CAR PRICE PREDICTION PROJECT

Name of Data Scientist: Vaibhav Tayade

Contact details: vaibhav_t29@rediffmail.com

Project submitted as a part of internship projects

Problem Statement:

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model. This project contains two phase

Data Collection Phase: We have to collect data from the various car selling websites like olx, cardekho etc. using web scrapping technique.

Model Building Phase: We have to build model using various machine learning techniques which can predict datasets.

Introduction:

Cars have always been a status symbol for many, but very less can actually afford it especially in the developing country like India. Inflation is another thing which keeps people away from buying new cars. With the eve of electric cars in the markets, new cars market has also been affected, people think twice before buying the new car. 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.

Dealers: They are ones that can be interested in results of this study. If used car sellers better understand what makes a car desirable, what the important features are for a used car, then they may consider this knowledge and offer a better service to the masses. Although there are websites that offers an estimate value of a car. They may have a good prediction model. However, having a second model may help them to give a better prediction to their users. Therefore, the model developed in this study may help online web services that tells a used car's market value.

There are lots of individuals who are interested in the used car market at some points in their life because they wanted to sell their car or buy a used car. In this process, it's a big corner to pay too much or sell less then it's market value. So making a used car prediction model helps such individuals in determining the true values of the used car.

Business Goal:

We are required to model the price of used cars with the available independent variables. It will be used by the management or masses in cars reselling business to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels. Further, the model will be a good way for management to understand the pricing dynamics of a new market.

PHASE I)

DATA COLLECTION: We are using web scrapping techniques in order to collect data from various websites which we'll use later for model building.

Here we are using ***www.olx.com*** website to collect data for used cars. We'll use selenium and beautiful soup library both in order to collect data. We have collected Car name, Manufacturing Date, Kilometers Run, Fuel Type, City Information, Car Estimated Price. We see that Car name has lots of data in it, We need to separate data from it, we'll do it in later phase of the project.

Web scrapping process:

We have imported various libraries for web scrapping and we have scrapped data using below code.

In [8]:

```
#creating empty lists
Modelname=[]
RunningKm=[]
City=[]
CarPrice=[]

while range(15):

    #scrapping car modelnames
    try:
        carmodels=driver.find_elements_by_xpath("//span[@class='_2tW1I']")
        for i in carmodels:
            Modelname.append(i.text)
    except NoSuchElementException:
        Modelname.append("No Details Available")

    #scrapping kmran of car:
    try:
        kmran=driver.find_elements_by_xpath("//span[@class='_2TVI3']")
        for i in kmran:
            RunningKm.append(i.text)
    except NoSuchElementException:
        RunningKm.append("No Details Available")

    #scrapping city of car:
    try:
        city=driver.find_elements_by_xpath("//span[@class='tjgMj']")
        for i in city:
            City.append(i.text)
    except NoSuchElementException:
        City.append("No Details Available")

    #scrapping extimated price of car:
    try:
        carprice=driver.find_elements_by_xpath("//span[@class='_89yzn']")
        for i in carprice:
            CarPrice.append(i.text)
    except NoSuchElementException:
        CarPrice.append("No Details Available")

    #clicking loadmore button number of times:
    try:
        btn=driver.find_element_by_xpath("//button[@class='rui-39-wj rui-3evoE rui-1JPTg']")
        btn.click()
        time.sleep(5)
    except NoSuchElementException:
        break
```

We made final data frame out of it.

In [9]:

```
#Make data frame
df=pd.DataFrame({"Car Modelname": Modelname,
                 "RunningKm": RunningKm,
                 "City": City,
                 "Car Price": CarPrice,
                })
```

Saving the dataframe to csv file.

In [12]:

```
df1.to_csv('car_price_prediction_datafile.csv')
```

PHASE II)

The very first step which we do in making machine learning model is importing various libraries.

In [1]:
```
# Importing libraries:
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import numpy as np
from scipy.stats import zscore
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.linear_model import Lasso, Ridge
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import KFold
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

Reading csv datafile:

In [2]:
```
# reading csv file and creating dataframe:
df = pd.read_csv('car_price_prediction_datafile.csv')
```

DATA CLEANING PROCESS:

Data Cleaning process is one of the most important process in machine learning model building projects, we need to remove null values, repeated values, deal with string type and numerical data types with respect to dataset.

Here in Car Price column, we have data values in strings data types, which is having ‘commas’ and ‘rupee’ sign in it, we need to deal with it, we need to convert it to pure numerical data type. We have done that with the following code.

In [10]:
```
# editing car price column
# car price columns has 'comma' in its values which makes it string datatype, we need to convert it to integer datatype
df['Car Price']=df['Car Price'].str.replace(',','')
```

In [11]:
```
# splitting car price column as its string column and making column pure numeric column out of it.
df["Car Price"] = df["Car Price"].str.split()
# making new column CarEstimatedPrice as integer datatype out of it.
df["CarEstimatedPrice"] = df["Car Price"].str[1].astype("int")
```

We have city column with very much like statement type, we need to extract Main City and Suburb data from it. We have done that with the following code.

In [12]:
```
# splitting city column and making new maincity and suburb column out of it.
df["City"] = df["City"].str.split(",")
df["MainCity"] = df["City"].str[-1]
df["Suburb"] = df["City"].str[0]
```

We have RunningKm column in string datatype we need to convert it into pure numerical data type and we have to extract Manufacturing Date of the vehicle from it. We have done that with the following code.

In [13]:
```
# as runningKm column is string in nature we need to convert it to nuemeric column.
df["RunningKm"] = df["RunningKm"].str.split("-")
df["ManufacturingDate"] = df["RunningKm"].str[0]
df["KmRan"] = df["RunningKm"].str[-1]
```

In [14]:
```
# converting KMRan column to float column.
df["KmRan"]=df["KmRan"].str.replace(',','')
df["KmRan"] = df["KmRan"].str.split()
df["KilometersRan"] = df["KmRan"].str[0].astype("float")
```

We have got CarModel name column which has Car Manufacturer and Car Model name both in the same column, we need to extract both data into different columns. We have done that with the following code.

In [15]:
```
# splitting Car Modelname column and making new carname column and fuel column out of it.
df["Car Modelname"] = df["Car Modelname"].str.split(",")
df["CarName"] = df["Car Modelname"].str[0]
df["Fuel"] = df["Car Modelname"].str[2]
```

In [16]:
```
# splitting carname column and making carmanufacturer and model columns out of it.
df["CarName"]=df["CarName"].str.split()
df["CarManufacturer"]=df["CarName"].str[0]
df["Model"]=df["CarName"].str[1]
```

We can see that there are lots of irrelevant entries in the data set we have deleted those.

We can also see lots of repeated entry values with spelling changes or lower/Upper case differences, we need to deal with this. We have done that with the following code.

In [21]:
```
#cleaning the dataset
newdataframe['CarManufacturer']=newdataframe['CarManufacturer'].str.replace('Mahendra','Mahindra')
newdataframe['CarManufacturer']=newdataframe['CarManufacturer'].str.replace('MAHINDRA','Mahindra')
newdataframe['CarManufacturer']=newdataframe['CarManufacturer'].str.replace('MARUTI','Maruti')
newdataframe['CarManufacturer']=newdataframe['CarManufacturer'].str.replace('FORD','Ford')
newdataframe['CarManufacturer']=newdataframe['CarManufacturer'].str.replace('Hyundai.','Hyundai')
newdataframe['Model']=newdataframe['Model'].str.replace('SUZUKI','Suzuki')
newdataframe['Model']=newdataframe['Model'].str.replace('FIESTA','Fiesta')
newdataframe['Model']=newdataframe['Model'].str.replace('SCORPIO','Scorpio')
newdataframe['Model']=newdataframe['Model'].str.replace('bolero','Bolero')
```

We need to check the data types of the various columns.

