FLIP ROBO

CAR PRICE PREDICTION

Submitted by:

VANISREE P.G

ACKNOWLEDGMENT

First I would like to thank the Almighty for his wonderful presence with me throughout this project and helped me to make it as a successful one.

For my internship I had the pleasure of working at FILP ROBO Was a great chance for acquired knowledge, personal and Professional development. I extend whole hearted thanks to FILP ROBO under whom I worked and learned a lot and for enlightening me with their knowledge and experience to grow with the corporate working

This is a great pleasure to express my deep sense of gratitude and thanks to SME for his valuable ideas, instantaneous help, effective support and continued encouragement which enabled for the successful completion of the project. I also like to thank the data trained mentors and Technical team members for helping me with technical queries.

These are the following website which I referred for the references.

1. https://scikit-learn.org/
2. https://kaggle.com
3. www.google.com
4. www.geeksforgeeks.org

INTRODUCTION

- Business Problem Framing

A car price prediction has been a high interest research area, as it requires noticeable effort and knowledge of the field expert. Considerable number of distinct attributes are examined for the reliable and accurate prediction.

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.

- **Motivation for the Problem Undertaken**

To understand real world problems where Machine Learning and Data Analysis can be applied to help to predict the prices in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data

Accurate car price prediction involves expert knowledge, because price usually depends on many distinctive features and factors. Typically, most significant ones are brand and model, age, horsepower and mileage. The fuel type used in the car as well as fuel consumption per mile highly affect price of a car due to a frequent changes in the price of a fuel. Different features like exterior color, door number, type of transmission, dimensions, safety, air condition, interior, whether it has navigation or not will also influence the car price.

The price of a new car in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But, due to the increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features

Analytical Problem Framing

- **Mathematical/ Analytical Modeling**

the whole research process various mathematical, statistical and analytics modelling has been done. There has been reduction of the columns because few of them was not necessary for the problem solving. And few of them was removed due to very less correlation with dependent variable. Since the dataset contains a lot of features hence feature selection has been also done.

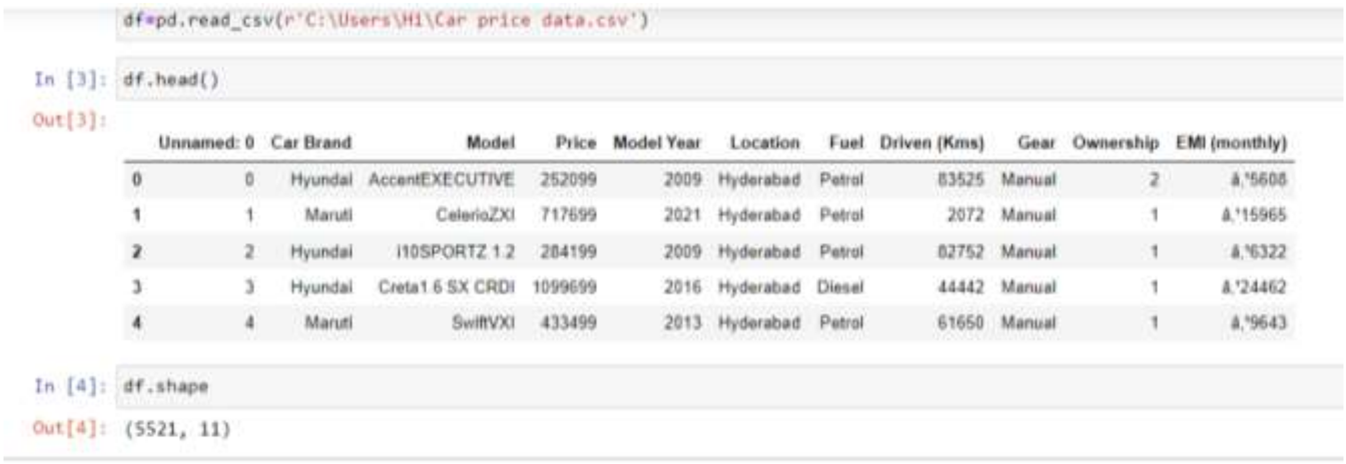
- **Data Sources**

Data is collected from a local web portal for selling and buying cars .The data is scraped from the cars24 website The data descriptions are as follow:- (5521, 11) rows and columns.To predict car prices using Regression. I will start by importing all the necessary libraries that we need for this task and import the dataset.

1) Importing libraries

2) Importing the dataset

They are totally 5521 rows and 11 columns in a train.csv file. Our target is to find the insights of the data and to do thorough data analysis.



```
        df=pd.read_csv(r'C:\Users\H1\Car price data.csv')

In [3]: df.head()
Out[3]:
```

	Unnamed: 0	Car Brand	Model	Price	Model Year	Location	Fuel	Driven (Kms)	Gear	Ownership	EMI (monthly)
0	0	Hyundai	AccentEXECUTIVE	252099	2009	Hyderabad	Petrol	83525	Manual	2	â‚¹5608
1	1	Maruti	CelerioZXI	717699	2021	Hyderabad	Petrol	2072	Manual	1	â‚¹15965
2	2	Hyundai	I10SPORTZ 1.2	284199	2009	Hyderabad	Petrol	82752	Manual	1	â‚¹6322
3	3	Hyundai	Creta1.6 SX CRDI	1099699	2016	Hyderabad	Diesel	44442	Manual	1	â‚¹24462
4	4	Maruti	SwiftVXI	433499	2013	Hyderabad	Petrol	61650	Manual	1	â‚¹9643

```
In [4]: df.shape
Out[4]: (5521, 11)
```

- ## Data Pre-processing Done

The raw data is taken and performed various steps to reduce skewness, outlier, class imbalance and scaling. There were null value was present and removed the values from the dataset. Many outlier removal and skewness removal methods are tested and best method Is chosen in order to prevent data loss.

· The dataset contains 5521 rows and 11 columns

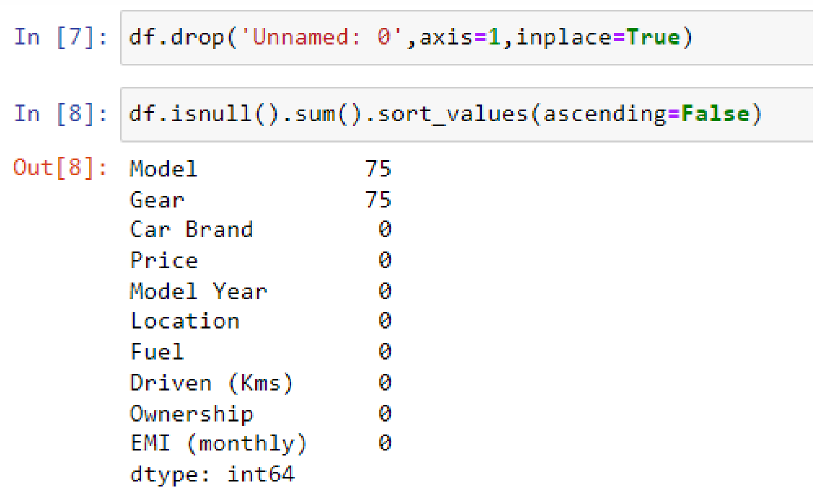
· Price is our dependent variable.

· We created new features from old ones.

· There are no null values in the dataset.

· Removed empty cells

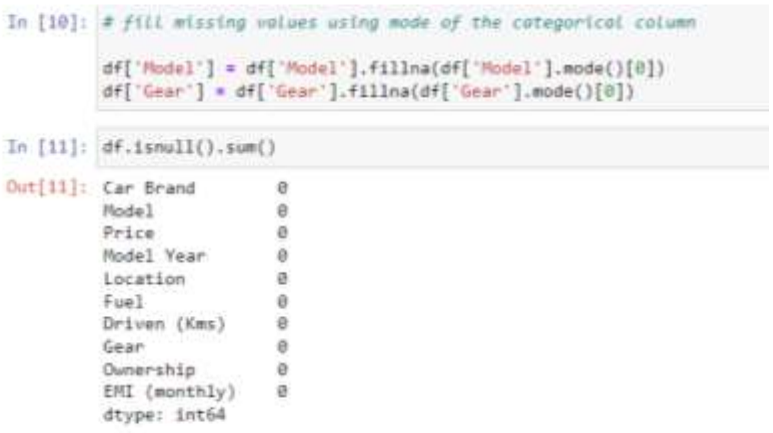
a) **Checking missing value from the data set.**



```
In [7]: df.drop('Unnamed: 0',axis=1,inplace=True)

In [8]: df.isnull().sum().sort_values(ascending=False)

Out[8]: Model            75
        Gear             75
        Car Brand         0
        Price             0
        Model Year        0
        Location          0
        Fuel              0
        Driven (Kms)      0
        Ownership         0
        EMI (monthly)     0
        dtype: int64
```

➢ There were null value was present in the dataset but there are some outliers which also get too removed.

- **Missing values**



```
In [10]: # fill missing values using mode of the categorical column

         df['Model'] = df['Model'].fillna(df['Model'].mode()[0])
         df['Gear'] = df['Gear'].fillna(df['Gear'].mode()[0])

In [11]: df.isnull().sum()

Out[11]: Car Brand        0
         Model            0
         Price            0
         Model Year       0
         Location         0
         Fuel             0
         Driven (Kms)     0
         Gear             0
         Ownership        0
         EMI (monthly)    0
         dtype: int64
```

➢ We drop large number of missing value columns. some other columns have null values these are Filled by mode.

b) **CORRELATION**

Correlation between all the columns in the datasets. In the correlation Heat map, we have the following observations:

1.Driven(kms) has 80 percent correlation with target column which can be considered as a good bond.

2.Location has 35 percent correlation with target column which can be considered as a good bond.

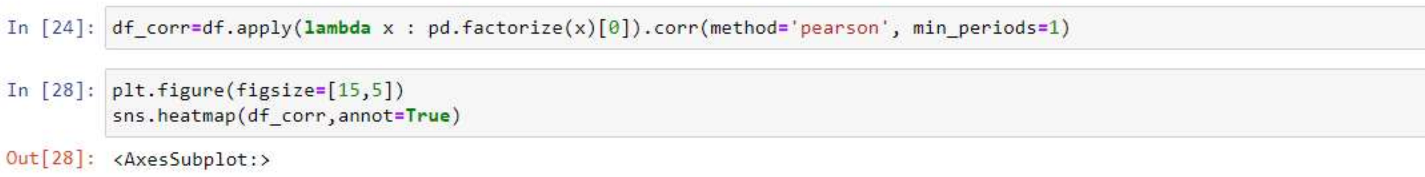
3.Model since the has 12 percent correlation with target column which can be

considered as a good bond.

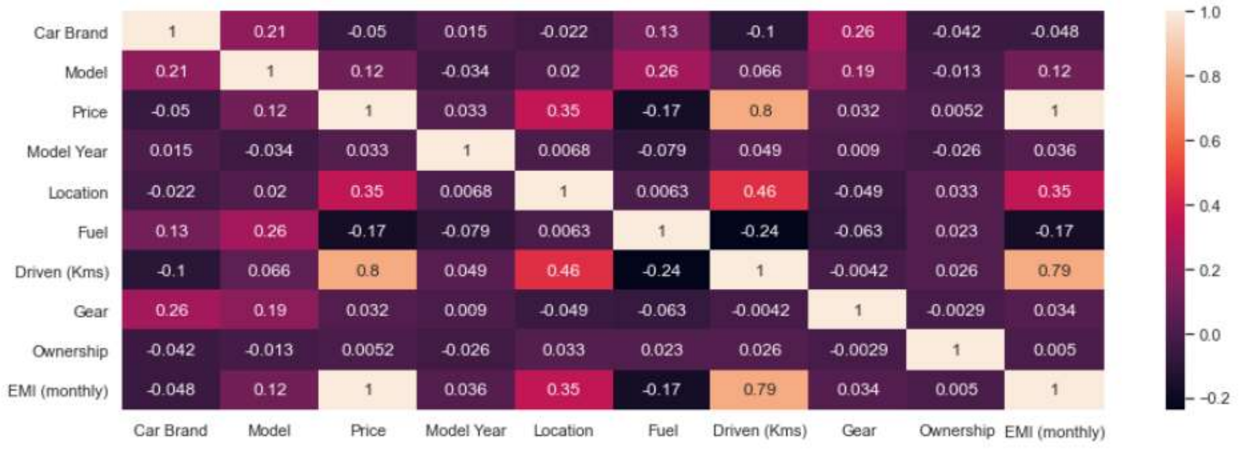
4.Model year has 3.3 percent correlation with target column which can be

considered as a bad bond

5.Gear has 3.2 percent correlation with target column which can be considered as a bad bond.



```
In [24]: df_corr=df.apply(lambda x : pd.factorize(x)[0]).corr(method='pearson', min_periods=1)

In [28]: plt.figure(figsize=[15,5])
         sns.heatmap(df_corr,annot=True)

Out[28]: <AxesSubplot:>
```



	Car Brand	Model	Price	Model Year	Location	Fuel	Driven (Kms)	Gear	Ownership	EMI (monthly)
Car Brand	1	0.21	-0.05	0.015	-0.022	0.13	-0.1	0.26	-0.042	-0.048
Model	0.21	1	0.12	-0.034	0.02	0.26	0.066	0.19	-0.013	0.12
Price	-0.05	0.12	1	0.033	0.35	-0.17	0.8	0.032	0.0052	1
Model Year	0.015	-0.034	0.033	1	0.0068	-0.079	0.049	0.009	-0.026	0.036
Location	-0.022	0.02	0.35	0.0068	1	0.0063	0.46	-0.049	0.033	0.35
Fuel	0.13	0.26	-0.17	-0.079	0.0063	1	-0.24	-0.063	0.023	-0.17
Driven (Kms)	-0.1	0.066	0.8	0.049	0.46	-0.24	1	-0.0042	0.026	0.79
Gear	0.26	0.19	0.032	0.009	-0.049	-0.063	-0.0042	1	-0.0029	0.034
Ownership	-0.042	-0.013	0.0052	-0.026	0.033	0.023	0.026	-0.0029	1	0.005
EMI (monthly)	-0.048	0.12	1	0.036	0.35	-0.17	0.79	0.034	0.005	1

- ## Data Inputs- Logic- Output Relationships

  The input data contains 5521 rows and 11 columns.Predictor variable are

  - Car Brand, Model , Model Year, Location, Fuel, Driven (Kms), Gear, Ownership, EMI (monthly)
  - Target variable is Price of the car

- ## Hardware and Software Requirements and Tools Used

  Hardware used for doing the project is a 'Laptop' with high end specification and stable internet connection .while coming to the software part I had used 'python jupyter notebook' for do my python program and data analysis.

