New Car Pricing

Michael Harrington
Springboard - Capstone Two Report

Problem Statement and Introduction

What will, or should, the MSRP (manufacturer's suggested retail price) of a new car be? How accurately can one predict what the original price of a given car was in the past; or typically more business relevant, can one project how a new car will be priced based on data?

The automotive industry is truly monumental in size and scope. Many of us are surrounded by it daily and will often see hundreds, even thousands, of cars a day. In 2019, in the USA alone, customers spent almost 500 billion dollars on over 17 million new cars (CNBC). There are a plethora of companies and individuals involved, from those directly involved in their manufacturing and distribution all the way down to the network of smaller support companies such as detailers, independent mechanics, and aftermarket part suppliers. Any of these, along with investors and business analysts in the field, could potentially be clients interested in how new cars will be priced. Most likely the individual manufacturers themselves, or their competitors, would be the ones most interested in being able to project or predict pricing for cars - whether it be their own or their competitors. I am quite sure that most of them in fact already do.

So the aim of this project is to use a dataset to create a machine learning model to predict new car prices. In this case cars that have already been sold on the market, but one that could also be potentially used to project in the future (obviously assuming the relevant data is available).

Dataset

The idea for this project was inspired by the dataset rather than the other way around. I have always had a passion for cars, and really driving in all forms, and I searched several sources for a dataset that was related in some way. The dataset that was chosen was found on kaggle.com and can be seen in the following link - new car prices. As one can see in the link, the dataset came in two different forms (a cleaned dataset and the uncleaned original). I decided to work with the original (*new_cars_price.csv*) in order to practice cleaning and wrangling the data myself. The initial dataset contained 32,316 new cars produced between 1990 and 2019 for the American market. It is quite a feature rich dataset, as it contains 57 unique columns or features.

Data Wrangling and Cleaning

Even before exploring the dataset in earnest (just seeing it in part on kaggle) it was quite clear that a good amount of work was going to need to be done in just cleaning and extracting certain features. As mentioned in the dataset section this was done mostly by choice in deciding to use just the un-cleaned original dataset rather than the cleaned one which was also available.

1. Data Exploration

To start out I did a good thorough exploration of the dataset. This is important to get a good understanding of what all is in a given dataset, particularly one with over 50 features. This included, but wasn't limited to, taking a good look at all the column names and their units, checking out the types of all the columns, calculating their summary statistics, checking for any duplicates, and finding the number of unique as well as missing values in each. There were no duplicate values and the summary statistics seemed reasonable for the numeric features, but there were quite a lot of missing values which would need to be addressed. In example one feature was dropped as it contained over 2/3rds missing values. A few categorical features contained either redundant or irrelevant

data and were thus also dropped. Also some features had incorrect dtypes that would need to be addressed.

2. Feature Extraction

One thing that was made clear from the exploration of the data was that certain columns contained too much information on each car and it would be preferable to split or extract multiple features from them. A good example of this was the *Model* column which contained the year, manufacturer, model name and often a subtitle or trim level for each car (for example *2019 Acura RDX Specs: AWD w/Technology Pkg*). Intuitively it is easy to believe that those would each be quite important features for predicting new car pricing. So from this column *Model Year*, *Manufacturer*, and *Car Model* were created as unique features. In a similar way various other additional features I thought could possibly be relevant (though most likely not as important as the prior features) were extracted from the original dataset including *Engine Type*, *Front Wheel Diameter*, and *Propulsion Source* etc.

3. Data Cleaning

The last major thing to do to the dataset before EDA and modeling can be properly done is to clean it. The dataset included extraneous, non-numeric characters in many of the features. Some examples include dollar signs in the MSRP column values, and units, commas, and backslashes in others. Still other categorical features had multiple different strings representing the same or similar values (as in AWD and awd and All-wheel or v6, V6, and v-6). Many of these features required many conditional loops to filter for certain values and output all as a single value. This portion was quite time consuming, as most of the categorical features required some or a lot of this work to be in a usable form. Some features were renamed for the sake of clarity, and a number required a change in dtype. Also as was mentioned before, the dataset had a relatively large number of missing values. In an effort to keep as many observations as possible (and since there are quite a lot of features thus rendering it illogical to drop a row if it had any missing values) all missing values were imputed in some form. For the numeric features the missing values were essentially all imputed with the median value (to better ignore large outliers, though both were usually quite close in value), while for the categorical features ‘missing’ or ‘unavailable’ was used.

Exploratory Data Analysis

With the dataset having been cleaned and all the presumably relevant new features added it was time to get a deeper, more intuitive understanding of our data by examining some visual relationships.

To start out, I like to take a good look at the distributions of all the numeric features with a histogram. The results can be seen in **Figure 1**. The distributions look logical for the features for the most part. We can see that the number of observations (cars) increases with *Model Year*, but this is a good thing for most practical uses (probably more likely to be interested in predicting new car prices going into the future, thus the newer the car the more relevant). Wanting a clearer picture of our target feature *(MSRP)*, a far more detailed histogram for visualizing it was created and can be seen in **Figure 2.** From this it is clear that the MSRP values are mostly clustered in the 20,000-40,000 dollar range, which aligns with expectations. However, there is a long tail to the right which represents probably the small number of expensive, high-end luxury and supercars.

