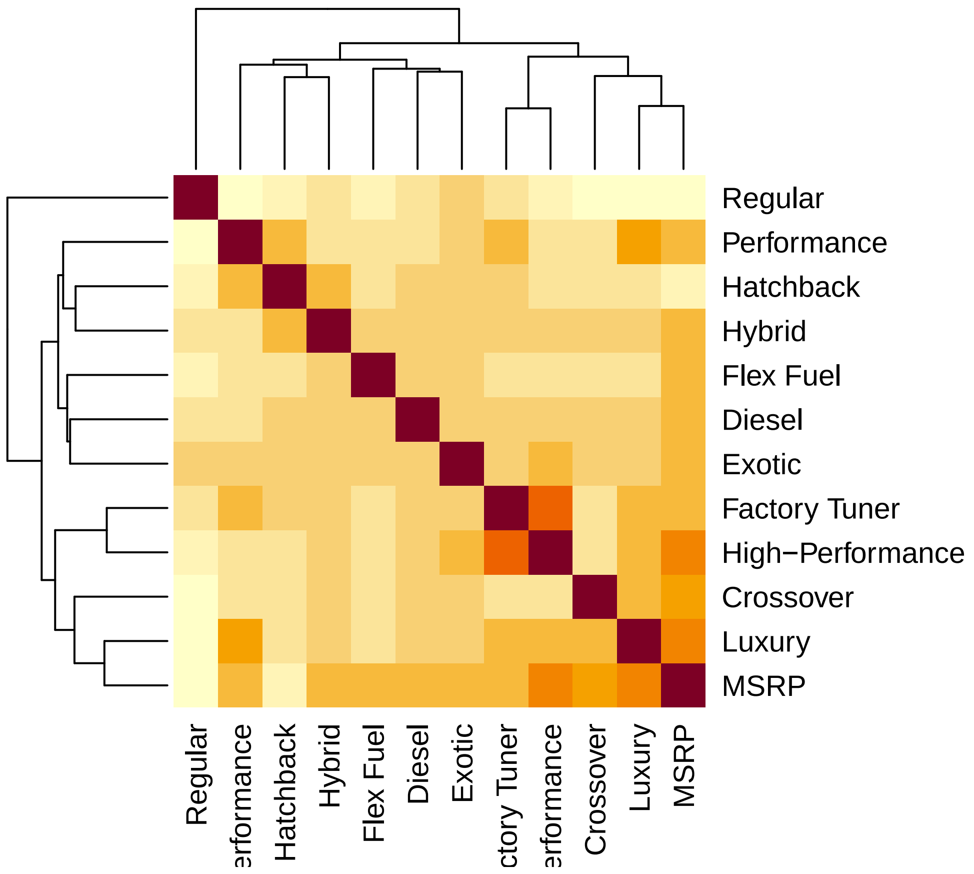
Before performing a correlation test between the new binary variables and the MSRP, it is important to understand the distribution of MSRP in each market category. As seen in the figure below, market categories like “Exotic”, “High-Performance”, “Luxury” and “Regular” seem to have a very different distribution and MSRP average for 0 and 1. This is a positive sign that these new features capture a huge amount of variability. Nonetheless, remember that some cars belong to 2 or more markets, meaning that there could be high correlation between some of these features.



After performing a correlation test, results below show that "High-Performance" and "Luxury" have a positive correlation relatively high with MSRP. Also notice how Regular is negatively correlated with almost all of the other features. Also notice how "Fatory Tuner" is positively corrrelated with "High-Performance".



To be able to picture this and partition how the correlations interact, the following heatmap is show below:



The following table shows some of summary statistics of the most important market categories. Notice how the confidence intervals (lower and upper bounds) between 0 and 1 for each individual market do not overlap in any of these features. This is a positive sign since it seems these categorical variables are capturing a lot of the variability within MSRP. Moreover, notice how the confidence interval for the positive "Fatory Tuner" variable, and the confidence interval for the positive "Luxury" variable overlap. It seems that a lot of the cars that are considered "Luxury" are also considered "Factory Tuner".

Market	value	mean	median	n	sd	me	lower	upper
Exotic	0	28549.44	28392.5	8648	15946.074	282.0481	28267.40	28831.49
Exotic	1	63042.65	65690.0	17	8278.655	3302.6501	59740.00	66345.30
Factory Tuner	0	28223.99	28147.5	8354	15752.276	283.4806	27940.51	28507.47
Factory Tuner	1	39177.11	39375.0	311	18955.144	1767.9670	37409.15	40945.08
High-Performance	0	27348.44	27427.0	8168	15322.829	278.8742	27069.56	27627.31
High-Performance	1	49467.38	48100.0	497	12155.489	896.8535	48570.53	50364.24
Luxury	0	25077.33	25915.0	6607	13301.963	269.1788	24808.15	25346.51
Luxury	1	39981.24	42675.0	2058	18481.163	670.0911	39311.15	40651.33
Performance	0	27430.50	27100.0	7123	15689.615	305.7793	27124.72	27736.28
Performance	1	34098.48	32382.5	1542	16320.907	683.6430	33414.83	34782.12
Regular	0	33203.72	32662.5	5688	15363.353	335.0684	32868.66	33538.79
Regular	1	19853.72	21665.0	2977	13337.765	402.0878	19451.64	20255.81

For final testing of these new features, a very simple aproach was taken to just verify linear significance with the estimated price. Therefore a linear regression was performed only for testing purposes. High level of significance was obtained from almost all features except "Regular". Obviously the approach is simple and does not have enough reach to capture the real MSRP behavior. This is why this was performed just to calculate P-values of these features as a linear regression test.

```
##
## Call:
## lm(formula = MSRP ~ ., data = feature_testing)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -40752  -5884   1022   7091  49142
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          20943.5      465.7  44.976  < 2e-16 ***
## ‘Factory Tuner‘      -2850.4      825.2  -3.454 0.000555 ***
## Luxury               10998.8      399.0  27.564  < 2e-16 ***
## ‘High-Performance‘   23020.8      742.1  31.022  < 2e-16 ***
## Performance           8318.0      442.6  18.795  < 2e-16 ***
## ‘Flex Fuel‘          11157.7      572.7  19.482  < 2e-16 ***
## Regular              -1089.8      519.8  -2.096 0.036080 *
## Hatchback            -6300.8      537.4 -11.725  < 2e-16 ***
## Hybrid               10215.6      835.2  12.231  < 2e-16 ***
## Diesel               10809.5     1022.5  10.572  < 2e-16 ***
## Crossover             9260.2      460.8  20.096  < 2e-16 ***
## Exotic               17784.4     3131.6   5.679  1.4e-08 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 12610 on 8653 degrees of freedom
## Multiple R-squared:  0.3804, Adjusted R-squared:  0.3796
## F-statistic:   483 on 11 and 8653 DF,  p-value: < 2.2e-16
```

Moving on to new variables, grouping by transmission type and driven wheels provides information with respect to how variable the data is with the combination of these variables. As seen in the table below, most of the cars are automatic. Furthemore, “Automated_Manual” cars seem to be the most expensive but with a ver low count of observations. Manual transmission type seems to be the cheapest of of all 5 types.