Excel file and Microsoft excel are required for the data handling. In jupyter notebook I had imported lot of python libraries are carried to this project.

- 1.Pandas-a library which is used to read the data ,visualisation and analysis of data.
- 2.Numpy-used for working with array and various mathematical operations in python.
- 3.Seaborn- visualization for plotting different type of plot.
- 4.Matplotlib- It provides an object-oriented API for embedding plots into applications .

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches**

Regression Model with following algorithms

- Linear Regression
- DecisionTreeRegressor
- Random forest regressor
- Gradient Boosting Regressor
- Ridge

Evaluation metrics

- Mean square error
- Mean absolute error
- R2 score
- Root Mean Squared Error

- Testing of Identified Approaches (Algorithms)

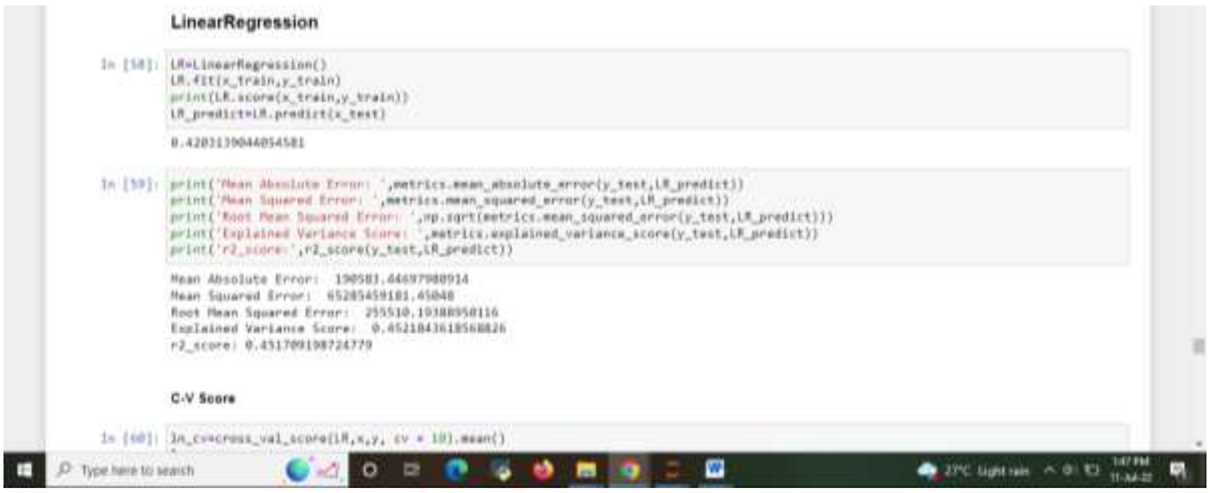
Listing down all the algorithms used for the training and testing.

- LR=LinearRegression()
- DT=DecisionTreeRegressor()
- GBR=GradientBoostingRegressor()
- rf=RandomForestRegressor()

- **Run and Evaluate selected models**

Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

Train-test data splits were conducted. In this situation, we split the data into training and test sets, then fit candidate models on the training set, evaluate and select them on the test set.



LinearRegression

```
In [58]: LR=LinearRegression()
         LR.fit(x_train,y_train)
         print(LR.score(x_train,y_train))
         LR_predict=LR.predict(x_test)

0.42031390440545 81

In [59]: print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,LR_predict))
         print('Mean Squared Error: ',metrics.mean_squared_error(y_test,LR_predict))
         print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,LR_predict)))
         print('Explained Variance Score: ',metrics.explained_variance_score(y_test,LR_predict))
         print('r2_score: ',r2_score(y_test,LR_predict))

Mean Absolute Error:  190583.44697980914
Mean Squared Error:  65285459181.45048
Root Mean Squared Error:  255510.19388950116
Explained Variance Score:  0.45218436185688 26
r2_score: 0.451709198724779
```

C-V Score

```
In [60]: ln_cv=cross_val_score(LR,x,y, cv = 10).mean()
```

1.Linear Regression

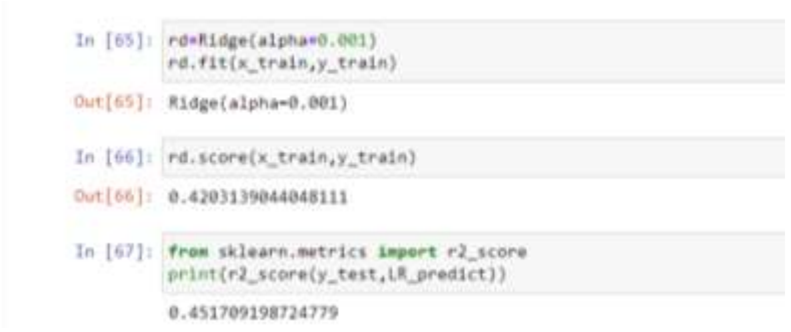
Ridge and lasso regression allow you to regularize ("shrink") coefficients.

Lasso-Rigid regression

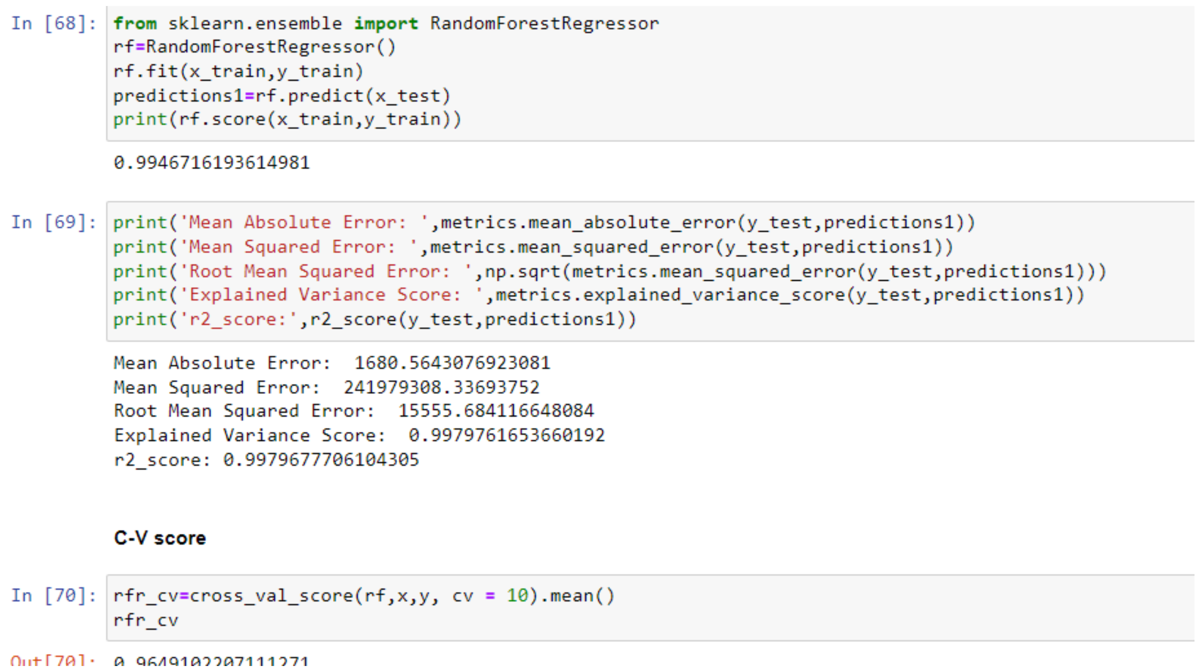
```
In [62]: from sklearn.linear_model import Lasso,Ridge
         ls=Lasso(alpha=0.001)
         ls.fit(x_train,y_train)
         ls.score(x_train,y_train)

Out[62]: 0.4203139044054629
```

```
In [63]: import numpy as np
         from sklearn import metrics
         print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,LR_predict))
         print('Mean Squared Error: ',metrics.mean_squared_error(y_test,LR_predict))
         print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,LR_predict)))
         print('Explained Variance Score: ',metrics.explained_variance_score(y_test,LR_predict))

         Mean Absolute Error:  190583.44697980914
         Mean Squared Error:  65285459181.45048
         Root Mean Squared Error:  255510.19388950116
         Explained Variance Score:  0.4521843618568826
```

```
In [64]: from sklearn.metrics import r2_score
         print(r2_score(y_test,LR_predict))

         0.451709198724779
```



```
In [65]: rd=Ridge(alpha=0.001)
         rd.fit(x_train,y_train)

Out[65]: Ridge(alpha=0.001)

In [66]: rd.score(x_train,y_train)

Out[66]: 0.4203139044048111

In [67]: from sklearn.metrics import r2_score
         print(r2_score(y_test,LR_predict))

         0.451709198724779
```

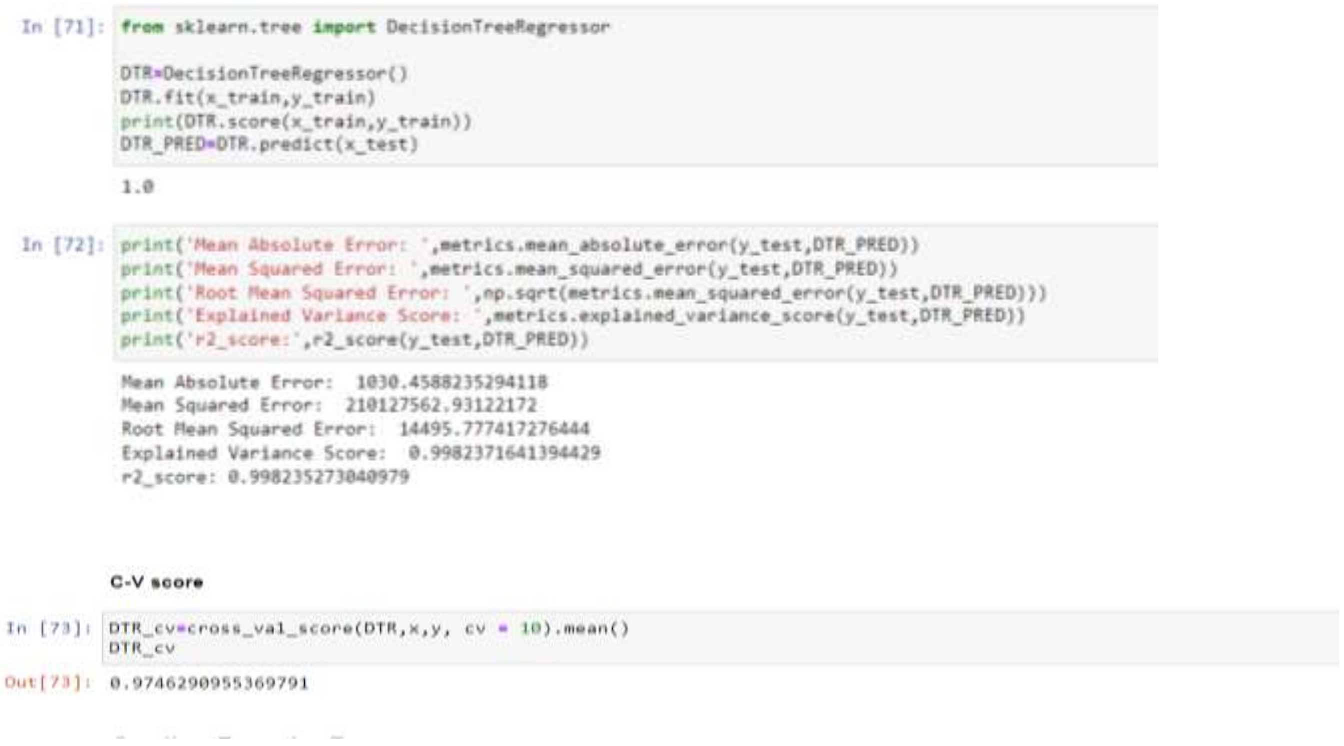


```
In [68]: from sklearn.ensemble import RandomForestRegressor
         rf=RandomForestRegressor()
         rf.fit(x_train,y_train)
         predictions1=rf.predict(x_test)
         print(rf.score(x_train,y_train))

         0.9946716193614981

In [69]: print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,predictions1))
         print('Mean Squared Error: ',metrics.mean_squared_error(y_test,predictions1))
         print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,predictions1)))
         print('Explained Variance Score: ',metrics.explained_variance_score(y_test,predictions1))
         print('r2_score:',r2_score(y_test,predictions1))

         Mean Absolute Error:  1680.5643076923081
         Mean Squared Error:  241979308.33693752
         Root Mean Squared Error:  15555.684116648084
         Explained Variance Score:  0.9979761653660192
         r2_score: 0.9979677706104305
```