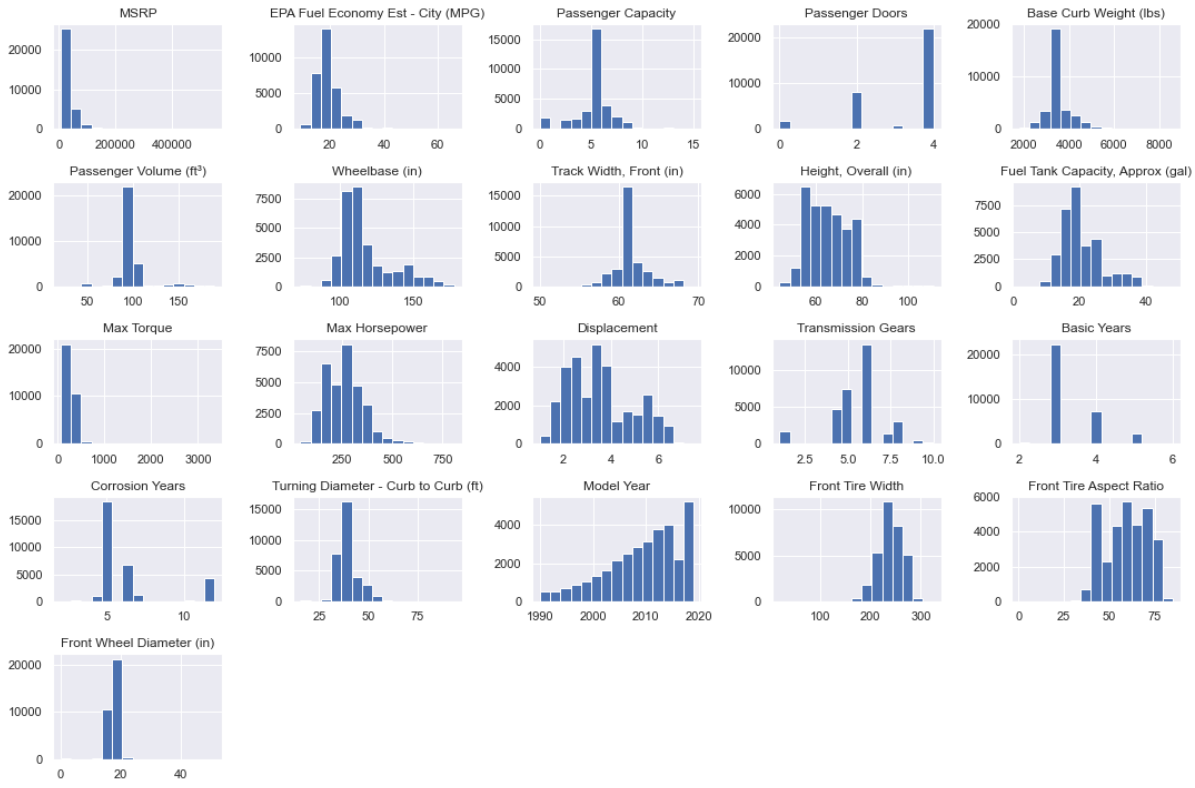

Figure 1: A histogram of the numeric features in the dataset

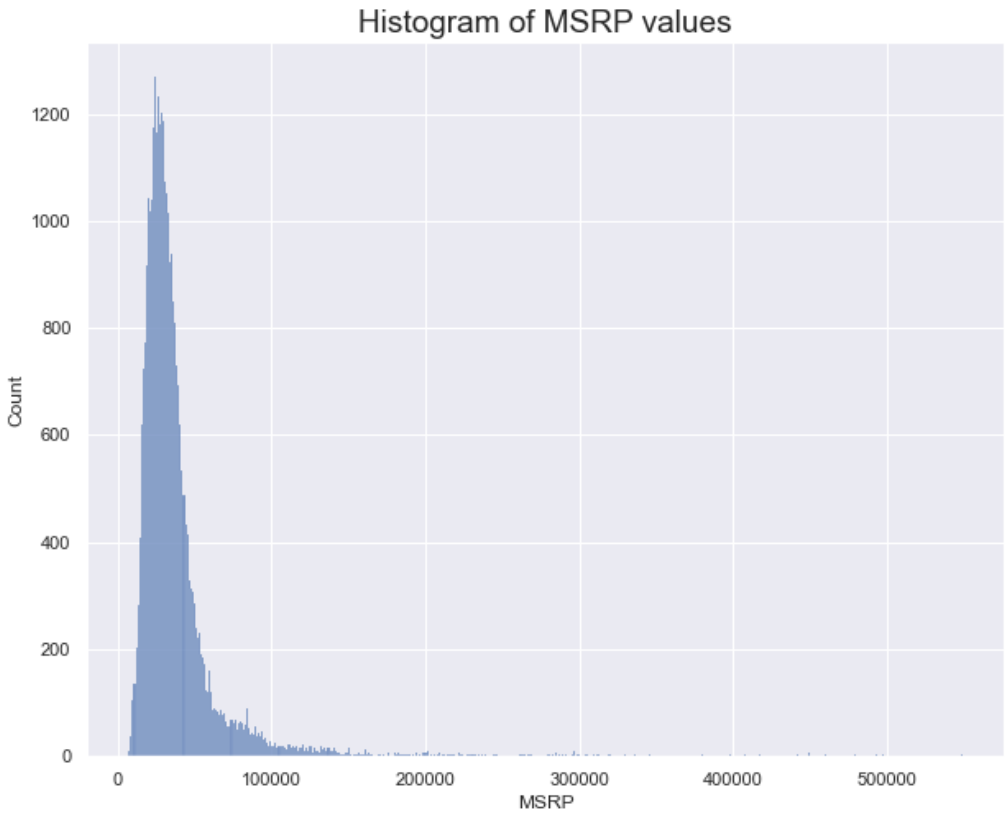

Figure 2: A more detailed histogram of our target feature *MSRP*. Clearly there are a few very expensive cars at top skewing the histogram.

