NAME OF THE PROJECT

Car Price Prediction

Submitted by:

Arshi Maitra

Acknowledgement:

This project has been completed with the help of training documents and live classes recordings from Data Trained Education. Few helps on coding have also been taken from few data science websites like Toward Data Science, Geek for Geeks, Stack Overflow. All data related to the car price has been taken from www.carwale.com

INTRODUCTION

Many used vehicles are sold each year . Effective pricing strategies can help any company to efficiently sell its products in a competitive market and making profit. In the automotive sector, pricing analytics play an essential role for and understanding the factors that determine the price of a car is a huge advantage for the seller. Below is a dataset used from www.carwale.com to find what are the different factors that determine the pricing of a car.

The main goal of the project will be to determine the following:

- *To determine the price of various cars in India*
- *To understand the factors that are responsible for the price of a car*

Analytical Problem Framing

Webscraping:

Before importing the necessary libraries for data analysis and prediction, the data has been first scrapped from the website www.carwale.com by using Selenium method from Webscraping. The data is saved in CSV which is later on used for analysis:

In [88]:
```
1 UsedCar_Price
```

Out[88]:

	Name	Kilometer	Fuel	Location	EMI	Price
0	Skoda Superb L&K TSI AT	31,000 km	Petrol	Pune	EMI starts at₹44,838	₹ 27 Lakh
1	Maruti Suzuki SX4 VXI CNG BS-IV	76,000 km	CNG	Delhi	EMI starts at₹41,134	₹ 2.1 Lakh
2	Ford Endeavour Titanium 2.2 4x2 AT [2016-2018]	33,000 km	Diesel	Pune	EMI starts at₹31,552	₹ 24.77 Lakh
3	Mahindra Thar LX 4-STR Convertible Diesel AT	7,256 km	Diesel	Hyderabad	EMI starts at₹26,570	₹ 19 Lakh
4	Kia Seltos GTX Plus 1.4 [2020-2021]	19,000 km	Petrol	Delhi	EMI starts at₹13,268	₹ 16 Lakh
...	...	...	...	...	...	...
2747	Maruti Suzuki Ignis Alpha 1.2 AMT	14,410 km	Petrol	Mumbai	EMI starts at₹8,718	₹ 6.5 Lakh
2748	Maruti Suzuki S-Presso LXi (O) CNG	13,221 km	CNG	Mumbai	EMI starts at₹4,982	₹ 5.5 Lakh
2749	Hyundai Xcent SX 1.2 (O)	55,046 km	Petrol	Mumbai	EMI starts at₹1,494	₹ 4.95 Lakh
2750	Hyundai Eon Era + AirBag	63,202 km	Petrol	Chennai	EMI starts at₹7,722	₹ 3.1 Lakh
2751	Kia Sonet HTX 1.0 iMT [2020-2021]	16,952 km	Petrol	Mumbai	EMI starts at₹9,548	₹ 11.5 Lakh

2752 rows × 6 columns

In [90]:
```
1 UsedCar_Price.to_csv('Car Pricing Details.csv')
```

In [ ]:
```
1 #Saving it in CSP
```

Importing of necessary libraries:

```
In [59]:
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVC
from prettytable import PrettyTable
import warnings
warnings.filterwarnings('ignore') #Importing the necessary libraries
```

```
In [2]:
Data=pd.read_csv("Car Pricing Details.csv")
```

Data preprocessing/Data Cleaning:

The data has been loaded

```
In [2]:
Data=pd.read_csv("Car Pricing Details.csv")
```

```
In [66]:
Data.head()
```

Out[66]:

	Name	Kilometer	Fuel	Location	EMI	Price
0	163	102	2	14	86	60
1	122	223	0	5	85	41
2	9	113	1	14	68	59
3	86	210	1	6	62	34
4	72	55	2	5	23	28

Going through the columns of the data:

```
In [8]:  Data.columns # Checking the columns
Out[8]: Index(['Name', 'Kilometer', 'Fuel', 'Location', 'EMI', 'Price'], dtype='object')
```