Transmission.Type	Driven_Wheels	count	avg_msrp
AUTOMATED_MANUAL	all wheel drive	84	45176.964
AUTOMATED_MANUAL	front wheel drive	233	30801.180
AUTOMATED_MANUAL	rear wheel drive	10	52973.500
AUTOMATIC	all wheel drive	1387	39281.025
AUTOMATIC	four wheel drive	772	37551.201
AUTOMATIC	front wheel drive	2409	26012.645
AUTOMATIC	rear wheel drive	1557	32624.420
DIRECT_DRIVE	front wheel drive	6	33794.167
DIRECT_DRIVE	rear wheel drive	3	42400.000
MANUAL	all wheel drive	121	26597.050
MANUAL	four wheel drive	283	10850.647
MANUAL	front wheel drive	1103	14850.341
MANUAL	rear wheel drive	684	24117.982
UNKNOWN	four wheel drive	1	2578.000
UNKNOWN	front wheel drive	4	2000.000
UNKNOWN	rear wheel drive	8	3363.125

From what has been presented, the insights from the data visualization section, it was concluded that “Engine.HP” and “age” seem to have a signficant effect on MSRP. This can be confirmed by performing a linear regression feature test on these two variables. High significance for both variables was observed.

```
##
## Call:
## lm(formula = MSRP ~ age + Engine.HP, data = train_set_clean)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -26903  -5231   -883   4737  40912
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11026.285    324.510   33.98   <2e-16 ***
## age          -999.945     12.220  -81.83   <2e-16 ***
## Engine.HP     107.601      1.181   91.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8181 on 8662 degrees of freedom
## Multiple R-squared:  0.7389, Adjusted R-squared:  0.7388
## F-statistic: 1.225e+04 on 2 and 8662 DF,  p-value: < 2.2e-16
```

5. Methodology

This project consisted on the development of 3 machine learning algorithms that would predict the MSRP of a car with certain characteristics. These methods were used for effectiveness on the available data structure. The first model was developed using a K-Nearest Neighbour Regression approach, then a Regression Tree and finally Random Forest. To really understand and compare the effectiveness of these models, three primary measurements were used. RMSE was not one of them since it penalizes large errors, and since some rare cars are extremely expensive or extremely cheap, these errors could be misleading. First, the intention is to quantify whether the predictions are a good fit in comparison with real values. This is achieved by the following formula, where *cor* is the correlation between the predicted values x and the target values y:

$$R^2 = {cor_{x,y}}^2$$

Another important measurement used to quantify how far the predicted values were from the real ones, is by % error. This is calculated using the following formula:

$$error\% = \frac{x-y}{y} * 100$$

The Mean Absolute Error (MAE) is also used as a money reference:

```r
#Lets create a function that returns the Mean absolute error
MAE<- function(true_msrp, predicted_msrp){
  mean(abs(true_msrp-predicted_msrp))
}

#Calculates Rsquared:
rsq <- function (x, y){
  cor(x, y) ^ 2
}
```

5.1 k-Fold Cross Validation

A lot of times, the test error is quite different than the training set error. This could happen because of many reasons, like overfitting. k-Fold is a validation procedure that consists in separating the training set into k, most commonly 5 or 10, groups or folds of equal smaple size. The first group is used for validation and the other k-1 folds are used for training only. This approach is widely used to find the best tuning parameter and to prevent overtraining the data. This project will use a 10-Fold Cross Validation for the Trees Regression method and a 5-fold for Random forest, because of computational time constraints.

5.2 K-Nearest Neighbour (KNN) Regression

This is one of the most common and best-known non-parametric machine learning approaches for regression. This is a similar algorithm to bin smoothing but easier to adapt to higher dimensions. The algorithm starts by defining the distance between observations based on the different desired features. This approach is obviously similar to the K-Nearest Neighbour Classifier algorithm, but since this is not a classifier problem, the algorithm identifies the K closest data points to x_0 and compute the average of the values and assign it to x_0. Resulting in an estimation of the neighborhood. The higher the K, the smoother the estimate, but smaller the K, the more flexible, though wigglier the estimate. Since this is a problem of many dimensions, flexibility is more important than smoothness. The algorithm needs to adapt to the many different characteristics of a car. This approach needs numerical variables to calculate distances between the points.

With this approach only numeric variables are used. After all the visualization and correlation testing, the best features chosen for this particular algorithm were: Engine horsepower, number of doors, High-Performance, Popularity (by brand), and Fuel efficiency. Code, training results, and test results are located in the Results section of the report.

5.3 Regression Trees

This algorithm can be applied to both classification and regression problems. For regression, the algorithm stratifies or cuts the predictor space into several different simplified regions. To predict a value of the observation x_0 in Region R_j then it naturally uses the mean or mode of all the observations located in R_j to calculate the value of y_0. This approach allows us to incorporate more categorical variables, in comparison to the previous approach.

5.4 Random Forest

This procedure is considered a boost from regular decision trees. Random Forest algorithm builds many decision trees (the name comes from this). When building these trees, each time that a region must be partitioned, a random sample set of pedictors is considered to use at each split. In other words, the alogrithm is only allowed to choose a predictor from this "bag" of random sampled predictors.