C-V score

```
In [70]: rfr_cv=cross_val_score(rf,x,y, cv = 10).mean()
         rfr_cv

Out[70]: 0.9649102207111271
```

2.RandomForestRegressor

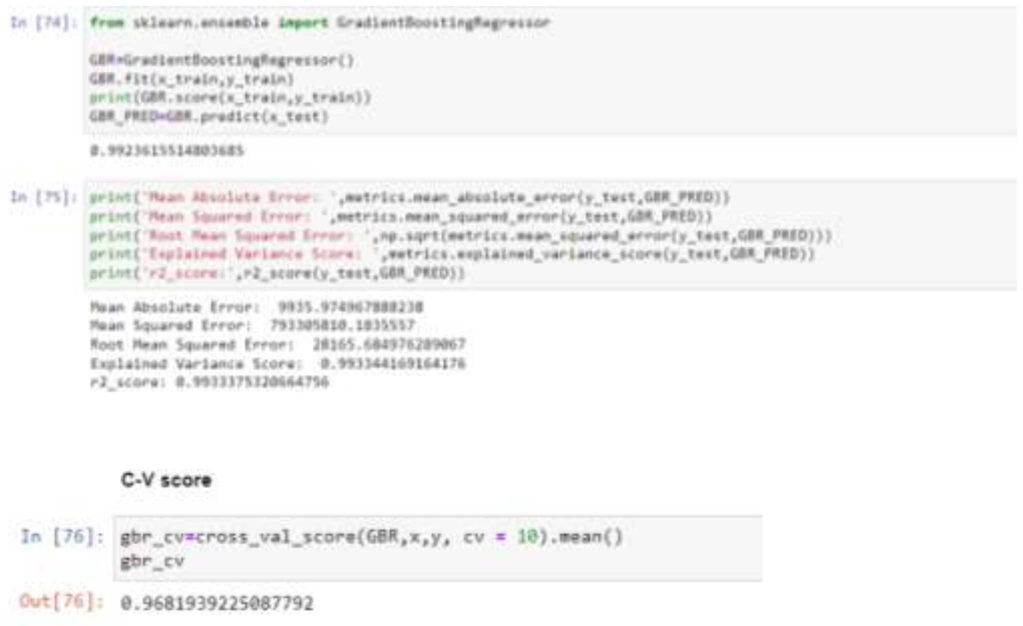


```
In [71]: from sklearn.tree import DecisionTreeRegressor

         DTR=DecisionTreeRegressor()
         DTR.fit(x_train,y_train)
         print(DTR.score(x_train,y_train))
         DTR_PRED=DTR.predict(x_test)

         1.0

In [72]: print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,DTR_PRED))
         print('Mean Squared Error: ',metrics.mean_squared_error(y_test,DTR_PRED))
         print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,DTR_PRED)))
         print('Explained Variance Score: ',metrics.explained_variance_score(y_test,DTR_PRED))
         print('r2_score:',r2_score(y_test,DTR_PRED))

         Mean Absolute Error:  1030.4588235294118
         Mean Squared Error:  210127562.93122172
         Root Mean Squared Error:  14495.777417276444
         Explained Variance Score:  0.9982371641394429
         r2_score: 0.998235273040979
```

C-V score

```
In [73]: DTR_cv=cross_val_score(DTR,x,y, cv = 10).mean()
         DTR_cv

Out[73]: 0.9746290955369791
```

3. DecisionTreeRegressor

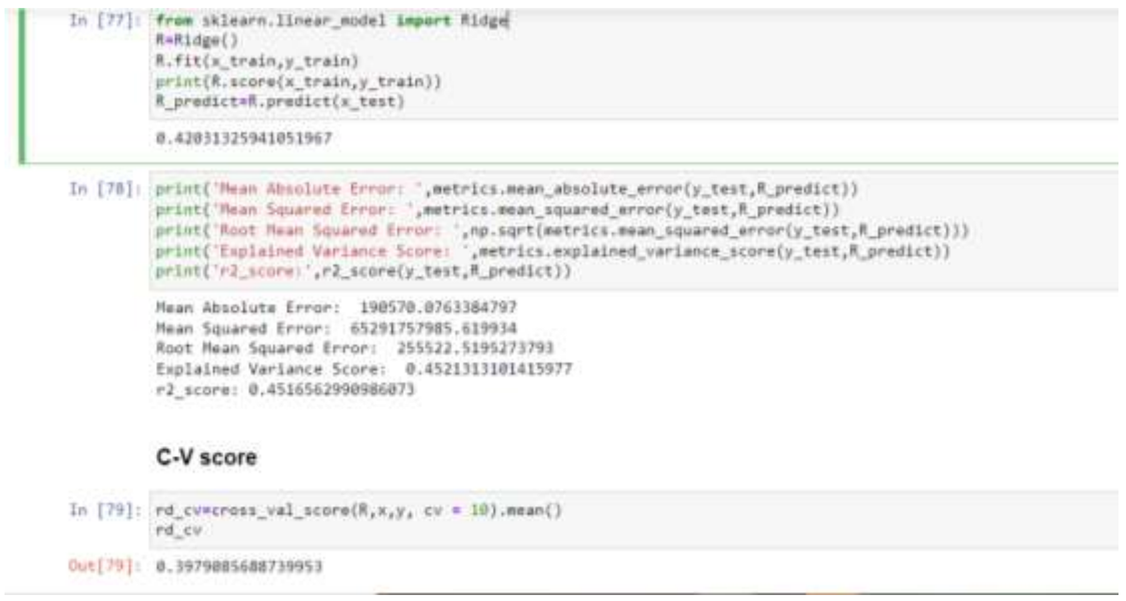


```
In [74]: from sklearn.ensemble import GradientBoostingRegressor

         GBR=GradientBoostingRegressor()
         GBR.fit(x_train,y_train)
         print(GBR.score(x_train,y_train))
         GBR_PRED=GBR.predict(x_test)

         0.9923615514803685

In [75]: print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,GBR_PRED))
         print('Mean Squared Error: ',metrics.mean_squared_error(y_test,GBR_PRED))
         print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,GBR_PRED)))
         print('Explained Variance Score: ',metrics.explained_variance_score(y_test,GBR_PRED))
         print('r2_score:',r2_score(y_test,GBR_PRED))

         Mean Absolute Error:  9935.974967888238
         Mean Squared Error:  793305810.1835557
         Root Mean Squared Error:  28165.684976289067
         Explained Variance Score:  0.993344169164176
         r2_score: 0.9933375320664756
```

C-V score

```
In [76]: gbr_cv=cross_val_score(GBR,x,y, cv = 10).mean()
         gbr_cv

Out[76]: 0.9681939225087792
```

4.GradientBoostingRegressor



```
In [77]: from sklearn.linear_model import Ridge
         R=Ridge()
         R.fit(x_train,y_train)
         print(R.score(x_train,y_train))
         R_predict=R.predict(x_test)

         0.42031325941051967

In [78]: print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,R_predict))
         print('Mean Squared Error: ',metrics.mean_squared_error(y_test,R_predict))
         print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,R_predict)))
         print('Explained Variance Score: ',metrics.explained_variance_score(y_test,R_predict))
         print('r2_score:',r2_score(y_test,R_predict))

         Mean Absolute Error:  190570.0763384797
         Mean Squared Error:  65291757985.619934
         Root Mean Squared Error:  255522.5195273793
         Explained Variance Score:  0.4521313101415977
         r2_score: 0.4516562990986073
```

C-V score

```
In [79]: rd_cv=cross_val_score(R,x,y, cv = 10).mean()
         rd_cv

Out[79]: 0.3979085688739953
```

5.**Ridge regression**

Evaluating the model accuracy is an essential part of the process of creating machine learning models to describe how well the model is performing in its predictions. The MSE, MAE, and RMSE metrics are mainly used to evaluate the prediction error rates and model performance in regression analysis.

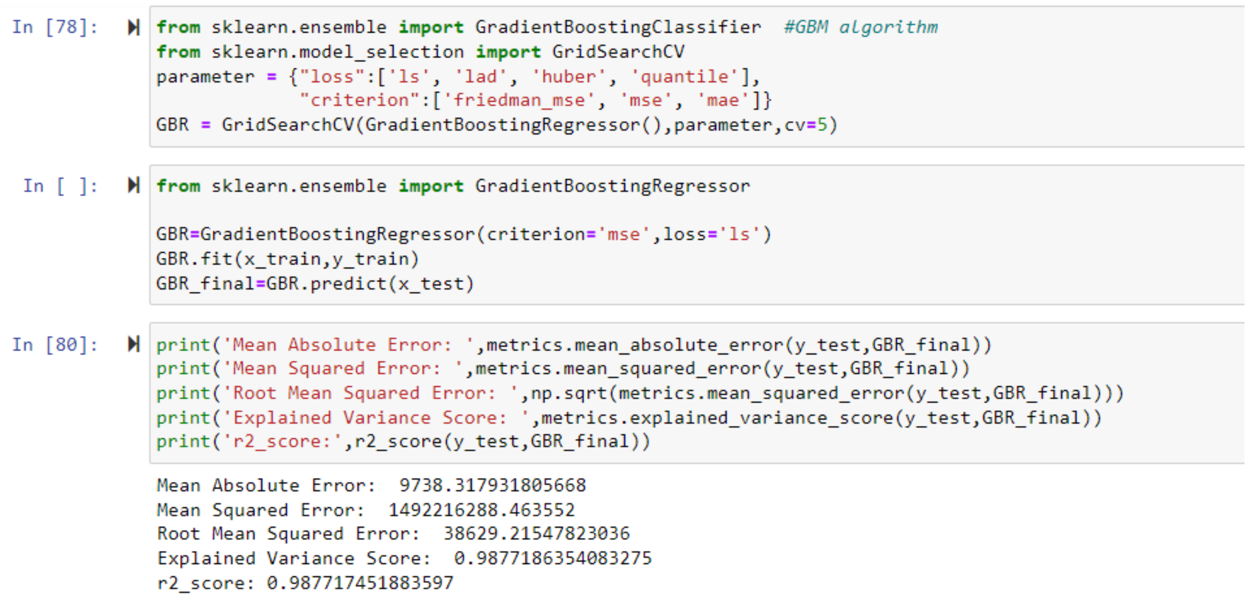
- MAE (Mean absolute error) represents the difference between the original and predicted values extracted by averaged the absolute difference over the data set.
- MSE (Mean Squared Error) represents the difference between the original and predicted values extracted by squared the average difference over the data set.
- RMSE (Root Mean Squared Error) is the error rate by the square root of MSE.

After evaluating the model based on MAE, MSE, RMSE, EVS, R2 SCORE the best model choose for hyper parameter tuning are Gradient Boosting Regressor, DecisionTreeRegressor.

1. GradientBoostingRegressor
2. DecisionTreeRegressor

Hyper parametertuning

1.GradientBoostingRegressor

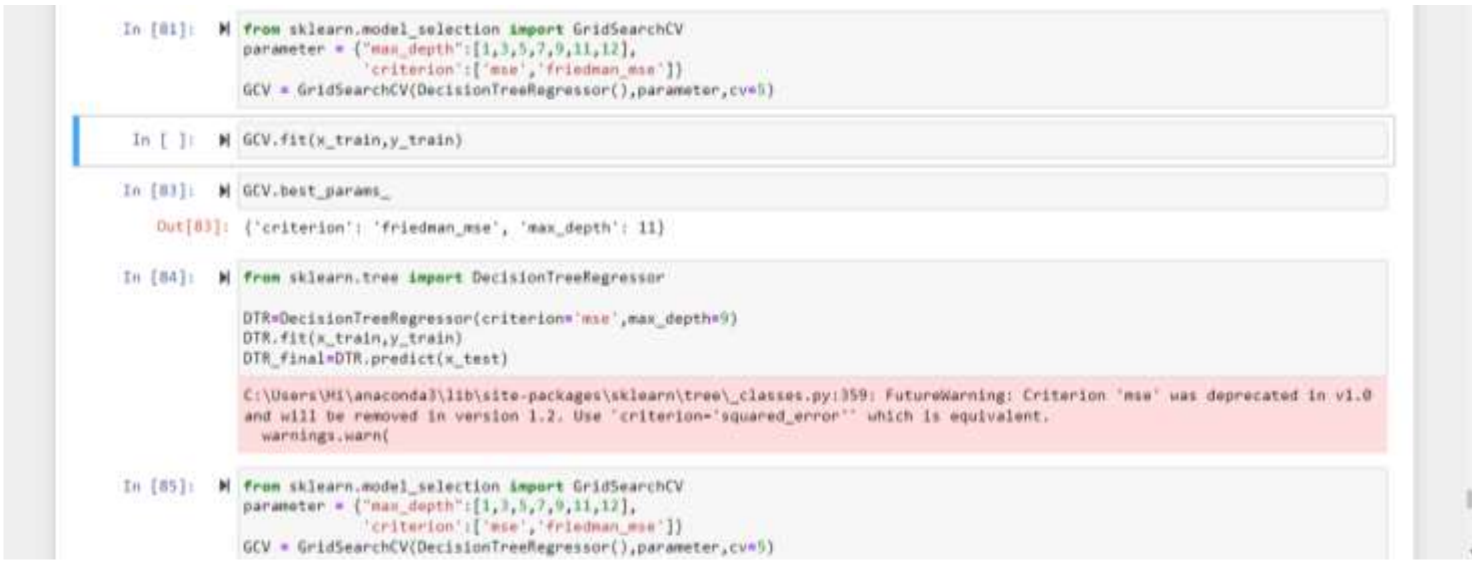


```
In [78]:  from sklearn.ensemble import GradientBoostingClassifier  #GBM algorithm
          from sklearn.model_selection import GridSearchCV
          parameter = {"loss":['ls', 'lad', 'huber', 'quantile'],
                       "criterion":['friedman_mse', 'mse', 'mae']}
          GBR = GridSearchCV(GradientBoostingRegressor(),parameter,cv=5)
```

```
In [ ]:   from sklearn.ensemble import GradientBoostingRegressor

          GBR=GradientBoostingRegressor(criterion='mse',loss='ls')
          GBR.fit(x_train,y_train)
          GBR_final=GBR.predict(x_test)
```

