FLIP ROBO

CAR-PRICE PREDICTION PROJECT

Submitted by:

Durgadhar Pathak

Internship 15

ACKNOWLEDGMENT

The internship opportunity I have with Flip Robo Technologies is a great chance for learning and professional development. I am also grateful to our SME Mr. Sajid Choudhary and Ms. Sapna Verma for their valuable and constructive suggestions during the planning and development of this project. Their quick support and references helped a lot in building this project. Also, I am very thankful to our SMEs & Flip Robo Team for understanding technical issue faced by me and provide quick resolution & providing enough time for submission.

Also, I am thankful to DT support Team for their continuous effort to resolve our queries during project building.

Research papers that helped me in this project was as follows: -

1- https://www.researchgate.net/publication/331994496_Car_price_prediction_using_machine_learning_techniques
2- http://www.ijirt.org/Article?manuscript=151705
3- https://www.hpej.net/journals/pakjet/article/view/1079

Articles that helped me in this project was as follows:

1- https://medium.com/geekculture/used-carprice-prediction-complete-machine-learning-project-d25559cf2d2a
2- https://ieeexplore.ieee.org/abstract/document/9396868/
3- https://arxiv.org/ftp/arxiv/papers/1711/1711.06970.pdf

References:

1- https://machinelearningmastery.com/
2- https://scikit-learn.org/stable/
3- https://www.geeksforgeeks.org/machine-learning/
4- https://pandas.pydata.org/
5- https://www.datacamp.com/
6- https://www.ibm.com/cloud/learn/machine-learning
7- https://www.selenium.dev/selenium/docs/api/py/common/selenium.common.exceptions.html
8- https://www.oreilly.com/library/view/web-scraping-with/9781491985564/
9- https://gallery.azure.ai/Experiment/Automobile-price-prediction-256

INTRODUCTION

• Business Problem Framing

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.

• Conceptual Background of the Domain Problem

This project contains two phase-

1- Data Collection Phase: -

We have to scrape at least 5000 used cars data. We can scrape more data as well, it's up to us. more the data better the model. In this section we need to scrape the data of used cars from websites (car Dekho, Cars24 etc.) We need web scraping for this. We have to fetch data for different locations. The number of columns for data doesn't have limit, it's up to us and our creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometres, fuel, number of owners, location and at last target variable Price of the car. This data is to give us a hint about important variables in used car model. We can make changes to it, we can add or we can remove some columns, it completely depends on the website from which we are fetching the data. We will try to include all types of cars in our data for example- SUV, Sedans, Coupe, minivan, Hatchback.

2- Model Building Phase: -

After collecting the data, we need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

Follow the complete life cycle of data science. Include all the steps like.

- Data Cleaning
- Exploratory Data Analysis
- Data Pre-processing
- Model Building
- Model Evaluation
- Selecting the best model

• Review of Literature

First, we need to collect used car data from different websites using web scraping techniques and then need to build a machine learning model. Machine learning algorithms enable the creation of a new model using existing anonymized historical data that would be used to train the model to make better predictions not only for car prices, but also for other variables like Driven kilometres, Foundation, Sale Condition etc. With use of good model, car companies could predict the price easily. To mitigate the subjective part of the decision-making process, different scoring models are introduced to evaluate certain parameters that could affect the car prices.

Models used: -

1- **Random Forest Regression**: - Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

2- **AdaBoost Regression**: - 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.

3- **k-nearest neighbors**: - K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors.

Other models used are: - Linear Regression, Decision Tree, Ridge & Lasso regression.

Hyper Parameter tuning: - For Lasso regression, k-nearest neighbors & AdaBoost Regression algorithms, we used Grid search cross-validation technique to choose the best hyper-parameters & for implementing best model using Random Forest Regression algorithm, we have used Randomized Search CV to find best hyperparameters.

Evaluation Matrix: - Coefficient of determination (R2 score), Cross Validation Score, Mean Absolute Error, Mean Squared Error & Root Mean Squared Error.

• Motivation for the Problem Undertaken

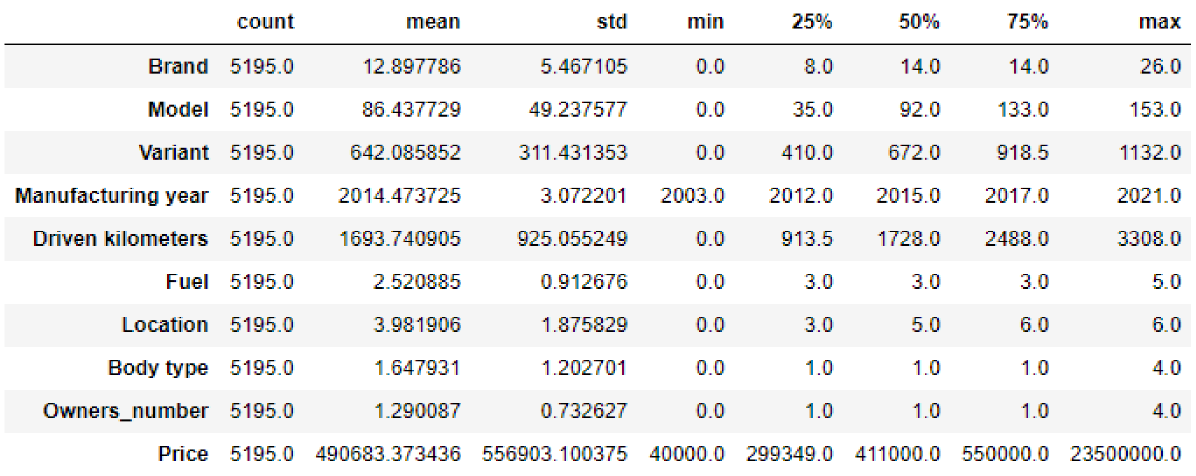
Deciding whether a used car is worth the posted price when we see listings online can be difficult. Several factors including Brand, Model, Variant, Body type, Fuel, Number of Owners, Driven Kilometres, mileage and Manufacturing year can influence actual worth of car. From perspective of a seller, it is also a dilemma to price a used car appropriately. Based on existing data sources, our aim is to use machine learning algorithms to develop models for predicting used car prices.

Analytical Problem Framing

• Mathematical/ Analytical Modelling of the Problem

In this project, input data is provided to the model along with the output data so it is a type of supervised learning. Also output Variable “Price" is continuous in nature so it is a Regression based problem and We have to predict the price of cars with available independent variables. We have performed regression tasks and it models a target prediction value based on independent variables and is mostly used for finding out the relationship between variables and forecasting. Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data and other attributes. We have checked statistical summary, correlation matrix, skewness, missing values & outliers in dataset and try to handle them very carefully.

Statistical Summary: summary statistics is used to summarize set of observations, in order to communicate the largest amount of information as simply as possible. It includes central Tendency, dispersion, skewness, variance, range, deviation etc.



	count	mean	std	min	25%	50%	75%	max
Brand	5195.0	12.897786	5.467105	0.0	8.0	14.0	14.0	26.0
Model	5195.0	86.437729	49.237577	0.0	35.0	92.0	133.0	153.0
Variant	5195.0	642.085852	311.431353	0.0	410.0	672.0	918.5	1132.0
Manufacturing year	5195.0	2014.473725	3.072201	2003.0	2012.0	2015.0	2017.0	2021.0
Driven kilometers	5195.0	1693.740905	925.055249	0.0	913.5	1728.0	2488.0	3308.0
Fuel	5195.0	2.520885	0.912676	0.0	3.0	3.0	3.0	5.0
Location	5195.0	3.981906	1.875829	0.0	3.0	5.0	6.0	6.0
Body type	5195.0	1.647931	1.202701	0.0	1.0	1.0	1.0	4.0
Owners_number	5195.0	1.290087	0.732627	0.0	1.0	1.0	1.0	4.0
Price	5195.0	490683.373436	556903.100375	40000.0	299349.0	411000.0	550000.0	23500000.0

Observations:

1- For input features, Driven kilometres has highest standard deviation of 925.05.

2- Maximum Price of car is 23500000.0 and minimum price is 40000.0.

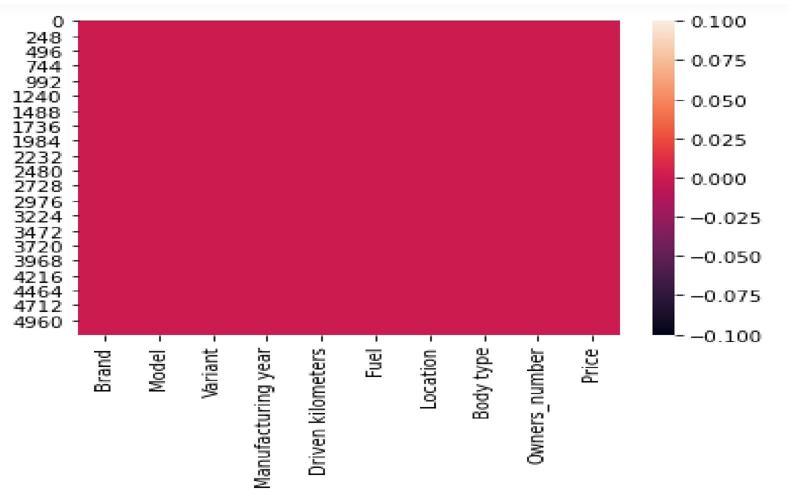
3- In body type & owners’ number, the value of mean is considerably greater than median so there are strong chances of positive skewness.

4- In remaining columns, value of median is greater than mean so the columns are negatively skewed.

Correlation: After seeing many correlated values we can say that many columns have correlation values and dropping some of these will be better for our dataset.

Skewness: If the skewness is between -0.5 and 0.5, then dataset is fairly symmetrical and symmetrical distribution will have a skewness of zero. So accordingly, we are removing skewness using NumPy mathematical functions log & square transform.

Missing Data: -



Dataset has no missing values.

- ## Data Sources and their formats

First, we need to collect used car data from different websites using web scraping techniques and then need to build a machine learning model. So, we have scraped used car data from different websites such as: - www.cardekho.com and www.cars24.com using Selenium web scraping methods. We have scraped following data: -

Dataset: - Different columns are as: -

1- Brand-> Brand name of Car

2-Model-> Model Information

3-Variant-> Model Variant

4-Manufacturing Year-> Year of Manufacturing

5-Driven Kilometres-> Distance Travelled

6-Fuel-> Type of Fuel

7-Location-> Location belongs to used car

8-Body Type-> Body type of Car

9-Owners_number-> Number of owners of used car

10-Ref. Website-> Source of information

11- Price-> Current Price of used car

We have combined data from all websites into a csv file and used it for car price prediction project. Total dataset has 5195 rows and 11 columns. We have also tried to equalize the dataset format for better understanding and use.

Target Variable: - Target Variable is Price in this project and it is continuous in nature so we will use Regression algorithms to make our model.

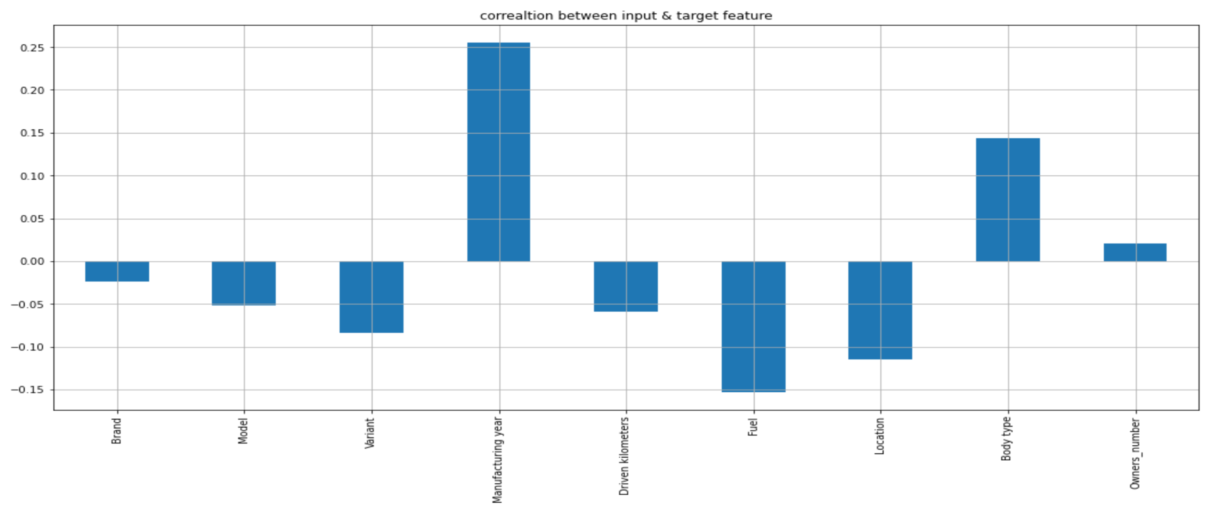
• Data Pre-processing Done

The data pipeline starts with collecting the data and ends with communicating the results & Data pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format. It involves 4 steps: - Cleaning, Formatting, Scaling, and Normalization. While building a machine learning model, if we haven't done any pre-processing like correcting outliers, handling different formats, normalization and scaling of data, or feature engineering, we might end up considering those 1% of results that are false. Some steps performed in this project are: -

- Column wise Empty cell analysis done & found no missing values.
- Checked unique values in each column to explore dataset more deeply.
- We had seen outliers in some columns so we were trying to remove them using Z scores and data loss was only 2.75%.
- Skewness was present in dataset. If the skewness is between -0.5 and 0.5, then dataset is fairly symmetrical and symmetrical distribution will have a skewness of zero. So accordingly, we removed skewness using NumPy mathematical functions log, cube root & square root transform.

- **Feature Encoding and Normalization**: - Features came in a variety of format, e.g., integers, floating numbers, string values, etc. So, we Checked Concise Summary of our Data Frame and we have noticed that 8 columns have object (str or mix str) data type. This was a challenge for us as these features cannot be directly used for training. To prevent regression biases towards certain features, we dropped some of the irrelevant features and make sure that we haven't lose important data. In the end, we also normalize the feature values so that all features are evaluated on the same scale.

- **Correlation:** - We checked the linear relationship between two variables. After seeing these correlated values, we had found that many columns had multicollinearity is also present between various columns. We have checked correlation between input variables and output variable "Price": -

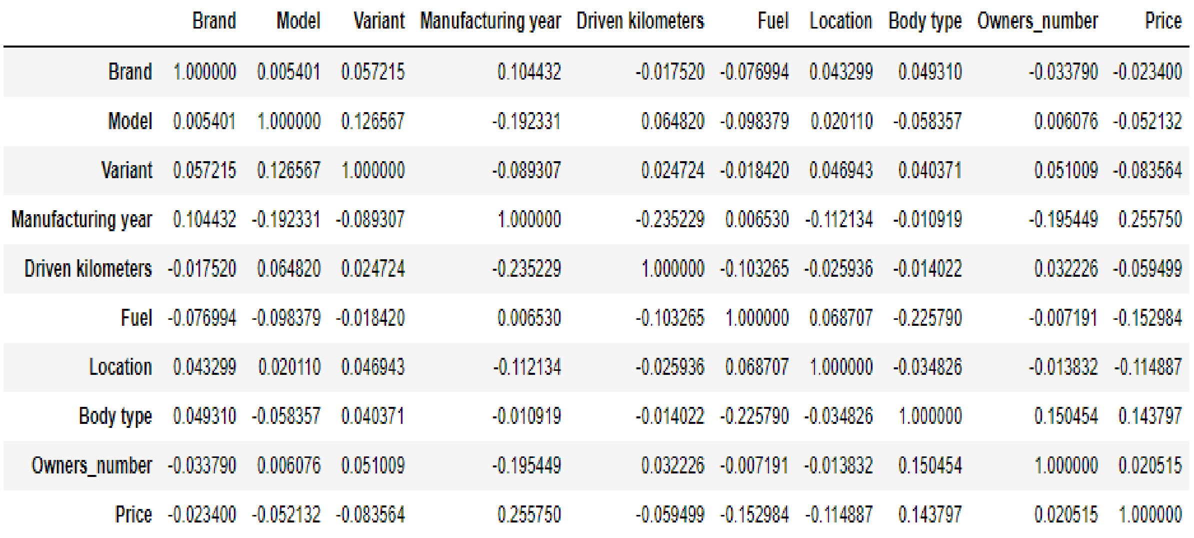


Observation:

1- Manufacturing Year column is most positively correlated with Price column.

