
# Used Cars Price Prediction and Data Evaluation using Data Mining Techniques

Submitted by:

**Rakesh Chaudhary**

# ACKNOWLEDGMENT

I would like to extend my thanks and appreciation to FilbRobo company, my internship company, for their continuous support and guidance during the capstone project, I can also never forget the efforts of all the seniors and colleagues that taught me during the Data Analytics program and guided me though the world of data, which was a new realm for me. Moreover, I would like to extend my gratitude to Data Trained Education for his guidance and patience with us during the whole journey.

## Resources

The data used in this project was scrapped from different online seller sites like Olx.com, Droom.com, Cars24.in, Cardekho.com…etc.

# INTRODUCTION

## Business Problem Framing

In this paper, we 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 updated online sites. The Pre-owned cars or so-called used cars have capacious market across the globe. Before acquiring a used car, the buyer should be able to decide whether the price affixed for the car is genuine. Several facts including mileage, year, model, model_make_year, run and many more are needed to be considered before getting a hold of any pre-owned car. Both the seller and the buyer should have a fair deal.

Keep above in mind we can say that used car price prediction is a topic of high interest. Because of the affordability of used cars in developing countries, people tend more purchase used cars. A primary objective of this project is to estimate used car prices by using attributes that are highly correlated with a label (Price).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.

# Conceptual Background of the Domain Problem

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. Accurate car price prediction involves expert knowledge, because price usually depends on many distinctive features and factors. Typically, most significant ones are brand and model, age, horsepower and mileage. The fuel type used in the car as well as fuel consumption per mile highly affect price of a car due to a frequent changes in the price of a fuel. Different features like exterior colour, door number, type of transmission, dimensions, safety, air condition, interior, whether it has navigation or not, will also influence the car price. In this paper, we applied different methods and techniques in order to achieve higher precision of the used car price prediction.

The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models. We will compare the performance of various machine learning algorithms like Linear Regression, Ridge Regression, Lasso Regression, Elastic Net, Decision Tree Regressor and choose the best out of it. Depending on various parameters we will determine the price of the car. Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been developed which acquires input from any user and displays the Price of a car according to user's inputs.

# Review of Literature

First of all we arepredicting the price of Used Car Using Machine Learning Techniques. Here, we investigate the application of supervised machine learning techniques to predict the price of used cars which are available in online sites like cars24, cardekho.com and many more. The predictions are based on historical data collected from daily uploaded cars in online sites. Different techniques like multiple linear regression analysis, k-nearest neighbours, Ensemble techniques and decision trees have been used to make the predictions.

After model building, we will evaluate the predicted price of models in second hand car system based on higher accuracy and least error. During this, the price evaluation model based on big data analysis is proposed, which takes advantage of widely circulated vehicle data and a large number of vehicle transaction data to analyse the price data for each type of vehicles by using the optimized techniques like MSE, MAE. It aims to establish a second-hand car price evaluation model to get the price that best matches the car.

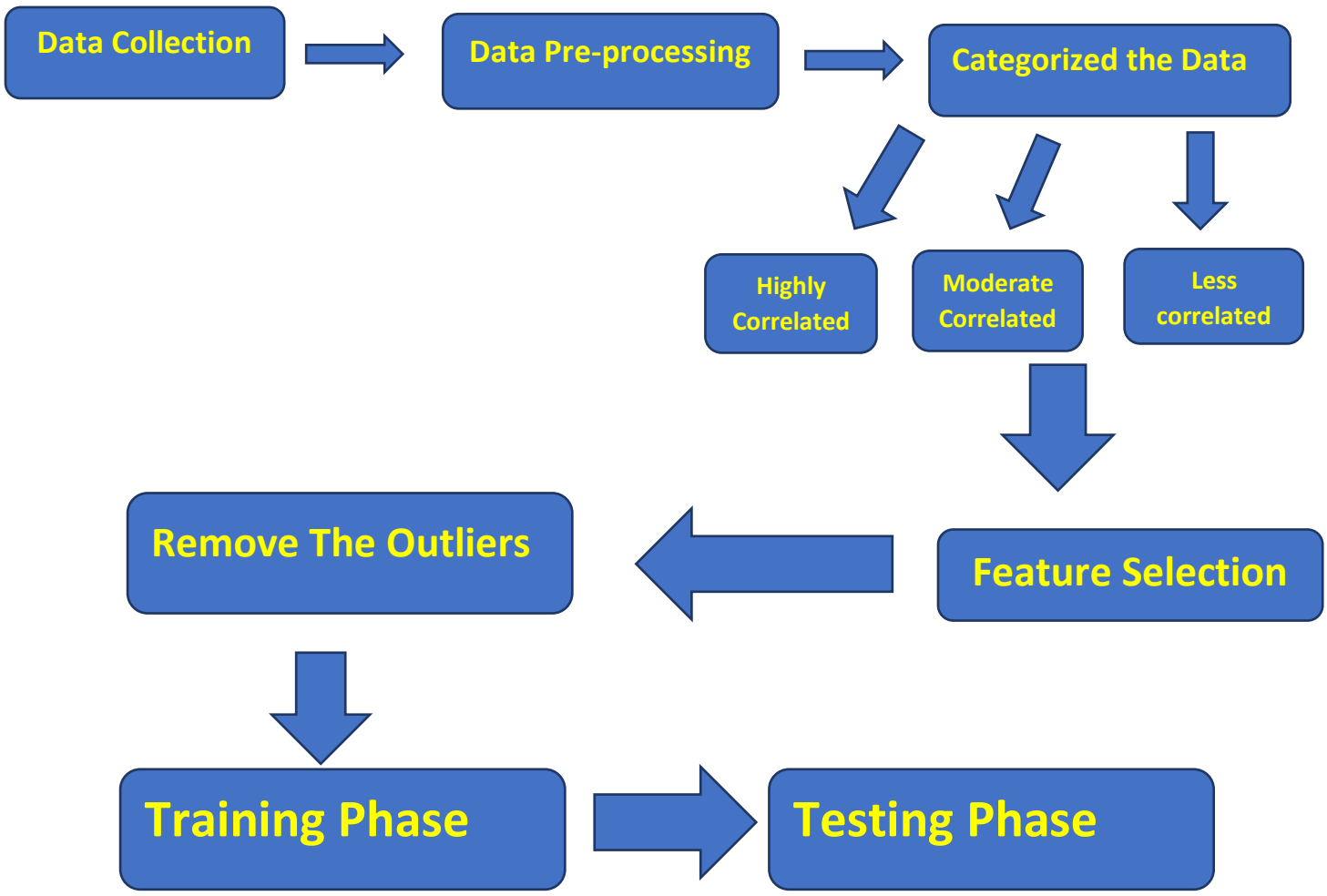
## Motivation for the Problem Undertaken

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

# Analytical Problem Framing

## Mathematical/ Analytical Modelling of the Problem:-

Approach for car price prediction proposed in this paper is composed of several steps, shown in Figure below..!



There are two primary phases in the system:

1. Training phase: The system is trained by using the data in the data set and fits a model (line/curve) based on the algorithm chosen accordingly.

2. Testing phase: the system is provided with the inputs and is tested for its working. The accuracy is checked. And therefore, the data that is used to train the model or test it, has to be appropriate. The system is designed to detect and predict price of used car and hence appropriate algorithms must be used to do the two different tasks. Before the algorithms are selected for further use, different algorithms were compared for its accuracy. The well-suited one for the task was chosen.

Since manual data collection is time consuming task, especially when there are numerous records to process, a "web scraper" as a part of this research is created to get this job done automatically and reduce the time for data gathering. Web scraping is well known technique to extract information from websites and save data into local file or database. Manual data extraction is time consuming and therefore web scrapers are used to do this job in a fraction of time. Web scrapers are programed for specific websites and can mimic regular users from website's point of view.

After raw data has been collected and stored to local database, data pre-processing step was applied. Many of the attributes were sparse and they do notcontain useful information for prediction. Hence, it is decided to remove them from the dataset. Like attributes "state", "city", and "damaged" were completely removed.

## Data Sources and their formats

Data collection is the most prime step for any project. We have designed the system for used cars in different region like Mumbai, Bangalore, Hyderabad, New Delhi, for which the data of used cars is collected using as on 1998 to 2021. Total more than 6500 records was scrapped using the selenium package. The data is fetched composed of null records as well. The feature captured including Model year, seller type, Driven in Kilometres, Number of owners, Engine capacity, Max Power, Mileage of the vehicle, seat capacity and sell price. Sell price is the dependent variable.

We used ensemble machine learning techniques to implement the system.

Ensemble technique build multiple models and at the last blend them. Therefore the accuracy increased than a single model would. Hence, Ensemble techniques are Supervised machine learning methods. Suppose you want to find a job but you are confused that how to find the suitable job? What is the process? What is the first step? How to start? These type of some question appear in your mind first. So you can one thing, you can ask your family, friends, neighbours for that. This task of accumulating feedback will give you a honest and accurate decisions. Then last you can here conclude that a group of miscellaneous people make better. Decisions as compared to an individual and the same is true if rather than using a single model, we use a group of diverse models. To achieve diversification in machine learnings, we have ensemble techniques. Bagging, boosting and voting are famous ensemble methods.

## Data Pre-processing Done:-

Pre-processing is a Data Mining technique that involves converting raw data into a comprehensible format. There is often a lack of specific activity or trend data, and many inaccurate facts are included in real-world data. Consequently, this may result in poor-quality data collection, and, in turn, poor-quality models constructed from the data. Such problems can be resolved by pre-processing the data.

Pre-processing in Machine Learning is the process of modifying, or encoding, data so that the machine can parse it more easily. Thus, the algorithm can now properly interpret the data.