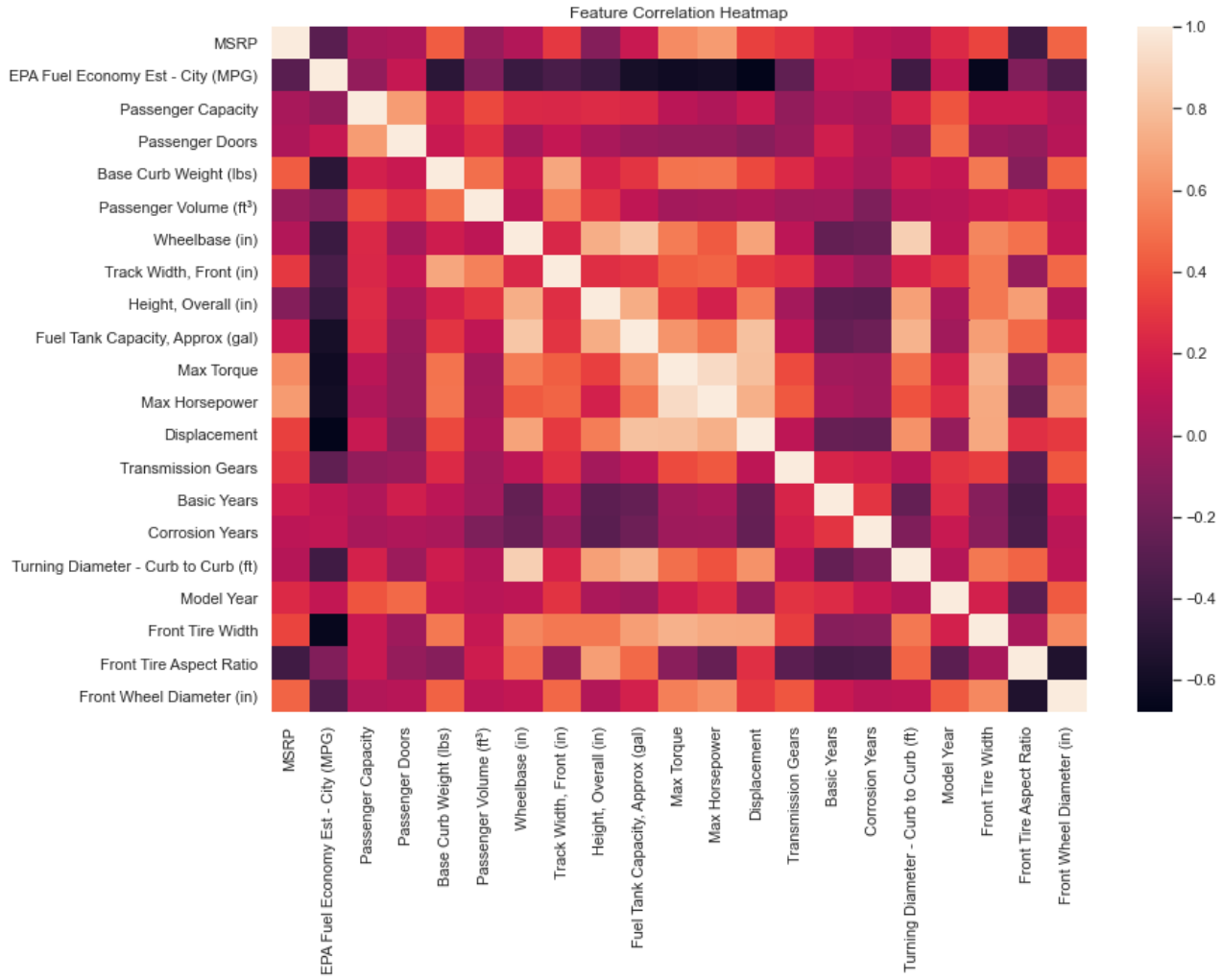

Figure 3: A correlation heatmap of all the numeric features in the dataset. The lighter the shade of color the greater the positive correlation. Of particular interest is our target feature *MSRP* at the top.

To further explore the numeric features in the dataset a correlation heatmap can be seen in **Figure 3**. The heatmap shows the amount of correlation (from large positive correlation to large negative correlation) found between every numeric feature. *MSRP* seems to have a large positive correlation with *Max Horsepower*, *Max Torque*, *Base Curb weight* and *Front Wheel Diameter* among others. These all match my expectations, but I am surprised to see it's low to negative correlation with other size related features such as

Height, *Passenger Volume*, *Wheelbase* etc. It appears that larger cars often don't actually equate to higher prices (traditionally the opposite was typically the case). *Front Tire Aspect Ratio* and *EPA Fuel Economy* exhibit the largest negative correlation with MSRP and that makes sense due to cheaper, economy based cars usually having higher values for both than the norm. It is interesting to note the negative correlation Fuel Economy has with almost every numeric feature. Finally I was surprised to see Model Year not have a higher positive correlation.

After seeing the heatmap, I was interested in visualizing how a few key features specifically relate to MSRP. So after being a little surprised at how Model Year wasn't even higher positively correlated I decided to look at the change in MSRP over the years in the dataset and the result is available in **Figure 4**. The graph highlights the clear almost linear increase in median MSRP over the years (keeping in mind the dataset has less observations the farther back in time). The median price has risen over 20,000 dollars in about 30 years.

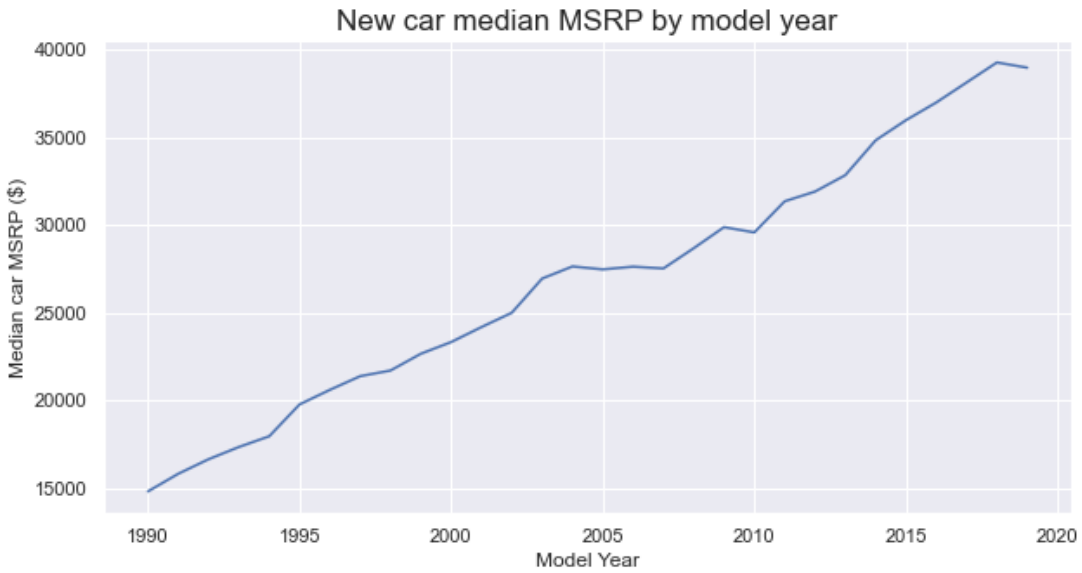

Figure 4: The median *MSRP* of cars in the dataset as a function of model year.

With *Max Horsepower* being the feature most positively correlated with *MSRP* I decided to examine that relationship in more detail with scatterplot comparing them with colors showing what I thought may be a key categorical feature in *Body Style*. The graph in

Figure 5 shows the strong positive correlation, though most cars are still clustered in the bottom left. Cars designated with a Body Style of "car" or "convertible" tend to make up more of the higher MSRP values, however 'car' is ubiquitous and the most common everywhere. There are a couple interesting nearly vertical lines of data points at a few specific horsepower values (approx. 250, 450, 560 and 600 hp respectively). This could be worth further examination, and I think it may represent engine sharing across models and brands along with just being popular horsepower targets for manufacturers for marketing and other reasons. Lastly I believed Manufacturer assuredly must be an important feature worth having a closer look at. **Figure 6** compares the boxplots of all the different MSRP values for each manufacturer containing cars in the dataset. As expected most brands are clustered below 50,000 but some have larger variances and a few high-end brands are much more expensive than the average. Clearly Manufacturer is a huge predictor for MSRP as expected.