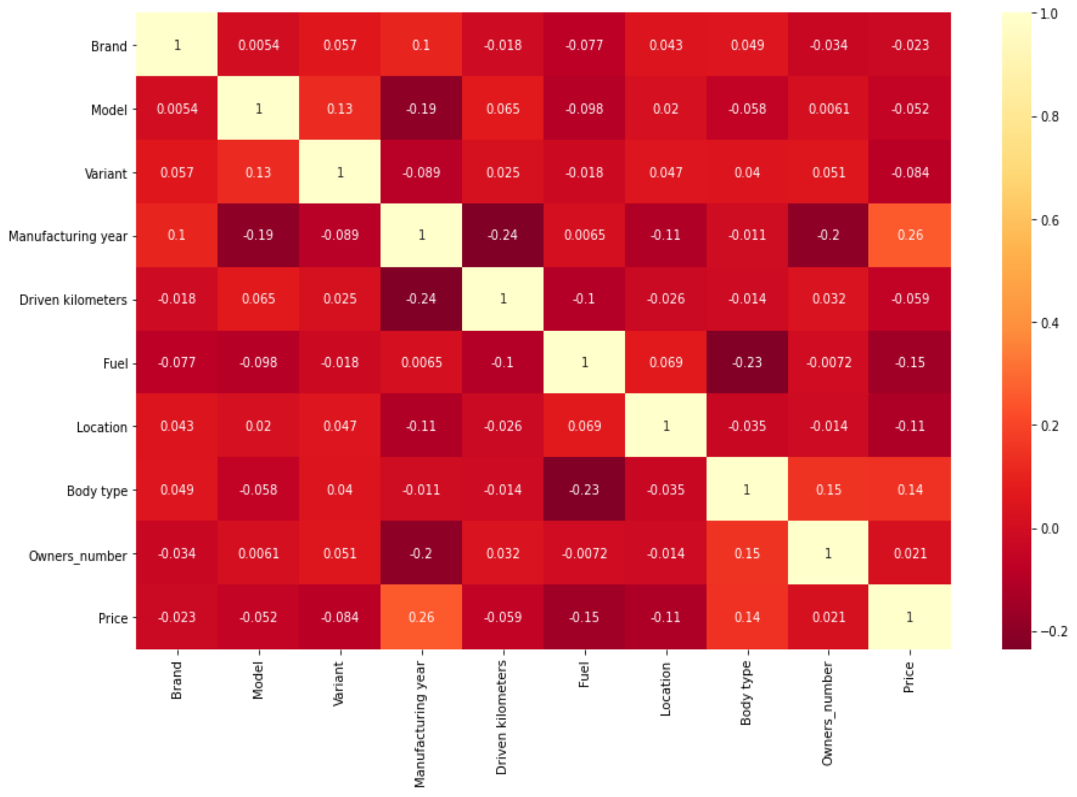
2- Fuel column is most negatively correlated with Price column.

Correlation Matrix: - A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses. A correlation matrix consists of rows and columns that show the variables.



	Brand	Model	Variant	Manufacturing year	Driven kilometers	Fuel	Location	Body type	Owners_number	Price
Brand	1.000000	0.005401	0.057215	0.104432	-0.017520	-0.076994	0.043299	0.049310	-0.033790	-0.023400
Model	0.005401	1.000000	0.126567	-0.192331	0.064820	-0.098379	0.020110	-0.058357	0.006076	-0.052132
Variant	0.057215	0.126567	1.000000	-0.089307	0.024724	-0.018420	0.046943	0.040371	0.051009	-0.083564
Manufacturing year	0.104432	-0.192331	-0.089307	1.000000	-0.235229	0.006530	-0.112134	-0.010919	-0.195449	0.255750
Driven kilometers	-0.017520	0.064820	0.024724	-0.235229	1.000000	-0.103265	-0.025936	-0.014022	0.032226	-0.059499
Fuel	-0.076994	-0.098379	-0.018420	0.006530	-0.103265	1.000000	0.068707	-0.225790	-0.007191	-0.152984
Location	0.043299	0.020110	0.046943	-0.112134	-0.025936	0.068707	1.000000	-0.034826	-0.013832	-0.114887
Body type	0.049310	-0.058357	0.040371	-0.010919	-0.014022	-0.225790	-0.034826	1.000000	0.150454	0.143797
Owners_number	-0.033790	0.006076	0.051009	-0.195449	0.032226	-0.007191	-0.013832	0.150454	1.000000	0.020515
Price	-0.023400	-0.052132	-0.083564	0.255750	-0.059499	-0.152984	-0.114887	0.143797	0.020515	1.000000

Checking correlation using Heatmap with annotations: -



Observations: -

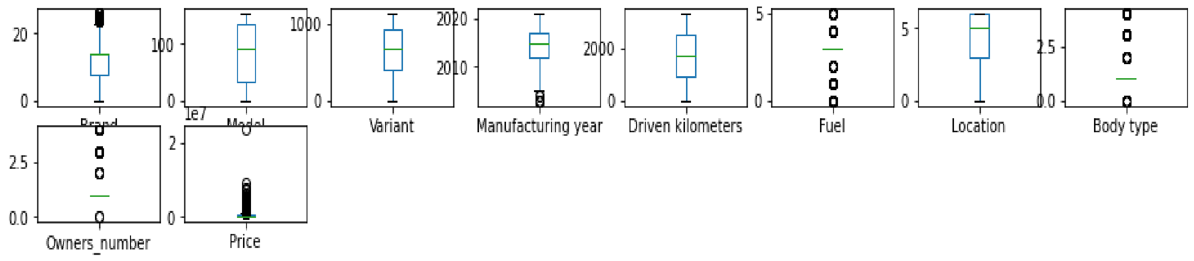
1- Price is highly correlated with Manufacturing Year & Body type columns.

2- Price is negatively correlated with Fuel.

Plotting Outliers: - An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. Outliers should be investigated carefully. So, we will use graphical techniques Box Plot for identifying outliers.

Box Plot- The box plot is a useful graphical display for describing the behaviour of the data in the middle as well as at the ends of the distributions. The box plot uses the median and the lower and upper quartiles (defined as the 25th and 75th percentiles). A box plot is constructed by drawing a box between the upper and lower quartiles with a solid line drawn across the box to locate the median.

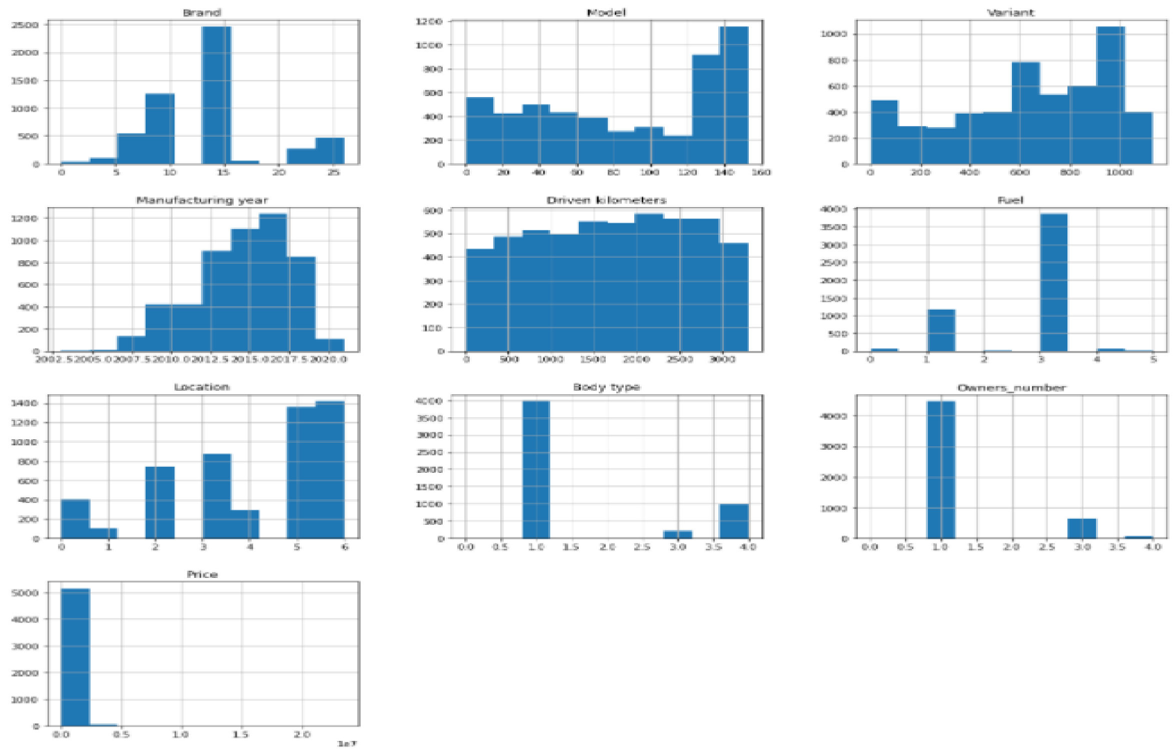
Univariate Analysis: - Univariate involves the analysis of a single variable. First, we are going to do univariate analysis using Box Plot method.



Observation:

Outliers are present in various columns.

Histogram: - we are creating histograms to get broader idea of the distribution.

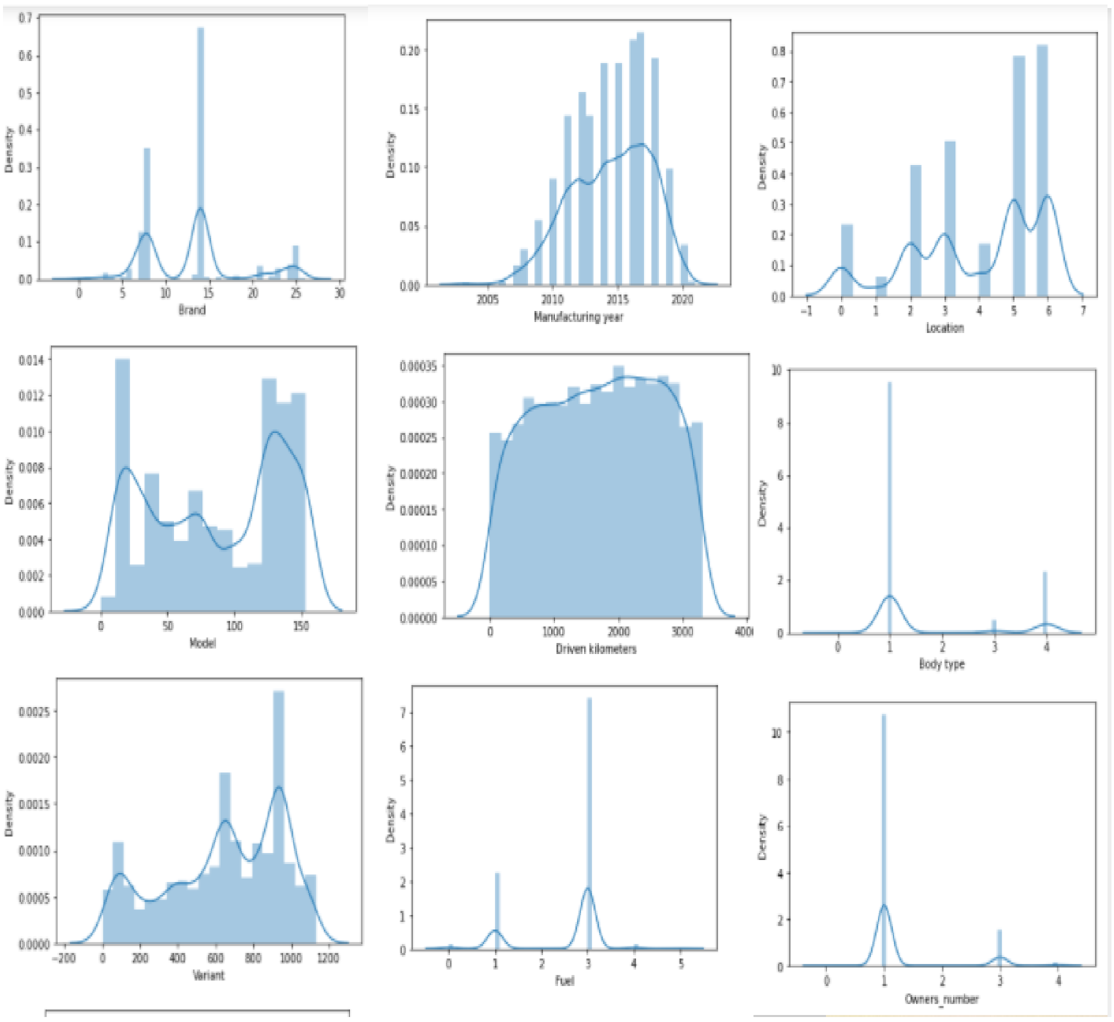


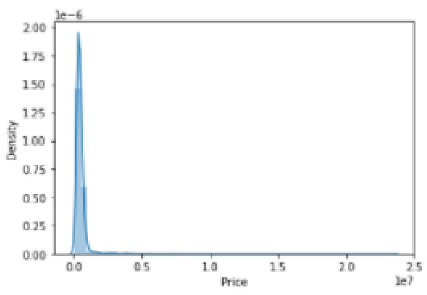
Observation:

Presence of unusual values in above histograms & also distribution is not normal in some columns and these things denote the possibility of potential outliers.

Skewness: - Skewness is a measure of asymmetry or distortion of symmetric distribution. It measures the deviation of the given distribution of a random variable from a symmetric distribution, such as normal distribution. A normal distribution is without any skewness, as it is symmetrical on both sides. Hence, a curve is regarded as skewed if it is shifted towards the right or the left. Skewness is of 2 types: - 1- Positive Skewness 2- Negative Skewness.

Distplot to check Distribution of Skewness: - Distplot plots a univariate distribution of observations. The distplot () function combines the matplotlib hist function with the seaborn kde plot () and rug plot () functions. For individual columns we are using Distplot.

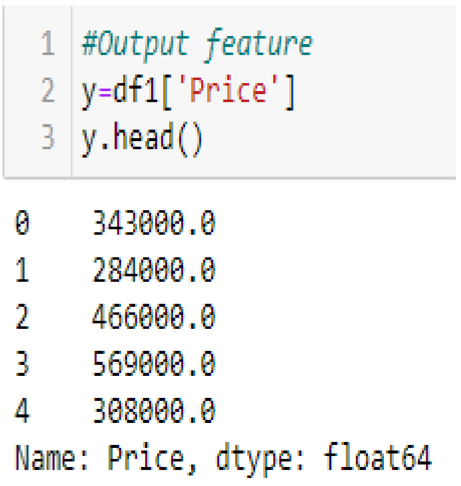




Observation: Skewness is present in various columns. So, we have removed most of skewness using NumPy mathematical functions cube root, log & square root transform.