In this project, following steps are preformed to pre-process the dataset:-

1. **Importing the data set:-**

```
import warnings
warnings.filterwarnings('ignore')

#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

```
## importing the dataset:
df=pd.read_excel("Final_Used_cars_data.xlsx")
df.head()
```

| | Model_year | Brand | KMs_driven | Fuel_type | Transmission_type | Seller | No_of_owners | Mileage | Engine | Max_power | Torque | Seats | Price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2012 | Rolls-Royce | 36500 | Petrol | Automatic | Dealer | Second Owner | 10.20 | 6592.0 | 563.00 | 820.0 | 4.0 | 18500000 |
| **1** | 2015 | Mercedes-Benz | 60000 | Diesel | Automatic | Dealer | First Owner | 13.00 | 2143.0 | 201.10 | 500.0 | 5.0 | 1775000 |
| **2** | 2021 | Audi | 4500 | Petrol | Automatic | Dealer | First Owner | 9.80 | 2995.0 | 335.25 | 500.0 | 5.0 | 10700000 |
| **3** | 2016 | Porsche | 50000 | Diesel | Automatic | Dealer | First Owner | 16.12 | 2967.0 | 245.00 | 550.0 | 5.0 | 6250000 |
| **4** | 2021 | Porsche | 600 | Petrol | Automatic | Dealer | First Owner | NaN | 2997.0 | 350.00 | 480.0 | 5.0 | 8900000 |

We have collected all the cars data from different sites like olx.com, cars24.in, cardekho.com etc and combine all the data in one sheet. We will use this data in this project and predict the price of used cars by our Machine Learning models.

```
## checking shape of our dataset:
df.shape
```

```
(6711, 13)
```

We have 6711 rows and 13 columns present in my dataset including target variable.

2. **Checking data type:-**

Those attributes values are in the varchar format and need to be converted into integer or float data type.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6711 entries, 0 to 6710
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Model_year         6711 non-null   int64  
 1   Brand              6711 non-null   object 
 2   KMs_driven         6711 non-null   int64  
 3   Fuel_type          6711 non-null   object 
 4   Transmission_type  6711 non-null   object 
 5   Seller             6711 non-null   object 
 6   No_of_owners       6711 non-null   object 
 7   Mileage            6618 non-null   float64
 8   Engine             5706 non-null   float64
 9   Max_power          6446 non-null   float64
 10  Torque             5420 non-null   float64
 11  Seats              6708 non-null   float64
 12  Price              6711 non-null   int64  
dtypes: float64(5), int64(3), object(5)
memory usage: 681.7+ KB
```

There are 3 features are object datatype and 6 are numeric data type. In our case we don't need to convert any feature into integer or categorical.

3. **Removing Duplicates:-**

   Duplicates are an extreme case of non-random sampling, and they bias your fitted model. Including them will essentially lead to the model overfitting this subset of points. So we have to remove them.

```
## checking duplicates:
df.duplicated().sum()
```

```
1652
```

There are lot of duplicate data present in my dataset. So we have to delete these duplicate values otherwise our data set could become biased. So we are going to delete all the duplicate from the dataset.

4. **Checking outliers:-**

```
### Mileage
df.Mileage=df.Mileage.fillna(df.Mileage.mean())
df.isna().sum()
```

```
Model_year              0
Brand                   0
KMs_driven              0
Fuel_type               0
Transmission_type       0
Seller                  0
No_of_owners            0
Mileage                 0
Engine               1002
Max_power             243
Torque               1268
Seats                   0
Price                   0
dtype: int64
```

Seems there are many null values present in our dataset.

**Filling Missing values:-**

```
### Torque, Engine, Max_power
df.Torque=df.Torque.fillna(df.Torque.mean())
df.Max_power=df.Max_power.fillna(df.Max_power.mean())
df.Engine=df.Engine.fillna(df.Engine.mean())

df.isna().sum()
```

```
Model_year           0
Brand                0
KMs_driven           0
Fuel_type            0
Transmission_type    0
Seller               0
No_of_owners         0
Mileage              0
Engine               0
Max_power            0
Torque               0
Seats                0
Price                0
dtype: int64
```

We have successfully fill the missing values.

5. **Encoding:-**

# Encoding of Categorical Features:

```
df_new.Model_year=df_new.Model_year.astype(str)
```

```
df=pd.get_dummies(df_new,drop_first=True)
print(df.shape)
df.head(10)
```

```
(4891, 71)
```

| | KMs_driven | Mileage | Engine | Max_power | Torque | Seats | Price | Model_year_1999 | Model_year_2001 | Model_year_2003 | ... | Fuel_type_Petrol | Fuel_type_Petrol + CNG | Fuel_type_F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.275526 | 13.00 | 1.261233 | 1.759229 | 2.146160 | 5.0 | 1775000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 3 | -0.016044 | 16.12 | 2.283761 | 2.143512 | 2.371639 | 5.0 | 6250000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 5 | 0.162555 | 17.11 | 0.978115 | 1.459244 | 1.508250 | 5.0 | 1125000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 6 | -0.723356 | 17.90 | 1.261233 | 0.890079 | 0.972076 | 5.0 | 1898000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 7 | 0.540097 | 18.12 | 1.023845 | 1.642286 | 1.626209 | 5.0 | 1350000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 8 | -0.046998 | 12.07 | 2.283761 | 2.115869 | 2.371639 | 7.0 | 2399000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 9 | -0.765138 | 16.00 | 1.027201 | 0.959817 | 1.117253 | 7.0 | 874000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 10 | -0.046998 | 17.68 | 0.978115 | 1.459244 | 1.508250 | 5.0 | 1550000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 11 | 0.783583 | 14.81 | 1.028877 | 1.320929 | 1.384539 | 5.0 | 550000 | 0 | 0 | 0 | ... | 0 | 0 | |
| 12 | -1.186069 | 16.60 | 1.028877 | 0.399126 | 0.737006 | 5.0 | 2199000 | 0 | 0 | 0 | ... | 0 | 0 | |

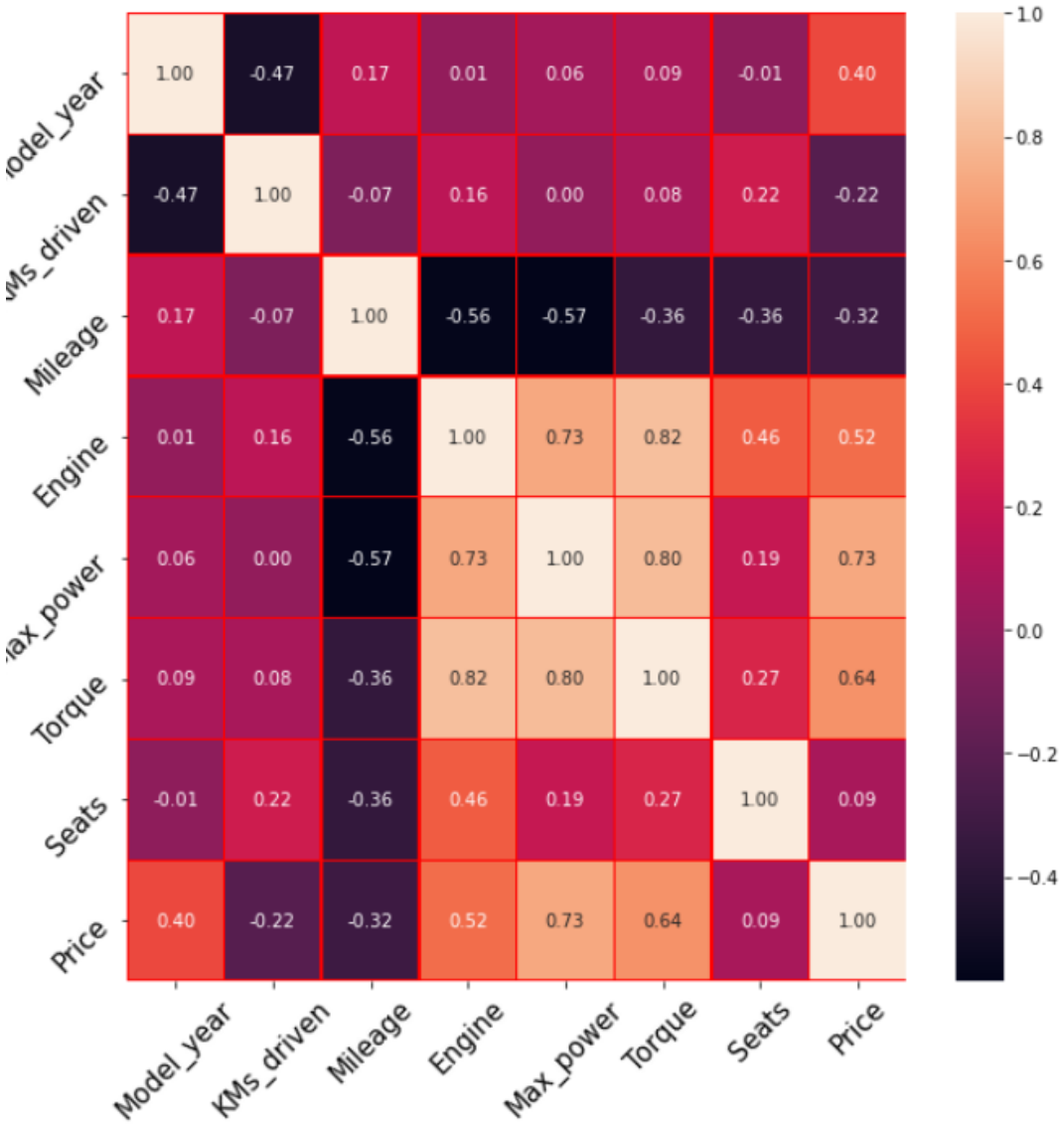
One hot encoding can be defined as the essential process of converting the categorical data variables to be provided to machine and deep learning algorithms which in turn improve predictions as well as classification accuracy of a model.

6. **Checking correlation:-**

**Heatmap:-**

A heat map is a two-dimensional representation of information with the help of colours. Heat maps can help the user visualize simple or complex information. Heat maps are used in many areas such as defence, marketing and understanding consumer behaviour.

After cleaning the data, we can visualize data and better understand the relationships between different variables. There are many more visualizations that you can do to learn more about your dataset, like scatterplots, histograms, boxplots, etc Using **sns.heatmap()**, we can see that the Engine, Max_power' features are positively correlated with 'Selling_*Price*' and 'Mileage' and 'Kms_drien' is negatively correlated with 'Selling_*Price*'.



