Great Learning
POWER AHEAD

PREDICTING FAIR MARKET PRICE OF PRE-OWNED CARS

Final Report

Group No. **3**

Batch: **PGPDSE-FTJAN22**

Location: **Bangalore**

Team Members:

Akshay K R
Basava Sawmya
Devansh Khanna
Harish S
Mageshwaran A

Under Esteemed Guidance of:

Ankush Bansal

Table of Contents

ACKNOWLEDGEMENT

At the outset, we are indebted to our Mentor **Mr. Ankush Bansal** for his time, valuable inputs and guidance. His experience, support and structured thought process guided us to be on the right track towards completion of this project. We are grateful to the course directors – DSE Program. Their in-depth knowledge coupled with their passion in delivering the subjects in a lucid manner has helped us a lot. We are thankful to them for the various sessions they have provided. We also thank all the course faculty of the DSE program for providing us a strong foundation in various concepts of analytics and machine learning. We would like to thank our respective families and friends for giving us the necessary support to complete this program.

Last but not the least, we would like to extend our warm wishes and thanks to all our fellow teammates for their commitments, encouragements and continued support during the course of this project.

PROJECT DETAILS

Overview:

The **Pre-owned cars** or so-called used cars have capacious markets across the globe. The demand for used cars has increased significantly in the past decade and it is prognosticated that with Covid-19 outbreak this requirement will augment considerably. Hence to enhance the reliability, with the expansion of the used car market, a model that can forecast the current market price of a used automobile on the basis of a variety of criteria.

This analysis can be used to study the trends in the industry, offer better insight into the market, and aid the community in its smooth workflow. The aim of this project is to predict the car price as per the data set (previous consumer data like engine capacity, distance traveled, year of manufacture, etc.) and understand which are the variables that actually effect the price of a used car. Before acquiring a used car, the prices of new cars are tuned by different facets as in manufacturer's cost, dealer's margin, transportation charges, GST levied on the car.

More upon, the worth of a car turns down every year by 20%. **Over the last 12 months, the average price of a new vehicle rose 12%, while used-car prices shot up 41%,** according to the US Consumer Price Index. Majority of the population cannot afford to buy a car from the showroom, rather prefer buying a pre-owned car.

BUSINESS PROBLEM UNDERSTANDING:

Predicting the Market value of a Used Car is very challenging as it is dependent on various factors which affect the market value of the vehicle. With lack of knowledge about the working of a car most of the prices are blindly guessed.
Even though there are websites that offer this service, their prediction method may not be the best as they are not using the best variables, algorithms and are not taking into consideration of various other factors. Besides, different models and systems may contribute on predicting power for a used car's actual market value. It is important to know their actual market value while buying a used car

BUSINESS OBJECTIVE:

CarGurus is a website which is offering an estimate value of a used car for both buyers and sellers. They may have a good prediction model. However, having a second model taking more features into consideration may help CarGurus give a better prediction of price to their users. Helping CarGurus understand the factors which contribute to the price of the used car and help them by giving an optimal price point for these used cars. This is a two-way deal of increasing revenue for CarGurus as well as improving customer satisfaction of the users of the website.

BUSINESS OUTCOME:

1. Help the sellers with an optimal price by taking relevant factors into consideration and determining the current market value. This could better help the sellers as a wide range of features are taken into consideration and they might not feel like they are underselling the car by considering only a limited feature.

2. For a buyer, this could also be helpful as the process is more standardized. When buying a new car, the price point is fixed taking all features into consideration. But, for a used car the process isn't very smooth as we need to consider additional features like current condition of the car, age of the car, mileage, number of previous owners and so on. This could be made easy through the framework being provided to CarGurus and the buyer doesn't feel like he/she is overpaying for the pre-owned car.

CURRENT SOLUTION TO THE PROBLEM:

We needed to check ourselves what is the current solution to the pricing of used cars which is existing in market, for the same we checked various websites to understand as to what inputs these websites need so as to predict the fair price, an example of the same is given below:

Inputs needed to get a fair car price in US popular Online website CARGURUS:

1. Make
2. Model
3. Manufacturing Year/Buying Year
4. Transmission
5. Trim
6. mileage
7. Registration State

Price range provided was $42000 - $92000 which is a very huge range and user can easily be manipulated if he doesn't know the worth of his own vehicle. We can see that even though an initial assessment is being done by the website to give us a price range of where our vehicle price might fall but the remnant depends largely on an expert and not a data driven analysis. This is what we want to take into consideration and try to nullify it, there are many more factors that actually affect the price which were never taken into consideration while giving the user the market value range.

COMPLEXITY INVOLVED:

1. The size of the data is huge (around 30L) and due to limited resources available for computation we had to sample the data which would better represent the entire population.
2. The price points of the used cars have a lot of variation and we have around 8000 used cars which are way above the normal range of the price which we have excluded for our analysis.
3. We have a lot of relevant features like number of previous owners, condition of frame of the car, if the car suffered from any accident, but due to copious number of missing values we had to exclude them for analysis.

LITERATURE SURVEY:

1. With the recent arrival of internet portals, buyers and sellers may obtain an appropriate status of the factors that ascertain the market price of a used automobile. Lasso Regression, Multiple Regression, and Regression Trees are examples of machine learning algorithms. We will try to develop a statistical model that can forecast the value of a pre-owned automobile based on prior customer details and different parameters of the vehicle. [1] This paper aims to compare the efficiency of different models' predictions to find the appropriate one.

2. To accurately anticipate the price of a car, many different approaches have been used in the digital world, ranging from machine learning approaches like multiple linear regression, k-nearest neighbor, and naive bayes to random forest and decision tree.

3. In [2], [3], [4], [5] and [6] all of these solutions took into account distinct sets of attributes when making predictions based on the historical data used to train the model. We attempted to construct a web application where a user may verify the effective market price of their automobiles using a model for prediction based on the factors that have the greatest impact on vehicle prices.

4. On the subject of used automobile price prediction, several previous studies have been conducted. To anticipate the value of pre-owned automobiles in Mauritius, Pudaruth employed naive Bayes, k-nearest neighbors, multiple linear regression, and decision trees. However, because there were fewer cars observed, their results were not good for prediction. In his article, Pudaruth concluded that decision trees and naive Bayes are ineffective for continuous-valued variables.[7]

5. In paper [7] Predicting the price of Used Car Using Machine Learning Techniques. In this paper, they investigate the application of supervised machine learning techniques to predict the price of used cars in Mauritius. The predictions are based on historical data collected from daily newspapers. Different techniques like multiple linear regression analysis, k-nearest neighbors,

6. To anticipate the price of a vehicle, Noor and Jan employed multiple linear regression. They used a variable selection methodology to determine the variables that had the highest influence and then eliminated the remainder. Only a few variables are included in the data, which were utilized to create the linear regression model. With an R-square of 98 percent, the outcome was outstanding. [4]

7. Peerun et al. conducted study to assess the neural network's performance in predicting used automobile prices. However, especially on higher-priced cars, the estimated value is not very close to the real price. In forecasting the price of a used car, they found that support vector machine regression outperformed neural networks and linear regression by a little margin. [7]

8. Overfitting and underfitting come into picture when we create our statistical models. The models might be too biased to the training data and might not perform well on the test data set. This is called overfitting. Likewise, the models might not take into consideration all the variance present in the population and perform poorly on a test data set. This is called underfitting. A perfect balance needs to be achieved between these two, which leads to the concept of Bias-Variance tradeoff.

9. Pierre Geurts [8] has introduced and explained how bias-variance tradeoff is achieved in both regression and classification. The selection of variables/attribute plays a vital role in influencing both the bias and variance of the statistical model.

10. Robert Tibshirani [9] proposed a new method called Lasso, which minimizes the residual sum of squares. This returns a subset of attributes which need to be included in multiple regression to get the minimal error rate. Similarly, decision trees suffer from overfitting if they are not pruned/shrunk. Trevor Hastie and Daryl Pregibon [10] have explained the concept of pruning in their research paper.

REFERENCES:

[1]. Pattabiraman Venkatasubbu, “Used Cars Price Prediction using Supervised Learning Techniques.”

[2]. Comparative Analysis of Used Car Price Evaluation Models, Tongji University, Shanghai 200000, China.

[3]. Nitis Monburinon, “Prediction of Prices for Used Car by Using Regression Models”, 5th International Conference on Business and Industrial Research, (ICBIR), Bangkok, Thailand, 2018

[4]. Jaideep A Muley, “Prediction of Used Cars’ Prices by Using SAS EM”, Oklahoma State University

[5]. Nabarun Pal, “A methodology for predicting used cars prices using Random Forest”, Future of Information and Communications Conference, 2018

[6]. Kuiper, Shonda, "Introduction to Multiple Regression: How Much Is Your Car Worth?" - Journal Of Statistics Education, 2008

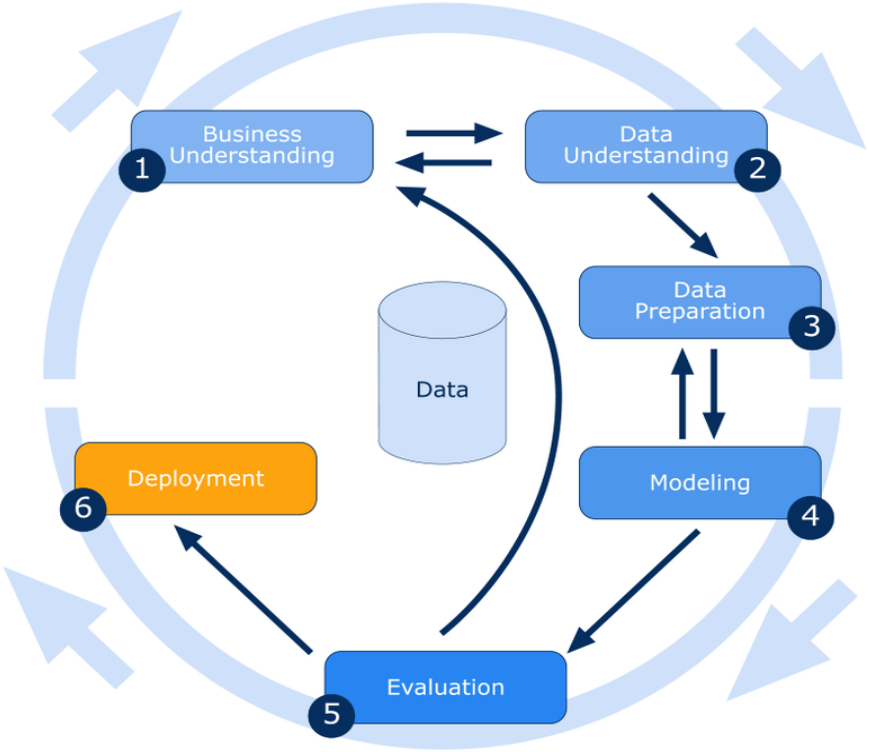
[7]. https://towardsdatascience.com/used-car-priceprediction-using-machine learninge3be02d977b2

[8]. Geurts P. (2009) Bias vs Variance Decomposition for Regression and Classification. In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA

[9]. Robert T. (1996) Regression Shrinkage and Selection Via the Lasso. In: Journal of the Royal Statistical Society: Series B (Methodological) Volume 58, Issue 1

[10]. Hastie, Trevor, and Daryl Pregibon. Shrinking trees. AT & T Bell Laboratories, 1990.

METHODOLOGY TO BE FOLLOWED:



BUSINESS UNDERSTANDING:

The prices of new cars in the industry are fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But due to the increased price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase.

There is a need for a used car price prediction system to effectively determine the worthiness of the car using a variety of features. Even though there are websites that offers this service, their prediction method may not be the best. Besides, different models and systems may contribute on predicting power for a used car's actual market value. It is important to know their actual market value while both buying and selling.

DATASET UNDERSTANDING:

This data was obtained by running a self-made crawler on **CarGurus inventory** in September 2020.

The dataset contains 3 Lakh real world used cars and their features like model, make, economy of car etc.,

- Dimensions of the Dataset: (300000, 66)
- Total Numerical Variables: 19
- Total Categorical Variables: 45
- Total Boolean Variables: 2
- Target/Dependent Variable: Price

DATASET LINK:

https://drive.google.com/file/d/1OT1-Fx2isXN65Pn2RC9Ee_PhgrVF3d5d/view?usp=sharing

Variable Description Link:

https://docs.google.com/spreadsheets/d/1fpT1cFOD8Qt01ZdElfiedLz7OtCIJMh9n_14Qts7dHQ/edit?usp=sharing

Count of missing/ null values:

https://drive.google.com/file/d/1eTf7ZOZ2InLJSFW0h8xsryGevd_kglLb/view?usp=sharing

Redundant Columns:

https://drive.google.com/file/d/1fvtSdG1g7U6Zc2XP3qEsZM_jDVygxgd_/view?usp=sharing

Alternate sources of data that can supplement the core dataset :

a) In our dataset, we have lot of relevant columns like-

1. Frame_damaged
2. has_accidents
3. No_of_previous_owners

which would help us in better predicting the price of the used car, but the challenge is the high percentage of missing values which is making us to not consider these columns.

b). If we had more information regarding the present condition of the car, they could be better predictors of the price of the used car.

c). More details on the location of the used car could also help in better understand any demographic patterns of the buyers.

DATA PREPARATION:

This stage, which is often referred to as data wrangling, has the objective to develop the final data set for EDA and modelling. It Covers all the activities to construct the final dataset from the initial raw data.

Some of the tasks include:

1. **Check the datatypes as per the data dictionary**

```
vin                     object
back_legroom            object
body_type               object
city                    object
city_fuel_economy      float64
daysonmarket             int64
dealer_zip              object
engine_cylinders        object
engine_displacement    float64
engine_type             object
exterior_color          object
fleet                   object
frame_damaged           object
franchise_dealer          bool
franchise_make          object
front_legroom           object
fuel_tank_volume        object
fuel_type               object
has_accidents           object
height                  object
highway_fuel_economy   float64
horsepower             float64
interior_color          object
isCab                   object
is_new                    bool
latitude               float64
length                  object
listed_date             object
listing_color           object
listing_id               int64
longitude              float64
main_picture_url        object
major_options           object
make_name               object
maximum_seating         object
mileage                float64
model_name              object
owner_count            float64
power                   object
price                  float64
salvage                 object
savings_amount           int64
seller_rating          float64
sp_id                  float64
sp_name                 object
theft_title             object
torque                  object
transmission            object
transmission_display    object
trimId                  object
trim_name               object
wheel_system            object
wheel_system_display    object
wheelbase               object
width                   object
year                     int64
dtype: object
```

2. **Change the datatypes of columns wherever necessary.**

   The Variables in our dataset has specific datatypes w.r.t domain, there is no need to change the type of variable

3. **Anomalies (Correct the human error entries) - Drop row wise duplicates**

   As per our Datasets there are no duplicate records present.

4. **Count and percentage (%) of Null values in the dataset**

   https://drive.google.com/file/d/1eTf7ZOZ2InLJSFW0h8xsryGevd_kglLb/view?usp=sharing

   In our dataset there is percentage of missing values, which will be imputed respectively.

5. **Treat the missing values by imputations**
   - Numerical columns:
     - if the missing values present, we have checked the distribution if it is right skew or left skew we have done the median imputation
     - if missing percentage value greater than 40 % we have dropped the column
   - Categorical Columns:
     - if missing percentage values less than 10 % we have done mode Imputation
     - if missing percentage values more than 40% we have dropped the columns

```
[8]  # Dropping Rowwise records where missing values in records is more than 40%

     a = (df.isna().sum(axis=1)/df.shape[1])*100
     c= a[a>40].index.to_list()
     df = df.drop(c,axis = 0)
```

```
[10] df.drop('Unnamed: 0',axis=1,inplace=True)
```

```
[11] df.drop('dealer_zip',axis=1,inplace=True)
```

we alreaady have city dealer zip is redundant in this case

```
[12] df.drop('wheel_system_display',axis=1,inplace=True)
```

Wheel system display is description of wheel system, adds no new information to the dataset

```
[13] df.drop('engine_cylinders',axis=1,inplace=True)
```

engine cylinders is same as engine type so it is a redundant column doesn't add any information to the dataset

```
[14] df_geo = df[['longitude','latitude']]
```

we can drop longitude and latitude in the original dataframe but let's save it for later if we want to do geo-spatial analysis

```
[15] df.drop('listing_id',axis=1,inplace=True)
```

listing_id is a redundant column we drop it

```
[19] def trim(x):
         if(isinstance(x,str)):
             return x[:-3]
```

```
[ ] df['back_legroom'].replace({'--':np.nan},inplace=True)
    df['back_legroom']=df['back_legroom'].apply(trim)
    df['back_legroom']=df['back_legroom'].astype('float')
    df['back_legroom'].replace(np.nan,df['back_legroom'].median(),inplace=True)
```

```
[21] df['front_legroom'].replace({'--':np.nan},inplace=True)
     df['front_legroom']=df['front_legroom'].apply(trim)
     df['front_legroom']=df['front_legroom'].astype('float')
     df['front_legroom'].replace(np.nan,df['back_legroom'].median(),inplace=True)
```

```
[22] c=df['body_type'].mode()[0]
     df['body_type'].replace({np.nan:c},inplace=True)
```

```
[23] df['engine_displacement'].replace(np.nan,df['engine_displacement'].median(),inplace=True)
```

```
c=df['engine_type'].mode()[0]
df['engine_type'].replace({np.nan:c},inplace=True)
```

```
[25] c=df['exterior_color'].mode()[0]
     df['exterior_color'].replace({np.nan:c},inplace=True)
```

```
[26] df['fleet'].replace({np.nan:'Unknown'},inplace=True)
```

```
[27] df['frame_damaged'].replace({np.nan:'Unknown'},inplace=True)
```

```
[28] df['franchise_make'].replace({np.nan:'Unknown'},inplace=True)
```

```
[53] df['year']=(df['year'].max()-df['year'])
```

```
[54] a=df[df['city_fuel_economy'].notnull()==False]
     b=df[df['city_fuel_economy'].notnull()==True]
     e=df.groupby(['city','make_name','engine_type']).median()['city_fuel_economy'].reset_index()
     merged = pd.merge(a,e, on=['city','make_name','engine_type'],how='left')
```

```
[55] merged.drop(['city_fuel_economy_x'],axis=1,inplace=True)
```

```
[56] merged.rename({'city_fuel_economy_y':'city_fuel_economy'},inplace=True,axis=1)
     df = pd.concat([merged, b], ignore_index=True)
     df['city_fuel_economy']=df['city_fuel_economy'].fillna(df['city_fuel_economy'].median())
```

```
[57] a=df[df['highway_fuel_economy'].notnull()==False]
     b=df[df['highway_fuel_economy'].notnull()==True]
     e=df.groupby(['city','make_name','engine_type']).median()['highway_fuel_economy'].reset_index()
     merged = pd.merge(a,e, on=['city','make_name','engine_type'],how='left')
```

```
[58] merged.drop(['highway_fuel_economy_x'],axis=1,inplace=True)
```

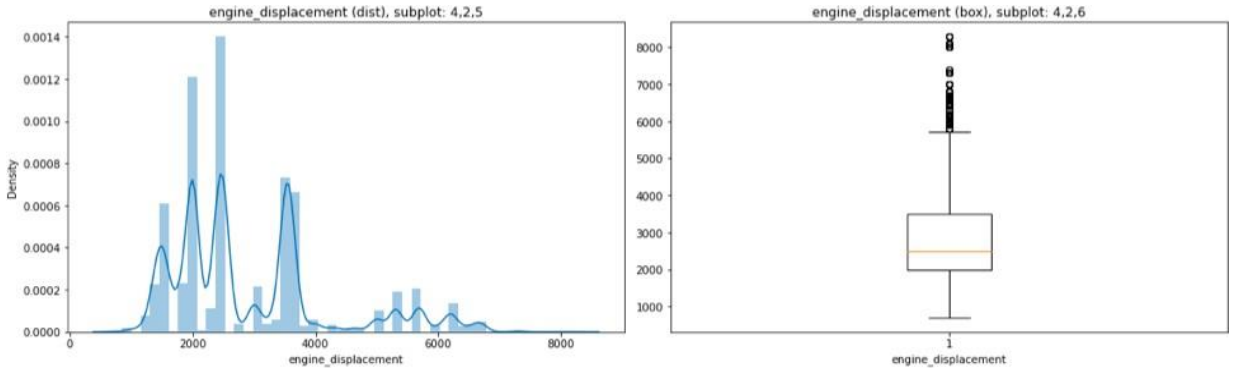
```
[59] merged.rename({'highway_fuel_economy_y':'highway_fuel_economy'},inplace=True,axis=1)
     df = pd.concat([merged, b], ignore_index=True)
     df['highway_fuel_economy']=df['highway_fuel_economy'].fillna(df['highway_fuel_economy'].median())
```

6. Perform the Univariate analysis & describe

UNIVARIATE ANALYSIS:

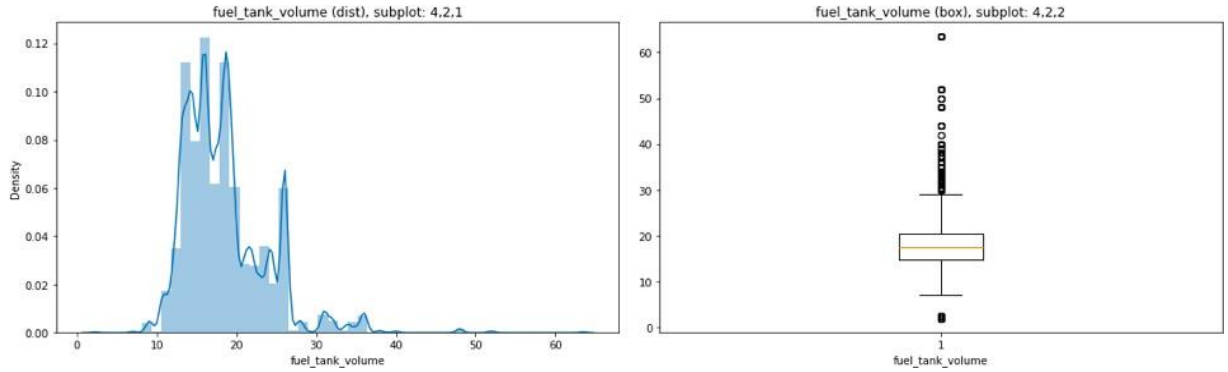
For numerical variables, we plot the distribution curve and box plot to study the variation of the numerical data.

