FLIP ROBO

USED CAR PRICE PREDICTION

Submitted by:
Kritanjay Singh

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot.

Some of the reference sources are as follows:

- Coding Ninjas
- Medium.com
- StackOverflow

INTRODUCTION

BUSINESS PROBLEM FRAMING

In this project,we have to make used car price valuation model using new machine learning models from new data. Because with the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

1.Firstly,we will prepare our own dataset using web scraping.

2. After that we will check whether the project is a regression type or a classification type.

3. We will also check whether our dataset is balanced or imbalanced. If it is an imbalanced one, we will apply sampling techniques to balance the dataset.

4. Then we will do model building and check its accuracy.

5. Our main motto is to build a model with good accuracy and for that we will also go for hyperparameter tuning.

REVIEW OF LITERATURE

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.

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE:

Device specifications

Mi NoteBook Horizon Edition 14

Device name	LAPTOP-ED8G2MH8
Processor	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
Installed RAM	8.00 GB (7.83 GB usable)
Device ID	05E09149-DB9B-49DE-88A4-9C13612E78F7
Product ID	00327-35882-06869-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Rename this PC

Windows specifications

Edition	Windows 10 Home Single Language
Version	1909
Installed on	29-09-2020
OS build	18363.1440

SOFTWARE*:*

Jupyter Notebook (Anaconda 3) – Python 3.7.6

Microsoft Excel 2016

LIBRARIES:

The tools, libraries, and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition, sklearn standardscaler, GridSearchCV, joblib.

from sklearn.preprocessing import StandardScaler

As these columns are different in scale, they are standardized to have common scale while building machine learning model. This is useful when you want to compare data that correspond to different units.

from sklearn.preprocessing import Label Encoder

Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

from sklearn.model_selection import train_test_split,cross_val_score

Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

Through pandas library we loaded our csv file 'Data file' into dataframe and performed data manipulation and analysis.

With the help of numpy we worked with arrays.

With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization.

With sklearn's standardscaler package we scaled all the feature variables onto single scale.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

If you look at data science, we are actually using mathematical models to model (and hopefully through the model to explain some of the things that we have seen) business circumstances, environment etc and through these model, we can get more insights such as the outcomes of our decision undertaken, what should we do next or how shall we do it to improve the odds. So mathematical models are important, selecting the right one to answer the business question can tremendous value to the organization.

Here I am using Random Forest Regressor with accuracy 90.8% after hyper parameter tuning.

DATA SOURCES AND THEIR FORMATS

Data Source: The read_csv function of the pandas library is used to read the content of a CSV file into the python environment as a pandas DataFrame. The function can read the files from the OS by using proper path to the file. Data description: Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.

DATA PREPROCESSING DONE

- I have checked for null values
- I have label encoded the object type columns in the dataset.
- I have checked the correlation between dependant and independent variables using heatmap. I have seen most of the independent variables are correlated with each other and the target variable is positively correlated with a very few independent variables.
- I have done some visualization using histogram.
- I have checked outliers using boxplots ,but no outliers are present.
- I also have checked for skewness in my data, but the skewness present is very negligible, so I don't consider it.
- I have splitted the dependant and independent variables into x and y.
- I have scaled the data using StandardScaler method and made my data ready for model building.

DATA DESCRIPTION

After loading all the required libraries we loaded the data into our jupyter notebook.

The dataset contains 6224 records (rows) and 10 features (columns).

Here, we will provide a brief description of dataset features. Since the number of features is 10, we will attach the data description i.e.,'Model', 'Engine', 'Owner(s)', 'Manufacturing_year', 'Driven_km', 'Fuel_type', 'Transmission', 'Selling_Price', 'location','Mileage'.

```
#Importing Libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error

from sklearn.model_selection import cross_val_score

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

Extracting_Dataset:

```
#Extract datasets from csv file
data = pd.read_excel(r'C:\Users\91977\Documents\Python Scripts\PROJECTS_FLIPROBO\Used_CarDetails.xlsx')
```

```
data  #display the datset
```

	Unnamed: 0	Unnamed: 0.1	Model	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Engine	Price	Location
0	12	12	Maruti Wagon R	2017	41174	Petrol	Automatic	1	20.51	998	430000	Ahmedabad
1	14	14	Hyundai Verna CRDi . AT SX Plus	2017	70000	Diesel	Automatic	1	22.00	1582	894999	Ahmedabad
2	58	58	Audi A TDI Premium Plus	2018	14667	Diesel	Automatic	1	18.25	1968	3200000	Ahmedabad
3	62	62	Honda City i VTEC CVT VX	2016	55000	Petrol	Automatic	1	18.00	1497	877999	Ahmedabad
4	63	63	Mercedes-Benz E-Class Exclusive E d BSIV	2019	30486	Diesel	Automatic	1	16.10	1950	4800000	Ahmedabad
...	...	...	...	...	...	...	...	...	...	...	...	...
6219	6411	6449	Ford EcoSport . Diesel Titanium BSIV	2019	30000	Diesel	Manual	1	23.00	1498	990000	Pune
6220	6412	6450	Maruti Wagon R VXI Plus	2017	40000	Petrol	Manual	1	20.51	998	450000	Pune
6221	6419	6457	Toyota Yaris G BSIV	2018	23643	Petrol	Manual	1	17.10	1496	1000000	Pune
6222	6422	6460	Hyundai Verna . VTVT	2012	69000	Petrol	Manual	1	17.43	1396	465000	Pune
6223	6423	6461	Maruti Zen Estilo LXI BSIII	2011	67000	Petrol	Manual	1	18.20	998	225000	Pune

6224 rows × 12 columns

After dropping the unnamed columns, this is the dataset that we will be working on

```
data #display the dataset
```

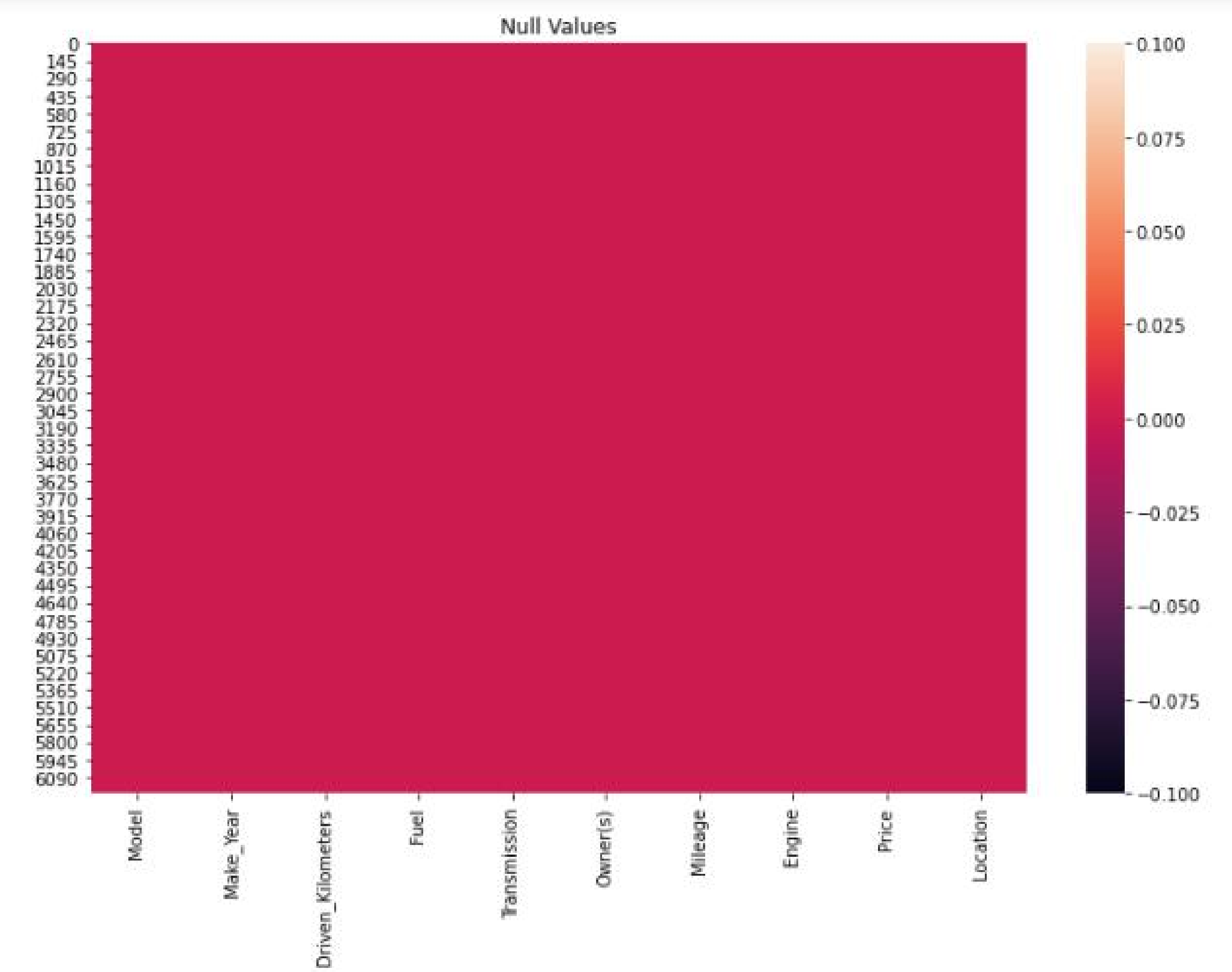
	Model	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Engine	Price	Location
0	Maruti Wagon R	2017	41174	Petrol	Automatic	1	20.51	998	430000	Ahmedabad
1	Hyundai Verna CRDi . AT SX Plus	2017	70000	Diesel	Automatic	1	22.00	1582	894999	Ahmedabad
2	Audi A TDI Premium Plus	2018	14667	Diesel	Automatic	1	18.25	1968	3200000	Ahmedabad
3	Honda City i VTEC CVT VX	2016	55000	Petrol	Automatic	1	18.00	1497	877999	Ahmedabad
4	Mercedes-Benz E-Class Exclusive E d BSIV	2019	30486	Diesel	Automatic	1	16.10	1950	4800000	Ahmedabad
...	...	...	...	...	...	...	...	...	...	...
6219	Ford EcoSport . Diesel Titanium BSIV	2019	30000	Diesel	Manual	1	23.00	1498	990000	Pune
6220	Maruti Wagon R VXI Plus	2017	40000	Petrol	Manual	1	20.51	998	450000	Pune
6221	Toyota Yaris G BSIV	2018	23643	Petrol	Manual	1	17.10	1496	1000000	Pune
6222	Hyundai Verna . VTVT	2012	69000	Petrol	Manual	1	17.43	1396	465000	Pune
6223	Maruti Zen Estilo LXI BSIII	2011	67000	Petrol	Manual	1	18.20	998	225000	Pune

```
data.info()  #information about the data

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6224 entries, 0 to 6223
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Model              6224 non-null   object
 1   Make_Year          6224 non-null   int64
 2   Driven_Kilometers  6224 non-null   int64
 3   Fuel               6224 non-null   object
 4   Transmission       6224 non-null   object
 5   Owner(s)           6224 non-null   int64
 6   Mileage            6224 non-null   float64
 7   Engine             6224 non-null   int64
 8   Price              6224 non-null   int64
 9   Location           6224 non-null   object
dtypes: float64(1), int64(5), object(4)
memory usage: 486.4+ KB
```

- These 10 columns comprises of both dimensions (categorical value) and measures (numeric value)

Feature Engineering has been used for cleaning of the data. Some unused columns have been deleted and even some columns have been bifurcated which was used in the prediction. We first looked percentage of values missing in columns and then proceeded with the outliers removal and skewness check

Heat Map for missing Value



We can clearly see that there is no null values in our dataset

STATISTICAL SUMMARY

To see statistical information about the non-numerical columns in our dataset:

```
#Let's check the overall metrics of each column


data.describe()
```

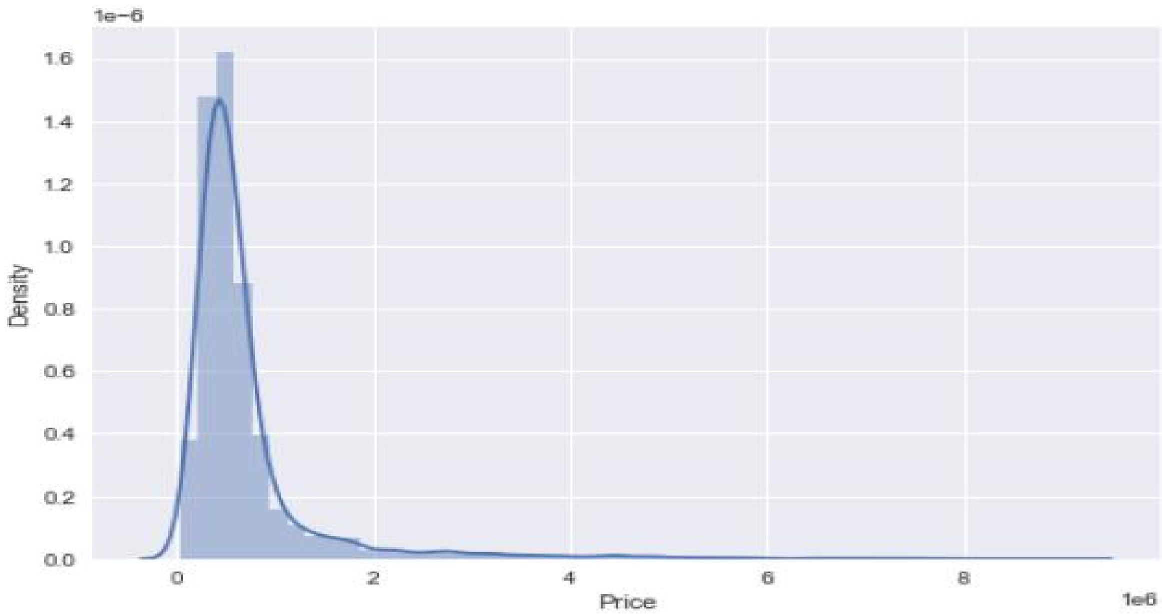
	Make_Year	Driven_Kilometers	Owner(s)	Mileage	Engine	Price
count	6224.000000	6224.000000	6224.000000	6224.000000	6224.000000	6.224000e+03
mean	2014.862789	58242.295148	1.214653	19.957942	1405.529724	7.030040e+05
std	3.056772	37702.893801	0.467354	3.872215	467.313843	7.639553e+05
min	2000.000000	500.000000	1.000000	7.500000	624.000000	4.500000e+04
25%	2013.000000	32119.250000	1.000000	17.400000	1197.000000	3.550000e+05
50%	2015.000000	55000.000000	1.000000	20.140000	1248.000000	5.000000e+05
75%	2017.000000	77072.250000	1.000000	22.540000	1498.000000	7.000000e+05
max	2021.000000	886253.000000	4.000000	36.000000	5000.000000	9.100000e+06

- From this statistical analysis we make some of the interpretations that,'Driven_Kilometers' and 'Engine', We see that there is disturbancy comparatively in our Mean and Median and "mean v/s std"
- Hence, we would need to check for the outliers and remove them

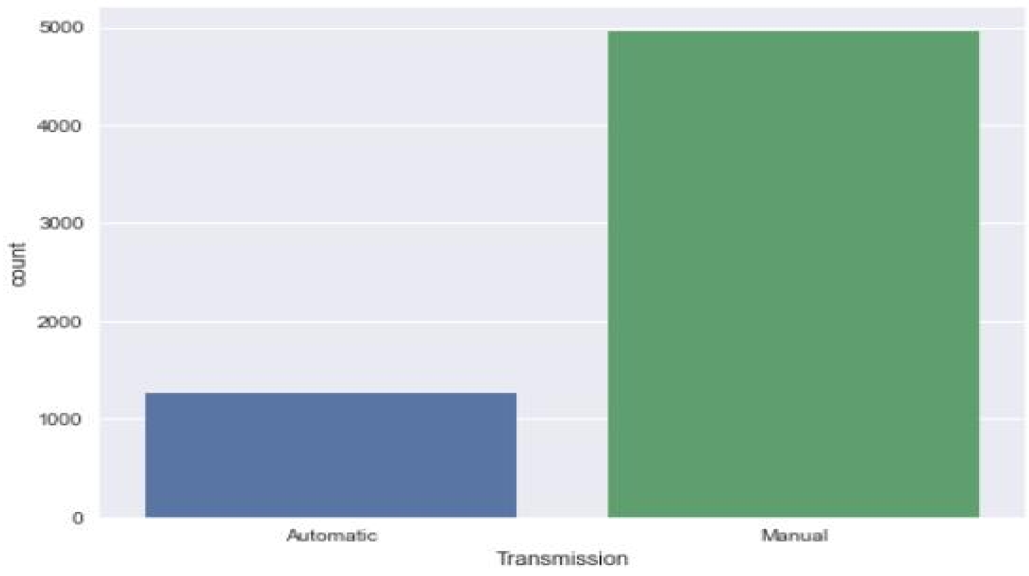
EDA(Exploratory Data Analysis)

Let us explore our data and visualize

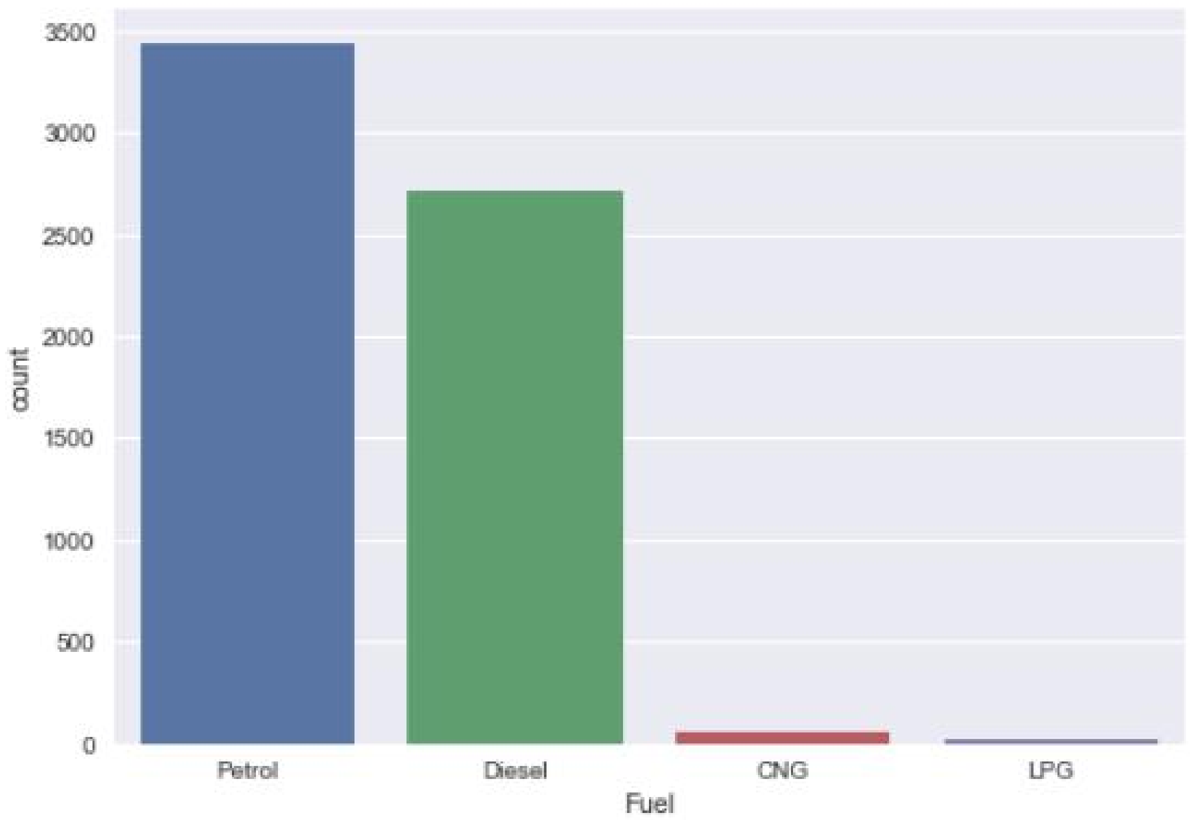
Target Variable (Selling Price)



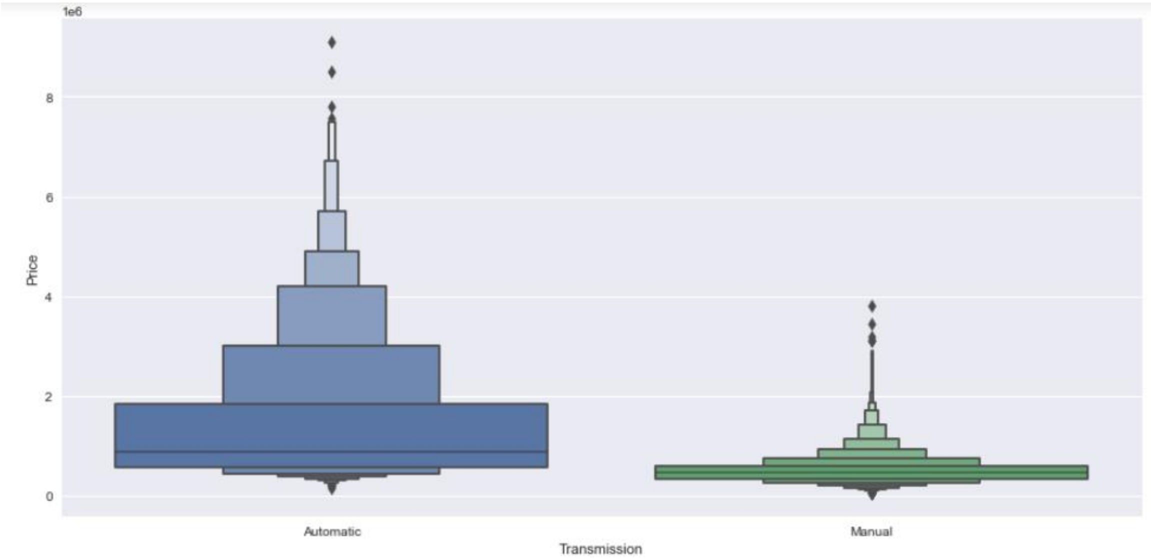
- Price column is not normally distributed
- we have some of the car prices with a high price than normal



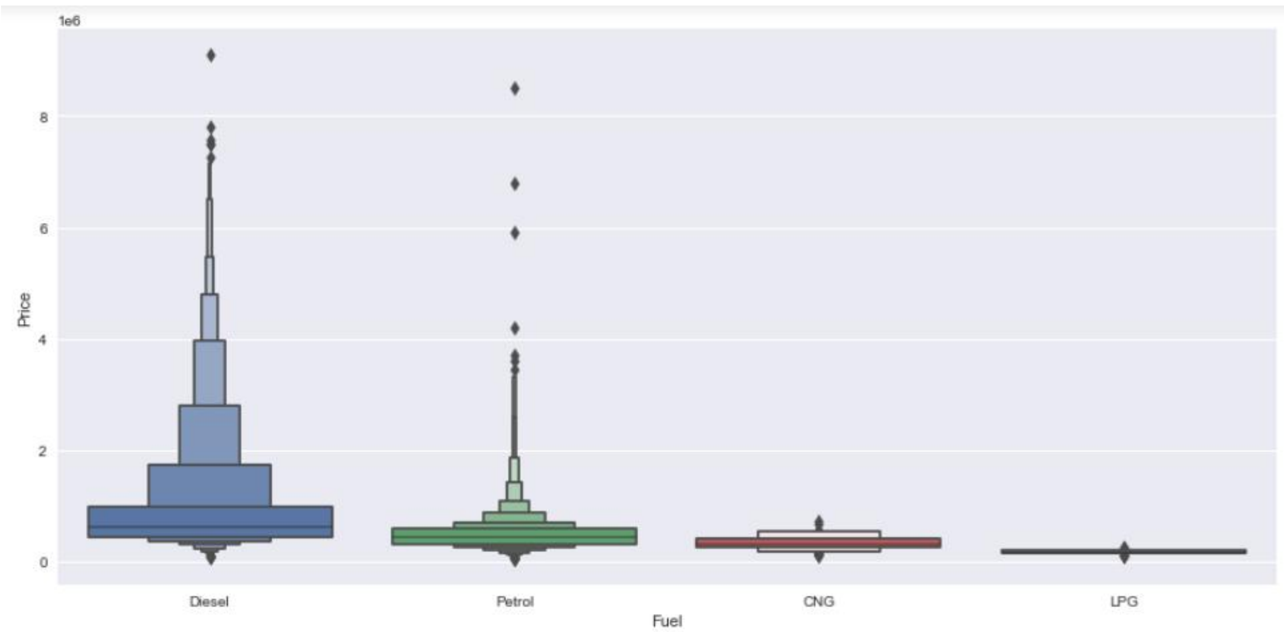
- (Transmission) Manual Used Car are mostly available for sale



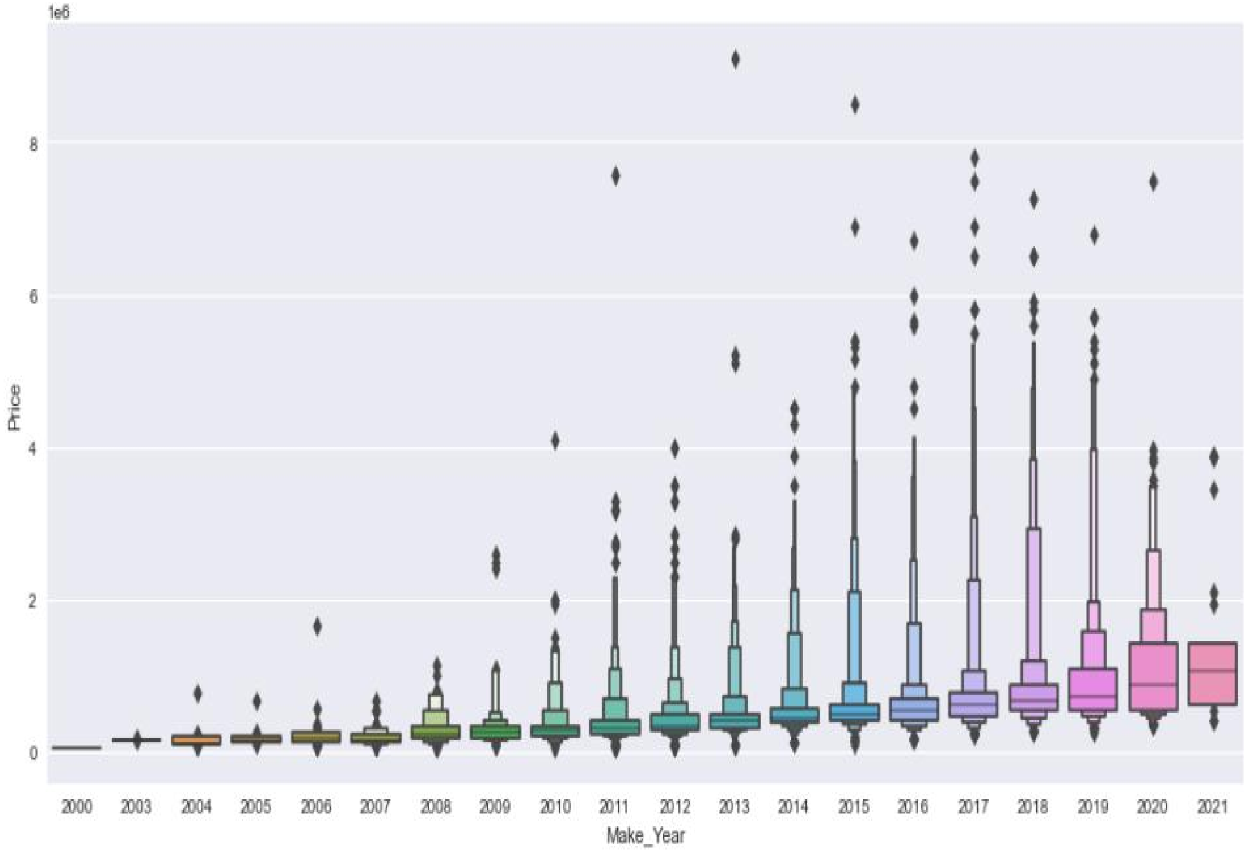
- Cars with fuel type "Petrol" and "Diesel" are highly available for sale



- Automatic Car Price is higher when compared to Manual Car transmission Car Price



- Again Used Cars with fuel type: “Diesel” and “Petrol” are mostly costly



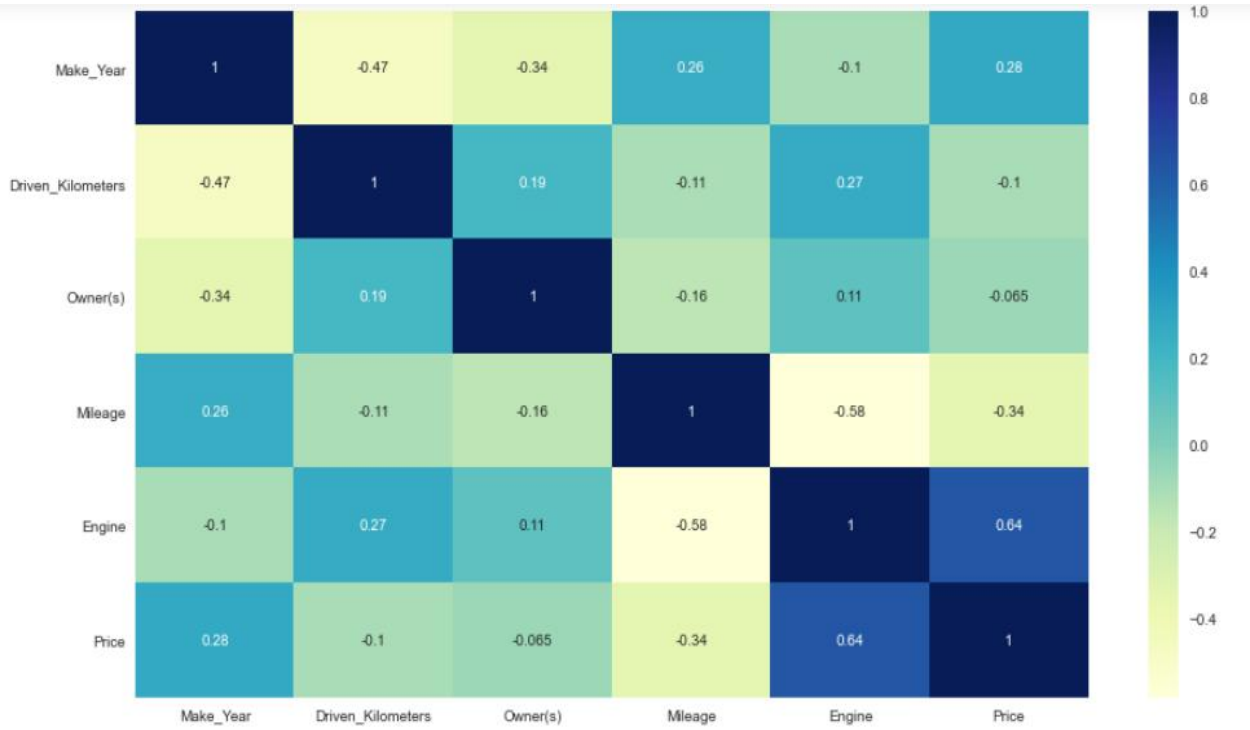
- During 2013 - 2017, people were selling the cars with high price, but due to this pandemic (covid-19) the used car sale price is drastically reduced

Correlation matrix:

A correlation matrix is simply a table which displays the correlation. The measure is best used in variables that demonstrate a linear relationship between each other. The fit of the data can be visually represented in a heatmap.

```
data.corr()
```

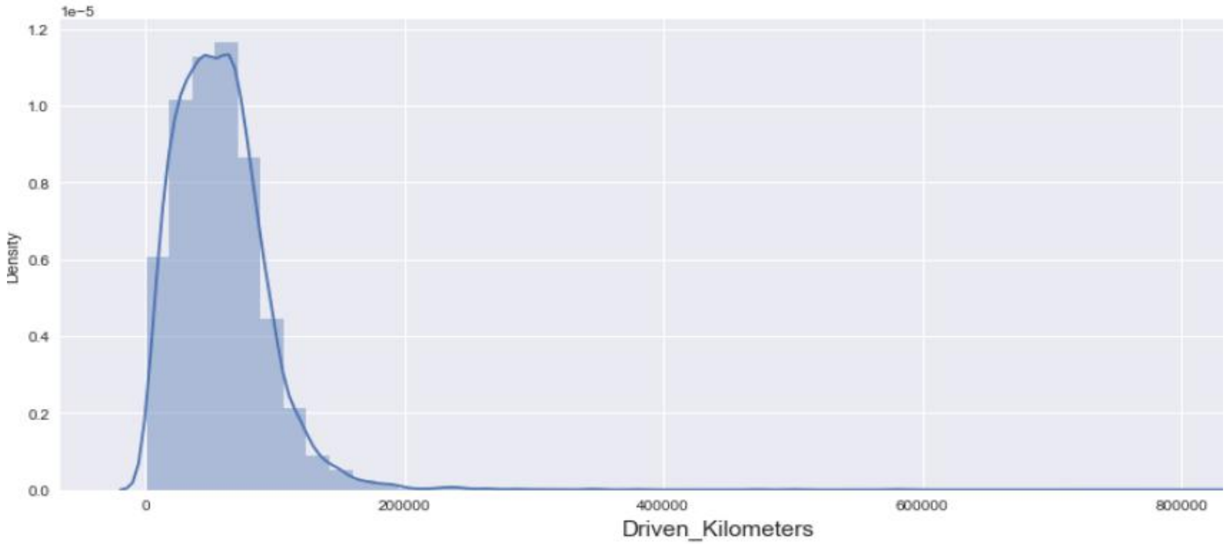
	Make_Year	Driven_Kilometers	Owner(s)	Mileage	Engine	Price
Make_Year	1.00000000	-0.46751572	-0.33809225	0.25822021	-0.10281409	0.27804739
Driven_Kilometers	-0.46751572	1.00000000	0.19364754	-0.10668879	0.26871062	-0.10012936
Owner(s)	-0.33809225	0.19364754	1.00000000	-0.15976243	0.11034169	-0.06469692
Mileage	0.25822021	-0.10668879	-0.15976243	1.00000000	-0.58217861	-0.33521777
Engine	-0.10281409	0.26871062	0.11034169	-0.58217861	1.00000000	0.63812188
Price	0.27804739	-0.10012936	-0.06469692	-0.33521777	0.63812188	1.00000000

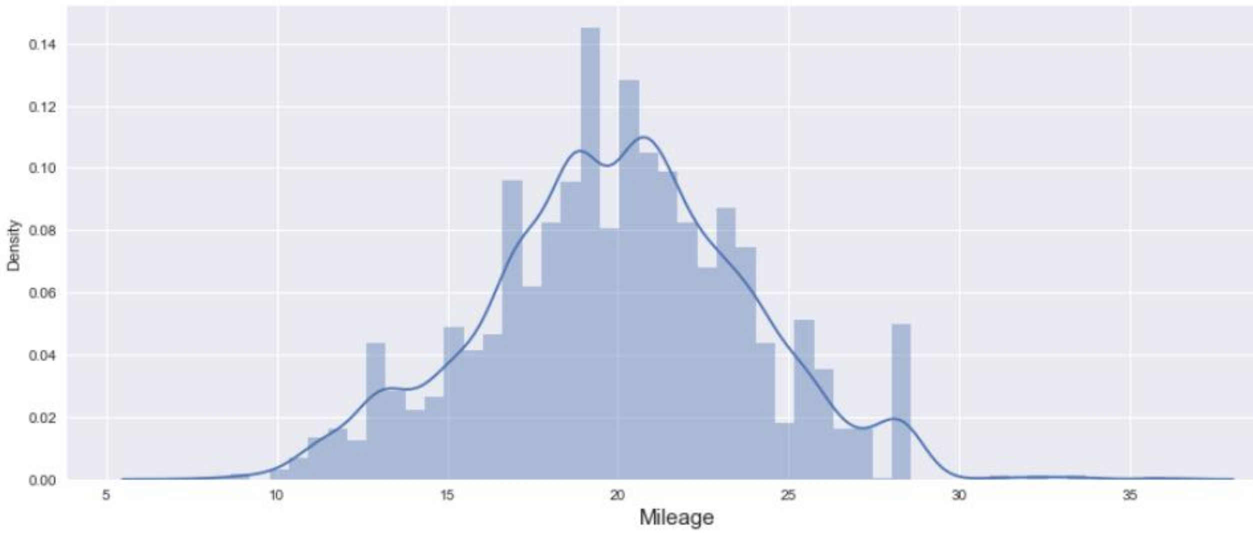


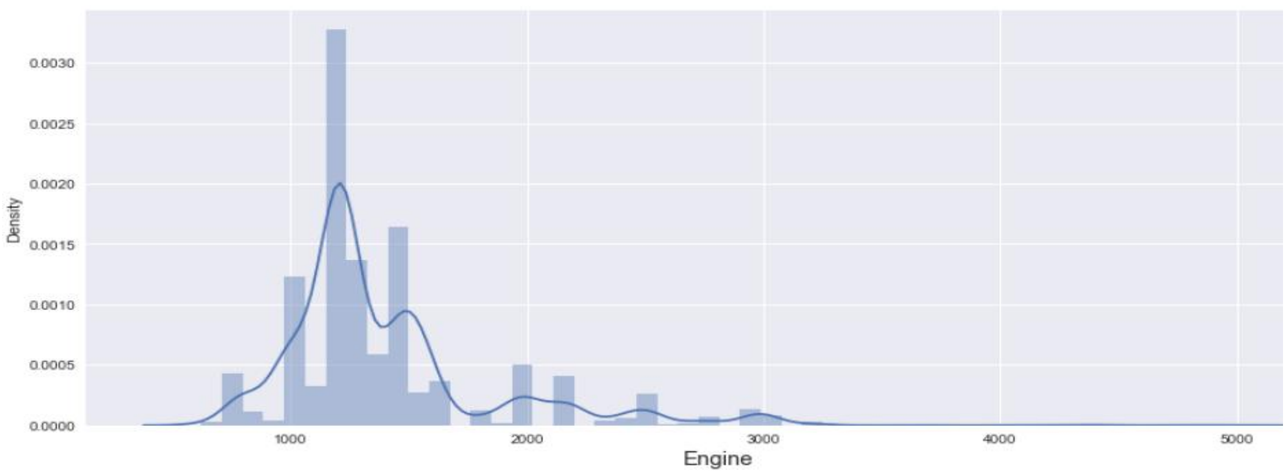
We see that,
- the largest correlated features are "Engine" and "Price" with correlated values: "0.64"
- the lowest correlated features are "Owner(s)" and "Price" with correlated values: "-0.065"

DATA PREPROCESSING

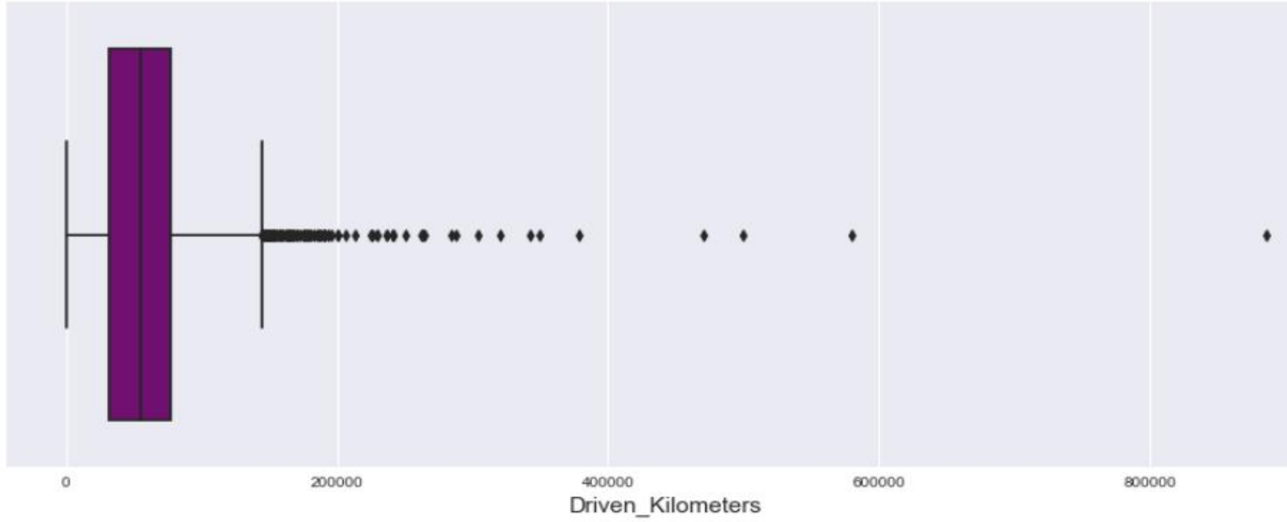
Checking the data distribution among all the columns.

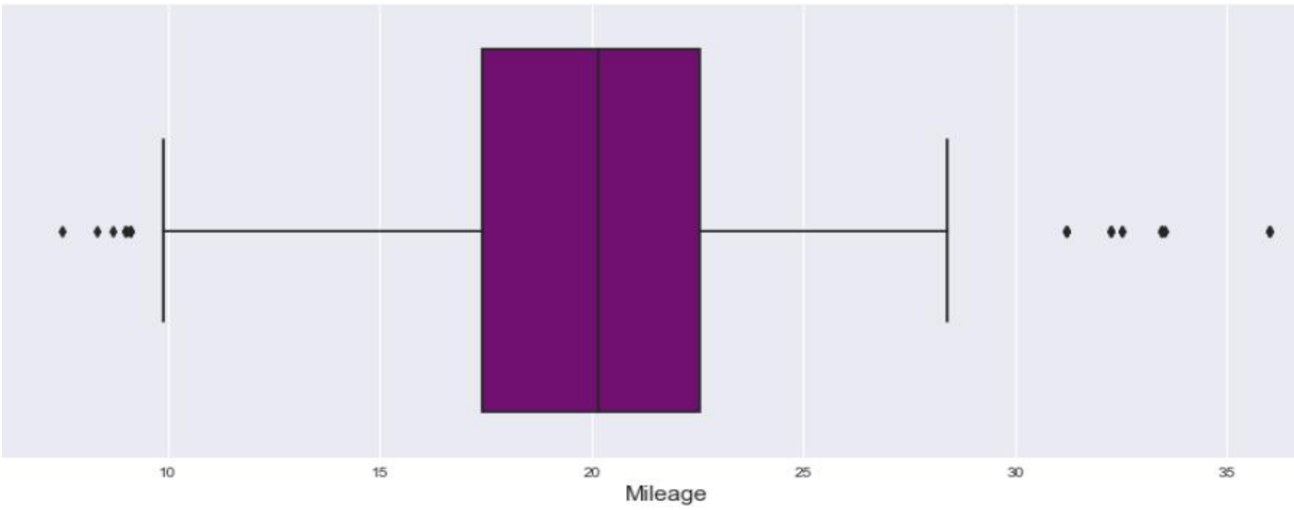


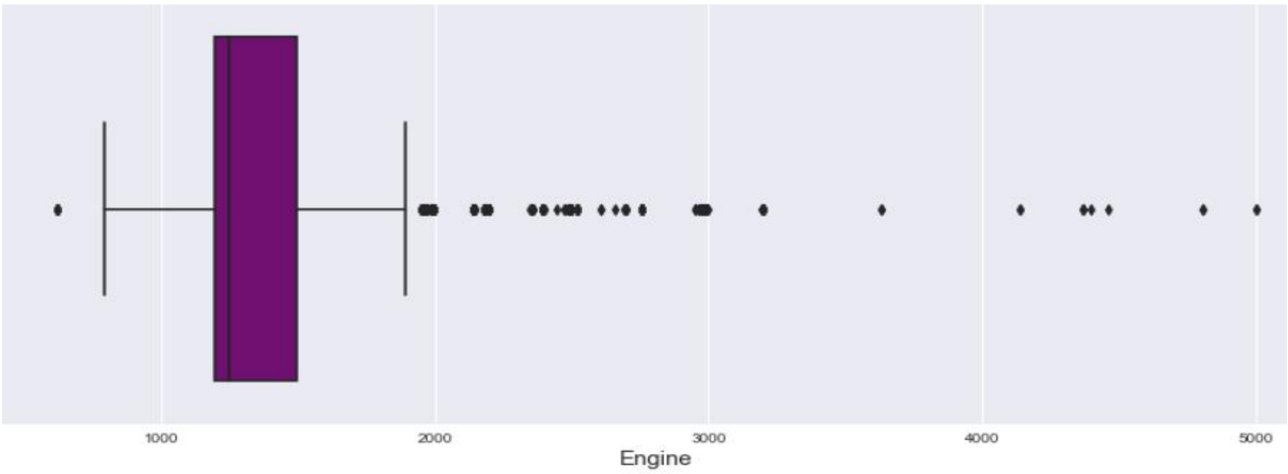




Checking the outliers using BOX plot:







features = ['Driven_Kilometers', 'Mileage', 'Engine']

#columns with outliers by checking the above plots, hence let's remove these outliers using th below techniques

Applying IQR Method

```
Q1 = data[features].quantile(0.25)
Q3 = data[features].quantile(0.75)
IQR = Q3-Q1

data_new1 = data[~((data[features] < (Q1-1.5*IQR)) | (data[features]> (Q3 + 1.5*Q3))).any(axis = 1)]
```

```
print('Shape - Before and After:\n')
print('Shape Before'.ljust(20),":",data.shape)
print('Shape After'.ljust(20),":",data_new1.shape)
print('Percentage Loss'.ljust(20),":",((data.shape[0]-data_new1.shape[0])/data.shape[0])*100)
```

```
Shape - Before and After:

Shape Before         : (6224, 10)
Shape After          : (6160, 10)
Percentage Loss      : 1.0282776349614395
```

Applying z-score Method

```
from scipy.stats import zscore  #importing zscore from library

z=np.abs(zscore(data[features]))
threshold = 3
data_new2 = data[(z<3).all(axis=1)]
```

```
print('Shape - Before and After:\n')
print('Shape Before'.ljust(20),":",data.shape)
print('Shape After'.ljust(20),":",data_new2.shape)
print('Percentage Loss'.ljust(20),":",((data.shape[0]-data_new2.shape[0])/data.shape[0])*100)
```

```
Shape - Before and After:

Shape Before         : (6224, 10)
Shape After          : (6017, 10)
Percentage Loss      : 3.3258354755784065
```

Observation:

(IQR Method)Percentage Loss : 1.0282776349614395 %

(z-score Method) Percentage Loss : 3.3258354755784065 %

Percentage of data loss is less after applying IQR technique. So, let's proceed with IQR method

SKEWNESS:

```
#Skewness after applying the outliers technique

data_new.skew()
```

```
Make_Year            -0.611907
Driven_Kilometers     0.701490
Owner(s)              2.244314
Mileage               0.012334
Engine                1.738002
Price                 4.160594
dtype: float64
```

Skewness is more in the columns:

`"Driven_Kilometers" and "Engine"`

- "Make_Year" and "Owner(s)" are ordinal data so skewness are ignored
- "Price" target variable so skewness is ignored

```
data_new['Driven_Kilometers'] = np.sqrt(data_new['Driven_Kilometers'])

data_new['Engine'] = np.log(data_new['Engine'])
data_new['Engine'] = np.cbrt(data_new['Engine'])
data_new['Engine'] = np.sqrt(data_new['Engine'])
```

```
data_new.skew()
```

```
Make_Year            -0.611907
Driven_Kilometers    -0.130465
Owner(s)              2.244314
Mileage               0.012334
Engine                0.746089
Price                 4.160594
dtype: float64
```

We have removed the maximum skewness from our dataset

Adding Features in Datasets

Adding new feature “Brand” in data frame

Index	Brand
0	Maruti
1	Hyundai
2	Audi
3	Honda
4	Mercedes-Benz
...	...
6155	Ford
6156	Maruti
6157	Toyota
6158	Hyundai
6159	Maruti

6160 rows × 1 columns

Dropped Features

- Models

Here is our Dataset which is ready for further steps

	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Engine	Price	Location	Brand
0	2017	202.91377479	Petrol	Automatic	1	20.51000000	1.37996639	430000	Ahmedabad	Maruti
1	2017	264.57513111	Diesel	Automatic	1	22.00000000	1.39489974	894999	Ahmedabad	Hyundai
2	2018	121.10739036	Diesel	Automatic	1	18.25000000	1.40170654	3200000	Ahmedabad	Audi
3	2016	234.52078799	Petrol	Automatic	1	18.00000000	1.39315133	877999	Ahmedabad	Honda
4	2019	174.60240548	Diesel	Automatic	1	16.10000000	1.40142338	4800000	Ahmedabad	Mercedes-Benz
...	...	...	...	...	...	...	...	...	...	...
6155	2019	173.20508076	Diesel	Manual	1	23.00000000	1.39317253	990000	Pune	Ford
6156	2017	200.00000000	Petrol	Manual	1	20.51000000	1.37996639	450000	Pune	Maruti
6157	2018	153.76280434	Petrol	Manual	1	17.10000000	1.39313010	1000000	Pune	Toyota
6158	2012	262.67851073	Petrol	Manual	1	17.43000000	1.39092406	465000	Pune	Hyundai
6159	2011	258.84358211	Petrol	Manual	1	18.20000000	1.37996639	225000	Pune	Maruti

6160 rows × 10 columns

Encoding Categorical Data

#Let's check each categorical column and their unique values present in their in dependent column

The Encoding Technique is used for this problem:
label encoding technique with multiple variables.

2. Getting Dummies

Firstly, proceed with Label encoding technique with multiple variables for parti cular features i.e., Brand

Let's encode the categorical data

```
#Let's use Label encoder for encoding some of the columns

l1 = ['Transmission','Fuel','Make_Year']

#Let's use Label Encoder method

from sklearn.preprocessing import LabelEncoder  #importing library

le = LabelEncoder()  #calling function

for i in l1:
    Used_Cars[i]= le.fit_transform(Used_Cars[i].values.reshape(-1,1))
Used_Cars.head()
```

	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Engine	Price	Location	Brand
0	15	202.91377479	3	0	1	20.51000000	1.37996639	430000	Ahmedabad	Maruti
1	15	264.57513111	1	0	1	22.00000000	1.39489974	894999	Ahmedabad	Hyundai
2	16	121.10739036	1	0	1	18.25000000	1.40170654	3200000	Ahmedabad	Audi
3	14	234.52078799	3	0	1	18.00000000	1.39315133	877999	Ahmedabad	Honda
4	17	174.60240548	1	0	1	16.10000000	1.40142338	4800000	Ahmedabad	Mercedes-Benz

Secondly, proceed with getting dummies for location and Brand

```
#Get dummies
l3=pd.get_dummies(Used_Cars['Location'])

#Concat with main dataframe by dropping workclass dataframe
Used_Cars=pd.concat([Used_Cars.drop('Location',axis=1),l3],axis=1)
```

- **No more Categorical data are present in our dataset**

Now, we can see all features is converted into numerical one after proceeding with encoding technique.

	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Engine	Price	Audi	BMW	Chevrolet	Datsun	Fiat	Force	Ford
0	15	202.91377479	3	0	1	20.51000000	1.37996639	430000	0	0	0	0	0	0	0
1	15	264.57513111	1	0	1	22.00000000	1.39489974	894999	0	0	0	0	0	0	0
2	16	121.10739036	1	0	1	18.25000000	1.40170654	3200000	1	0	0	0	0	0	0
3	14	234.52078799	3	0	1	18.00000000	1.39315133	877999	0	0	0	0	0	0	0
4	17	174.60240548	1	0	1	16.10000000	1.40142338	4800000	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6155	17	173.20508076	1	1	1	23.00000000	1.39317253	990000	0	0	0	0	0	0	1
6156	15	200.00000000	3	1	1	20.51000000	1.37996639	450000	0	0	0	0	0	0	0
6157	16	153.76280434	3	1	1	17.10000000	1.39313010	1000000	0	0	0	0	0	0	0
6158	10	262.67851073	3	1	1	17.43000000	1.39092406	465000	0	0	0	0	0	0	0
6159	9	258.84358211	3	1	1	18.20000000	1.37996639	225000	0	0	0	0	0	0	0

6160 rows × 49 columns

MODEL BUILDING

Splitting features and labels

```
X = Used_Cars.drop(columns = 'Price') #Features
Y = Used_Cars['Price'] #Label
```

```
#let's check for our dimensions after splitting the data

print('Features dimension:\t',X.shape,'\nLabel Dimension:\t',Y.shape)
```

```
Features dimension:      (6160, 48)
Label Dimension:         (6160,)
```

Scaling the data

Using the StandardScaler

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

X_scaled = Scaler.fit_transform(X)
```

Finding the Best Random State ¶

```
from sklearn.linear_model import LinearRegression

maxR2_Score = 0
maxRS = 0

for i in range(200):
    x_train,x_test,y_train,y_test = train_test_split(X_scaled,Y,test_size = 0.20,random_state = i)
    LR = LinearRegression()
    LR.fit(x_train,y_train)
    predrf = LR.predict(x_test)
    Score = r2_score(y_test,predrf)
    if Score>maxR2_Score:
        maxR2_Score = Score
        maxRS = i

print('The best accuracy is ',maxR2_Score, ' with Random State ',maxRS)
```

```
The best accuracy is  0.7852481160867094  with Random State  148
```

Splitting Training and Testing data

```
#Let's split our dataset for training and testing purpose

x_train,x_test,y_train,y_test = train_test_split(X_scaled, Y, test_size =0.20, random_state = maxRS)
```

Let's build the model

```
#Importing all required Libraries that will be used for building a model

from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import Lasso,Ridge
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeRegressor
```

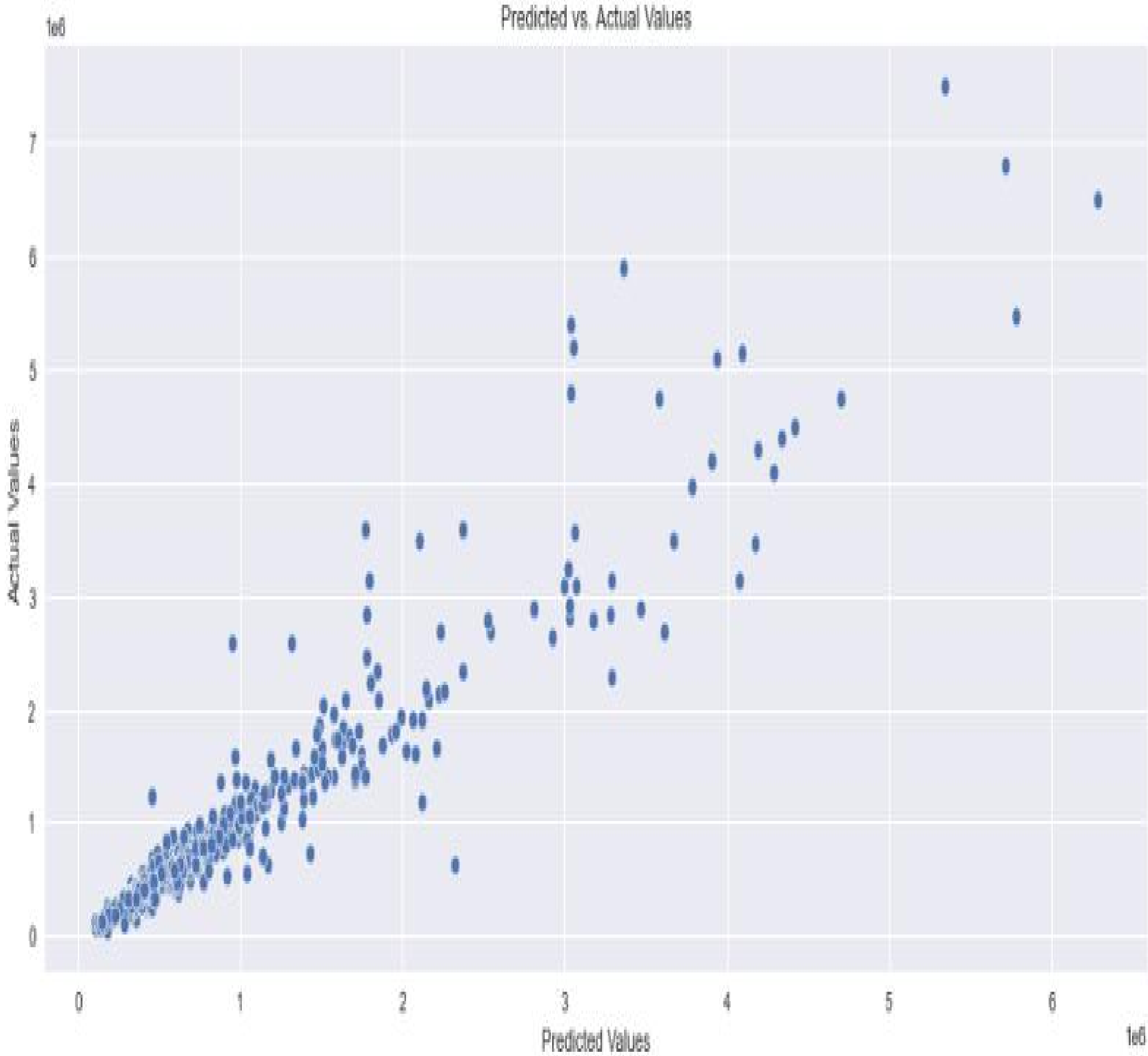
TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

The algorithms we used for the training and testing are as follows:-

- Random Forest
- k-nearest neighbors (KNN)
- Decision Tree
- Gradient Boosting
- Lasso
- Ridge

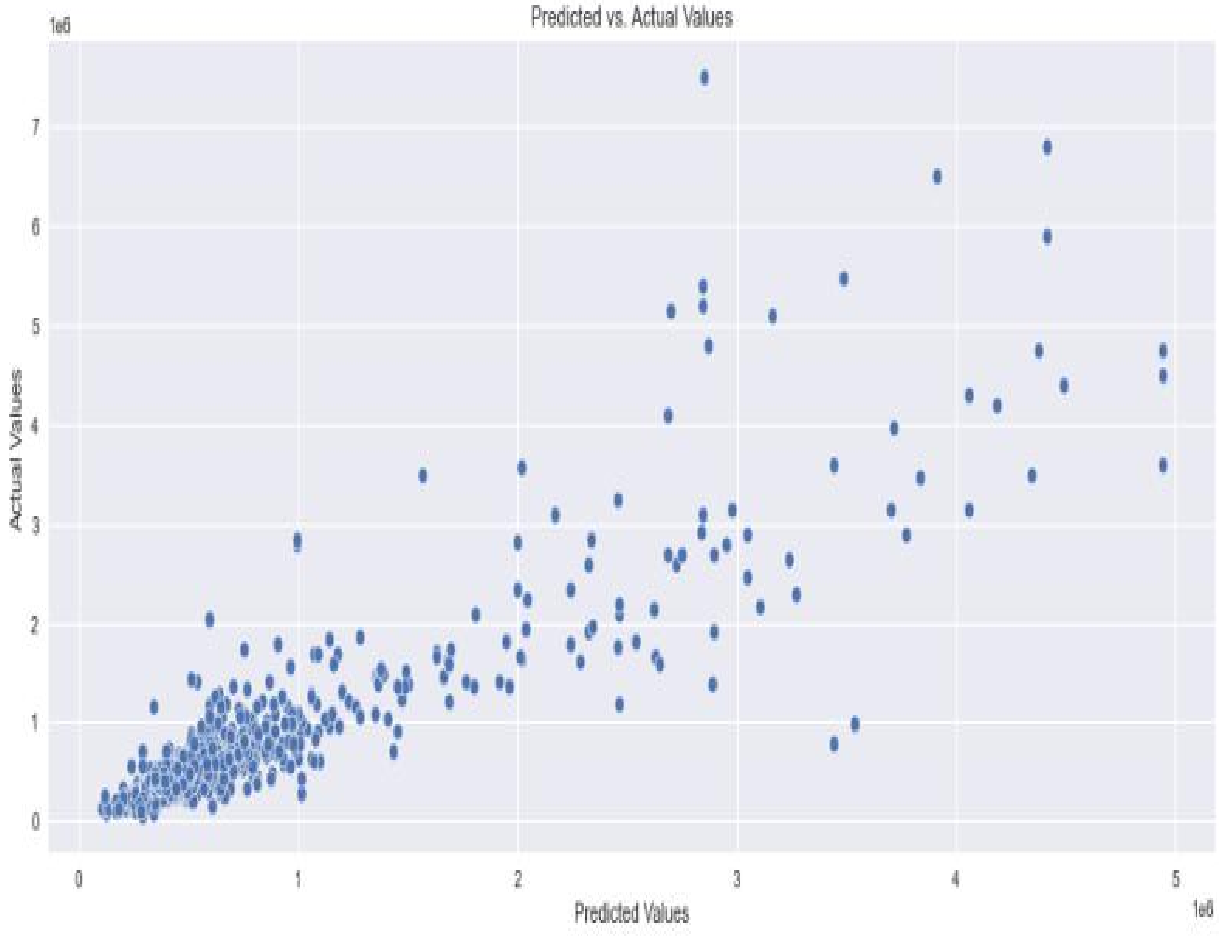
RadomForest Regressor Model

```
R Squared (R2):  90.69261822759161
Mean Squared Error (MSE):  52018880374.54322
Root Mean Squared Error (RMSE):  228076.47922252576
Mean Absolute Error (MAE):  83708.17577001198
```



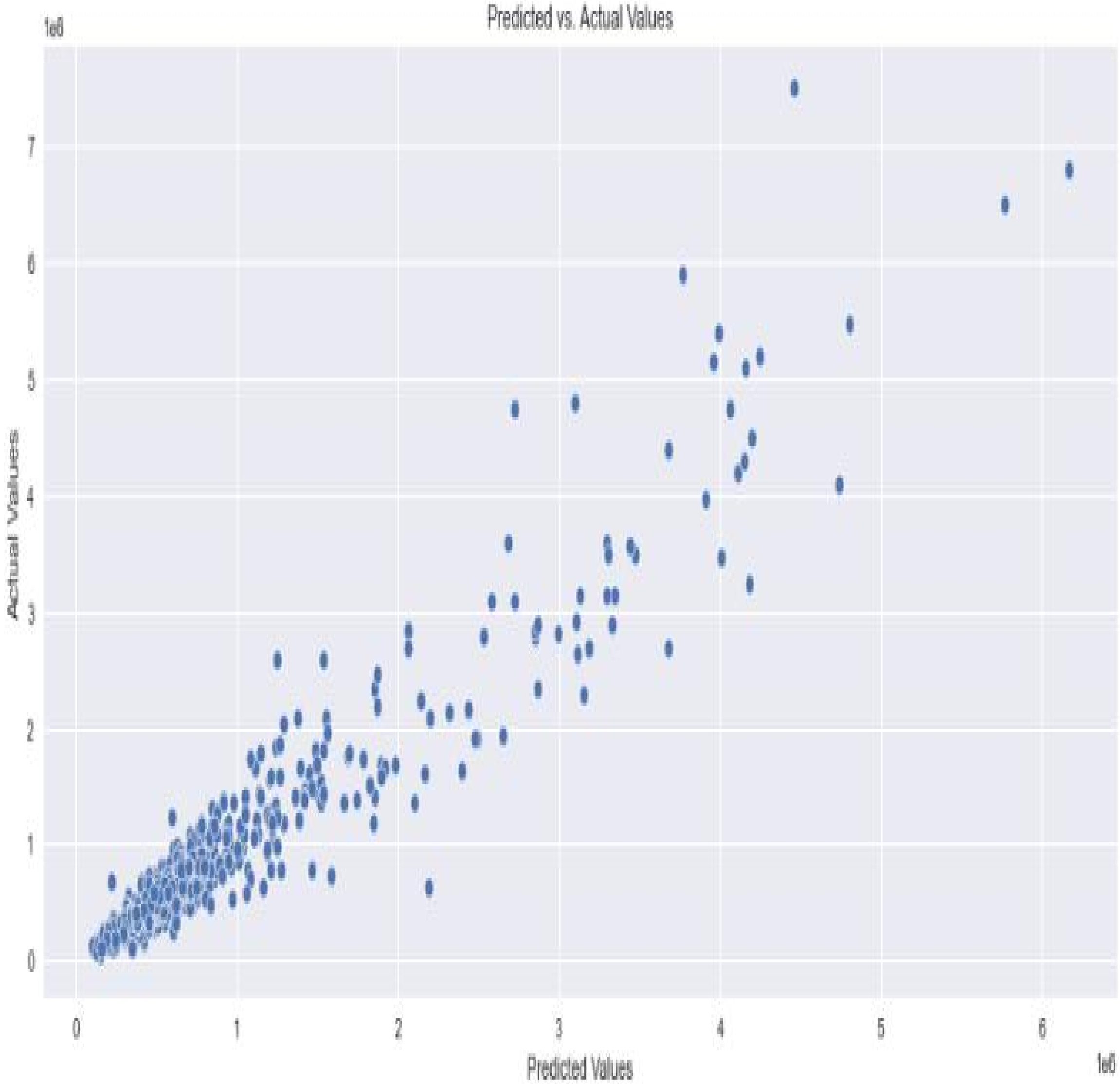
KNeighbors Regressor Model

```
R Squared (R2):  77.65874769809187
Mean Squared Error (MSE):  124865075842.8838
Root Mean Squared Error (RMSE):  353362.52750239917
Mean Absolute Error (MAE):  152638.39074675326
```



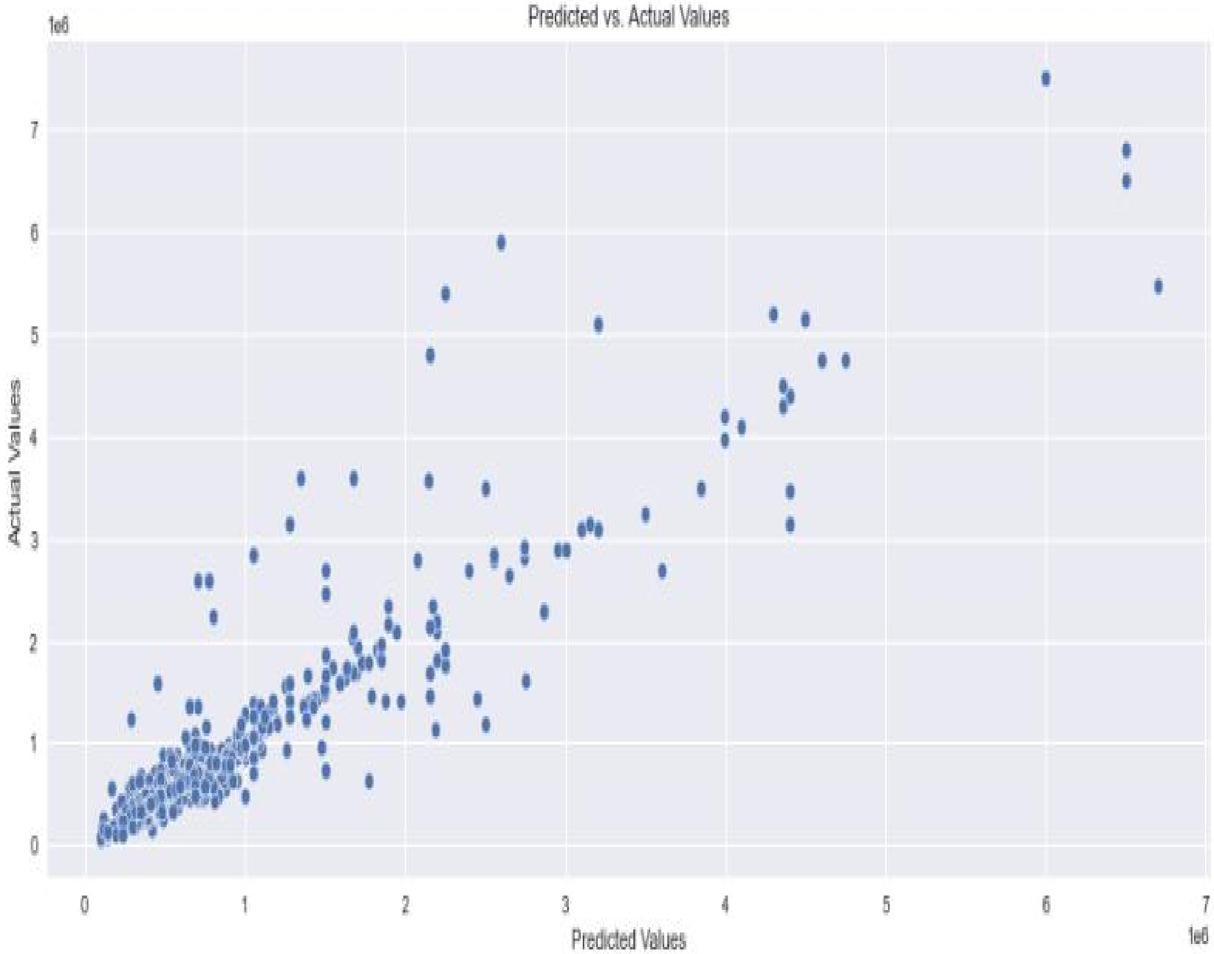
Gradient Boosting Regressor Model

```
R Squared (R2):  90.47935061276328
Mean Squared Error (MSE):  53210831324.315865
Root Mean Squared Error (RMSE):  230674.73057167718
Mean Absolute Error (MAE):  114985.21170524851
```



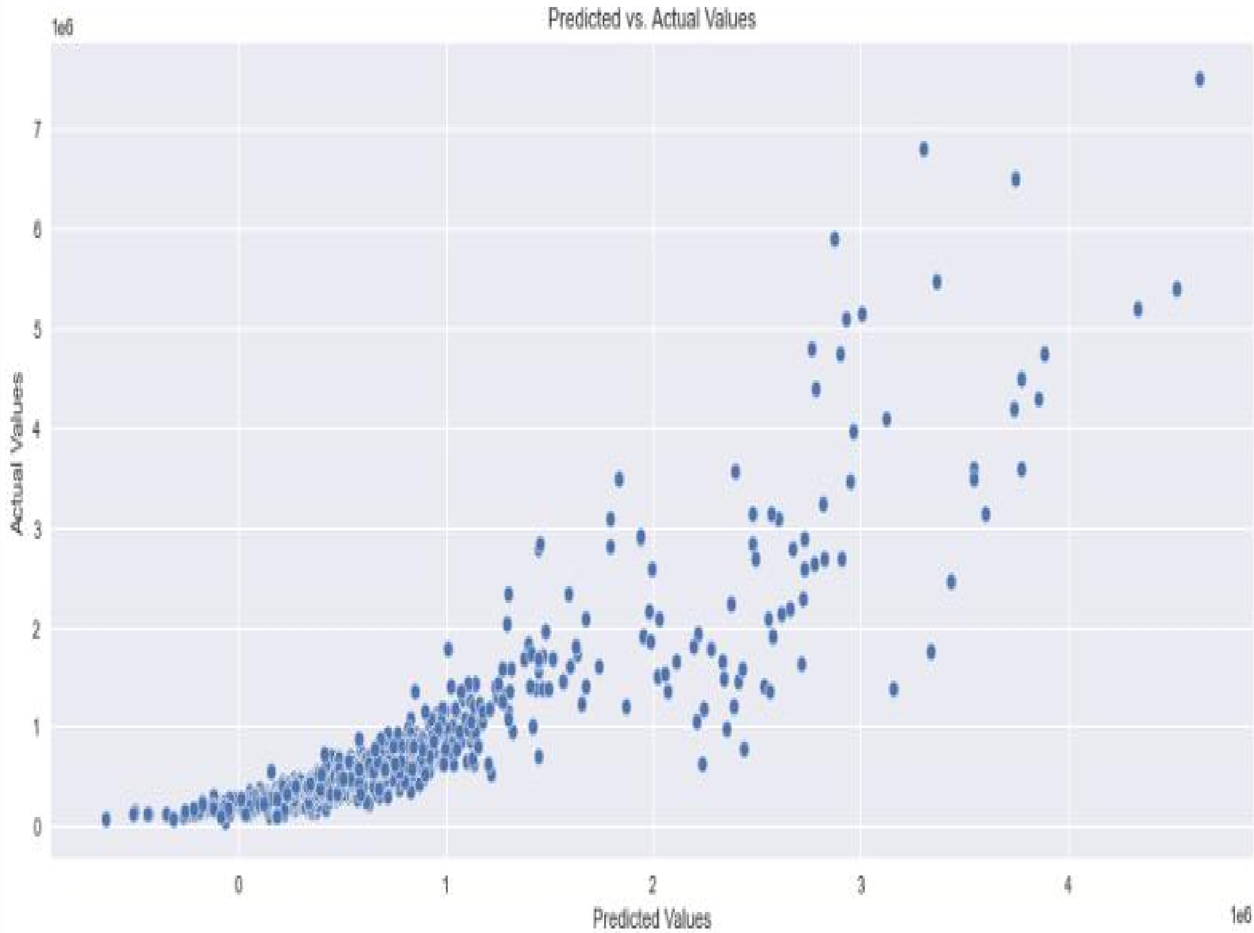
DecisionTreeRegressor Model

```
R Squared (R2):  86.30676373052702
Mean Squared Error (MSE):  76531385180.06169
Root Mean Squared Error (RMSE):  276643.0645797246
Mean Absolute Error (MAE):  96755.43506493507
```



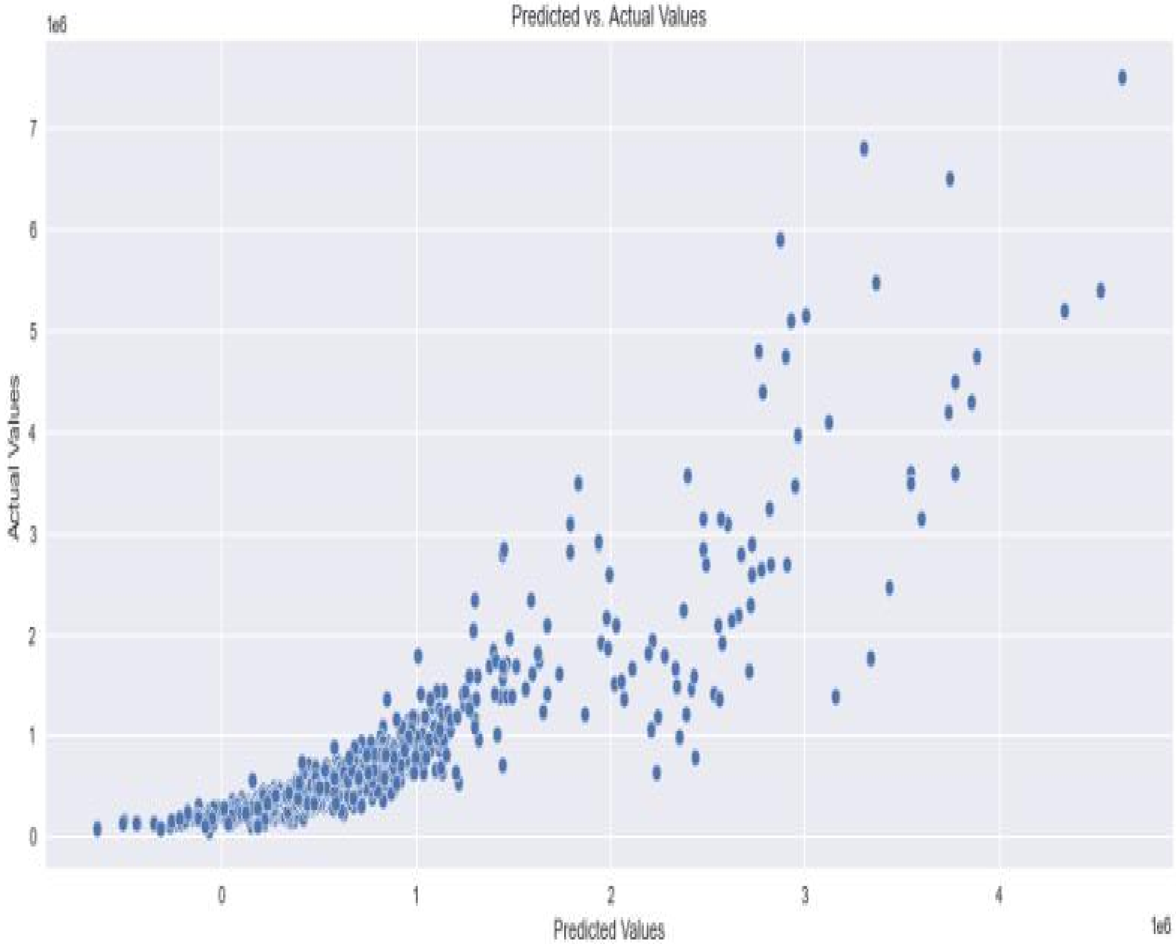
Lasso Model

```
R Squared (R2):  78.53383251107547
Mean Squared Error (MSE):  119974234001.72075
Root Mean Squared Error (RMSE):  346372.96950212604
Mean Absolute Error (MAE):  183979.71633021004
```



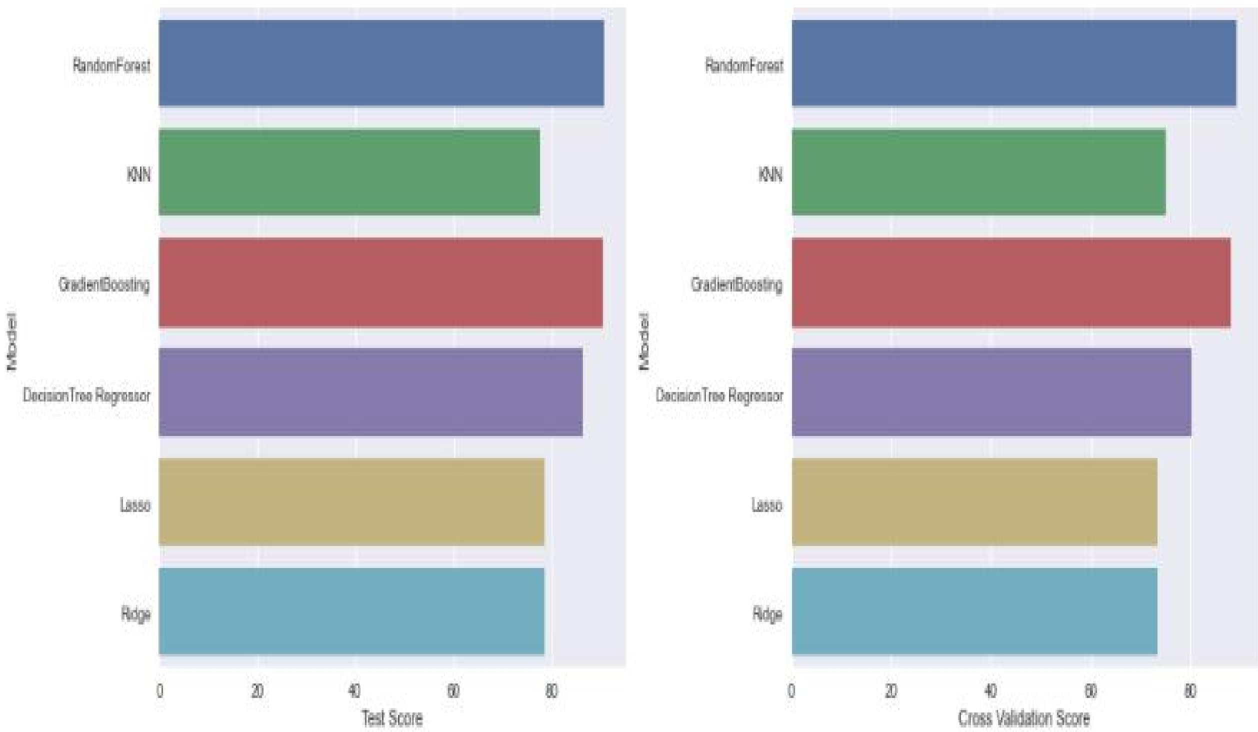
Ridge Model

```
R Squared (R2):  78.53290464518251
Mean Squared Error (MSE):  119979419836.58493
Root Mean Squared Error (RMSE):  346380.4553328391
Mean Absolute Error (MAE):  183970.279697627
```



Overall score of our models

	Model	Training Score	Test Score	Cross Validation Score	Difference
0	RandomForest	98.78516167	90.69261823	89.10247191	1.59014632
1	KNN	83.48265226	77.65874770	74.88136009	2.77738761
2	GradientBoosting	93.40261252	90.47935061	87.94187316	2.53747745
3	DecisionTree Regressor	99.99855838	86.30676373	80.23186179	6.07490194
4	Lasso	73.92046875	78.53383251	73.34839438	5.18543813
5	Ridge	73.92046686	78.53290465	73.34861818	5.18428647



According to performance metric, the random forest has higher R2 score, So this is our best model.

Hyper Tuning

The Hyper parameter tuning is carried out for Random Forest Regressor model.

Because performance metric score is 90.7%.

```
from sklearn.model_selection import GridSearchCV

#parameters
param_grid = {'n_estimators':[50,100],
              'max_features':['auto','sqrt'],
              'max_depth':[4,5,None],'min_samples_split' : [2, 5, 10],
              'criterion':['squared_error','mse'],'min_samples_leaf': [1, 2, 3]}

gridsearch=GridSearchCV(estimator = rf, param_grid = param_grid,cv=5)

gridsearch.fit(x_train,y_train) #training the model
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'criterion': ['squared_error', 'mse'],
                         'max_depth': [4, 5, None],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 3],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [50, 100]})
```

```
print(gridsearch.best_score_, gridsearch.best_params_)  #finding the best parameters
```

```
0.8972633283725265 {'criterion': 'mse', 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 100}
```

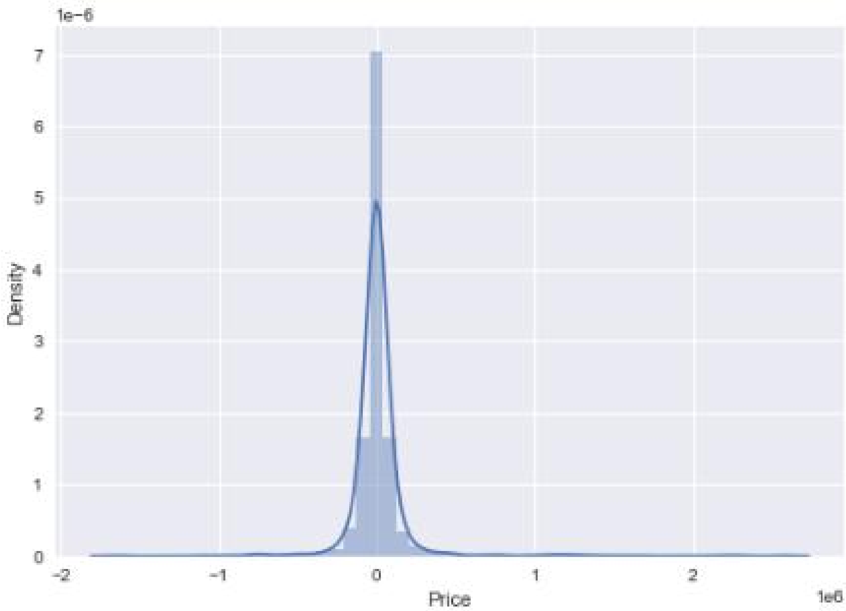
```
Rand_Final = RandomForestRegressor(n_estimators=100,max_features='auto',max_depth=None,criterion='mse',
                                   min_samples_split=2,min_samples_leaf=1)

Rand_Final.fit(x_train,y_train)  #training the model
predictions = Rand_Final.predict(x_test)  #predicting
```

```
R Squared (R2):  0.9082564822524792
Mean Squared Error (MSE):  51275376808.92997
Root Mean Squared Error (RMSE):  226440.66951175084
Mean Absolute Error (MAE):  83110.16246043987
```

Hyper parameter Tuning performance is carried out for Random Forest Regressor:

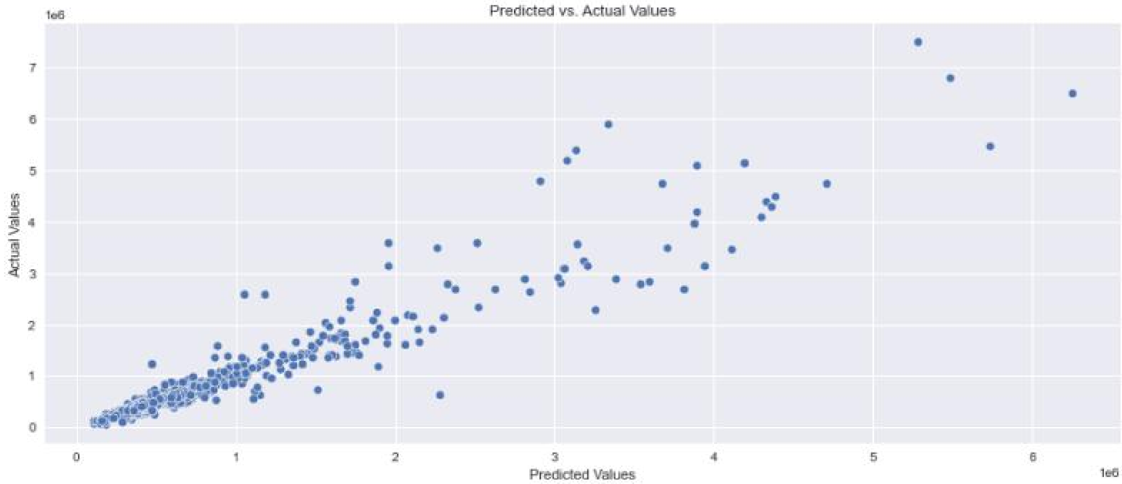
Hyper parameter Tuning i.e.,R2 score = 90.83% respectively. Finally, Random Forest Regressor is best model for these dataset.

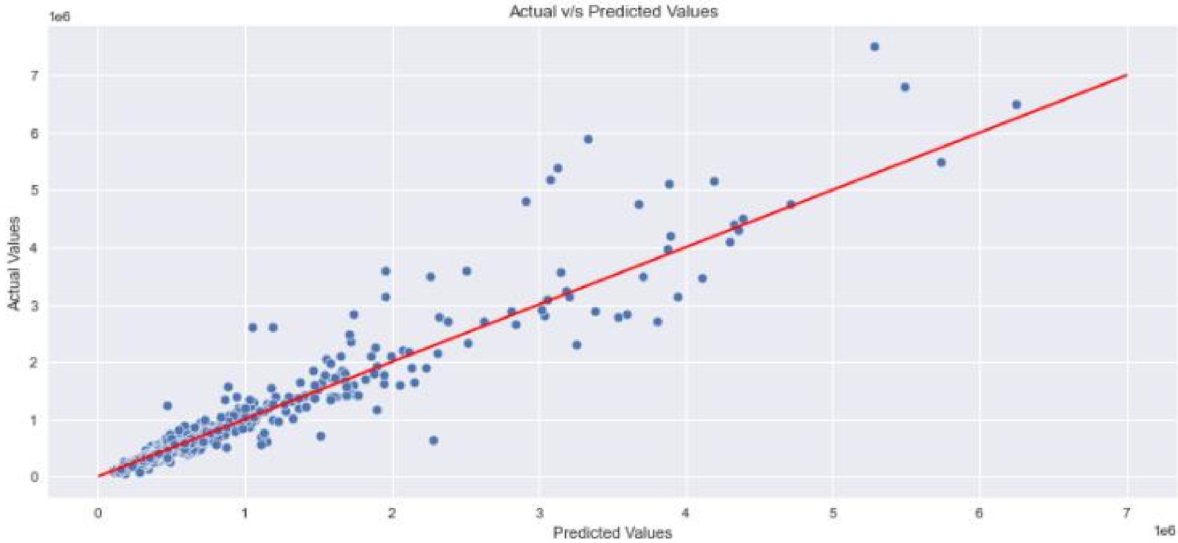
```
#Let's again plot the difference between the y_test price and our model predicted price

sns.distplot(y_test-predictions)
plt.show()
```



- We could now see there is a slight change and now we have most data with zero difference

```python
#plot predicted vs. actual values

plt.figure(figsize=[15,6])
sns.scatterplot(x=predictions, y=y_test)
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Predicted vs. Actual Values')
plt.show()
```





The above graph indicates that most of the "Actual and Predicted" values are quite close to each other

After hyper tuning , Our model score is now increased by 0.00133029998% of acuracy score

Hence, our model is ready with 90.83 % of Acuracy Score

Saving the Model

Saving the model for future prediction:

```
#Let's save our model for future predictions

import joblib

joblib.dump(Rand_final,'Used_Car_Price_Prediction.obj')
```

```
['Used_Car_Price_Prediction.obj']
```

Loading the saved model to predict the Used_Car_Price

```
#Load the saved model

loaded_model = joblib.load(open('Used_Car_Price_Prediction.obj','rb'))
```

```
# Predict the Labels using the reloaded Model
Predictions = loaded_model.predict(x_test)

Predictions
```

```
array([439616.29780208, 306203.08155818, 398503.68258777, ...,
       623758.80099066, 583180.03714536, 812343.2194945 ])
```

```
#create dataframe of actual and predicted values

list_of_tuples = list(zip(y_test, Predictions))
Used_Car_Price = pd.DataFrame(list_of_tuples, columns = ['Actual', 'Predicted'])
```

```
#Saving the dataframe of the actual v/s predicted values as a csv file

Used_Car_Price.to_csv('Predicted_car_Prices.csv')
```

CONCLUSION

In this paper, we built several regression models to predict the selling price of cars by given some of the cars features. We evaluated and compared each model to determine the one with highest performance. We also looked at how some models rank the features according to their importance. In this paper, we followed the data science process starting with getting the data, then cleaning and pre-processing the data, followed by exploring the data and building models, then evaluating the results.

As a recommendation, we advise to use this model (or a version of it trained with more recent data) by car market who want to get an idea about car price. The model can be used also with datasets that covered areas provided that they contain the same features. We also suggest that people take into consideration the features that were deemed as most important as seen in the previous section; this might help them estimate the car price is better.

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

The key findings are we have to study the data very clearly so that we are able to decide which data are relevant for our findings. The techniques that I have used are heatmap, SimpleImputer,LabelEncoder etc.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

This project has demonstrated the importance of sampling effectively, modelling and predicting data.

Through different powerful tools of visualization we were able to analyse and interpret different hidden insights about the data.

Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

The data was improper scaled, so we scaled it to a single scale using sklearns's package StandardScaler.

The columns were skewed due to presence of outliers which we handled through winsorization technique.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

The scope for future work is to collect as many data as we can so that the model can be built more efficiently.

Interpretation of the Results

In the visualization part, I have seen how my data looks like using heatmap, boxplot, distribution plots, histogram etc.
In the pre-processing part, I have cleaned my data using many methods like SimpleImputer,LabelEncoder etc.
In the modelling part, I have designed our model using algorithm like Random Forest Regressor.
The accuracy , Mean Absolute Error, Mean Squared Error, Root Mean Absolute Error are achieved for the model.