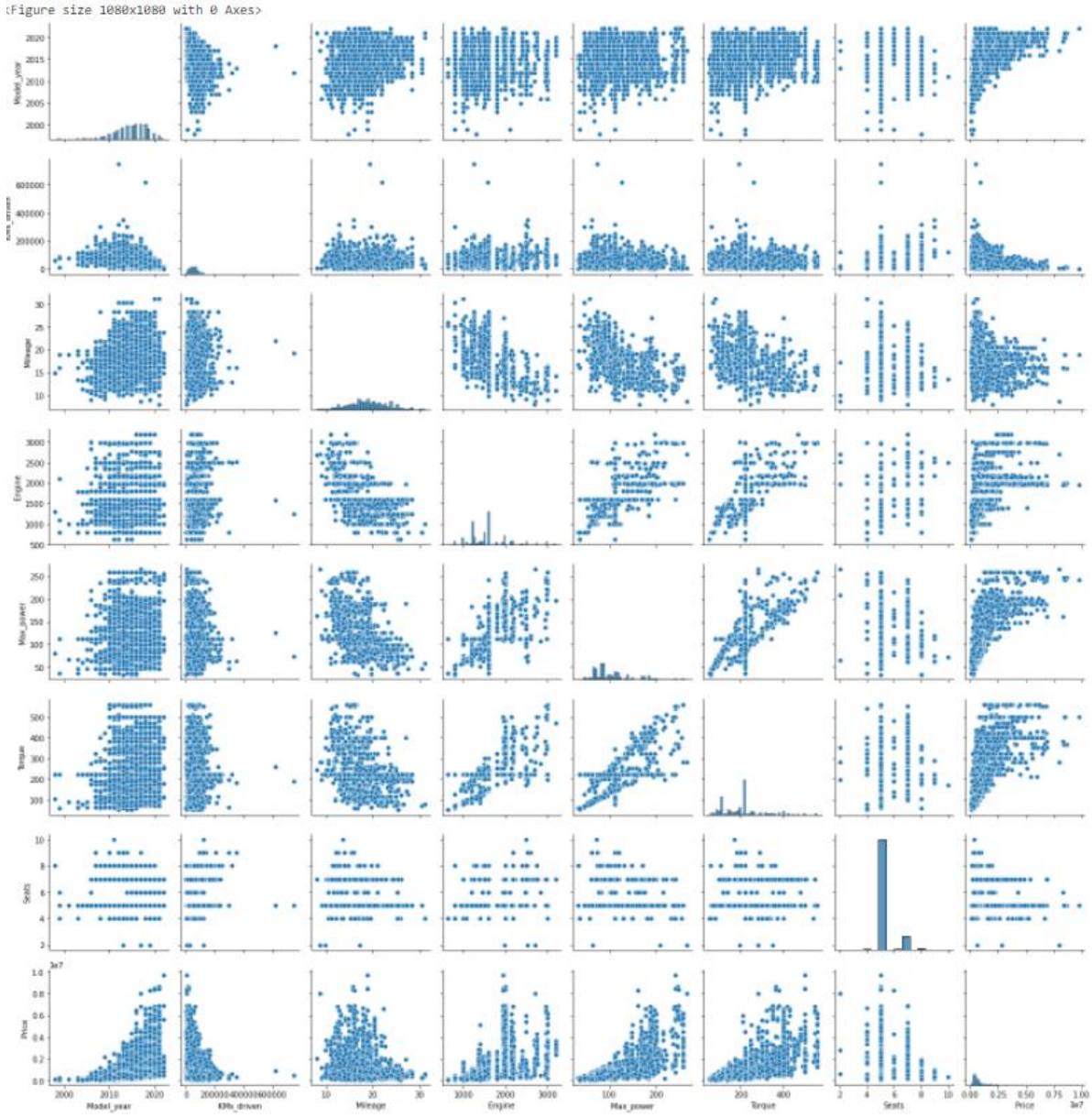

## Observation of Heatmap:

- As we can see that KMs_driven,Mileaage features are showing negative correlation between our target variable Price.
- Some features are showing strong relation with each other. Torque is strongly correlated with Max_power and Engine with 80% and 82% correlation respectively.
- Engine and Max_power is also showing strong relation with each other with 73% positive correlation.

# Data Inputs- Logic- Output Relationships:-

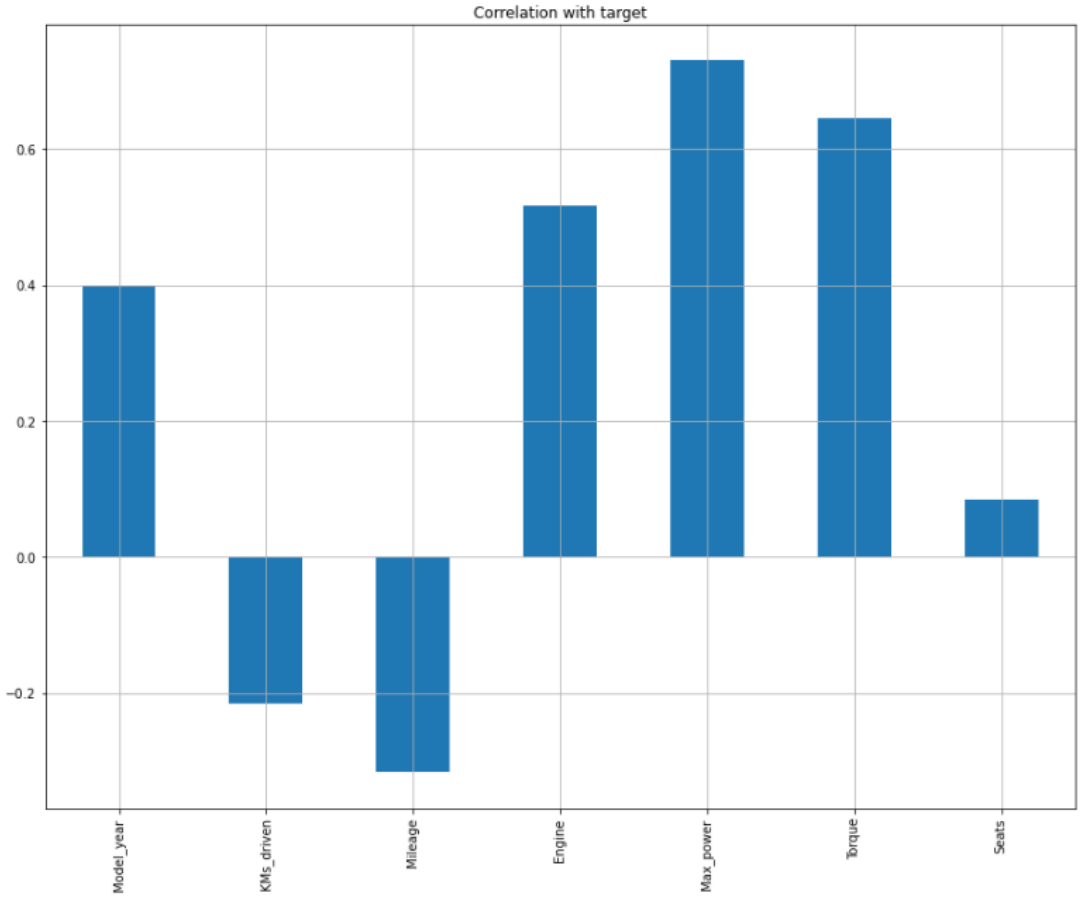
## Features Vs Features:-

Pair Plots are a really simple (one-line-of-code simple!) way to visualize relationships between each variable. It produces a matrix of relationships between each variable in your data for an instant examination of our data. It can also be a great jumping off point for determining types of regression analysis to use.




## Feature vs Target:-

```
df_new.drop('Price',axis=1).corrwith(df_new.Price).plot(kind='bar',grid=True,figsize=(12,10),title='Correlation with target')
plt.tight_layout()
```




# ## Observations:-

- Max_power is highly correlated with target variable.
- Seats feature is least correlated with target variable.

## State the set of assumptions (if any) related to the problem under consideration

While exploring the data, we will look at the different combinations of features with the help of visuals. This will help us to understand our data better and give us some clue about pattern in data.

Data cleaning is one of the processes that increases prediction performance, yet insufficient for the cases of complex data sets as the one in this research.

We assume that if we apply single machine learning algorithm on the dataset we cannot get high accuracy, therefore, the ensemble of multiple machine learning algorithms will be proposed and this combination of ML methods gains quite good accuracy.

## Hardware and Software Requirements and Tools Used

**Hardware requirements**

1. Operating system- Windows 7,8,10
2. Processor- dual core 2.4 GHz (i5 or i7 series Intel processor or equivalent AMD)
3. RAM-4GB Software Requirements
4. Python Pycharm PIP
5. 2.7 Jupyter Notebook Chrome

# Model/s Development and Evaluation

## Identification of possible problem-solving approaches (methods)

Predictive statistical analysis is **a type of statistical analysis that analyzes data to derive past trends and predict future events on the basis of them**. It uses machine learning algorithms, data mining, data modelling, and artificial intelligence to conduct the statistical analysis of data.

Summary of Stats:

```
df.describe()
```

| | Model_year | KMs_driven | Mileage | Engine | Max_power | Torque | Seats | Price |
|---|---|---|---|---|---|---|---|---|
| count | 5056.000000 | 5.056000e+03 | 5003.000000 | 4054.000000 | 4813.000000 | 3788.000000 | 5056.000000 | 5.056000e+03 |
| mean | 2015.234968 | 5.656322e+04 | 18.924186 | 1605.947213 | 112.107479 | 221.707328 | 5.256329 | 1.059848e+06 |
| std | 3.251747 | 4.438191e+04 | 4.121392 | 613.259547 | 56.864231 | 136.471392 | 0.761044 | 1.554005e+06 |
| min | 1998.000000 | 2.380000e+02 | 7.080000 | 624.000000 | 34.200000 | 51.000000 | 2.000000 | 5.000000e+04 |
| 25% | 2013.000000 | 3.124250e+04 | 16.100000 | 1197.000000 | 75.000000 | 113.750000 | 5.000000 | 3.860990e+05 |
| 50% | 2016.000000 | 5.300000e+04 | 18.700000 | 1461.000000 | 88.760000 | 190.000000 | 5.000000 | 5.800000e+05 |
| 75% | 2018.000000 | 7.373800e+04 | 21.750000 | 1968.000000 | 131.400000 | 300.000000 | 5.000000 | 9.750000e+05 |
| max | 2022.000000 | 1.900000e+06 | 34.050000 | 6592.000000 | 626.000000 | 1298.000000 | 10.000000 | 1.850000e+07 |

### Observation:

- There are some missing values present in our dataset.
- We have from 1998 to 2022 i.e. 20 years cars vehicle data present.
- The maximum 10 seats available in some cars.
- Mileage is quite well distributed.