6. Results

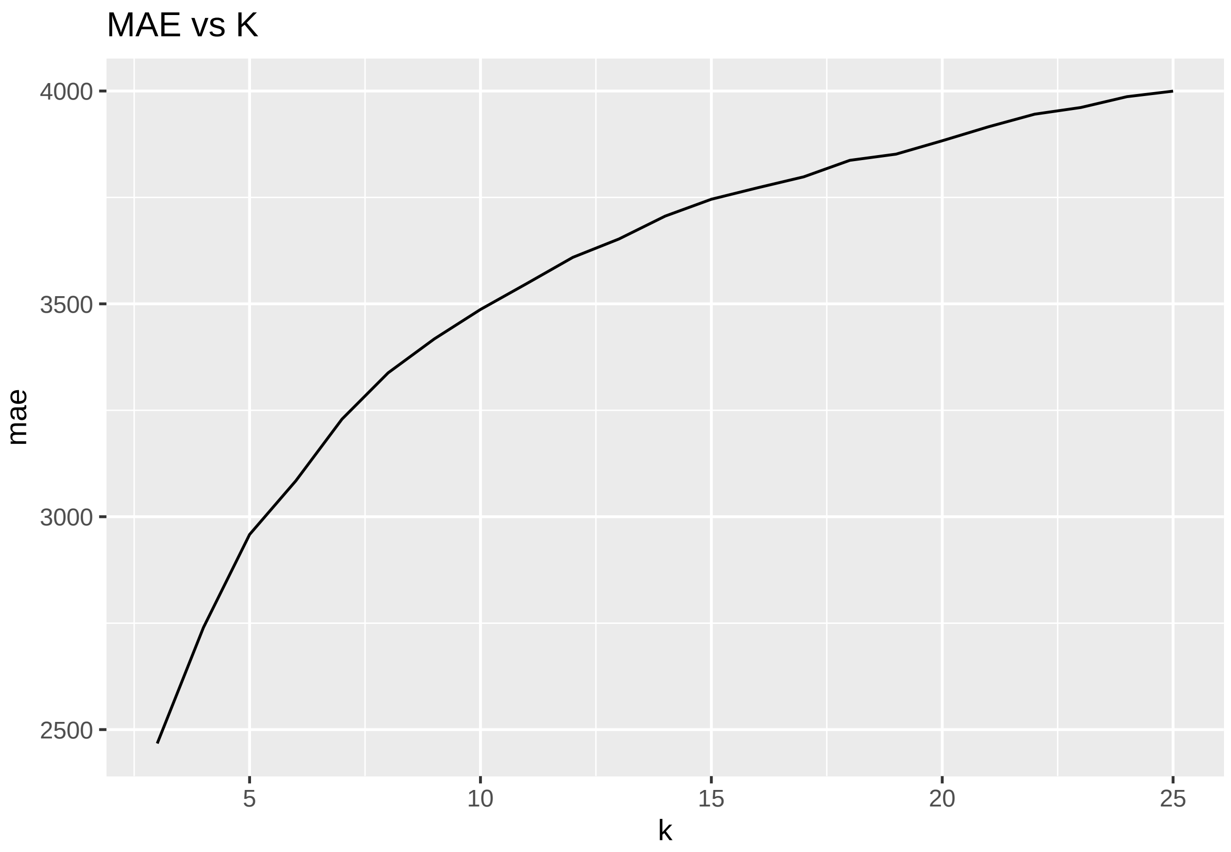
6.1 KNN Regression Results

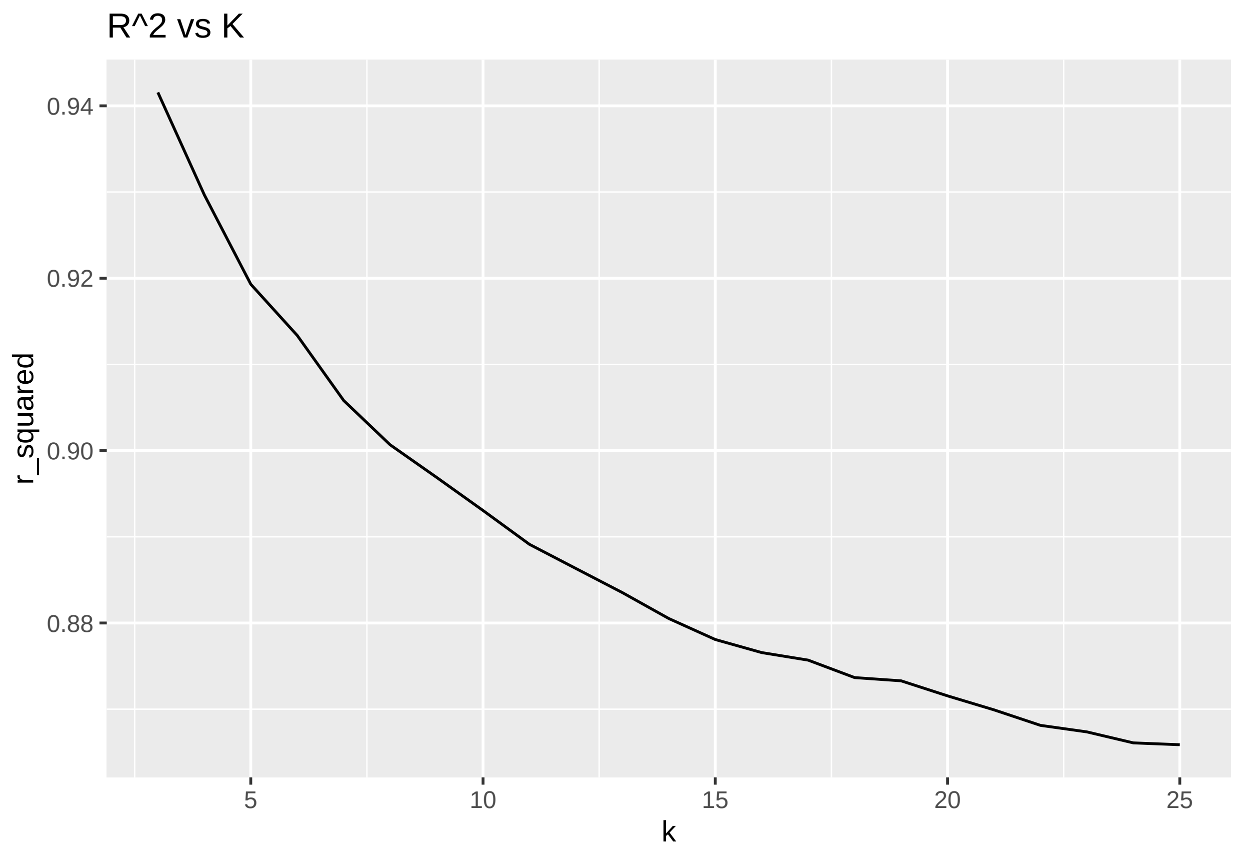
Training Performance

The code below represent the KNN approach taken and the R functions used to predict the MSRP values. Several number of k were used to determine the best tuning parameter. As mentioned before, it was expected that more flexibility was needed for better R^2 and MAE performance.

```r
#We create a table so that we can compare and store all RMSE results for different
#values of k
results_knnreg<-data.frame(k= as.numeric(), mae= as.numeric(), r_squared= as.numeric())

#Now we can run a loop to test for different k values for our knnreg algorithm
for (i in 3:25){
fit_knnreg<- knnreg(x=train_set_clean[,c(5,9,25,27,28)],y= train_set_clean$MSRP, k=i)

predictions<- data.frame(fitted= predict(fit_knnreg, train_set_clean[,c(5,9,25,27,28)]))
results_knnreg<-bind_rows(results_knnreg, data.frame(k=fit_knnreg$k,
                            mae= MAE(train_set_clean$MSRP, predictions$fitted),
                            r_squared= rsq(train_set_clean$MSRP, predictions$fitted)))
}
```

For the amount of different models and brands with respect to the number of observations, the performance of this model is definitely around epectations. The training set results are shown in the figures below. Optimal k is found to be at k=3 with an R^2 of above 0.94 and a MAE below 2500$. Because of the nature of this algorithm, we expect a lower performance in the test set.





