FLIP ROBO

Car Price Prediction Project

Submitted By:
SYED MUGERA BILAL
syedbilalarmaan@gmail.com

ACKNOWLEDGMENT

Working on this project ***“Car Price Prediction Project.”*** Was a source of immense knowledge to me. We would like to express my sincere gratitude to FlipRobo Technologies for guidance and valuable support throughout this project work.

We acknowledge with a deep sense of gratitude, the encouragement and inspiration received from our SME/Mentor. We would also like to thank our parents for their love and support.

–SYED MUGERA BILAL

CONTENTS

About the project:

Data Collection Phase

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

Model Building Phase

After collecting the data, you 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.

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

Business Goal:

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.

Data source and their format:

We collected the data from website www.cars24.com.The data is scrapped using Web scraping technique and the framework used is Selenium.

We scrapped nearly 10000 rows of the data and saved as csv file.

Hardware, Software and Tools:

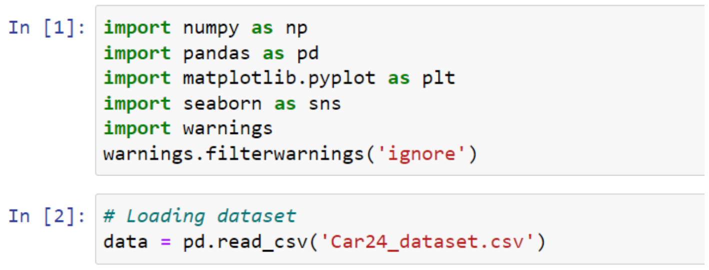
- ❖ For doing this project, we require laptop with high configuration and specification with a stable Internet connection.

- ❖ Microsoft office, Anaconda distribution as software.

- ❖ Python 3.x as programming language.

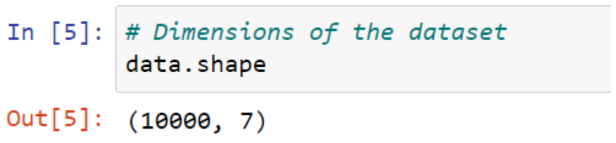
- ❖ Jupyter Notebook as Editor which is in Anaconda navigator.

- ❖ Some tools or libraries required like Numpy – used to numerical calculations. Pandas – used to data manipulation. Matplotlib and Seaborn – used to data visualization.



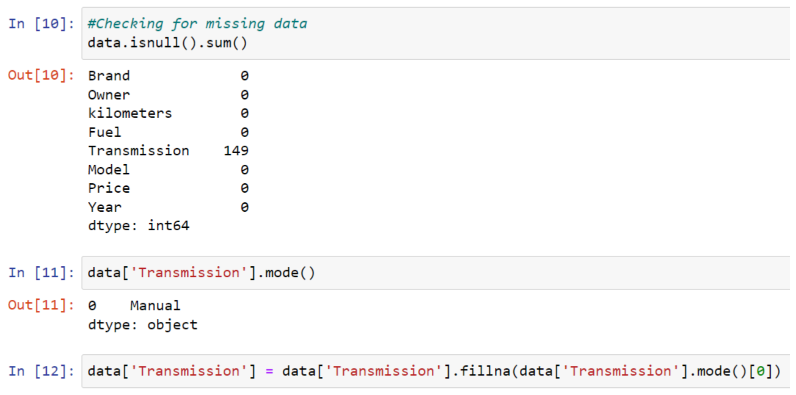
```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')

In [2]: # Loading dataset
        data = pd.read_csv('Car24_dataset.csv')
```

Data Analysis:



```
In [5]: # Dimensions of the dataset
        data.shape

Out[5]: (10000, 7)
```

The dataset has 10000 rows and 7 columns.