## Testing of Identified Approaches (Algorithms)

1. **Linear Regression:-**Linear Regression was chosen as the first model due to its simplicity and comparatively small training time. The features, without any feature mapping, were used directly as the feature vectors. No regularization was used since the results clearly showed low variance.
2. **Random Forest Random:-** Forest is an ensemble learning based regression model. It uses a model called decision tree, specifically as the name suggests, multiple decision trees to generate the ensemble model which collectively produces a prediction. The benefit of this model is that the trees are produced in parallel and are relatively uncorrelated, thus producing good results as each tree is not prone to individual errors of other trees. This uncorrelated behaviours is partly ensured by the use of Bootstrap Aggregation or bagging providing the randomness required to produce robust and uncorrelated trees. This model was hence chosen to account for the large number of features in the dataset and compare a bagging technique with the following gradient boosting methods.
3. **Gradient Boost**:-Gradient Boosting is another decision tree based method that is generally described as "a method of transforming weak learners into strong learners". This means that like a typical boosting method, observations are assigned different weights and based on certain metrics, the weights of difficult to predict observations are increased and then fed into another tree to be trained.
4. **Extreme Gradient Boosting or XGBoost**:- XGB is one of the most popular machine learning models in current times. XGBoost is quite similar at the core to the original gradient boosting algorithm but features many additive features that significantly improve its performance such as built in support for regularization, parallel processing as well as giving additional hyper parameters to tune such as tree pruning, sub sampling and number of decision trees.
5. **KNN Regressor:-**KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighbourhood.

## Run and Evaluate selected models

First of all we have to import all necessary libraries:-

```
from sklearn.model_selection import train_test_split as tts
```

```
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
### importing ensemble  models

from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
```

**Code:-**We are a defining a function for finding the best random state. We will use Linear Regression model for find the best random state. Code are given in below snapshots:-

## Define a Function for Best Random State:

We will use LinearRegression for finding the best random state.

```
def random_state(feature,target):
    max_r2=0
    for i in range(1,101):
        x_train,x_test,y_train,y_test=tts(feature,target,test_size=0.20,random_state=i)
        lr=LinearRegression()
        lr.fit(x_train,y_train)
        pred=lr.predict(x_test)
        score=r2_score(y_test,pred)
        if score>max_r2:
            max_r2=score

    return i
```

```
## Creating a list having all models:
all_model=[LinearRegression(),KNeighborsRegressor(),RandomForestRegressor(), AdaBoostRegressor(),
           GradientBoostingRegressor(),DecisionTreeRegressor(),SVR(),XGBRegressor(),Lasso(),Ridge()]
model_name=['linear regression','k-nearest neighbors','random forest','adaboost','gradientboosting','decisiontree',
            'svr','xgb','lasso','ridge']

### importing cross validation
from sklearn.model_selection import cross_val_score
```

## Define a Function for Model_Building;

```
def model_building(Models,Features,Target,n):
    x_train,x_test,y_train,y_test=tts(Features,Target,test_size=0.2,random_state=random_state(Features,Target))
    ## creating the empty list
    number=[]
    Mae=[]
    Mse=[]
    Smse=[]
    r2=[]
    MeanCV=[]
    ## building the model
    for i , j in enumerate(Models):
        j.fit(x_train,y_train)
        pred_test=j.predict(x_test)
        ## choosing cv=5



        score=cross_val_score(j,x_train,y_train,cv=5,scoring='r2')

        # appending the score with their respective list
        number.append(n[i])
        Mae.append(np.round(MAE(y_test,pred_test),3))
        Mse.append(np.round(MSE(y_test,pred_test),3))
        Smse.append(np.round(np.sqrt(MSE(y_test,pred_test)),3))
        r2.append(np.round(r2_score(y_test,pred_test),2))
        MeanCV.append(np.round(np.mean(score),4))

    ## make a dataframe to understand in a better way
    dataframe=pd.DataFrame()
    dataframe['Model_name']=number
    dataframe['Mean Absolute error']=Mae
    dataframe['Mean Squared error']=Mse
    dataframe['SquareRoot of Mean Squared error']=Smse
    dataframe["Model's R2 Score "]=r2
    dataframe['Mean of the Cross Validation']=MeanCV
    dataframe.set_index('Model_name',inplace=True)
    return dataframe
```

We are using many Machine learning algorithm and compare the accuracy of all of them.

```
model_building(all_model,x_comp,y,model_name)
```

| Model_name | Mean Absolute error | Mean Squared error | SquareRoot of Mean Squared error | Model's R2 Score | Mean of the Cross Validation |
|---|---|---|---|---|---|
| linear regression | 322267.871 | 2.703676e+11 | 519968.847 | 0.67 | -346.9296 |
| k-nearest neighbors | 178373.975 | 1.383642e+11 | 371973.448 | 0.83 | 0.8119 |
| random forest | 142864.737 | 6.996318e+10 | 264505.545 | 0.91 | 0.8819 |
| adaboost | 517177.570 | 3.593281e+11 | 599439.827 | 0.56 | 0.6741 |
| gradientboosting | 167297.096 | 8.428487e+10 | 290318.565 | 0.90 | 0.8751 |
| decisiontree | 205586.746 | 2.167556e+11 | 465570.161 | 0.73 | 0.7486 |
| svr | 480174.059 | 8.879491e+11 | 942310.524 | -0.09 | -0.1100 |
| xgb | 138915.385 | 6.682157e+10 | 258498.677 | 0.92 | 0.8826 |
| lasso | 322267.142 | 2.703666e+11 | 519967.843 | 0.67 | -42.0072 |
| ridge | 322244.135 | 2.703397e+11 | 519942.062 | 0.67 | -40.8227 |

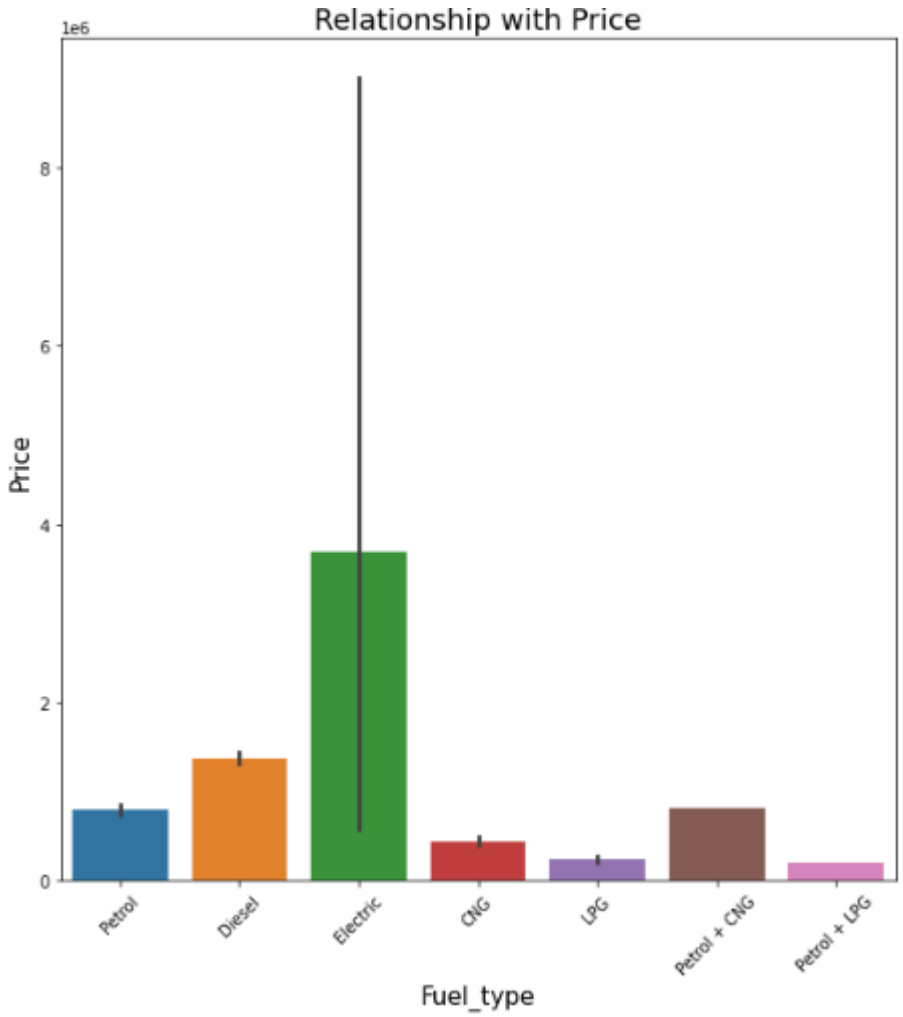
As we can see that Random Random forest and XGB give us the best accuracy.

## Key Metrics for success in solving problem under consideration

- **Mean Square Error(MSE):-**MSE is the single value that provides information about goodness of regression line.
- **Root Mean Square Error (RMSE):**RMSE is the quadratic scoring rule that also measures the average magnitude of the error. It is the square root of average squared difference between prediction and actual observation.
- **Mean Absolute Error (MAE):**This measure represents the average absolute difference between the actual and predicted values in the dataset. It represents the average residual from the dataset.