Let's set the training again with the algorithm using k=3:

```
#So we run the algorithm again but just for k=3
fit_knnreg<- knnreg(x=train_set_clean[,c(5,9,25,27,28)],y= train_set_clean$MSRP, k=3)
```

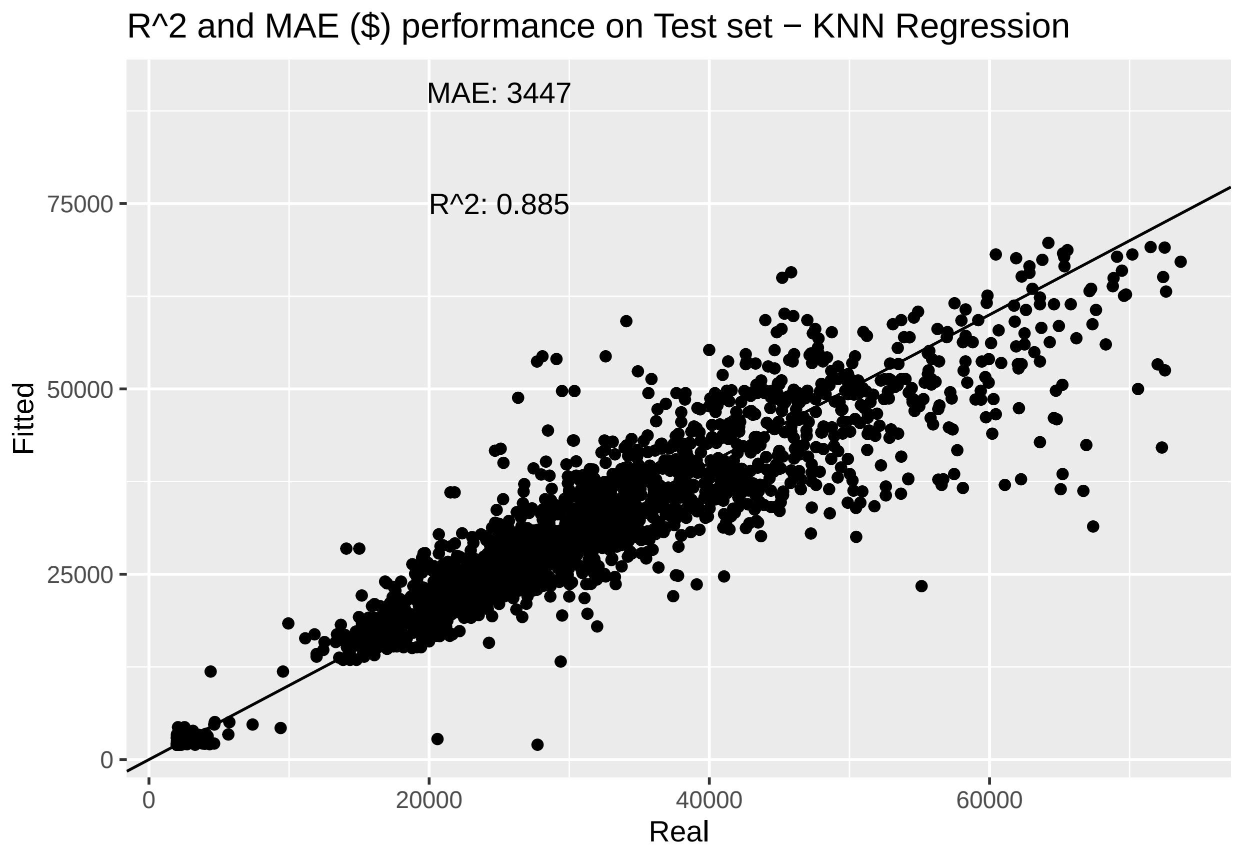
Test Results

The algorithm runs fast and predicts the MSRP values with a median percentage error of about 8.3%.

```
#Run on test set:
test_knnreg_results<-data.frame(Real= test_set_clean$MSRP,
                                Fitted= predict(fit_knnreg, test_set_clean[,c(5,9,25,27,28)]))

#Calculate %error of calculation and the median error %
test_knnreg_results<-test_knnreg_results %>%
  mutate(percentage_error= abs((Fitted-Real)/Real)*100)
median(test_knnreg_results$percentage_error)
```

```
## [1] 8.269051
```

As seen in the figure below, the performance of the algorithm in the test set is lower than the results of the training set. This could be a coincidence coming from the data partition or most probably because of some slight overfitting. Nonetheless, it is a somewhat postive result that serve as a baseline. the fitted values are close to the normal line, meaning that the results are good but it can still be improved by an algorithm that can take categorical variables as well.