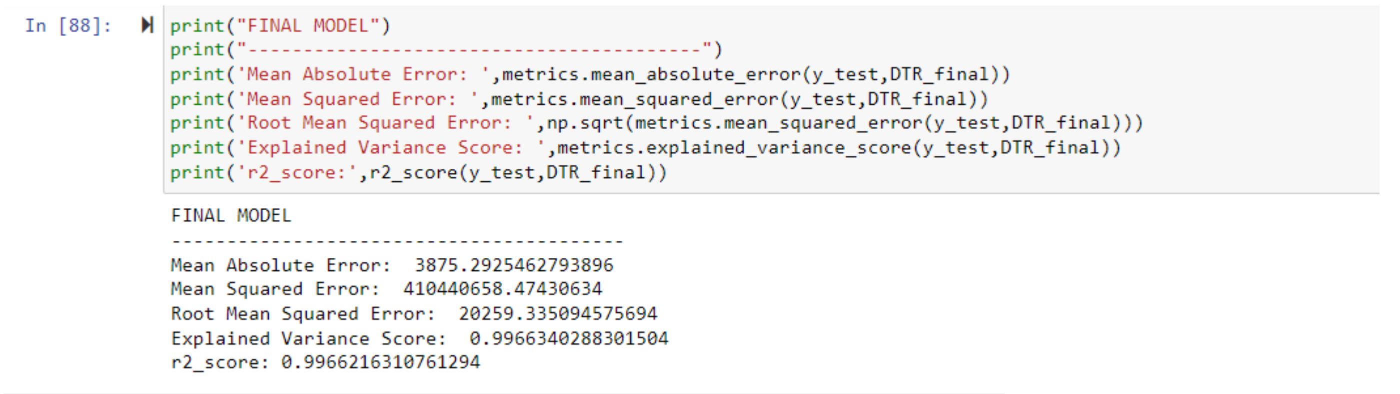
```
In [80]:  print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,GBR_final))
          print('Mean Squared Error: ',metrics.mean_squared_error(y_test,GBR_final))
          print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,GBR_final)))
          print('Explained Variance Score: ',metrics.explained_variance_score(y_test,GBR_final))
          print('r2_score:',r2_score(y_test,GBR_final))

          Mean Absolute Error:  9738.317931805668
          Mean Squared Error:  1492216288.463552
          Root Mean Squared Error:  38629.21547823036
          Explained Variance Score:  0.9877186354083275
          r2_score: 0.987717451883597
```

2.DecisionTreeRegressor

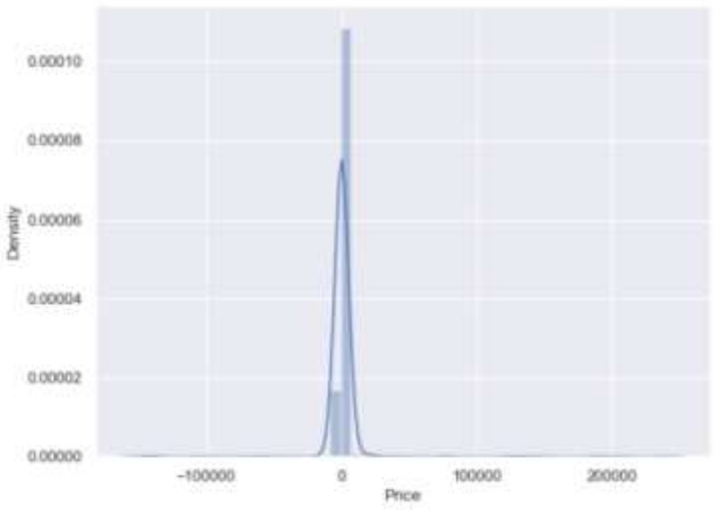


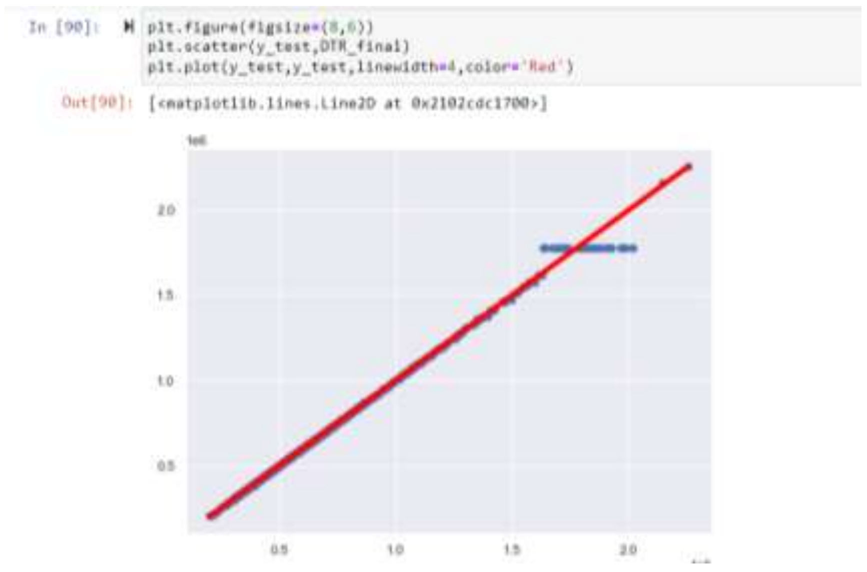
```
In [81]:  from sklearn.model_selection import GridSearchCV
          parameter = {"max_depth":[1,3,5,7,9,11,12],
                       'criterion':['mse','friedman_mse']}
          GCV = GridSearchCV(DecisionTreeRegressor(),parameter,cv=5)

In [ ]:   GCV.fit(x_train,y_train)

In [83]:  GCV.best_params_

Out[83]:  {'criterion': 'friedman_mse', 'max_depth': 11}

In [84]:  from sklearn.tree import DecisionTreeRegressor

          DTR=DecisionTreeRegressor(criterion='mse',max_depth=9)
          DTR.fit(x_train,y_train)
          DTR_final=DTR.predict(x_test)

          C:\Users\Hi\anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning: Criterion 'mse' was deprecated in v1.0
          and will be removed in version 1.2. Use `criterion='squared_error'` which is equivalent.
            warnings.warn(

In [85]:  from sklearn.model_selection import GridSearchCV
          parameter = {"max_depth":[1,3,5,7,9,11,12],
                       'criterion':['mse','friedman_mse']}
          GCV = GridSearchCV(DecisionTreeRegressor(),parameter,cv=5)
```

The best model after hyper parameter tuning is DecisionTreeRegressor



```
In [88]:  print("FINAL MODEL")
          print("-----------------------------------------")
          print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,DTR_final))
          print('Mean Squared Error: ',metrics.mean_squared_error(y_test,DTR_final))
          print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,DTR_final)))
          print('Explained Variance Score: ',metrics.explained_variance_score(y_test,DTR_final))
          print('r2_score:',r2_score(y_test,DTR_final))

          FINAL MODEL
          -----------------------------------------
          Mean Absolute Error:  3875.2925462793896
          Mean Squared Error:  410440658.47430634
          Root Mean Squared Error:  20259.335094575694
          Explained Variance Score:  0.9966340288301504
          r2_score: 0.9966216310761294
```



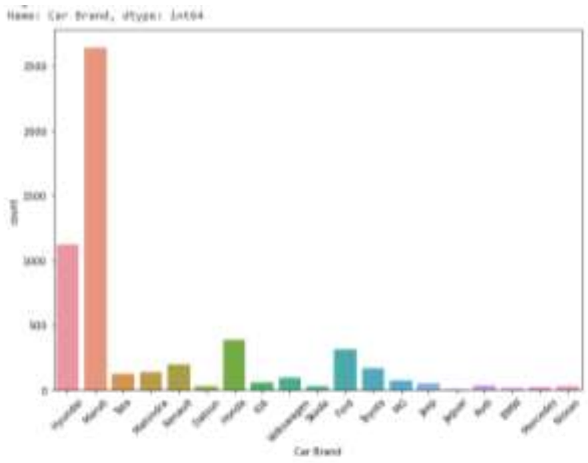


```
In [90]:  plt.figure(figsize=(8,6))
          plt.scatter(y_test,DTR_final)
          plt.plot(y_test,y_test,linewidth=4,color='Red')

Out[90]: [<matplotlib.lines.Line2D at 0x2102cdc1700>]
```

- Visualizations

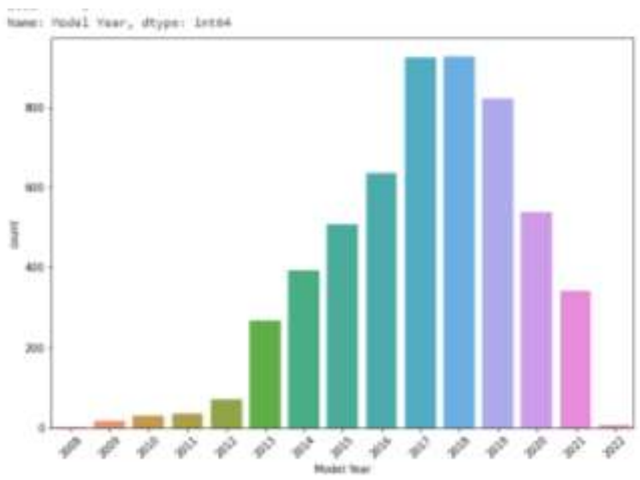
After cleaning the data, we can visualize data and better understand the relationships between different variables. There are many more visualizations that you can do to learn more about your dataset, like scatterplots, histograms, boxplots.

➢ Maruti and Hyundai are the highest demanding car brand. BMW and Jagu ar are the lowest demanding car brand.



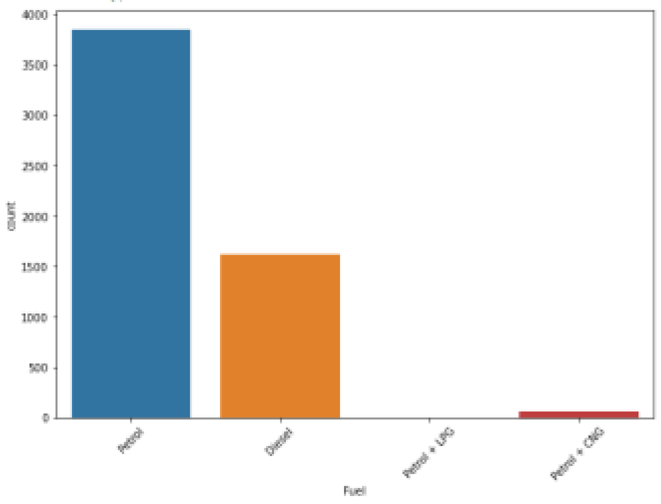
```
Name: Car Brand, dtype: int64
```

➢ 2017,2018 and 2019 model year car have most wanted by customers.

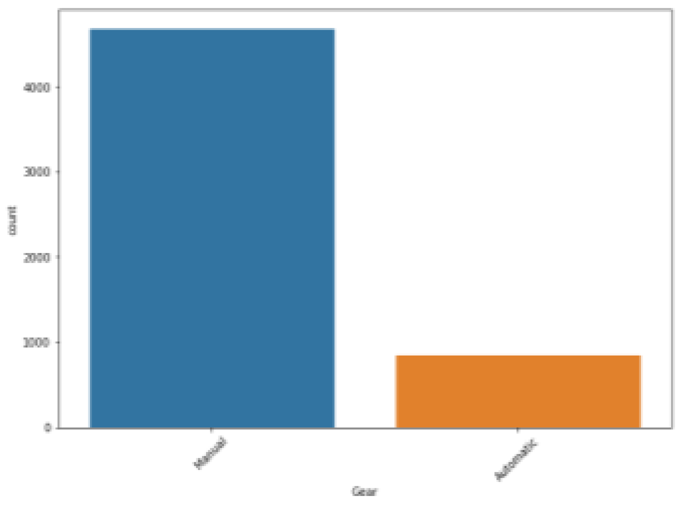


```
Name: Model Year, dtype: int64
```

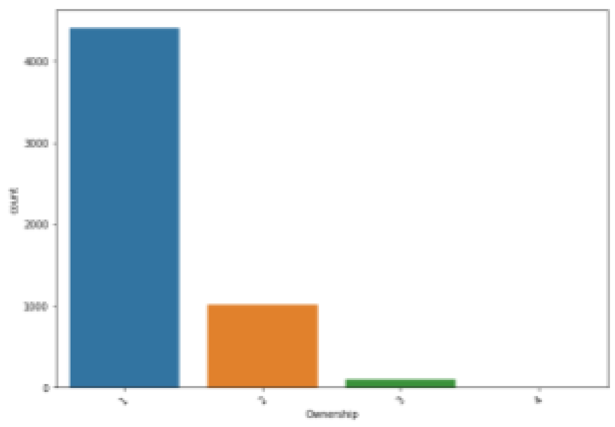
➢ Petrol and Diesel cars are high demand for customers.