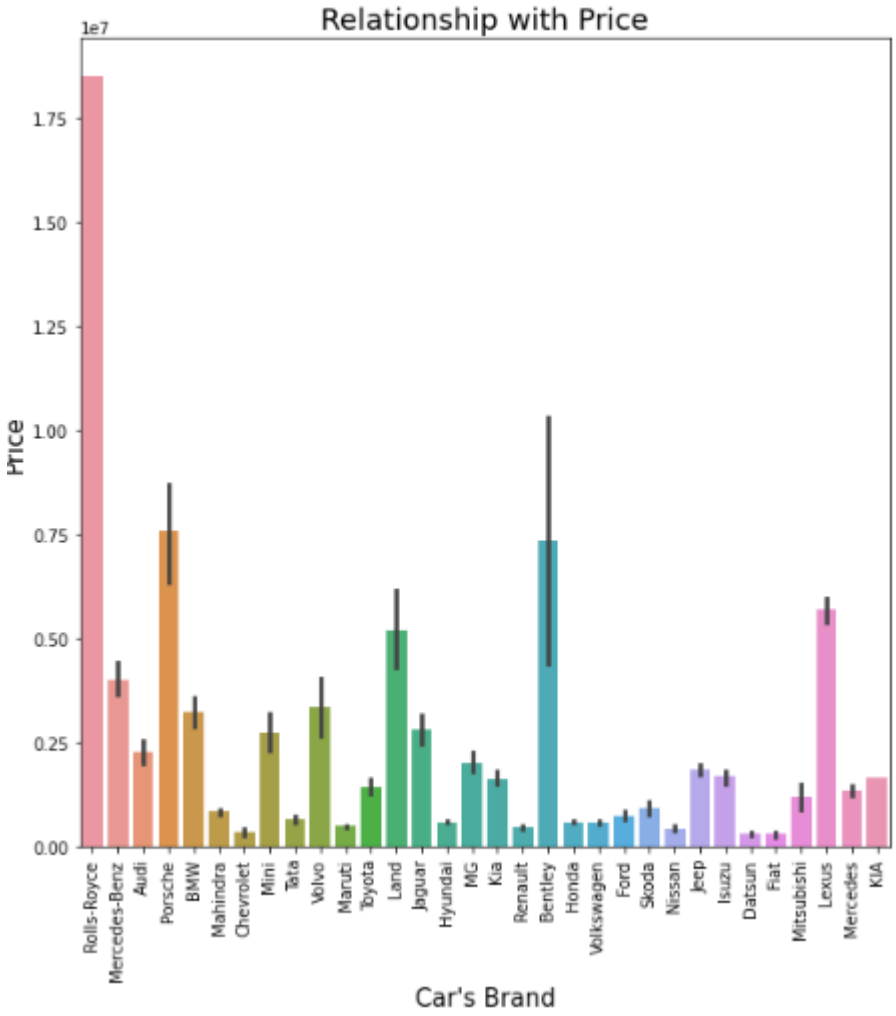
## Visualizations

```
Fuel_type Vs Price
```



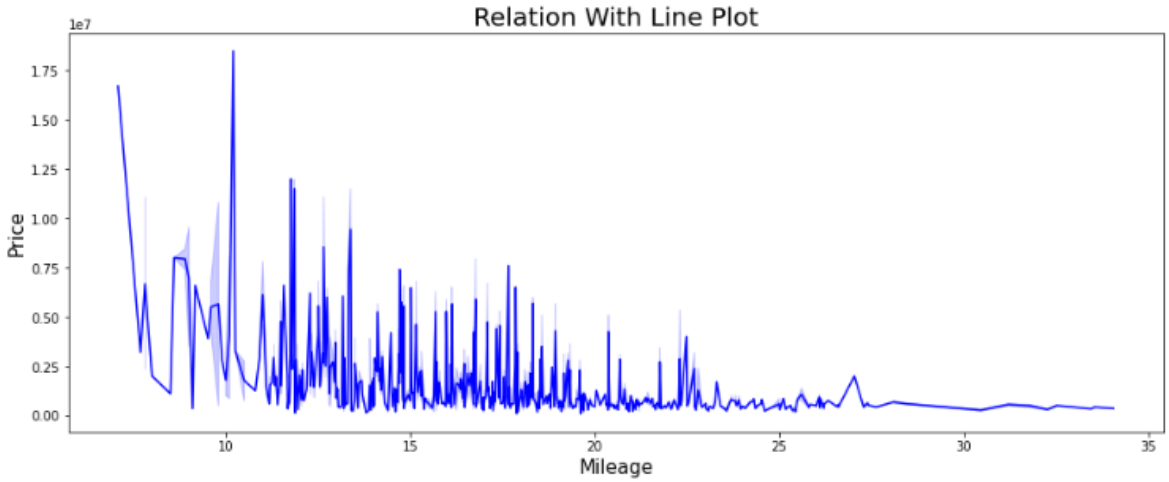

- Electric car's price is height with others.
- Electric cars are very less in the market right now, they are marked by advanced technology, so it could be reason to get it more costly.

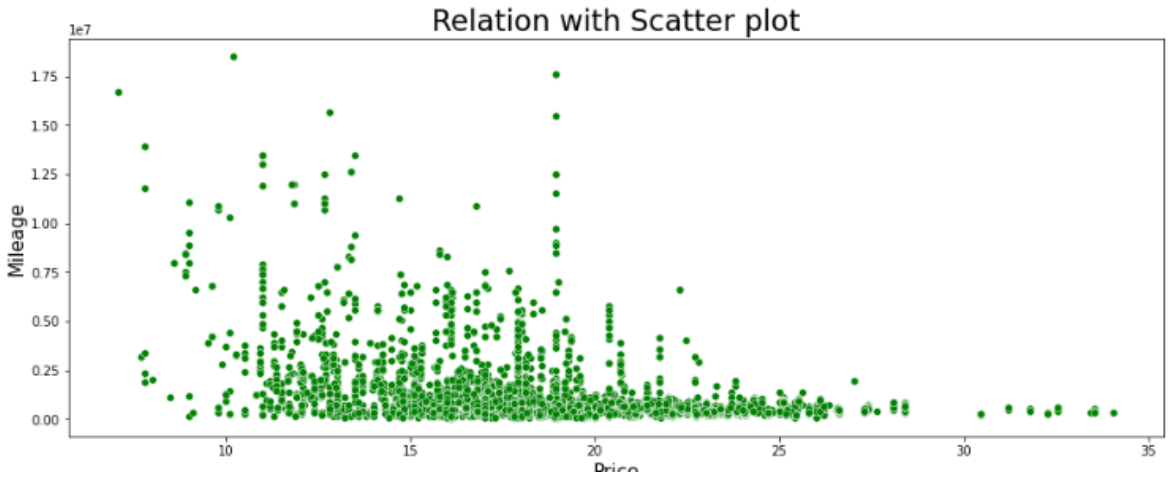
```
Brand Vs Price
```




- As we all know Rolls-Royce make world's most luxury cars and their price also going very high.
- Similarly Porsche, BMW, Land, Bentley, Lexus, Mercedes Brands are known for make Luxury and expensive cars. That's why we can see their higher price in above graph.

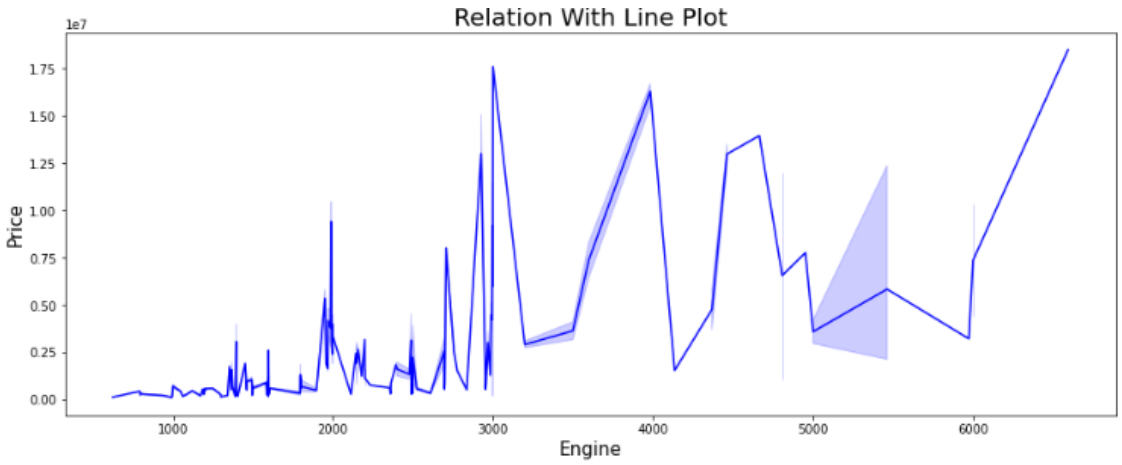
## Mileage Vs Price

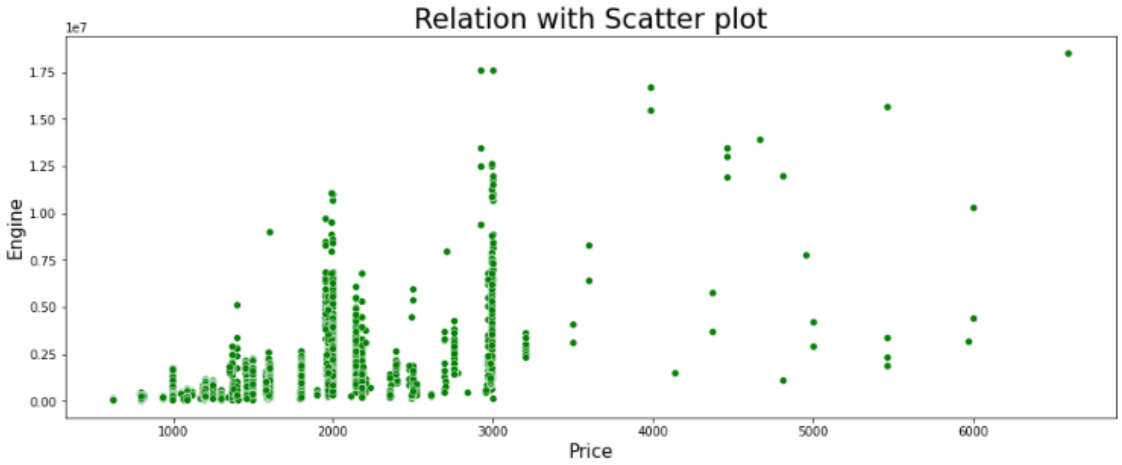



- We can see the clear Negative relation between Mileage and Price.
- As Mileage is high the price of the vehicle is low, but as mileage is decreasing, the price of the car is also increasing.
- It is obvious that expensive cars have heavy engine that's why their mileage is getting down.