1. Engine Displacement



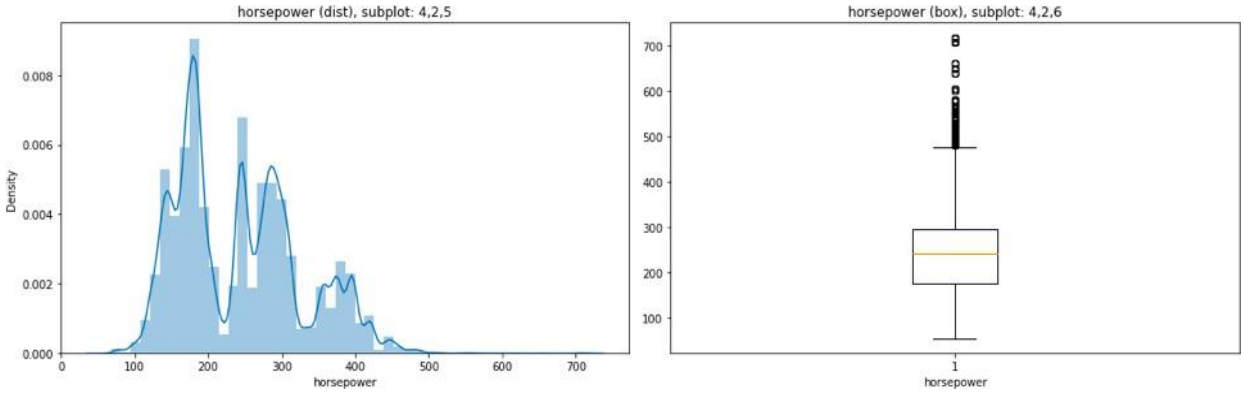
Engine Displacement is multimodal with many outliers, there are many local modes (local maxima) at differentengine displacements which means data is concentrated around several points.

2. Fuel_Tank_Volume



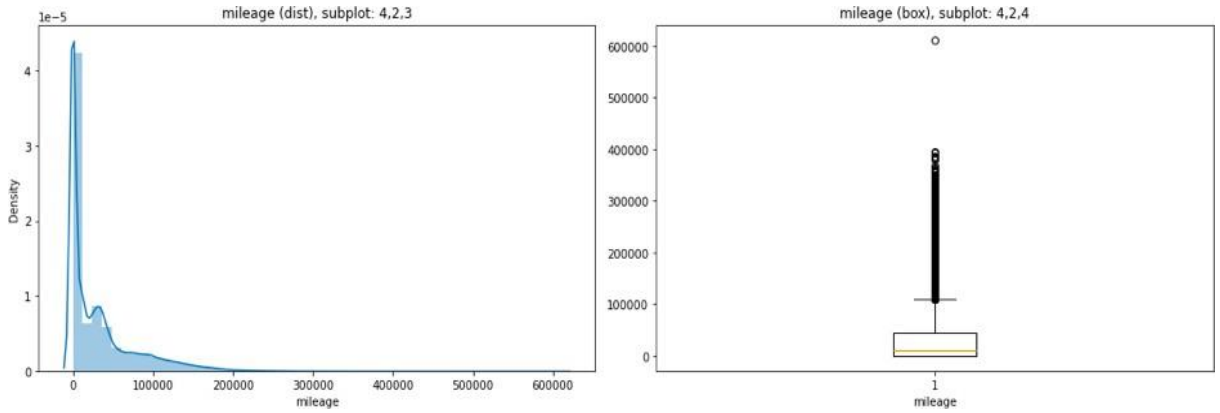
Fuel tank volume is right skewed with outliers, some local modes suggest high concentration around certain Points.

3. Horsepower



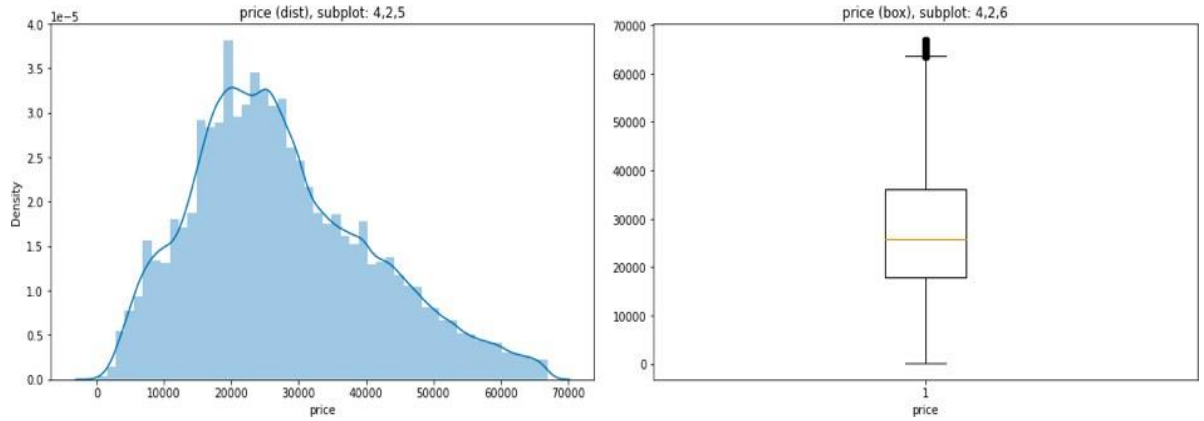
Horsepower is the original horsepower of the car; it is right skewed with no negative values with local maxima's Suggesting concentrated grouped data

4. Mileage



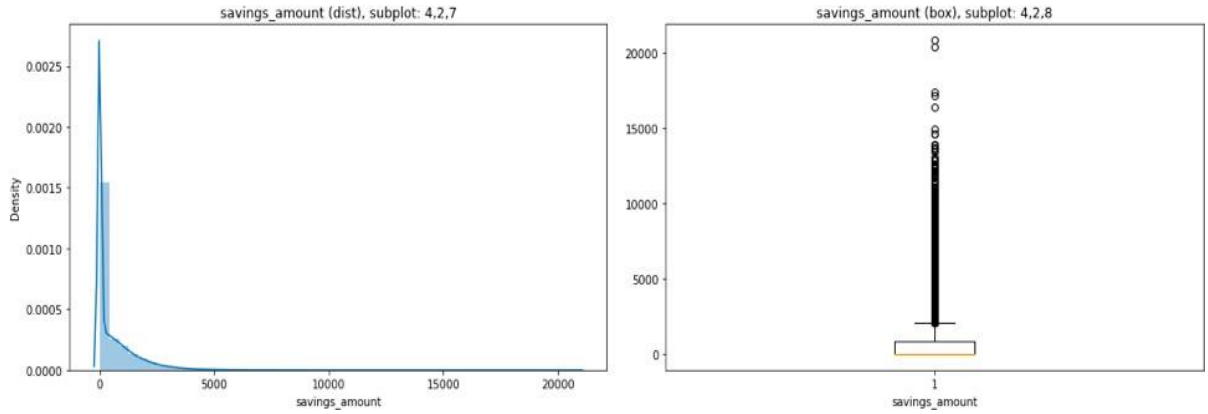
Mileage is highly right skewed with many outliers; loss of local maxima suggests data is concentrated about themode

5. Price



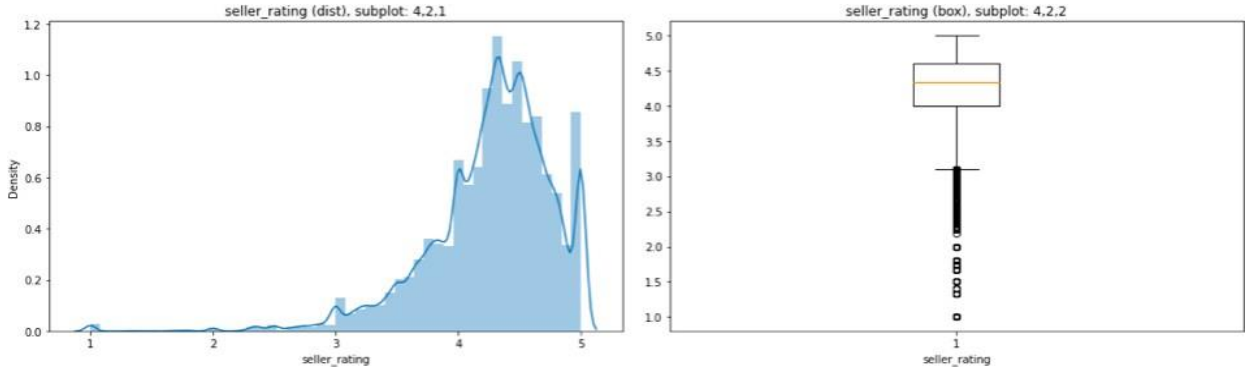
Price is our target variable, we have already treated the outliers of our target, and we can see the only borderlineoutliers are there which our model should be able to predict

6. Savings Amount



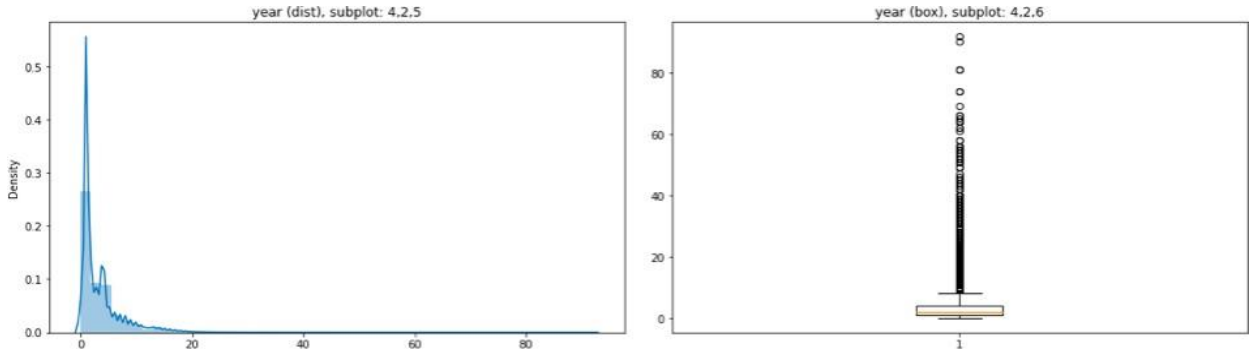
Savings amount suggests the amount saved in the transaction, we can see from the plots that the savings are on thelower side, right skewed with many outliers

7. Seller_Rating



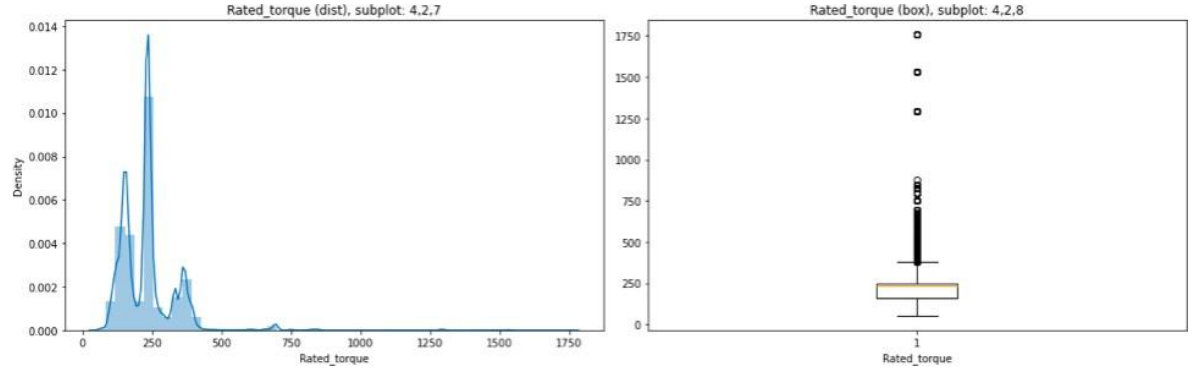
Seller Rating is the rating provided by the customers to the sellers, we can see that many customers were happywith their purchase, with some outliers in the lower range. The feature is left skewed,

8. Age



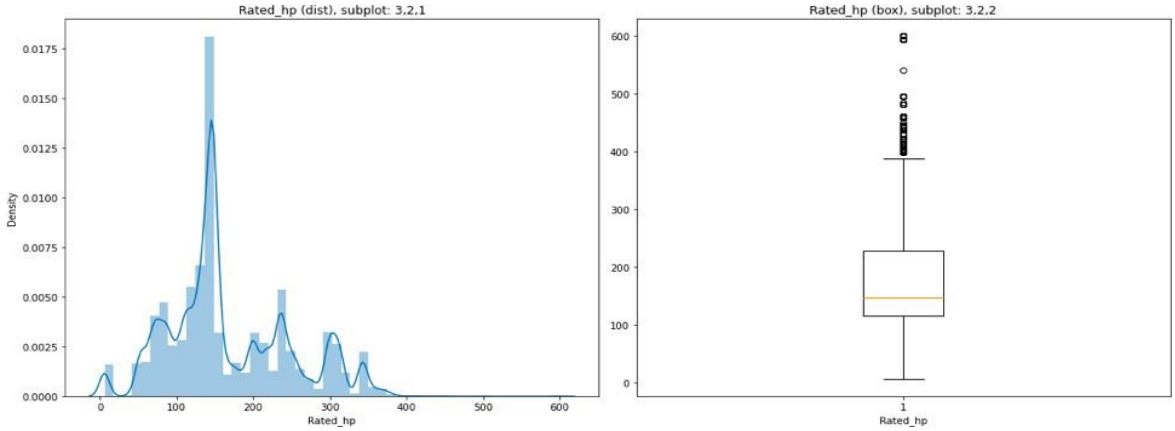
Age of the vehicle is the years the car has spent on the road after being originally purchased, we can see that mostof the cars are grouped between the ages of 0-10 with outliers of upto 90 yrs of age. The data is right skewed

9. Rated Torque



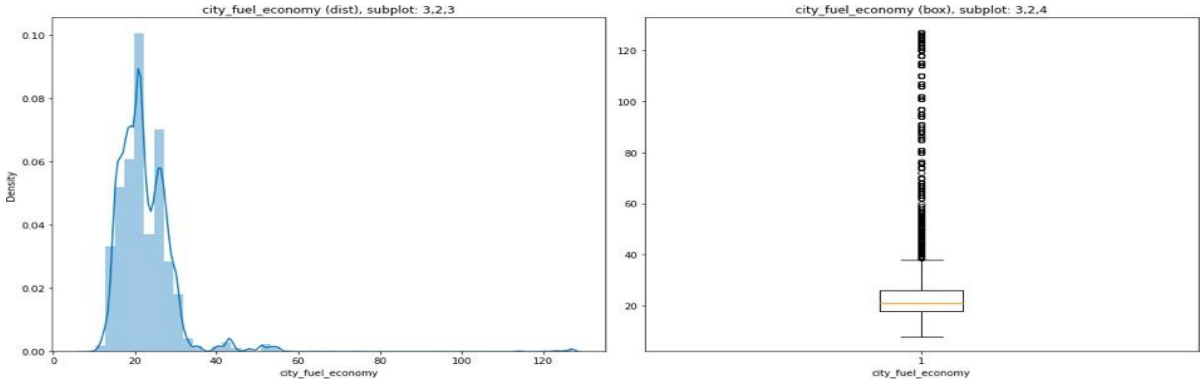
Rated torque is extracted from horsepower and rpm output of the car, we can see that it is on the lower side which makes sense since, with increasing age the torque output should be less. The data in this feature is right skewed withmany outliers.

10. Rated_Hp



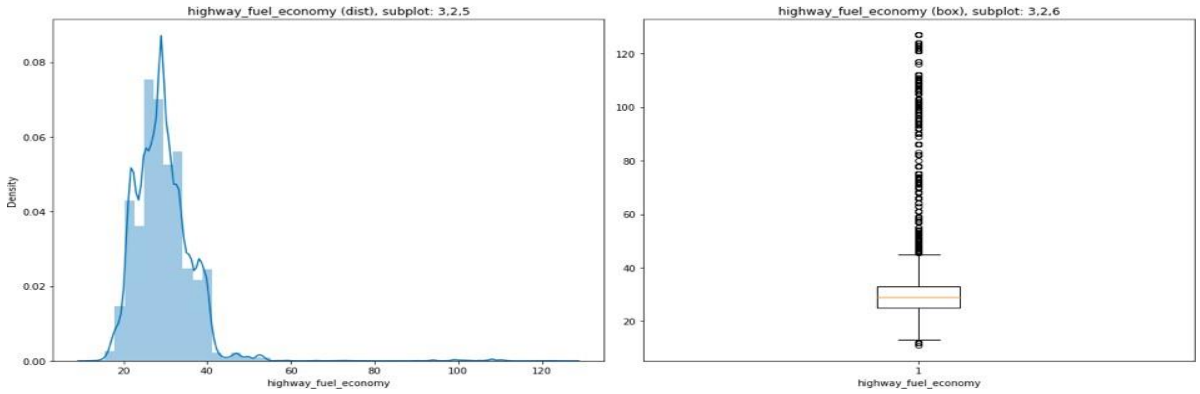
Rated Hp is derived from output power of the vehicle, the data is right skewed with few outliers with high Hp, thisHp will be lesser than our original Hp with increase in age.

11. City_Fuel_Economy



The economy of a vehicle may be low because of heavy traffic, more police barricading and the cities being heavily monitored, we can see that data is highly right skewed with a few cars giving higher economy within the city.

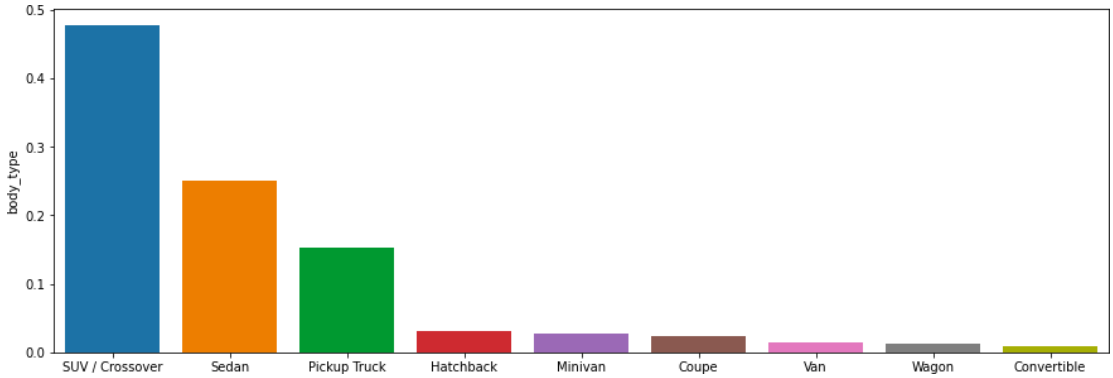
12. Highway_Fuel_Economy



Highway fuel economy can be slightly higher, since the highways have less traffic, wider roads and less surveillance. We can see that the data is right skewed with many cars on the higher side.

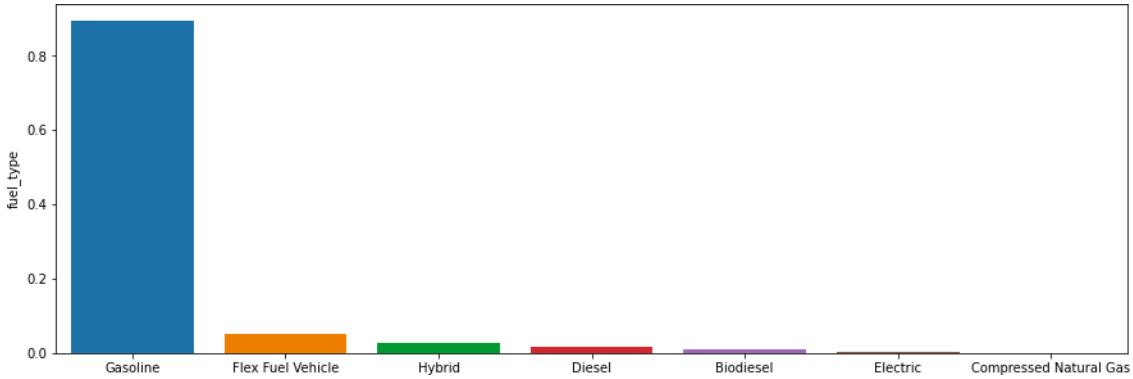
For categorical variables, We plot a combination of bar graph and pie chart to understand the distribution of categorical data in the dataset.