Checking the null values from the data:

```
In [10]:  Data.isnull().sum() # To check null values
Out[10]: Name         0
         Kilometer    0
         Fuel         0
         Location     0
         EMI          0
         Price        0
         dtype: int64
```

Dropping the unnecessary column:

```
In [5]:  Data.drop('Unnamed: 0',axis=1,inplace=True)
```

Encoding the object data types:

```
In [21]:  enc= LabelEncoder() #Encoding the object data type

In [23]:  columns=['Name','Kilometer','Fuel','Location','EMI','Price']
          Data[columns] = Data[columns].apply(enc.fit_transform) #Encoding the object data type into int data type
```

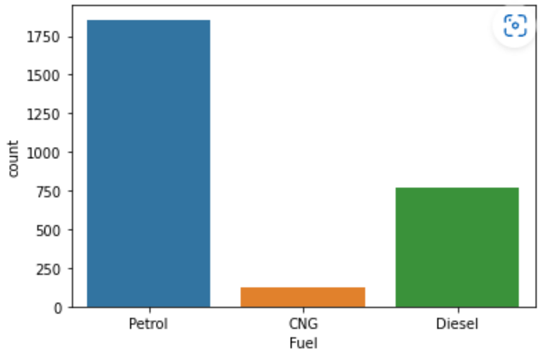
Checking the feature data

1) Name:

```
In [12]:  Data['Name'].value_counts()
Out[12]: MG Hector Sharp 1.5 DCT Petrol [2019-2020]          77
         Honda City V Petrol [2017-2019]                     55
         Ford EcoSport Titanium 1.5L Ti-VCT                  55
         Hyundai Creta SX 1.6 AT Petrol                      44
         Kia Seltos GTX Plus AT 1.4 [2019-2020]              44
                                                             ..
         Toyota Innova Crysta 2.8 ZX AT 7 STR [2016-2020]    11
         MINI Cooper JCW Hatchback                           11
         Toyota Innova Crysta 2.4 ZX 7 STR [2016-2020]       11
         Toyota Innova Crysta 2.4 G 8 STR [2016-2017]        11
         Mahindra Alturas G4 4WD AT [2018-2020]              11
         Name: Name, Length: 188, dtype: int64
```

2) Fuel type:

```
In [15]:  sns.countplot(Data['Fuel'])
Out[15]: <AxesSubplot:xlabel='Fuel', ylabel='count'>
```



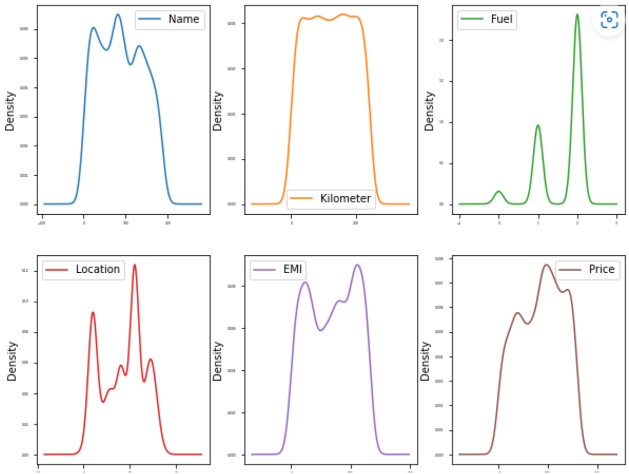
3) Location:

```
In [19]:  Data['Location'].value_counts()
Out[19]: Mumbai        744
         Bangalore     550
         Kanpur        297
         Pune          224
         Salem         198
         Hyderabad     162
         Delhi         113
         Kolkata        78
         Vadodara       77
         Chennai        67
         Nagpur         66
         Patna          44
         Aurangabad     44
         Bhopal         33
         Jamshedpur     22
         Lucknow        11
         Ahmedabad      11
         Vijaywada      11
         Name: Location, dtype: int64
```