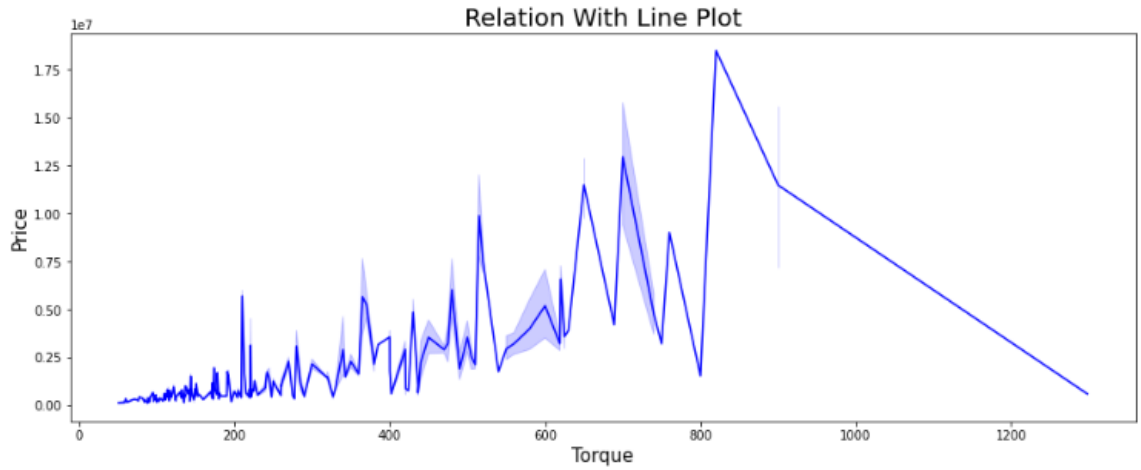
## Engine Vs Price




- We can see there is positive relation between Engine and Price.
- As unit of Engine is increasing the Price of the vehicle is also increasing.
- Expensive cars having big capacity Engine.

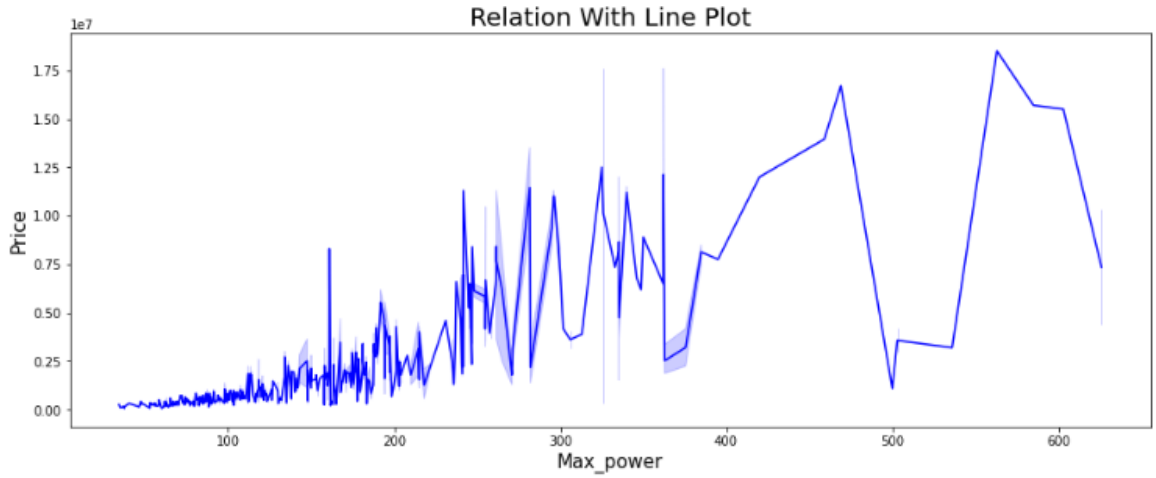
```
Torque Vs Price
```





- Here we can see that the strong relation between price and Torque.
- As torque increasing and price is also increasing.
- We can see that there is also sharp decreasing at the last. We could see that there is a only one data point available. It could be an outlier of wrong entry. We will handle it in upcoming stages.

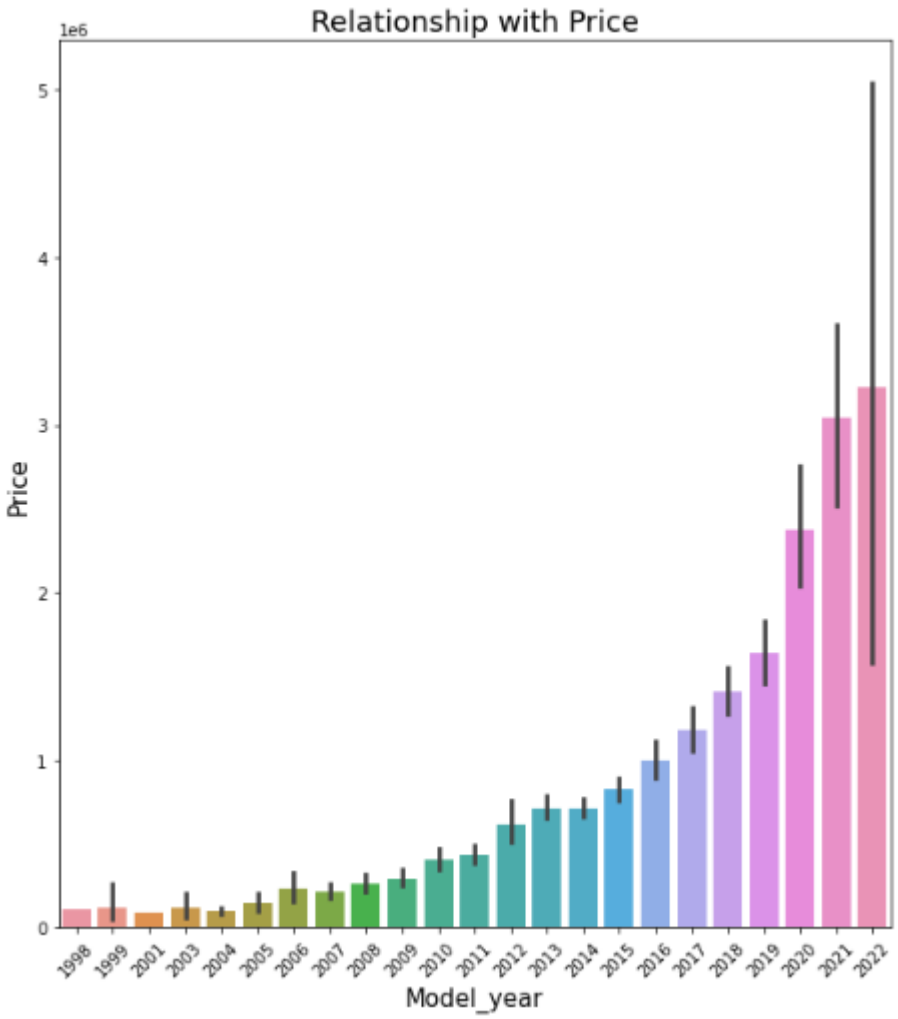
```
Max_power_ Price
```





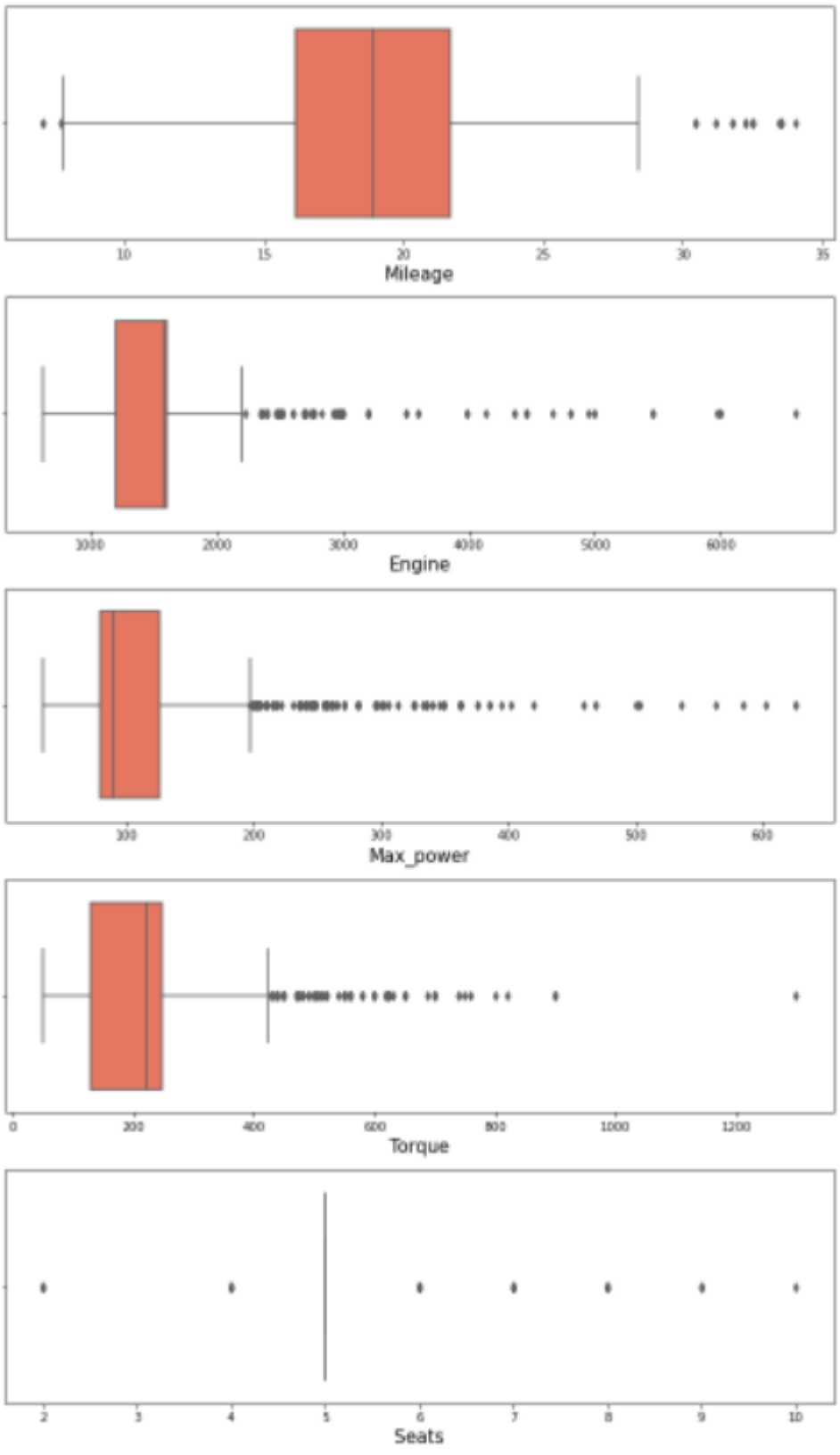
- In the starting of the price the Max_power showing a good positive relation with Max_power. But as price increasing that max_power sometimes increasing and sometime decreasing.
- So we can say that there is strong relation in the beginning but at the last point their relation getting destroying.

```
Model_Year Vs Price
```




