Car Price Prediction

Structure of Presentation:

Structure of Presentation:

Introduction

Problem Formulation

Dataset Description

Exploratory Data Analysis

Methodology (Data Cleaning, Encoding)

Machine Learning Model Comparison

Conclusions

Introduction

Industry Overview: The car industry generates $2.86 trillion in revenue annually with about 77 million cars sold each year.

Pricing Challenge (Motivation): Accurate pricing is crucial as buyers avoid overpriced cars and sellers avoid underpricing.

Evolution of Pricing: Traditionally handled by experts, now enhanced by extensive data analytics.

Our Approach: Utilize a comprehensive dataset from an online car marketplace to develop a machine learning model that predicts prices based on key variables.

Problem Formulation

To predict the car price for retail companies, based on Manufacturer Suggested Retail Price (MSRP), we have to consider the questions chronologically.

1. Look at the car dataset and see the counting for each variables?
2. Are there any missing values in each variable of the dataset?
3. How is city mpg and highway mpg correlated using box plot? Address the outliers as well when comparing and for future machine learning test.
4. Which features are heavily correlated to one another by using heat map?
5. Check for NULL missing values, and make sure the data is filled or drop the variable so the machine learning will not encounter any problems?
6. Create a new column for Present Year (2024) and plot the Years of Manufacture, this variable will be considered as well for the machine learning model?
7. Check the data again for variables that has catergorical values, we have to address them by changing to numerical values for the machine learning to analyse?
8. Plot the different machine learning model and check which prediction model has the best result, in terms of lowest mean squared error (MSE) and lowest mean absolute error (MAE), when also checking for which range of values work best?

Dataset Description

https://www.kaggle.com/datasets/CooperUnion/cardataset

Dataset Size: 11,914 x 16

- Make: 48 unique manufacturers
- Model: 915 different models
- Year: 28 unique years of manufacture
- Engine Fuel Type: 10 types of fuel
- Engine HP: 356 variations in horsepower
- Engine Cylinders: 9 different cylinder counts
- Transmission Type: 5 types of transmissions
- Driven Wheels: 4 wheel configurations
- Number of Doors: 3 different door counts
- Market Category: 71 market segments
- Vehicle Size: 3 size categories
- Vehicle Style: 16 styles
- Highway MPG: 59 levels of highway fuel efficiency
- City MPG: 69 levels of city fuel efficiency
- Popularity: 48 levels of brand popularity
- MSRP: 6,049 price points