***** Basic details identified are Maximum cars are sold from Mumbai with the brand name MC Hector sharp and majority cars has its fuel as petrol***

Checking Data Skewness :

In [29]:
```
Data.plot(kind='kde',subplots=True,layout=(2,3),sharex=False,legend=True,fontsize=3,figsize=(10,8))
plt.show() # ploting the data and observing high skewness
```



Skewness Score:

In [31]:
```
Data.skew().sort_values(ascending=False) #checking the skewness
```

Out[31]:
```
Name         0.106926
Kilometer   -0.003635
EMI         -0.079938
Price       -0.170924
Location    -0.189716
Fuel        -1.254091
dtype: float64
```

Importing Power Transformation.

As the data is skewed it will affect the prediction score. Therefore data has to be transformed before taking any further action:

In [32]:
```
from sklearn.preprocessing import power_transform
```

In [33]:
```
New_Data=power_transform(Data)
```

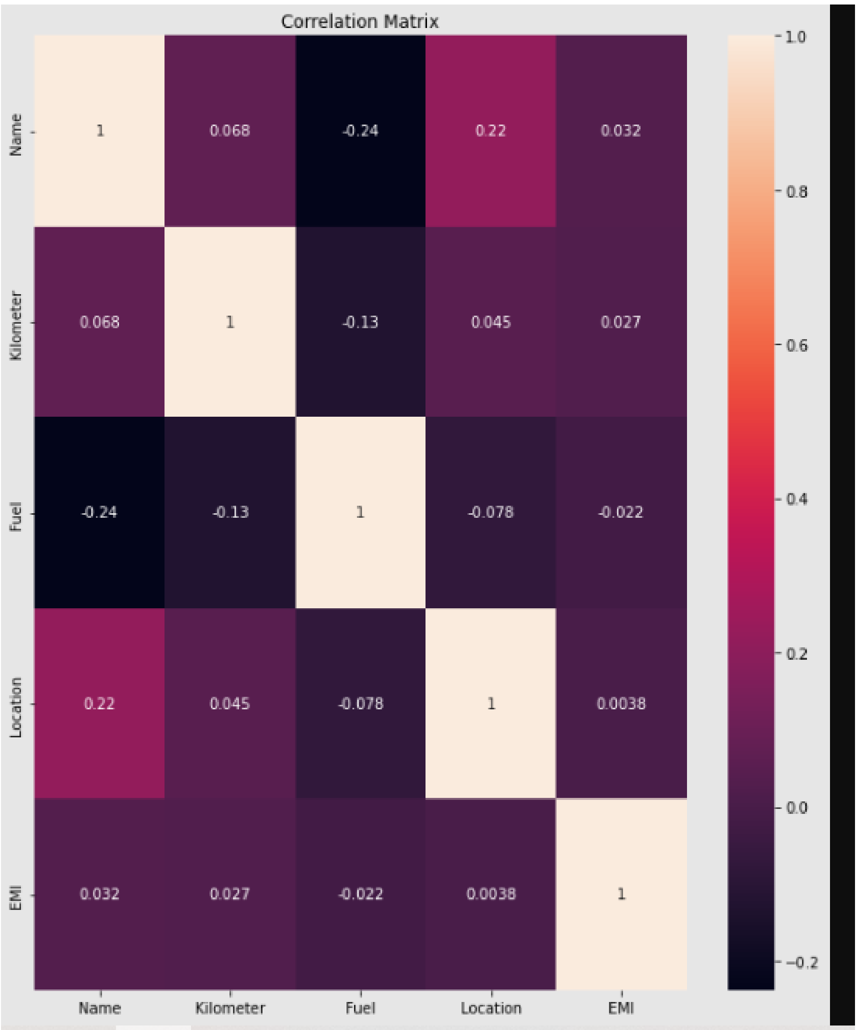
In [34]:
```
pd.DataFrame(New_Data,columns=Data.columns).skew().sort_values(ascending=False) # transforming the data to reduce skewness
```