6.2 Regression Tree Results

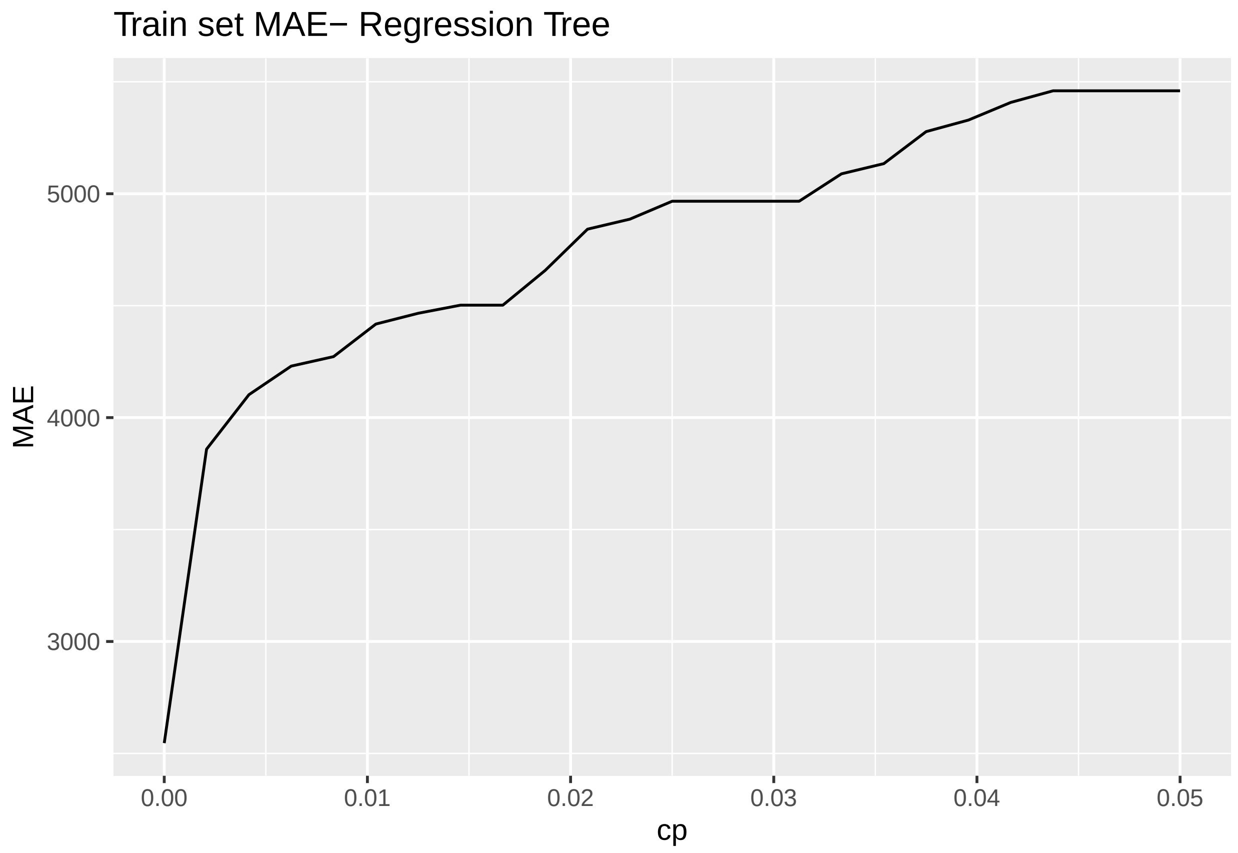
Training Performance

The code below shows the Tree Regression approach with a 10-fold cross validation to avoid overfitting. For this process to work some tasks had to be done beforehand. First, give valid column names for the caret package code to properly read these variables. Second, all variables created had to be set as factors since that is the best way to describe them. This code shows all the variables included in the model to better define the predictor space of car characteristics.

```
#for 10-fold cross-validation control
control<- trainControl(method="cv", number= 10, p=0.9)

# Make Valid Column Names for both the test and training set
colnames(train_set_clean) <- make.names(colnames(train_set_clean))
colnames(test_set_clean) <- make.names(colnames(test_set_clean))

#Convert these market category variables into factors
train_set_clean[,10:20]<-train_set_clean %>% select(c(colnames(train_set_clean[,10:20]))) %>%
  mutate_if(is.numeric, as.factor)
test_set_clean[,10:20]<-test_set_clean %>% select(c(colnames(test_set_clean[,10:20]))) %>%
  mutate_if(is.numeric, as.factor)

set.seed(1, sample.kind= "Rounding")
#Lets perform a decision regression tree to be able to incorporate categorical variables as well
```

```
rpart_fit<- train(MSRP~ Make +Luxury+ Performance +  Regular + Flex.Fuel + Factory.Tuner +
                    Engine.HP + Transmission.Type+ Driven_Wheels+
                    High.Performance+ Hatchback+ Hybrid+ Diesel+ Vehicle.Style+
                    Exotic+ Crossover+ age+ Fuel_eff,
                  method= "rpart",
                  tuneGrid=data.frame(cp=seq(0,0.05,len=25)),
                 trControl= control,data= train_set_clean)
```

The graph below defines the behavior of the MAE with respect to the different tuning parameters being tested. In this case, the tuning parameters is called "cp" which comes from complexity parameter, and it is used to determine the optimal tree size. In this case, the larger the tree the better results colected.