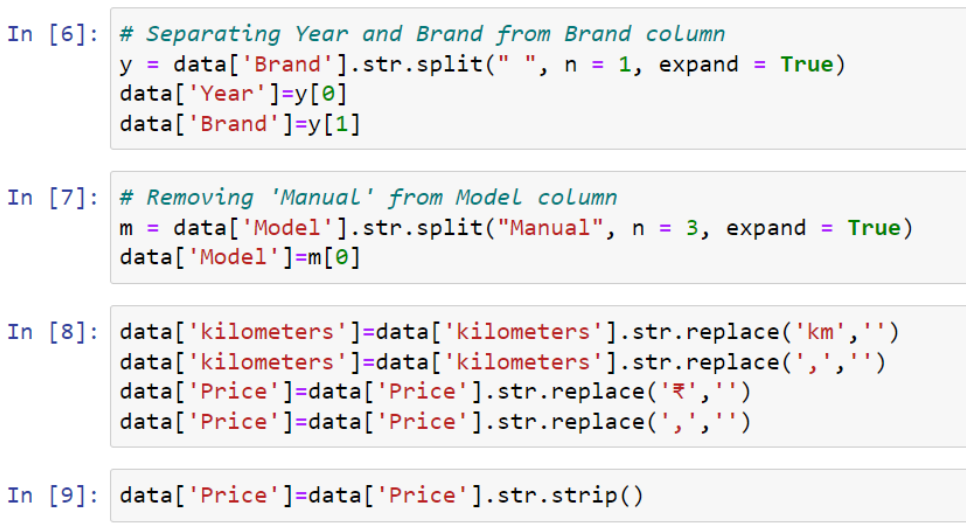


```
In [10]: #Checking for missing data
         data.isnull().sum()

Out[10]: Brand             0
         Owner             0
         kilometers        0
         Fuel              0
         Transmission    149
         Model             0
         Price             0
         Year              0
         dtype: int64

In [11]: data['Transmission'].mode()

Out[11]: 0    Manual
         dtype: object

In [12]: data['Transmission'] = data['Transmission'].fillna(data['Transmission'].mode()[0])
```

We have seen that, in Transmission column there are 149 missing values are available. We have find mode of Transmission column and filled with mode value. (I.e. Manual)

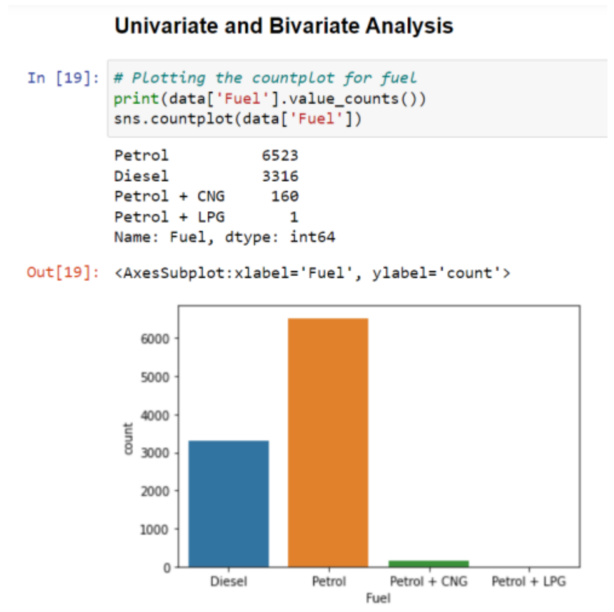
Data Cleaning:

❖ Removing words from object data and converting them into int data type.
❖ Removing ',' , ' ' replacing missing data with highest weightage data.
❖ Removing unnecessary features.



```
In [6]: # Separating Year and Brand from Brand column
        y = data['Brand'].str.split(" ", n = 1, expand = True)
        data['Year']=y[0]
        data['Brand']=y[1]

In [7]: # Removing 'Manual' from Model column
        m = data['Model'].str.split("Manual", n = 3, expand = True)
        data['Model']=m[0]

In [8]: data['kilometers']=data['kilometers'].str.replace('km','')
        data['kilometers']=data['kilometers'].str.replace(',','')
        data['Price']=data['Price'].str.replace('₹','')
        data['Price']=data['Price'].str.replace(',','')

In [9]: data['Price']=data['Price'].str.strip()
```