- We observe that as time getting modern the number of new inventions are going high. That's their price is also going high.

## Checking Outliers:



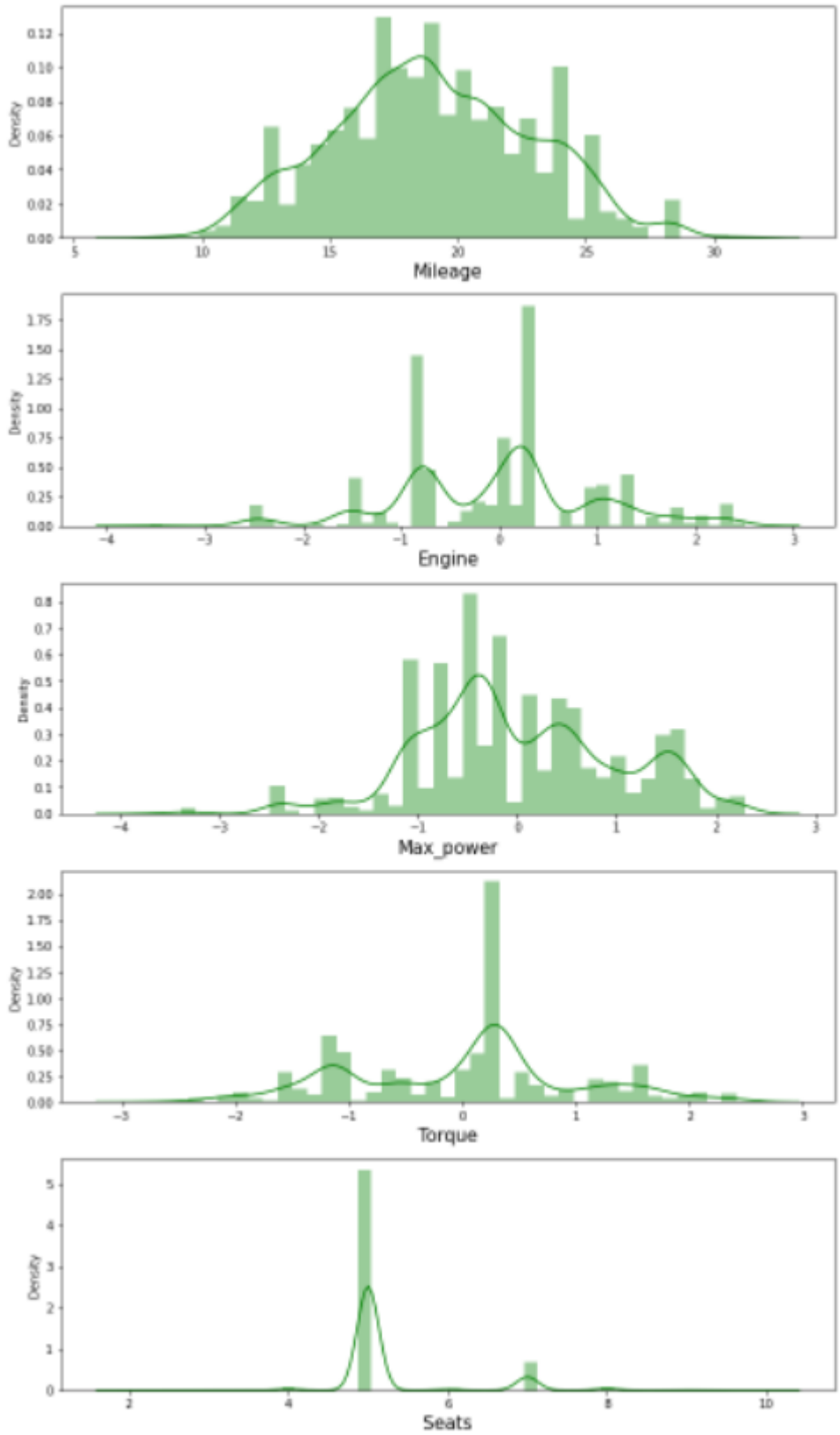

- Max_power has heights amount of outliers present.
- Mileage has least number of outliers present.

- Seats feature is a nominal data type, so we will not remove outliers from it.

## Skewness of Data:

```
df_new.skew()
```

```
Model_year   -0.578756
KMs_driven    3.351230
Mileage       0.107049
Engine        1.116215
Max_power     1.228935
Torque        1.053708
Seats         2.216769
Price         3.333466
dtype: float64
```




- Now our dataset showing fine distributed.

# Interpretation of the Results

We have used PCA before model building. PCA technique is particularly useful in processing data where multi-collinearity exists between the features/variables. PCA can be used when the dimensions of the input features are high (e.g. a lot of variables). PCA can be also used for demonising and data compression.

## Handling Multicolinearity with PCA

When we are in a confusion that which one feature we should keep and which one we should drop. Then comes PCA and it will take care of multi-collinearity problem.

First scaling the data.

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

```
## splitting  target and feature
x=df.drop('Price',axis=1)
y=df['Price']
x.isna().sum().any()
```

```
False
```

```
### scaling
for i in x.columns:
    x[i]=scaler.fit_transform(x[i].values.reshape(-1,1))
x.head()
```

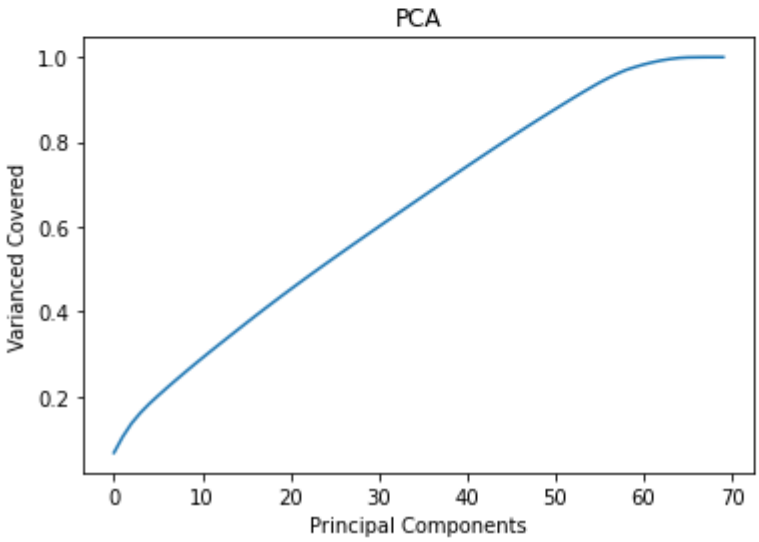
| | KMs_driven | Mileage | Engine | Max_power | Torque | Seats | Model_year_1999 | Model_year_2001 | Model_year_2003 | Model_year_2004 | ... | Fuel_type_Petrol | Fuel_type_Petro + CNG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.275526 | -1.552971 | 1.261233 | 1.759229 | 2.146160 | -0.352969 | -0.024774 | -0.0143 | -0.028609 | -0.035046 | ... | -1.077189 | -0.014 |
| 3 | -0.016044 | -0.747310 | 2.283761 | 2.143512 | 2.371639 | -0.352969 | -0.024774 | -0.0143 | -0.028609 | -0.035046 | ... | -1.077189 | -0.014 |
| 5 | 0.162555 | -0.491668 | 0.978115 | 1.459244 | 1.508250 | -0.352969 | -0.024774 | -0.0143 | -0.028609 | -0.035046 | ... | -1.077189 | -0.014 |
| 6 | -0.723356 | -0.287671 | 1.261233 | 0.890079 | 0.972076 | -0.352969 | -0.024774 | -0.0143 | -0.028609 | -0.035046 | ... | -1.077189 | -0.014 |
| 7 | 0.540097 | -0.230861 | 1.023845 | 1.642286 | 1.626209 | -0.352969 | -0.024774 | -0.0143 | -0.028609 | -0.035046 | ... | -1.077189 | -0.014 |

5 rows × 70 columns

```
## Apply PCA
from sklearn.decomposition  import PCA
pca=PCA()
pca.fit_transform(x)
```

```
array([[ 4.19764320e+00, -1.54349824e+00, -2.58144308e-02, ...,
         6.51441104e-02, -1.09683454e-03, -4.33162295e-04],
       [ 4.60499563e+00, -2.04070495e+00, -2.36820442e-02, ...,
        -8.38583377e-03,  7.52125089e-06, -2.90513617e-03],
       [ 2.89336603e+00, -5.55352128e-01, -6.49182017e-01, ...,
        -7.77592096e-02,  6.12623785e-05,  8.53305578e-04],
       ...,
       [ 3.20646898e+00,  4.44604190e+00,  2.09408079e+00, ...,
        -5.13543178e-02, -2.46464228e-03,  7.09259195e-05],
       [-4.93405825e-01, -4.16616823e-02,  2.55289034e+00, ...,
        -1.03963633e-02, -6.57842974e-03,  4.23728101e-03],
       [-1.14583199e+00,  2.23301318e+00,  2.77008478e+00, ...,
         7.69879043e-02, -1.49925732e-02, -2.57774170e-03]])
```

```
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Principal Components')
plt.ylabel('Varianced Covered')
plt.title('PCA')
plt.show()
```




- We can see that only 62 component is enough for telling about our dataset more then 95%.
- so we will selected n_components=62

```
pca=PCA(n_components=62)
new_comp=pca.fit_transform(x)
x_comp=pd.DataFrame(new_comp,columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10',
                                      'PC11','PC12','PC13','PC14','PC15','PC16','PC17','PC18','PC19','PC20',
                                      'PC21','PC22','PC23','PC24','PC25','PC26','PC27','PC28','PC29','PC30',
                                      'PC31','PC32','PC33','PC34','PC35','PC36','PC37','PC38','PC39','PC40',
                                      'PC41','PC42','PC43','PC44','PC45','PC46','PC47','PC48','PC49','PC50',
                                      'PC51','PC52','PC53','PC54','PC55','PC56','PC57','PC58','PC59','PC60',
                                      'PC61','PC62'])
x_comp.head()
```

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | PC7 | PC8 | PC9 | PC10 | ... | PC53 | PC54 | PC55 | PC56 | PC57 | PC58 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4.197643 | -1.543498 | -0.025814 | -1.628289 | -0.577224 | 1.183407 | 1.095151 | 0.514588 | -0.727729 | -0.325097 | ... | 0.135551 | 0.198416 | -0.351379 | 0.185615 | -0.039563 | 0.726609 |
| 1 | 4.604996 | -2.040705 | -0.023682 | -2.067453 | -1.563406 | 1.092497 | 0.719289 | -1.137922 | 0.604877 | -3.155273 | ... | 0.595771 | -0.017700 | -0.428680 | -0.196266 | 0.056805 | -0.731733 |
| 2 | 2.893366 | -0.555352 | -0.649182 | -0.763898 | -0.758820 | 0.647484 | -0.525125 | 0.927458 | 0.568636 | 0.072743 | ... | -0.467930 | 0.212570 | -0.606850 | 0.222851 | 0.328031 | 0.547468 |
| 3 | 3.157918 | -0.068339 | 1.265277 | -3.121363 | 0.375284 | -0.743678 | 0.621559 | 0.394781 | -0.665197 | 0.085816 | ... | -0.118920 | 0.236216 | -0.005051 | 0.417630 | 0.330657 | 0.160337 |
| 4 | 3.995018 | 0.832298 | 1.244687 | -3.642693 | -0.138776 | -0.311813 | -0.542681 | 0.655545 | 0.954961 | 0.115464 | ... | -0.173935 | -0.310959 | -0.940761 | -1.180681 | 0.656502 | 0.443758 |

5 rows × 62 columns

So Now we will give our ml model input as a x_comp, whose are the components of the features.

# CONCLUSION

## Key Findings and Conclusions of the Study

## Observation:-.

We observe that, RandomForest and XGBRegressor models are giving us the Best Performance Let's Tune the Parameters for them.

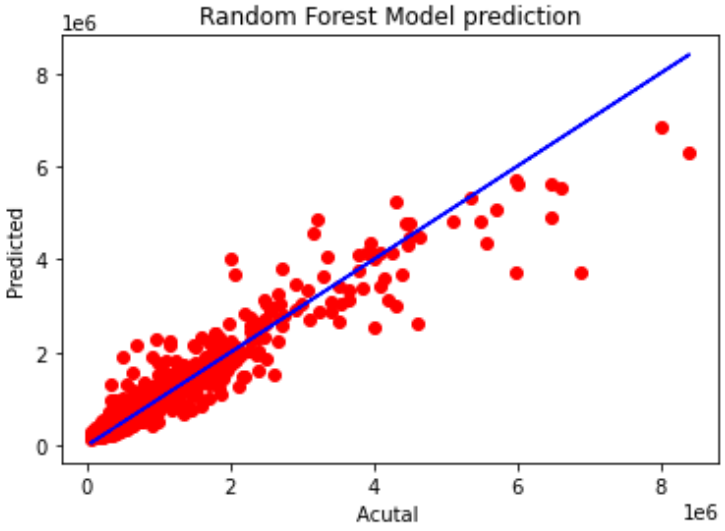
# Model Evaluation For Random Forest

```
rf=RandomForestRegressor(n_estimators=300,min_samples_split=3,min_samples_leaf=1)
rf.fit(x_train,y_train)
rf.predict(x_train)
y_pred=rf.predict(x_test)
score=r2_score(y_test,y_pred)
print("\nRandom Forest Regressor R2 Score is : ",score)
print("\nMean Squared Erros is : ",MSE(y_test,y_pred))
print("\nMean Abosute Erros is : ",MAE(y_test,y_pred))
print("\nRoot Mean Squared Error is: ",np.sqrt(MSE(y_test,y_pred)))
```

```
Random Forest Regressor R2 Score is :  0.9070656980867717

Mean Squared Erros is :  84319766060.79878

Mean Abosute Erros is :  150030.46290025633

Root Mean Squared Error is:  290378.6597889018
```

```
## plotting graph
plt.scatter(x=y_test,y=y_pred,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Acutal ')
plt.ylabel('Predicted')
plt.title('Random Forest Model prediction')
```

```
Text(0.5, 1.0, 'Random Forest Model prediction')
```



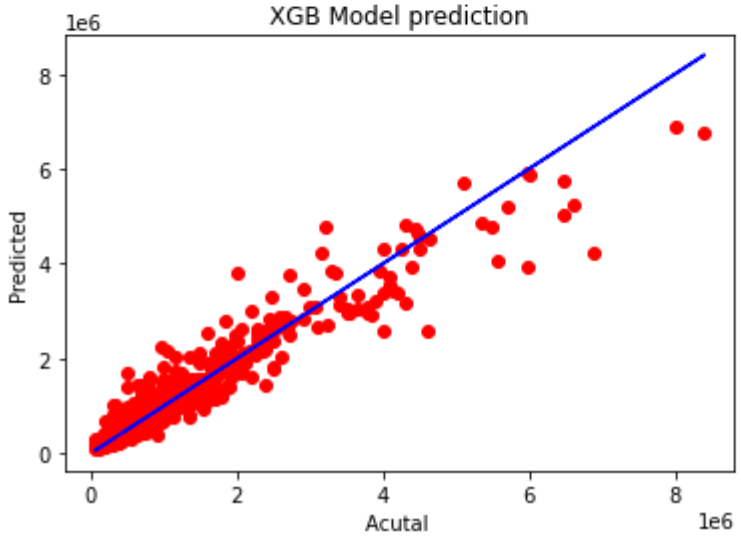
## Model Evaluation for XGB

```
xgb=XGBRegressor(colsample_bytree=0.7,gamma=0.1,learning_rate=0.1)
xgb.fit(x_train,y_train)
xgb.predict(x_train)
pred_y=xgb.predict(x_test)
score=r2_score(y_test,pred_y)
print("\nRandom Forest Regressor R2 Score is : ",score)
print("\nMean Squared Erros is : ",MSE(y_test,pred_y))
print("\nMean Abosute Erros is : ",MAE(y_test,pred_y))
print("\nRoot Mean Squared Error is: ",np.sqrt(MSE(y_test,pred_y)))
```

```
Random Forest Regressor R2 Score is :  0.9194068087806129

Mean Squared Erros is :  73122613392.59776

Mean Abosute Erros is :  145856.80180652085

Root Mean Squared Error is:  270411.93278514495
```

```
## plotting graph
plt.scatter(x=y_test,y=pred_y,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Acutal ')
plt.ylabel('Predicted')
plt.title('XGB Model prediction')
```

```
Text(0.5, 1.0, 'XGB Model prediction')
```




Here both of the models are performing quite well with our data. But XGB is slightly performing well in comparing with accuracy.

The increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. The proposed system will help to determine the accurate price of used car price prediction.

**So we have Finalize the model with XGB.**

## Learning Outcomes of the Study in respect of Data Science

Metrics considered to test the strength of the algorithm are mean Absolute Error, Mean Squared Error, Root Mean Squared Error. These three metrics are used evaluate both the regression algorithms. The three metrics have lower values for XGBoost, hence it has higher accuracy over the random forest regressor algorithm.
Using data mining and machine learning approaches, this project proposed a scalable framework for Indian based used cars price prediction. Cardekho.com website was scraped using the Selenium scraping tool to collect the benchmark data. An efficient machine learning model is built by training, testing, and evaluating two machine learning regressors named Random Forest Regressor, XGB Regressor. As a result of pre-processing and transformation, XGB came out on top with 92% accuracy followed by Random Forest Regressor with 91%.Each experiment was performed in real-time within the scikit-learn collaboration.In comparison to the system's integrated Jupyter notebook and Anaconda's platform, algorithms took less training time in scikit learn colab.

## Limitations of this work and Scope for Future Work

In the future, more data will be collected using different web-scraping techniques, and deep learning classifiers will be tested.

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 second-hand market. Hence

having a regular Data collection and analysis is required periodically, ideally, we would be having a real time processing program.

In this data set, one of the biggest challenges is that the distribution of the predictor variables was violation normality. That's why, classical statistical methods were not much helpful for analysing this data. Besides, in the dataset there are only 6 numerical variables among 13 variables. This limits our opportunity to use Pearson-r correlation.