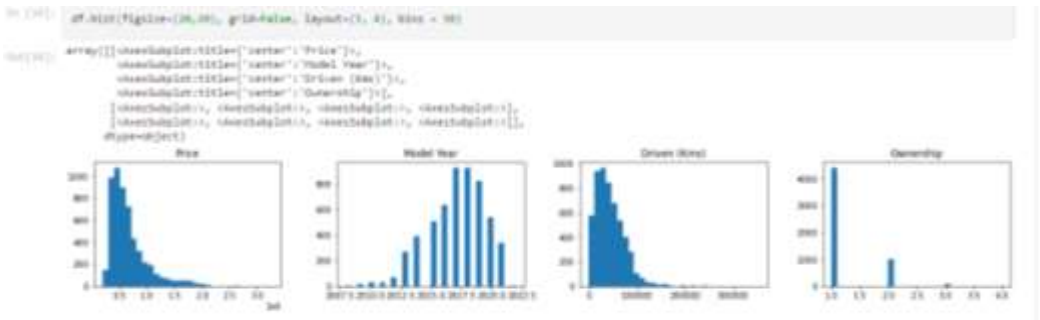
➢ Manual gear type cars are the most demanding by customers.



- Plot show ownership of the car.



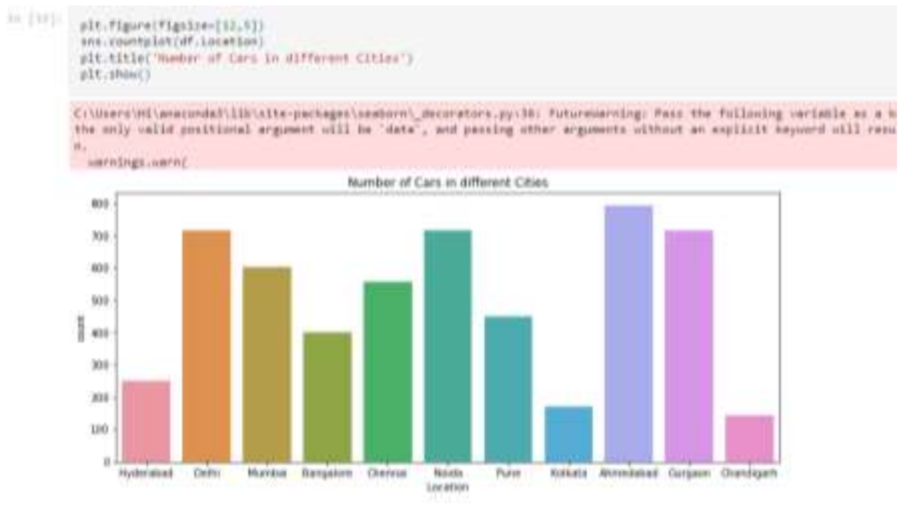
- Price and driven(Kms) data are highly skew.



```
df.hist(figsize=(20,20), grid=False, layout=(1, 4), bins = 30)
```

```
array([[<AxesSubplot:title={'center':'Price'}>,
        <AxesSubplot:title={'center':'Model Year'}>,
        <AxesSubplot:title={'center':'Driven (Kms)'}>,
        <AxesSubplot:title={'center':'Ownership'}>],
      [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
      [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]],
     dtype=object)
```

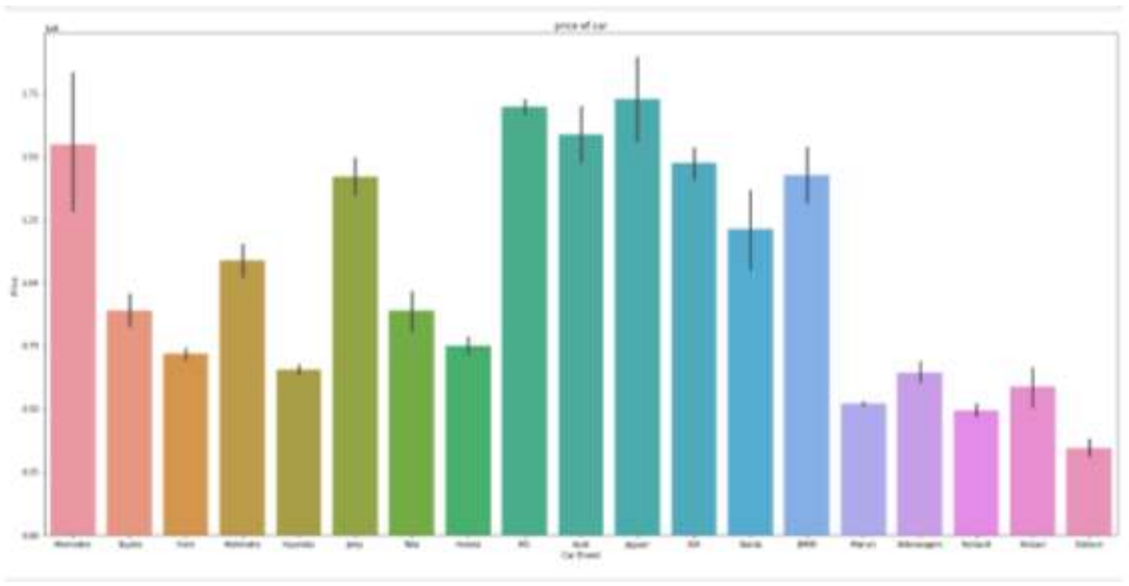
- Ahmedabad ,Noida ,Delhi ,Gurgaon and Mumbai have large number of deman
  d for car.



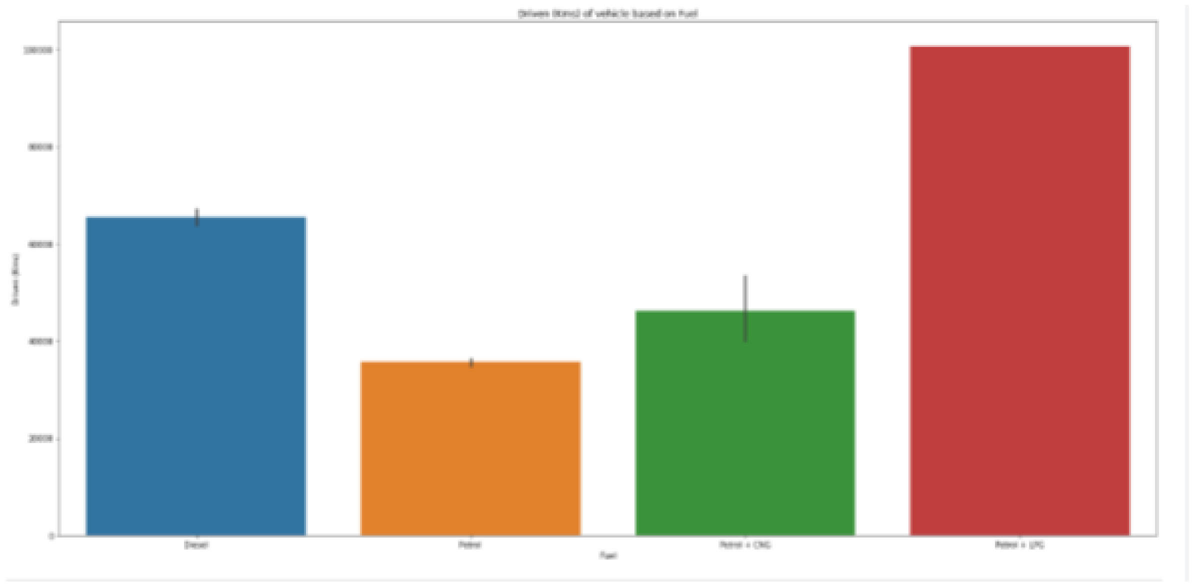
```
plt.figure(figsize=[12,5])
sns.countplot(df.Location)
plt.title('Number of Cars in different Cities')
plt.show()
```

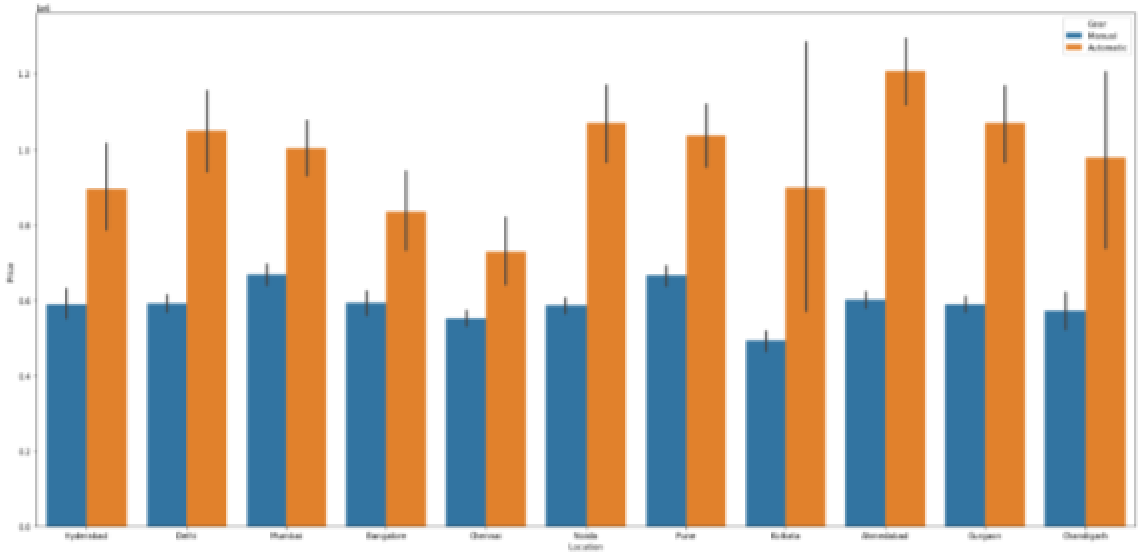
```
C:\Users\HI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a k
the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will resu
n.
  warnings.warn(
```

- MG, Jaguar, Audi and Mercedes car brand have highest price compare with others cars.
- Datsun,Renault and Maruti are the lowest price among other car brand.

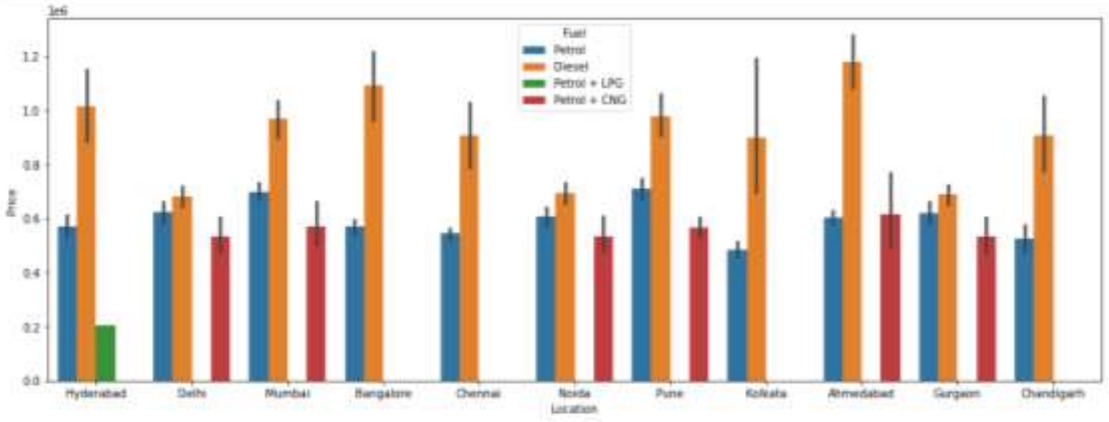


- Petrol+LPG has the Average Car Last.