1. **Body Type**



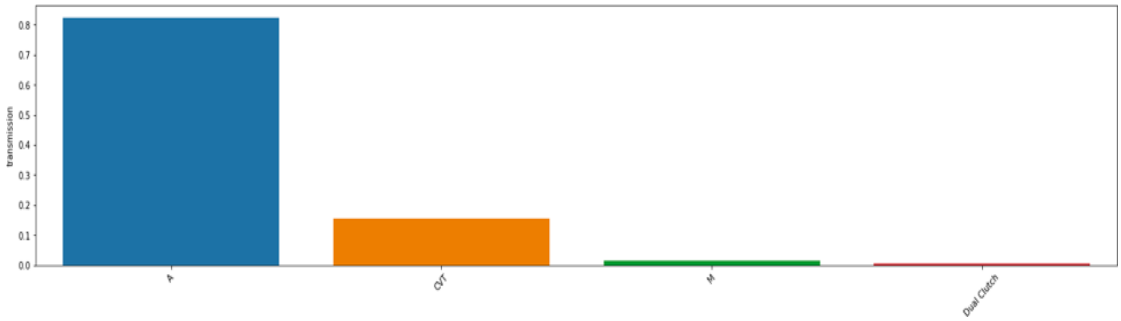
This is a plot of the percentage of different body types, we can clearly see that **SUV/Crossover** are the mostpreferred body-type followed by Sedan and Pickup Truck, others have a very similar count.

2. **Fuel Type**



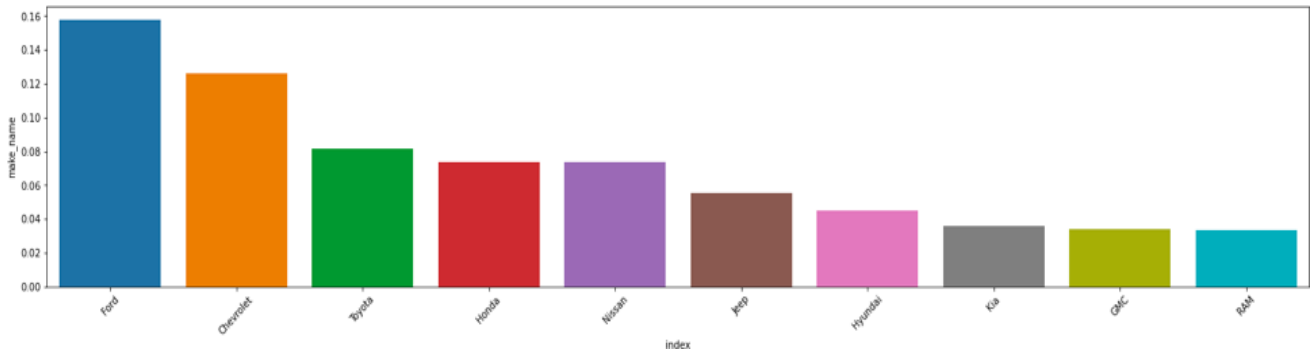
As we can see that **Gasoline** is the most preferred fuel type since it is a conventional fuel type right now,while the other fuel types which are unconventional are not being preferred much

3. **Transmission**



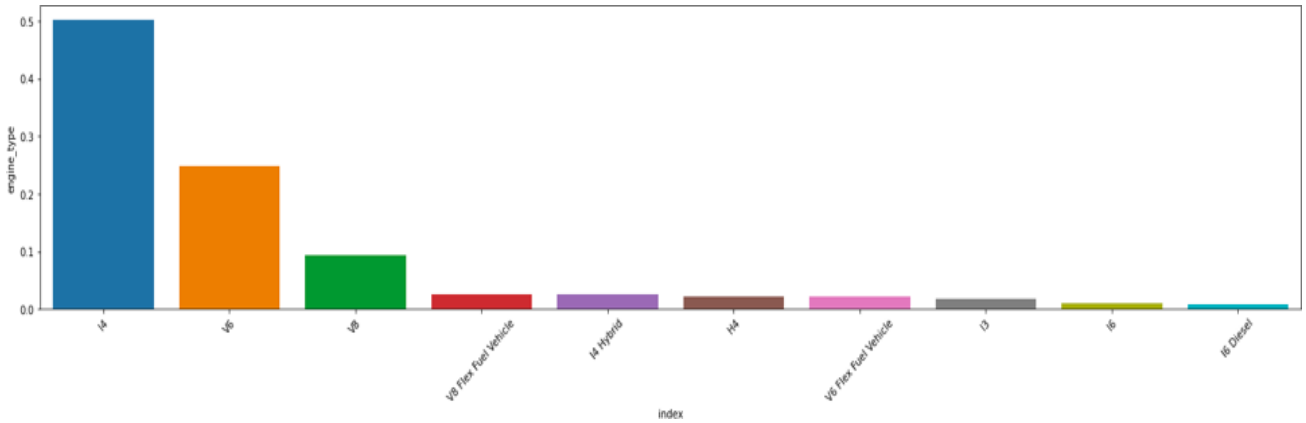
Automatic is the most preferred transmission system, while Continuously Variable Transmission is preferred less, while dual clutch and manual transmission is nearly obsolete in the US

4. Make Name



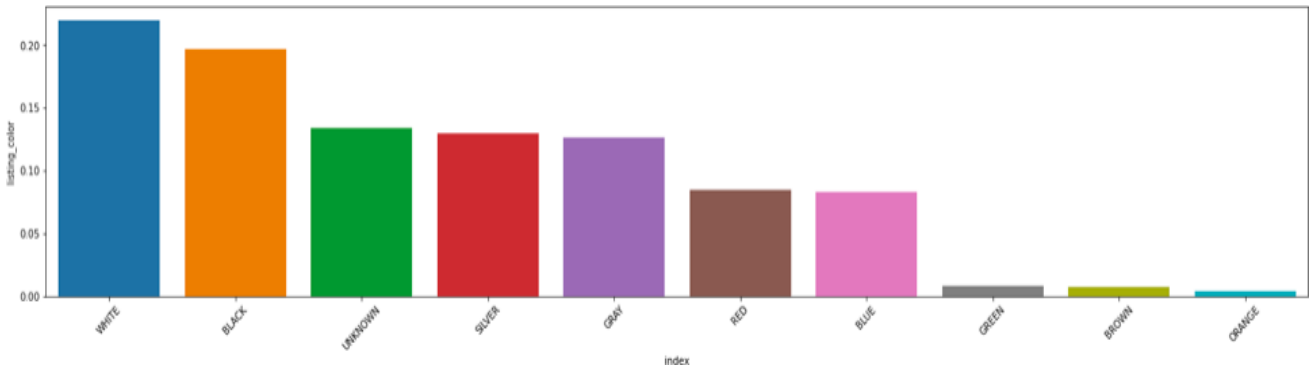
In the above figure we can see the top 10 preferred vehicles in our dataset, the top preferred is Fordfollowed by Chevrolet, Toyota.

5. Engine_Type



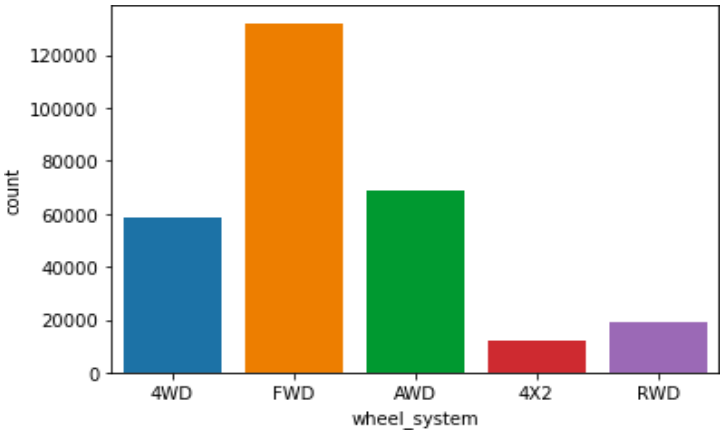
We can see from the figure above that the I4 is the most preferred engine type, because of its economicalnature.

6. Listing Color



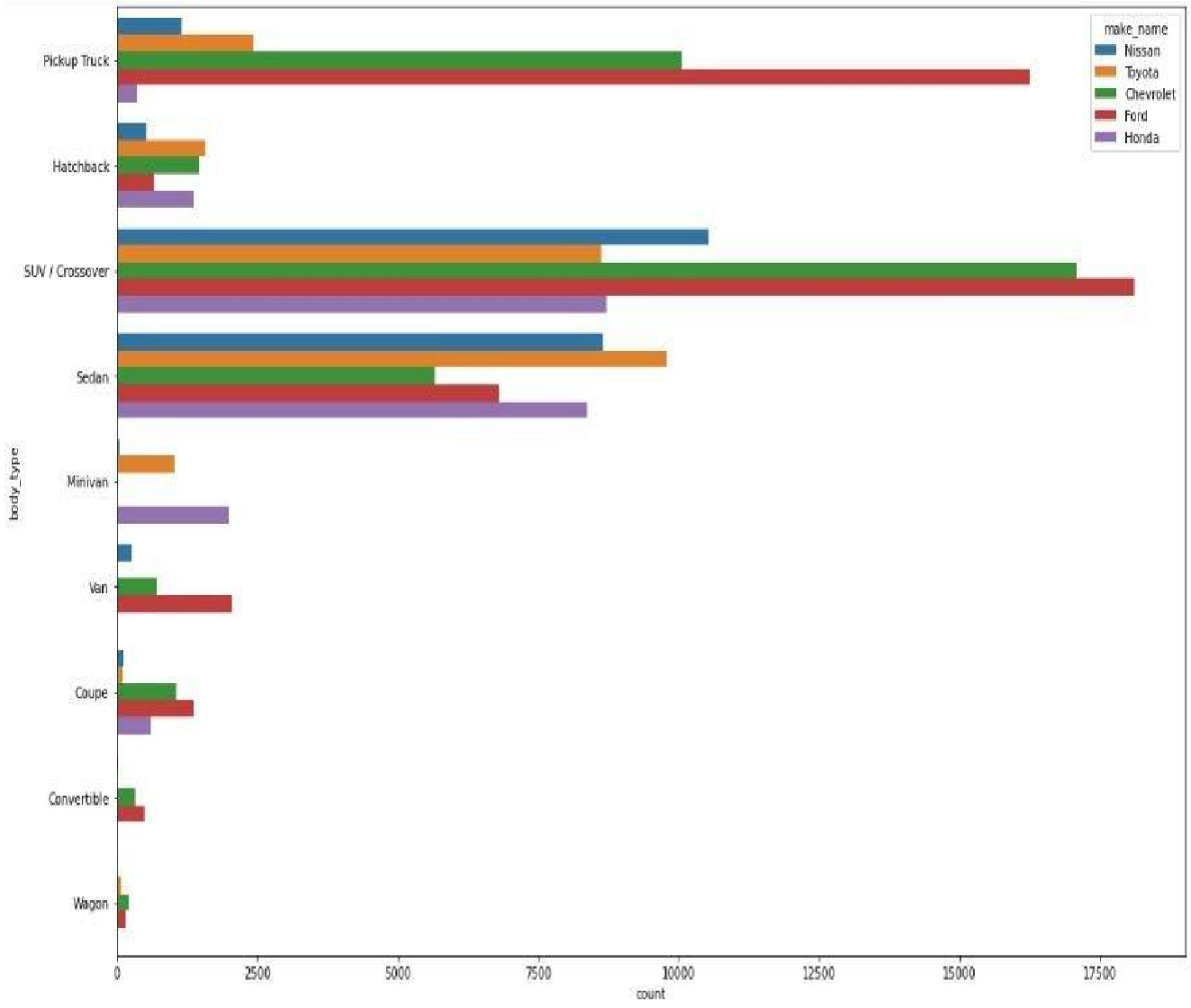
We can see from the above plot that white and black are the most preferred colors in the dataset, but a majorportion of car color is unknown so maybe color doesn't play that important a role in selection of a car.

7. Wheel_System



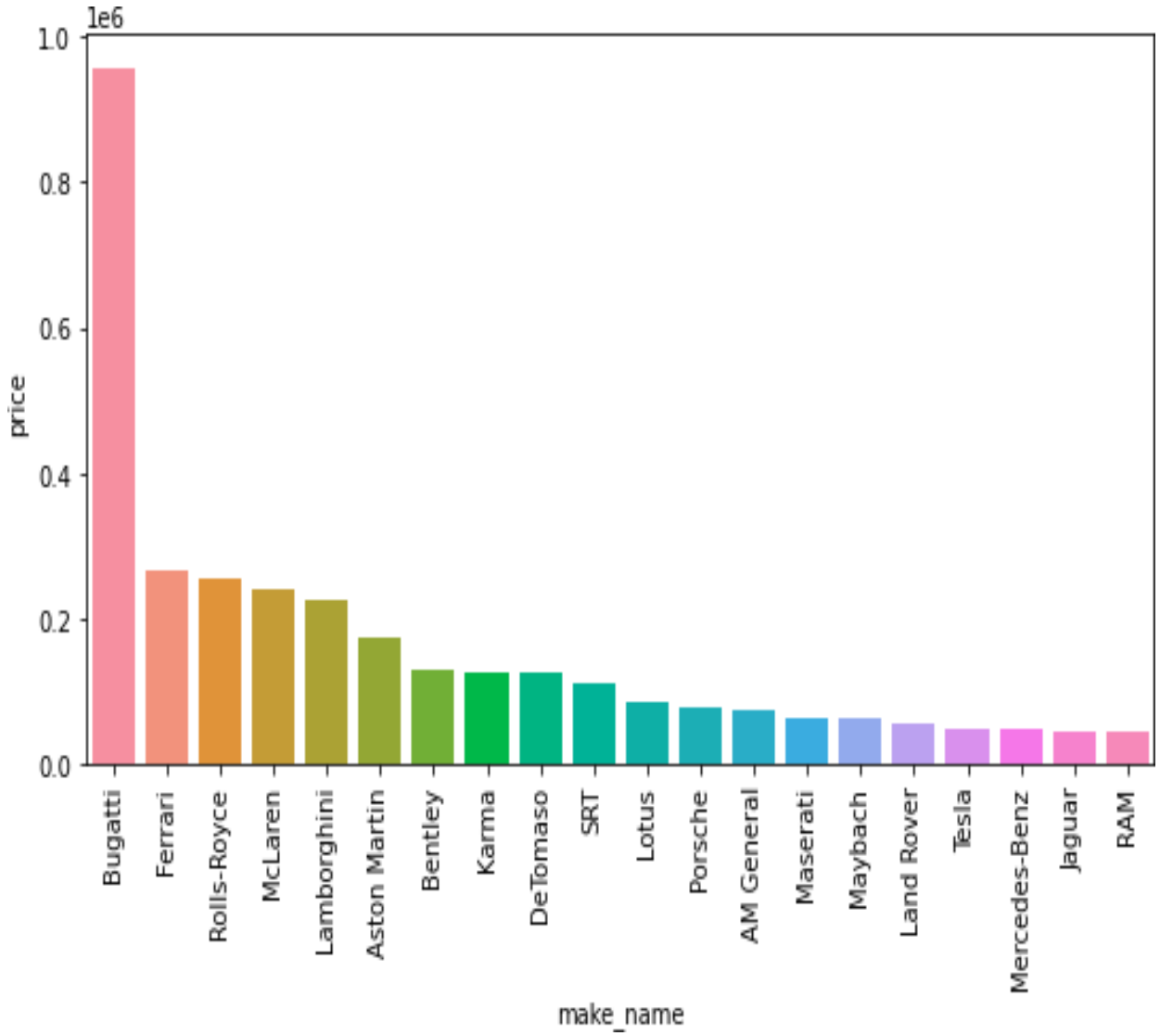
We can see from the above plot that Front-Wheel Drive is the most preferred drivetrain system followed byAll-wheel drive and 4-wheel drive.

8. Body Type



- Ford is most dominating manufacturer across most body types which are listed on the website
- Honda is the most preferred with respect to minivan
- Chevrolet is mostly preferred when it comes to wagon

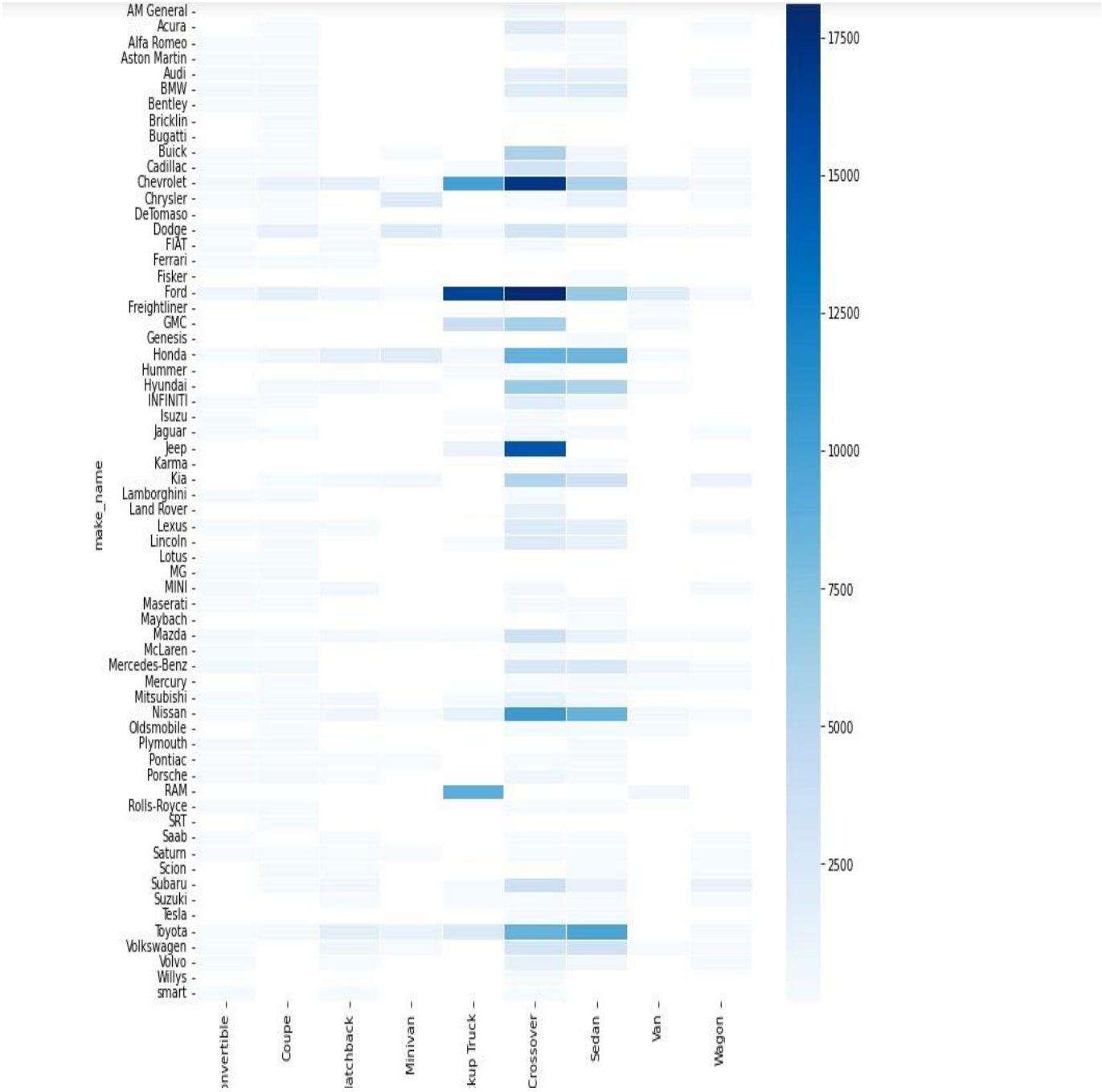
BIVARIATE ANALYSIS:



1. Price Vs Make Name

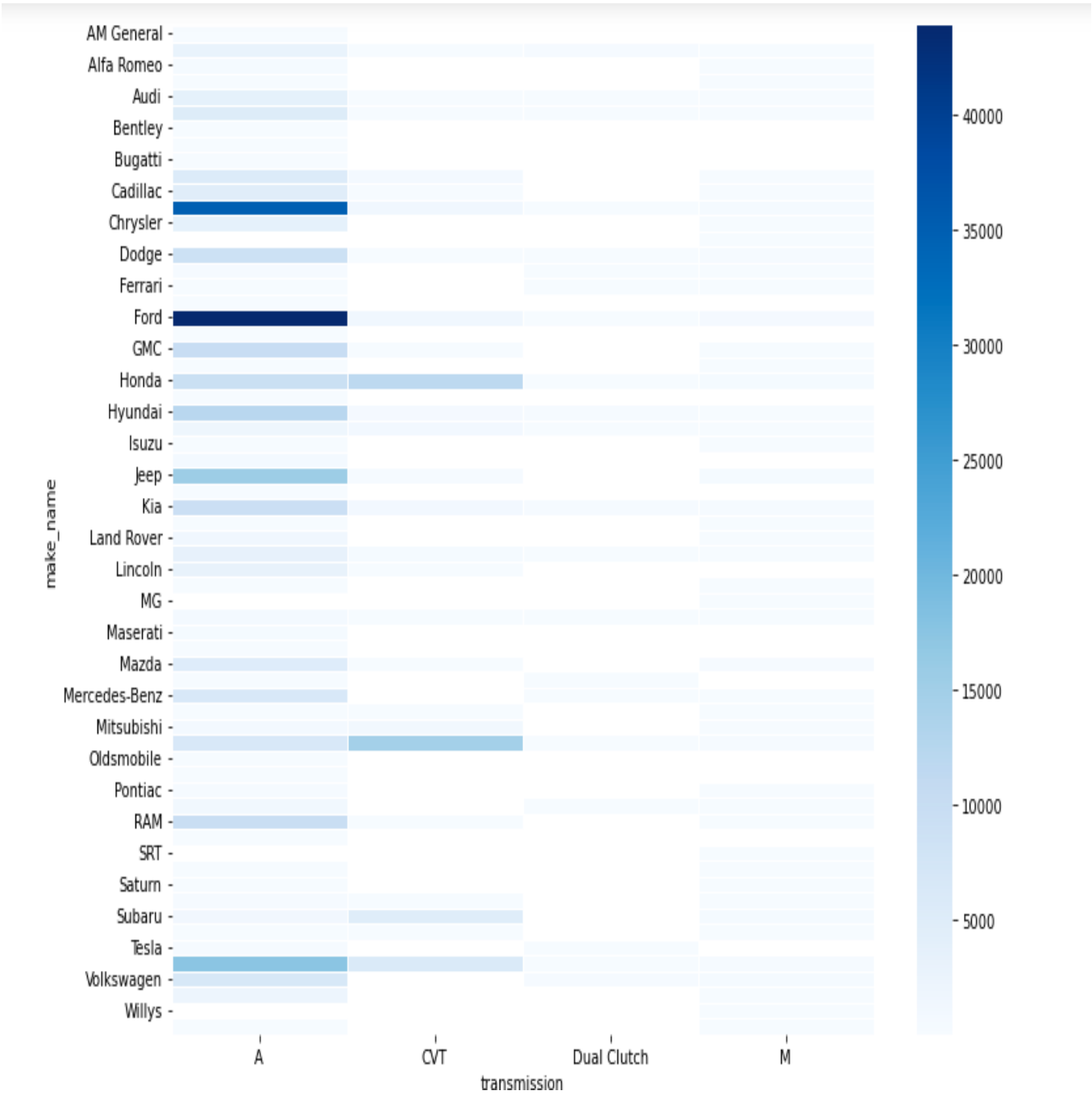
We can see from the above graph that brands like Buggati, Ferrari, Rolls-Royce extract a higher price than the other brands

2. Make Name Vs Body Type



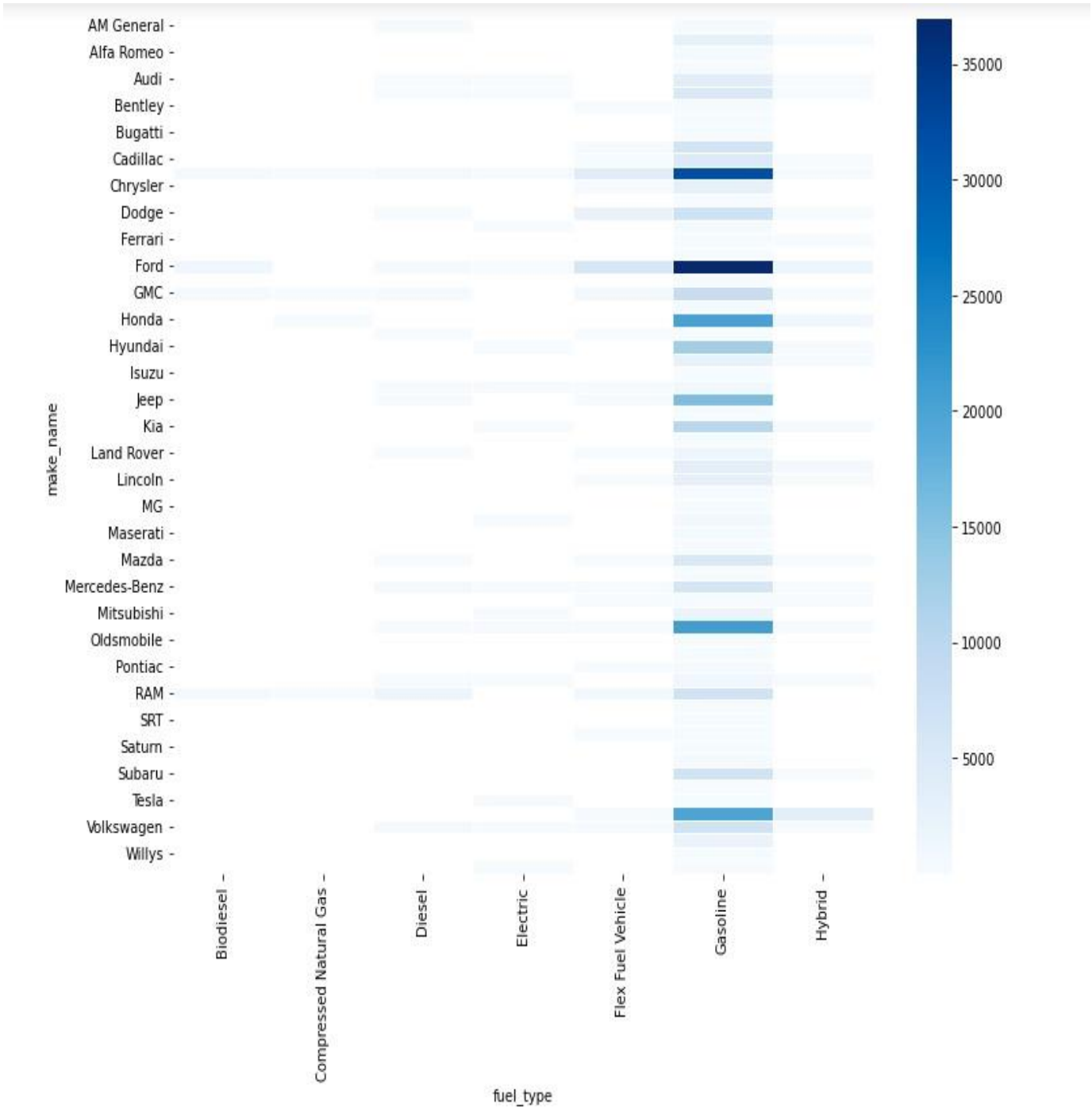
From the above heatmap we are trying to understand which cat body type is more preferred and from which brand, we can high no of cars being preferred from Ford in Pickup Truck, Crossover and Sedan body types, same goes for Chevrolet.

3. Make_Type Vs Transmission



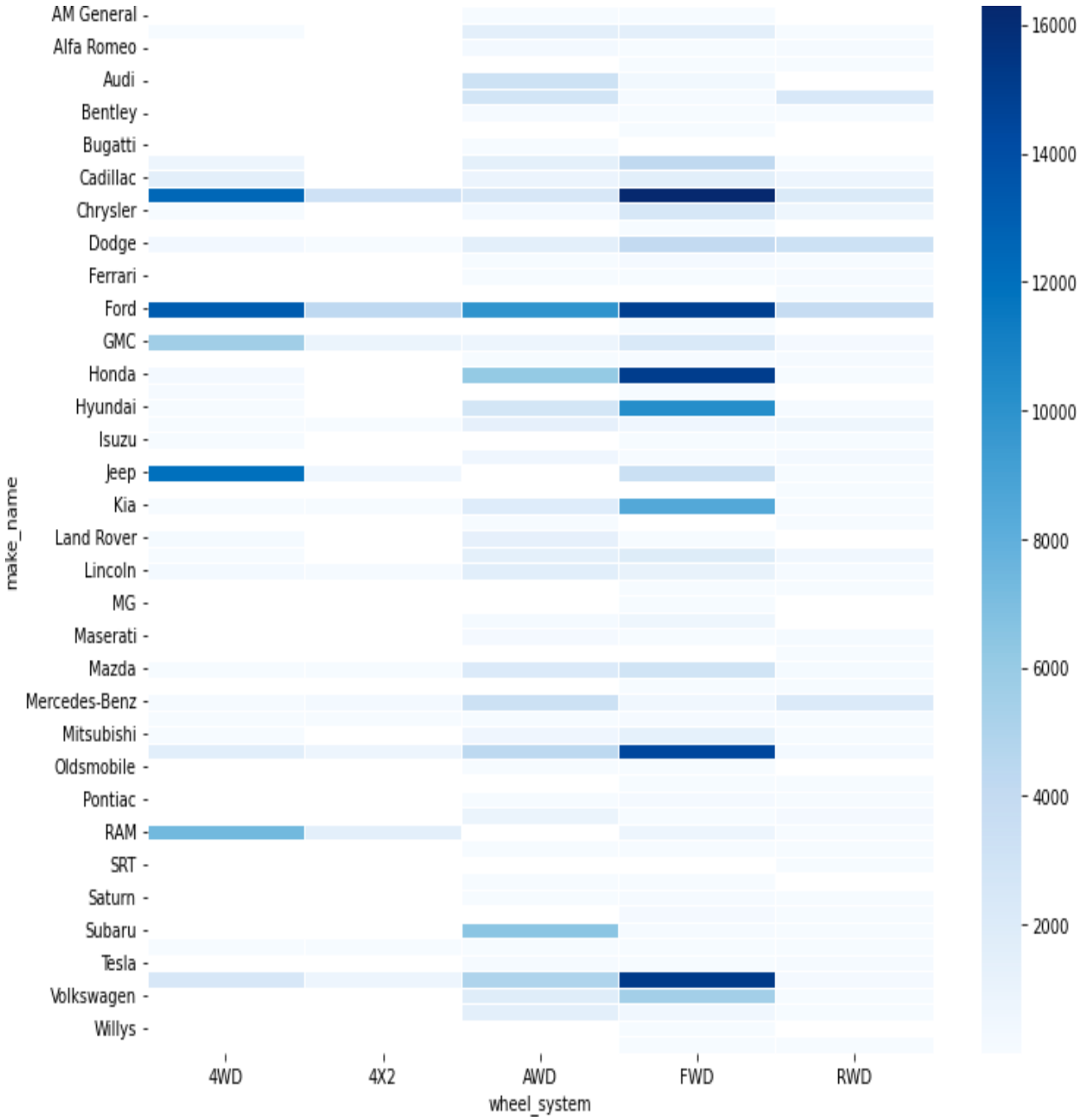
Via the above plotted heatmap we are trying to understand the preferred transmission system for each of the brands, we can see that brands like Ford and Cadillac have a higher customer referability when it comes to Automatic transmission while brands like Mitsubishi, Honda and Volkswagen attract in the Continuously Variable Transmission.

4. Make_Name Vs Fuel_Type



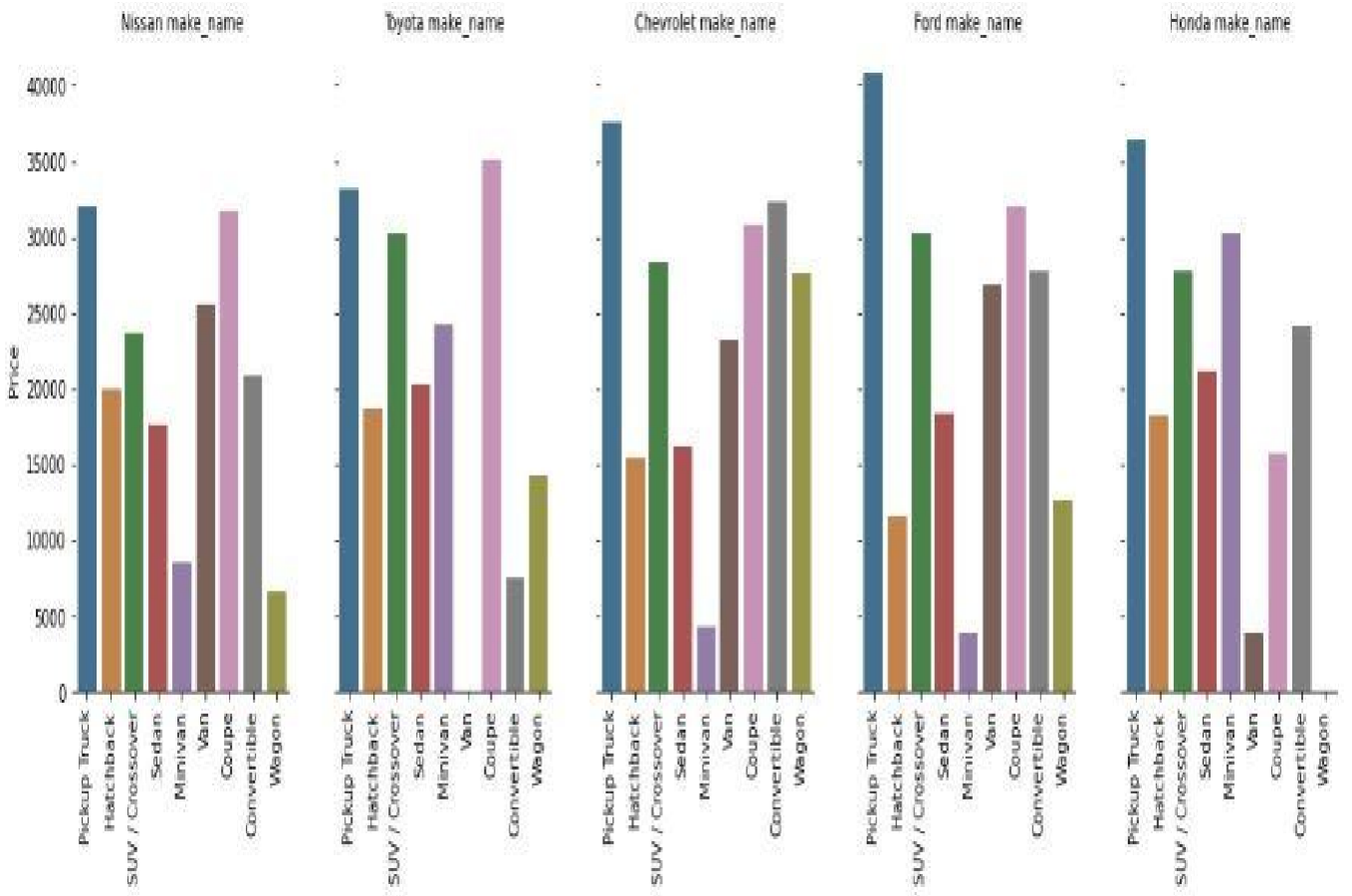
In the above plot we can see that Ford is preferred in the Flex Fuel, Gasoline and Hybrid category of fuel type, while RAM is preferred in Diesel category. For brands like Chrysler and Dodge preferred fuel typeis Flex Fuel and Gasoline.

5. Make Name Vs Wheel Type



From the above heatmap we can easily infer that in the Forward Wheel Drive system, Volkswagen, Oldsmobile, Honda, Ford, Chrysler are preferred by people while in 4 Wheel drive system brands like RAM, Jeep, Ford and Cadillac have a stronger hold. All wheel drive systems have preferred brands like Subaru, Mercedes-Benz, Audi, Bentley, Honda.

MULTIVARIATE ANALYSIS

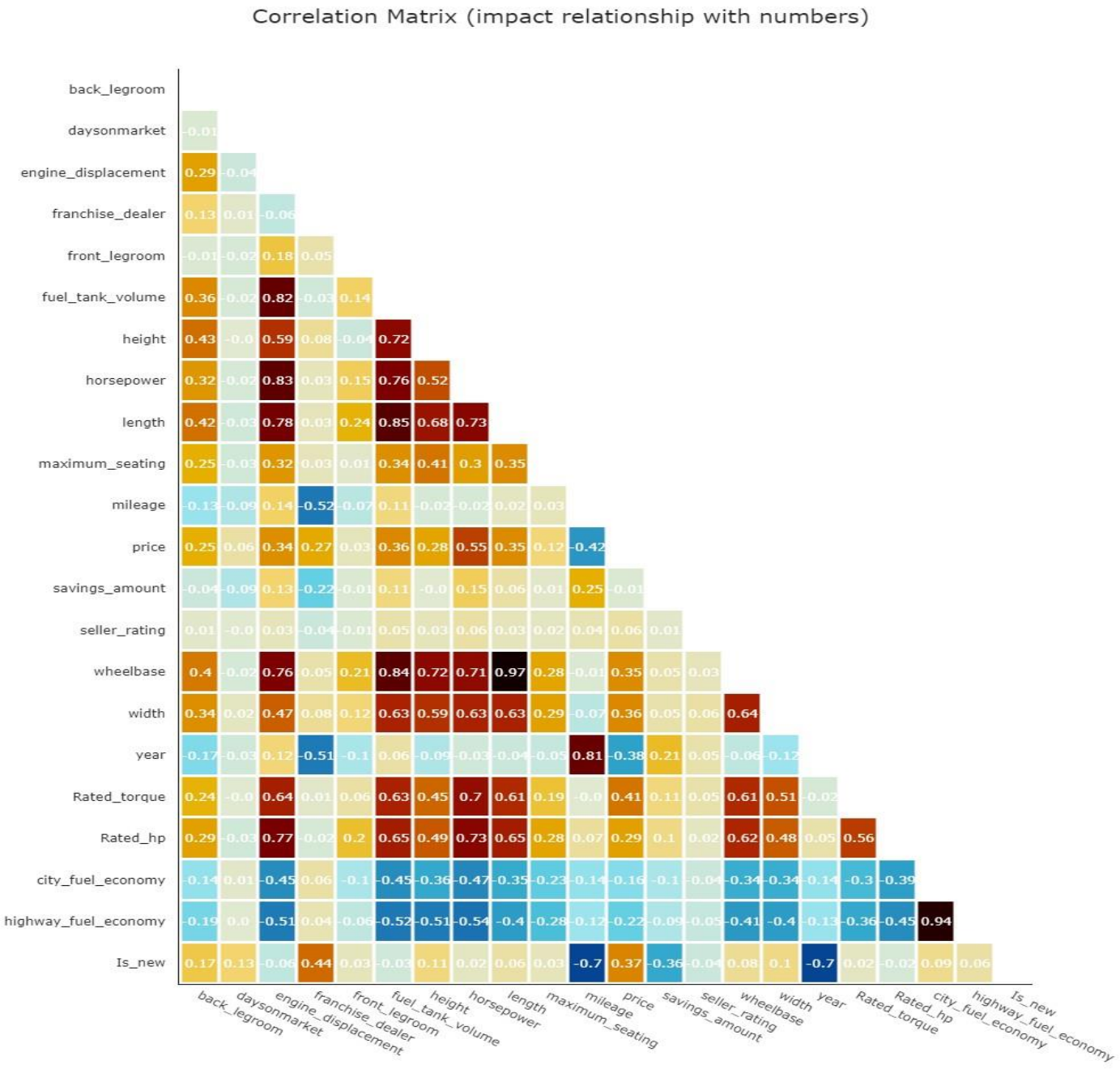


- This view provides the average price of different body types across differentmanufacturers.
- This could help the buyer to better understand the price comparison of cars acrossthe top 5 manufacturers

CORRELATION MATRIX

Heat-Map - Pearson Correlation Matrix :

(Assumption: For the Pearson correlation, both variables should be normally distributed. Other assumptions include linearity and homoscedasticity. It gives a measure of how much two numeric variables are linearly correlated. It tries to obtain a best fit line between two numeric variables and how close the points are to a fitted line.



INSIGHTS FROM CORRELATION MATRIX:

1. Engine Displacement-Fuel Tank Volume

There is no correlation between fuel tank size in 'lbs.' & engine size in 'L'. However, the full tank for any vehicle should

last at the least for a continuous drive of 300 miles or 450 kms or 10 hrs, depending on the engine's fuel consumption in miles/gallon or kms/lt.

• Bigger the vehicle, bigger the engine, higher is its Power and greater would be its fuel consumption.

fuel tank volume-height

• Height is a design feature and fuel tank volume are a performance feature technically there shouldn't be any relation between the same, but our data suggests otherwise.

2. Engine Displacement-Horsepower

Engine displacement means an engine can move more air and fuel, giving it the potential to make more horsepower. Whether or not it does make more power depends on a combination of the internal parts and the engine's size.

3. Fuel Tank Volume-Horsepower

As indicated earlier that the full tank for any vehicle should last at the least for a continuous drive of 300 miles or 450 kms or 10 hrs, depending on the engine's fuel consumption in miles/gallon or kms/lt. So, it is no surprise that fuel tank volume and horsepower are highly correlated, although there is no direct relation between them.

4. Engine-Displacement – Length

• Analysis of EPA data on 1977 automobiles shows that the rate of fuel consumption increases roughly linearly with automobile weight. This is largely the consequence of engine size that is proportional, on the average, to the 1.8th power of the weight. However, increased consumption due to large engine size is partially offset by a specific engine efficiency that improves linearly with increasing engine size.

• The specific consumption, measured as gal/(in3-lb-mi), is inversely proportional to engine displacement. It is suggested that sacrifice of body size (i.e. weight) to obtain improved overall fuel consumption is only one possible trade-off; an alternative might be sacrifice of performance while retaining size which might be more acceptable to one section of the market.

5. Fuel Tank Volume – Length

• Length is a design feature and fuel tank volume is a performance feature technically there shouldn't be any relation between the same, but our data suggests otherwise. Moreover, as suggested in length correlation if we increase length of the car our car's weight will increase thereby decreasing performance.

6. **Mileage-Year**

In our case the mileage is the no of miles the car has run, it is quite obvious, more the age of the car there is a higher chance that it would've run more

7. **Horsepower-Rated Torque**

Mathematically, horsepower equals torque multiplied by rpm. H = T x rpm/5252, where H is horsepower, T is pound-feet, rpm is how fast the engine is spinning, and 5252 is a constant that makes the units jibe. So, to make more power an engine needs to generate more torque, operate at higher rpm, or both.

Rated torque is actually computed using given horsepower parameters, while horsepower in our dataset is the original horsepower of the vehicle.

8. **Displacement-Rated Hp**

Generally speaking, the bigger (and stronger) the engine, the greater the displacement, leading to higher cc ratings. In contrast, “horsepower” measures the work an engine can do. One horsepower is officially equivalent to 746 watts of power.

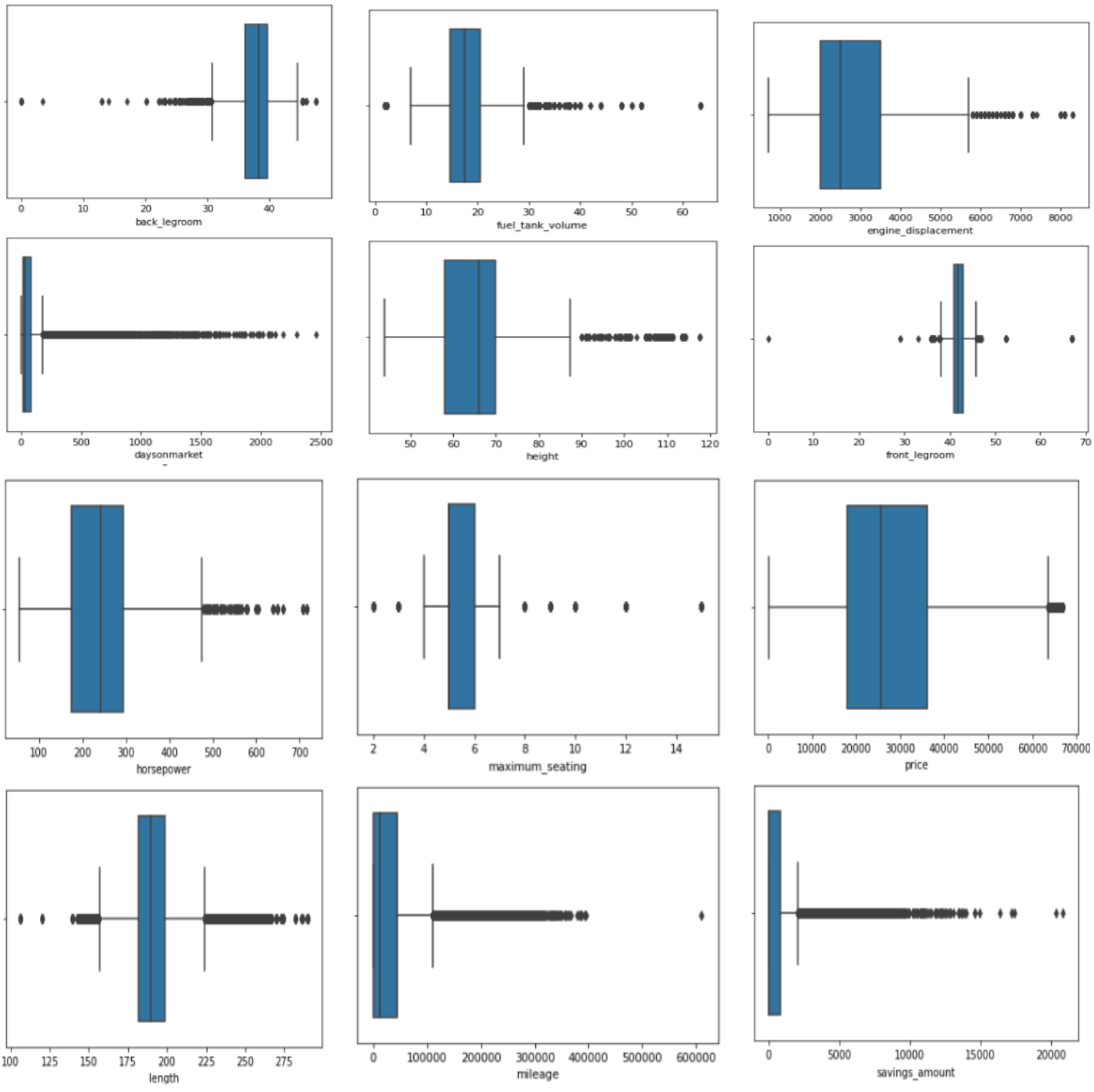
9. **Horsepower-Rated Hp**

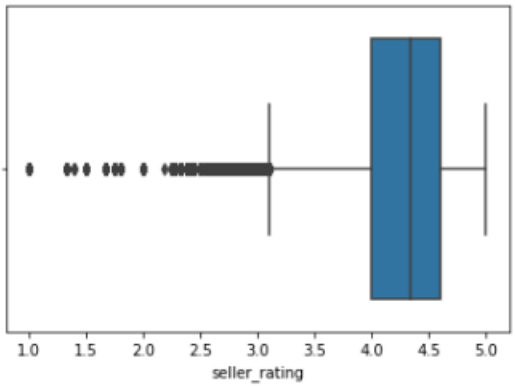
Horsepower is the original hp output of the engine, while rated hp is the calculated hp value based on given power inputs, here age of the vehicle affects the hp so we can see decrease of hp from original hp.

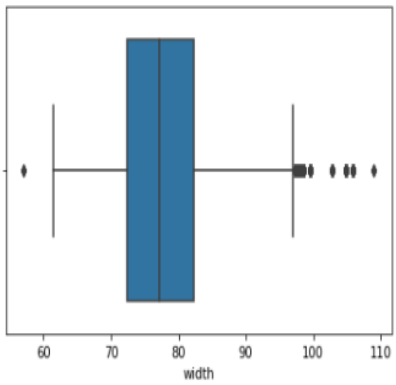
PRESENCE OF OUTLIERS AND OUTLIER TREATMENT:

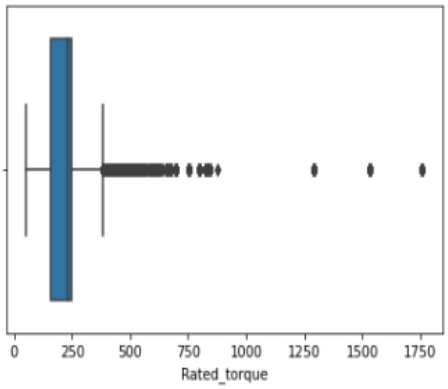
As we have seen from univariate analysis, from the distribution plot and boxplot between various variables we came to conclusion that in numerical columns there is lot of skewness whichindicates there is a presence of outliers which boxplot confirms it respectively.

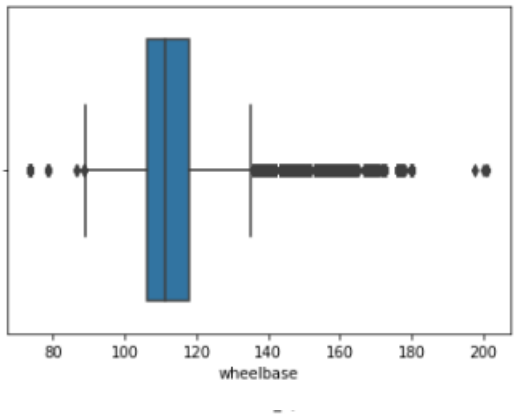
As we are building the regression model, outliers influence the model accuracy so, the extreme outliers are removed, using quantile wherever the extreme jump of outliers noticed we are removing those outliers

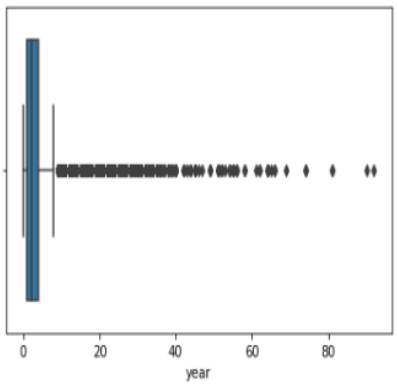


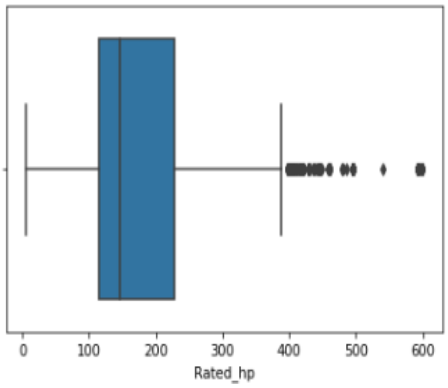

1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
seller_rating

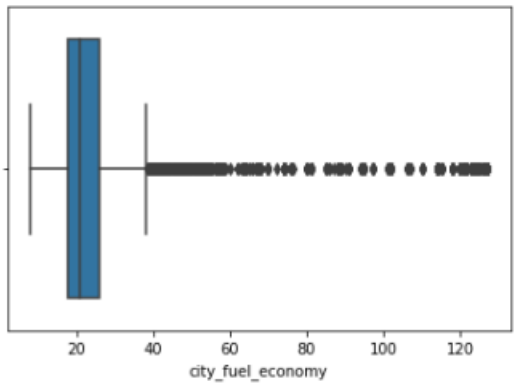

60
70
80
90
100
110
width

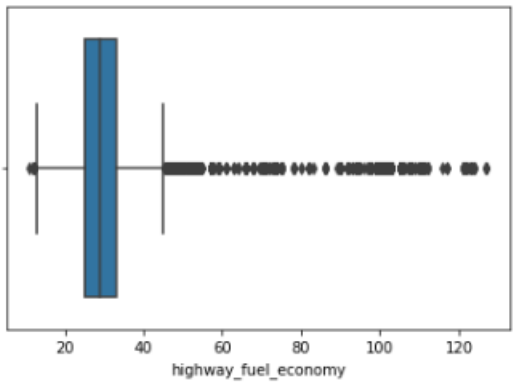

0
250
500
750
1000
1250
1500
1750
Rated_torque


80
100
120
140
160
180
200
wheelbase


0
20
40
60
80
year


0
100
200
300
400
500
600
Rated_hp


20
40
60
80
100
120
city_fuel_economy


20
40
60
80
100
120
highway_fuel_economy

Outlier Treatment:

We will remove the extreme as we do not want to lose the data.

```
[75] # Removing extreme outliers from back legroom
     df1=df1[df1['back_legroom']>20]
```

```
[76] # Removing extreme outliers from
     df2=df1[df1['daysonmarket']<2200]
     df2.shape
```

```
(282477, 46)
```

```
[77] # Removing ouliers from rated hp
     df3=df2[df2['Rated_hp']<500]
     df3.shape
```

```
(282428, 46)
```

```
[78]
     # Removing outliers from rated torque
     df4=df3[df3['Rated_torque']<1000]
```

```
[79] # Removing outliers from year
     df5=df4[df4['year']<70]
```

```
[80]
     # Removing outliers from wheelbase
     df6=df5[df5['wheelbase']<190]
     df6.shape
```

```
(282041, 46)
```

```
     # Removing outliers from seller rating
     df7=df6[df6['seller_rating']>2.2]
     df7.shape
```

```
(280884, 46)
```

```
[82] # Removing outliers from savings amount
     df8=df7[df7['savings_amount']<15000]
     df8.shape
```

```
(280879, 46)
```

```
[83] # Removing outliers from mileage
     df9=df8[df8['mileage']<500000]
```

```
[84]
     # Removing outliers from length
     df10=df9[df9['length']>130]
```

```
[85]
     # Removing outliers from horsepower
     df11=df10[df10['horsepower']<600]
```

```
[86] # Removing outliers from fuel tank volume
     df12=df11[df11['fuel_tank_volume']<43]
```

```
[87]
     # Removing outliers from front legroom
     df13=df12[df12['front_legroom']<50]
```

```
[88]

     # Removing outliers from engine displacement
     df14=df13[df13['engine_displacement']<7100]
```

```
[89] dffinal=df14.reset_index(drop=True)
     dffinal.shape
```

```
(279630, 46)
```

FEATURE ENGINEERING:

Feature engineering or feature extraction is the process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data. The motivation is to usethese extra features to improve the quality of results from a machine learning process, compared with supplying only the raw data to the machine learning process.

Torque is the capacity to do work, while power is how quickly some strenuous tasks can be accomplished. In other words, power is the rate of completing work (or applying torque) in a given amount of time. Mathematically, horsepower equals torque multiplied by rpm. H = T x rpm/5252, where H is horsepower, T is pound-feet, rpm is how fast the engine is spinning, and 5252 is a constant that makes the units equate. So, to make more power an engine needs to generate more torque, operate at higher rpm, or both.

Horsepower-

So, from the above given formula using excel (data>text to columns) we separated RPM from Horsepower and calculated rated torque as:
Torque = Horsepower x 5252 / rpm

Torque-

Similar to the previous column we separated RPM from Torque using excel (data>text to columns) and calculated rated Horsepower using below mentioned formula:
Horsepower = Torque x RPM / 5252

	power	torque
0	260 hp @ 6,000 RPM	240 lb-ft @ 4,400 RPM
1	370 hp @ 2,800 RPM	850 lb-ft @ 1,700 RPM
2	305 hp @ 4,900 RPM	379 lb-ft @ 3,600 RPM
3	166 hp @ 6,000 RPM	162 lb-ft @ 4,200 RPM

	Rated_hp	Rated_torque
0	275.133283	694.014286
1	259.786748	326.910204
2	123.381569	138.302667
3	175.932978	656.500000
4	298.990861	332.939286

Major options-

When we have received data, we received the column “major options” in the form of a list. Each element in the list represents extra add-on which was added to the car as per the request of the customer. We had around 150 different categories in the major options which we have categorized them into buckets namely-

1.Suspension packages

2.Entertainment package

3.Control package

4.Wheels package

5.Control packages

6.Headlamp packages

7.Mid and Luxury equipment group

8.Others

9.Seating package

10.high end package

11. sensors packages

All these 10 buckets are 10 different broad categories which were divided based on research. For each unique pre owned car we have mentioned the count of respective add-ons in their respective buckets. If there are no add-ons in the bucket, we mentioned the count as 0. The same process is repeated for all the pre-owned cars in the sample selected.

```
df['major_options'].head()
```

```
4]: 0    ['Leather Seats', 'Navigation System', 'Adapti...
1                                        ['Alloy Wheels']
2                                                unknown
3    ['Leather Seats', 'Navigation System', 'Alloy ...
4    ['Leather Seats', 'Navigation System', 'Alloy ...
Name: major_options, dtype: object
```

]:

	Mid and Luxury Equipment Group	Entertainment packages	control package	Wheels Package	Headlamp Package	others	Suspension Package	Seating package	sensor package	high end package
0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	2	1	0	1	0	2	1	1
3	0	0	2	1	0	0	0	2	1	1
4	0	0	0	0	0	0	0	0	0	0

Feature Transformations:

When we checked the skewness of the data, majority of the columns are positively skewed and around 4 columns were negatively skewed.

- For positively skewed columns, we have handled it using log transformation

```
df1=np.log(dfnum1[['engine_displacement','front_legroom','fuel_tank_volume','height','horsepower','length','maximum_seating',
 'wheelbase','width','Rated_torque','Rated_hp','city_fuel_economy','highway_fuel_economy']])
```

- For negatively skewed columns, we have handled it using square transformation

```
df3=np.squares(dfnum1[['daysonmarket','mileage','savings_amount','year']])
```

Why we need transformation?

Linear models are very sensitive to outliers. Having a more Gaussian distribution of the variables could be any day preferred. But in real world scenarios we can aim for near-Gaussian distribution with a tolerance level of +/- 0.5. We do transformations to either satisfy the linearity condition of independent variable or to satisfy the normality condition of the residuals.

Before Transformations-Skewness

back_legroom	-1.523535
daysonmarket	3.788580
engine_displacement	1.173055
front_legroom	1.411285
fuel_tank_volume	1.604048
height	0.715508
horsepower	0.590662
length	1.025027
maximum_seating	1.846953
mileage	1.903797
price	0.568104
savings_amount	2.888586
seller_rating	-1.482920
wheelbase	1.648792
width	0.990916
year	2.886025
Rated_torque	3.362094
Rated_hp	0.585690
city_fuel_economy	6.086126
highway_fuel_economy	3.719403

dtype: float64

After Transformation-Skewness

engine_displacement	0.338172
front_legroom	0.086362
fuel_tank_volume	0.208340
height	0.283879
horsepower	-0.034290
length	0.696700
maximum_seating	-0.419774
wheelbase	1.380992
width	0.725526
Rated_torque	0.155104
Rated_hp	-2.295028
city_fuel_economy	1.266188
highway_fuel_economy	0.658344
back_legroom	-0.303677
seller_rating	-0.928874
Mid and Luxury Equipment Group	5.972317
Entertainment packages	2.079374
control package	0.837924
Wheels Package	-0.475975
sensor package	0.429699
high end package	0.429699
daysonmarket	1.338698
mileage	0.653826
savings_amount	0.969587
year	1.011987

dtype: float64

FEATURE SCALING:

We have scaled the entire data in order to bring down the range of the individual columns in the dataset. Scaling of the data is helpful because-

Scaling brings down the range of values which enables the algorithm to converge faster and lead to better predictions. It mainly reduces the time taken to train the data.

For distance-based algorithms, it is always better to scale the data as the higher range feature would dominate during the distance computation making the distance value unreliable. So, if the features are in similar range the distance computation is more accurate.

```
[133] dfnum2.shape
```

```
(279630, 29)
```

```
[134] from sklearn.preprocessing import StandardScaler
      sc=StandardScaler()
      df_sc=sc.fit_transform(dfnum2)
      df_sc=pd.DataFrame(df_sc,columns=dfnum2.columns)
```

```
[135] dffinal=pd.concat([df_sc,df_encod,yconcat],axis=1)
```

STATISTICAL TESTS

1. We performed statistical tests to know the significance of independent variables on a target variable.

As our target variable is numeric, we performed numerical (independent feature) vs numerical (target)

With Pearson's test to get significance of variables.

```
# Numerical vs Numerical
HN = 'Variables doesnot have corelation'
HA = 'Variables are corelated'
```

```
for i in main.select_dtypes(include = np.number).columns:
    stat , p_val = stats.pearsonr(main[i] , np.sqrt(main['price']))
    if p_val > 0.05:
        print('{} Is insignificant p_val is{}'.format(i ,p_val))
    else :
        print('{} Is significant p_val is {}'.format(i , p_val))
```

```
back_legroom Is significant p_val is 0.0
daysonmarket Is significant p_val is 6.43234297774452e-76
engine_displacement Is significant p_val is 0.0
front_legroom Is significant p_val is 0.0
fuel_tank_volume Is significant p_val is 0.0
height Is significant p_val is 0.0
horsepower Is significant p_val is 0.0
length Is significant p_val is 0.0
maximum_seating Is significant p_val is 0.0
mileage Is significant p_val is 0.0
price Is significant p_val is 0.0
savings_amount Is significant p_val is 0.0
seller_rating Is significant p_val is 1.859368981415221e-130
wheelbase Is significant p_val is 0.0
width Is significant p_val is 0.0
year Is significant p_val is 0.0
Rated_torque Is significant p_val is 0.0
Rated_hp Is significant p_val is 0.0
city_fuel_economy Is significant p_val is 0.0
highway_fuel_economy Is significant p_val is 0.0
```

P_values of Pearson test for all numeric variables

Interpretation: p_values of most of the features are less than significance level.

Hence, we reject null, the features are correlated with the target variable.

2. We performed categorical vs categorical independent features to know the weather the categorical features Dependent or independent of each other, if they are dependent, we can drop any one column by which we can reduce multicollinearity simultaneously we can reduce the creation of dummies and Model won't become complex.

```
table=pd.crosstab(main['transmission'],main['transmission_display'])
observed=table.values
observed
```

```
4]: array([[   449,      0,      0,     40,      9,     35,      0,   3601,
              792,      0,      0,   4187,   1196,      0,      0,  41663,
             8517,      0,      0,      0,      0,   3965,    172,      0,
                0,      0,  30607,   1136,      0,      0,  13639,    376,
           121411,      0,      0],
           [     0,     53,      0,      0,      0,      0,      0,      0,
                 0,      5,      0,      0,      0,      0,      0,      0,
                 0,     58,      0,      0,      0,      0,      0,    197,
                 0,      0,      0,      0,     20,      0,      0,      0,
                 0,  44944,      0],
           [     0,      0,     30,      0,      0,      0,      0,      0,
                 0,      0,      0,      0,      0,      0,      0,      0,
                 0,      0,    573,      0,      0,      0,      0,      0,
               164,      0,      0,      0,      0,    289,      0,      0,
                 0,      0,      0],
           [     0,      0,      0,      0,      0,      0,      3,      0,
                 0,      0,     24,      0,      0,    764,     71,      0,
                 0,      0,      0,   2034,     74,      0,      0,      0,
                 0,     40,      0,      0,      0,      0,      0,      0,
                 0,      0,   1541]], dtype=int64)
```

```
chistat,pval,dof,arr=stats.chi2_contingency(observed=observed,correction=False)
pval
```

```
5]: 0.0
```

```
#This shows that transmission and transmission display are dependent on each other
```

So as the variables are dependent on each other, transmission variable derived from transmission display, so we dropped transmission column and retained transmission display column.

3. We performed categorical (independent feature) vs Numerical feature (dependent) with more than 2 samples we used Anova test to check whether the categorical feature more than 2 subclass has an influence on target or not.

```
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
```

```
#H0 : The averages of all treatments are the same.
#H1 : At least one treatment has a different average.
mod=ols('price ~ body_type',data=main).fit()
aov_table=sm.stats.anova_lm(mod,typ=2)
print('\n',aov_table)
```

```
                  sum_sq        df            F  PR(>F)
body_type  1.016353e+13       8.0  8756.175625     0.0
Residual   4.101283e+13  282670.0          NaN     NaN
```

```
#Here p<0.05,atleast one of the means is significantly different
```

```
#Here we see that means are quite different from each other
main.groupby(['body_type'])['price'].mean()
```

```
]: body_type
   Convertible        26658.803357
   Coupe              27552.470160
   Hatchback          17027.870854
   Minivan            24882.945927
   Pickup Truck       38836.554833
   SUV / Crossover    28857.646756
   Sedan              20925.912650
   Van                28646.360050
   Wagon              18685.262124
   Name: price, dtype: float64
```

We can see that body type has a influence on target price.

```
mod=ols('price ~ fuel_type',data=main).fit()
aov_table=sm.stats.anova_lm(mod,typ=2)
print('\n',aov_table)
```

```
                    sum_sq         df             F  PR(>F)
fuel_type   1.338488e+12        6.0  1265.279836     0.0
Residual    4.983787e+13   282672.0           NaN     NaN
```

```
#Here p<0.05,atleast one of the means is significantly different
```

```
#Here we see that means are quite different from each other
main.groupby(['fuel_type'])['price'].mean()
```

```
fuel_type
Biodiesel                 49877.159184
Compressed Natural Gas    20987.875000
Diesel                    39051.744138
Electric                  29003.907291
Flex Fuel Vehicle         26525.852344
Gasoline                  27441.176696
Hybrid                    27148.554442
Name: price, dtype: float64
```

So, we can see that the means are different for different categories, so fuel type has effect on price.

ENCODING:

Encoding is a technique of converting categorical variables into numerical values so that itcould be easily fitted to a machine learning model.

Dummy encoding:

In dummy encoding, for each level of a categorical feature, we create a new variable. Eachcategory is mapped with a binary variable containing either 0 or 1. Here, 0 represents the absence, and 1 represents the presence of that category.

```
df_encod.head()
```

	body_type_Coupe	body_type_Hatchback	body_type_Minivan	body_type_Pickup Truck	body_type_SUV / Crossover	body_type_Sedan	body_type_Van	body_type_Wagon	engine_type_H4 Hybrid	engine_type_H6	...
0	0	0	0	1	0	0	0	0	0	0	...
1	0	1	0	0	0	0	0	0	0	0	...
2	0	0	0	1	0	0	0	0	0	0	...
3	0	0	0	1	0	0	0	0	0	0	...
4	1	0	0	0	0	0	0	0	0	0	...

TRAIN TEST SPLIT:

1. Splitting data into Training dataset and Test Dataset. (70:30)
2. A training set is implemented to build up a model, while a test (or validation) set is to validate the model built

```
y=dffinal['price']
x=dffinal.drop('price',axis=1)
x=sm.add_constant(x)

X_train,X_test,Y_train,Y_test=train_test_split(x,y,random_state=10,test_size=0.3)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(195741, 130)
(83889, 130)
(195741,)
(83889,)
```

MODEL FITTING:

LINEAR REGRESSION (BASE MODEL)

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables.

The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

Naming the Variables. There are many names for a regression's dependent variable. It may be called an outcome variable, criterion variable, endogenous variable, or regress and The independent variables can be called exogenous variables, predictor variables, or regressors.

Three major uses for regression analysis are

(1) determining the strength of predictors,

(2) forecasting an effect, and

(3) trend forecasting.

Multiple Linear Regression:

Multiple regression model is used when multiple predictor variables [X1 , X2 , X3 , …, Xn] are used to predict the response variable Y

$$y = \beta 0 + \beta 1\ x1 + \beta 2\ x2 + \beta 3\ x3 + \ldots + \beta n\ x\ n + \varepsilon$$

β0 , β1 , β2 , β3 , …, βn are the parameters of the linear regression model with n independent variable

We ran linear regression on our model, we got the output as follows

OLS Regression Results

Dep. Variable:	price	R-squared:	0.807
Model:	OLS	Adj. R-squared:	0.807
Method:	Least Squares	F-statistic:	6704.
Date:	Tue, 12 Jul 2022	Prob (F-statistic):	0.00
Time:	11:23:39	Log-Likelihood:	-1.9766e+06
No. Observations:	195741	AIC:	3.953e+06
Df Residuals:	195618	BIC:	3.955e+06
Df Model:	122		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.02e+04	494.745	81.259	0.000	3.92e+04	4.12e+04
engine_displacement	-1222.4112	43.611	-28.030	0.000	-1307.888	-1136.934
front_legroom	-344.9901	17.212	-20.044	0.000	-378.725	-311.255
fuel_tank_volume	1274.3044	41.493	30.711	0.000	1192.980	1355.629
height	231.9128	51.737	4.483	0.000	130.509	333.316
horsepower	5618.7194	45.506	123.473	0.000	5529.529	5707.910
length	-335.1239	68.266	-4.909	0.000	-468.924	-201.323
maximum_seating	247.0186	19.903	12.411	0.000	208.009	286.028
wheelbase	878.8071	73.728	11.920	0.000	734.302	1023.312
width	330.0958	24.356	13.553	0.000	282.359	377.833
Rated_torque	98.9095	32.853	3.011	0.003	34.519	163.300
Rated_hp	-367.5369	21.082	-17.434	0.000	-408.856	-326.218
city_fuel_economy	2828.4851	69.105	40.931	0.000	2693.042	2963.928
highway_fuel_economy	-2255.8770	61.337	-36.779	0.000	-2376.095	-2135.659
back_legroom	308.6698	17.696	17.443	0.000	273.986	343.354
seller_rating	524.8761	13.428	39.089	0.000	498.558	551.194
Mid and Luxury Equipment Group	-231.9430	15.906	-14.582	0.000	-263.118	-200.768
Entertainment packages	-104.3383	15.189	-6.870	0.000	-134.107	-74.569
control package	1251.0696	17.122	73.070	0.000	1217.512	1284.627
Wheels Package	-551.5564	14.788	-37.297	0.000	-580.541	-522.571
Headlamp Package	-59.3689	13.593	-4.368	0.000	-86.010	-32.728
others	290.3895	15.820	18.356	0.000	259.383	321.396
Suspension Package	703.6377	14.765	47.657	0.000	674.699	732.576
Seating package	780.9937	18.384	42.482	0.000	744.961	817.026
sensor package	-23.9012	8.548	-2.796	0.005	-40.655	-7.147
high end package	-23.9012	8.548	-2.796	0.005	-40.655	-7.147
daysonmarket	19.2153	14.104	1.362	0.173	-8.428	46.859
mileage	-4867.7877	32.362	-150.418	0.000	-4931.216	-4804.360
savings_amount	310.9361	17.592	17.675	0.000	276.456	345.416
year	-3199.2275	33.802	-94.647	0.000	-3265.478	-3132.977
body_type_Coupe	-3429.5894	181.895	-18.855	0.000	-3786.100	-3073.079
body_type_Hatchback	-4031.3173	196.915	-20.472	0.000	-4417.267	-3645.368
body_type_Minivan	-8519.6785	215.862	-39.468	0.000	-8942.763	-8096.594
body_type_Pickup Truck	-5570.5279	211.235	-26.371	0.000	-5984.543	-5156.513
body_type_SUV / Crossover	-6020.0718	192.942	-31.201	0.000	-6398.234	-5641.910
body_type_Sedan	-5290.4046	172.533	-30.663	0.000	-5628.566	-4952.243
body_type_Van	-6558.3838	264.477	-24.798	0.000	-7076.752	-6040.016
body_type_Wagon	-4117.3376	219.433	-18.764	0.000	-4547.420	-3687.255
transmission_display_8-Speed Dual Clutch	-2117.4360	541.712	-3.909	0.000	-3179.180	-1055.693
transmission_display_9-Speed Automatic	-4214.2197	363.358	-11.598	0.000	-4926.393	-3502.047
transmission_display_9-Speed Automatic Overdrive	-6328.3631	518.477	-12.206	0.000	-7344.566	-5312.160
transmission_display_Automatic	-4558.7127	358.662	-12.710	0.000	-5261.682	-3855.743
transmission_display_Continuously Variable Transmission	-5728.5495	362.900	-15.785	0.000	-6439.826	-5017.273
transmission_display_Manual	-2591.6762	402.329	-6.442	0.000	-3380.232	-1803.120
wheel_system_4X2	-3605.6271	74.289	-48.535	0.000	-3751.232	-3460.022
wheel_system_AWD	-216.9931	58.835	-3.688	0.000	-332.309	-101.677
wheel_system_FWD	-3145.7694	60.336	-52.137	0.000	-3264.027	-3027.511
wheel_system_RWD	-2443.8458	84.412	-28.952	0.000	-2609.290	-2278.401
Is_new_True	-85.5560	54.075	-1.582	0.114	-191.542	20.430

Omnibus:	25506.690	Durbin-Watson:	1.991
Prob(Omnibus):	0.000	Jarque-Bera (JB):	54267.496
Skew:	0.800	Prob(JB):	0.00
Kurtosis:	5.024	Cond. No.	1.61e+16

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.5e-27. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

After Removing the Multicollinearity, we ran again the model:

OLS Regression Results

Dep. Variable:	price	R-squared:	0.741
Model:	OLS	Adj. R-squared:	0.741
Method:	Least Squares	F-statistic:	5179.
Date:	Tue, 12 Jul 2022	Prob (F-statistic):	0.00
Time:	11:30:58	Log-Likelihood:	-2.0054e+06
No. Observations:	195741	AIC:	4.011e+06
Df Residuals:	195632	BIC:	4.012e+06
Df Model:	108		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.223e+04	568.925	56.655	0.000	3.11e+04	3.33e+04
daysonmarket	0.5560	0.149	3.729	0.000	0.264	0.848
mileage	-0.1229	0.001	-221.319	0.000	-0.124	-0.122
savings_amount	0.3264	0.018	17.896	0.000	0.291	0.362
Rated_hp	1.4089	0.322	4.377	0.000	0.778	2.040
city_fuel_economy	-117.1675	4.524	-25.898	0.000	-126.035	-108.300
Mid and Luxury Equipment Group	452.7368	102.519	4.416	0.000	251.803	653.671
Entertainment packages	-129.4693	34.345	-3.770	0.000	-196.785	-62.154
control package	1984.9328	21.946	90.445	0.000	1941.919	2027.947
Wheels Package	-2292.5842	34.600	-66.259	0.000	-2360.400	-2224.768
Headlamp Package	-57.2356	278.117	-0.206	0.837	-602.338	487.867
others	305.3630	14.733	20.727	0.000	276.487	334.239
Suspension Package	4097.5517	79.589	51.484	0.000	3941.559	4253.544
Seating package	1423.8765	22.929	62.100	0.000	1378.937	1468.816
sensor package	-86.3528	22.548	-3.830	0.000	-130.546	-42.160
body_type_Coupe	-321.3190	208.428	-1.542	0.123	-729.832	87.194
body_type_Hatchback	-2682.2181	211.305	-12.694	0.000	-3096.370	-2268.066
body_type_Minivan	-1326.3538	217.037	-6.111	0.000	-1751.742	-900.966
body_type_Pickup Truck	3863.8084	203.657	18.972	0.000	3464.646	4262.971
body_type_SUV / Crossover	-188.6101	191.677	-0.984	0.325	-564.292	187.072
body_type_Sedan	-1522.8485	190.620	-7.989	0.000	-1896.459	-1149.237
body_type_Van	397.4289	225.105	1.766	0.077	-43.772	838.630
body_type_Wagon	-436.2146	235.935	-1.849	0.064	-898.643	26.213
engine_type_H4 Hybrid	-9254.9427	2804.661	-3.300	0.001	-1.48e+04	-3757.875
engine_type_H6	5271.4484	534.529	9.862	0.000	4223.784	6319.113
engine_type_I2	2506.2857	865.883	2.894	0.004	809.176	4203.395
engine_type_I3	-4861.2287	163.002	-29.823	0.000	-5180.708	-4541.749
engine_type_I3 Hybrid	1.602e+04	5695.398	2.813	0.005	4860.476	2.72e+04
engine_type_I4	-34.5150	114.760	-0.301	0.764	-259.442	190.413
engine_type_I4 Compressed Natural Gas	-5568.4774	2710.094	-2.055	0.040	-1.09e+04	-256.758
engine_type_I4 Diesel	-1.747e+04	2083.813	-8.386	0.000	-2.16e+04	-1.34e+04
engine_type_I4 Flex Fuel Vehicle	-3333.9708	230.983	-14.434	0.000	-3786.693	-2881.249
engine_type_I4 Hybrid	-2773.2340	1319.118	-2.102	0.036	-5358.673	-187.795
engine_type_I5	1239.6298	370.346	3.347	0.001	513.760	1965.500
engine_type_I5 Biodiesel	7485.8207	2055.326	3.642	0.000	3457.430	1.15e+04
engine_type_I5 Diesel	-7945.5869	4441.530	-1.789	0.074	-1.67e+04	759.706
engine_type_I6	6860.1198	209.211	32.791	0.000	6450.072	7270.167
engine_type_I6 Diesel	-449.6821	2068.906	-0.217	0.828	-4504.688	3605.324
engine_type_I6 Hybrid	5.83e-11	1.83e-11	3.182	0.001	2.24e-11	9.42e-11
engine_type_R2	-4661.1993	2061.684	-2.261	0.024	-8702.051	-620.348
engine_type_V10	2301.2666	1161.722	1.981	0.048	24.320	4578.214
engine_type_V12	9445.9628	1908.619	4.949	0.000	5705.115	1.32e+04
engine_type_V6	4591.8705	122.575	37.462	0.000	4351.626	4832.115
transmission_display_Manual	3969.0765	480.950	8.253	0.000	3026.426	4911.726
wheel_system_4X2	-3409.4305	85.040	-40.092	0.000	-3576.106	-3242.755
wheel_system_AWD	-664.0673	64.255	-10.335	0.000	-790.006	-538.129
wheel_system_FWD	-5318.7389	64.491	-82.473	0.000	-5445.139	-5192.339
wheel_system_RWD	-1756.0306	92.348	-19.015	0.000	-1937.030	-1575.031
Is_new_True	4949.4092	48.337	102.394	0.000	4854.670	5044.148

Omnibus:	25176.426	Durbin-Watson:	1.991
Prob(Omnibus):	0.000	Jarque-Bera (JB):	48777.695
Skew:	0.823	Prob(JB):	0.00
Kurtosis:	4.809	Cond. No.	1.22e+16

```
# still the multicollinearity is failing we cannot proceed with this model
```

```
# After Model Assumptions:
```

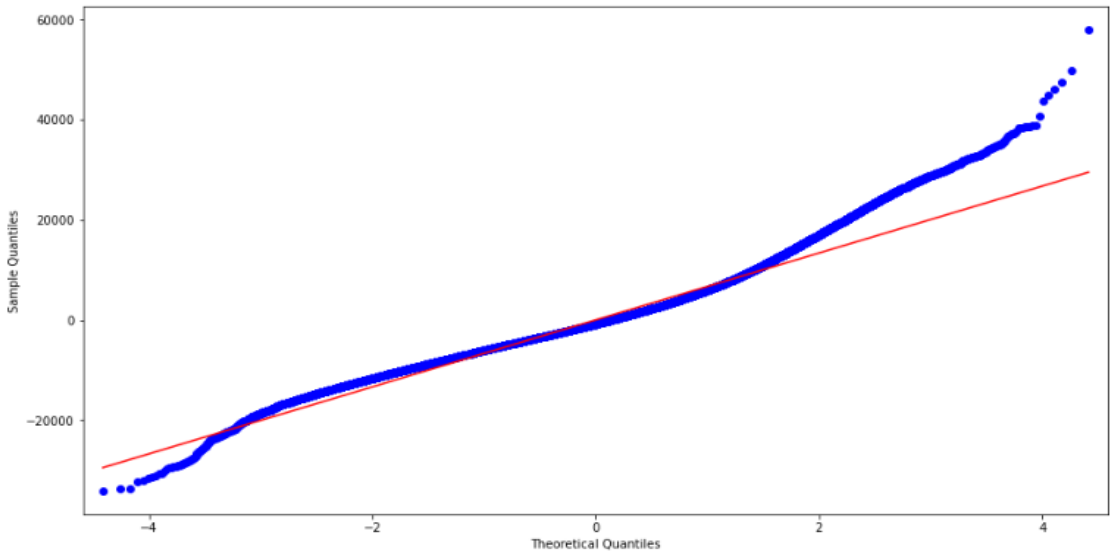
```
# 1) Autocorelation - from above summary we observe that value obtained from the Durbin Watson Test statistic is 2,
#                     Thus we coclude there is no auto corelation in data
```

```
# assumption says there should be Homoskedicity present in data:

# 2) Hetroskedicity - Ho= There is Homoskedicity present in data
#                     Ha= There is no Homoskedicity present in data
name=['Fvalue','P-value']
test=sms.het_breuschpagan(mlrmodel3.resid,mlrmodel3.model.exog)
lzip(name,test[2:])

# we observe that p value < 0.05 reject null, means there is no Homoskedicity in data
# our assumption is failing
```

```
[('Fvalue', 228.12837899994378), ('P-value', 0.0)]
```

```
# Assumption test for Normality - as per assumption residuals should be normally distributed.
# lets check with Q-Q plot
```

```
plt.rcParams['figure.figsize']=[15,8]
qqplot(mlrmodel3.resid,line='r')
plt.show()
```




```
# jarque Bera test

# Ho= Residuals are normally distributed
# Ha= Residuals are not normally distributed

# from summary report we can see jarque bera p value < 0.05 reject null , we can say residuals are not normally distributed
# our assumption is failing
```

	Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
0	Base Model-Linear Regression with All Variables	0.806997	0.806877	5877.4892	5863.9995
1	Linear Regression After Removing Multicollinea...	0.740859	0.740741	6810.4795	6804.0691

So, we see that, all our linear Regressions assumptions were failing, so we try to improvise by various feature selection techniques and regularization techniques to avoid overfitting.

Backward feature Selection Technique:

OLS Regression Results

Dep. Variable:	price	R-squared:	0.807
Model:	OLS	Adj. R-squared:	0.807
Method:	Least Squares	F-statistic:	6704.
Date:	Tue, 12 Jul 2022	Prob (F-statistic):	0.00
Time:	11:36:10	Log-Likelihood:	-1.9766e+06
No. Observations:	195741	AIC:	3.953e+06
Df Residuals:	195618	BIC:	3.955e+06
Df Model:	122		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.02e+04	494.745	81.259	0.000	3.92e+04	4.12e+04
engine_displacement	-1222.4112	43.611	-28.030	0.000	-1307.888	-1136.934
front_legroom	-344.9901	17.212	-20.044	0.000	-378.725	-311.255
fuel_tank_volume	1274.3044	41.493	30.711	0.000	1192.980	1355.629
height	231.9128	51.737	4.483	0.000	130.509	333.316
horsepower	5618.7194	45.506	123.473	0.000	5529.529	5707.910
length	-335.1239	68.266	-4.909	0.000	-468.924	-201.323
maximum_seating	247.0186	19.903	12.411	0.000	208.009	286.028
wheelbase	878.8071	73.728	11.920	0.000	734.302	1023.312
width	330.0958	24.356	13.553	0.000	282.359	377.833
Rated_torque	98.9095	32.853	3.011	0.003	34.519	163.300
Rated_hp	-367.5369	21.082	-17.434	0.000	-408.856	-326.218
city_fuel_economy	2828.4851	69.105	40.931	0.000	2693.042	2963.928
highway_fuel_economy	-2255.8770	61.337	-36.779	0.000	-2376.095	-2135.659
back_legroom	308.6698	17.696	17.443	0.000	273.986	343.354
seller_rating	524.8761	13.428	39.089	0.000	498.558	551.194
Mid and Luxury Equipment Group	-231.9430	15.906	-14.582	0.000	-263.118	-200.768
Entertainment packages	-104.3383	15.189	-6.870	0.000	-134.107	-74.569
control package	1251.0696	17.122	73.070	0.000	1217.512	1284.627
Wheels Package	-551.5564	14.788	-37.297	0.000	-580.541	-522.571
Headlamp Package	-59.3689	13.593	-4.368	0.000	-86.010	-32.728
others	290.3895	15.820	18.356	0.000	259.383	321.396
Suspension Package	703.6377	14.765	47.657	0.000	674.699	732.576
Seating package	780.9937	18.384	42.482	0.000	744.961	817.026
sensor package	-23.9012	8.548	-2.796	0.005	-40.655	-7.147
high end package	-23.9012	8.548	-2.796	0.005	-40.655	-7.147
daysonmarket	19.2153	14.104	1.362	0.173	-8.428	46.859
mileage	-4867.7877	32.362	-150.418	0.000	-4931.216	-4804.360
savings_amount	310.9361	17.592	17.675	0.000	276.456	345.416
year	-3199.2275	33.802	-94.647	0.000	-3265.478	-3132.977
body_type_Coupe	-3429.5894	181.895	-18.855	0.000	-3786.100	-3073.079
body_type_Hatchback	-4031.3173	196.915	-20.472	0.000	-4417.267	-3645.368
body_type_Minivan	-8519.6785	215.862	-39.468	0.000	-8942.763	-8096.594
body_type_Pickup Truck	-5570.5279	211.235	-26.371	0.000	-5984.543	-5156.513
body_type_SUV / Crossover	-6020.0718	192.942	-31.201	0.000	-6398.234	-5641.910
body_type_Sedan	-5290.4046	172.533	-30.663	0.000	-5628.566	-4952.243
body_type_Van	-6558.3838	264.477	-24.798	0.000	-7076.752	-6040.016
body_type_Wagon	-4117.3376	219.433	-18.764	0.000	-4547.420	-3687.255
engine_type_H4 Hybrid	-1944.8424	2421.010	-0.803	0.422	-6689.964	2800.279
engine_type_H6	3976.0246	464.368	8.562	0.000	3065.874	4886.175
engine_type_I2	6451.9537	806.303	8.002	0.000	4871.619	8032.289
engine_type_I3	-3282.6003	149.598	-21.943	0.000	-3575.808	-2989.393
engine_type_I3 Hybrid	1.915e+04	4916.698	3.895	0.000	9513.030	2.88e+04

Omnibus:	25506.690	Durbin-Watson:	1.991
Prob(Omnibus):	0.000	Jarque-Bera (JB):	54267.496
Skew:	0.800	Prob(JB):	0.00
Kurtosis:	5.024	Cond. No.	1.61e+16

	Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
2	Linear Regression with Backward Feature Selection	0.806997	0.806877	5877.4892	5863.9995

We saw, till now whatever models we ran, our model is suffering from overfitting, we will use Regularization technique.

LASSO REGGRESSION:

```
from sklearn.metrics import r2_score
lasso = Lasso()
lasso_model = lasso.fit(X_train , Y_train)
```

```
update_score_card(algorithm_name='Lasso regression', model=lasso_model)
```

```
tuned_paramaters = [{'alpha':[1e-15,1e-14,1e-13,1e-12,1e-11,1e-10,1e-9,1e-8,1e-7,1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1,2,3,4,5,10,20]}]
lasso_grid = GridSearchCV(estimator = lasso,
                          param_grid = tuned_paramaters,
                          cv = 3)
lasso_grid_model =lasso_grid.fit(X_train, Y_train)
```

```
lasso_grid_model.best_params_
```

```
{'alpha': 0.01}
```

```
lst = Lasso(**lasso_grid_model.best_params_)
lasso_tune = lst.fit(X_train,Y_train)
```

RIDGE REGRESSION:

```
ridge = Ridge(alpha=1)
ridge_model = ridge.fit(X_train , Y_train)
```

```
update_score_card(algorithm_name='Ridge regression', model=ridge_model)
```

```
tuned_paramaters = [{'alpha':[1e-15,1e-14,1e-13,1e-12,1e-11,1e-10,1e-9,1e-8,1e-7,1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1,2,3,4,5,10,20]}]
ridge_grid = GridSearchCV(estimator = ridge,
                          param_grid = tuned_paramaters,
                          cv = 3)
ridge_grid_model =ridge_grid.fit(X_train, Y_train)
```

```
ridge_grid_model.best_params_
```

```
{'alpha': 1e-13}
```

```
rst = Ridge(**ridge_grid_model.best_params_)
ridge_tune = rst.fit(X_train,Y_train)
```

Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
Lasso regression	0.806268	0.806140	5888.5843	5877.1789
Lasso regression Tuned	0.806995	0.806868	5877.5130	5864.1020
Ridge regression	0.806986	0.806859	5877.6605	5864.7515
Ridge regression Tuned	0.806997	0.806870	5877.4892	5863.9739

SUMMARY OF LINEAR MODELS:

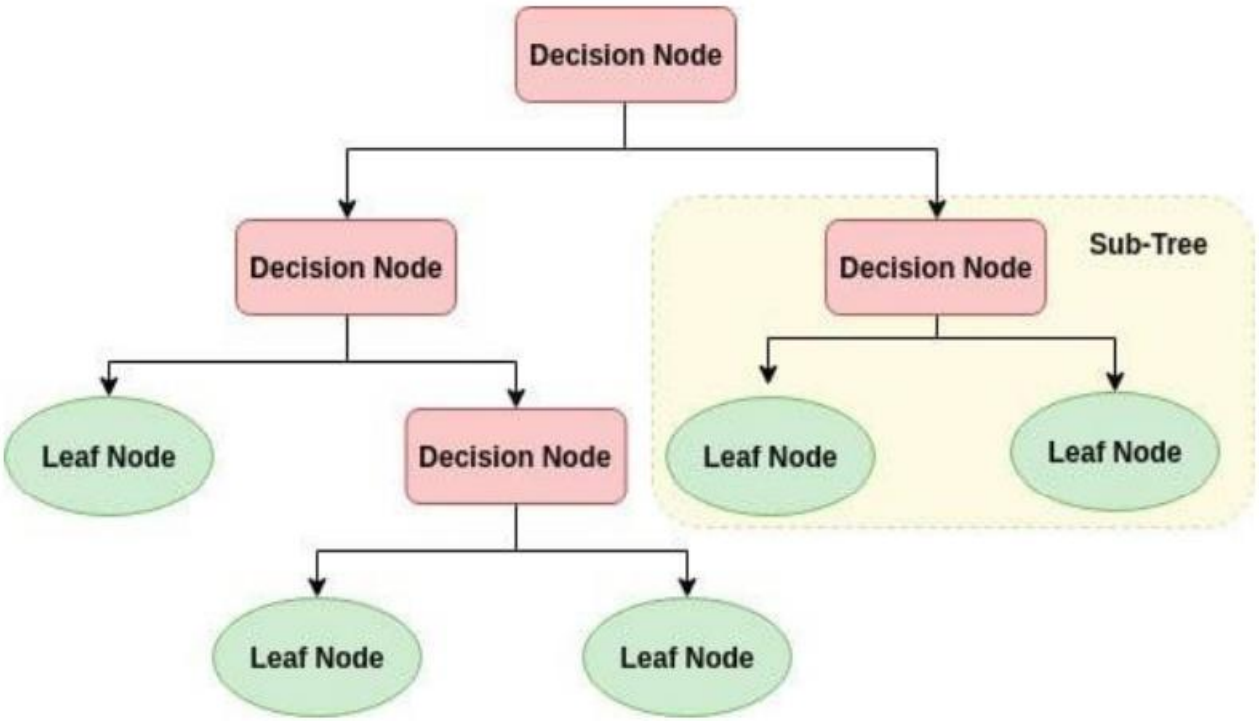
	Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
0	Base Model-Linear Regression with All Variables	0.806997	0.806877	5877.4892	5863.9995
1	Linear Regression After Removing Multicollinea...	0.740859	0.740741	6810.4795	6804.0691
2	Linear Regression with Backward Feature Selection	0.806997	0.806877	5877.4892	5863.9995
3	Lasso regression	0.806268	0.806140	5888.5843	5877.1789
4	Lasso regression Tuned	0.806995	0.806868	5877.5130	5864.1020
5	Ridge regression	0.806986	0.806859	5877.6605	5864.7515
6	Ridge regression Tuned	0.806997	0.806870	5877.4892	5863.9739

So, we can see with linear regressions, and regularization techniques we are not able to solve overfitting and assumptions were failing so we try to **fit Non linear models.**

NON-LINEAR MODELS

DECISION TREE

A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.



Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

```
from sklearn.tree import DecisionTreeRegressor
```

```
dt=DecisionTreeRegressor(random_state=10)
dt.fit(X_train,Y_train)
```

```
DecisionTreeRegressor(random_state=10)
```

+ Code + Te

```
update_score_card1(algorithm_name='Decision Tree with all variables', model=dt)
```

```
score_card1
```

	Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
0	Decision Tree with all variables	0.999985	0.999985	51.8964	3943.6301

```
dt1=DecisionTreeRegressor(random_state=10)

params = {'max_depth': sp_randint(2,10),
      'min_samples_leaf': sp_randint(10,50),
      'min_samples_split':sp_randint(10,20)
     }

rsearch1 = RandomizedSearchCV(estimator = dt1, param_distributions = params, cv = 3, scoring ='neg_root_mean_squared_error')

rsearch1.fit(X_train, Y_train)
```

```
rsearch1.best_params_
```

```
{'max_depth': 8, 'min_samples_leaf': 38, 'min_samples_split': 18}
```

```
dt1 = DecisionTreeRegressor(**rsearch1.best_params_)
dt1.fit(X_train, Y_train)
```

```
DecisionTreeRegressor(max_depth=8, min_samples_leaf=38, min_samples_split=18)
```

```
update_score_card1(algorithm_name='Decision Tree Tuned', model=dt1)
```

```
score_card1
```

	Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
0	Decision Tree with all variables	0.999985	0.999985	51.8964	3943.6301
1	Decision Tree Tuned	0.856662	0.856567	5065.1345	5089.9070

RANDOM FOREST:

Random Forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

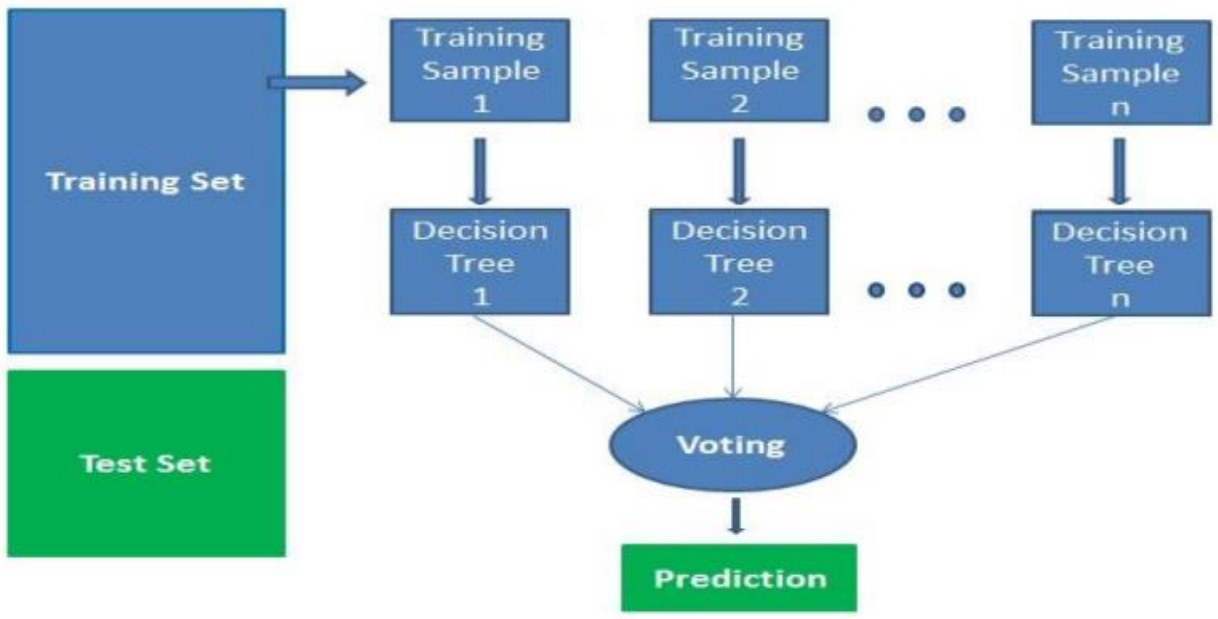
Working of a Random Forest It works in four steps:

1. Select random samples from a given dataset.

2. Construct a decision tree for each sample and get a prediction result from each decision tree.

3. Perform a vote for each predicted result.

4. Select the prediction result with the most votes as the final prediction.



```
from sklearn.ensemble import RandomForestRegressor
```

```
rf=RandomForestRegressor(random_state=10)
rf.fit(X_train,Y_train)
```

```
RandomForestRegressor(random_state=10)
```

```
update_score_card1(algorithm_name='Random Forest with all variables', model=rf)
```

```
score_card1.head()
```

```
rf1=RandomForestRegressor(random_state=10)

params = {'n_estimators':range(10,500,10),
      'max_depth': range(2,10),
      'min_samples_leaf': range(10,50),
      'min_samples_split':range(10,20)
     }

rsearch = RandomizedSearchCV(estimator = rf1, param_distributions = params, cv = 3, scoring ='neg_root_mean_squared_error')

rsearch.fit(X_train, Y_train)
```

```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(random_state=10),
                   param_distributions={'max_depth': range(2, 10),
                                        'min_samples_leaf': range(10, 50),
                                        'min_samples_split': range(10, 20),
                                        'n_estimators': range(10, 500, 10)},
                   scoring='neg_root_mean_squared_error')
```

```
rsearch.best_params_
```

```
{'n_estimators': 480,
 'min_samples_split': 14,
 'min_samples_leaf': 36,
 'max_depth': 9}
```

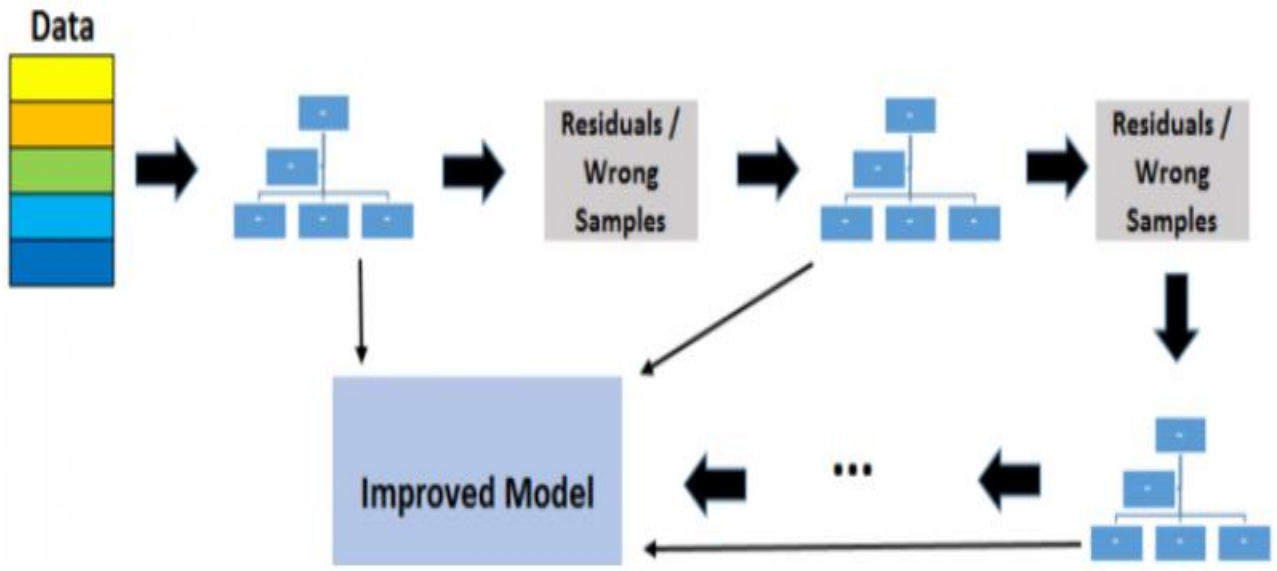
	Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
1	Random Forest with all variables	0.993761	0.993757	1056.7687	2769.3675
2	tuned random forest	0.891797	0.891726	4400.7776	4429.1253

GRADIENT BOOSTING:

In gradient boosting decision trees, we combine many weak learners to come up with one strong learner.

The weak learners here are the individual decision trees. All the trees are connected in series and each tree tries to minimize the error of the previous tree. Due to this sequential connection, boosting algorithms are usually slow to learn (controllable by the developer using the learning rate parameter), but also highly accurate.

In statistical learning, models that learn slowly perform better. The weak learners are fit in such a way that each new learner fits into the residuals of the previous step so as the model improves. The final model adds up the result of each step and thus a stronger learner is eventually achieved.



A loss function is used to detect the residuals.

For instance, mean squared error (MSE) can be used for a regression task .It is worth noting that existing trees in the model do not change when a new tree is added.

The added decision tree fits the residuals from the current model.

```
from sklearn.ensemble import GradientBoostingRegressor
```

```
gb=GradientBoostingRegressor(random_state=10)
gb.fit(X_train,Y_train)
```

```
GradientBoostingRegressor(random_state=10)
```

```
update_score_card1(algorithm_name='Gradient Boosting with all variables', model=gb)
```

```
gb1 = GradientBoostingRegressor(random_state=10)
params = {'n_estimators':range(10,200,10),
          'max_depth':range(2,15),
          'learning_rate':[0.0001,0.001,0.01,0.1,0.1,0.2,0.3],
          'min_samples_split':range(10,40),
          'min_samples_leaf':range(10,20)
         }
rsearch2 = RandomizedSearchCV(estimator = gb1, param_distributions = params, cv = 3, scoring = 'neg_root_mean_squared_error')
rsearch2.fit(X_train,Y_train)
```

```
rsearch2.best_params_
```

```
{'learning_rate': 0.1,
 'max_depth': 13,
 'min_samples_leaf': 19,
 'min_samples_split': 25,
 'n_estimators': 80}
```

```
gb2 =GradientBoostingRegressor(**rsearch2.best_params_)
gb2.fit(X_train, Y_train)
```

```
GradientBoostingRegressor(max_depth=13, min_samples_leaf=19,
                          min_samples_split=25, n_estimators=80)
```

```
update_score_card1(algorithm_name='gradient boosting Tuned', model=gb2)
```

Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
Gradient Boosting with all variables	0.882857	0.882780	4578.9748	4558.2562
gradient boosting Tuned	0.975443	0.975427	2096.5239	2718.037

XGBOOSTING (EXTREME GRADIENT BOOSTING):

XGBoost is an implementation of Gradient Boosted decision trees

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results.

The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model.

It can work on regression, classification, ranking, and user-defined prediction problems.

Thus, boosting algorithms merge various weak learners to get one strong learner. The weak learners generally do not perform well on whole data but are efficient of subset of data.

That's why when we merge them together, each model increases in efficiency and give better predictions as whole

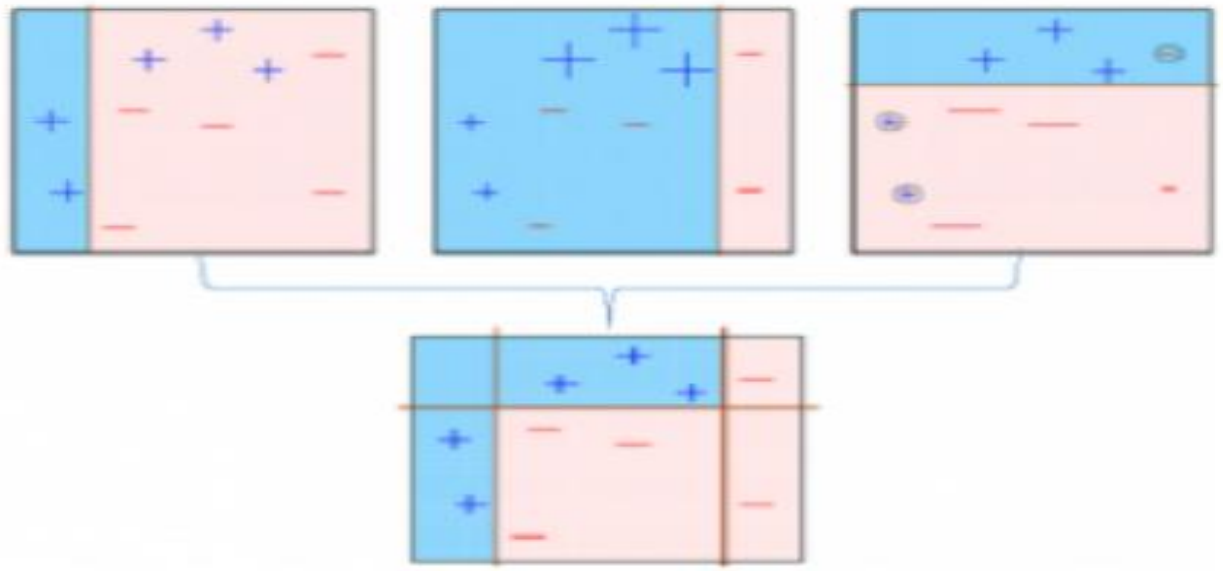
Procedure:

Various subset is created using original dataset.

1. At first all the point is given equal weight and a first model is trained using the subset of data and now this model is used to make the prediction on the actual dataset.

2. Actual data values with predicted value are used to calculate the error in prediction. The incorrectly predicted data points are given higher weight and the second model is trained on new dataset with changed weights. The second model is also trying to reduce the error made by the previous dataset.

3. Sequentially, various models are trained and each model is minimizing the errors from the last model. The final model will be a strong learner is which is the weighted mean of all the weak learners



```
#Xgboost
from xgboost import XGBRegressor
```

```
xgb=XGBRegressor(random_state=10)
xgb.fit(X_train,Y_train)
```

```
[12:59:43] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor(random_state=10)
```

```
xgb1 = XGBRegressor(random_state=10)
params = {'n_estimators':range(10,200,10),
          'max_depth':range(2,15),
          'learning_rate':[0.0001,0.001,0.01,0.1,0.1,0.2,0.3],
          'min_samples_split':range(10,40),
          'min_samples_leaf':range(10,20)
         }
rsearch4 = RandomizedSearchCV(estimator = xgb1, param_distributions = params, cv = 3, scoring = 'neg_root_mean_squared_error')
rsearch4.fit(X_train,Y_train)
```

```
rsearch4.best_params_
```

```
{'learning_rate': 0.2,
 'max_depth': 12,
 'min_samples_leaf': 12,
 'min_samples_split': 12,
 'n_estimators': 180}
```

```
xgb2 =XGBRegressor(**rsearch4.best_params_)
xgb2.fit(X_train, Y_train)
```

```
[20:42:53] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor(learning_rate=0.2, max_depth=12, min_samples_leaf=12,
             min_samples_split=12, n_estimators=180)
```

S.No	Model Description	R2	Adj R2	Train RMSE	Test RMSE
7	XGBoost with all variables	0.883	0.882	4583.241	4564.19
8	XGBoost Tuned	0.991	0.991	1233.264	2659.869

LIGHTGBM:

Light GBM is an ensemble method that combines decision trees (as weak learners) in a serial fashion (boosting).

What makes the LightGBM more efficient?

The starting point for LightGBM was the histogram-based algorithm since it performs better than the pre-sorted algorithm.

For each feature, all the data instances are scanned to find the best split with regards to the information gain. Thus, the complexity of the histogram-based algorithm is dominated by the number of data instances and features.

To overcome this issue, LightGBM uses two techniques

- GOSS (Gradient One-Side Sampling)
- EFB (Exclusive Feature Bundling)

GOSS (Gradient One-Side Sampling)

GOSS provides a way to sample data based on gradients while taking the data distribution into consideration.

Here is how it works:

- The data instances are sorted according to the absolute value of their gradients
- Top ax100% instances are selected
- From the remaining instances, a random sample of size bx100% is selected
- The random sample of small gradients is multiplied by a constant equal to (1-a) / b when the information gain is calculated

What GOSS eventually achieves is that the focus of the model leans towards the data instances that cause more loss (i.e., under-trained) while not affecting the data distribution much.

EFB (Exclusive Feature Bundling)

Datasets with a high number of features are likely to have sparse features (i.e. lots of zero values). These sparse features are usually mutually exclusive which means they do not have non-zero values simultaneously. Consider the case of one-hot encoded text data. In a particular row, only one column indicating a specific word is non-zero and all other rows are zero.

EFB is a technique that uses a greedy algorithm to combine (or bundle) these mutually exclusive into a single feature (e.g., exclusive feature bundle) and thus reduce the dimensionality. EFB reduces the training time of GDBT without affecting the accuracy much because the complexity of creating feature histograms is now proportional to the number of bundles instead of the number of features (#bundles is much less than #features).

```
from lightgbm import LGBMRegressor
```

```
lgbm=LGBMRegressor(random_state=10)
lgbm.fit(X_train,Y_train)
```

```
LGBMRegressor(random_state=10)
```

```
update_score_card1(algorithm_name='Light Gradient Boosting with all variables', model=lgbm)
```

```
lgb1 = LGBMRegressor(random_state=10)
params = {'num_iterations':range(100,500,10),
          'max_depth':range(10,30,2),
          'learning_rate':[0.0001,0.001,0.01,0.1,0.1,0.2,0.3],
          'num_leaves' :range(10,50,2)
         }
rsearch3 = RandomizedSearchCV(estimator = lgb1, param_distributions = params, cv = 3, scoring = 'neg_root_mean_squared_error')
rsearch3.fit(X_train,Y_train)
```

```
rsearch3.best_params_
```

```
{'learning_rate': 0.1,
 'max_depth': 14,
 'num_iterations': 480,
 'num_leaves': 46}
```

```
lgb2 = LGBMRegressor(**rsearch3.best_params_)
lgb2.fit(X_train, Y_train)
```

```
LGBMRegressor(max_depth=14, num_iterations=480, num_leaves=46)
```

```
update_score_card1(algorithm_name='Light Gradient Boosting with Tuning', model=lgb2)
```

Model_Name	R-Squared	Adj. R-Squared	Train_RMSE	Test_RMSE
Light Gradient Boosting with all variables	0.936938	0.936896	3359.6560	3394.9341
Light Gradient Boosting with Tuning	0.962696	0.962671	2583.9704	2799.7582

COMPARISION OF MODELS:

The following are the set of linear models that we tried:

Linear Models

S.No	Model Description	R2	Adj R2	Train RMSE	Test RMSE
1	Base Model-Linear Regression with all variables	0.807	0.807	5877.489	5863.999
2	Linear Regression removing multicollinearity in numerical variables	0.74	0.74	6810.4795	6804.069
3	Linear Regression with backward feature selection	0.807	0.807	5877.489	5863.999
4	Lasso Regression	0.806	0.806	5888.584	5877.1789
5	Lasso Tuned	0.806	0.806	5877.67	5864.7515
6	Ridge Regression	0.806	0.806	4877.489	5863.973
7	Ridge Tuned	0.806	0.806	5877.498	5863.973

The following are the set of non-linear models that we tried:

Non Linear Models

S.No	Model Description	R2	Adj R2	Train RMSE	Test RMSE
1	Decision tree with all variables	0.999	0.999	51.896	3943.69
2	Decision tree tuned	0.856	0.856	5065.134	5089.907
3	Random Forest with all variables	0.994	0.994	1056.768	2769.367
4	Random Forest Tuned	0.891	0.891	4400.778	4429.125
5	Gradient Boosting with all variables	0.883	0.883	4578.975	4558.256
6	Gradient Boosting Tuned	0.975	0.975	2096.524	2718.037
7	XGBoost with all variables	0.883	0.882	4583.241	4564.19
8	XGBoost Tuned	0.991	0.991	1233.264	2659.869

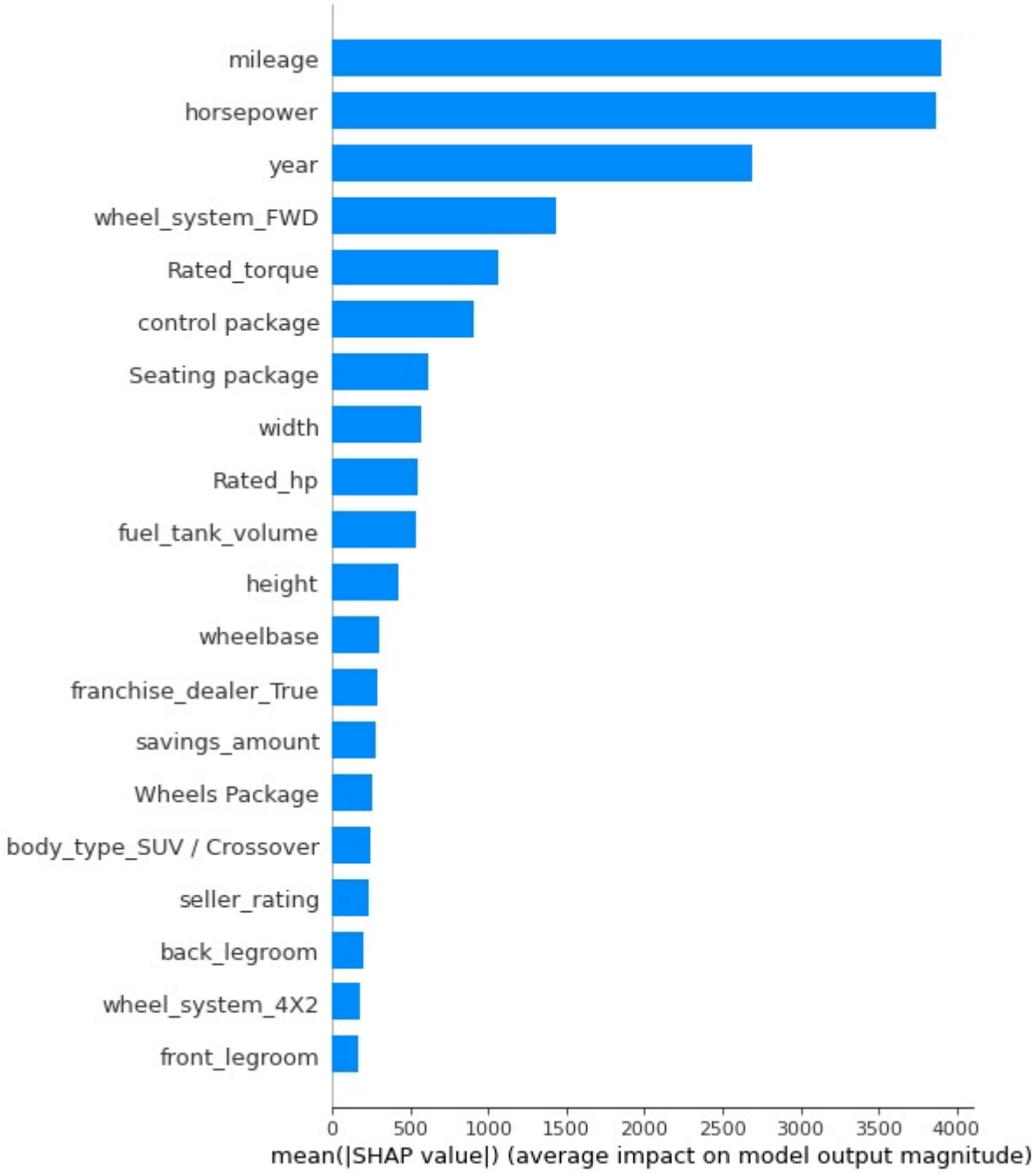
We have observed that **Random Forest** after tuning is the best model. We have arrived at the decision based on the following:

1.There is less gap between the R2 and Adjusted R2 which shows that most of the variables are significant.

2.There is less gap between train and test RMSE which means the model is not Ovefitting or Underfitting.

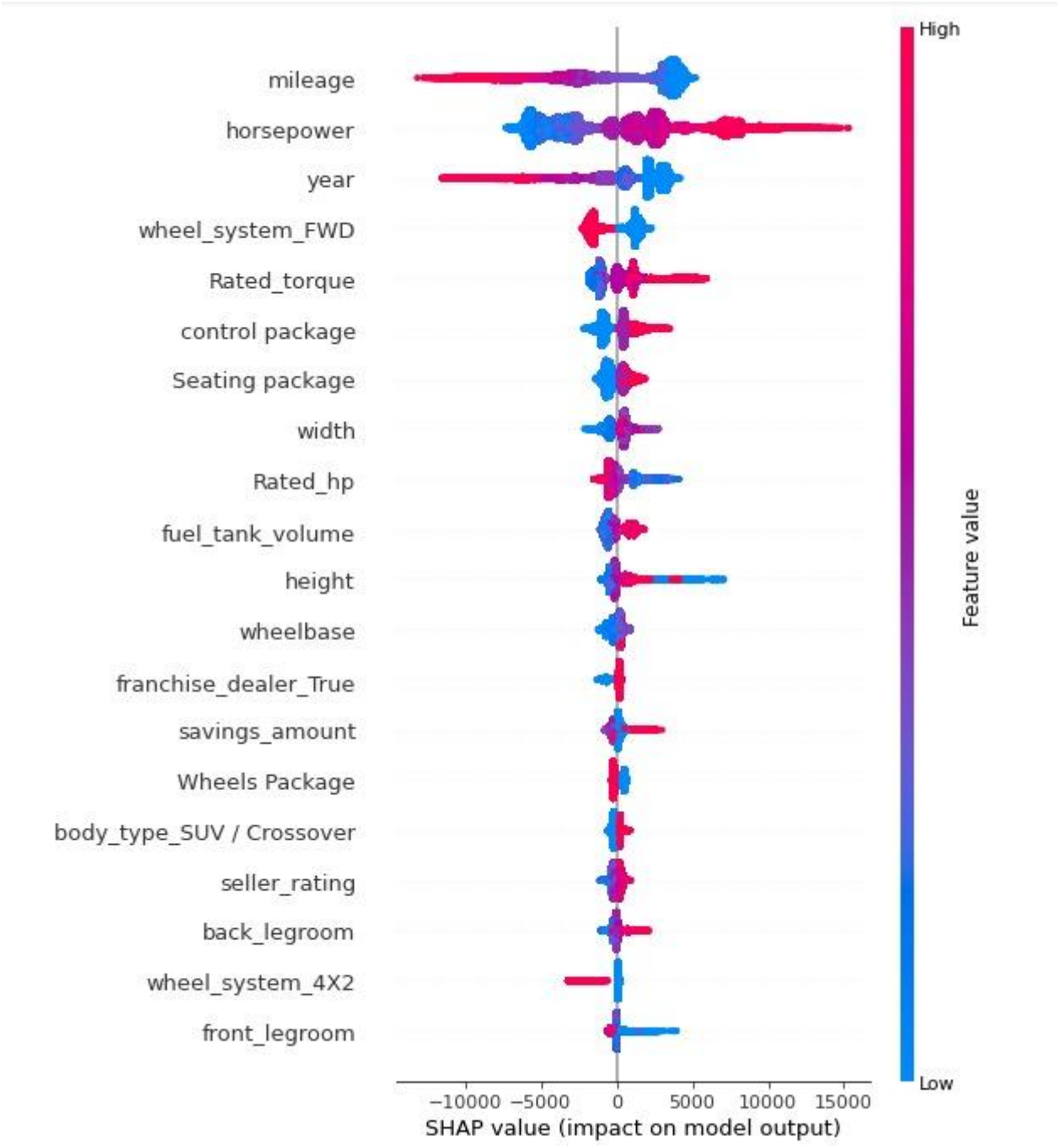
FEATURE IMPORTANCE OF BEST MODEL:

The below picture represents the feature importance's from Random Forest.



The below picture represents the feature importance's from the library SHAP.

```
explainer = shap.TreeExplainer(gb)

shap_values = explainer.shap_values(X_train)
features=X_train.columns
shap.summary_plot(shap_values, X_train, feature_names=features, plot_type="bar")
shap.summary_plot(shap_values, X_train, feature_names=features)
```



SHAP:

SHAP (SHapley Additive explanations) is a game-theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions.

Business interpretation of top 5 features:

1.Mileage:

Mileage, as the name suggests, is the total number of miles your car has driven. Essentially, it's an indicator of how well-used your car is. A car with more mileage has basically worked more, so this may result in more servicing being needed than on a car that hasn't covered as much ground.

Two cars might be completely identical in make, model, and age, but mileage can still be the factor that makes one car worth more than the other. A low mileage car will be more likely to get a better price than a car that has a higher-than-average mileage, but this is by no means guaranteed.

High mileage also reflects the sort of wear and tear a dealer or buyer will expect from a car. A low mileage car, for example, will likely have brake pads that are still in good condition. Tire condition, unless you have recently changed them, will also be affected.

2.Year of manufacturing:

This is probably the most important factor in a resale. Greater the mileage, more the car has been used; hence higher wear and tear and impact on the engine. A lower reading on the odometer will always command a higher price.

Ever hear the familiar expression that the minute you drive a new car off the lot, it drops in value? No lie. A car can lose 20% or more of its original value within the first year.

A car that's aged on paper, has aged according to the norms of country's automotive authorities, which would make it unfit for plying on road, regardless of it's odometer reading. Therefore, a car that has been used sparingly and has only say about 2-3 years of life left, is actually nearing towards being scrapped.

Unless government comes up with measures that can practically access each car's condition and provide fitness certificates accordingly, cars that have aged on paper will continue to get scrapped regardless of their actual condition. It's one of the biggest factors that affect the valuation price.

3.Torque:

Torque, like horsepower, is responsible for the speed of your car. Torque is related to the amount of twisting force something can push out. So when you are trying to twist open a sticky doorknob, the force needed to twist and open the door refers to the amount of torque being used.

Now when it comes to how fast your car can go, that's where the horsepower takes over. Horsepower is why certain cars with turbocharged V8 or better engines can hit top speeds of 300 mph. The force

produced in the engine by the horsepower at relatively high rpm is what's going to allow you to reach high speeds.

So, which is better for you and your racing dreams? Well, if you just want to go fast and hit 140 mph, then horsepower would be more effective for you. However, if you want a strong car that can pull boulders and take off quickly, a high torque might be more important to you. In short, torque makes your vehicle quick. Horsepower makes it fast.

4.Horsepower-

Horsepower is the maximum power available to a vehicle. In theory, the higher the number, the more power is sent to the wheels and the faster it will go. This is why you see some sports cars with astonishingly high horsepower figures.

Power figures can be calculated in different ways. Typically, the horsepower or "hp" figure quoted is the maximum available to that vehicle. Sometimes, though, the figure may be calculated at different engine revs, which gives us brake horsepower.

However, the amount of acceleration you get depends on many factors, including the weight of the car and its torque. This means that the highest horsepower vehicle might not always be the fastest.

Torque is how hard can the engine turn. It's that simple how much turning force can the engine generate. A measure of force.

5.Wheel_System_FWD-

In the model we can see that FWD drivetrain is the most preferred wheel system in used cars and ones with so will fetch a higher price what is essentially means for a customer is. If you have a vehicle with a FWD drivetrain, this means the engine delivers power to the front wheels of the vehicle. Essentially, the front wheels are doing all the work while the rear wheels do not receive any power. This is one of the more common drivetrains you will see.

The pros of a FWD vehicle are that they typically get better fuel economy and emits less carbon dioxide. Since the weight of the engine is located over the driving wheels, a FWD vehicle can maintain better traction in the snow. However, performance enthusiasts have claimed FWD vehicles are less fun to drive.

Since all the weight is located in the front of the vehicle, front-wheel drive cars tend to understeer. During sudden acceleration, front-wheel drive vehicles tend to veer to the right or left because of something called "torque steer." Front-wheel drive tends to have a lower towing capacity than rear-wheel or 4WD/AWD drivetrains. Front-wheel drive has worse acceleration than rear-wheel drive, which is why most sporty and race cars use rear-wheel drive. With all the weight up front, front-wheel drive can make handling more difficult. CV joints/boots in FWD vehicles tend to wear out sooner than rear-wheel drive vehicles.

RECOMMENDATIONS AND INTERPRETATION:

1. We have seen there are lot of technical features which are good predictors of price.
2. We have also seen that add-ons also contribute to the price such as add-ons from the group control and seating packages. This tells us that while predicting the price of the used car, additional packages also contribute to price.
3. Older the car, definitely reduces the price of the car.
4. Mileage is an important factor in predicting the price of the car. More the distance travelled would cause the car to have some kind of wear and tear. So, mileage is an import factor in predicting the car price.
5. We see that wheel system FWD is a major contributor to price. Front wheel drive is more suitable in bad weather, affordable and more mileage. So, it is more preferred.
6. Many factors other than the make and model of the car, are good predictors of the car.
7. We can clearly see the difference between what companies like Car Guru, Autotrader, CarMax are requesting for the price prediction and what all features actually effect the price point of the car, many features like control add on, seating add on, drive system, Rated torque and rated Horsepower which are never asked by these companies in the price prediction actually effect the final cost of a used car.
8. In the future, more data will be collected using different web-scraping techniques, and deep learning classifiers will be tested. Algorithms like Quantile Regression, ANN and SVM will be tested.
9. Afterwards, the intelligent model will be integrated with web and mobile-based applications for public use. Moreover, after the data collection phase Semiconductor shortages have incurred after the pandemic which led to an increase in car prices, and greatly affected the secondhand market. Hence having a regular Data collection and analysis is required periodically, ideally, we would be having a real time processing program