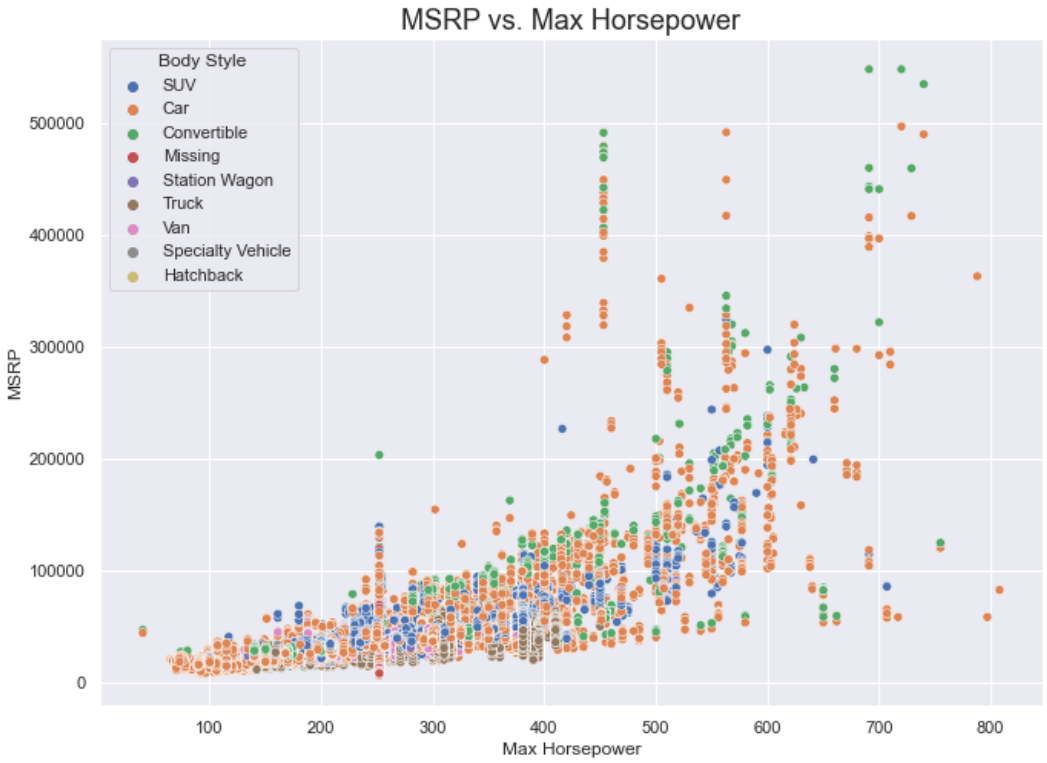

Figure 5: A scatterplot of MSRP and Max Horsepower. The colors of the data points show the body style of the car. The positive correlation is visible.

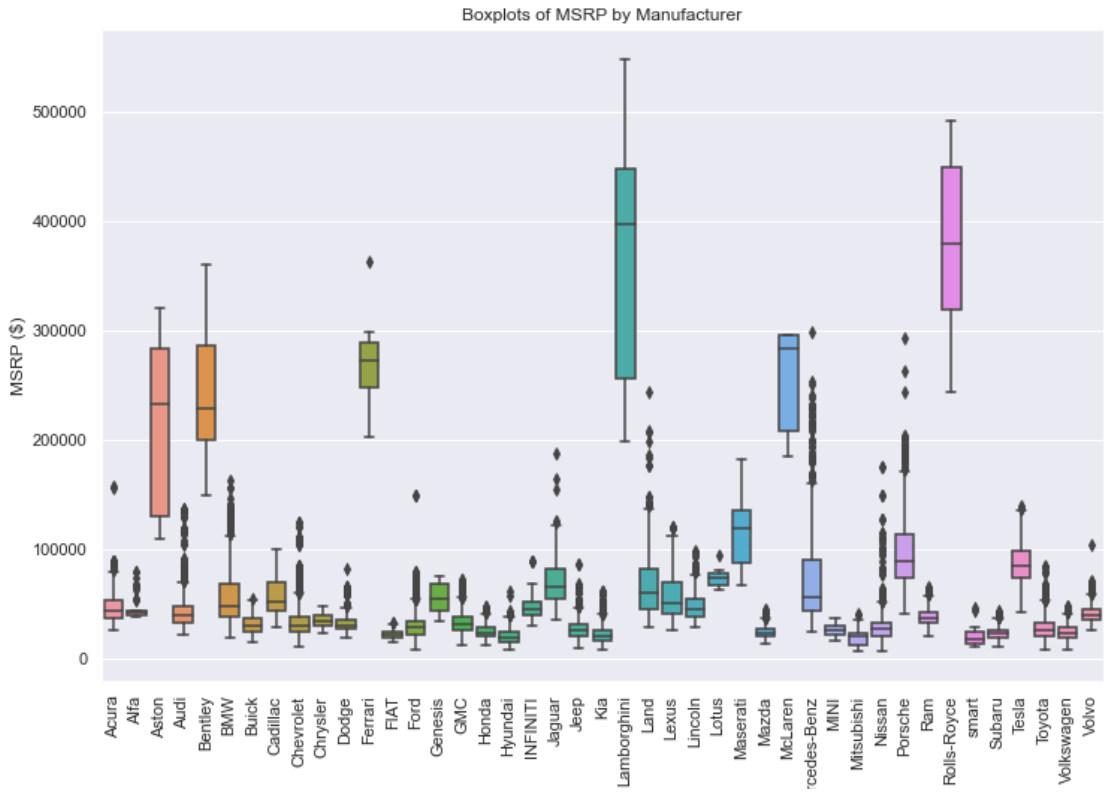

Figure 6: Boxplots of the MSRP distributions for each car Manufacturer found in the dataset.

Preprocessing and Training

Now having finished cleaning, engineering features for, and exploring the dataset visually, it is now time to perform a final few steps before we can move on to creating proper machine learning models using the data. In this case this consists of addressing all the categorical features (since all of the data needs to be entirely numerical in order for machine learning algorithms to be implemented) and splitting the data into training and

testing sets so that it is possible to evaluate our models on unseen data. Lastly the independent, explanatory features need to be scaled or standardized.

As mentioned before the final dataset consists of 57 features and approximately half of those are categorical. Among those some were simple binary yes/no features (a few examples include whether or not the car had traction control, parking aids, or a back-up camera) while the others had a variety of unique values (such as manufacturer, drivetrain type, and engine type). The binary features were dummy encoded while the multi-value features were one-hot encoded in order to preserve user interpretability (e.g. if a car manufacturer was say Porsche I want that information as a feature for model evaluation).

With all the data now fully in numeric form it is truly in a position to be used for modeling. Before that is done the dataset needs to be split into train and test sets. This is a vital step because any predictive model is only useful if it performs well on new, unseen data. The training set is the data that all the models will be built on and learn from while the test data is the data that will be used for the evaluation of the models. First the dataset needs to be separated into X and y arrays where X is the independent features (in this case everything but *MSRP*) and y is our target feature (*MSRP*). In this case I went with a 70/30 percent train/test split of the data. Lastly the independent features (the X_train and X_test arrays) need to be standardized. The data was scaled using StandardScaler() in sklearn. Once again it is important to remember to scale or standardize data after the train/test split to prevent data leakage (don't want any influence on our train data from our test data). So the scaler was fit to the X_train data and then both the X_train and X_test arrays were transformed using that scaler.

Modeling

Since the target variable is a linear, numeric value, regression models are required. Several different regression machine learning models were applied to the dataset to compare their effectiveness and to see which provided the best model for predicting our target feature of new car MSRP. The three primary types of algorithms that were tested were standard *Linear Regression*, an *Elastic Net*, and *Random Forest Regression.* As a baseline for comparison to the actual models a dummy regressor using the mean y_train value (*MSRP*) as a prediction (which turned out to be 37,649.37 dollars) yielded a coefficient of determination (referred to as R^2 value from now on) of -3.40E-05 and a mean absolute error (MAE henceforth) of 16796.45 when evaluated on the test set. Of note since our target feature is in dollars the MAE will also always be in dollars.

1. Linear Regression

I decided to try out the simple, classic Linear Regression algorithm first if for no other reason then to at least have a benchmark to compare to the other models. A Linear Regression model utilizing selectkbest for feature selection (particularly important as the dataset has so many features) in it's default form (wherein k=10, thus only the top 10 features are used in the model). This model upon evaluation on the test set produced a **R^2 = 0.815** and a **MAE = 8,820.44**. Interestingly, this model oddly performed better (in terms of R^2 score) on the test set then the train set. The train set had a R^2 = 0.804. This is quite unusual, but it is possible (maybe the arbitrary train/test split yielded a train set with more variance). To check this cross-validation was done on the train set using the model and it produced a mean and median both of approximately 0.8 (for the R^2 value) and very little variance, so the value does seem to be properly representative. The model had already shown quite a large improvement over the dummy regressor, but I thought it could still be improved. Thus hyperparameter tuning was done next to try and see if any

improvement could be made by trying different k values for selectkbest. GridSearchCV produced a best hyperparameter value of k=50. This new best Linear Regression model's evaluation metrics were **R^2 = 0.897** and a **MAE = 6490.21**. The mean of the y_test actual values is 37,842.99 thus this MAE yields an average error of 17.15 percent. The predicted vs actual values of the MSRP of the cars in the test set for this model can be seen in **Figure 7**. The model overall does pretty well, but especially so for cheaper new cars wherein the large majority exist. It doesn't work quite as well on the more expensive vehicles. In order to better understand the model's predictions and inner workings the feature importances were plotted and can be seen in **Figure 8**. The top 30 features in the model can be seen, and the result seems to mostly align with my expectations. Features such as *Max horsepower, Max Torque, Engine Type_V12* along with various other high end manufacturers are all features that one would expect to be highly positively correlated with price (this also agrees well with the heatmap in **Figure 3**). I was a bit surprised *Model Year* and *Base Curb Weight* weren't more important though and that *Displacement* wasn't even among them.

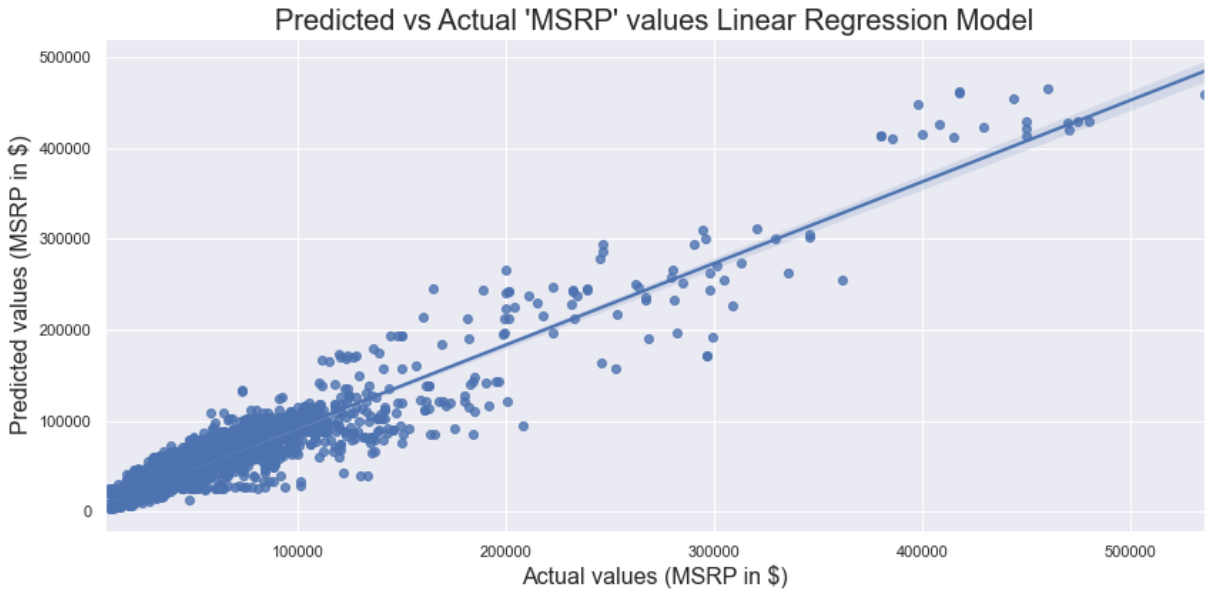

Figure 7: The predicted vs. actual *MSRP* values by the best Linear Regression model.

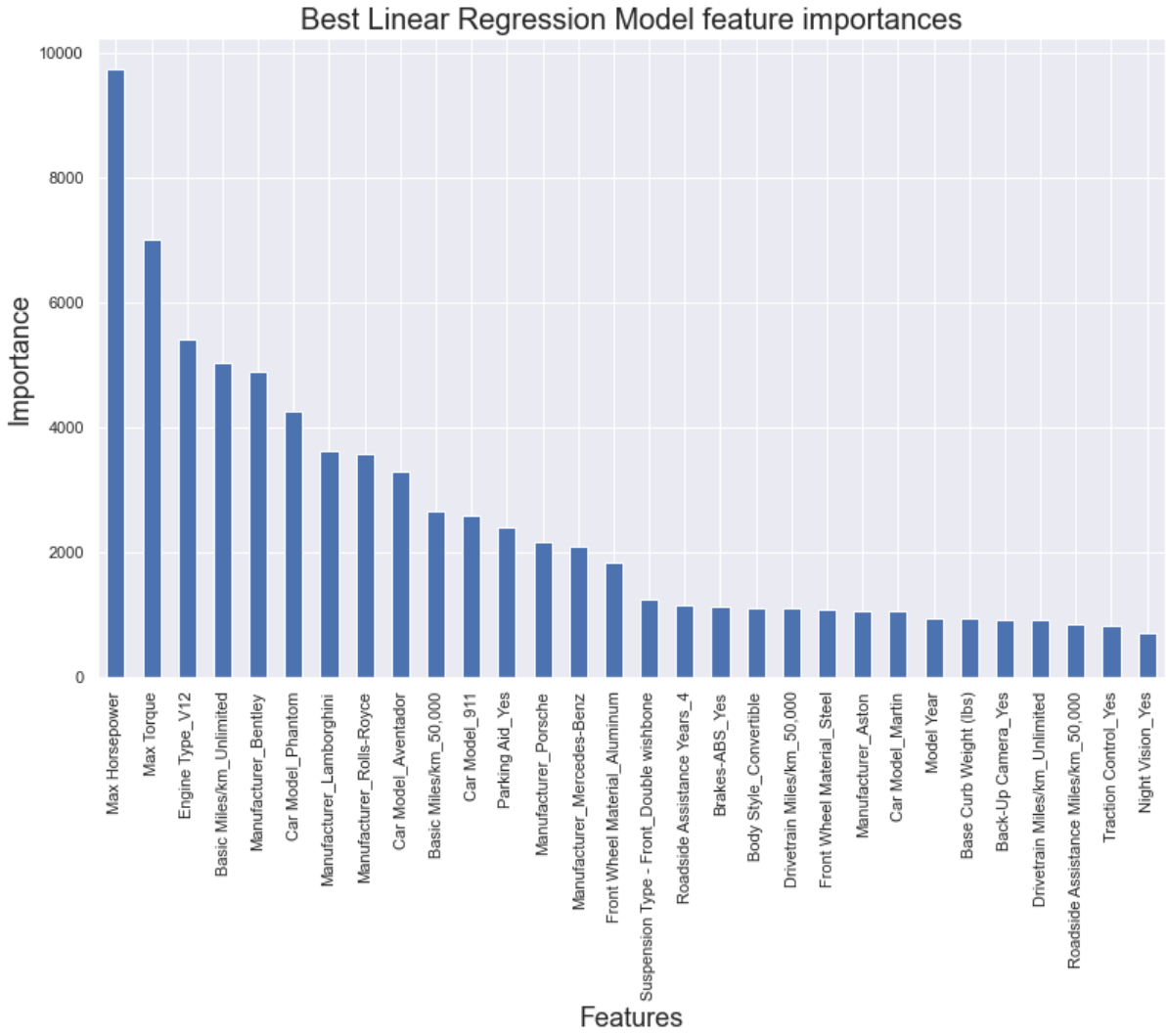

Figure 8: Feature importances for the best Linear Regression model.

2. Elastic Net

Having seen it in use in a few applications, I chose to try out an Elastic Net linear model as a form of regularized regression next on the data. The Elastic Net seemed to be the superior choice over Ridge or Lasso regression, seeing as it is essentially the combination of the two. An Elastic Net model with max_iter = 3000 was instantiated and because I knew that I wanted to tune the l1_ratio hyperparameter from the beginning a GridSearchCV was done to find it's optimum value. The l1 hyperparameter value can lie within the following range: 0 <= l1_ratio <= 1, with l1_ratio = 1 corresponding to an L1

penalty (equivalent to a Lasso linear model) whereas a l1_ratio = 0 corresponds to an L2 penalty and everything else being a linear combination of the two. So I had the grid search over 30 values from 0 to 1 for the Elastic Net model to find the one that produces the best score. The best value ended up being l1_ratio = 1.0, therefore the Lasso model did end up in fact being the best for this application. The model was then evaluated on the test set with the resulting metrics yielded : **R^2 = 0.958** and **MAE = 3630.07**. This MAE yields an

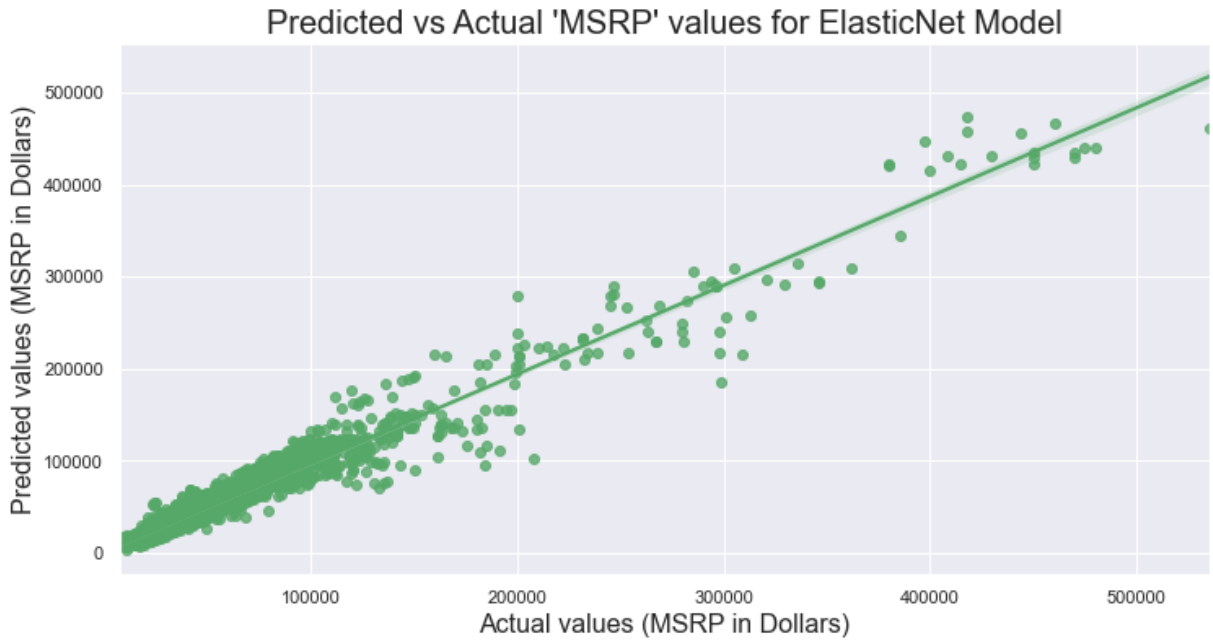

Figure 9: The predicted vs actual *MSRP* values for the best Elastic Net model.

average error percentage wise of 9.59% for our predictions. This was a substantial improvement over the best Linear Regression model in every metric. The MAE was much lower with a value almost half that of the LR model, and the R^2 score improved by 0.06. In **Figure 9** above, as in the LR model, the predicted vs. actual values can be seen for this model. Here the improved accuracy is also evident as the points are more closely clustered.

The feature importances for the Elastic Net model can be seen in **Figure 10**. It is interesting that the top feature *Roadside Assistance Years_Missing*, that is a tough one to fully comprehend. That could be worth further study and closer examination. Nonetheless the

rest of the features that were more important to the model once again align with a strong positive correlation with *MSRP*. Once again I am a bit surprised *Model Year* isn't higher.

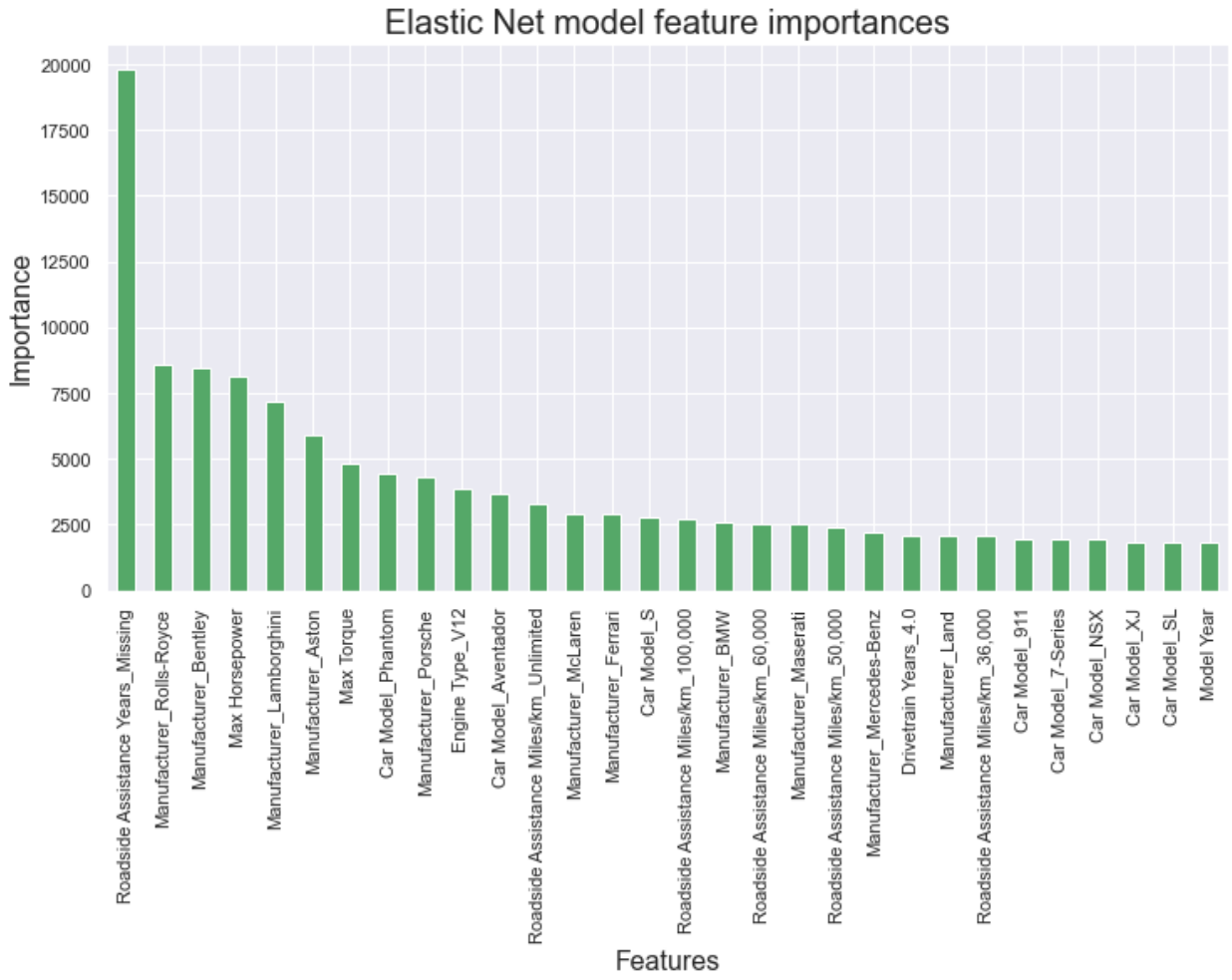

Figure 10: The Elastic Net model feature importances.

3. Random Forest Regression

The final model type tested out was Random Forest Regression. Based on my knowledge and prior experience I did expect this to almost certainly end up performing the best of the three. My hypothesis ended up proving true, as the initial, un-tuned RF model immediately outscored the other models in every evaluation metric. On the test set the RF model gave the following scoring metrics: **R^2 = 0.987** and **MAE = 1,587.04**. The quite low

MAE yields a low average error of only 4.19 percent for the MSRP predictions, less than half that of the Elastic Net model and less than a fourth of the best LR model. So even with no hyperparameter tuning the RF model is substantially better than the other models in every evaluation metric. A plot of the predicted vs. actual values for the test set is shown in **Figure 11**.

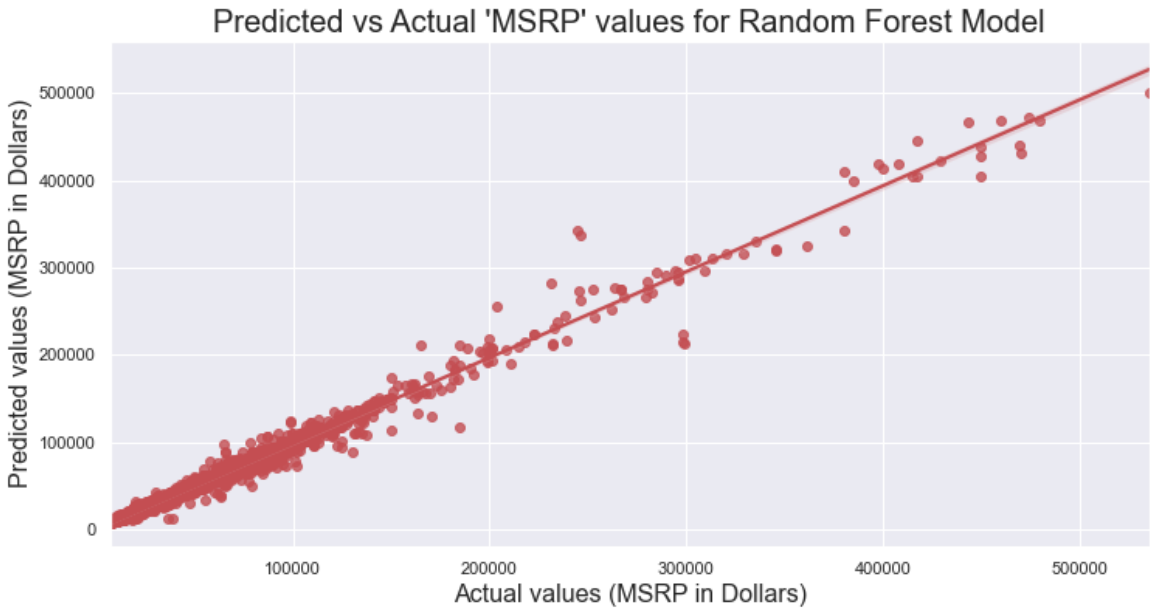

Figure 11: Predicted vs. actual MSRP values for the best Random Forest Regressor Model. The improved accuracy over the other models can be clearly seen.

The RF model is already a pretty accurate model, but it is always important to strive to explore if improvements can be made. That being said, a combination of GridSearchCV and RandomizedSearchCV was done on a variety of RFR hyperparameters deemed to be worth examining (such as max_depth, max_features, n_estimators etc.). I won't go into any more depth here as the hyperparameter tuning yielded a best model that performed essentially identically on the test set as the initial RF model. For details on the hyperparameter tuning see the following link to the Modeling notebook (Modeling). Another downside to the tuned model when compared to the initial model, other then the metrics being equal, was that it took many times longer to fit and predict.

So coming back to the initial RF model with it being the best RF model it was time to examine it a bit more. The feature importances of the model are available in **Figure 12**. The RF model clearly heavily focused on just a few features, particularly *Max Horsepower* and *Basic Miles/km_Unlimited.* It is impressive how accurate the RF model is in comparison to the other models. In the end it is the easy choice to be the final model.

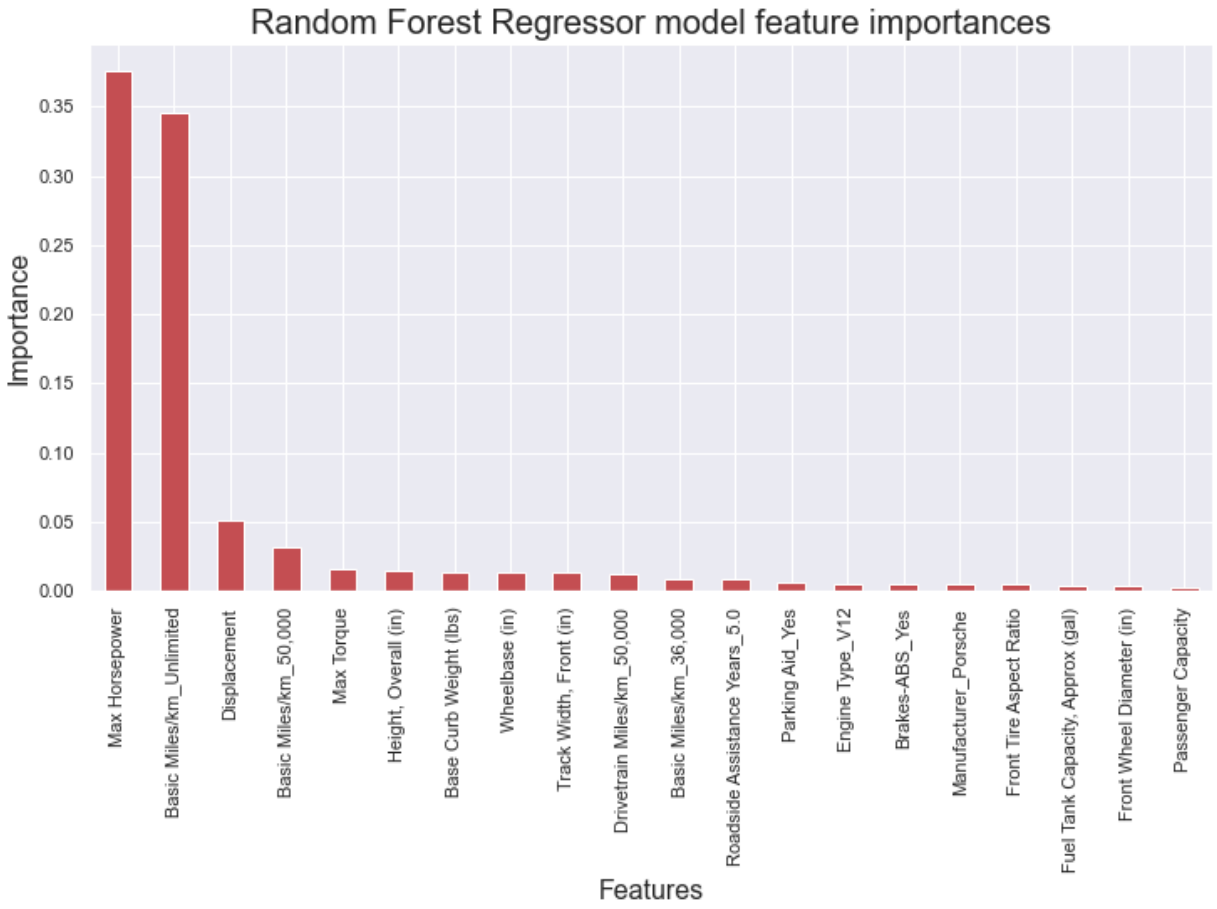

Figure 12: The best Random Forest Regressor model's feature importances.

Summary and Recommendations

In this project the goal was to use a dataset containing features of new cars available for sale in the USA from 1991 to 2019 to try and create a regression machine learning model to predict the MSRP (price in dollars). Wrangling, feature extraction, cleaning and EDA were performed to prepare our data for modeling and to better understand the relationship between the various features. The data was properly pre-processed and split into train and test sets. Several different machine learning algorithms were created, trained on the train set and evaluated on the test set. The three different models tried were Linear Regression, an Elastic Net, and Random Forest Regression. All three were iterated on several times and that included hyperparameter tuning. The evaluation metrics for the best model in each category is summarised in **Table 1**. The final model chosen was the Random Forest Regressor as it performed better in every metric and was consistent. As the metrics show it performed quite well.

A client such as a manufacturer could use a model like this to help set the price of their new cars, or maybe more likely to project competitor's prices which could therefore influence theirs. On the other hand, a client such as an institutional investor could use it in conjunction with other models to make smarter investments in or projections related to manufacturers or other industry related companies such as part suppliers.

While the model ended up performing well on this dataset there is always room for improvements. For one the dataset only contained data on cars up to 2019, it would be nice to have more up to date data, particularly in the case of future projections. Also the dataset contains no sales data, it would be quite interesting and useful to see how various features and MSRP related to how well cars actually sold. The dataset also contains no information on the actual costs of the cars, though obviously this is complicated (of course the companies keep this information internal, this data is pretty vital for positioning the MSRP of a company's own cars for financial viability).

Model Metrics Summary

Model	R^2 score	MAE (in US dollars)	Relative Avg. Error (percentage)
Linear Regression	0.897	6,490.21	17.15
Elastic Net	0.958	3,630.07	9.59
Random Forest Regression	0.987	1,587.04	4.19

Table 1: The evaluation metrics for the best models of each category. The relative avg. error is the percentage of the MAE to the median of the actual values.