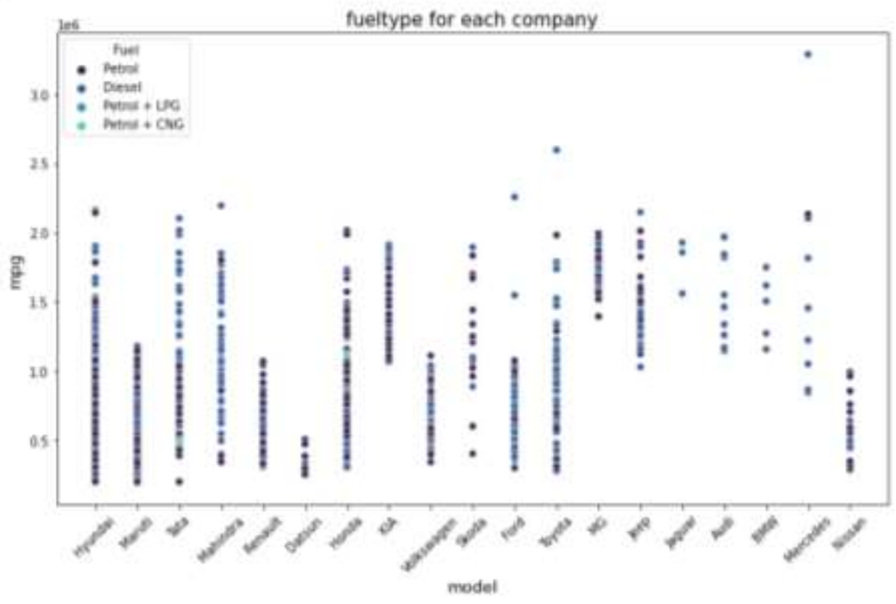




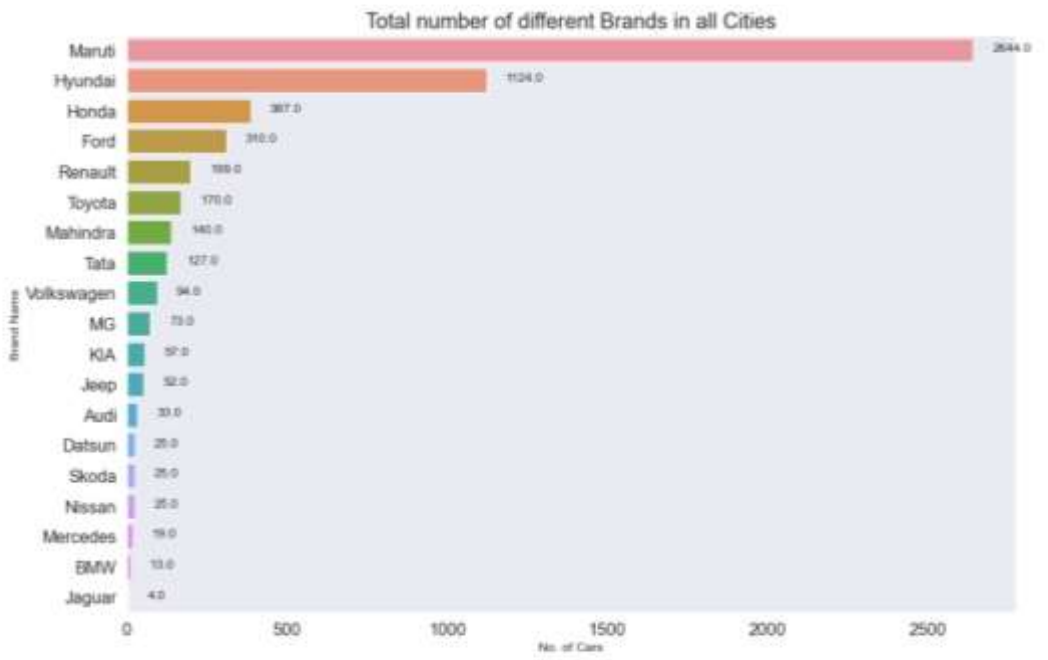
In [24]:
```
plt.figure(figsize=[17,6])
ax = sns.barplot(x="Location", y='Price', hue='Fuel', data=df)
```



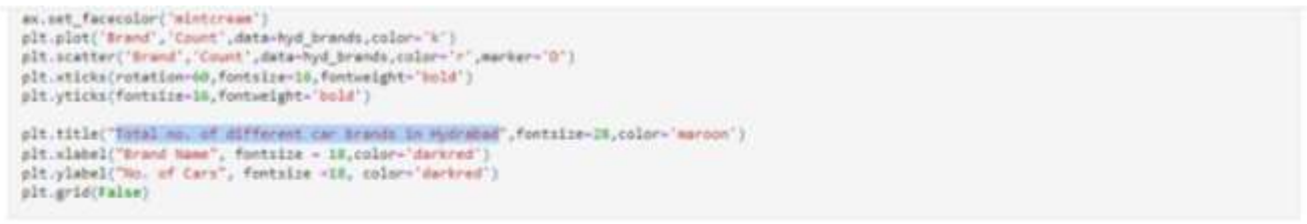
In [28]:
```
plt.figure(figsize=(12,7))
sns.scatterplot(x="Car Brand",y="Price",data=df[0:10000],hue="Fuel",palette="mako")
plt.title("fueltype for each company",size=15)
plt.xlabel("model",size=13)
plt.ylabel("mpg" , size=13)
plt.xticks(rotation=45)
plt.show()
```



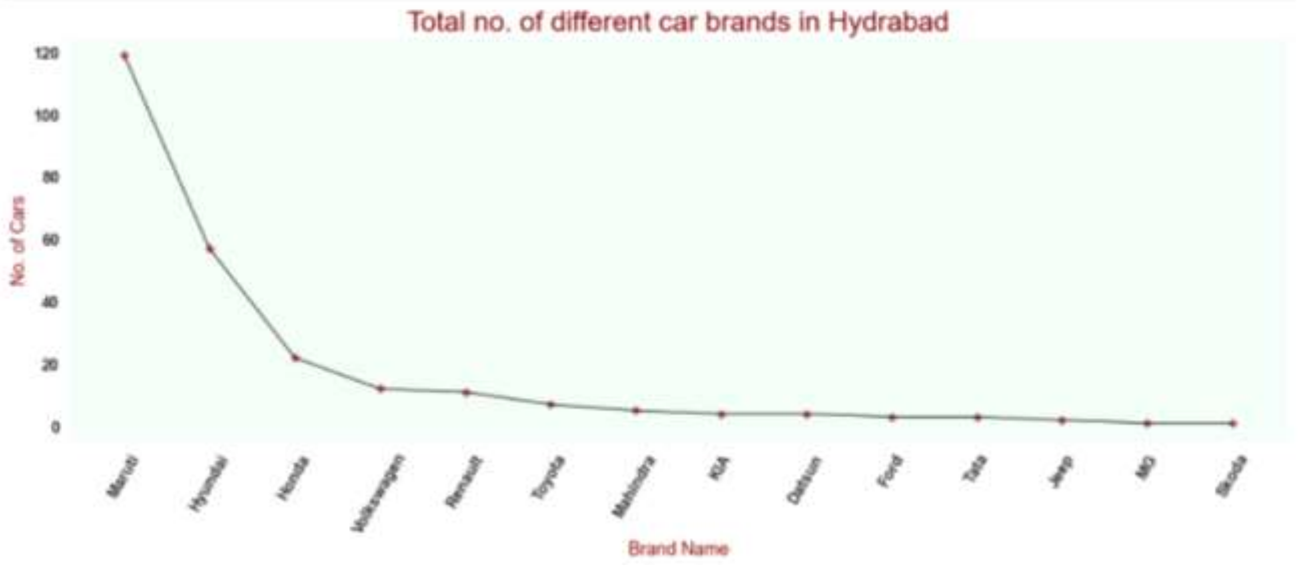
- Maruti has high demand car in all cities. BMW AND Jaguar have few cars in cities.



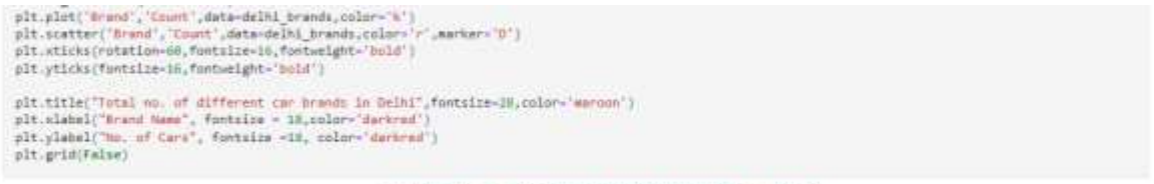
- Total no. of different car brands in Hyderabad.



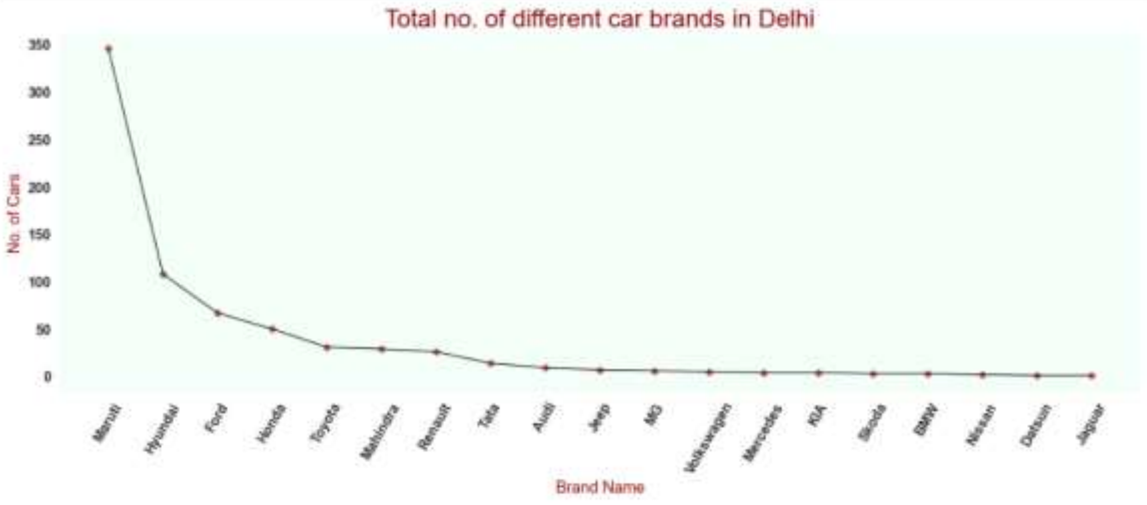
```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=hyd_brands,color='k')
plt.scatter('Brand','Count',data=hyd_brands,color='r',marker='o')
plt.xticks(rotation=60,fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold')

plt.title("Total no. of different car brands in Hydrabad",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```



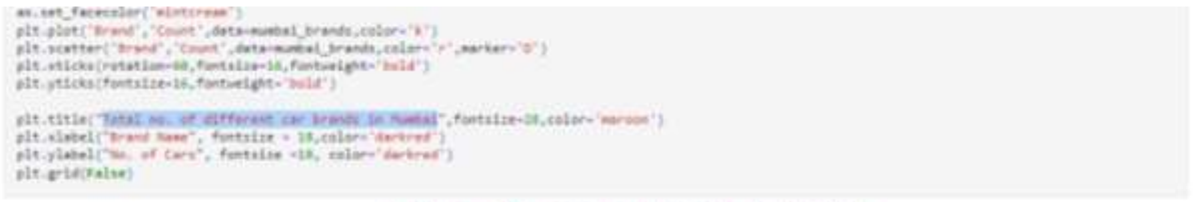
➢ Total no. of different car brands in Delhi



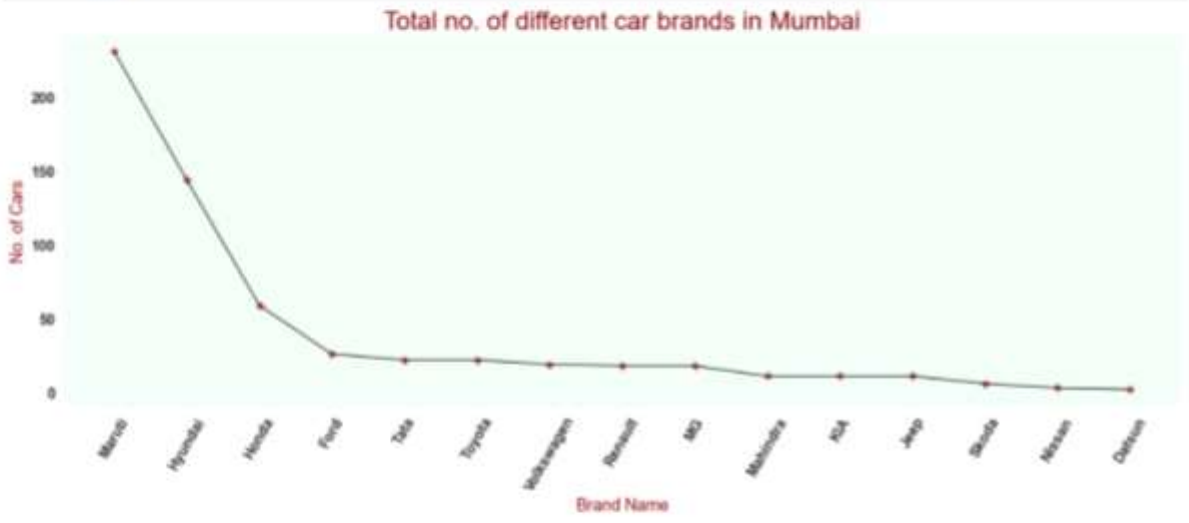
```
plt.plot('Brand','Count',data=delhi_brands,color='k')
plt.scatter('Brand','Count',data=delhi_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Delhi",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```