Out[34]:
```
Name        -0.238199
Location    -0.262248
Kilometer   -0.280738
EMI         -0.305570
Price       -0.314064
Fuel        -0.769798
dtype: float64
```

Checking Multicollinearity:

i) Heat Map:

```
In [37]: corr_mat=X.corr()
         plt.figure(figsize=[20,25])
         sns.heatmap(corr_mat,annot=True)
         plt.title("Correlation Matrix")
         plt.show() #Checking correlation
```



From the heat map it can be observed that there is no issue of multicollinearity in the data..

ii) VIF

In [73]:

```
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

vif_data
```

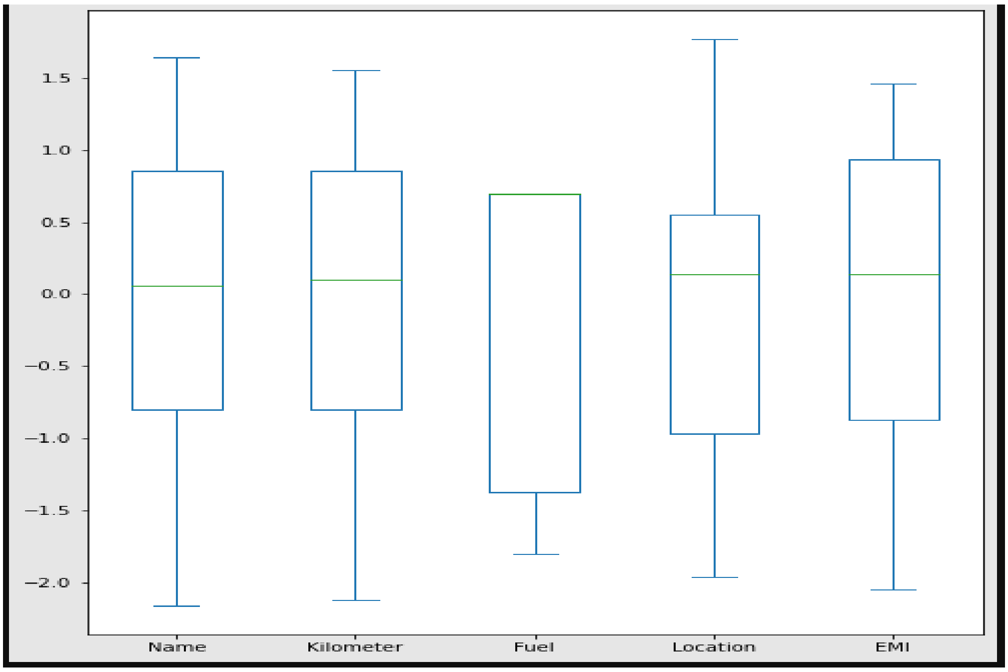
Out[73]:

	feature	VIF
0	Name	1.112584
1	Kilometer	1.016852
2	Fuel	1.069319
3	Location	1.054684
4	EMI	1.001953

There is no issue of multicollinearity in the data.

Detecting Outliers:

Plotting BarPlot:



Z-Score

In [42]:
```
from scipy.stats import zscore
```

In [43]:
```
(np.abs(zscore(X)<3)).all()
```

Out[43]:
```
Name         True
Kilometer    True
Fuel         True
Location     True
EMI          True
dtype: bool
```

In [44]:
```
# There is no Outliers present
```

There is no outliers present in the data

Scaling the data:

In [78]:
```
Scalar=StandardScaler()
```

In [79]:
```
X_Scaled=Scalar.fit_transform(X)
X_Scaled
```

Out[79]:
```
array([[ 1.30335468, -0.1651872 ,  0.69232658,  1.169651  ,  0.50093715],
       [ 0.68237518,  1.34569011, -1.80337293, -0.74085935,  0.47868891],
       [-1.72006883, -0.0121331 , -1.37479261,  1.169651  ,  0.08867915],
       ...,
       [-0.31814124,  0.62496062,  0.69232658,  0.55445741, -2.05401882],
       [-0.7355369 ,  1.01113316,  0.69232658, -0.96945725,  0.89037018],
       [-0.12600824, -1.02088189,  0.69232658,  0.55445741,  1.36239217]])
```

Model/s Development and Evaluation

(Linear Regression, Decision Tree Regressor, Random Forest Regressor and Gradient Boosting Regressor)

Linear Regression: -25%

```
In [80]: LR=LinearRegression()
```

```
In [81]: X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y,test_size=0.20,random_state=50)
         LR.fit(X_train,y_train)
         pred_test=LR.predict(X_test)


         print('R-Squared:',r2_score(y_test,pred_test)*100)

R-Squared:  -0.2585450144481083
```

Decision Tree Regressor: 100%

Decision Tree Regressor

```
In [82]: DT=DecisionTreeRegressor()
```

```
In [83]: X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y,test_size=0.20,random_state=50)
         DT.fit(X_train,y_train)
         pred_test=DT.predict(X_test)


         print('R-Squared:',r2_score(y_test,pred_test)*100)

R-Squared:  100.0
```

Random Forest Regressor::99.98%

```
In [84]: rf=RandomForestRegressor()
```

```
In [85]: X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y,test_size=0.20,random_state=50)
         rf.fit(X_train,y_train)
         pred_test=rf.predict(X_test)


         print('R-Squared:',r2_score(y_test,pred_test)*100)

R-Squared:  99.9827439963225
```

Gradient Boosting Regressor:83.43%

```
In [86]: GB=GradientBoostingRegressor()
```

```
In [87]: X_train,X_test,y_train,y_test=train_test_split(X_Scaled,Y,test_size=0.20,random_state=50)
         GB.fit(X_train,y_train)
         pred_test=GB.predict(X_test)


         print('R-Squared:',r2_score(y_test,pred_test)*100)

R-Squared:  83.43835601351172
```

After getting the r squared scores for all the models, it needs to be checked whether any one of them are overfitted, hence cross validation technique has been used.

As score for Linear Regression is extremely poor hence only taking Decision Tree, Gradient Boosting and Random Forest Regressor for cross validation

Cross Validation for DT:

```
In [88]:
for i in range(2,6):
    DT_Val=cross_val_score(DT,X_Scaled,Y,cv=i)
    print("The cross validation score for Decision Tree Regressor",i,"is",DT_Val.mean())
```

```
The cross validation score for Decision Tree Regressor 2 is 1.0
The cross validation score for Decision Tree Regressor 3 is 1.0
The cross validation score for Decision Tree Regressor 4 is 1.0
The cross validation score for Decision Tree Regressor 5 is 1.0
```

Cross Validation for RF:

```
In [63]:
for i in range(2,6):
    RF_Val=cross_val_score(rf,X_Scaled,Y,cv=i)
    print("The cross validation score for",i,"is",RF_Val.mean()*100)
```

```
The cross validation score for 2 is 99.76139112656959
The cross validation score for 3 is 99.97862258015583
The cross validation score for 4 is 99.98929143559769
The cross validation score for 5 is 99.9925685478573
```

Cross Validation for GB:

```
In [89]:
for i in range(2,6):
    GB_Val=cross_val_score(GB,X_Scaled,Y,cv=i)
    print("The cross validation score for",i,"is",GB_Val.mean()*100)
```

```
The cross validation score for 2 is 82.48519788780038
The cross validation score for 3 is 82.6144004427058
The cross validation score for 4 is 82.73572552686163
The cross validation score for 5 is 83.27377037662131
```

As none of the models are overfitted, and based on the r squared score and cross validation scores, Decision Tree Regressor model is best for this dataset.

As the score is already 100, hence hyper parameter tuning is not performed

Conclusion

Key Findings:

In this data none of the features had any specific correlation with the car pricing. However, our model Decision Tree Regressor has given an excellent performance in such a big dataset and it has performed consistently throughout the Training and Testing process.

Our goal of the project was to create a model that was able to estimate the price of used cars and we achieved it.