Data Inputs- Logic- Output Relationships

Output Feature: - In this project Target Variable = dependent variable = y and shape of our dependent variable is (5052,1). The head of output feature is as: -



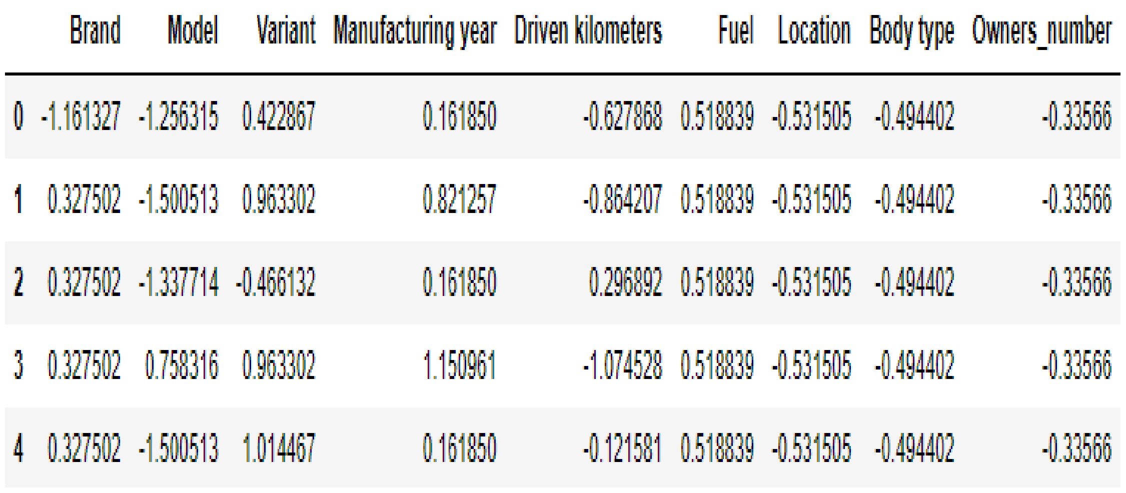
```
#Output feature
y=df1['Price']
y.head()
```

```
0    343000.0
1    284000.0
2    466000.0
3    569000.0
4    308000.0
Name: Price, dtype: float64
```

Output Variable “Price" is continuous in nature so it is a Regression based problem and We have to predict the price of cars with available independent variables.

Input Feature: - The Independent variable in this project = feature vector = x and shape of Independent   Variable is 5052 rows × 9 columns. After certain transformations and analysis input feature is as: -

1- Data Pre-processing: - Using Standard Scaler’s fit _ transform method we have tried to bring input features(x) to common scale and modified the input feature as x1. The shap e was remained same and head of input feature is as: -

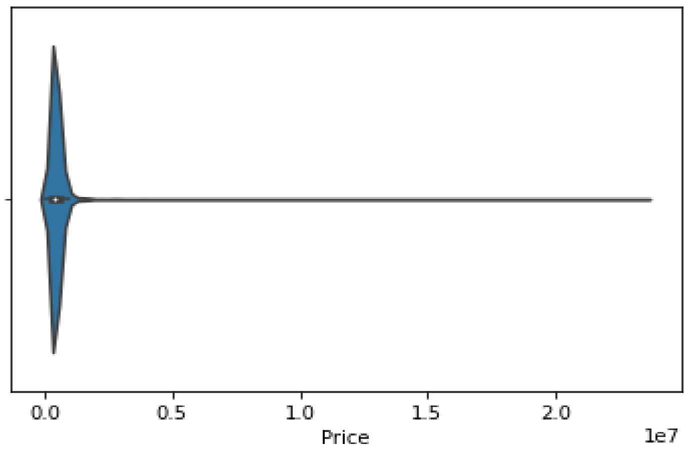


	Brand	Model	Variant	Manufacturing year	Driven kilometers	Fuel	Location	Body type	Owners_number
0	-1.161327	-1.256315	0.422867	0.161850	-0.627868	0.518839	-0.531505	-0.494402	-0.33566
1	0.327502	-1.500513	0.963302	0.821257	-0.864207	0.518839	-0.531505	-0.494402	-0.33566
2	0.327502	-1.337714	-0.466132	0.161850	0.296892	0.518839	-0.531505	-0.494402	-0.33566
3	0.327502	0.758316	0.963302	1.150961	-1.074528	0.518839	-0.531505	-0.494402	-0.33566
4	0.327502	-1.500513	1.014467	0.161850	-0.121581	0.518839	-0.531505	-0.494402	-0.33566

Correlation between input and output features: -In above steps, we have checked already about correlation between input variables and output variable "Price". As per observations “Manufacturing Year” column is most positively correlated with output and “ Fuel” column is most negatively correlated with output feature.

Data Visualization: Data visualization is the graphical representation of information and data. Different visualizations for our input and output features are as: -

Exploring Output Feature: - Violin plot for our target variable & value counts are as: -



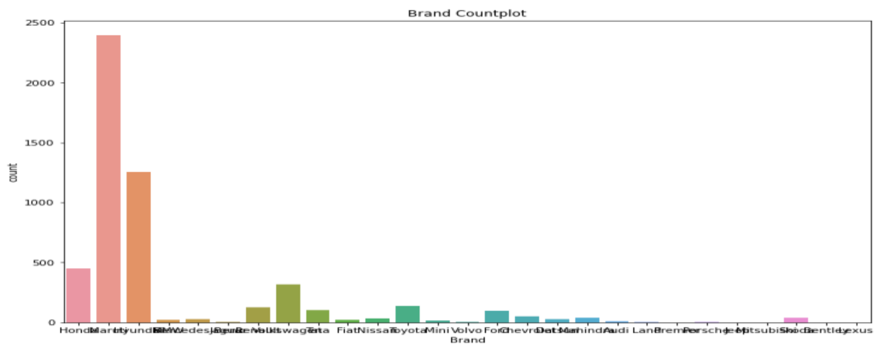
```
275000.0    54
425000.0    53
350000.0    51
325000.0    49
450000.0    49
            ..
449699.0     1
366399.0     1
528000.0     1
268099.0     1
292899.0     1
Name: Price
```

Observation:

Maximum number of car Prices range between 275000-450000.

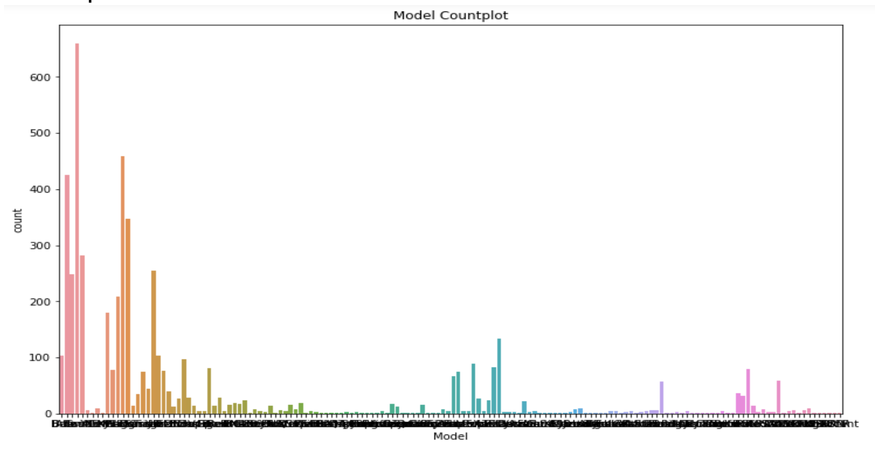
Exploring Input Features: - Plots for different input variables are as: -

1- Count plot for " Brand" feature: -



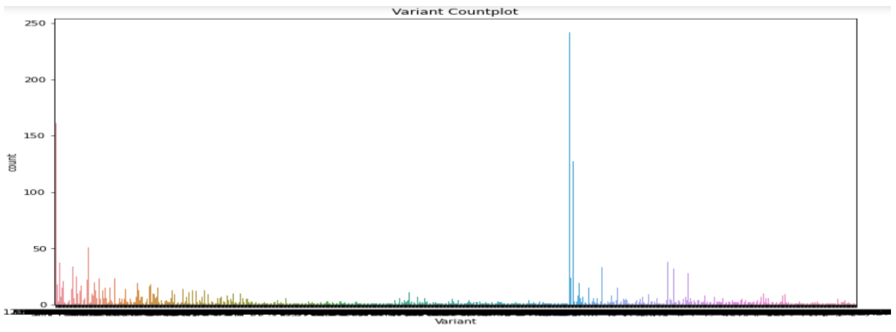
Observation: Maruti & Hyundai are most popular brands.

2- Count plot for " Model" feature: -



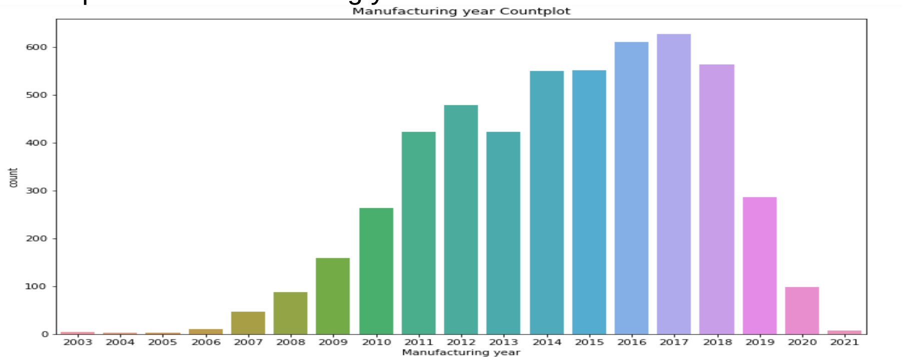
Observation: Swift, Wagon & Alto are most used models.

3- Count plot for “Variant" feature: -



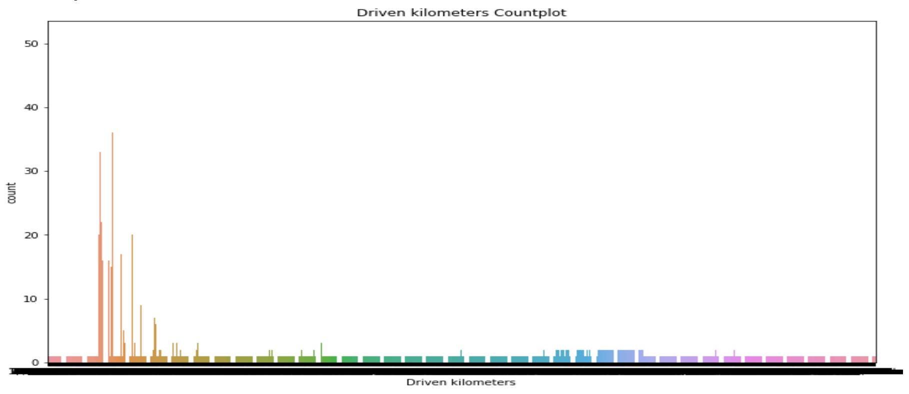
Observation: VXI and LXI variants are mostly used.

4- Count plot for “Manufacturing year" feature: -



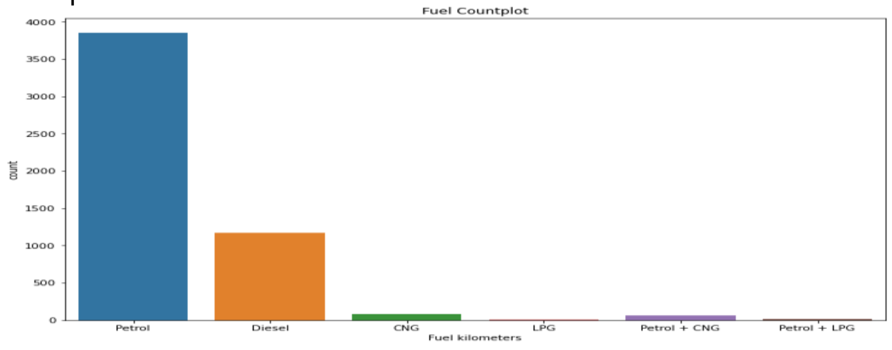
Observation: Highest used cars were manufactured in 2016 & 2017 respectively.

5- Count plot for “Driven kilometres" feature: -



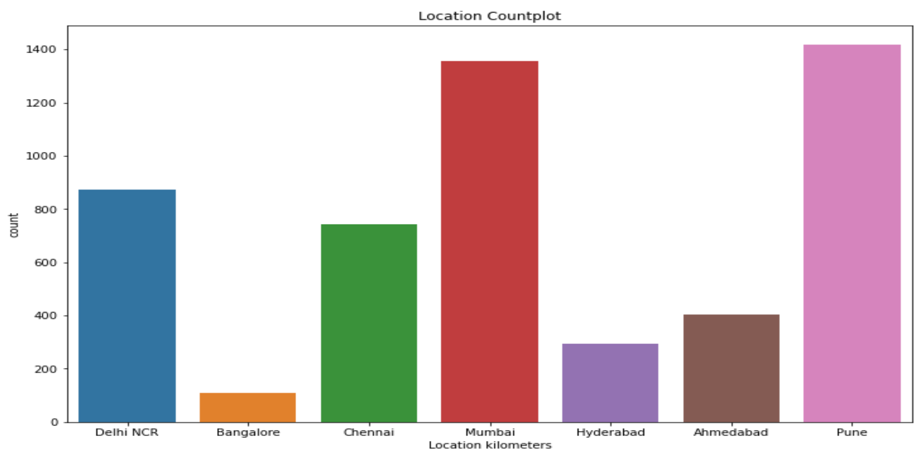
Observation: Maximum Travelled Distance of used cars is between 35000-60000 kms.

6- Count plot for “Fuel" feature: -



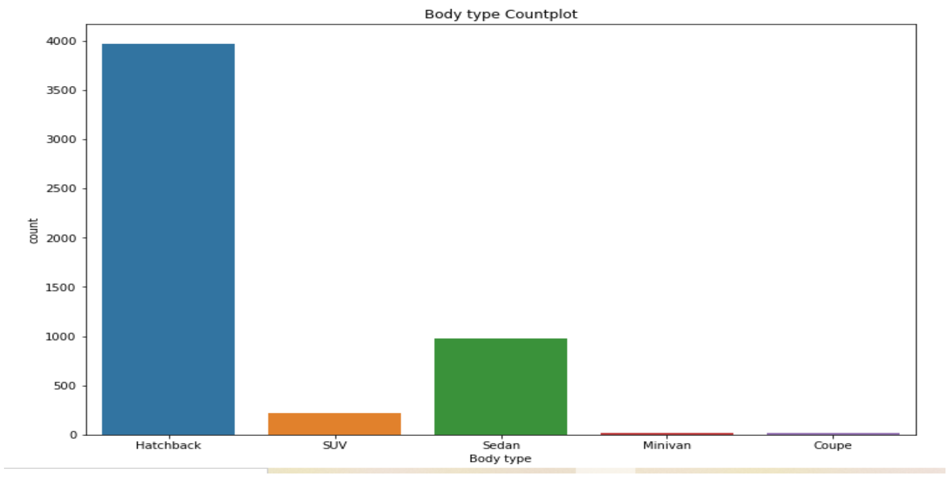
Observation: Maximum cars are using Petrol Fuel.

7- Count plot for “Location" feature: -



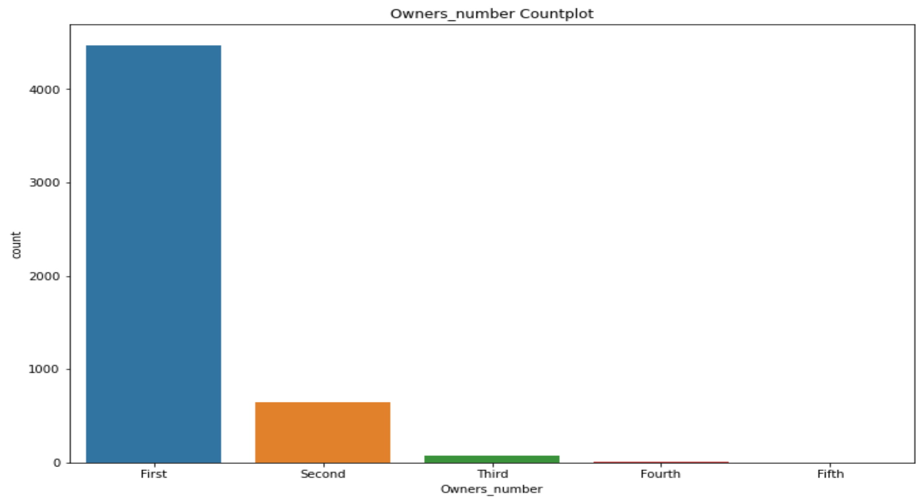
Observation: Mostly used cars are available for sale in Pune & Mumbai.

8- Count plot for “Body Type" feature: -



Observation: Maximum number of used cars have Hatchback body.

9- Count plot for “Owners_ number" feature: -

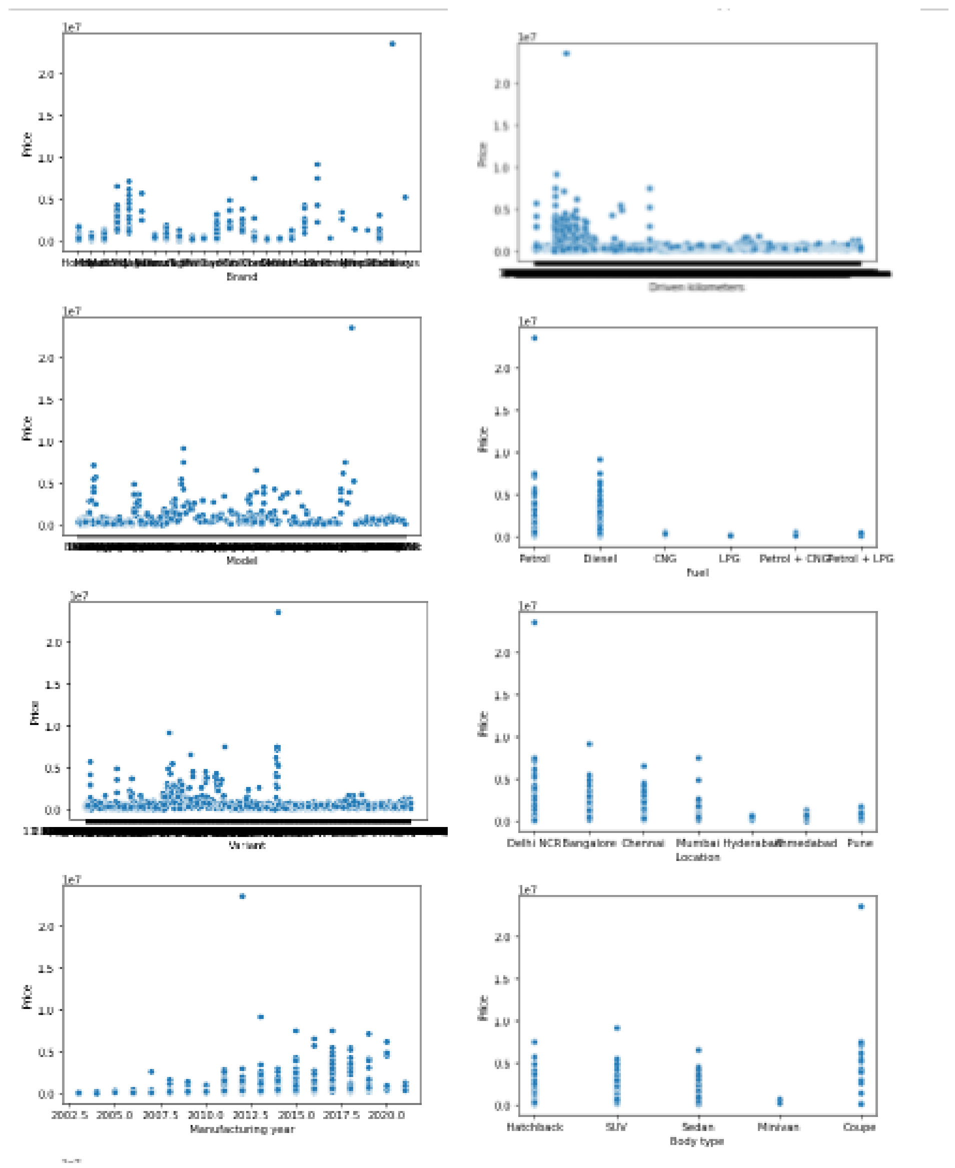


Observation: Maximum cars are owned by First owner. Also, when number of owners are high then it will become hard to sell those cars.

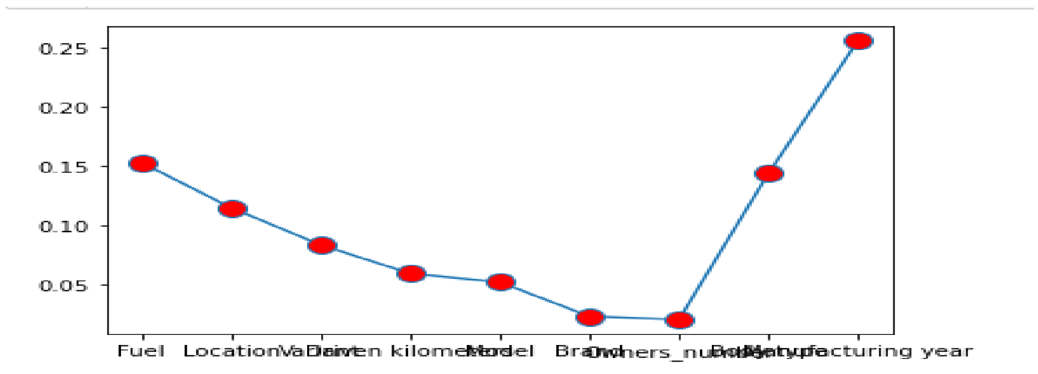
Bivariate Analysis: - It involves the analysis of two variables, for the purpose of determining the empirical relationship between them. We are using scatterplot for this purpose.

Scatter Plot- Scatter plot reveals relationships or association between two variables.

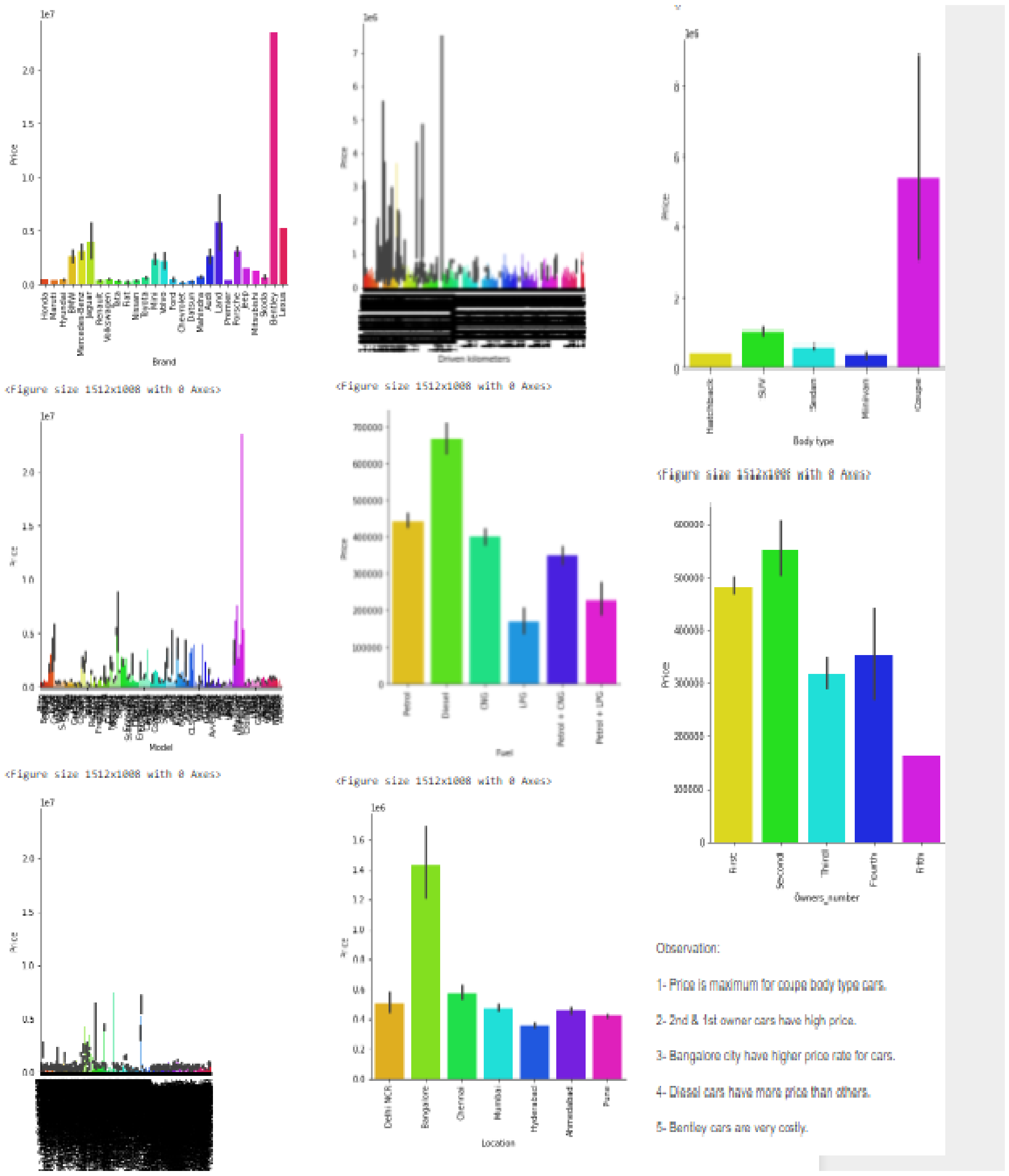
Scatter Plot between Output and Input Features: -



Marker Plot for checking highly correlated values with Output variable Price: -



Cat Plot between Output and Input Features (Categorical Type): -



```
<Figure size 1512x1008 with 0 Axes>
<Figure size 1512x1008 with 0 Axes>
<Figure size 1512x1008 with 0 Axes>
<Figure size 1512x1008 with 0 Axes>
<Figure size 1512x1008 with 0 Axes>
```

Observation:

1- Price is maximum for coupe body type cars.

2- 2nd & 1st owner cars have high price.

3- Bangalore city have higher price rate for cars.

4- Diesel cars have more price than others.

5- Bentley cars are very costly.

Observations:

1- Price is maximum for coupe body type cars.

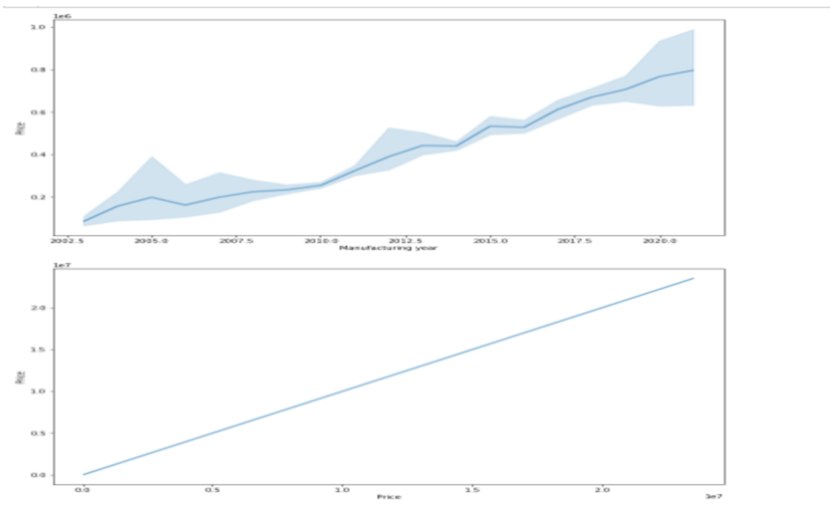
2- 2nd & 1st owner cars have high price.

3- Bangalore city have higher price rate for cars.

4- Diesel cars have more price than others.

5- Bentley cars are very costly.

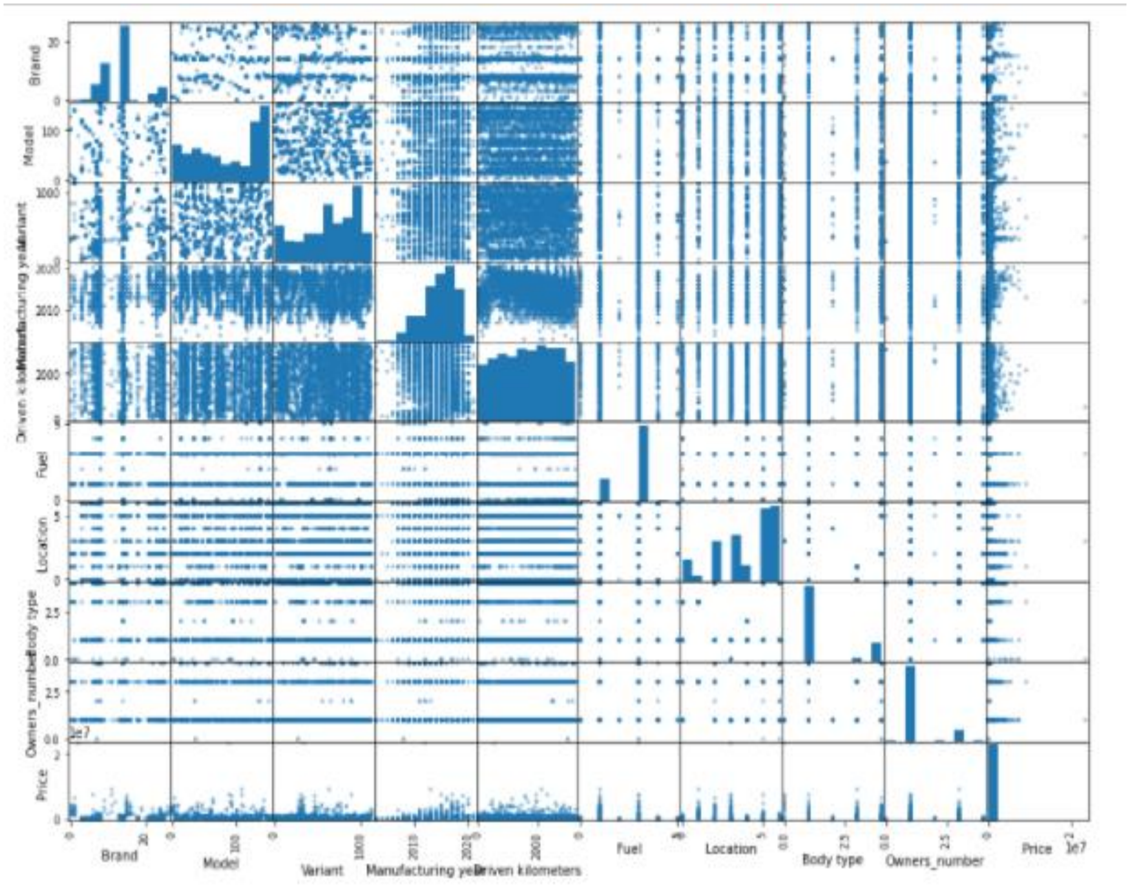
Line Plot between Output and Input Features (Continuous Type): -



Observation: Car prices are increasing year by year.

Multivariate Analysis: - Multivariate analysis is used to study more complex sets of data. It is a statistical method that measures relationships between two or more response variables.

Scatter plot matrix: -Scatter plot matrix is a grid (or matrix) of scatter plots used to visualize bivariate relationships between combinations of variables.



Observation: Using multivariate analysis, we can look at interactions between variables. Scatter plots of all pair of attributes helps us to spot structured relationship between input variables.

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

- By looking into the target variable "Price", we assume that this project is a Regression based problem as target variable is continuous in nature.
- We have removed irrelevant column which does not have more impact on dataset.
- We had found outliers in dataset so tried to remove them using Z scores and data loss was nearly 2.75%.
- We have tried to remove skewness using transformation methods (sqrt, log, cube root).

Hardware and Software Requirements and Tools Used

Hardware:4GB RAM, Intel I3 Processor.
System Software: 64Bit O/S Windows 10(x64-based processor)

Software Tools: Software Tools used in this project are as: -

1- Anaconda3 (64-bit)
2- Jupyter Notebook 6.1.4
3- Python 3.8
4- MS-Office 2019 (Excel, Word, Power point)
5- Notepad
6- Google Chrome Web Browser
7- Selenium Driver

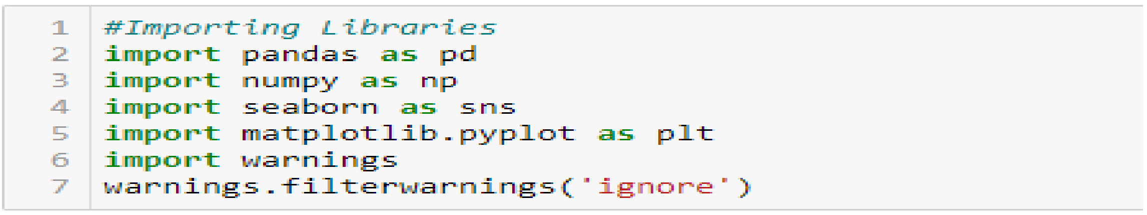
Libraries & Packages used:

We have used Python and Jupyter Notebook to compute the majority of this project. For analysis, visualization, statistics, machine learning & evaluation, we have used these: -

1- Pandas (data analysis)
2- NumPy (matrix computation)
3- Matplotlib (Visualization)
4- Seaborn (visualization)
5- Scikit-Learn (Machine Learning)
6- SciPy (Z-score)
7- Selenium Web driver & Exceptions
8- Warnings (filter warnings) & etc. Microsoft Excel (for calculations and Data Handling).

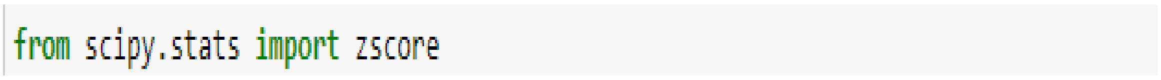
Packages and libraries used in this project are: -

1- For Data Analysis & Visualization: -



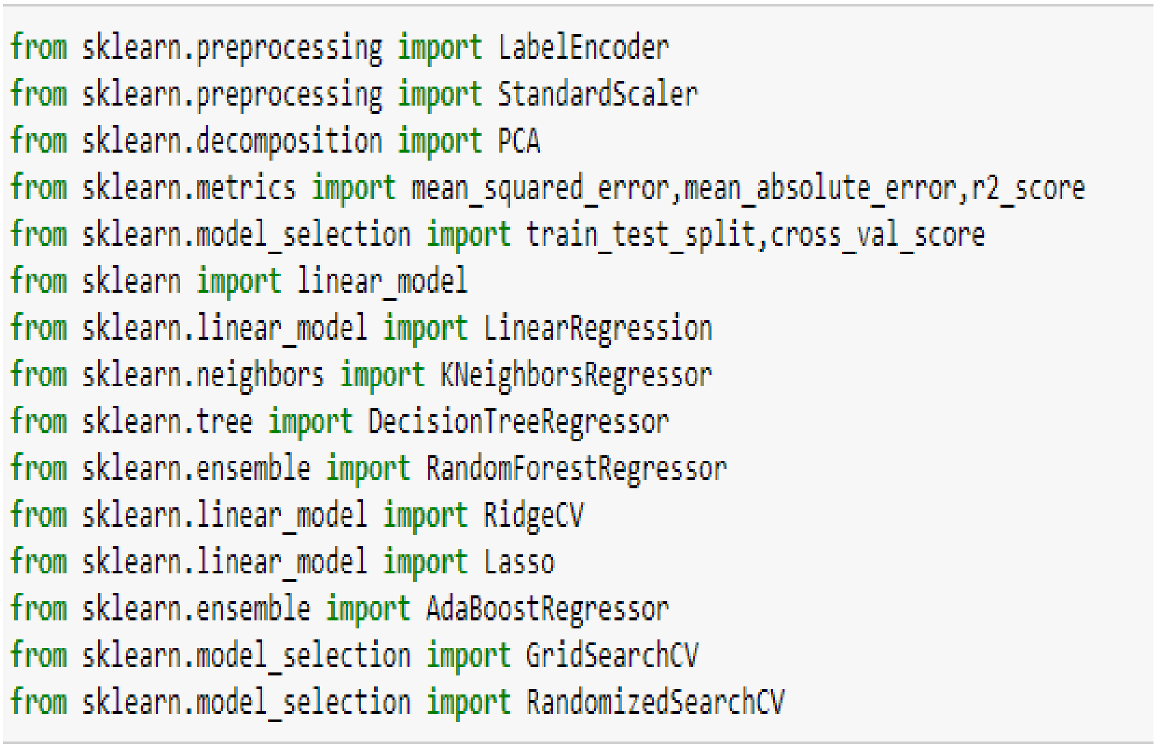
```
#Importing Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

2- For Z score: -



```python
from scipy.stats import zscore
```

3- From Scikit-Learn Library: -

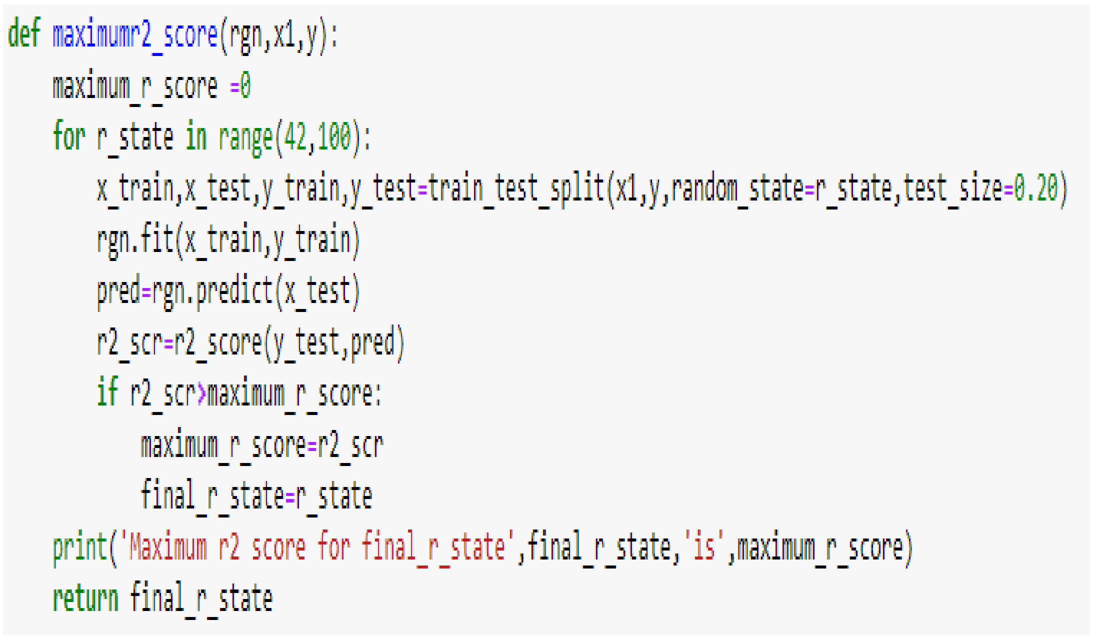


```python
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import Lasso
from sklearn.ensemble import AdaBoostRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

4- For saving the final model: -

```python
import joblib
```

Function to calculate maximum R2 score at best random state: -



```python
def maximumr2_score(rgn,x1,y):
    maximum_r_score =0
    for r_state in range(42,100):
        x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=r_state,test_size=0.20)
        rgn.fit(x_train,y_train)
        pred=rgn.predict(x_test)
        r2_scr=r2_score(y_test,pred)
        if r2_scr>maximum_r_score:
            maximum_r_score=r2_scr
            final_r_state=r_state
    print('Maximum r2 score for final_r_state',final_r_state,'is',maximum_r_score)
    return final_r_state
```

Model/s Development and Evaluation

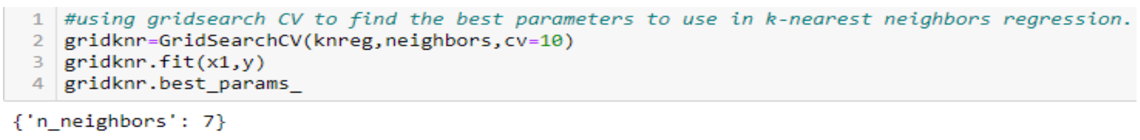
- **Identification of possible problem-solving approaches (methods)**

- As per visualizations, Target Variable "Price" is maximum between 275000-450000.Also, it is continuous in nature so we will use different regression algorithms to try and find the features that have the best explanation of the target variable.
- Explored input features and their values using count plot.
- Checking missing values in dataset.
- Checking Summary Statistics to summarize set of observations as- central Tendency, dispersion, skewness, variance, range, deviation etc.
- Checking Correlation between target variable and input features using Correlation Matrix & correlation Heatmap using Seaborn.
- To check distribution & spread of data we used Histogram.
- Checked Scatter Plots between input & output feature for bivariate analysis.
- To check highly correlated values with Output variable Price used Marker Plot.
- Divided input features into category type & continuous type features.
- Checked Cat plot for category type & Line plot for continuous type features.
- Used Scatter matrix for multivariate analysis.
- Removed irrelevant columns which does not have more impact on dataset.
- Used Label Encoding to encode categorical data in to numerical format.
- Used Boxplot for summarizing variations & check outliers.
- Removed outliers using Z scores method and data loss was only 2.75%.
- Divided dataset into input and output sets to explore more briefly.
- Used Distplot to check distribution of skewness and removed skewness using NumPy mathematical functions log, cube root & square root transformations.
- Standardization is useful to speed up the learning algorithm and it rescales the features so that they will have the properties of the standard normal distribution with $\mu = 0$ and $\sigma = 1$.
- We have used Standard Scaler to standardize the data.
- We will use Coefficient of determination(R2) score as our metric.
- After Splitting data in to Training & Test Sets, checked scores at best random state after applying different regression algorithms.
- Used Cross validation to check how accurately a predictive model will perform in practice.
- To check error of forecasting model, we used Error Metric (MAE, MSE, RMSE).
- To choose set of optimal hyperparameters for learning algorithm, we did Hyperparameter Tuning. We used Grid Search CV to find estimators/neighbors/alpha for learning algorithms and Randomized search CV to find best parameters for final model.
- AdaBoost is used as ensemble method to combine several machine learning techniques into one predictive model in order to decrease variance, bias & improve predictions.
- Compared all algorithms on basis of scores, plots and errors & finalized the best model.
- Implementing the best model, calculating scores/errors & also checking predicted values.
- Checked scatterplot between Predicted values and Test values.
- Saved final model using job lib.
- Test Dataset sheet checked, handled missing values, removed irrelevant columns, did feature engineering, standardization, principal component & variance analysis.
- Loading saved model to predict values for Test Dataset.
- In last, we saved predicted values of test dataset in a CSV file.

- ## Testing of Identified Approaches (Algorithms)

1-Linear Regression: - In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. Linear Regression fits a linear model with coefficients w = (w1, …) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

2-k-nearest neighbors: - K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. We have used Grid Search CV method to find n_neighbors.



```
#using gridsearch CV to find the best parameters to use in k-nearest neighbors regression.
gridknr=GridSearchCV(knreg,neighbors,cv=10)
gridknr.fit(x1,y)
gridknr.best_params_
```

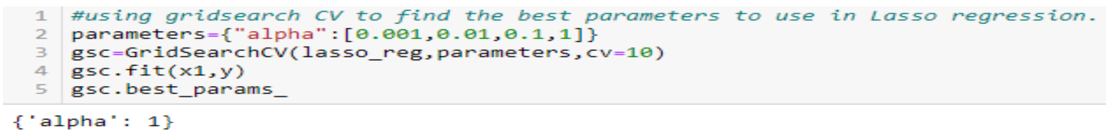
```
{'n_neighbors': 7}
```

3-Decision Tree Regression: - Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output which means that the output is not discrete.

4- **Random Forest Regression**: - 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. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

5- **Ridge Regression**: - Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where independent variables are highly correlated. This method performs L2 regularization. It reduces the model complexity by coefficient shrinkage.

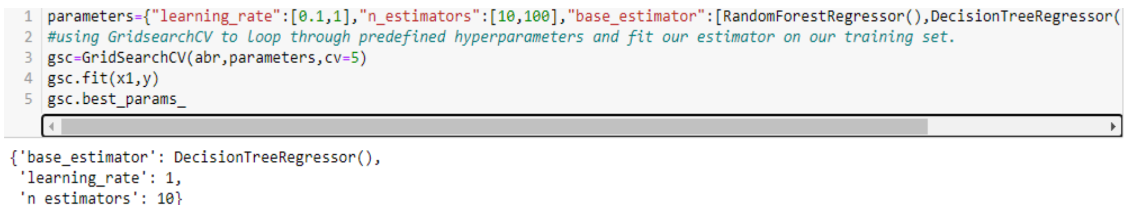
6- **Lasso Regression**: - Lasso (Least Absolute Shrinkage and Selection Operator) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. We have used Grid Search CV to find best parameters for Lasso regression.



```
#using gridsearch CV to find the best parameters to use in Lasso regression.
parameters={"alpha":[0.001,0.01,0.1,1]}
gsc=GridSearchCV(lasso_reg,parameters,cv=10)
gsc.fit(x1,y)
gsc.best_params_
```

```
{'alpha': 1}
```

7-AdaBoost Regression: - 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. We have used Grid Search CV to find best parameters.



```
parameters={"learning_rate":[0.1,1],"n_estimators":[10,100],"base_estimator":[RandomForestRegressor(),DecisionTreeRegressor(
#using GridsearchCV to loop through predefined hyperparameters and fit our estimator on our training set.
gsc=GridSearchCV(abr,parameters,cv=5)
gsc.fit(x1,y)
gsc.best_params_
```

```
{'base_estimator': DecisionTreeRegressor(),
 'learning_rate': 1,
 'n estimators': 10}
```

- ## Run and evaluate selected models

Selected Models: -

1- Linear Regression ()

2- K Neighbors Regressor (n_neighbors=7)

3- Decision Tree Regressor ()

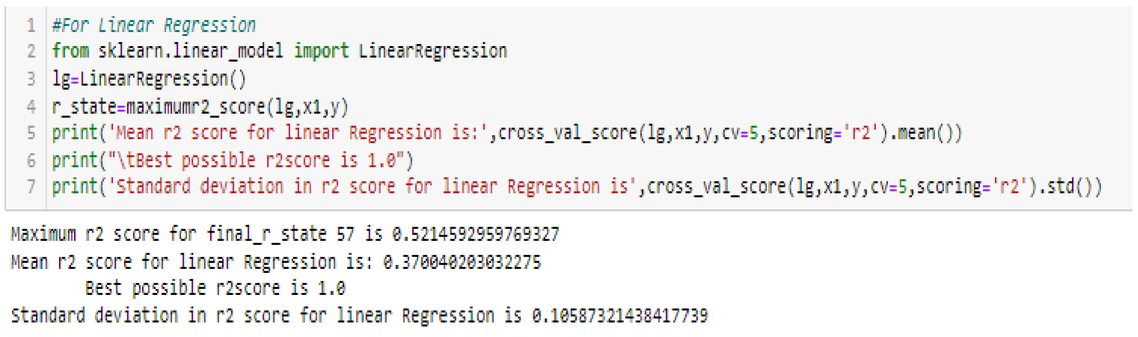
4- Random Forest Regressor ()

5- Ridge CV ()

6- Lasso(alpha=1)

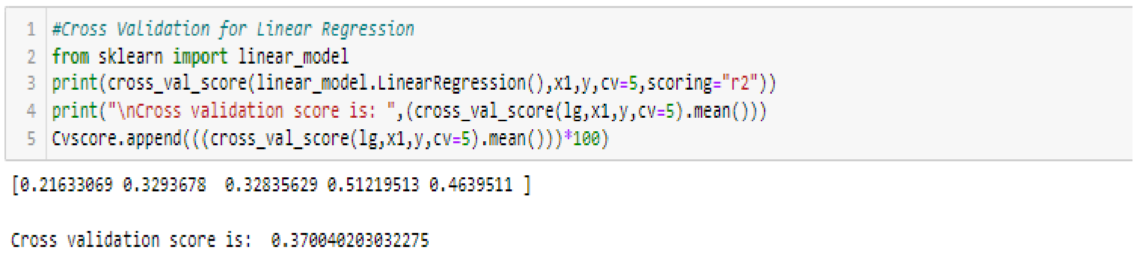
7-Ada Boost Regressor (base estimator=Decision Tree Regressor (), learning_ rate=1, n_ estimators=10)

Linear Regression: -



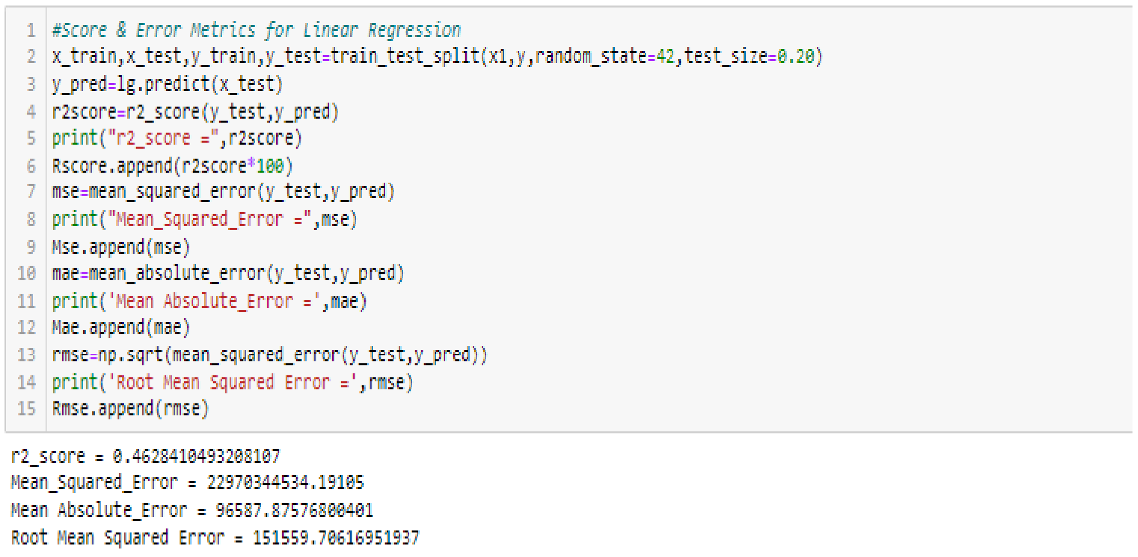
```
#For Linear Regression
from sklearn.linear_model import LinearRegression
lg=LinearRegression()
r_state=maximumr2_score(lg,x1,y)
print('Mean r2 score for linear Regression is:',cross_val_score(lg,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for linear Regression is',cross_val_score(lg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 57 is 0.5214592959769327
Mean r2 score for linear Regression is: 0.370040203032275
        Best possible r2score is 1.0
Standard deviation in r2 score for linear Regression is 0.10587321438417739
```



```
#Cross Validation for Linear Regression
from sklearn import linear_model
print(cross_val_score(linear_model.LinearRegression(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(lg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(lg,x1,y,cv=5).mean()))*100)
```

```
[0.21633069 0.3293678  0.32835629 0.51219513 0.4639511 ]

Cross validation score is:  0.370040203032275
```



```
#Score & Error Metrics for Linear Regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=42,test_size=0.20)
y_pred=lg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.4628410493208107
Mean_Squared_Error = 22970344534.19105
Mean Absolute_Error = 96587.87576800401
Root Mean Squared Error = 151559.70616951937
```

k-nearest neighbors: -

```
#r2 scores for k-nearest neighbors regression.
knreg=KNeighborsRegressor(n_neighbors=7)
r_state=maximumr2_score(knreg,x1,y)
print('Mean r2 score for KNN Regression is:',cross_val_score(knreg,x1,y,cv=5,scoring='r2').mean())
print('standard deviation in r2 score for KNN Regression is:',cross_val_score(knreg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 86 is 0.7333184194962706
Mean r2 score for KNN Regression is: 0.5041918293321668
standard deviation in r2 score for KNN Regression is: 0.1438154525615673
```

```
#Cross Validation for KNeighborsRegressor
print(cross_val_score(KNeighborsRegressor(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(knreg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(knreg,x1,y,cv=5).mean()))*100)
```

```
[0.31349876 0.54624468 0.63181681 0.71138971 0.31543458]

Cross validation score is:  0.5041918293321668
```

```
#Score & Error Metrics for k-nearest neighbors regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=89,test_size=0.20)
y_pred=knreg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.7404165664421474
Mean_Squared_Error = 13630107906.744804
Mean Absolute_Error = 70287.55023314964
Root Mean Squared Error = 116748.05311757796
```

Decision Tree Regression: -

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
r_state=maximumr2_score(dt,x1,y)
print('Mean r2 score for Decision Tree Regression is:',cross_val_score(dt,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for Decision Tree Regression is',cross_val_score(dt,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 67 is 0.8255996821565016
Mean r2 score for Decision Tree Regression is: -0.18284511730285402
	Best possible r2score is 1.0
Standard deviation in r2 score for Decision Tree Regression is 1.323619489373976
```

```
#Cross Validation for Decision Tree regression
print(cross_val_score(DecisionTreeRegressor(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(dt,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(dt,x1,y,cv=5).mean()))*100)
```

```
[ 0.477927    0.63516909 -0.18968685  0.82105894 -2.68239479]

Cross validation score is:  -0.19876947684761026
```

```
#Score & Error Metrics for DecisionTreeRegressor
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=80,test_size=0.20)
y_pred=dt.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.977706021507752
Mean_Squared_Error = 1134178524.4817014
Mean Absolute_Error = 9704.202769535113
Root Mean Squared Error = 33677.56708079878
```

Random Forest Regression: -

```python
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor()
r_state=maximumr2_score(rf,x1,y)
print('Mean r2 score for Random Forest Regression is:',cross_val_score(rf,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for Random Forest Regression is',cross_val_score(rf,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 96 is 0.8987740583158399
Mean r2 score for Random Forest Regression is: 0.4766359203874398
        Best possible r2score is 1.0
Standard deviation in r2 score for Random Forest Regression is 0.2540721361781564
```

```python
#Cross Validation for Random Forest Regression
print(cross_val_score(RandomForestRegressor(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(rf,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(rf,x1,y,cv=5).mean()))*100)
```

```
[0.5270276  0.75503176 0.37255165 0.86615837 0.0978547 ]

Cross validation score is:  0.5173508777969934
```

```python
#Score & Error Metrics for RandomForestRegressor
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=60,test_size=0.20)
y_pred=rf.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.9707012573610098
Mean_Squared_Error = 1387238008.940139
Mean Absolute_Error = 21482.434585747258
Root Mean Squared Error = 37245.64416062822
```

Ridge Regression: -

```python
from sklearn.linear_model import RidgeCV
rg=RidgeCV()
r_state=maximumr2_score(rg,x1,y)
print('Mean r2 score for Ridge Regression is:',cross_val_score(rg,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for Ridge Regression is',cross_val_score(rg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 57 is 0.521368769822706
Mean r2 score for Ridge Regression is: 0.3703732387307142
        Best possible r2score is 1.0
Standard deviation in r2 score for Ridge Regression is 0.10602918540672017
```

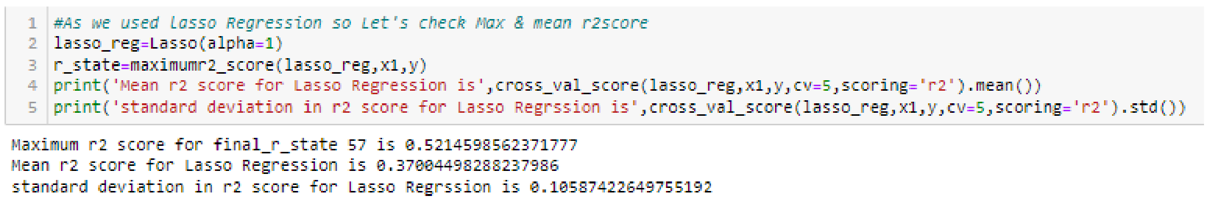
```python
#Cross Validation for Ridge Regression
print(cross_val_score(linear_model.RidgeCV(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(rg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(rg,x1,y,cv=5).mean()))*100)
```

```
[0.21635866 0.32942552 0.32887978 0.5124377  0.46476453]

Cross validation score is:  0.3703732387307142
```

```python
#Score & Error Metrics for Ridge Regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=42,test_size=0.20)
y_pred=rg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

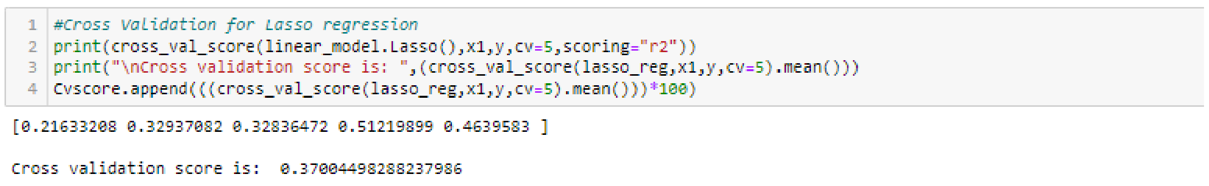
```
r2_score = 0.4628567131046538
Mean_Squared_Error = 22969674709.15857
Mean Absolute_Error = 96582.13128545426
Root Mean Squared Error = 151557.49638060987
```

Lasso regression: -



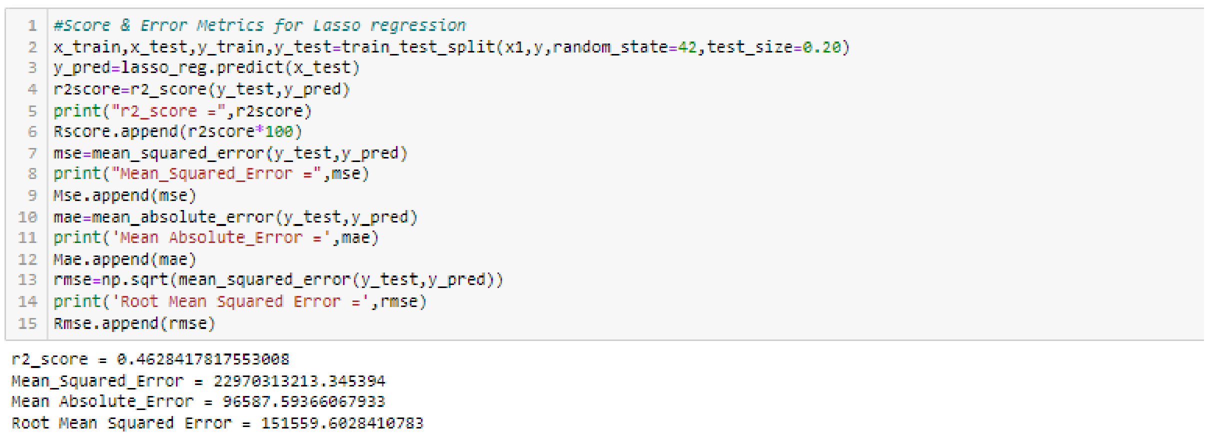
```
#As we used lasso Regression so Let's check Max & mean r2score
lasso_reg=Lasso(alpha=1)
r_state=maximumr2_score(lasso_reg,x1,y)
print('Mean r2 score for Lasso Regression is',cross_val_score(lasso_reg,x1,y,cv=5,scoring='r2').mean())
print('standard deviation in r2 score for Lasso Regrssion is',cross_val_score(lasso_reg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 57 is 0.5214598562371777
Mean r2 score for Lasso Regression is 0.37004498288237986
standard deviation in r2 score for Lasso Regrssion is 0.10587422649755192
```



```
#Cross Validation for Lasso regression
print(cross_val_score(linear_model.Lasso(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(lasso_reg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(lasso_reg,x1,y,cv=5).mean()))*100)
```

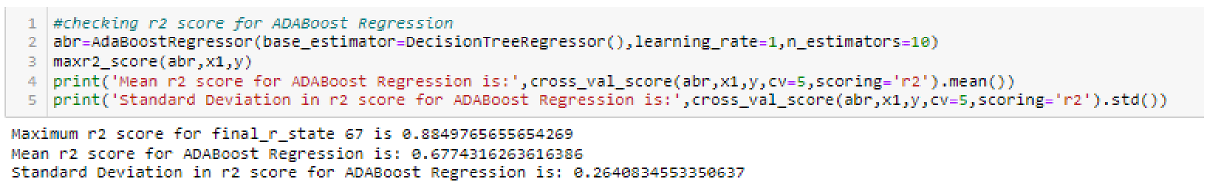
```
[0.21633208 0.32937082 0.32836472 0.51219899 0.4639583 ]

Cross validation score is:  0.37004498288237986
```



```
#Score & Error Metrics for Lasso regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=42,test_size=0.20)
y_pred=lasso_reg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

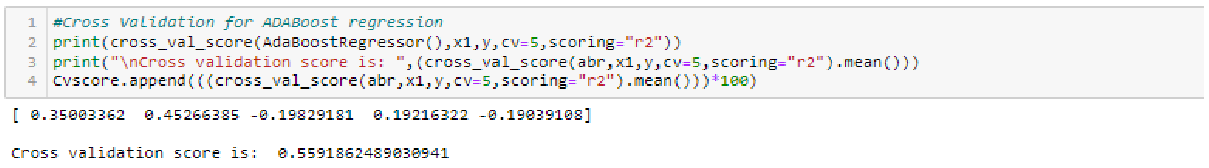
```
r2_score = 0.4628417817553008
Mean_Squared_Error = 22970313213.345394
Mean Absolute_Error = 96587.59366067933
Root Mean Squared Error = 151559.6028410783
```

ADA Boost Regression: -



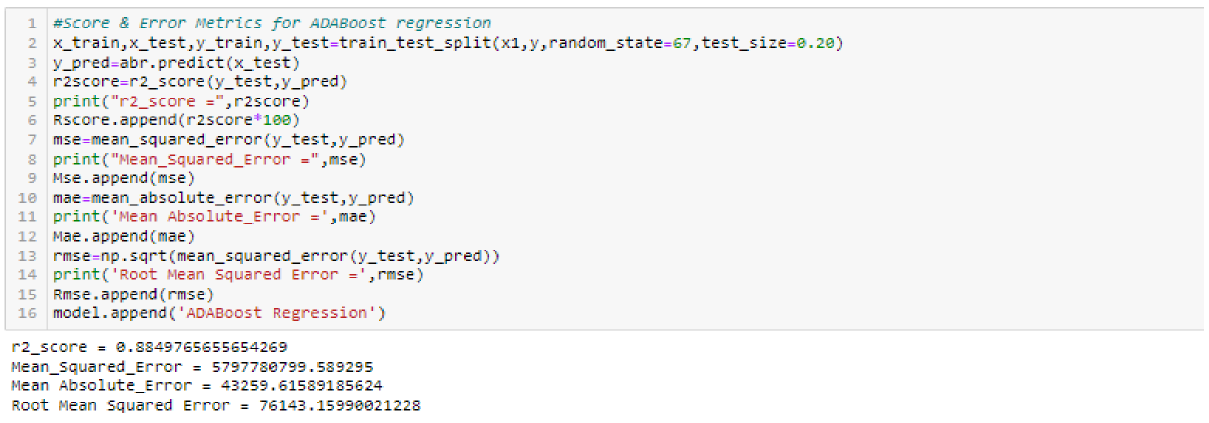
```
#checking r2 score for ADABoost Regression
abr=AdaBoostRegressor(base_estimator=DecisionTreeRegressor(),learning_rate=1,n_estimators=10)
maxr2_score(abr,x1,y)
print('Mean r2 score for ADABoost Regression is:',cross_val_score(abr,x1,y,cv=5,scoring='r2').mean())
print('Standard Deviation in r2 score for ADABoost Regression is:',cross_val_score(abr,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 67 is 0.8849765655654269
Mean r2 score for ADABoost Regression is: 0.6774316263616386
Standard Deviation in r2 score for ADABoost Regression is: 0.2640834553350637
```



```
#Cross Validation for ADABoost regression
print(cross_val_score(AdaBoostRegressor(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(abr,x1,y,cv=5,scoring="r2").mean()))
Cvscore.append(((cross_val_score(abr,x1,y,cv=5,scoring="r2").mean()))*100)
```

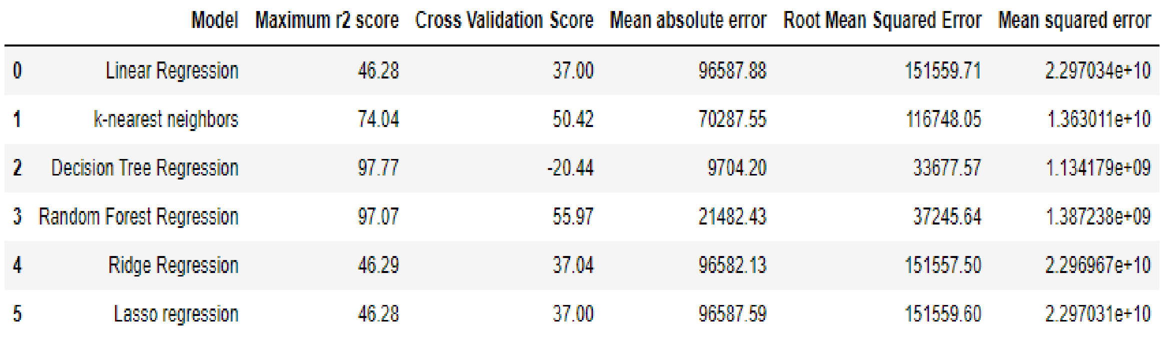
```
[ 0.35003362  0.45266385 -0.19829181  0.19216322 -0.19039108]

Cross validation score is:  0.5591862489030941
```



```
#Score & Error Metrics for ADABoost regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=67,test_size=0.20)
y_pred=abr.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
model.append('ADABoost Regression')
```

```
r2_score = 0.8849765655654269
Mean_Squared_Error = 5797780799.589295
Mean Absolute_Error = 43259.61589185624
Root Mean Squared Error = 76143.15990021228
```

- **Key Metrics for success in solving problem under consideration: -**



	Model	Maximum r2 score	Cross Validation Score	Mean absolute error	Root Mean Squared Error	Mean squared error
0	Linear Regression	46.28	37.00	96587.88	151559.71	2.297034e+10
1	k-nearest neighbors	74.04	50.42	70287.55	116748.05	1.363011e+10
2	Decision Tree Regression	97.77	-20.44	9704.20	33677.57	1.134179e+09
3	Random Forest Regression	97.07	55.97	21482.43	37245.64	1.387238e+09
4	Ridge Regression	46.29	37.04	96582.13	151557.50	2.296967e+10
5	Lasso regression	46.28	37.00	96587.59	151559.60	2.297031e+10

Evaluation Metrics used: -

1- **r2 score**: - Coefficient of determination, denoted R2 or r2, is the proportion variation in the dependent variable that is predictable from the independent variables. It is a regression score function and best possible score is 1.0. We have calculated maximum & mean r2 score for models and will select model with best r2 score.

2-**Cross validation score**: - It is a score evaluated by cross-validation. When we want to estimate how accurately a predictive model will perform in practice and our goal is prediction then we use cross validation. cross validation score returns score of test fold where cross validation predicts returns predicted y values for the test fold. We will select model with best cross validation score.

Error Metrics used: - We will select model with minimum errors.

1-Mean Absolute Error (MAE)- MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. MAE score is calculated as the average of the absolute error values. MAE can be calculated as follows:

MAE = 1 / N * sum for i to N abs(y_i – yhat_i)

2-Mean Squared Error (MSE): - MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset. MSE can be calculated as follows:

MSE = 1 / N * sum for i to N (y_i – yhat_i)^2

3-Root Mean Squared Error (RMSE): - RMSE is an extension of the mean squared error. It’s the square root of the average of squared differences between prediction and actual observation. RMSE can be calculated as follows:

RMSE = sqrt(1 / N * sum for i to N (y_i – yhat_i)^2)

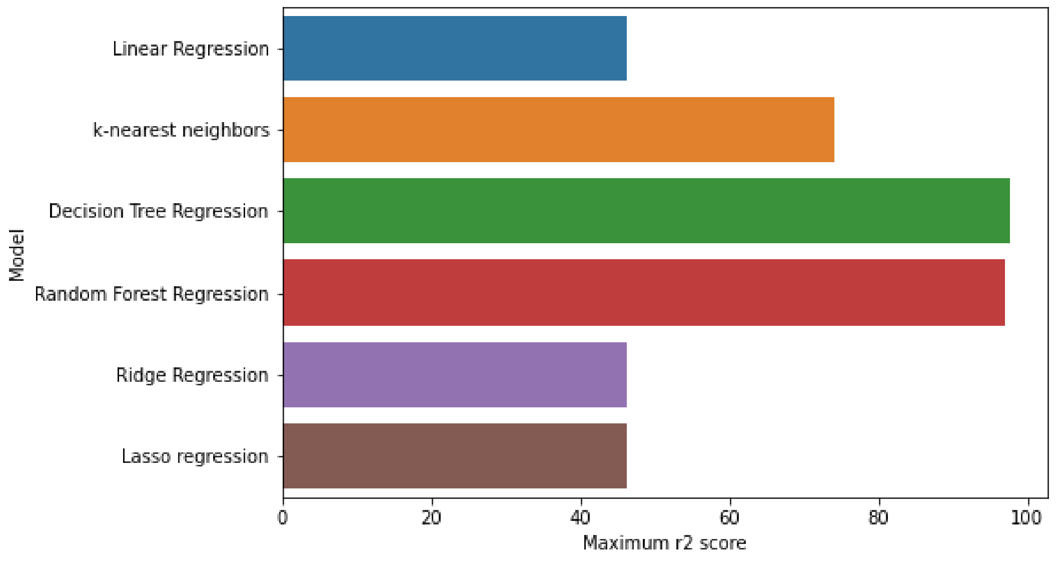
Hyperparameter Tuning: - Hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. We used Grid Search CV to find estimators /neighbors/alpha for learning algorithms and Randomized search CV to find best parameters for implementation of final selected model.

- ## Visualizations (machine learning): -

All visualizations done before applying machine learning algorithms are shown above. We have done visualizations for exploring output variable, input features, relationship & correlation between input and output features, to check outliers, skewness & missing values, PCA analysis, etc. Here we are visualizing different regression model's performances, scores after applying ensemble methods & Hyperparameter Tuning, Final model selection, relation between true & predicted values & test dataset.

1- **Before using ensemble methods**: - After applying different regression algorithms on model and using cross validation we have obtained different coefficient of determination & cross validation scores and after using Error metrics we have calculated MAE, MSE & RMSE for all models. All scores & errors are shown above. Now we are seeing visualization using Bar Plot of r2 scores: -

```
<AxesSubplot:xlabel='Maximum r2 score', ylabel='Model'>
```



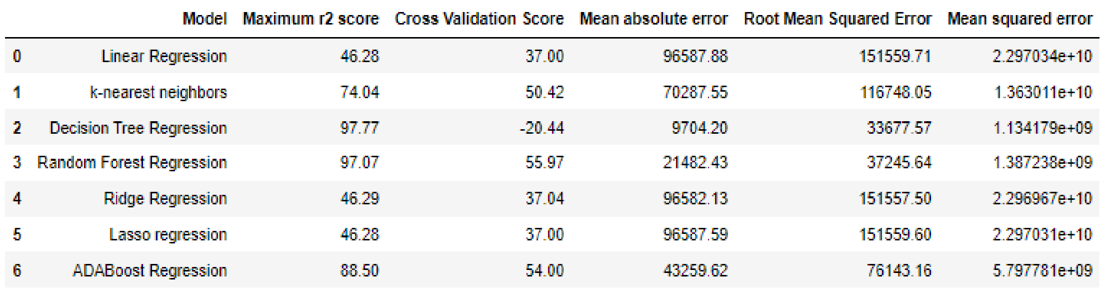
Observations: After comparing above 6 models, these 3 models are good: -

1- Random Forest Regression (R2 & Cross validation scores are higher & RMSE is minimum & other errors are less.)

2- Decision Tree Regression (R2 score is good & MAE is minimum & also other errors are less.)

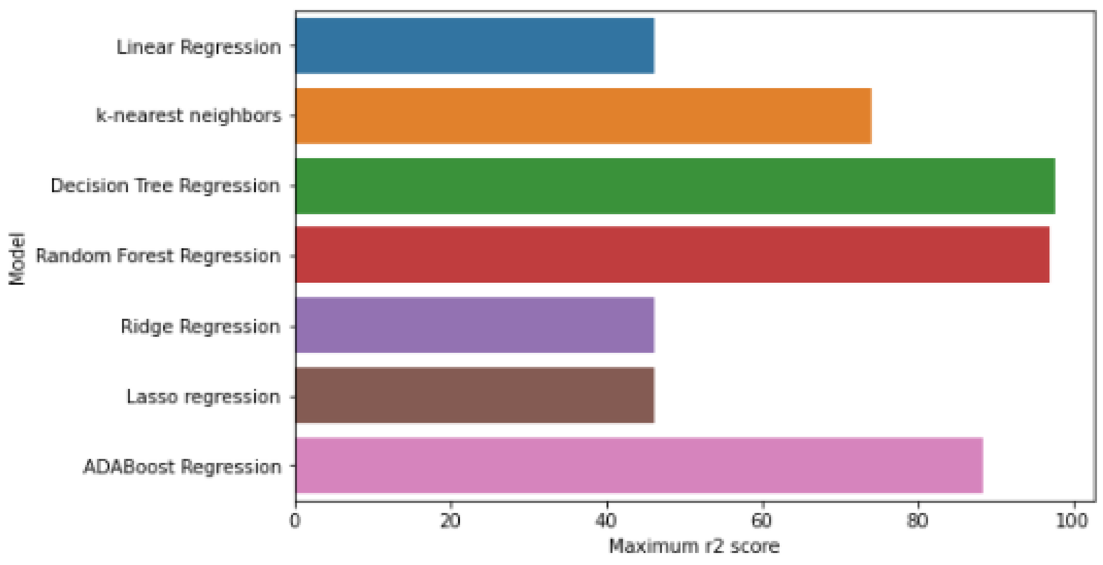
3- k-nearest neighbors (R2 & CV score is Good.)

2- **After using ensemble methods**: - AdaBoost is used as ensemble method to combine several machine learning techniques into one predictive model in order to decrease variance, bias & improve predictions. After applying ensemble methods on model and using cross validation we have obtained different coefficient of determination & cross validation scores and after using Error metrics we have calculated MAE, MSE & RMSE for all models. Now we are seeing visualization using Bar Plot of r2 scores: -



	Model	Maximum r2 score	Cross Validation Score	Mean absolute error	Root Mean Squared Error	Mean squared error
0	Linear Regression	46.28	37.00	96587.88	151559.71	2.297034e+10
1	k-nearest neighbors	74.04	50.42	70287.55	116748.05	1.363011e+10
2	Decision Tree Regression	97.77	-20.44	9704.20	33677.57	1.134179e+09
3	Random Forest Regression	97.07	55.97	21482.43	37245.64	1.387238e+09
4	Ridge Regression	46.29	37.04	96582.13	151557.50	2.296967e+10
5	Lasso regression	46.28	37.00	96587.59	151559.60	2.297031e+10
6	ADABoost Regression	88.50	54.00	43259.62	76143.16	5.797781e+09

```
<AxesSubplot:xlabel='Maximum r2 score', ylabel='Model'>
```



Observations:

1- After comparing above 7 models on basis of scores and errors, & also after using ensemble methods still these 2 models Random Forest regression & Decision Tree Regression are giving good performance.

2- Ada boost Regression is giving good scores but errors are high so we will not select this option.

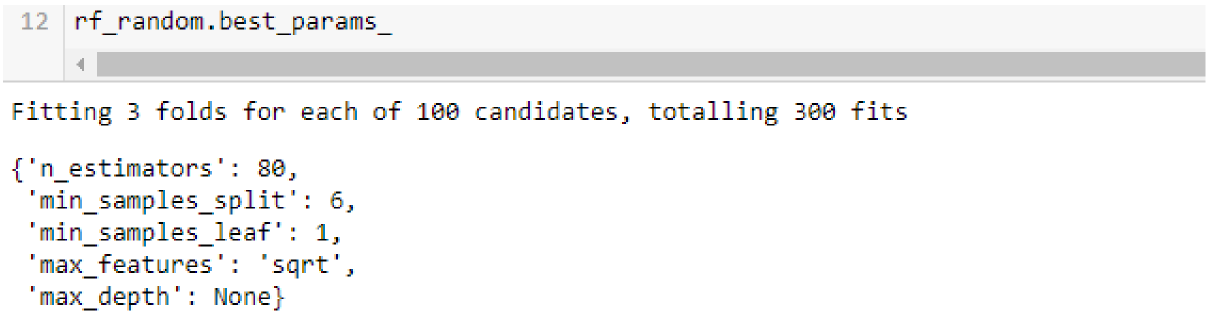
3- But when we see all the parameters very carefully and making a final decision about selection then Random Forest regression is the best option because all scores are higher, RMSE is less and also other errors are less.

4- Now we are going to do Hyperparameter Tuning for Random Forest regression models using Randomized Search CV approach for best model selection so that we will get best results after model implementation.

- ## Interpretation of the Results: -

- After comparing above 7 models on basis of scores and errors, & after using ensemble methods we have selected **Random Forest regression** for this project. To get best scores, now are using randomized search cv for hyperparameter tuning.

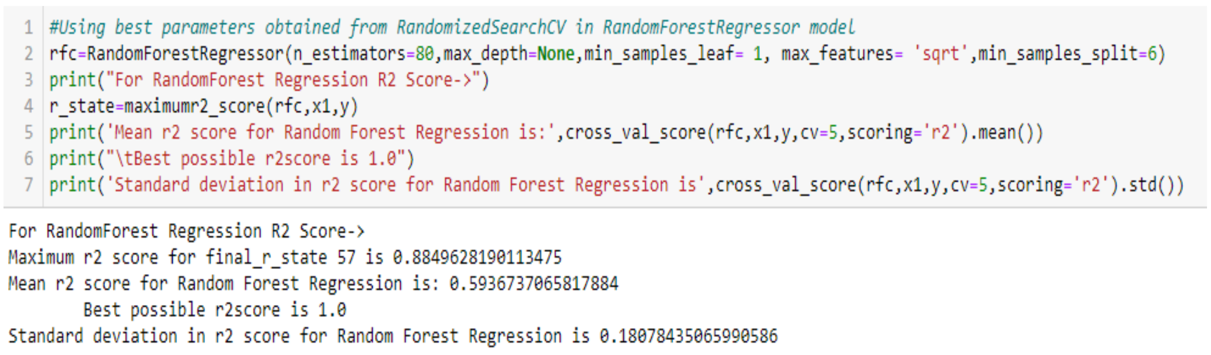
- **Hyperparameter tuning using randomized search cv** – Using Scikit-Learn 's Randomized Search CV method, we can define a grid of hyperparameter ranges, and randomly sample from the grid, performing K-Fold CV with each combination of values.



```
rf_random.best_params_
```

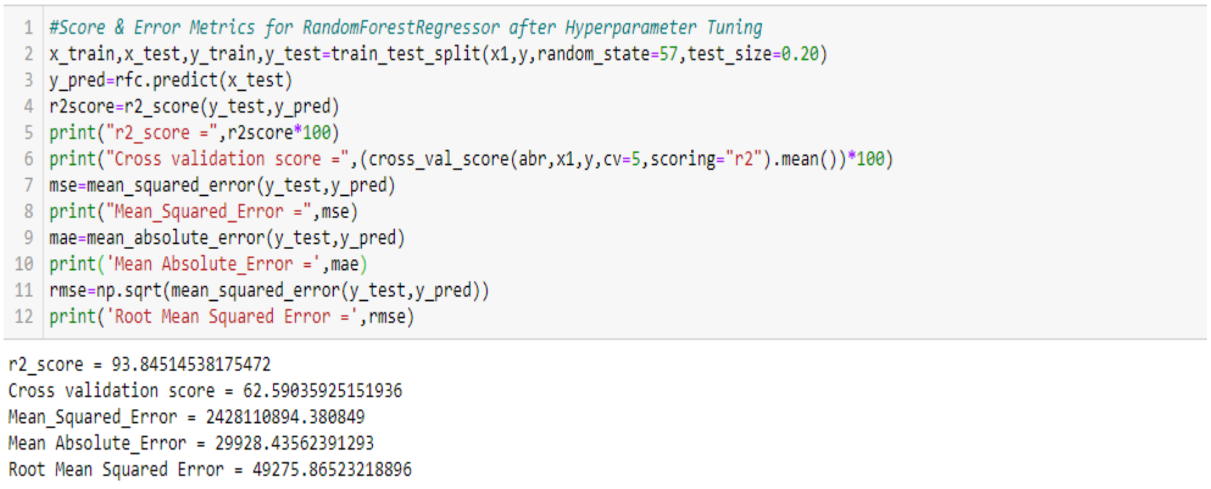
```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

{'n_estimators': 80,
 'min_samples_split': 6,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': None}
```

- ## Evaluation & error metrics for final model: -



```
#Using best parameters obtained from RandomizedSearchCV in RandomForestRegressor model
rfc=RandomForestRegressor(n_estimators=80,max_depth=None,min_samples_leaf= 1, max_features= 'sqrt',min_samples_split=6)
print("For RandomForest Regression R2 Score->")
r_state=maximumr2_score(rfc,x1,y)
print('Mean r2 score for Random Forest Regression is:',cross_val_score(rfc,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for Random Forest Regression is',cross_val_score(rfc,x1,y,cv=5,scoring='r2').std())
```

```
For RandomForest Regression R2 Score->
Maximum r2 score for final_r_state 57 is 0.8849628190113475
Mean r2 score for Random Forest Regression is: 0.5936737065817884
        Best possible r2score is 1.0
Standard deviation in r2 score for Random Forest Regression is 0.18078435065990586
```



```
#Score & Error Metrics for RandomForestRegressor after Hyperparameter Tuning
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=57,test_size=0.20)
y_pred=rfc.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score*100)
print("Cross validation score =",(cross_val_score(abr,x1,y,cv=5,scoring="r2").mean())*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
```

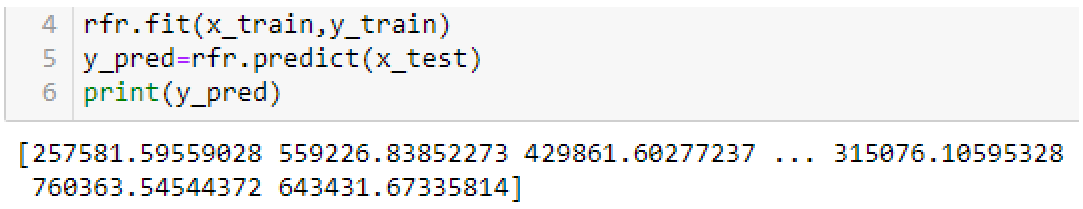
```
r2_score = 93.84514538175472
Cross validation score = 62.59035925151936
Mean_Squared_Error = 2428110894.380849
Mean Absolute_Error = 29928.43562391293
Root Mean Squared Error = 49275.86523218896
```

Observation:

Random Forest Regression gives best performance before Hyper Parameter Tuning & also after Hyper Parameter Tuning as compared to other models so we are going to implement Random Forest Regression model in our project.

After implementation of Random Forest regression in our model we are going to check predicted values, true values and relationship between them using visualizations and also Checking Scores and errors after model fitting.

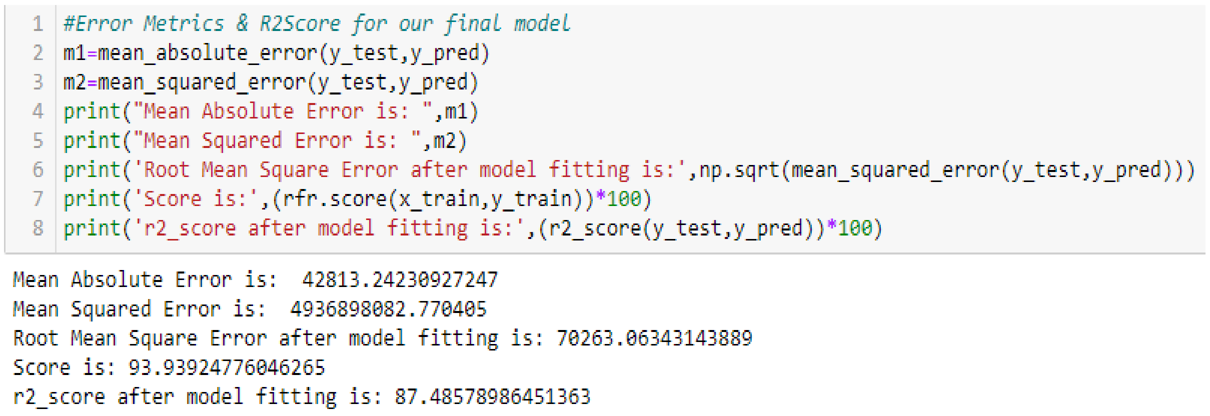
Predictions after Model Fitting: -



```
rfr.fit(x_train,y_train)
y_pred=rfr.predict(x_test)
print(y_pred)
```

```
[257581.59559028 559226.83852273 429861.60277237 ... 315076.10595328
 760363.54544372 643431.67335814]
```

Error Metrics & R2Score after Model Fitting: -

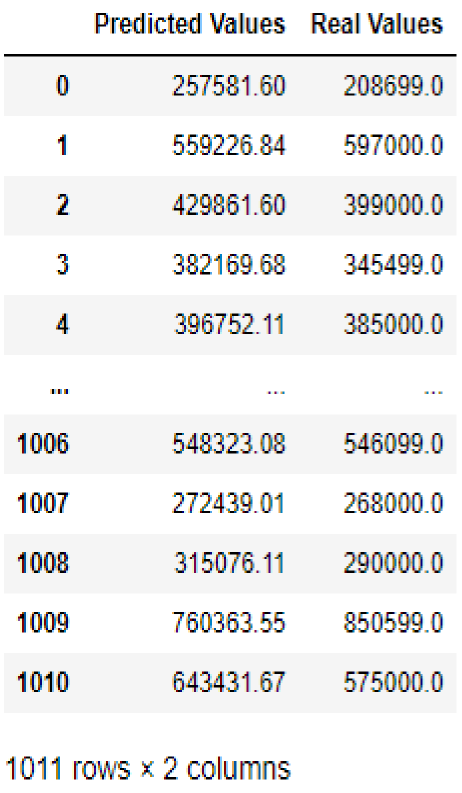


```
#Error Metrics & R2Score for our final model
m1=mean_absolute_error(y_test,y_pred)
m2=mean_squared_error(y_test,y_pred)
print("Mean Absolute Error is: ",m1)
print("Mean Squared Error is: ",m2)
print('Root Mean Square Error after model fitting is:',np.sqrt(mean_squared_error(y_test,y_pred)))
print('Score is:',(rfr.score(x_train,y_train))*100)
print('r2_score after model fitting is:',(r2_score(y_test,y_pred))*100)
```

```
Mean Absolute Error is:  42813.24230927247
Mean Squared Error is:  4936898082.770405
Root Mean Square Error after model fitting is: 70263.06343143889
Score is: 93.93924776046265
r2_score after model fitting is: 87.48578986451363
```

Saving Final model: - We have saved final model using job lib in *.pkl format.

Evaluate Predictions: -



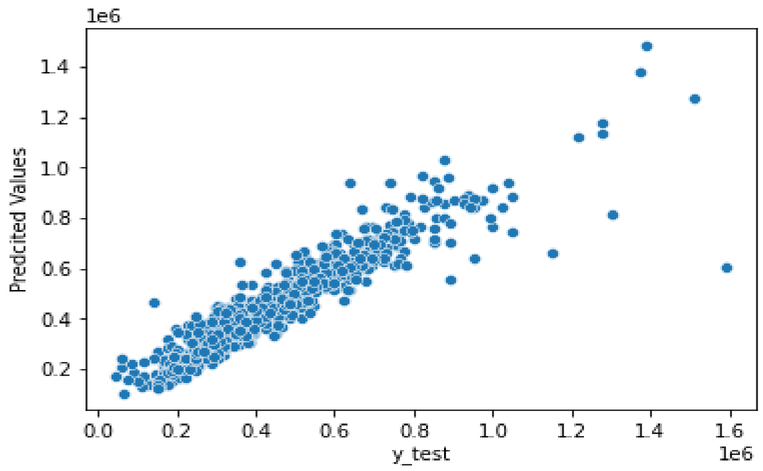
	Predicted Values	Real Values
0	257581.60	208699.0
1	559226.84	597000.0
2	429861.60	399000.0
3	382169.68	345499.0
4	396752.11	385000.0
...	...	...
1006	548323.08	546099.0
1007	272439.01	268000.0
1008	315076.11	290000.0
1009	760363.55	850599.0
1010	643431.67	575000.0

1011 rows × 2 columns

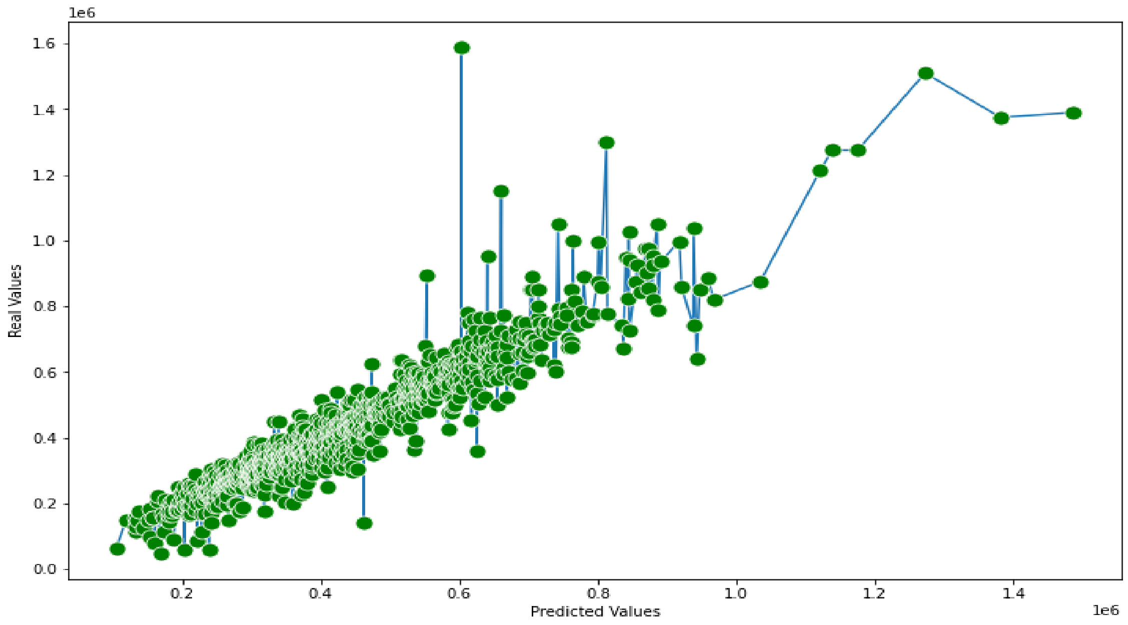
Visualizations to find relation between real and Predicted values: -

1- Scatter Plot: -

```
Text(0, 0.5, 'Predcited Values')
```



2- Line Plot: -



Observation: -

1- Above Plot shows Predicted values are nearly close to real values.

2- Graph is linear except few deviations and it shows good relation between predicted and real values.

3- R2 Score is best and error is minimum for our selected model.

4- Random Forest Regressor is best selection for this project.

CONCLUSION

• Key Findings and Conclusions of the Study

- In this project, we demonstrated the use of machine learning algorithms on a very challenging dataset to predict housing prices. To achieve the best performance, we showed that data pre-processing, a careful selection of techniques of balancing dataset, handling missing values, performed data cleaning, removing skewness, performed regression algorithms are all very important. Random Forest & Decision Tree regressors work quite well on our dataset, and the use of AdaBoost Regression is also effective. In the future, we want to continue exploring more sophisticated learning algorithms and dimension reduction techniques to further improve model performance on this important prediction task.

- Also, we have tested these machine learning algorithms on 2 different PCs of different technical configurations and have also compared their performance using evaluation & error metrics & and then chose the best performing model.

• Learning Outcomes of the Study in respect of Data Science

Learning outcomes are as: -

1- Data cleaning is quite tedious task and also very time taking and while working on this project we had encountered multiple issues but after research, study and guidance we neutralized these issues and also cleaned data properly.
2- Using Data visualization, we can easily identify outliers, skewness, missing values & correlation etc. Also, we can identify the relation between target & other features using it. In this project we have used Matplotlib & Seaborn library for data visualization.
3- Random Forest Regression have worked best in terms of r2 score & cross validation and RMSE is minimum and other errors are also less. Even it gave best parameters during hyperparameter tuning and we have used these parameters in our final model.

• Limitations of this work and Scope for Future Work

Following limitations and scope of future work are as: -

1- Selection of best random state to calculate the maximum accuracy of model is very time taking especially in case of Random Forest and AdaBoost Algorithms.
2- Hyperparameter tuning using Grid Search CV is very time consuming specially in prediction of best parameters for AdaBoost, Lasso & k-nearest neighbors. So, we used randomized search cv to get best parameters for random forest regression.
3- Dataset have many missing values so it took huge time & steps to handle this issue.
4- Size of dataset is huge so sometimes it was difficult to handle this dataset but after completion of this project we got enough confidence to handle big datasets.
5- There are some irrelevant columns in this dataset so we have tried our best to check and remove them and also tried not to lose important data.
6- We were trying to remove outliers using Z score and data loss was only 2.75%.
7- We have handled skewness using NumPy methods.
8- In future we will work on more algorithms to make more efficient model.

||Thank you||