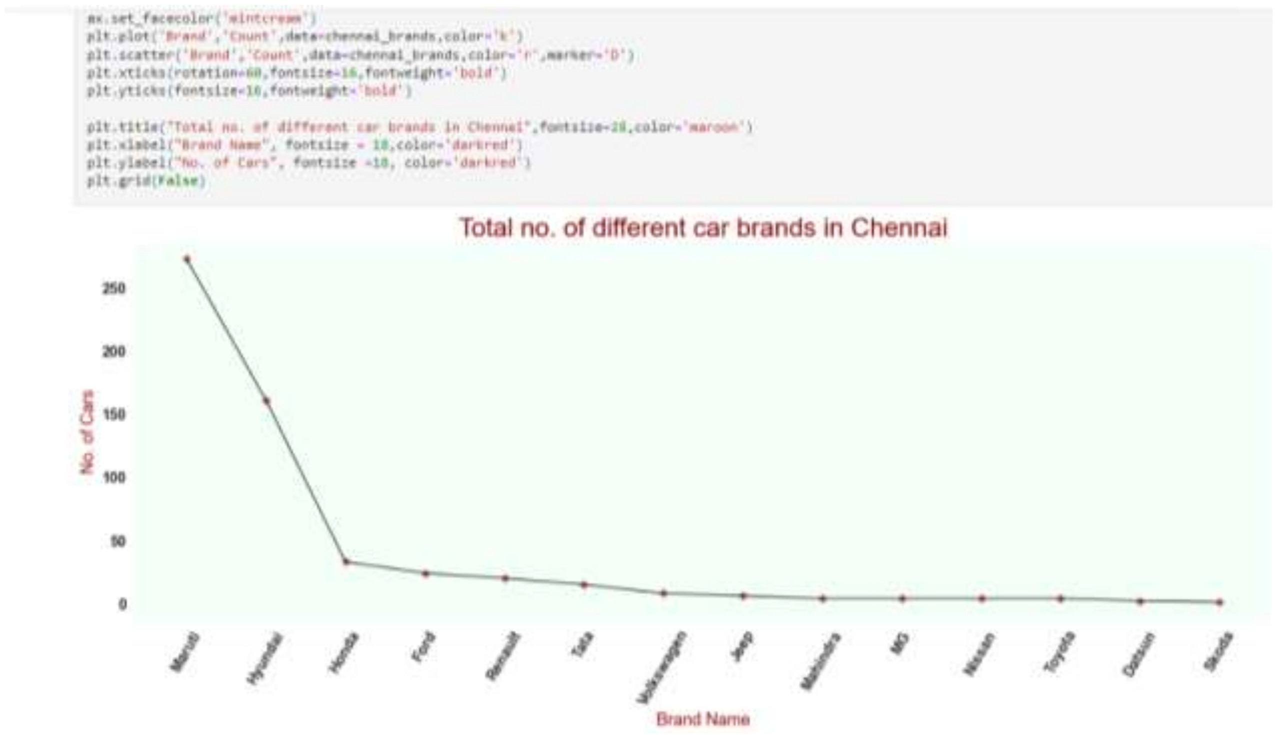
➢ Total no. of different car brands in Mumbai



```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=mumbai_brands,color='k')
plt.scatter('Brand','Count',data=mumbai_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Mumbai",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```

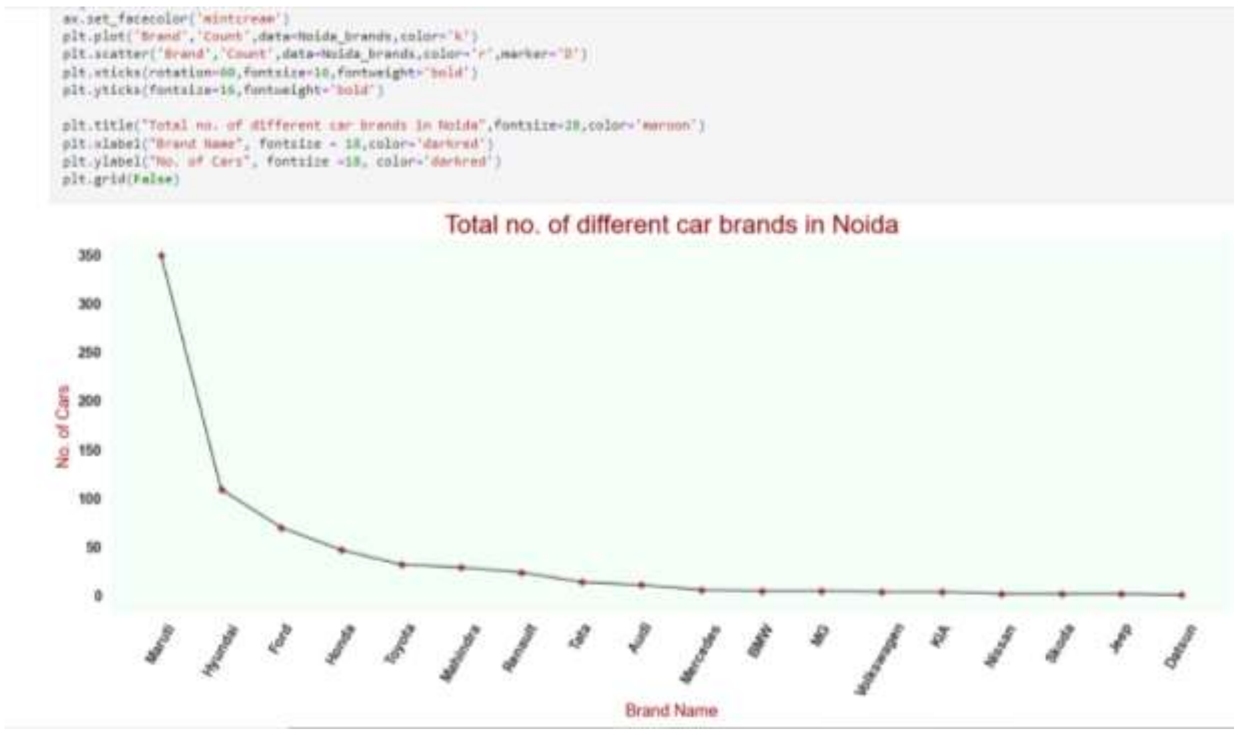


➢ Total no. of different car brands in Chennai



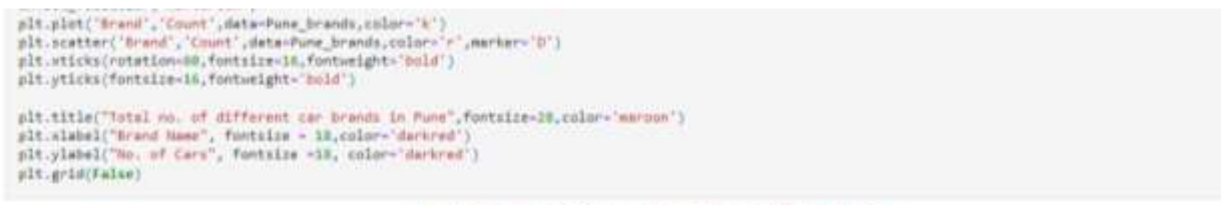
```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=chennai_brands,color='k')
plt.scatter('Brand','Count',data=chennai_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Chennai",fontsize=28,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```

➢ Total no. of different car brands in Noida

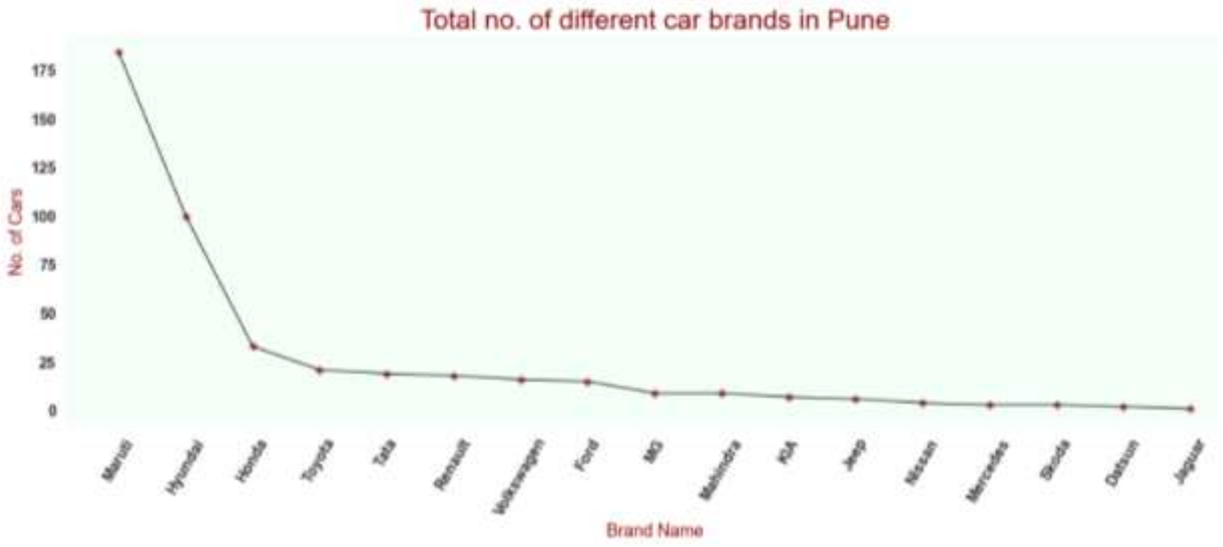


```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=Noida_brands,color='k')
plt.scatter('Brand','Count',data=Noida_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Noida",fontsize=28,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```

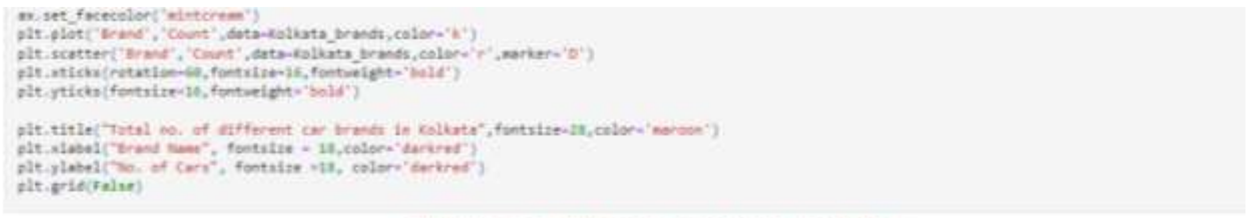
➢ Total no. of different car brands in Pune



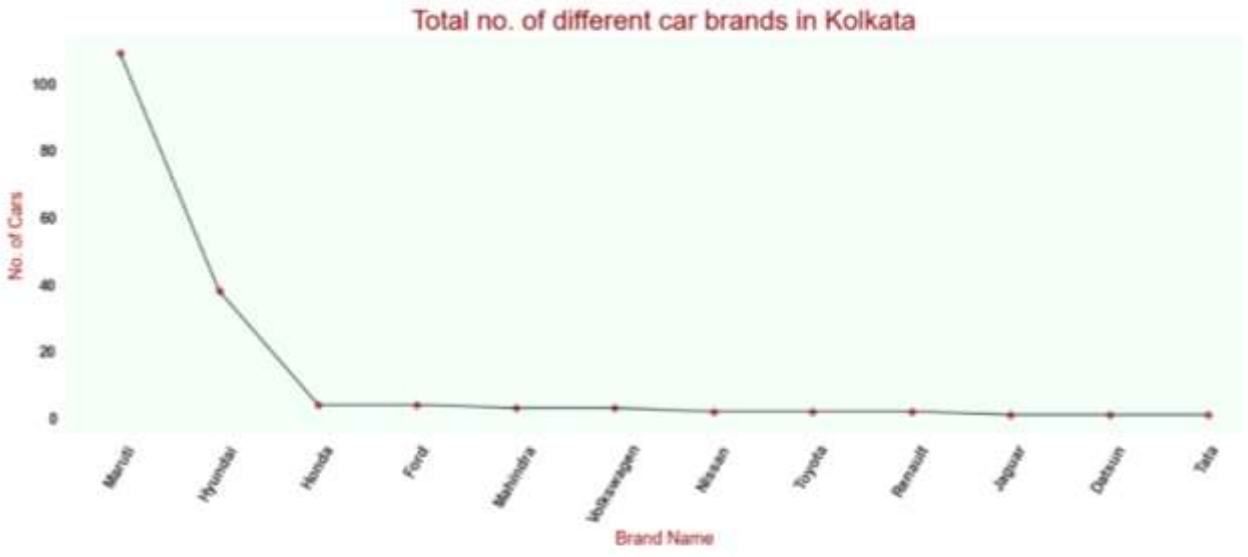
```
plt.plot('Brand','Count',data=Pune_brands,color='k')
plt.scatter('Brand','Count',data=Pune_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Pune",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```



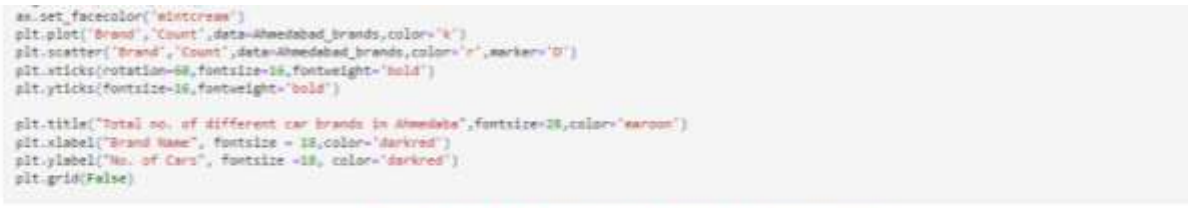
➢ Total no. of different car brands in Kolkata