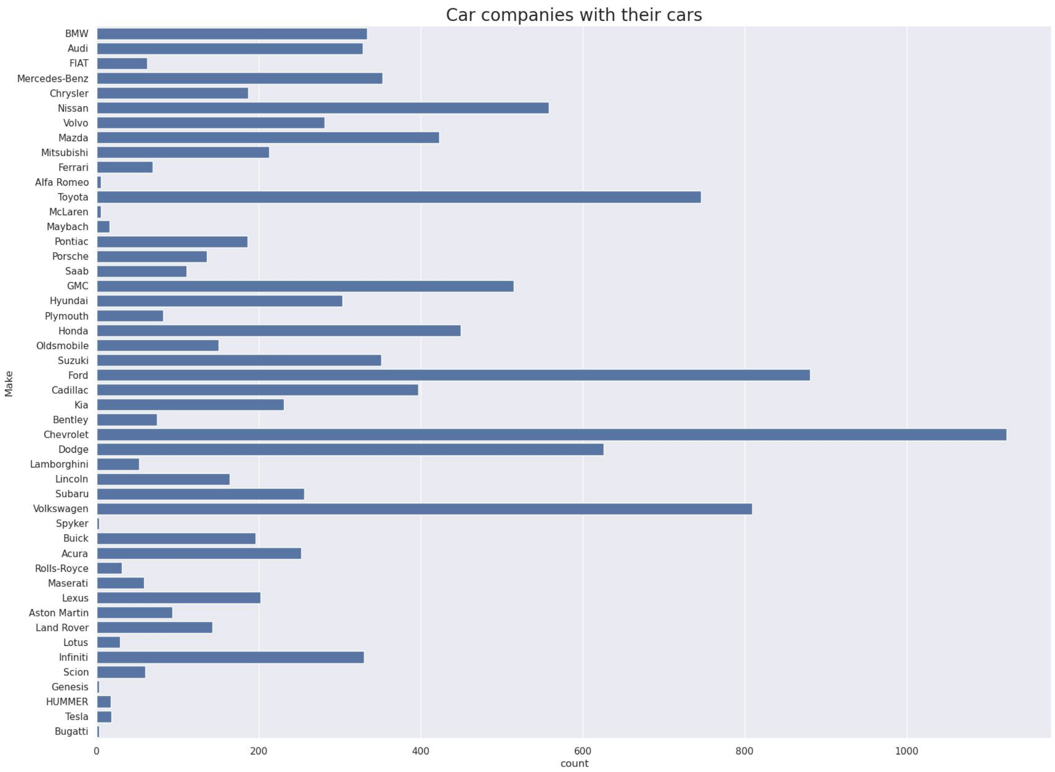
Snapshot of dataset

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

Exploratory Data Analysis

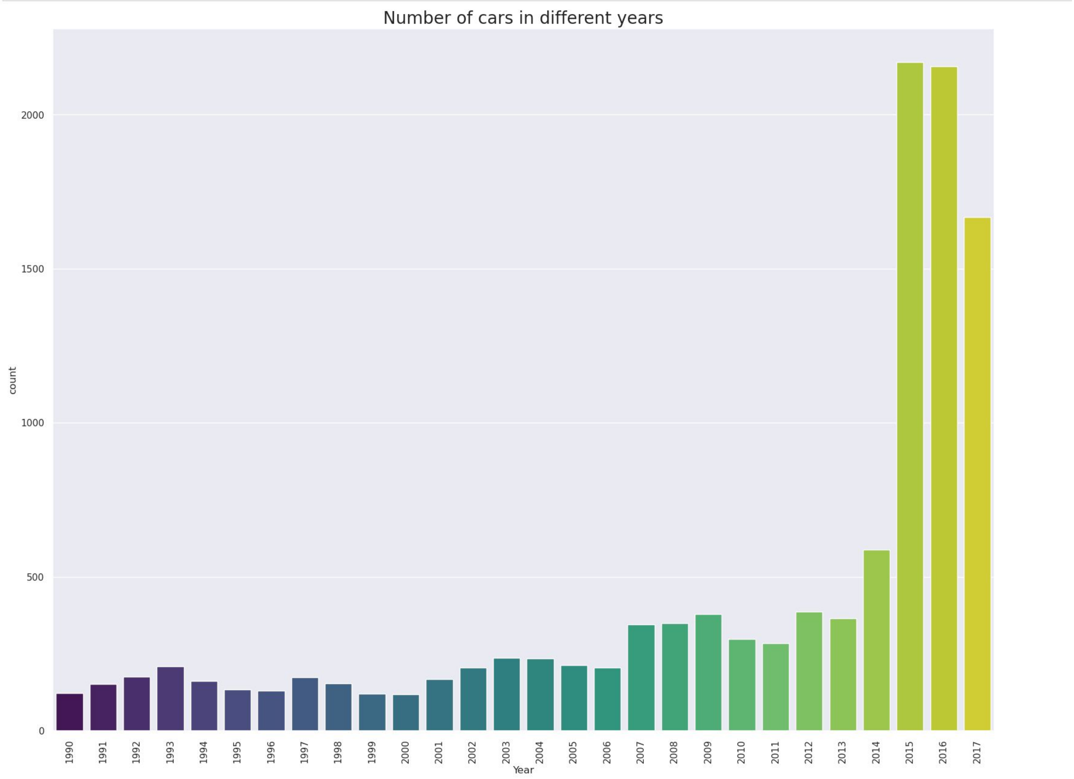
Car Make Distribution

- Chevrolet' has the highest frequency in the dataset, indicating popularity or a wide range of models.
- Luxury brands like 'Bugatti' and 'McLaren' have fewer models, mirroring their exclusivity in the market



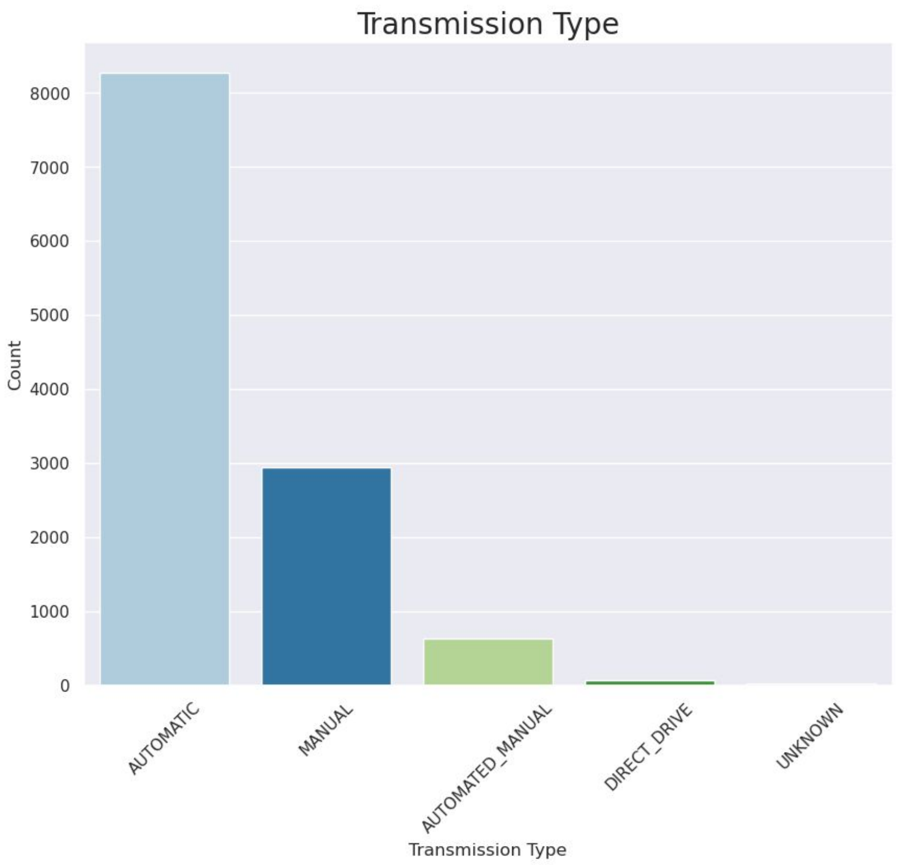
Cars Over the Years

- More recent models (2015-2017) dominate the dataset, aligning with our focus on predicting prices for newer cars



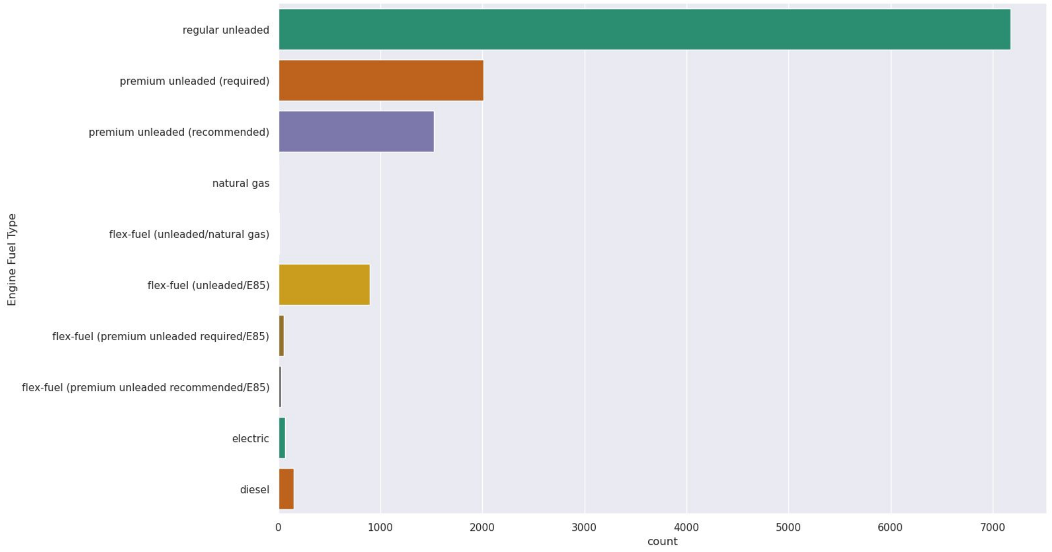
Transmission Types

- Automatic cars are most prevalent, reflecting modern manufacturing trends and consumer preferences.