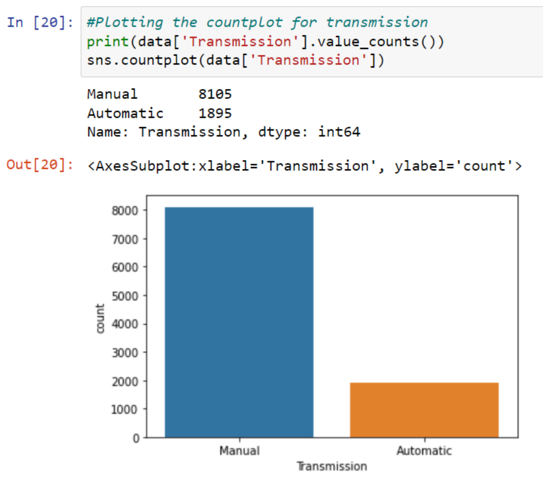
Exploratory Data Analysis (EDA):



Univariate and Bivariate Analysis

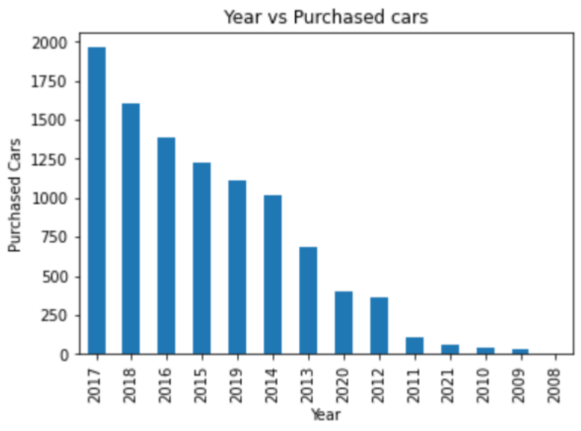
```
In [19]: # Plotting the countplot for fuel
         print(data['Fuel'].value_counts())
         sns.countplot(data['Fuel'])

         Petrol          6523
         Diesel          3316
         Petrol + CNG     160
         Petrol + LPG       1
         Name: Fuel, dtype: int64

Out[19]: <AxesSubplot:xlabel='Fuel', ylabel='count'>
```

Observations:
We can see that Petrol is the maximum fuels used by cars, whereas Diesel, CNG and LPG are the least used.



```
In [20]: #Plotting the countplot for transmission
         print(data['Transmission'].value_counts())
         sns.countplot(data['Transmission'])

         Manual       8105
         Automatic    1895
         Name: Transmission, dtype: int64

Out[20]: <AxesSubplot:xlabel='Transmission', ylabel='count'>
```