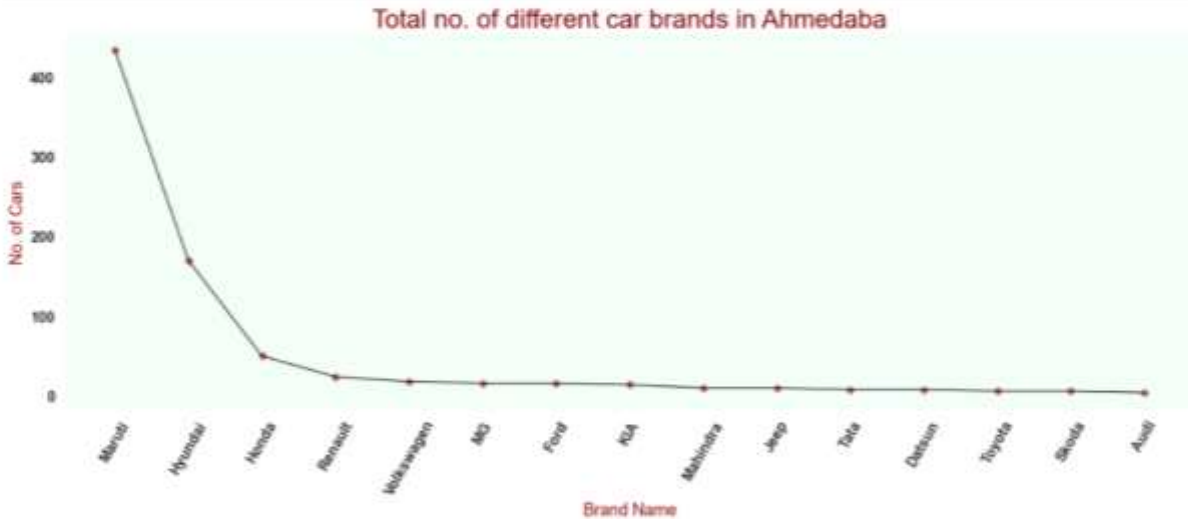
```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=Kolkata_brands,color='k')
plt.scatter('Brand','Count',data=Kolkata_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Kolkata",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```



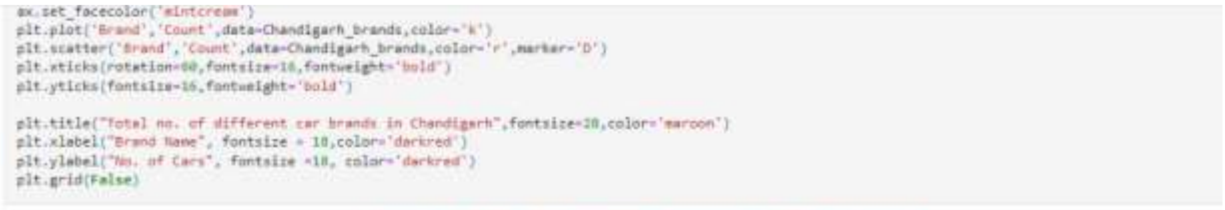
- Total no. of different car brands in Ahmedabad



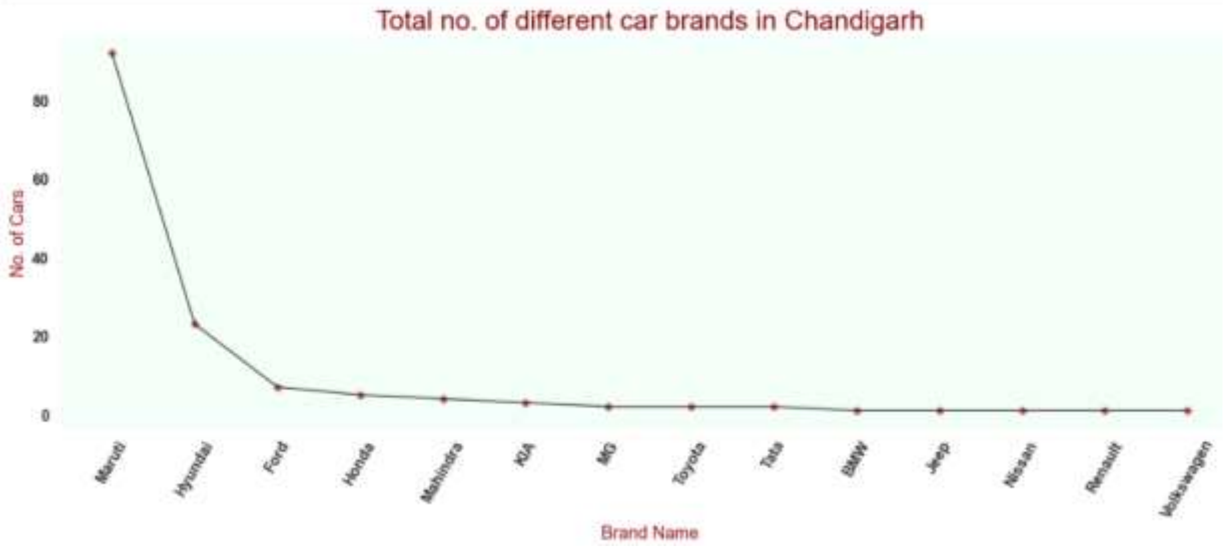
```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=Ahmedabad_brands,color='k')
plt.scatter('Brand','Count',data=Ahmedabad_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Ahmedaba",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```



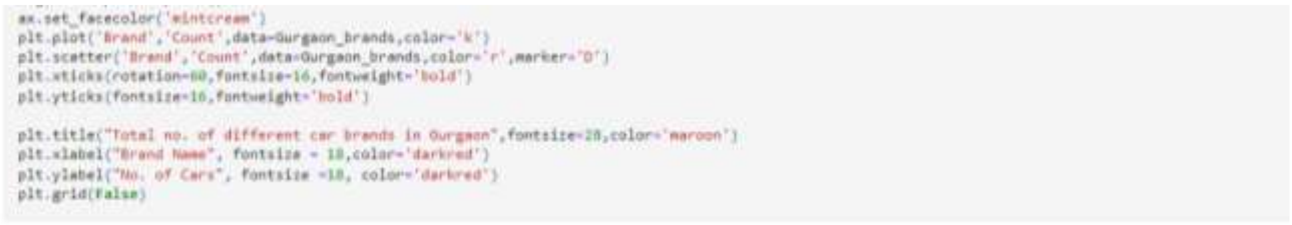
- Total no. of different car brands in Chandigarh



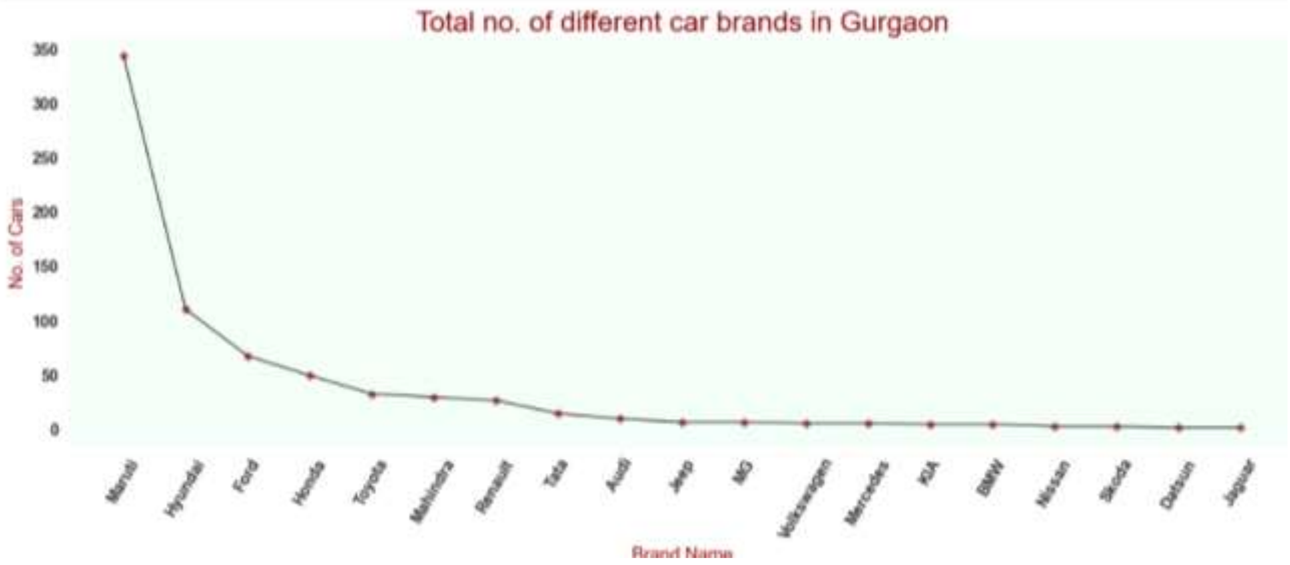
```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=Chandigarh_brands,color='k')
plt.scatter('Brand','Count',data=Chandigarh_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Chandigarh",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```



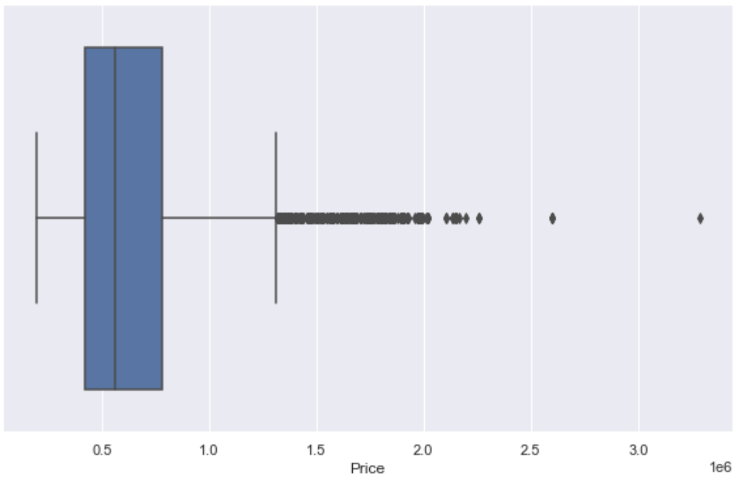
- Total no. of different car brands in Gurgaon

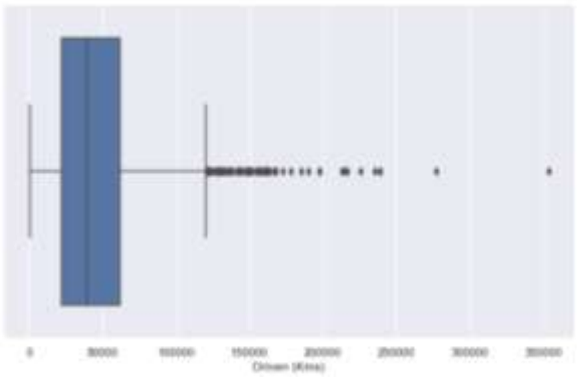


```
ax.set_facecolor('mintcream')
plt.plot('Brand','Count',data=Gurgaon_brands,color='k')
plt.scatter('Brand','Count',data=Gurgaon_brands,color='r',marker='D')
plt.xticks(rotation=60,fontsize=16,fontweight='bold')
plt.yticks(fontsize=16,fontweight='bold')

plt.title("Total no. of different car brands in Gurgaon",fontsize=20,color='maroon')
plt.xlabel("Brand Name", fontsize = 18,color='darkred')
plt.ylabel("No. of Cars", fontsize =18, color='darkred')
plt.grid(False)
```

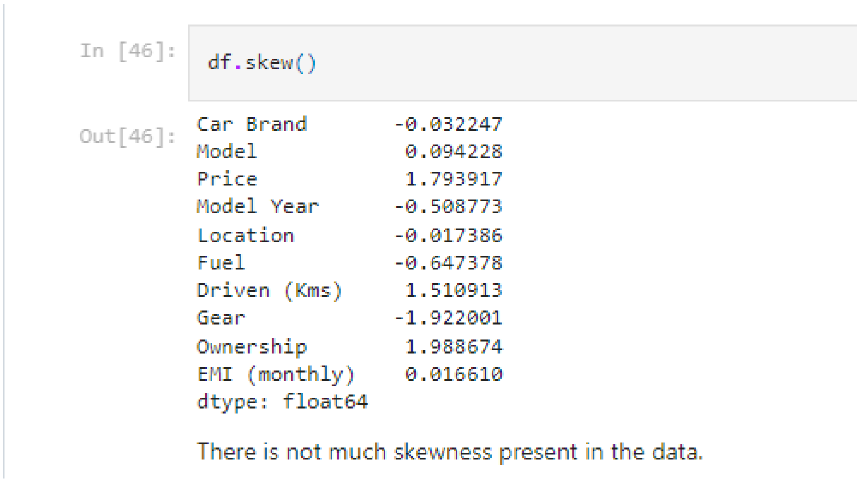


- Checking the outliers





- Price and Driven have high outliers
- Skewness removal



```
In [46]: df.skew()

Out[46]: Car Brand       -0.032247
         Model            0.094228
         Price            1.793917
         Model Year      -0.508773
         Location        -0.017386
         Fuel            -0.647378
         Driven (Kms)     1.510913
         Gear            -1.922001
         Ownership        1.988674
         EMI (monthly)    0.016610
         dtype: float64
```

There is not much skewness present in the data.

Observations

- BalenoDELTA 1.2 K12 car model is a most demand among the customers
- MG, Jaguar, Audi and Mercedes car brand have highest price compare with others cars.
- 'Driven (Kms) of vehicle based on Fuel Petrol + LPG has high demand.
- Compare petrol ,diesel cars has more count among all cities.
- Compared to other 4 cities 'Hyderabad'(415+) has less available cars.
- 'Maruti' brand cars are widely available with a count of around 2800 cars in all the cities, followed by Hyundai(1240+), Honda(449), Toyota(280+)
- Most of the cars runs with 'Petrol' with a count of 3663.
- High budget car among all cities:
- Toyota Land CruiserLC200 VX 2 PREMIUM (3495000/-), 2010 Model, available in Mumbai
- Low budget car among all cities:
- Maruti AltoLX (91000/-), 2008 Model, available in Delhi

CONCLUSION

• Key Findings and Conclusions of the Study

From this dataset I get to know that each feature play a very import role to understand the data. Data format plays a very important role in the visualization and Appling the models and algorithms. Importance of removing the skewness and outlier is important. Finding the best parameters for the algorithm also plays a important role in performance and accuracy of the model.

• Learning Outcomes of the Study in respect of Data Science

Learnt how to process the large number of data. Tried and learnt more about distribution of the data. The power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important step to remove missing value or null value fill it by mean median or by mode or by 0.Setting a good parameters is more important for the model accuracy. Finding a best random state played a vital roll in finding a better model.

• Limitations of this work and Scope for Future Work

The techniques to increase the speed of the model need to be constructed. The future model can be constructed with the most co related data with the target variable in order to increase the speed of the model.

In future this machine learning model may bind with various website which can provide real time data for price prediction. Also we may add large historical data of car price which can help to improve accuracy of the machine learning model. We can build an android app as user interface for interacting with user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset.