**Used Car Price Prediction Model**

**Final Semester Project (STA-6704) – SUMMER 2021**

**Syed Muhammad Sabih**

**Final Project Write Up**

**PROJECT STATEMENT**

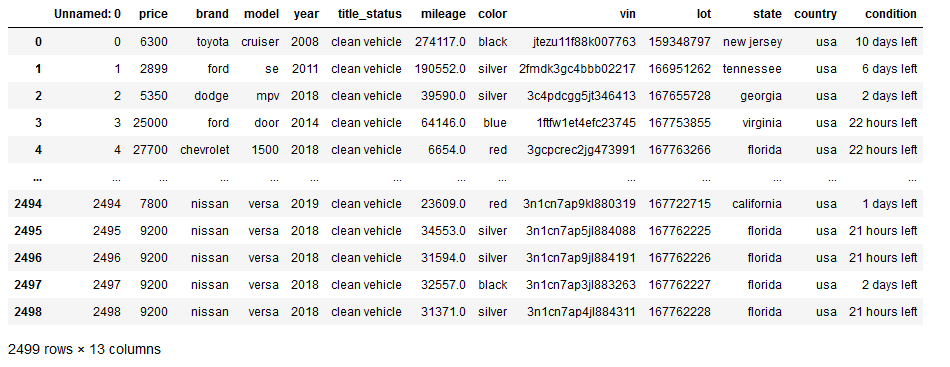
For the US Cars Dataset, the analysis for this project pertains to build a Machine Learning Model that predicts the price of the car based on the imperative features.

**THE DATASET**

The dataset used for the analysis in the project is from the Kaggle repository and was scraped from auctionreport.com. The dataset includes information about 28 brands of clean and used vehicles for sale un the US. The dataset comprises of 11 features:

* Price: The sale price of the vehicle
* Year: Vehicle registration year
* Brand: Brand/Make of the vehicle
* Model: Model of the vehicle
* Color: Color of the vehicle
* State: The location where the vehicle is for sale
* Mileage: Miles traveled by the vehicle
* VIN: Vehicle identification number
* Title status: Clean or salvage title
* lot: lot number from the manufacturer
* condition: time

The dataset in the .csv format was loaded into the Python environment and the raw form of data was visualized.



At this point, it can be seen that there are 2,499 data points and 13 variable/feature columns including the target variable *price*. The *Unnamed: 0* column looks like an index column and can be removed in the future when preparing the data. The features other than the target variable comprise of numeric and categorical predictors that will be used in the analysis.

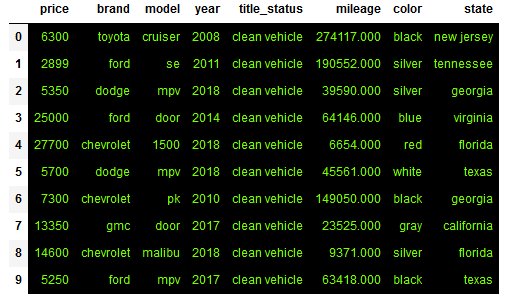
**EXPLORATORY DATA ANALYSIS**

The analyses and predictive modeling algorithms for this project will be implemented using the Python scripting language. All the visualizations in this project are produced using matplotlib and seaborn library in Python. All the machine learning algorithms used in this project are from the sklearn library in Python.

Removing any irrelevant columns:

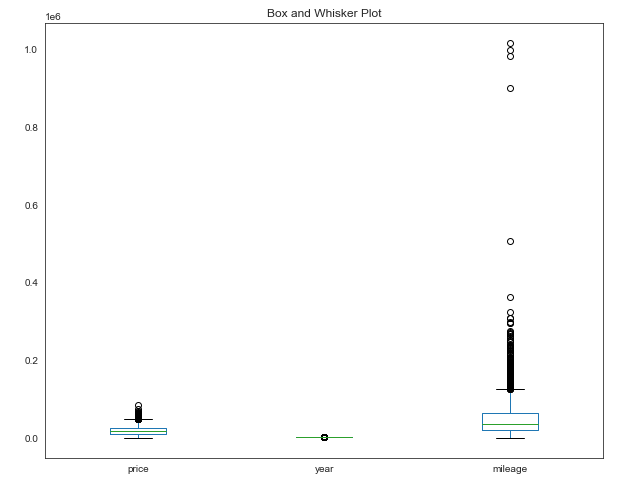
Based on my experience with observing variation in car prices, the price usually depends on the brand, model, year, title status, mileage, color, and the location of the car. For the purpose of building a model that will predict a used car’s price, any unwanted columns that would not contribute to the price of the car were removed from the dataset.   
After removing the unwanted columns, here are the features that will be used in the analysis going forward:

*price, brand, model, year, title\_status, mileage, color, and state.*



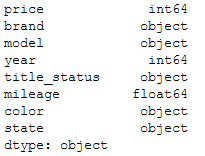
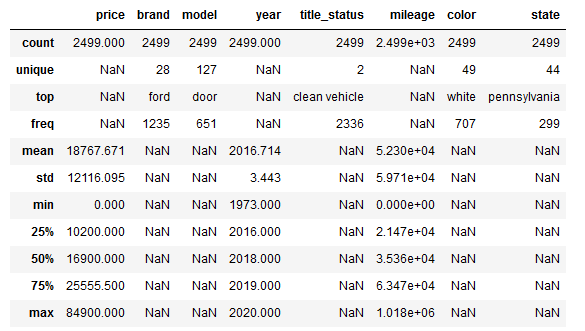
Initial view of the numerical variables:

The following box and whisker plot shows an initial visualization of the numeric variables *price, year, and mileage.* Since this is only an initial view of the raw data, substantial insights cannot be produced at this point in the analysis. Visualizations would make more sense after further investigation of the variables and the dataset.



Investigating Numeric and Categorical Variables:

The types of the variables were investigated and a table for descriptive statistics was produced to make sure that the variables that are assumed to be numeric and categorical are indeed numeric and categorical.

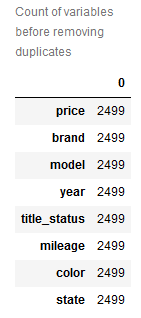


From the results above, it can be seen that all categorical predictors such as *brand, model, title\_status, color,* and *state* with different levels do not produce an output in the descriptive statistic measures due to having character variables. On the other hand, *price, year*, and *mileage* produce an output for descriptive statistic. However, the measures of dispersion for *year* can be misleading and would be treated as numeric values. For example, the mean for *year* 2016.7 in predictive modeling would be treated as a numeric value rather than how old or new the car is relative to the current year. This is an important insight, which can be used to modify the year variable in the analysis later on.  
At this point, *price* and *mileage* can be treated as numeric variables, whereas *brand, model, title\_status, color,* and *state* can be treated as categorical variables. The feature *year* will be treated as a numeric variable after it is modified.

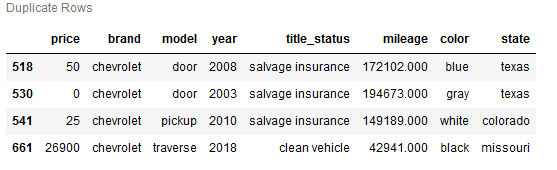
Checking for Duplicate Datapoints in the dataset and removing them:

After the investigation of the variables, further in the investigation of the dataset, the dataset was investigated for duplicate values. This redundancy in the data needed to be reduced.

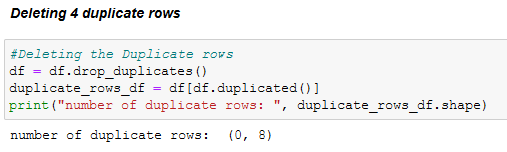
First, the count of the variables was produced:



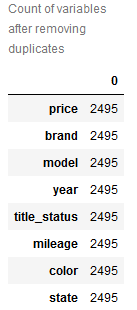
Then, the duplicate rows were located:


After locating the duplicate rows, they were removed from the dataset:



Finally, the count of the variables in the dataset after removing the duplicate values was produced to ensure that the duplicate rows were removed:

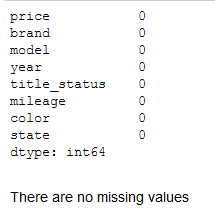


New shape of the dataset is as follows:



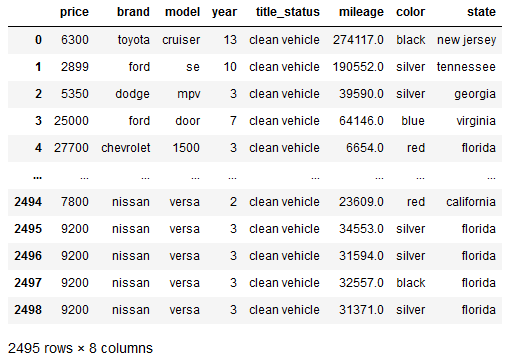
Investigating any missing values in the dataset:

After investigating any duplicate values in the dataset, the dataset was checked for any missing values. There were no missing values in the dataset.



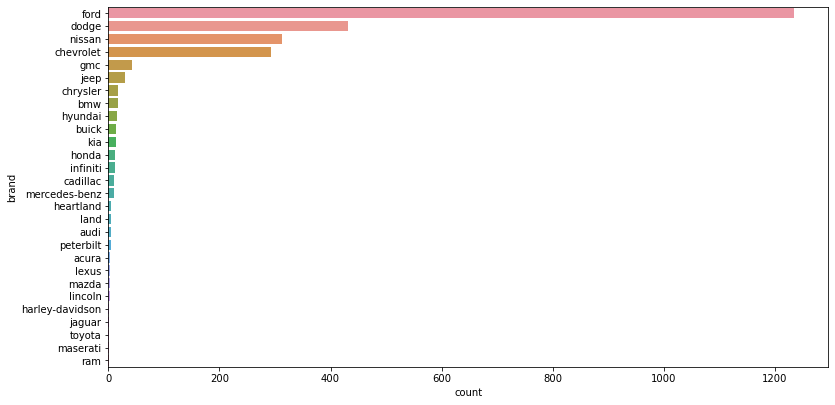
Modifying the *year* variable:

At this point, the variable *year* can be modified to be a numeric variable where it reflects on a numeric scale whether the car is new or old. The new year variable was produced and now if a car *year* is 2008, the new value would be 13, which implies that the car is 13 years old. The output is as follows:



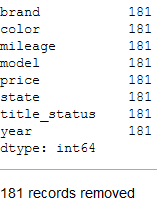
Investigating the brands of the vehicle:

After the dataset was somewhat clean and ready to be analyzed, the top brands of the vehicles by count were to be investigated. The rationale behind was to visualize if there are more vehicles of one brand in the dataset than the other. If this was the case, then the analysis would be biased and other brands which do not have a substantial effect on the dataset would alter the analysis. The visualization of the brands in the dataset was produced:



The dataset had an overwhelming number of Ford vehicles and the only substantial number of brands in the dataset seemed to be Ford, Dodge, Nissan, Chevrolet, and GMC.

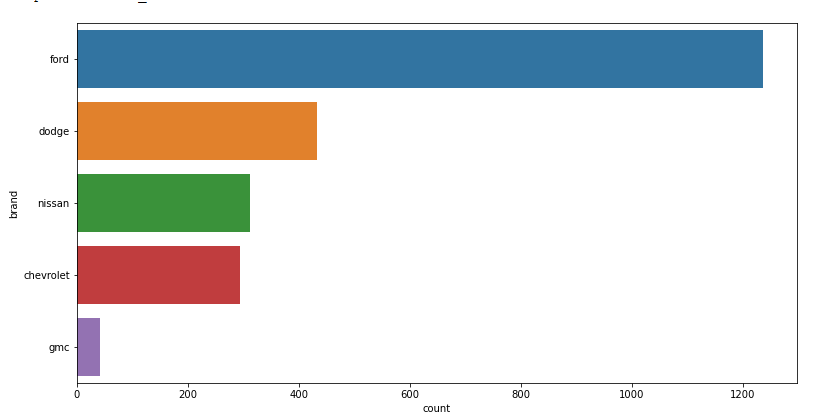
In order to tackle this issue, the datapoints of the top 5 brands were included and the rest of the brands were removed from dataset. The number of records removed was produced:



The new dataset had 2,314 datapoints. The removal of the brands that were not as influential to the dataset was not too drastic as only 181 records were removed from the 2,495 records. The visualization of the new dataset was produced:



The visualization of the brands after removing the irrelevant brands was produced:

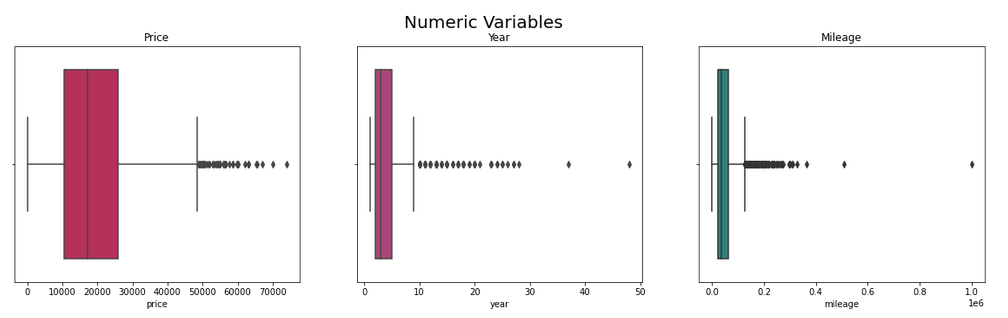


The analysis can be more efficient without the brands that did not contribute a lot to the dataset. Out of 28 brands, the top 5 brands seem reasonable to work with for the analysis to predict used car price. This also provides an insight to the analysis that brand is not going to be the most useful predictor in the prediction of price. This can be addressed later on in this analysis.

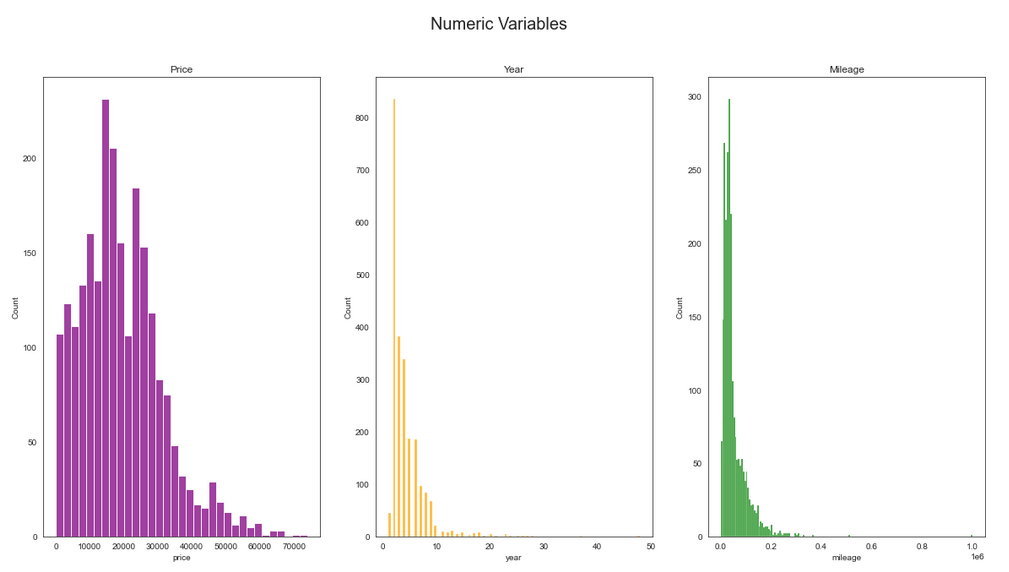
Visualizing the Numeric Variables:

At this point in the analysis, the visualizations for *price, year, and mileage* were produced to investigate the spread of the data:

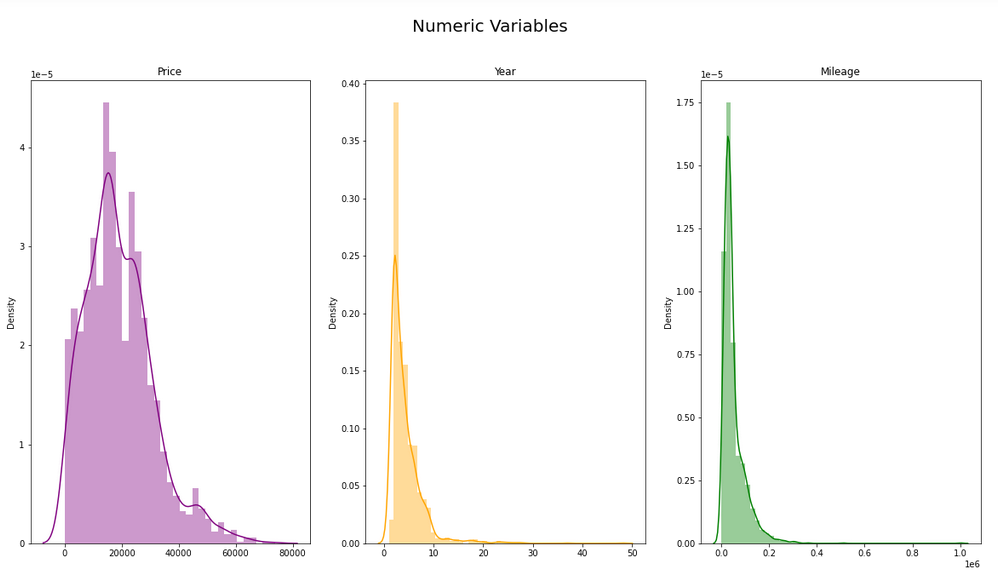
*Box and Whisker Plot*



*Histogram*

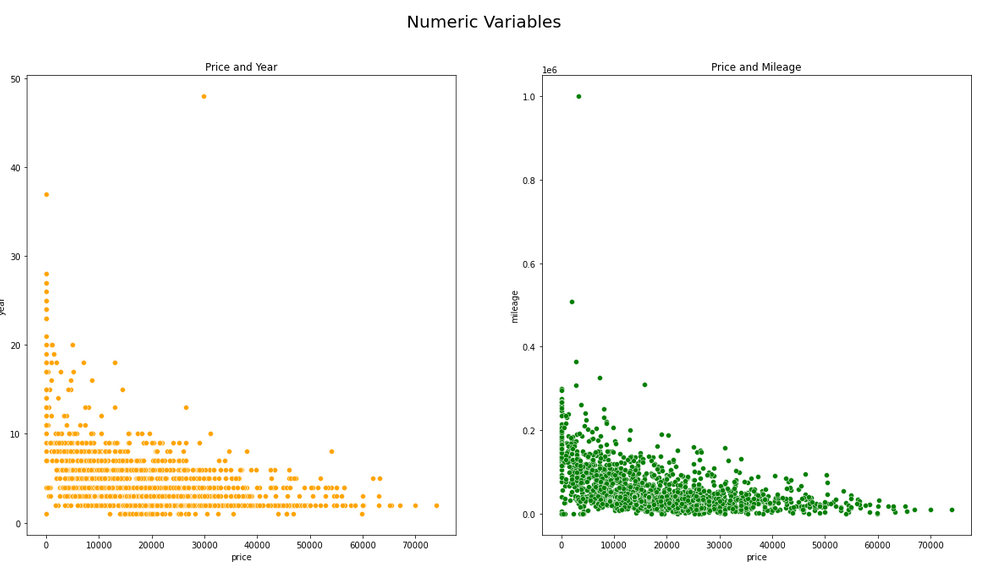


*Density Plot*



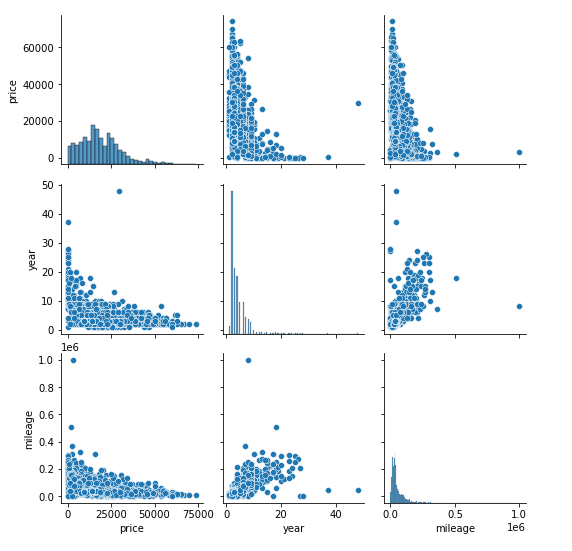
The distribution for all three variables shows the data is skewed to the left with some outliers towards the right side of the graphs. This visualization makes the analysis stronger and now the assumption can be made that the lower *year* value (newer model of the car) is directly related to a lower *mileage* car. The *price* values are more spread since the *price* varies depending also on the *model, brand,* and other factors, however, we can see the relationship between *price* and *mileage* and *price* and *year* in the bivariate visualizations below:

*Scatterplot*

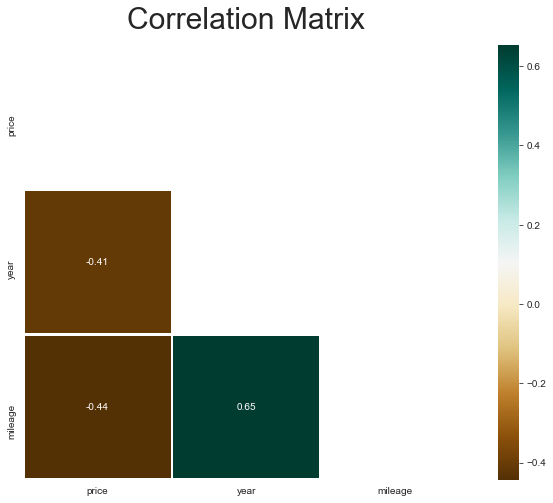


In the scatterplots above, we can see that the *price* values tend to stay on the average to a higher side when the car is older, however, this assumption is still hard to make as there are other factors that are also affecting the *price* of a used car. We can also see from the above scatter plots that the *price* values seem to be on the average to a higher side when the car has less *mileage*, however, once again this assumption is hard to make due to other factors affecting the *price*, for example if a car has a *title\_status* of salvage and had almost no *mileage* on it, the price would be low even though it’s a car that was barely driven.

*Pair Plot*



*Correlation Matrix*

**

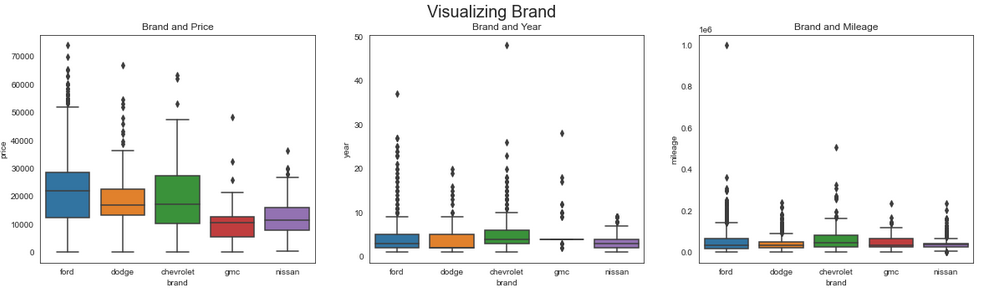
From the pair plot and the correlation matrix, a very important insight that we see is the relationship between the *year* and *mileage*, which makes sense in the real world where when a car is newer, it has not been driven that much and as old as it gets, the more it gets driven.

The negative correlation makes sense in the real world where the *price* tends to increase or decrease based on the decrease or increase of the *age* or *mileage* of the car.

Visualizing the Categorical Variables:

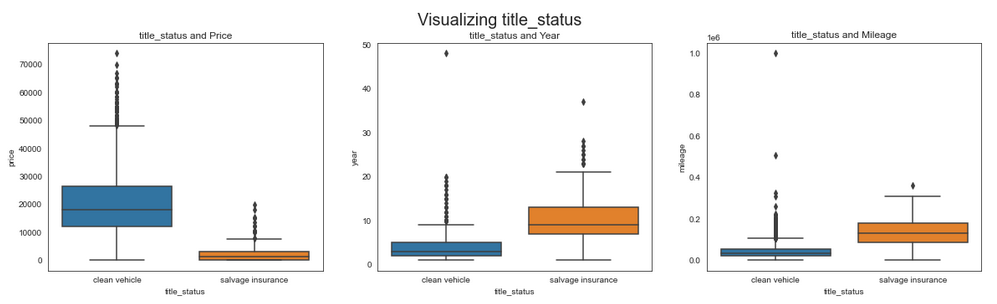
After visualizing the numeric variables, the visualizations for categorical variables could be produced to gain insight on how the numeric variables are spread across the categories:

*Box and Whisker Plot – Brand*



Looking at the box and whisker plots for brand, we can see that price, year, and mileage were spread almost similarly across the brands of the vehicles. The outliers and the variability were deemed important at this point since some anomalies might contribute positively to the analysis as depending on car *model, color,* or *state* the prices might vary. A little variation will take anomalies into account, as we want to predict all US car prices, and not only the luxurious or average day to day cars.

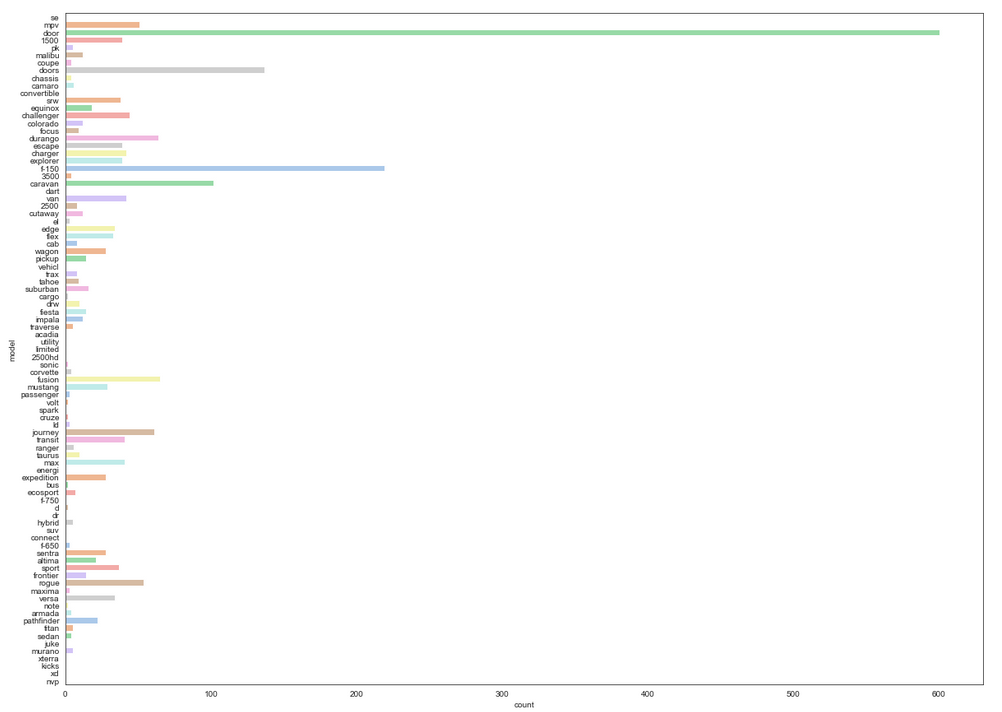
*Box and Whisker Plot – Title Status*



Visualizing the box and whisker plots for title status of a vehicle provided some very useful insights:

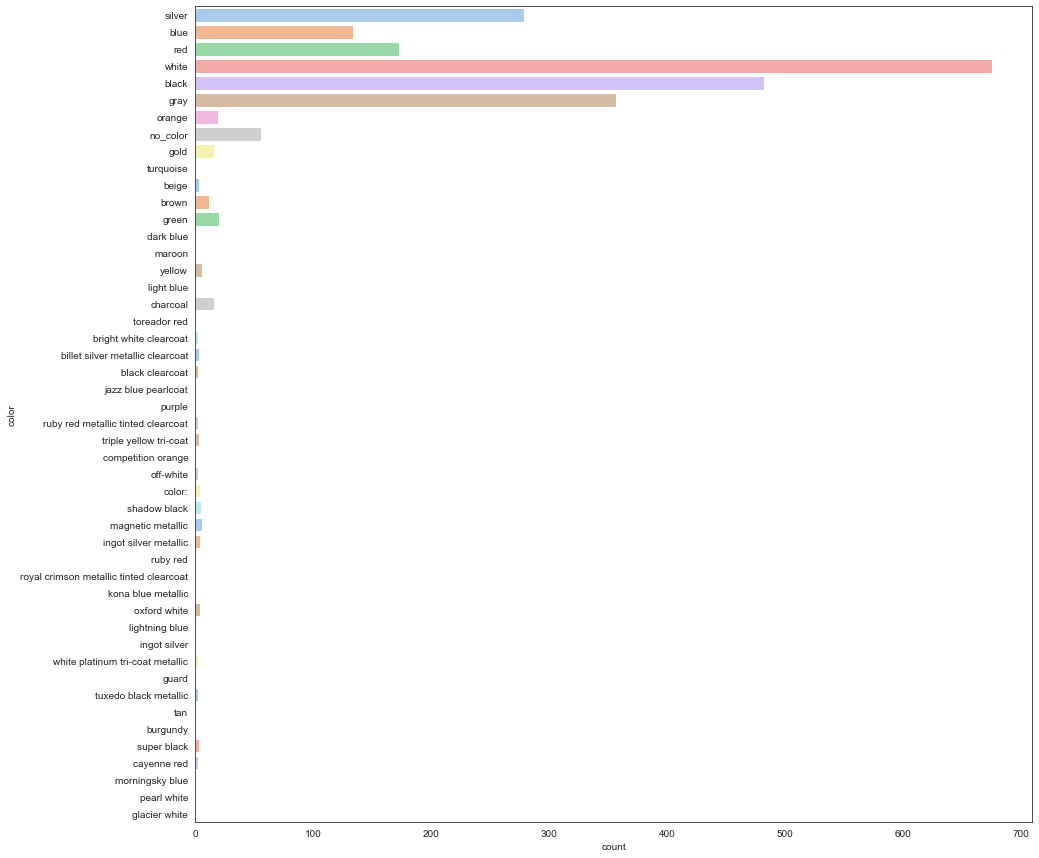
* The vehicles with clean status had the *price* values spread across the higher values, whereas the salvage insurance vehicles were mostly spread across lower *price* values.
* The vehicles with clean status were mostly spread across the newer *year* of the vehicles whereas the salvage vehicles seemed to be spread on older vehicle *year*
* The clean *title\_status* vehicles were on average spread on low *mileage* vehicles while the salvage vehicles were spread on high *mileage* vehicles.

*Count Plot – Model*



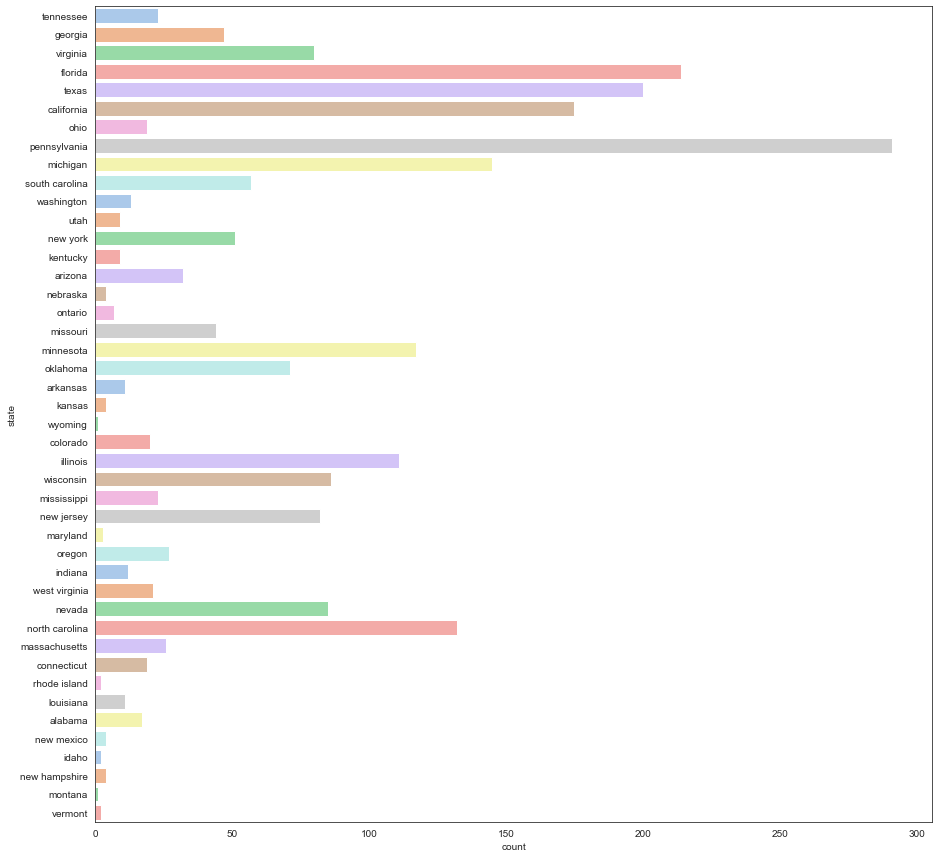
We can see that the vehicle *model* “door” has an overwhelming count in the dataset. Another vehicle *model* “F-150” has a high count in the dataset, whereas the other models are spread evenly.

*Count Plot – Color*



After looking at the visualization for the *color* variable, we can see the majority of the vehicles are white, black, gray, silver, red, and gray. This further shows us that the *color* could also be a useful feature for the *price* in the analysis.

*Count Plot – State*



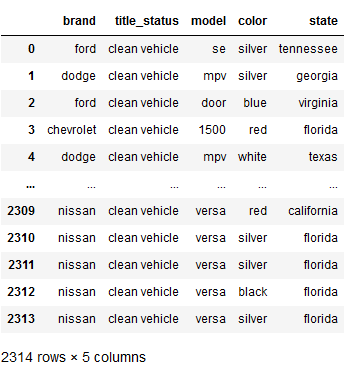
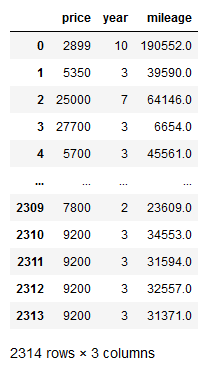
After looking at the visualization for *state,* we can see that the vehicles are mostly spread around the state in terms of count and the largest number of vehicles are in Pennsylvania.

**DATA PREPARATION**

After visualizing the dataset and deciding the features that need to be used in the analysis and the inclusion of outliers as they might contribute to the analysis, we can now prepare the data for predictive modeling.

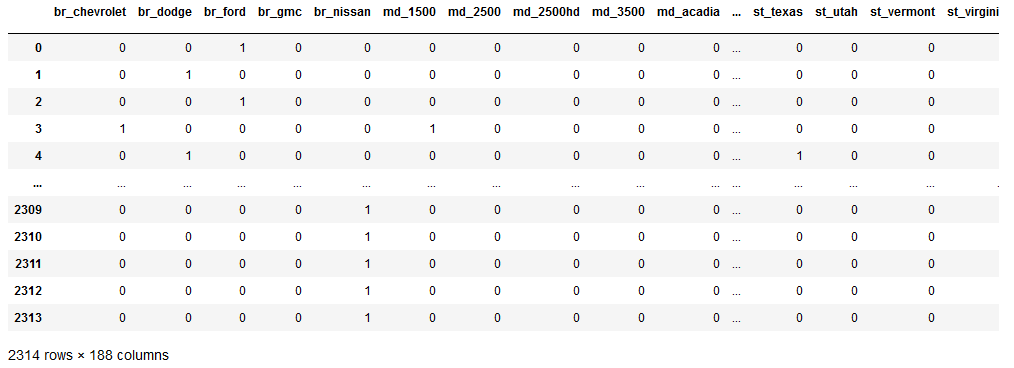
Separating the Categorical and Numeric Variables:

At this point, the data needed to be split in to two dataframes with categorical and numeric variables so that the categorical variables can be prepared for modeling purposes.

OneHot Encode the Categorical Variables:

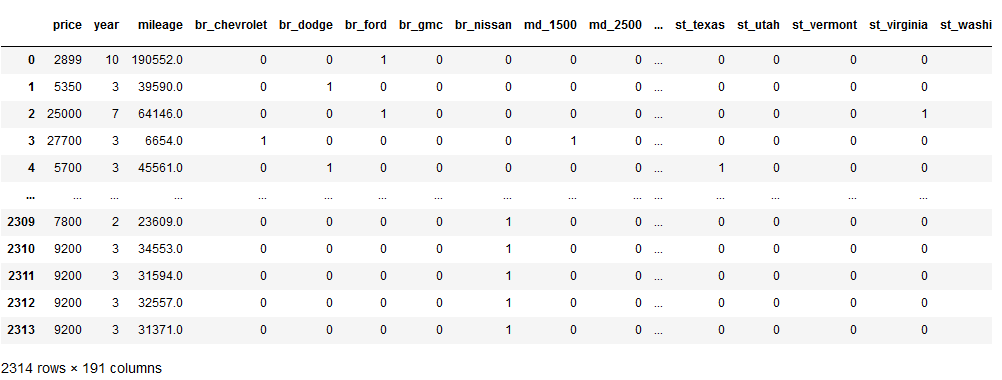
The categorical variables were encoded using OneHot Encoder, which would derive the categories based on the unique values in each feature and make dummy variables with binary values. The output below shows how the Categorical variable levels were now binary features:



Instead of having 5 categorical predictor variables, now the dataset had 188 categorical predictor variables, which were now binary numeric.

Concatenate the numerical and Categorical dataframes:

After the encoding of the categorical predictors, the dataframe with categorical encoded variables and the dataframe with numeric variables were merged again to have a final dataframe that can be used for modeling. The snippet of the resulting dataframe is below:

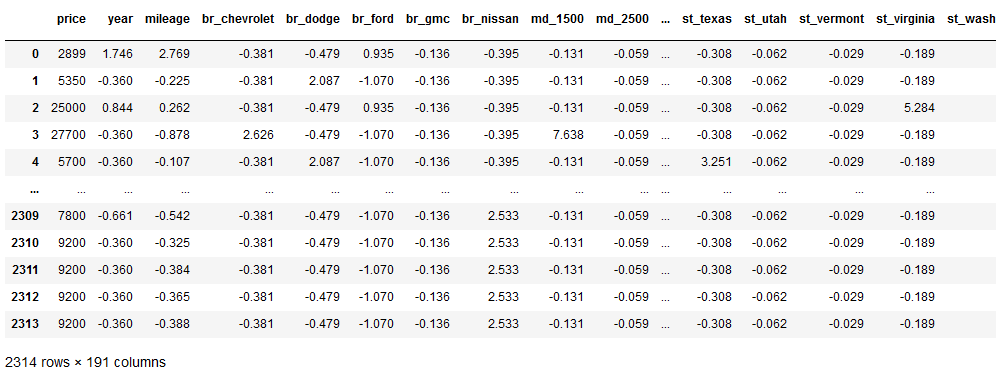


The dataframe had 190 predictor columns and 1 target column.

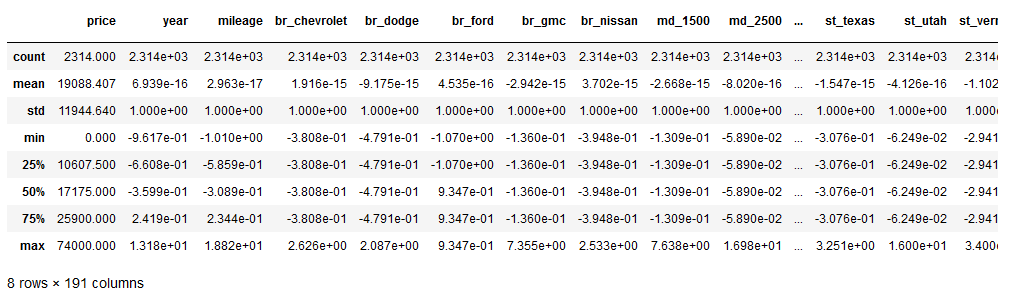
Scaling the input variables:

Another step in preparing the data for modeling is scaling the data. The data was scaled using StandardScaler(), which standardizes the features by removing the mean and scaling the values to unit variance.  
The resulting dataframe and the descriptive statistics for the resulting dataframe are shown below:

*Snippet – Dataframe*



*Descriptive Statistics – Dataframe*



At this point, the data is prepared for Predictive Modeling.

**PREDICTIVE MODELING**

For modeling, the dataset did seem to have a high number of features, however, there are algorithms that take care of selecting features as one of the processes when modeling. The approach for modeling would be to create a model that is substantially fit when making predictions for the *price.* A substantial R-Squared value to show a substantially fit model is usually between 0.75 and 0.80.   
The base models would be constructed without any feature selecting techniques and any feature selecting techniques might be selected depending on the algorithm performances.

Splitting the data in to Train and Test datasets:

The scaled dataset with the unscaled target variable was split into 70% test dataset and 30% test dataset.

Spot-checking a few Regressor Models with Cross-validation on the Training Dataset:

The models to be fitted for cross-validation on the ***training*** dataset were ***Random Forest Regressor, XGBoosting Regressor, K-Nearest Neighbors Regressor, Decision Tree Regressor,*** *and* ***Gradient Boosting Regressor***. The scoring method for cross-validation was set to r-squared and the cross-validation method was KFold() with 10 folds, shuffle, and a random state of 7. The results of R-Square values are shown below:

Random Forest: 0.660

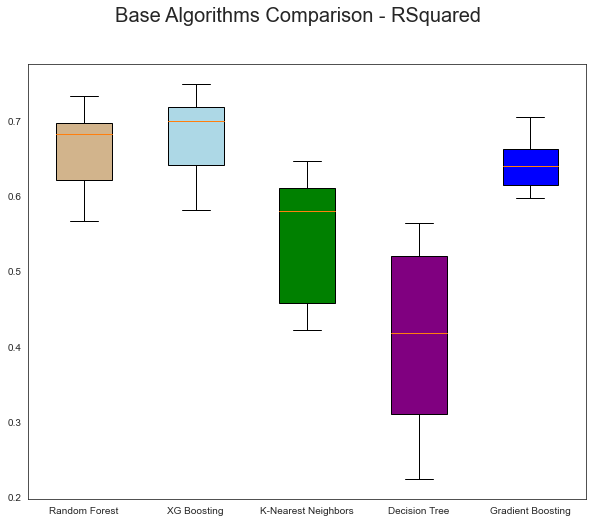
XG Boosting: 0.680

K-Nearest Neighbors: 0.543

Decision Tree: 0.408

Gradient Boosting: 0.642

The results can also be visualized to see the overall performance of the base algorithms when they are implemented on the training dataset and evaluated using cross-validation:



The results were evident that *RandomForestRegressor*() and *XGBRegressor*() were showing performing much better than the other algorithms. The reason for this could be very easily because of the feature selection performed by these algorithms using tree-based strategies.

Evaluating Random Forest and XGBoosting on making predictions:

After evaluating the algorithms using cross-validation on training datasets, we can take the 2 top performing base algorithms and evaluate them based on how they make predictions.  
Base algorithms *RandomForestRegressor*() and *XGBRegressor*() were fitted on the *training* dataset to predict the *price* variable. The predicted *price* and the real *price* from the *testing* dataset were then used to measure the performances of these algorithms:

*RandomForestRegressor():*

Mean Absolute Error: 3957.5076115107913

Mean Squared Error: 36840561.981187195

Root Mean Squared Error: 6069.642656795142

R\_2 = 72.93 %

Variance score = 72.93 %

*XGBRegressor():*

Mean Absolute Error: 4016.4639647422077

Mean Squared Error: 35517633.58071563

Root Mean Squared Error: 5959.667237414823

R\_2 = 73.90 %

Variance score = 73.90 %

We can see that both algorithms almost perform similar, however, the R-squared value for XGBoosting is better than Random Forest.

Hyperparameter Tuning the XGBRegeressor() algorithm to see if better results can be produced:

Since XGBoosting performed better than Random Forest in making predictions for the testing dataset, we can tune the hyperparameters for XGBRegressor() using GridsearchCV() to see if we can improve the performance for the algorithm. The performance measure is based on improving the r-squared value which provides an insight for a better fit model since we want to reduce the error, which is the distance from the actual data points and the predicted datapoints (residuals).

After the hyperparameter tuning using GridSearchCV(), the best parameters were:

{'colsample\_bytree': 0.7,

'learning\_rate': 0.1,

'max\_depth': 5,

'min\_child\_weight': 1,

'n\_estimators': 500,

'objective': 'reg:squarederror',

'subsample': 0.95}

The XGBRegressor() model was fit on the *training* dataset using the parameters shown above and evaluated on making predictions for the *testing* dataset. The predicted *price* and the real *price* from the *testing* dataset were then used to measure the performance of the tuned XGBoosting algorithm:

MSE: 33511036.622043252

R2: 75.37912108608108 %

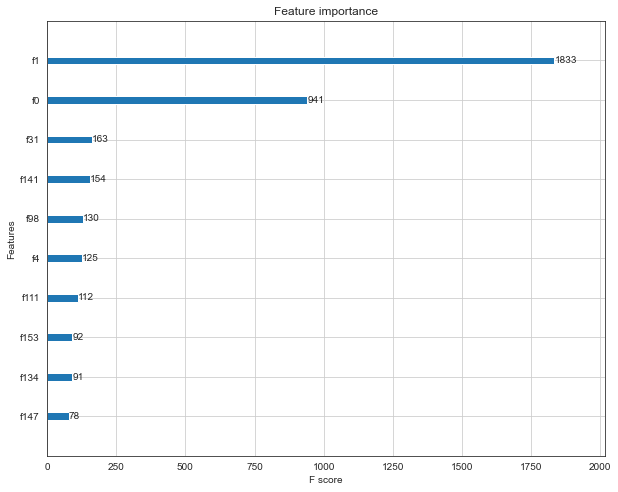
MAE: 3874.3044465428634

RMSE: 5788.871791812568

The r-squared value increased from 73.90% to 75.4%, which is a significant improvement and produces a substantially fit model.

Important features for tuned XGBoostRegressor():

After finding a substantially fit model for XGBoosting, a plot for important features was improved to see which features were the most important in producing these results:

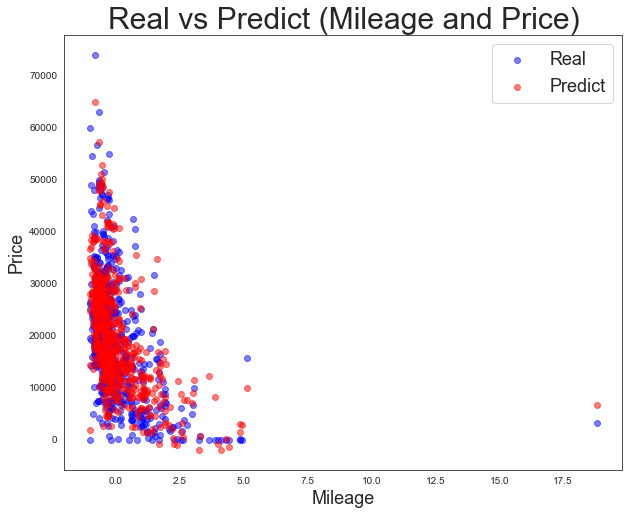


We can see that f1 and f0 are the most important predictors in the model which predicts price. F1 is the mileage predictor while f0 is the year of the vehicle. This provides a very imperative insight in our analysis which tells us that mileage and year are the most important features when looking at a used car’s price.

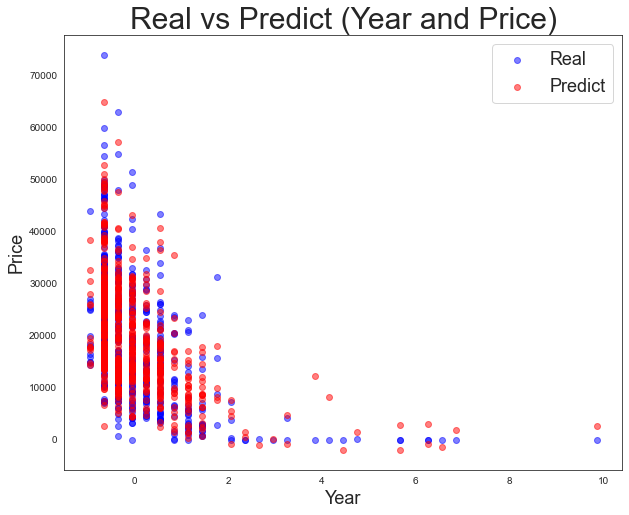
Visualizing the comparison between Real and Predicted Values:

To have a better idea visually of a model that has an r-squared value of about greater or equal to 75%, we can plot the real *price* data points and the predicted *price* datapoints to see how accurate the model is predicting the *price.* We can use the most important predictors; year and mileage to visualize the results.

*Scatter plot – Mileage and Price*

**

*Scatter plot – Year and Price*

**

The predicted data points seem to be inside the bounds of real data points, which tells us that the model is performing well in predicting the *price* values.