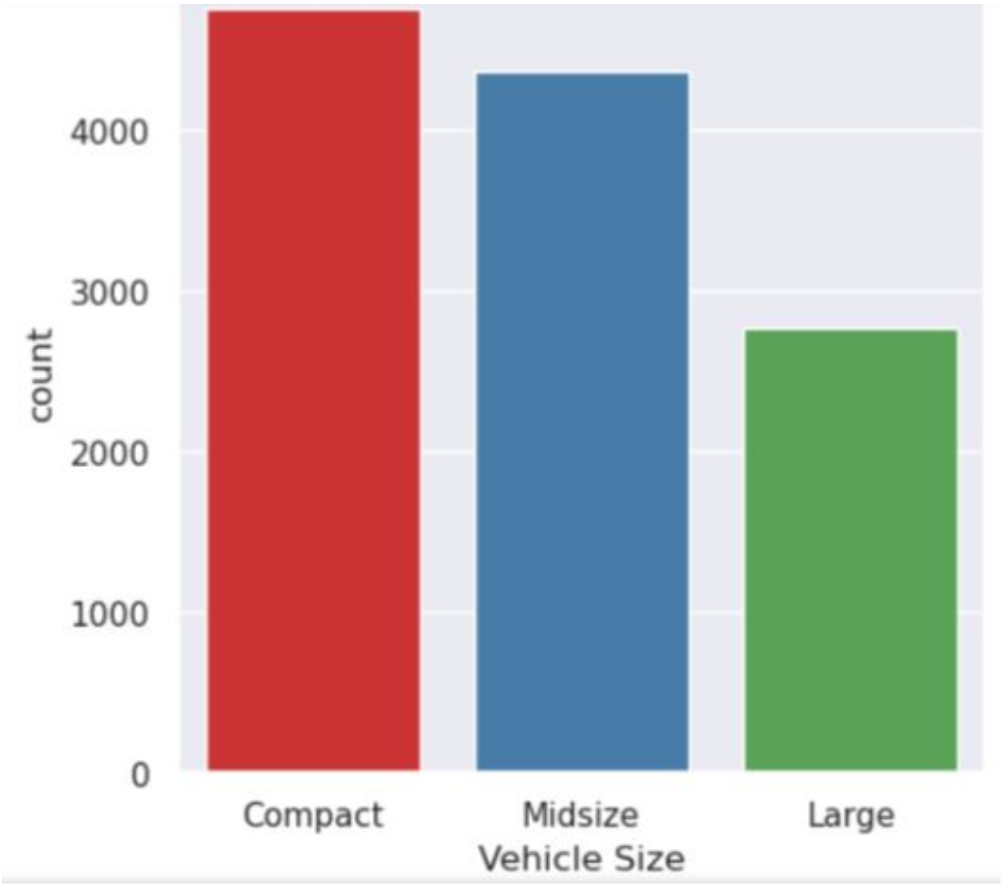
Engine Fuel Types

- The dominance of 'regular unleaded' fuel type suggests a focus on standard consumer vehicles.
- Notably, electric cars are less common but represent a growing segment.



Vehicle Size Analysis

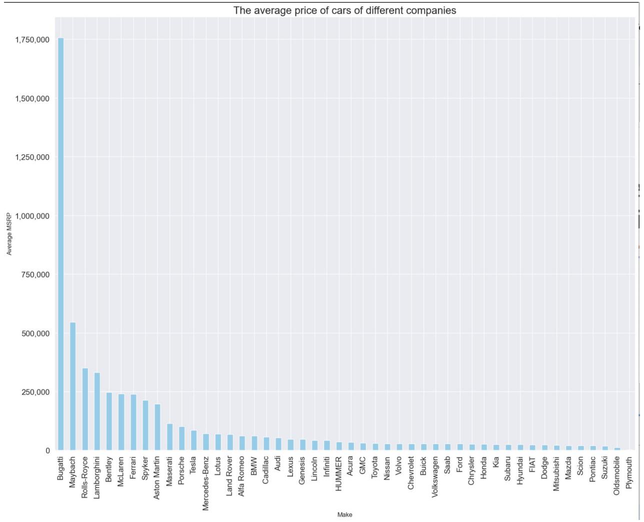
- The dataset shows 'Compact', 'Midsize', and 'Large' vehicles, each affecting price points due to consumer demand and manufacturing costs.



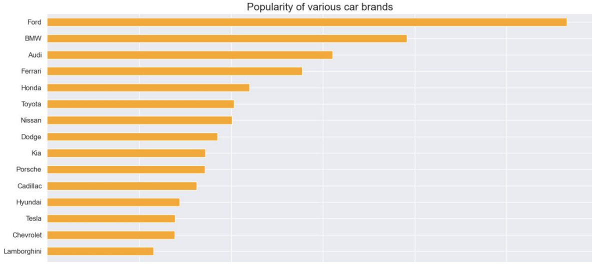
Brand Price and Popularity

- Bugatti is expensive, Affordable brands are Toyota, Nissan.
- BMW, Ford are most popular, Nissan, Kia are less popular

Price:

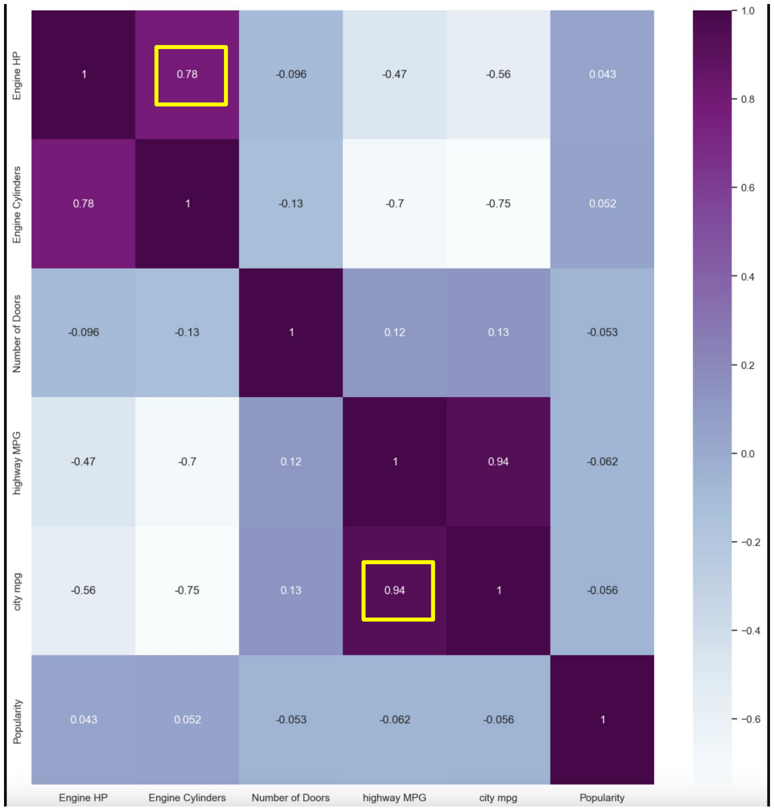


Popularity:



EDA: Heat Map Correlation

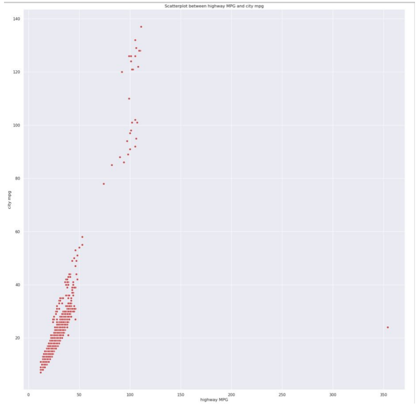
- The heatmap shows strong relation of Engine HP with Cylinders
- High Relation of city and highway mpg



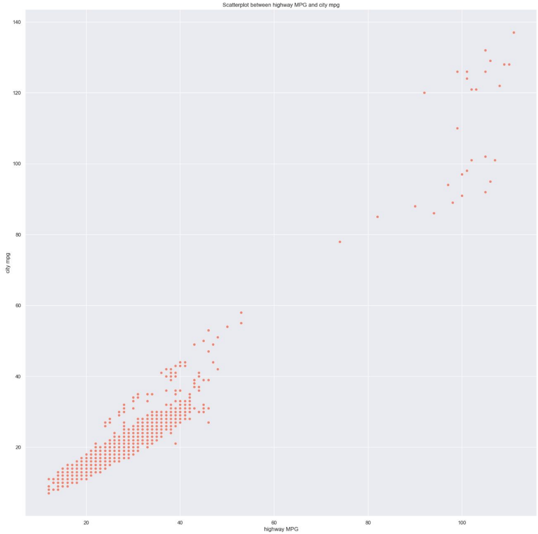
EDA: Scatter Plot of city and highway mpg

- Satterplots of 'highway MPG' and 'city mpg' to show fuel efficiency’s impact on car valuation.
- Must be related, remove outlier value above 350

Before:

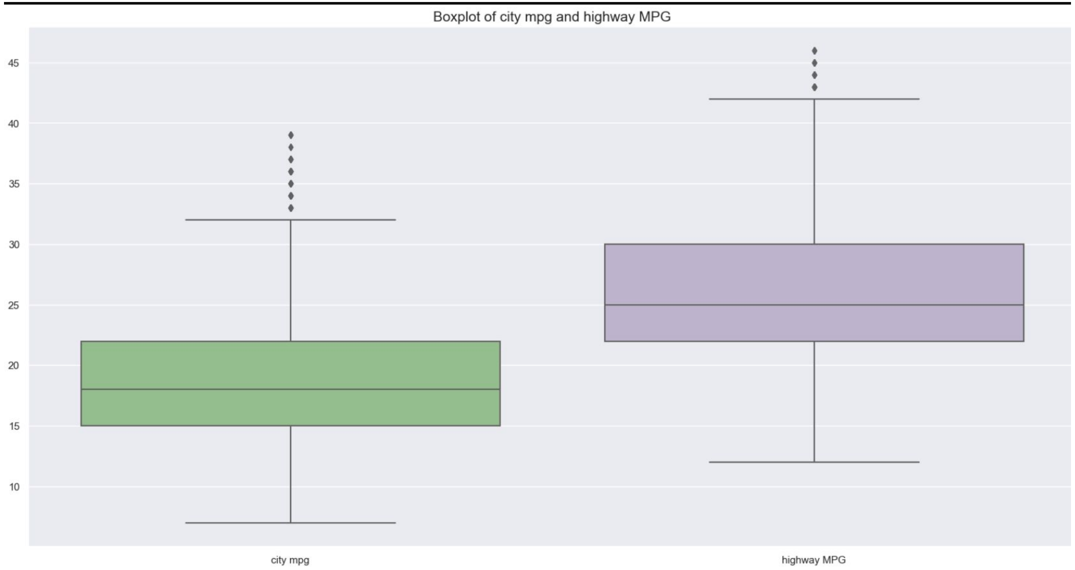


After:



EDA: Boxplot and Remove Outlier city and highway mpg

- Boxplot of 'highway MPG' and 'city mpg', ‘city mpg’ range between 15 to 22, while ‘highway MPG’ range 22 to 30 for most data.



Methodology: Data Cleaning

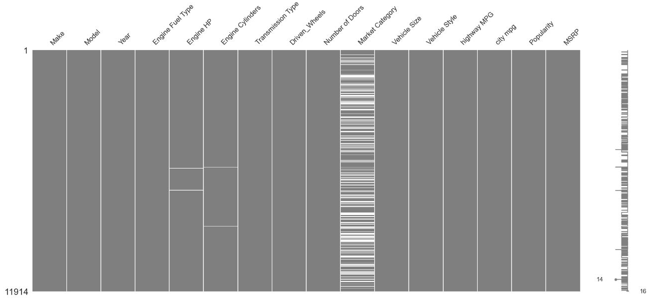
- Use Median, better representer of data than Mean, to fill the NULL values
- Drop ‘Market Category’ due to unique text require NLP techniques

2.2 Missingno

We want to Missingno library from python as it gives us a good graphic representation of which variables has missing values so that later on we can either fill up the missing values with median or we can drop the variable feature completely. We can see that the variable ' Market Category' has the most missing values, and some variable 'Engine HP' Engine Fuel Type' has abit of missing values so we need to address them before we can test the machine learning model. We will use encoding and one hot encoding to do the machine learning later.

```
msno.matrix(data, color = (0.5, 0.5, 0.5))
```

```
<Axes: >
```



Before:

```
data.isnull().sum()
```

```
Make                    0
Model                   0
Year                    0
Engine Fuel Type        3
Engine HP              21
Engine Cylinders       20
Transmission Type       0
Driven_Wheels           0
Number of Doors         1
Market Category      3737
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                    0
dtype: int64
```

After:

```
data.isnull().sum()
```

```
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
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                    0
Years Of Manufacture    0
dtype: int64
```

Methodology: Encoding of Make, Model, and Year

- Encoding helps to change categorical to numerical for machine learning

3.3.2 Encoding the Model Column

```
# Initialize the encoder
encoder = TargetEncoder(cols='Model')

# Fit the encoder on the training set
encoder.fit(X_train['Model'], y_train)

# Transform both the training and the test sets
X_train['Model'] = encoder.transform(X_train['Model'])
X_test['Model'] = encoder.transform(X_test['Model'])
```

Before:

	Make	Model	Year
1354	Oldsmobile	Alero	2003
896	Saab	900	1997
2635	Chevrolet	C/K 1500 Series	1997
1165	Aston Martin	V8 Vantage	2015
2554	Honda	Civic	2015

```
X_train.head()
```

After:

	Make	Model	Year
1354	10812.757938	30176.543012	36784.190660
896	28423.023983	25245.937696	2558.613101
2635	28230.392090	5061.892819	2558.613101
11165	196884.138144	97860.899828	46953.929157
2554	26660.798742	22687.787683	46953.929157

Methodology: One Hot Encoding

- One Hot Encoding helps to convert variables to set of 0 or 1 if present.

3.4.1 One Hot Encoding the rest Engine Fuel Type, Transmission Type, Driven_Wheels, Vehicle Size, Vehicle Style Column

```
encoder = OneHotEncoder()
encoder.fit(X_train[['Engine Fuel Type', 'Transmission Type', 'Driven_Wheels', 'Vehicle Size', 'Vehicle Style']])
one_hot_encoded_output_train = encoder.transform(X_train[['Engine Fuel Type', 'Transmission Type', 'Driven_Wheels',
one_hot_encoded_output_test = encoder.transform(X_test[['Engine Fuel Type', 'Transmission Type', 'Driven_Wheels', 'V
```

We will concatenate the features with the X_train and X_test and remove the actual categorical features as they should not be feeded into to the machine learning model as it may cause error as it is categorical.

```
X_train = pd.concat([X_train, one_hot_encoded_output_train], axis = 1)
X_test = pd.concat([X_test, one_hot_encoded_output_test], axis = 1)
```

```
X_train.drop(['Engine Fuel Type', 'Transmission Type', 'Driven_Wheels', 'Vehicle Size', 'Vehicle Style'], axis = 1,
X_test.drop(['Engine Fuel Type', 'Transmission Type', 'Driven_Wheels', 'Vehicle Size', 'Vehicle Style'], axis = 1, i
```

Before:

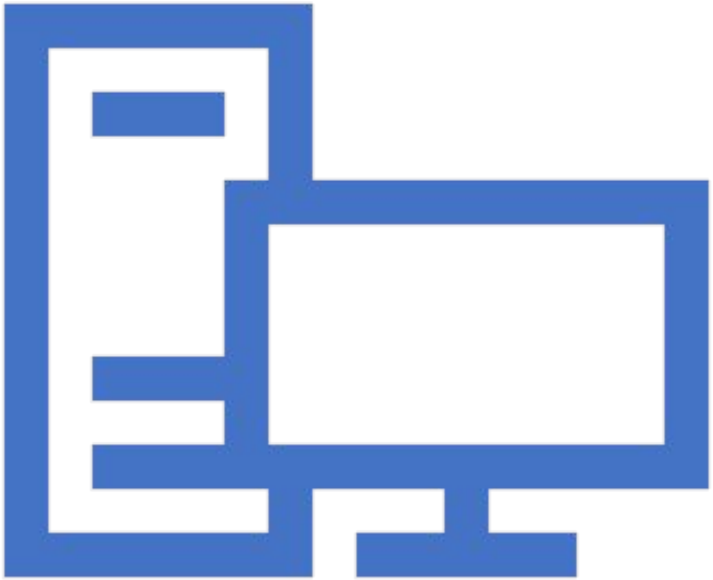
```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Make                 11705 non-null  object
 1   Model                11705 non-null  object
 2   Year                 11705 non-null  int64
 3   Engine Fuel Type     11705 non-null  object
 4   Engine HP            11705 non-null  float64
 5   Engine Cylinders     11705 non-null  float64
 6   Transmission Type    11705 non-null  object
 7   Driven_Wheels        11705 non-null  object
 8   Number of Doors      11705 non-null  float64
 9   Vehicle Size         11705 non-null  object
 10  Vehicle Style        11705 non-null  object
 11  highway MPG          11705 non-null  int64
 12  city mpg             11705 non-null  int64
 13  Popularity           11705 non-null  int64
 14  MSRP                 11705 non-null  int64
 15  Years Of Manufacture 11705 non-null  int64
dtypes: float64(3), int64(6), object(7)
memory usage: 1.5+ MB
```

After:

```
<class 'pandas.core.frame.DataFrame'>
Index: 9364 entries, 1354 to 2564
Data columns (total 47 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Make                  9364 non-null   float64
 1   Model                 9364 non-null   float64
 2   Year                  9364 non-null   float64
 3   Engine HP             9364 non-null   float64
 4   Engine Cylinders      9364 non-null   float64
 5   Number of Doors       9364 non-null   float64
 6   highway MPG           9364 non-null   int64
 7   city mpg              9364 non-null   int64
 8   Popularity            9364 non-null   int64
 9   Years Of Manufacture  9364 non-null   int64
 10  Engine Fuel Type_1    9364 non-null   int64
 11  Engine Fuel Type_2    9364 non-null   int64
 12  Engine Fuel Type_3    9364 non-null   int64
 13  Engine Fuel Type_4    9364 non-null   int64
 14  Engine Fuel Type_5    9364 non-null   int64
 15  Engine Fuel Type_6    9364 non-null   int64
 16  Engine Fuel Type_7    9364 non-null   int64
 17  Engine Fuel Type_8    9364 non-null   int64
```

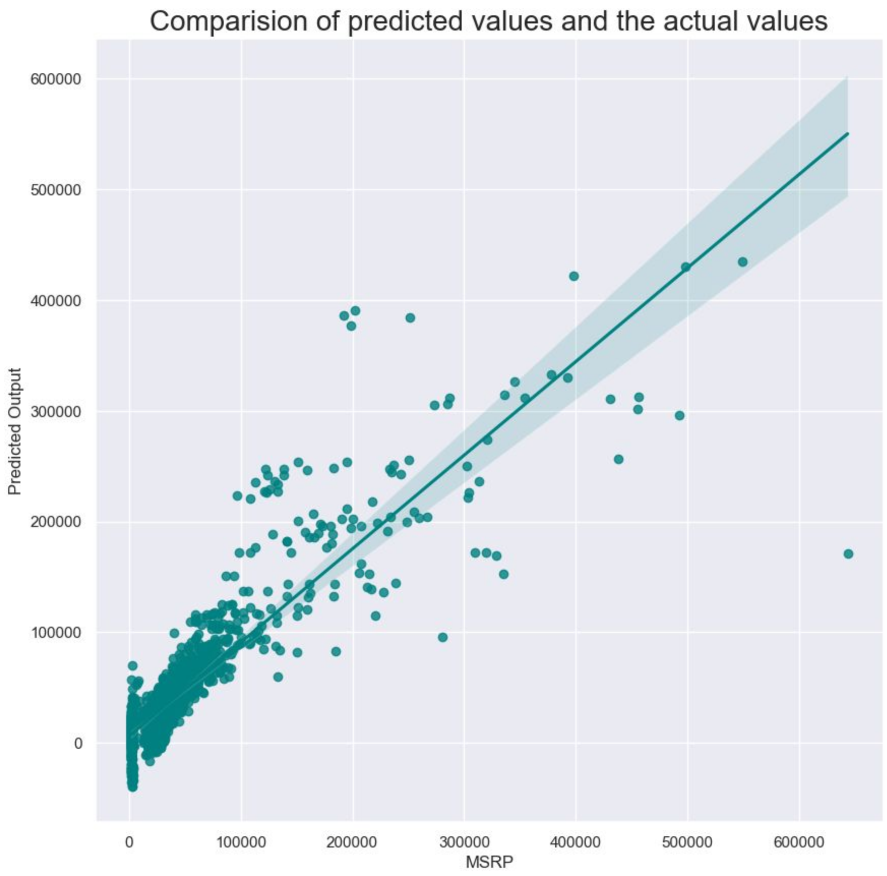
Machine learning

- In this section, we'll evaluate different machine learning models to determine which best predicts car prices based on our dataset
- Models analyzed: Linear Regression, K-Neighbors Regressor, Decision Tree Regressor, Gradient Boosting Regressor.
- Evaluation Metrics: Mean Absolute Error (MAE) and Mean Squared Error (MSE) for assessing model performance



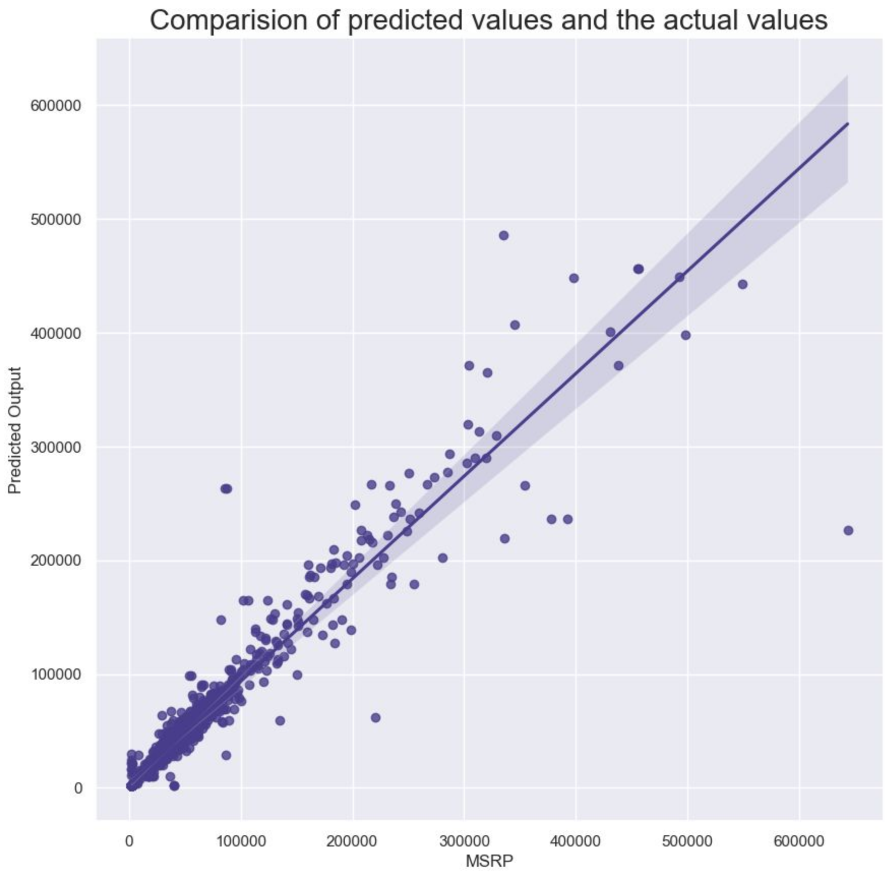
Linear Regression - Simplicity and Effectiveness

- Linear Regression is known for its simplicity and effectiveness in predicting continuous outcomes.
- Here is a regplot showing the predicted values against the actual MSRP, highlighting the linear relationship.



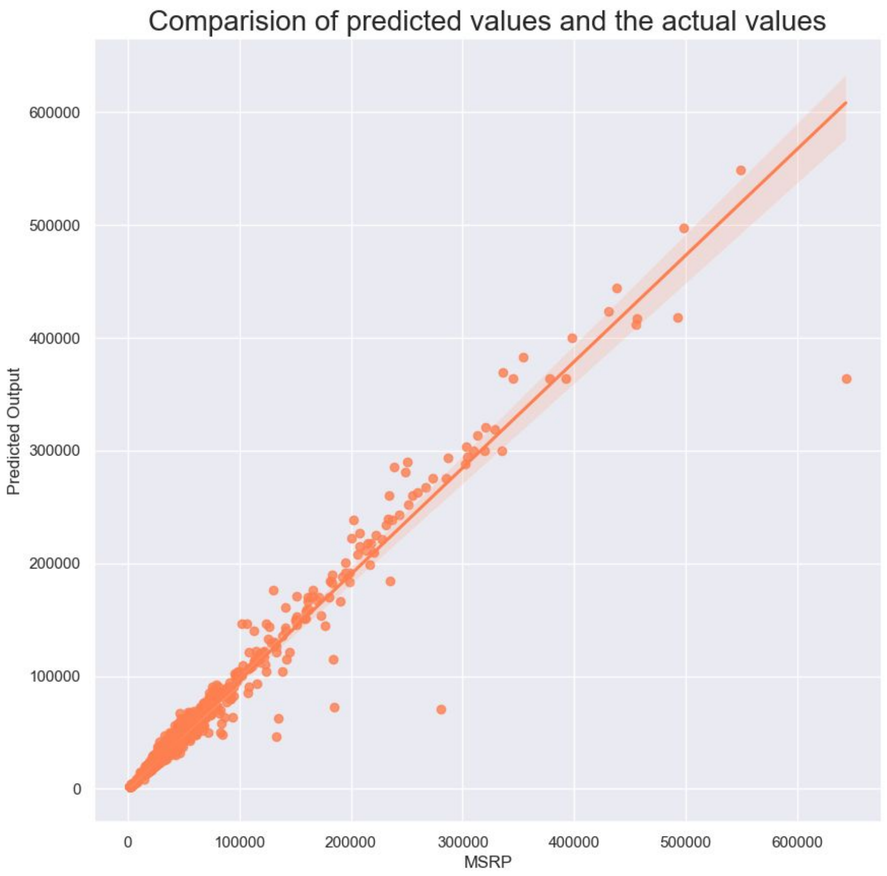
K-Neighbors Regressor - Balancing Simplicity and Precision

- K-Neighbors Regressor leverages the local data structure to make predictions, useful for datasets with nonlinear relationships.
- This regplot shows the spread of predicted values around actual prices, demonstrating model fit.



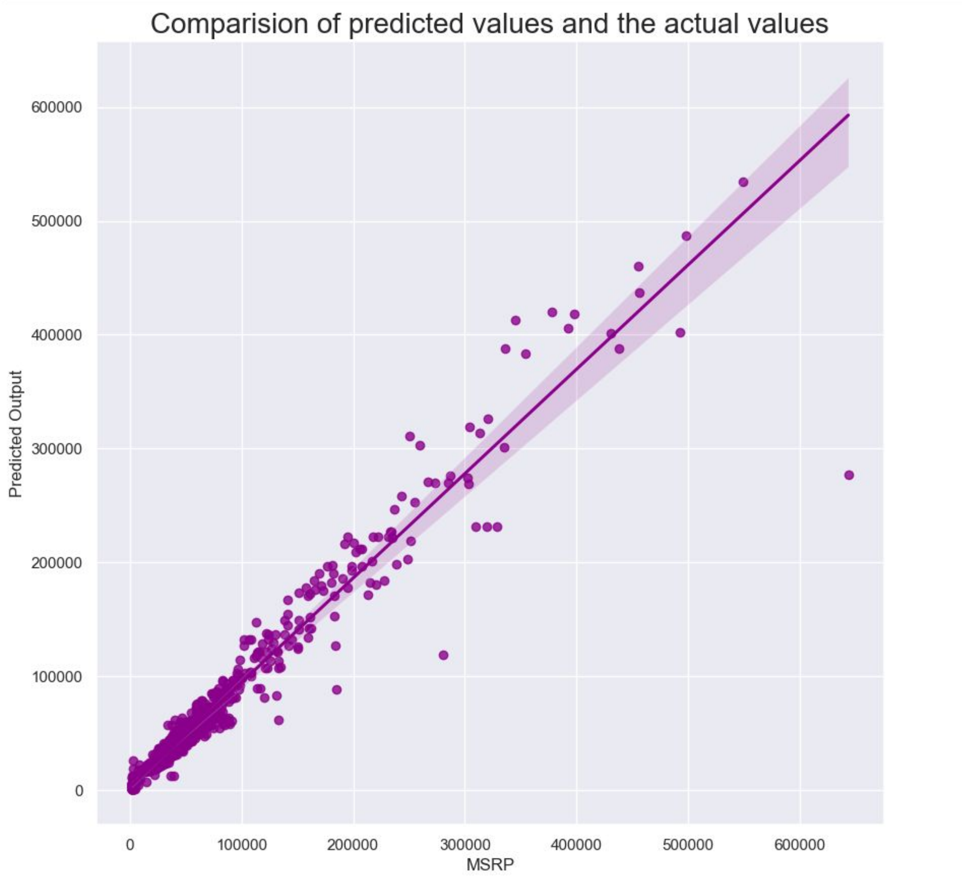
Decision Tree Regressor - Capturing Non-linear Patterns

- Decision Trees are effective for complex datasets by modeling nonlinear relationships between features.
- The regplot illustrates how well the Decision Tree captures the variability in car prices.

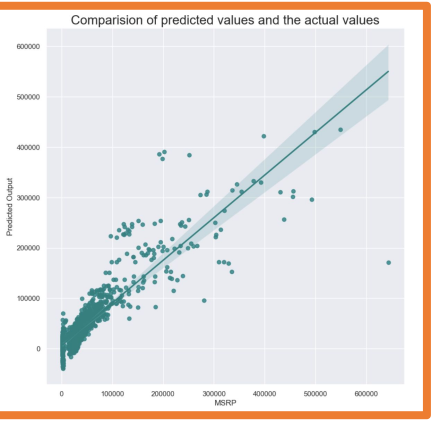


Gradient Boosting Regressor Analysis

- Gradient Boosting builds on Decision Trees by improving model predictions through learning from previous errors.
- This regplot demonstrates the precision of the Gradient Boosting model, especially in dense data clusters.

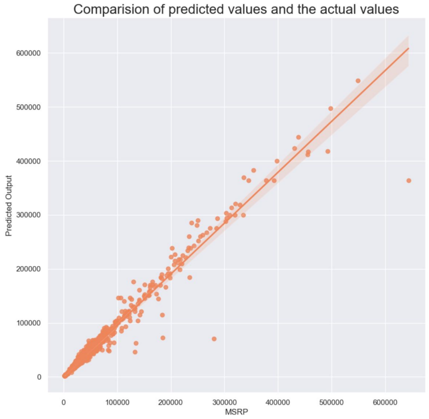


Comparison of regplot based on compactness

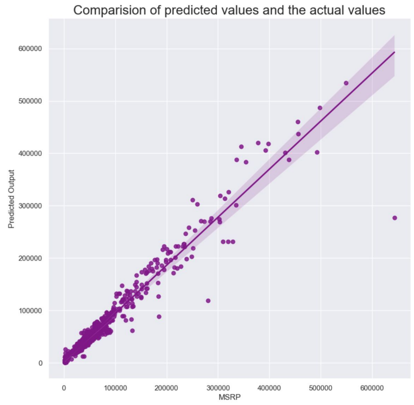


Linear
Regression
4th performance

K-Neighbors
Regressor
2nd performance



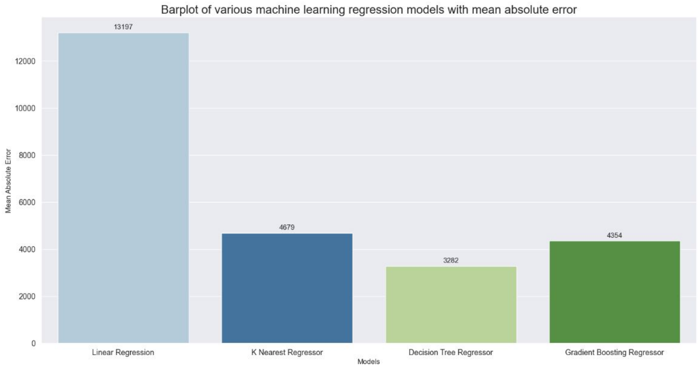
Decision Tree
Regressor
3rd performance



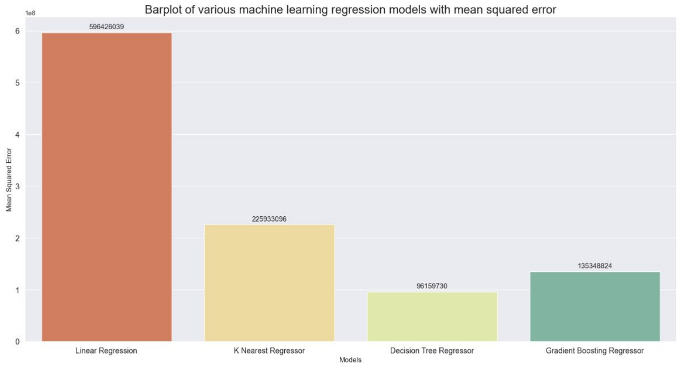
Gradient Boosting
Regressor
1st performance

Barplot of Machine Learning Models

MAE



MSE



	Models	Mean Absolute Error	Mean Squared Error
0	Linear Regression	13197	596426039
1	K Nearest Regressor	4679	225933096
2	Decision Tree Regressor	3282	96159730
3	Gradient Boosting Regressor	4354	135348824

Conclusion

- Highway MPG and City MPG, Engine HP and Engine Cylinder are all related, reflected to car in general..
- Remove missing and outliers for machine learning.
- Methodology such as encoding is required for categorical become numerical data.
- We can see that using different machine learning models would lead to different values of MAE and MSE.
- Decision Tree Regressor is the most accurate predictor and generalise well to new and unseen data for future use.

Recommendation

- **Update the Dataset**: Our current dataset stops at 2017. We should include newer models, especially electric and hybrid cars, to reflect modern market trends.
- **Refine the Model**: The Decision Tree Regressor has proven most effective and should be used as our primary model to ensure accuracy and adaptability to new data.
- **Develop a Pricing Tool**: Create an website tool that uses car features to provide real-time estimates, helping users make informed decisions quickly.
- **Continuous Updates**: Regularly refresh our model with the latest data to keep our predictions accurate and relevant.
- **Ensure Fairness**: Monitor and adjust our model to prevent biases and ensure our price predictions are fair across all car types and brands.

Thank You!