```
In [23]:
1 #checking datatypes of the dataframe
2 dfnew.dtypes
```

```
Out[23]: CarManufacturer       object
         Model                 object
         ManufacturingDate     object
         Fuel                  object
         KilometersRan        float64
         MainCity              object
         Suburb                object
         CarEstimatedPrice      int32
         dtype: object
```

We need to convert Manufacturing data column to YearsOld (i.e. Cars Age) column and that too into integer data types.

```
In [24]:
1 # converting manufacturing date columns to age of the car:
2 # first converting manufacturingdate column to integer:
3 dfnew['ManufacturingDate'] = dfnew['ManufacturingDate'].astype("int")
```

```
In [25]:
1 # creating new column 'YearsOld' from ManufacturingDate column:
2 dfnew['YearsOld'] = 2021 - dfnew['ManufacturingDate']
```

Exploratory Data Analysis (EDA):

Exploratory Data Analysis is the technique in Data Analysis to understand the dataset which we are working to get thoroughly information.

Checking Description of the dataset:

```
In [29]:
1 dfnew.describe()
```

Out[29]:

	KilometersRan	CarEstimatedPrice	YearsOld
count	3859.000000	3.859000e+03	3859.000000
mean	69074.441824	7.474891e+05	7.735683
std	49885.260063	1.077293e+06	4.435308
min	0.000000	2.600000e+04	0.000000
25%	45000.000000	2.500000e+05	4.000000
50%	64000.000000	4.500000e+05	7.000000
75%	85000.000000	7.750000e+05	10.000000
max	610000.000000	8.700000e+06	38.000000

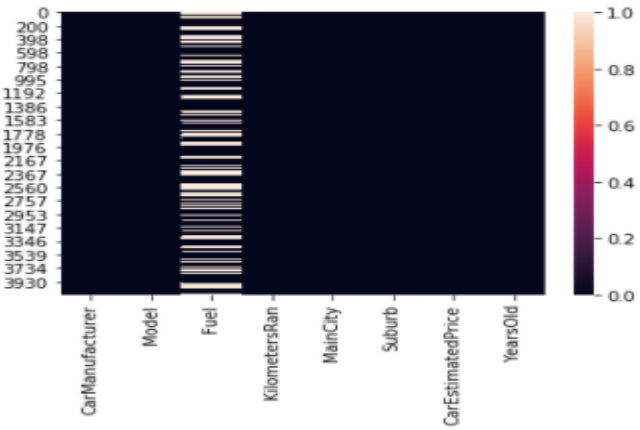
Observation: We can see that there are Mean values of the columns higher than the Median values which indicates presence of Outliers in the dataset.

Checking Null Values:

We have observed there are many null values present in the dataset by plotting heatmap.

```
In [30]:  1 # checking null values using heatmap.
          2 sns.heatmap(dfnew.isnull())

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8d63b7748>
```



We can see that there are Null Values present in the Fuel Column, we can deal with that by filling Null values with the mode values of the Fuel column which in this case is 'Diesel' we got to know that by checking count plot of the 'Fuel' column in the dataset.

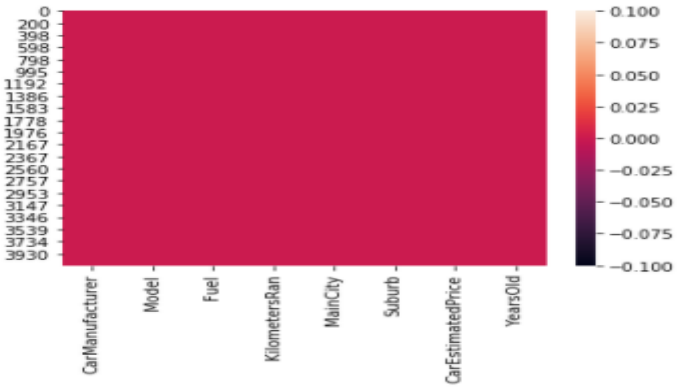
```
In [33]:  1 dfnew['Fuel'].value_counts()

Out[33]: Diesel          2755
         Petrol          1007
         CNG & Hybrids     97
         Name: Fuel, dtype: int64
```

After filling Null values with the mode value of the Fuel Column, and again checking the heatmap of the dataset. We have ensured that now there are no Null Values left in the dataset.

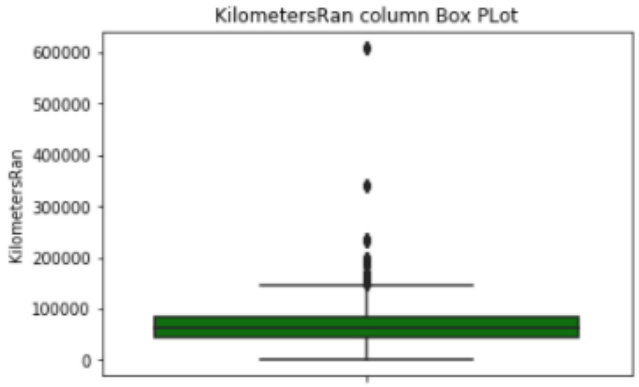
```
In [34]:  1 # checking null values using heatmap.
          2 sns.heatmap(dfnew.isnull())

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8d6bc4108>
```



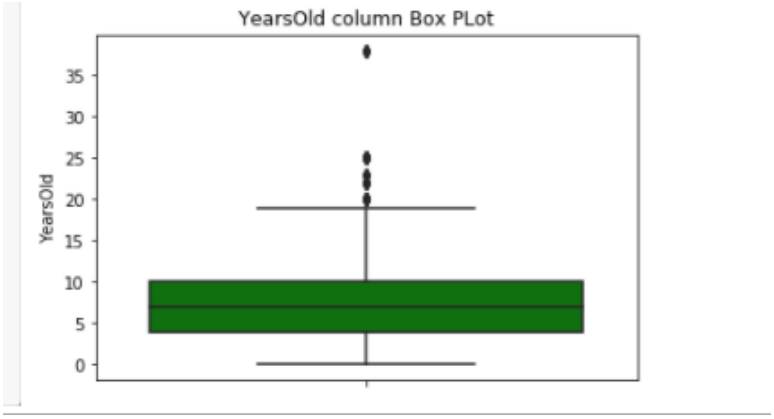
Checking Outliers: Outliers are something like irrelevant data values which are present in the dataset, it might hamper the performance of the model building we need to check for outliers in all the numerical columns. We can do that using box plot of the column.

Box plot 1:



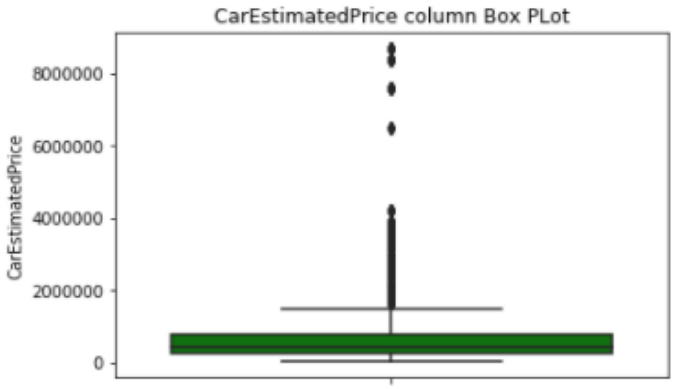
Observation: We can say that there are outliers present in the KilometersRan column.

Box plot 2:



Observation: We can say that there are outliers present in the YearsOld column.

Box plot 3:

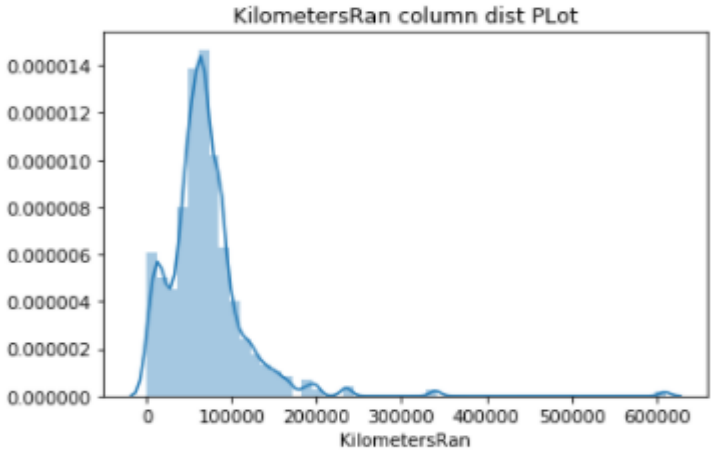


Observation: We can say that there are outliers present in the CarEstimatedPrice column.

Checking Skewness:

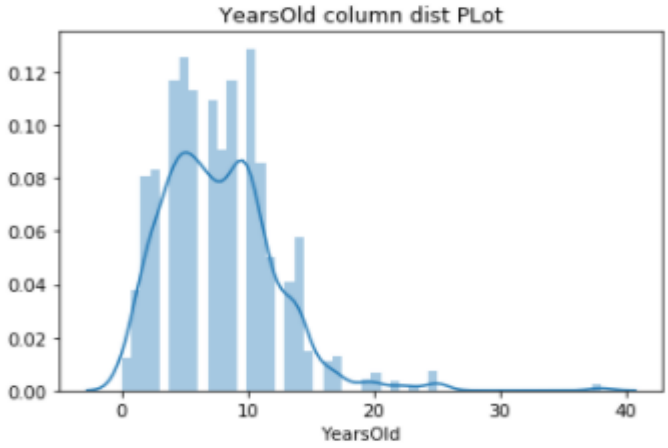
After checking the presence of outliers we need to check the presence of skewness in the datasets.

Distplot 1)



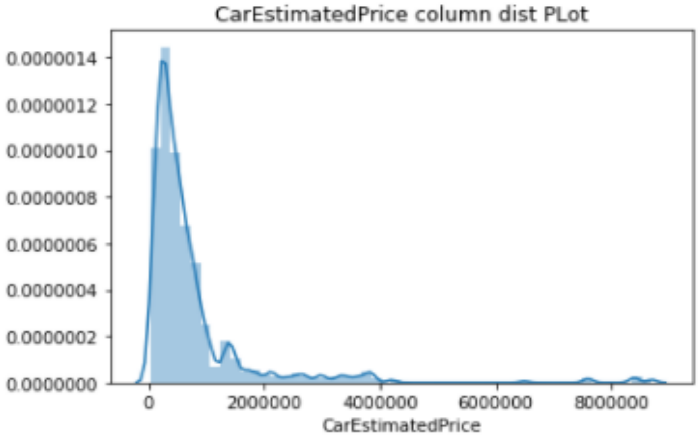
Observation: 'KilometersRan' Column is Right-hand side skewed.

Distplot 2)



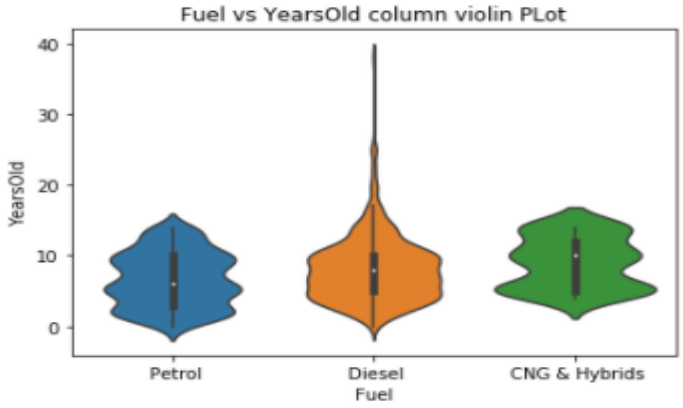
Observation: 'YearsOld' Column is Right-hand side skewed.

Distplot 3)



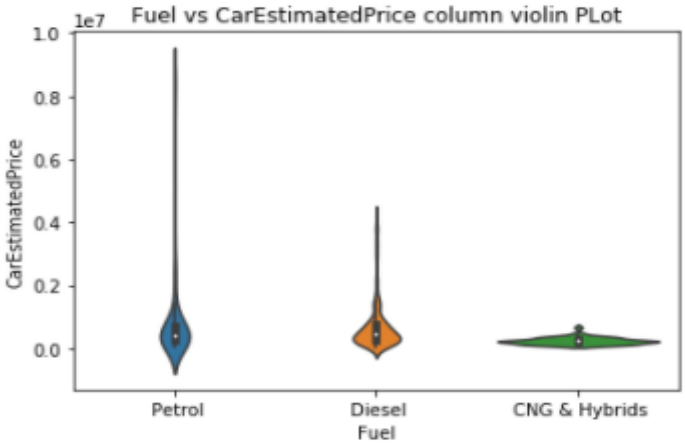
Observation: 'CarEstimatedPrice' Column is Right-hand side skewed.

Violin Plot 1) :



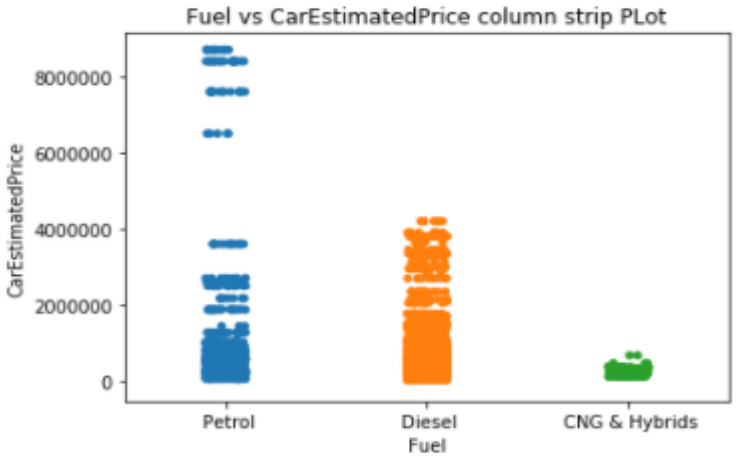
Observation: We can see that from the above Violin Plot, Diesel has more datasets with respect to petrol and CNG-Hybrid cars with YearsOld data

Violin Plot 2) :



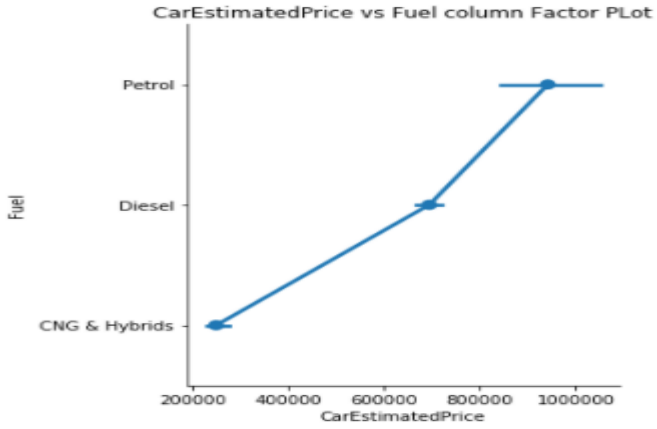
Observation: We can see that from the above violin plot that Petrol cars are highly priced than Diesel cars while CNG&Hybrid cars are least priced.

Strip Plot:



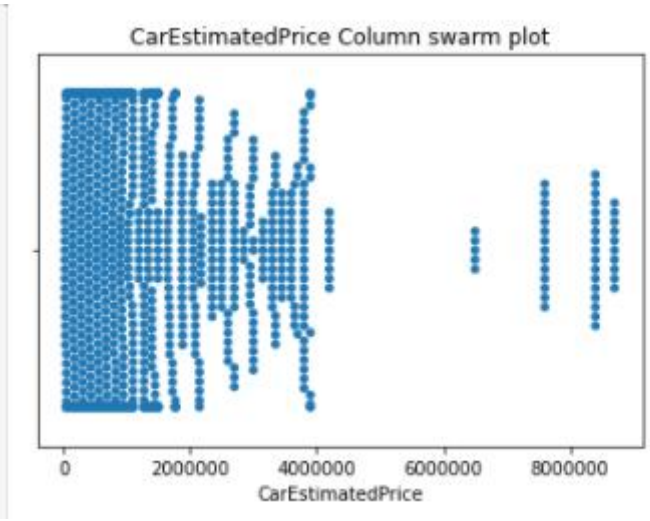
Observation: We can see that from the above strip plot that Petrol cars are highly priced than Diesel cars while CNG&Hybrid cars are least priced.

Factor Plot:



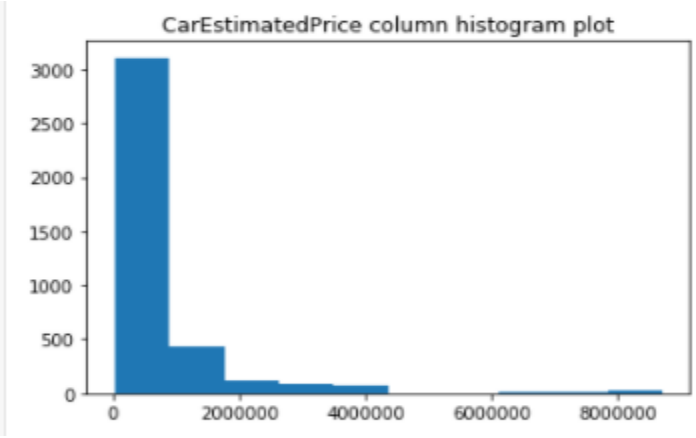
Observation: We can see that from the above Factor Plot that, Petrol cars are highly priced than Diesel cars while CNG&Hybrid cars are least priced.

Swarm Plot:



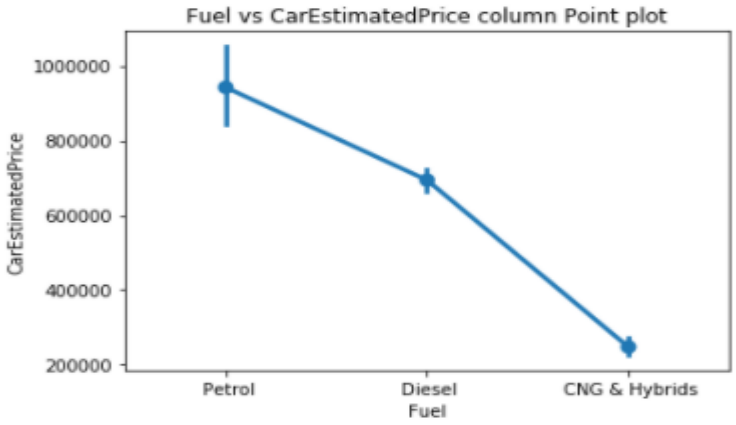
Observation: we can see that from the above swarmplot, 'CarEstimatedPrice' datasets are majorly in the 0 to 4lks price range and very less values in the high price range.

Histogram Plot:



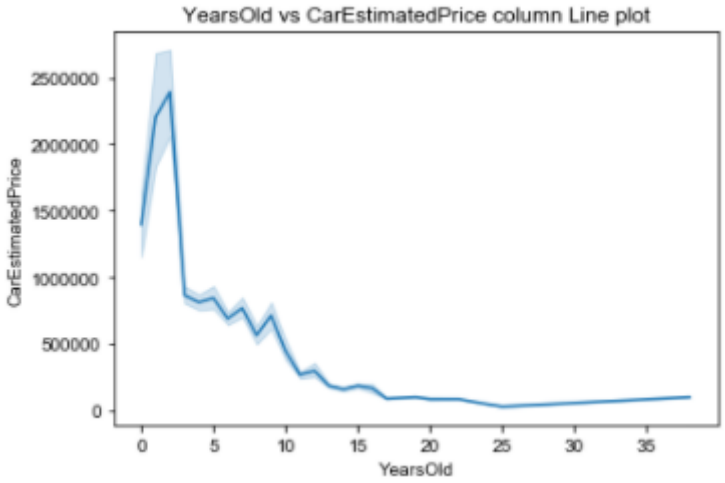
Observation: we can see that from the above histogram plot CarEstimatedPrice datasets are majorly in the 0 to 4lks price range and in that also majorly in the 0 to 2 lks price range and very less values in the high price range.

Column Plot:



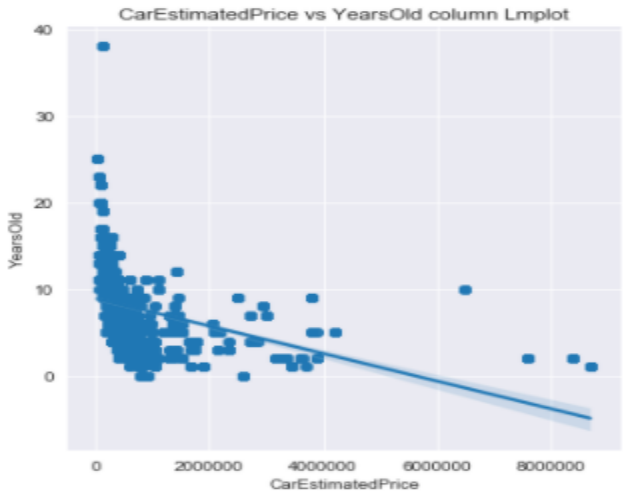
Observation: We can see that Petrol cars are highly priced than Diesel cars while CNG&Hybrid cars are least priced.

Line Plot:



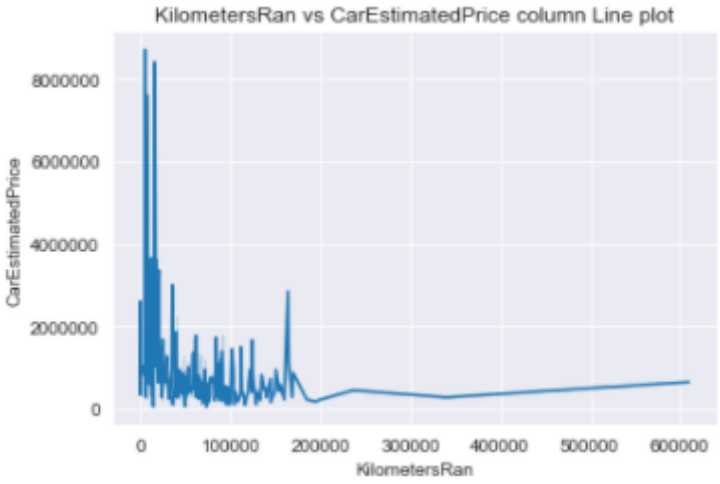
Observation: We can see that from the above lineplot, YearsOld values are inversely proportional to the CarsEstimatedPrice columns.

LmPlot:



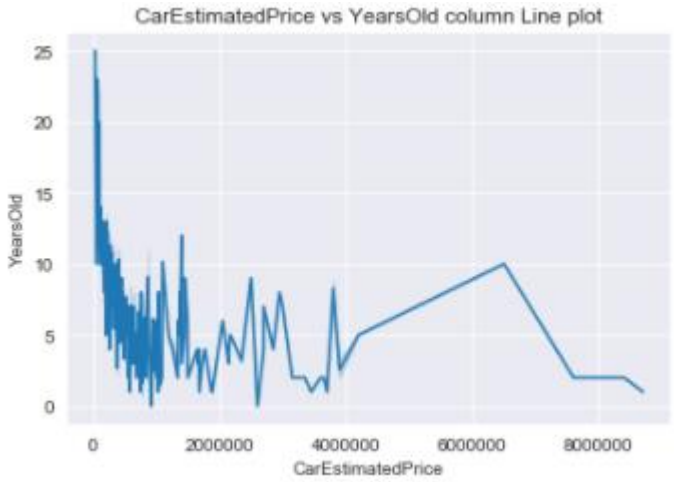
Observation: We can see that from the above lmplot, YearsOld values are inversely proportional to the CarsEstimatedPrice columns.we can also see the distribution of the datasets in the in the graphs.

LinePlot:



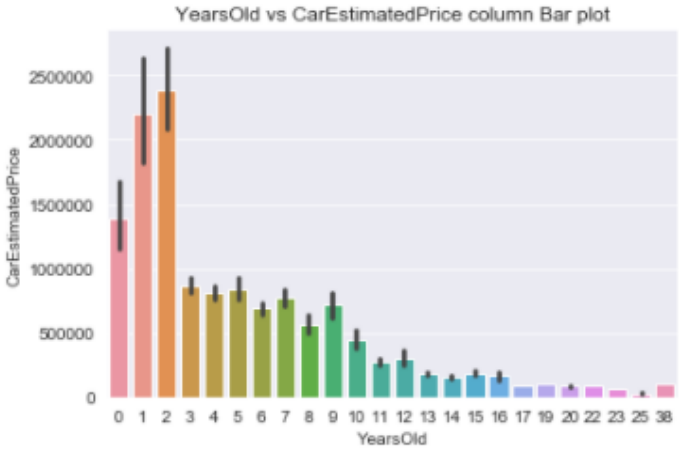
Observation: We can see that from the above lineplot that 'KilometersRan' column are inversely proportional to the CarEstimatedPrice colum, but datasets distribution are very much widely distributed.

Line Plot 2)



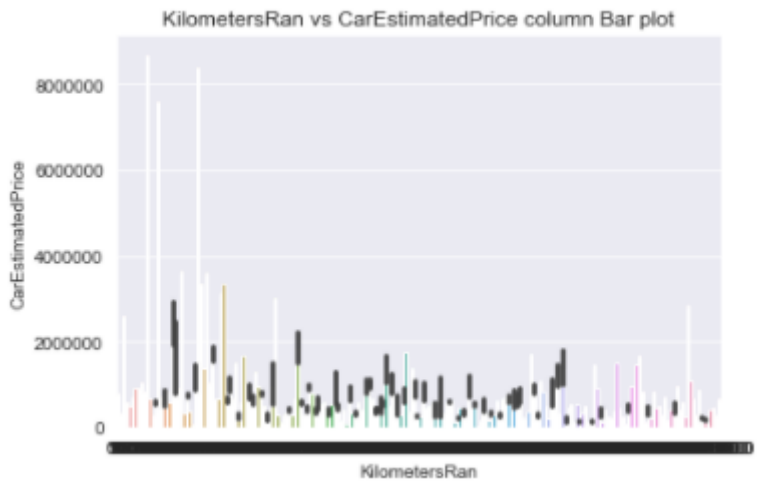
Observation: We can see that from the above lineplot that 'YearsOld' column are inversely proportional to the CarEstimatedPrice colum, but datasets distribution are very much widely distributed.

Bar plot 1:



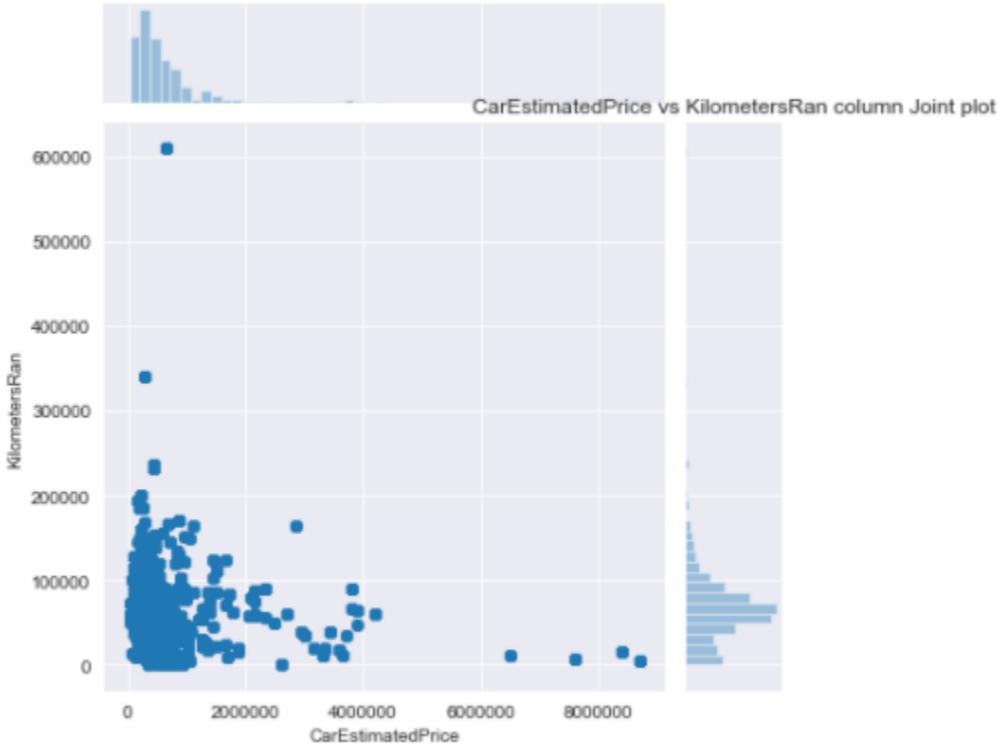
Observation: We can see that from the above barplot that 'YearsOld' column are inversely proportional to the CarEstimatedPrice colum, but datasets distribution are very much widely distributed. we have 1 and 2 yearsold cars having highly priced with respect to others while cars more than 12 years old are very mcuh less priced.

Bar plot 2:



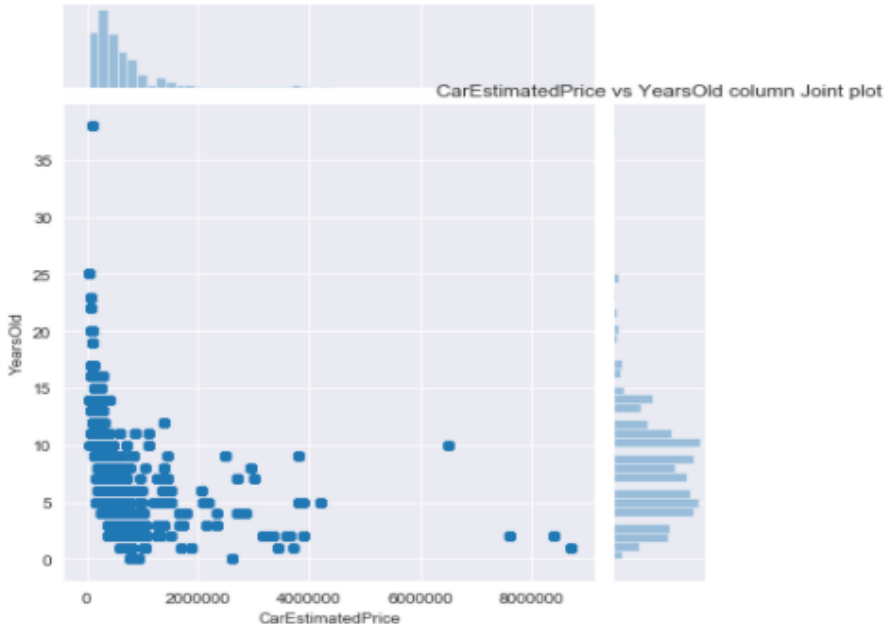
Observation: KilometersRan column is inversely proportional to the CarsEstimatedPrice column.

Column Joint Plot:



Observation: We can see the data distribution in the above joint plot, we can see that Cars Price are widely distribited between 0 to 4cr and KilometersRan values of 0 to 2lakh Kms.

Column Joint Plot 2):



Observation: We can see the data distribution in the above joint plot, we can see that Cars Price are widely distribited between 0 to 4 lks and YearsOld values of 0 to 15 years.

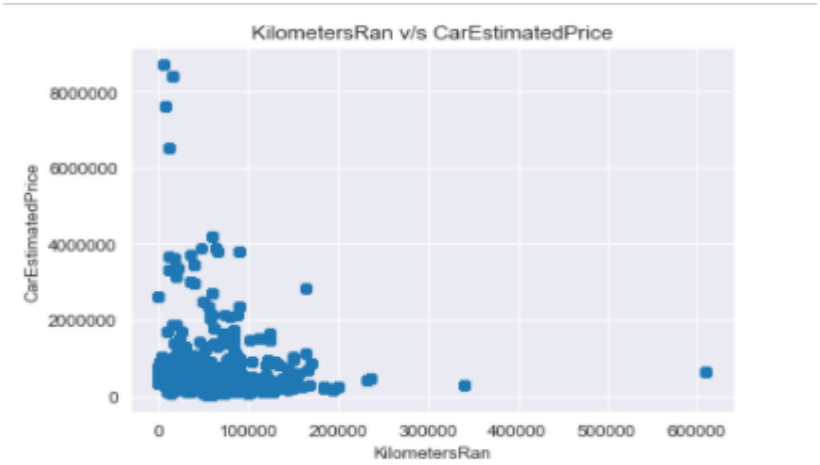
Checking Amount of Skewness:

```
1 #checking amount of skewness in the dataset:
2 dfnew.skew()
```

```
KilometersRan        4.237455
CarEstimatedPrice    4.431625
YearsOld             1.231332
dtype: float64
```

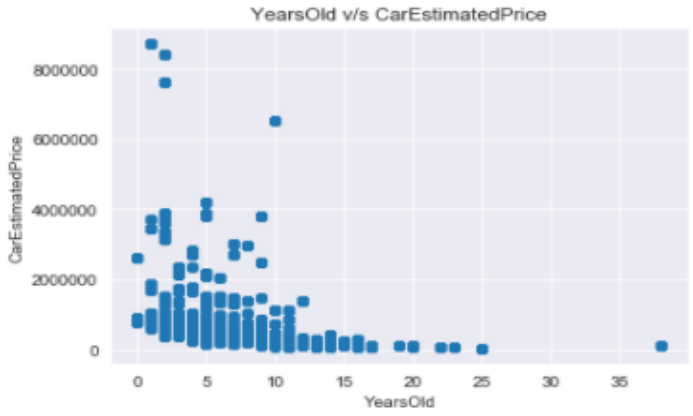
Observation: we can see the amount of skewness in the above code. CarEstimatedPrice column is highly skewed while YearsOld column is less skewness comparately.

Scatter plot 1)



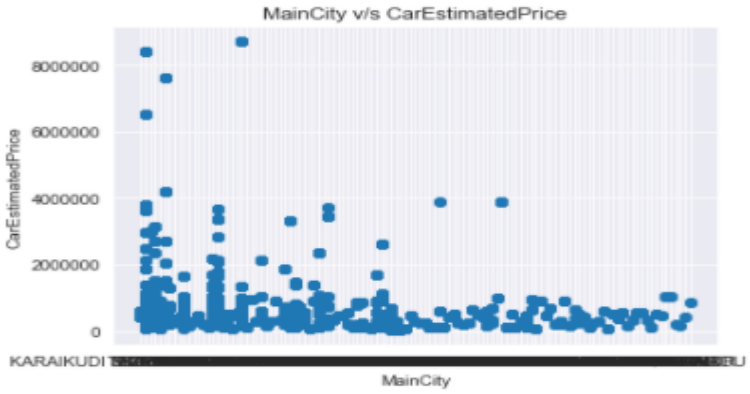
Observation: We can see that kilometersRan column is inversely proportional to the CarEstimatedPrice column. Less the KilometerRan value medium to high the price is while High the value of KilometersRan column is Less the price.

Scatter plot 2)



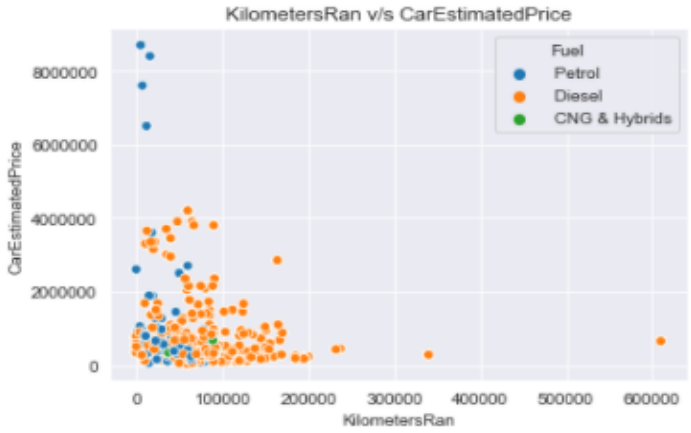
Observation: YearsOld column is also seems to be inversely proportional to CarEstimatedPrice. Less yearsold value is mediam to high price of car it is, and while more the yearsold value is less the price.

Scatter plot 3)



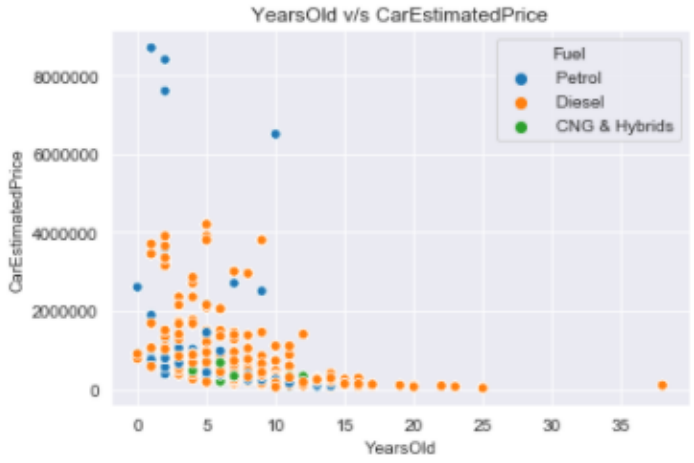
Observation: We can see the city wise distribution of cars estimated prices.

Scatter plot 4)



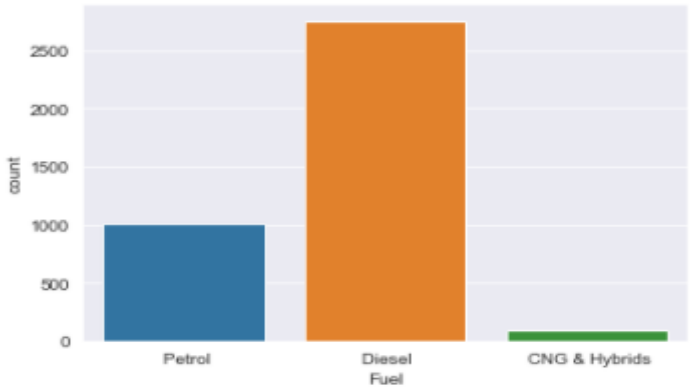
Observation: 1) we can see that petrol cars are highly priced if they are less KilometerRan 2) Diesel cars are usually less priced than Petrol Cars if they are less driven.

Scatter plot 5)



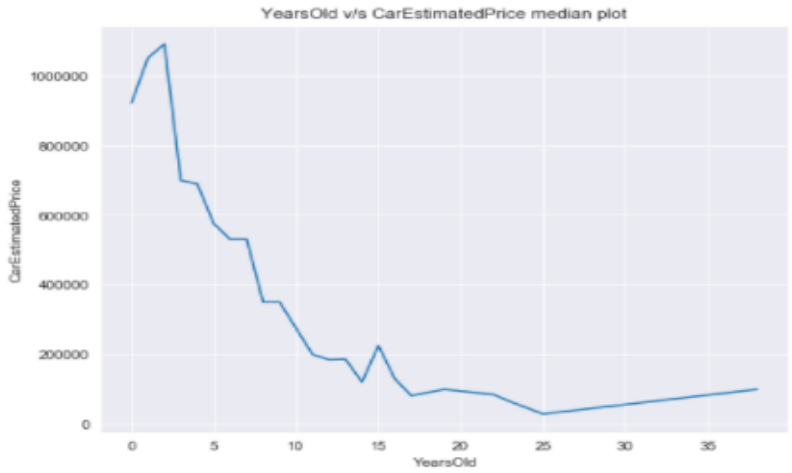
Observation: 1) we can see that petrol cars are highly priced if they are less YearsOld. 2) Diesel cars are usually less priced than Petrol Cars if they are less YearsOld.

Count Plot:



Observation: We can see the dataset distribution, Diesel cars are having high dataset while Petrol cars next to that while CNG & Hybrid having least datasets.

Median Line Plot:



Observation: We can see that from the above median plot, YearsOld column is inversely proportional to CarEstimatedPrice column.

LABEL ENCODING THE DATASET:

Before proceeding further, we need to confirm that all columns are numeric in nature, but we found out that there are columns which are of Object or Categorical data type in nature. We need to convert it into numeric data types, we can do that using Label Encoding technique for converting this column in to numeric. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning. We Label Encode the categorical data columns like 'CarManufacturer', 'Model', 'Fuel', 'MainCity', 'Suburb', so that we can work on model building process smoothly.

REMOVING OUTLIERS USING Z-SCORE METHOD:

Here in this dataset, we have used z-score method of removing outliers. A z-score (also called a standard score) gives you an idea of how far from the mean a data point is. But more technically it's a measure of how many standard deviations below or above the population mean a raw score is. A z-score can be placed on a normal distribution curve. We can choose a threshold value, Here, in this problem we have choosen 3 as a threshold value above which all are outliers. After removing outliers using z-score method, we found that we are loosing almost 3.75 percent of data, which is very much in the limits.

CORRELATION MATRIX: Correlation matrix is something by which we get to know how correlated all the variables with eachother. The correlation matrix is a matrix structure that helps the programmer analyze the relationship between the data variables. It represents the correlation value between a range of 0 and 1. The positive value represents good correlation and a negative value represents low correlation and value equivalent to zero(0) represents no dependency between the particular set of variables.

Observation:

1) CarManufacturer column is highly coorelated with CarEstimatedPrice column with respect to all other columns. 2) YearsOld column is **negatively 0.46 units** correlated with CarEstimatedPrice column.

Creating X-Y Dataset: Basically we are splitting dataset into dependable variables target variables, here in this case, Y is the our target variable and X is our dependable variable.

Scaling the Dataset: The most important step in machine learning model making is to ensure all data columns are in same scale of values, we can bring them in same scale by scaling them using Standard Scaler scaling library. Standard Scaler removes the mean and scales each feature/variable to unit variance. This operation is performed feature-wise in an independent way. StandardScaler can be influenced by outliers (if they exist in the dataset, but we have already removed outliers) since it involves the estimation of the empirical mean and standard deviation of each feature.

Creating Train-Test Dataset: The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets Here we have kept 20% of dataset for testing the model while 80% of dataset for training the model.

MODEL MAKING PROCESS: Here in our problem we will try making machine learning model with six different types of algorithms i.e. Linear Regression, Lasso - Ridge Regularization method, ElasticNet Regularization method, Random Forest Regressor, Ada Boost Regressor, Support Vector Regressor methods. We will try to find out each model's Accuracy score, Mean Absolute error, Mean Squared Error and Root Mean Squared Error as well. Accuracy score will give us the percentage accuracy of the model in predicting the test datasets. Mean Absolute Error is nothing but the magnitude of difference between the prediction of an observation and the true value of that observation. Mean Squared Error is nothing but the average of the square of the difference between the original values and the predicted values. Root mean square error or root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance.

LINEAR REGRESSION MODEL: Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). The term linear model implies that the model is specified as a linear combination of features. Based on training data, the learning process computes one weight for each feature to form a model that can predict or estimate the target value. Our LinearRegression Model has shown accuracy score of **22.95%**

MEAN ABSOLUTE ERROR: 382407.85487295274

MEAN SQUARED ERROR: 387198540173.02576

ROOT MEAN SQUARED ERROR: 622252.7944276552

r2 Score of Linear Regression model: 0.25714473491560175

Cross Validation Score of the Linear Regression model: **23.11%**

LASSO REGULARIZATION MODEL: In statistics and machine learning, lasso (least absolute shrinkage and selection operator) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

Our Lasso regression model has shown accuracy of **22.95%**

RIDGE REGULARIZATION MODEL: Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

Our Ridge Regression model has shown accuracy of **22.95%**

ELASTICNET REGULARIZATION MODEL METHOD: Elastic net is a popular type of regularized linear regression that combines two popular penalties, specifically the L1 and L2 penalty functions. Elastic Net is an extension of linear regression that adds regularization penalties to the loss function during training.

Our ElasticNet Regression model has shown accuracy of **22.95%**

MEAN ABSOLUTE ERROR: 382154.92607071984

MEAN SQUARED ERROR: 387327952837.8983

ROOT MEAN SQUARED ERROR: 622356.7729509323

Cross validation score of ElasticNet Regularization method is **23.12%**

SUPPORT VECTOR REGRESSION METHOD: SVR is built based on the concept of Support Vector Machine or SVM. It is one among the popular Machine Learning models that can be used in classification problems or assigning classes when the data is not linearly separable.

Our **linear, poly and rbf kernel** all types of SupportVector Model has **shown negative accuracy score**.

RANDOM FOREST REGRESSION METHOD: Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

Our Random Forest Regression model has shown accuracy of **99.96%**

MEAN ABSOLUTE ERROR: 4447.080067294751

MEAN SQUARED ERROR: 741768493.3749887

ROOT MEAN SQUARED ERROR: 27235.427174454024

Cross validation score of our random forest regressor model is **99.91%**

ADA BOOST REGRESSION METHOD: AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re□assigned to each instance, with higher weights assigned to incorrectly classified instances. An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

Our Ada Boost Regression model has shown accuracy of **52.83%**

MEAN ABSOLUTE ERROR: 415214.10512553796

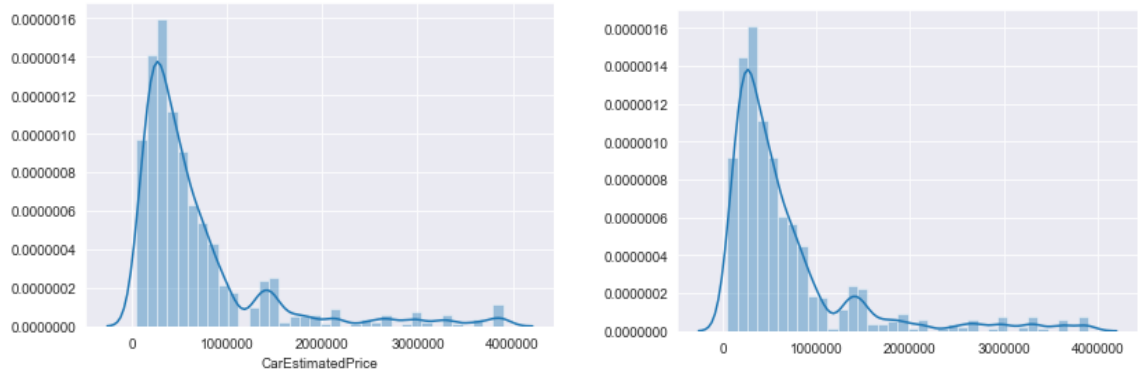
MEAN SQUARED ERROR: 244706819416.81442

ROOT MEAN SQUARED ERROR: 494678.5010659089

Cross validation score of our Ada Boost Regressor model is **50.38%**

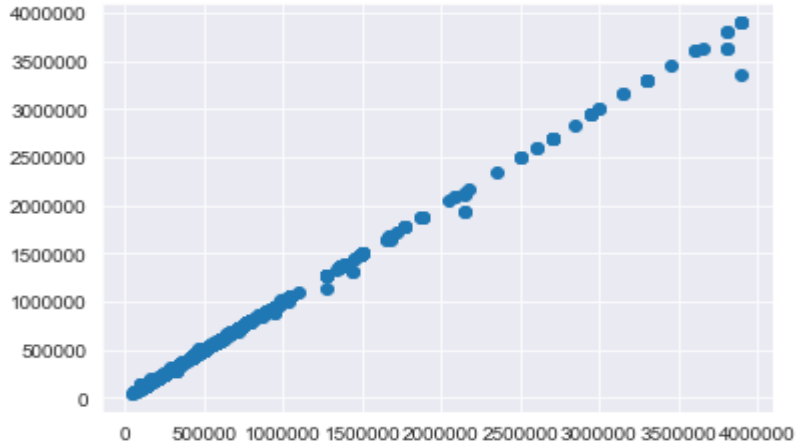
FINAL MODEL SELECTION: After checking all models accuracy score and cross validation score and also their errors, we found out that **Random Forest Regression Model** seems to be the best model, as it has good accuracy score and least error comparatively. The difference between the accuracy score

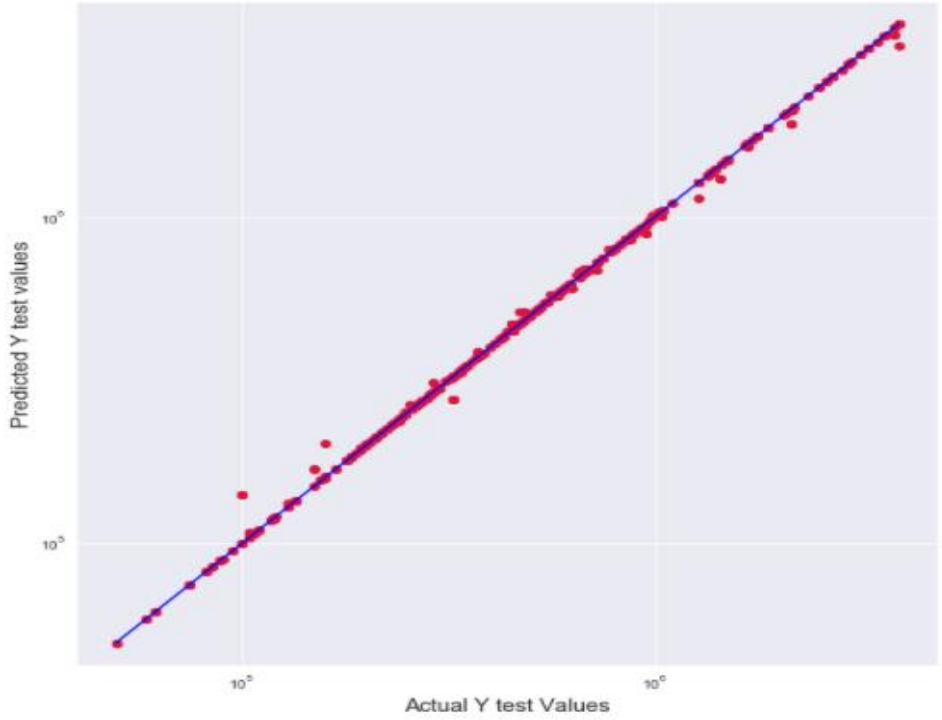
and cross validation score is also minimal. So we'll consider this **Random Forest Regression Model** model as our final model which is having **accuracy** of **99.96%**



In order to study efficiency of our model we checked the **distplot of y_test column and distplot of pred column**, we found out that they both look exactly **similar.**

We even checked the **scatter plot of y_test and predicted y_test** as below.





Observation: we can see that both the **y_test and predicted y_test datasets falls in one line.** we can say that our model is very well made and have good accuracy score.

SAVING THE MODEL:

Our final model is saved in pkl format with '**Vaibhav_Car_Price_Prediction_project_Model.pkl**' file.

CONCLUSION:

1) **How many Kilometers has car ran is inversely proportional to Car's Estimated price.**
2) **How many years old the car is inversely inversely proportional to Car's Estimated price.**
3) **Petrol Cars are usually high priced than Diesel cars and then CNG-Hybrid cars.**
4) **Most of the cars are priced between 0 to 4 lakhs and in that too between 0 to 2 lakhs.**
5) **Cars more than 15 years old, are very much less priced.**
6) **Random Forest Regression model seems to be the best fit model as its showing less overfitting, verified by cross validation score.**
7) **Random Forest Regression model is showing accuracy of 99.96%**

THANK YOU