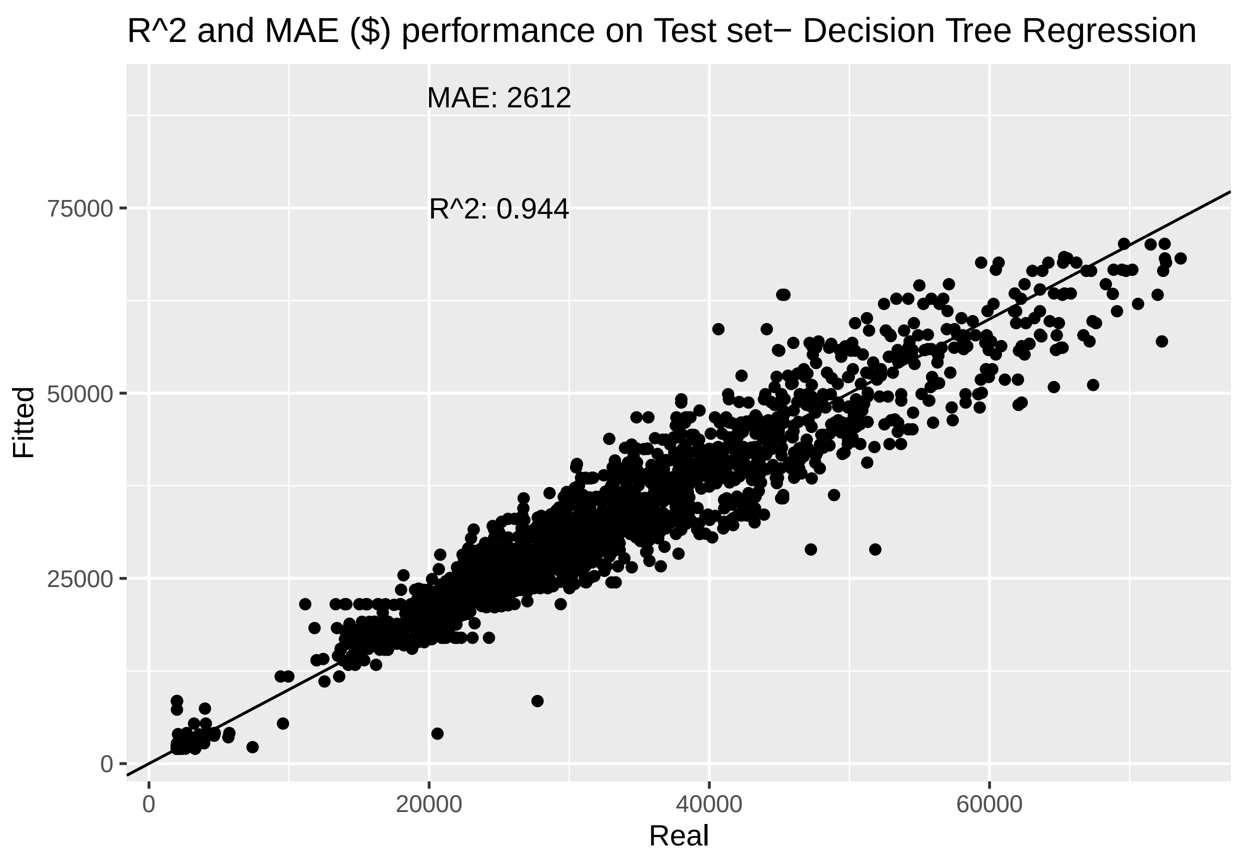


Test Results

Outstanding performance of this model on this dataset. The results show a median percentage error of 7%, which is definitely an improvement from KNN Regression. In addition, R^2 increases to 0.944 while the MAE decreases drastically to just 2612$.

```
## [1] 7.039715
```

The figure below shows how well of a fit these predictions are. They all gather around the normal line, getting closer and closer to the real values.



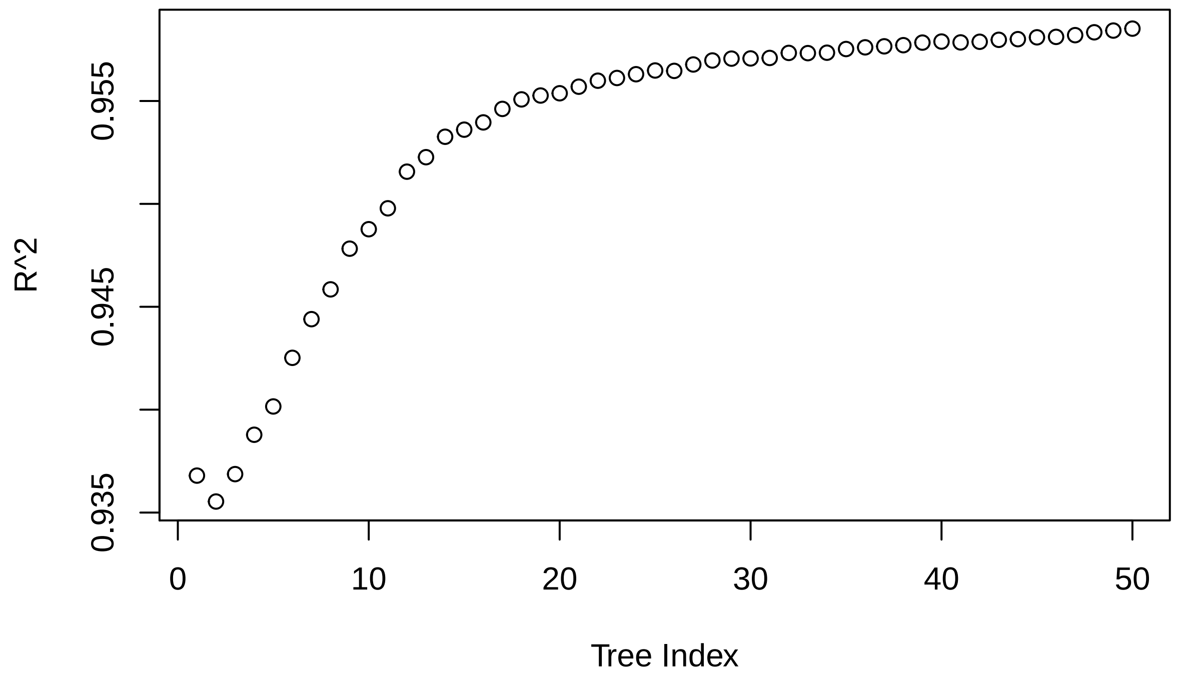
6.3 Random Forest Results

Training Performance

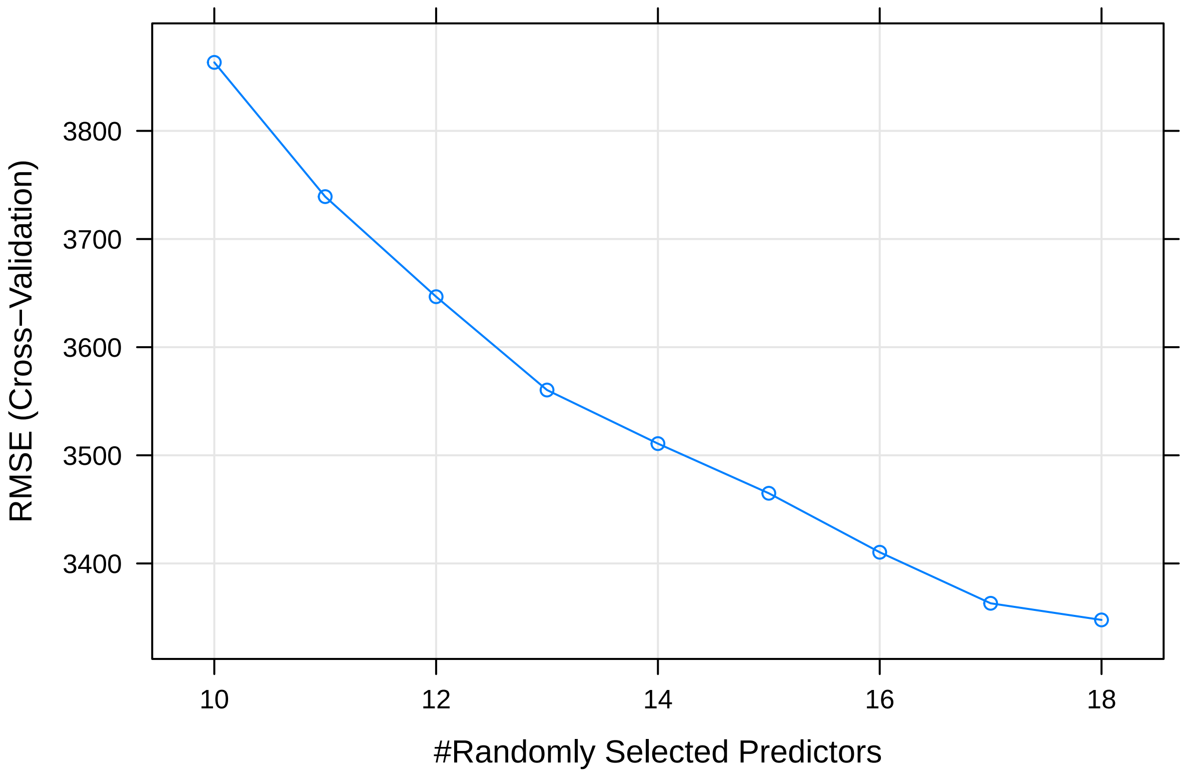
The code below shows the Random Forest approach with a 5-fold cross validation to avoid overfitting. This code shows all the variables included in the model to better define the predictor space of car characteristics. All variables are the same as in the Regression Tree, meaning that the results are very well comparable. Remember that some people consider Random Forest as a boost to decision trees. This means that is safe to expect better performance than the last section. Different levels of the tuning parameter “mtry” are tested. This represents the number of randomly selected variables to use in each node. The number of trees was set to 50 for computational time purposes.

```r
#for 5-fold cross-validation control
control_rf<- trainControl(method="cv", number= 5, p=0.9)

set.seed(1, sample.kind= "Rounding")
#set tunnig paramater sequence to test randomly selected variables
grid<-expand.grid(mtry= seq(10,18))
rf_fit<- train(MSRP~ Make+ Luxury+ Performance +  Regular + Flex.Fuel + Factory.Tuner +
                 Engine.HP + Transmission.Type+ Driven_Wheels+
                 High.Performance+ Hatchback+ Hybrid+ Diesel+ Vehicle.Style+
                 Exotic+ Crossover+ age+ Fuel_eff,
               method= "rf", tuneGrid=grid,
               ntree=50, trControl= control_rf, data= train_set_clean)
```

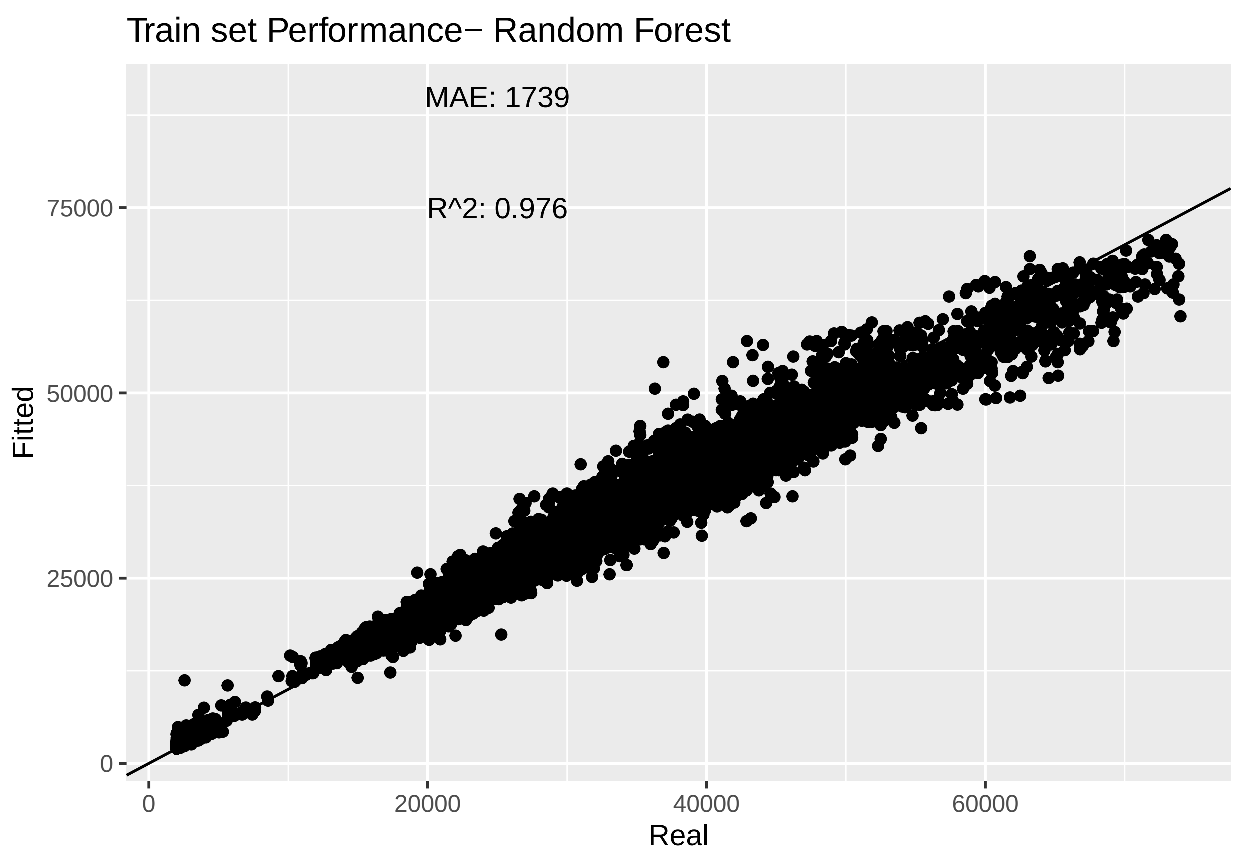
The graph below shoes the performance of R^2 as you increase the tree index. Great performance is shown again as you move towards the last trees.



As seen in the figure below the performance of the model improves as you increase the number of randomly selected predictors to 18, which is 1 less than the total amount of features.



The graph below is an important one and the key success of this project. It clearly shows an amazing fit with a R^2 of 0.976 and an MAE of 1739$. These values are a major improvement. The hope is that it will perform this well on the test set. The predicted values get closer and closer to the target line.



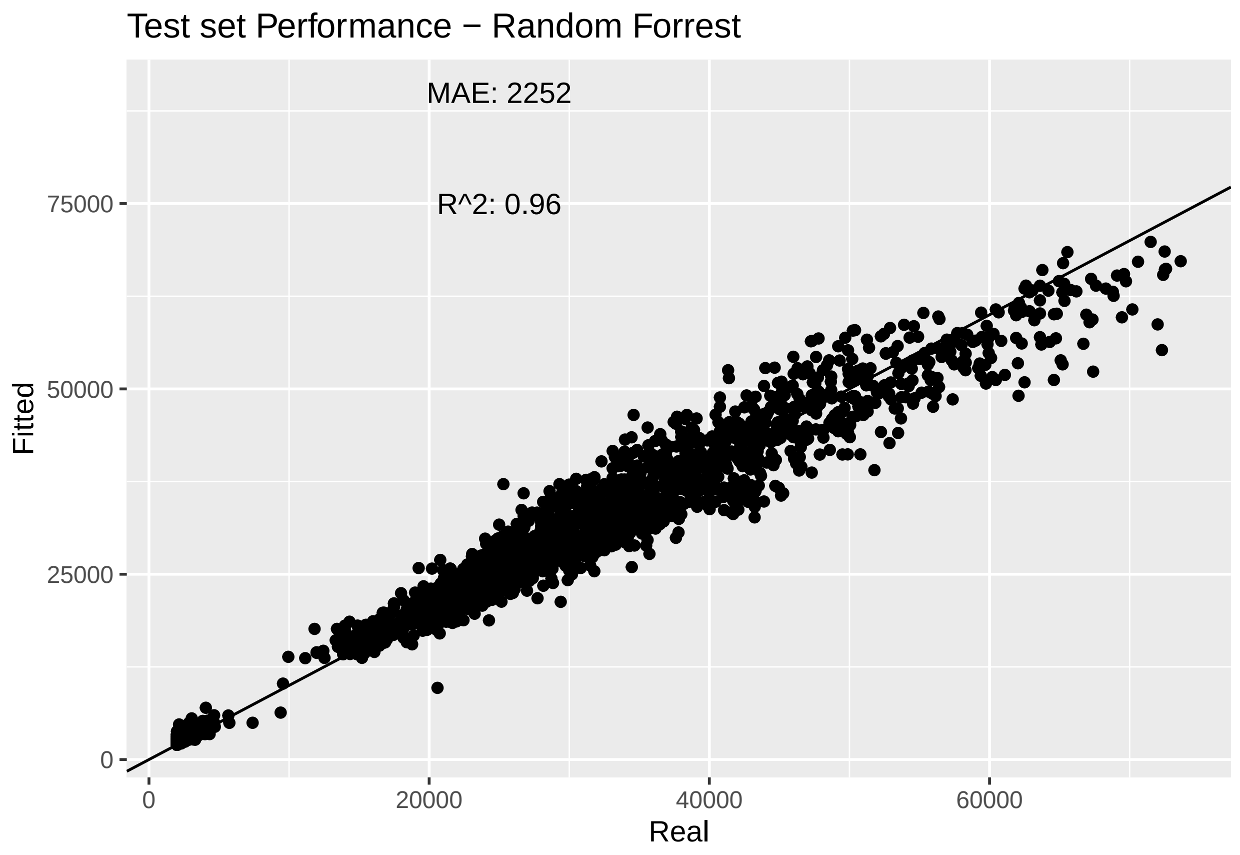
Test Results

The results of Random Forest are extremely positive. It definitely performed above expectations. The results show a median percentage error of 6.6%. This is an improvement from the 7% of the Regression Tree.

```
#Run on test set:
rf_results<-data.frame(Real= test_set_clean$MSRP,
                       Fitted= predict(rf_fit, test_set_clean[,c(1,5,7,8,10:20,22,27,28)]))
#Calculating error% of each observation
rf_results<- rf_results %>% mutate(percentage_error= abs((Fitted-Real)/Real)*100)
#Calculating the median %error
median(rf_results$percentage_error)
```

```
## [1] 6.66535
```

The figure below describes an outstanding fit of prediction to real values, with a R^2 of 0.96 and a MAE of 2252$. You can see that all data points, with a few exceptions, are extremely close to the fitted line. No overtraining was observed.



7. Conclusions

New technologies and online services are pushing E-commerce to expand to every possible industry. It is time to embrace the change and create new online car services for quote estimations and price optimization that would create new profits and at the same time make customers happier. Different machine learning algorithms were used to estimate MSRP on thousands of different cars. KNN Regression results show slight overfitting but still test results were generally good (R^2= 0.89) with a median percentage error of 8.3%, with a fast and steady performance. The lack of numerical variables challenged the algorithm, but results matched expectations. Later on, a Regression Tree model was developed to target MSRP with both numerical and categorical features. It performed extremely well, beating expectations (R^2= 0.944). Training and testing was computationaly fast and achieved a median percentage error of 7%. Finally, the Random Forest algorithm, which ran slower because of greater complexity, was run for different tuning parameters and achieved an outstanding performance (R^2= 0.96) with a median percentage error of 6.3%. KNN Regression was not performed under any type of cross validation or bootstrap approach, meaning that future work con be done with this model to better estimate MSRP in the test set. Adquisition of additional data to be able to estimate rare cars might also be beneficial for a more well-rounded MSRP estimator. Some limitations were found in performing this project. Computational time constrained the amount of testing of tuning parameters that could have been performed.

This type of work is extremely beneficial, especially in today's world in which e-commerce has taken a kew role in sales and supply chain. A lot of customers do not like going to car dealers and spending a couple of hours on negotiations and paperwork to buy a car. Constant pricing estimation is a key part of every business, and not just in the car industry. This type of work can easily be adapted to new pricing strategies in different industries acrross the country.