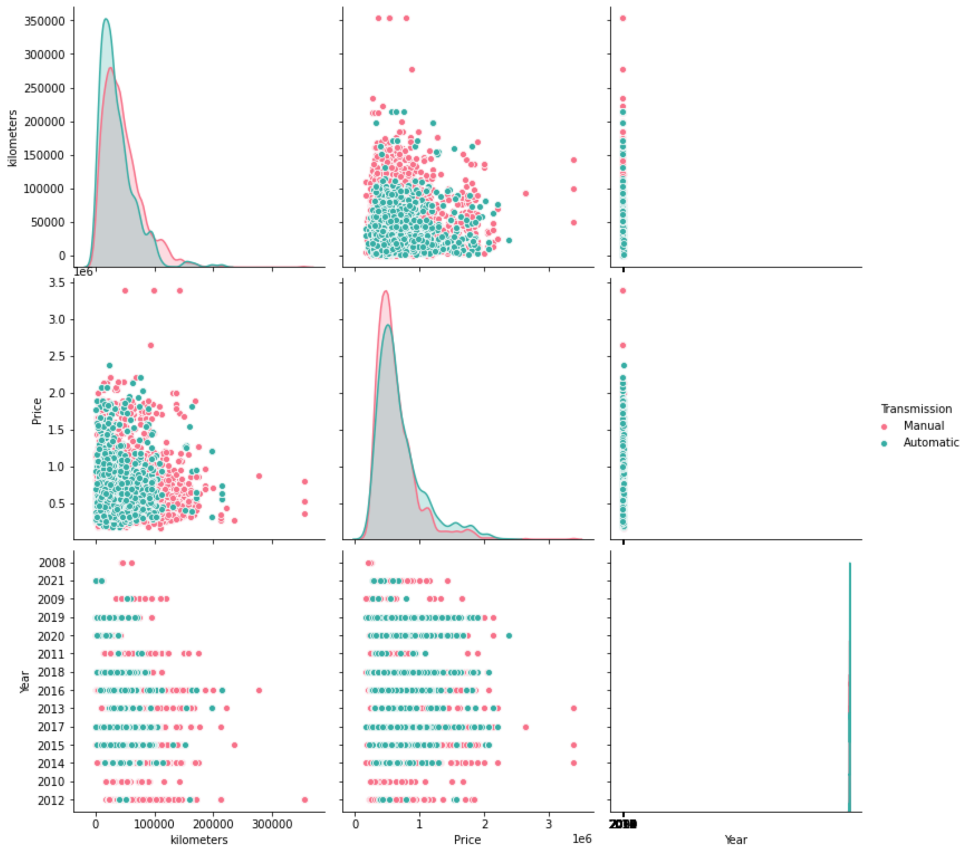
Observations:
Most of the cars have manual transmission as the highest weightage.



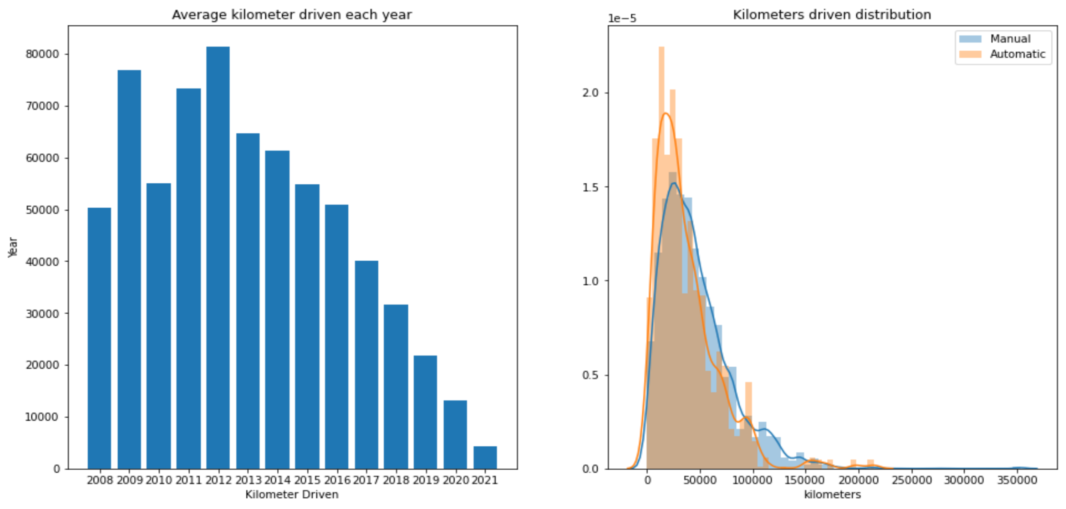
Observations:

Maximum number of cars are bought in the year 2017 whereas minimum number of cars were brought in 2009 and 2008.

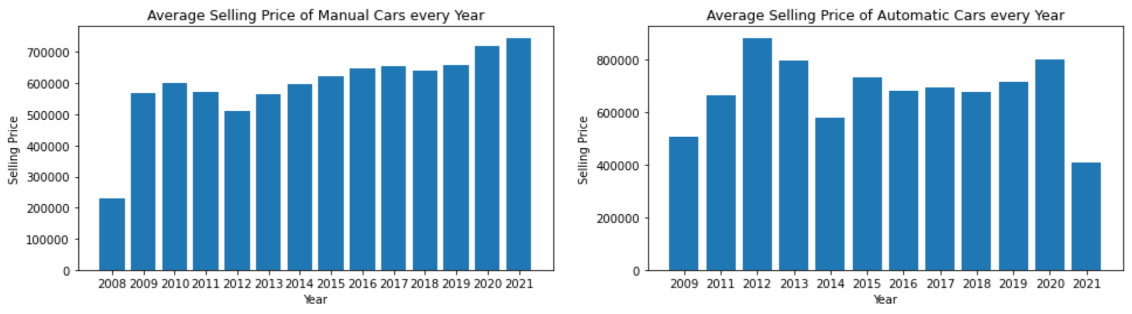
```
In [22]: # Plotting pairplot for transmission
         sns.pairplot(data,hue = 'Transmission',diag_kind = "kde",kind = "scatter",palette = "husl",height=3.5)
         plt.show()
```



```
In [25]: # Plotting for year vs km_driven
km_mean = data.groupby('Year')['kilometers'].mean()

fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(16,8))

ax[0].bar(km_mean.index,km_mean)
sns.distplot(manual['kilometers'],ax=ax[1])
sns.distplot(automatic['kilometers'],ax=ax[1])

ax[0].set_title('Average kilometer driven each year')
ax[0].set_xlabel('Kilometer Driven')
ax[0].set_ylabel('Year')

ax[1].set_title('Kilometers driven distribution')
ax[1].legend(['Manual','Automatic'])

plt.show()
```

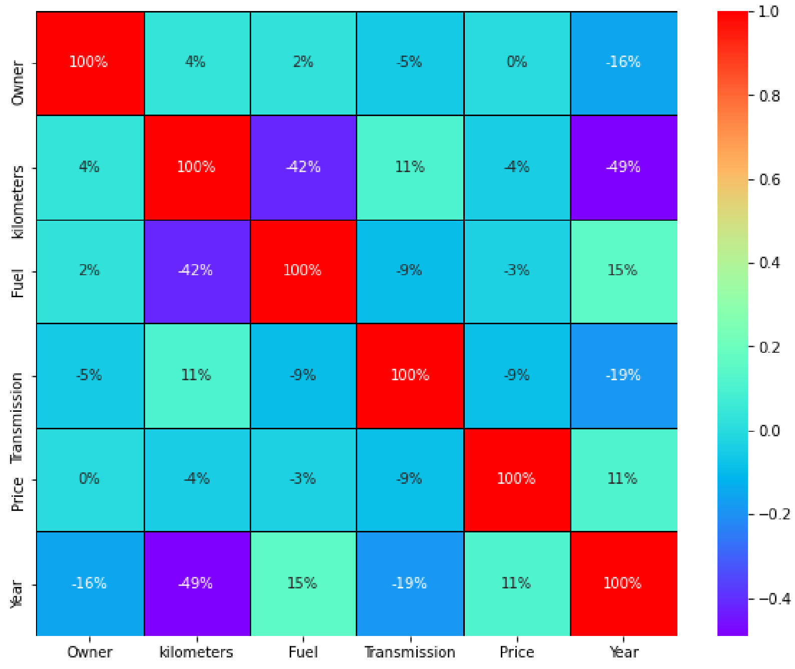


```
In [26]: # Year and selling price
year_mean_manual = data[data['Transmission']=='Manual'].groupby('Year')['Price'].mean()
year_mean_automatic =data[data['Transmission']=='Automatic'].groupby('Year')['Price'].mean()

fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(16,4))

ax[0].bar(year_mean_manual.index,year_mean_manual)
ax[1].bar(year_mean_automatic.index,year_mean_automatic)

ax[0].set_title('Average Selling Price of Manual Cars every Year')
ax[0].set_xlabel('Year')
ax[0].set_ylabel('Selling Price')

ax[1].set_title('Average Selling Price of Automatic Cars every Year')
ax[1].set_xlabel('Year')
ax[1].set_ylabel('Selling Price')

plt.show()
```



Multivariate Analysis

```
In [32]: # Plotting heatmap for visualizing the correlation
         plt.figure(figsize=(10,8))
         sns.heatmap(corr,linewidth=0.5,linecolor='black',fmt='.0%',cmap='rainbow',annot=True)
         plt.show()
```



Observations:

1. There are more negative correlations present in the dataset.

2. Highest positive correlated column has a value of 15%.

3. Highest negatively correlated column has a value of -49%

4. Positively correlated columns have a high impact with the target variable whereas negatively correlated columns have less or zero impact with the target variable.

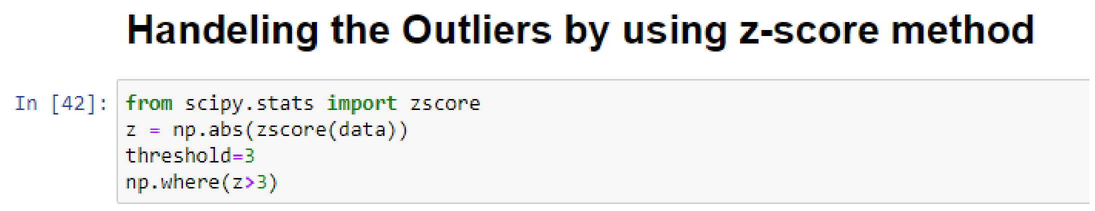
Checking for outliers:

- Outliers are the data points that differ significantly from other observations. Any data points greater than +3 and -3 standard deviations are called as outliers.

- Z-score is the automated method used for handling outliers and it's important to remove outliers as it impacts on the Accuracy of the Model.

- During the outlier's analysis, we found that we are losing nearly 6% of data and it's not a big loss.

- After removing outliers, we have 9405 rows and 6 columns.



Handeling the Outliers by using z-score method

```
In [42]: from scipy.stats import zscore
         z = np.abs(zscore(data))
         threshold=3
         np.where(z>3)
```

Checking for skewness:

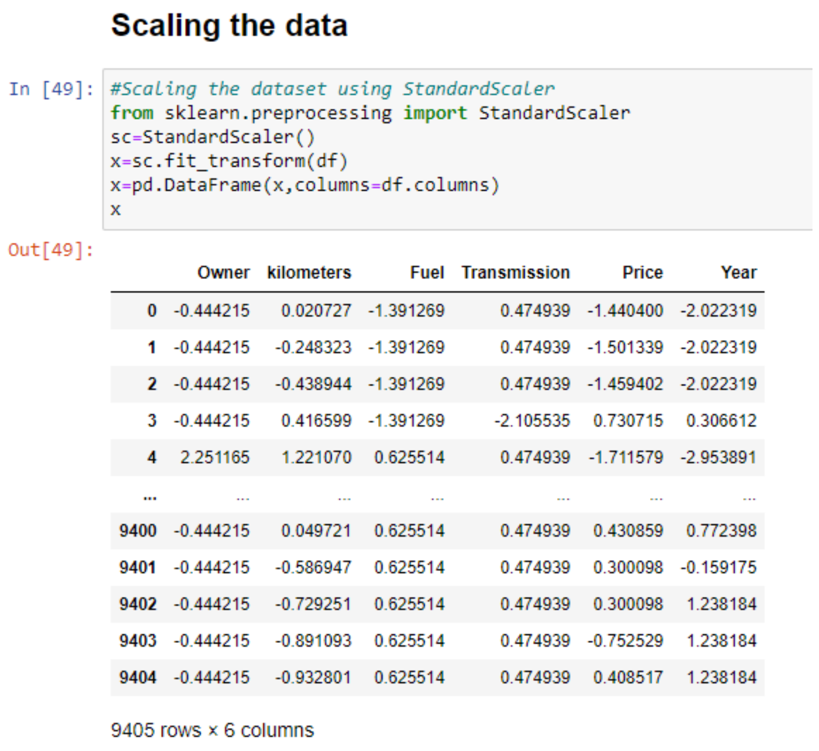
- Skewed data are not normally distributed; either they are positive skewed or negative skewed. If the data is skewed, it impacts on the accuracy of the model. So, it's very important to remove the skewness for right and left skewed data by using transform methods like square root transformation.

- For visualization, we use distplot to check the distribution of data points and the shape of the curve. Any value greater than 0.55 or less than -0.55 is considered to be skewed data.

- In our case, most of the data are skewed and hence we have to remove the skewness during Scaling because if we remove the skewness by square root transformation method it will induce nan values. Sometimes while using root transforms, it can cause to form nan values. It is better to remove those values before scaling because it will show ValueError while running the code.

Standardization / Scaling:

As the values in the dataset have high ranges, it becomes complex for a ML model to understand and read the data, hence data training becomes difficult which is not a proper way to deal with data to achieve good accuracy and get accurate predictions. Therefore, it is very important to normalize/standardize data which means getting data within certain range to have proper understanding of data. In this project we have used StandardScaler() techniques to normalize the data which brings data between the range of 0 to 1.



Scaling the data

```
In [49]: #Scaling the dataset using StandardScaler
         from sklearn.preprocessing import StandardScaler
         sc=StandardScaler()
         x=sc.fit_transform(df)
         x=pd.DataFrame(x,columns=df.columns)
         x
```

Out[49]:

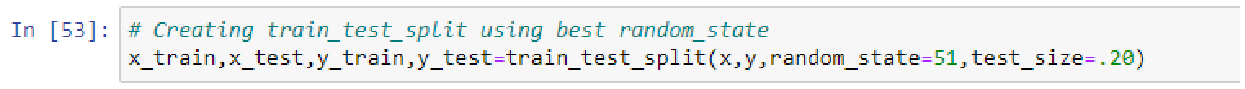
	Owner	kilometers	Fuel	Transmission	Price	Year
0	-0.444215	0.020727	-1.391269	0.474939	-1.440400	-2.022319
1	-0.444215	-0.248323	-1.391269	0.474939	-1.501339	-2.022319
2	-0.444215	-0.438944	-1.391269	0.474939	-1.459402	-2.022319
3	-0.444215	0.416599	-1.391269	-2.105535	0.730715	0.306612
4	2.251165	1.221070	0.625514	0.474939	-1.711579	-2.953891
...	...	...	...	...	...	...
9400	-0.444215	0.049721	0.625514	0.474939	0.430859	0.772398
9401	-0.444215	-0.586947	0.625514	0.474939	0.300098	-0.159175
9402	-0.444215	-0.729251	0.625514	0.474939	0.300098	1.238184
9403	-0.444215	-0.891093	0.625514	0.474939	-0.752529	1.238184
9404	-0.444215	-0.932801	0.625514	0.474939	0.408517	1.238184

9405 rows × 6 columns

<u>**Splitting the data:**</u>

Using scikit learn library of python we have imported train test split to divide data into train data and test data. We have use test size is 0.2 (20%), rest of 0.8 (80%) data as train data. Random state provides us a seed to the random number generator by using following codes.

from sklearn.model_selection import train_train_split
x_train, x_test, y_train, y_test = train_train_split (x, y, test_size=0.2, random_state=51)

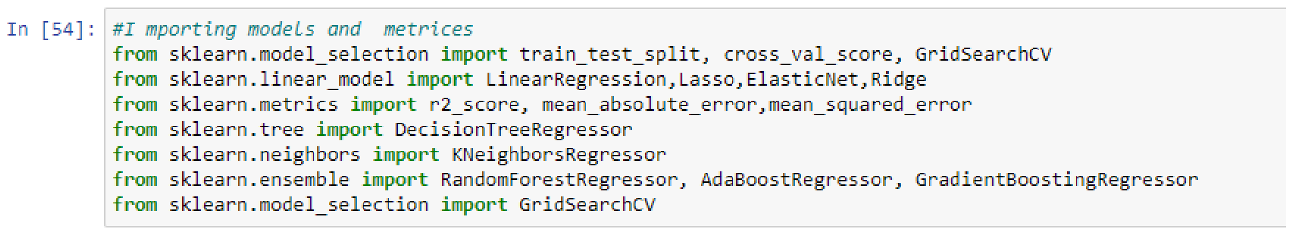


```
In [53]: # Creating train_test_split using best random_state
         x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=51,test_size=.20)
```

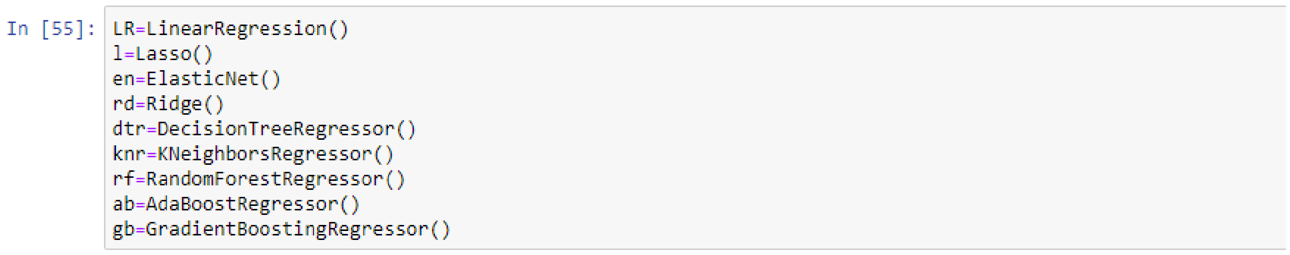
Building Machine Learning Algorithms:

The following classifier algorithms we used are:

- Linear Regression,
- Lasso Regression
- Ridge Regression
- ElasticNet
- Decision Tree Regressor
- KNeighbors Regressor
- Random Forest Regressor,
- AdaBoost Regressor
- Gradient Boosting Regressor



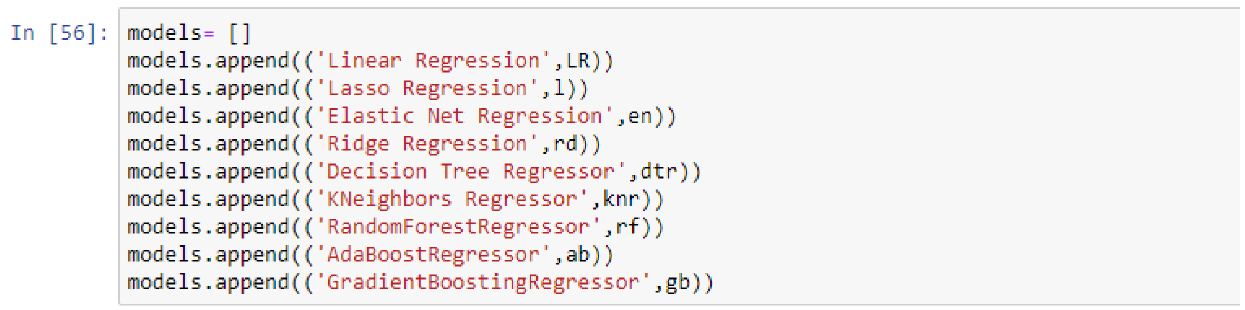
```
In [54]: #I mporting models and  metrices
         from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
         from sklearn.linear_model import LinearRegression,Lasso,ElasticNet,Ridge
         from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.neighbors import KNeighborsRegressor
         from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
         from sklearn.model_selection import GridSearchCV
```



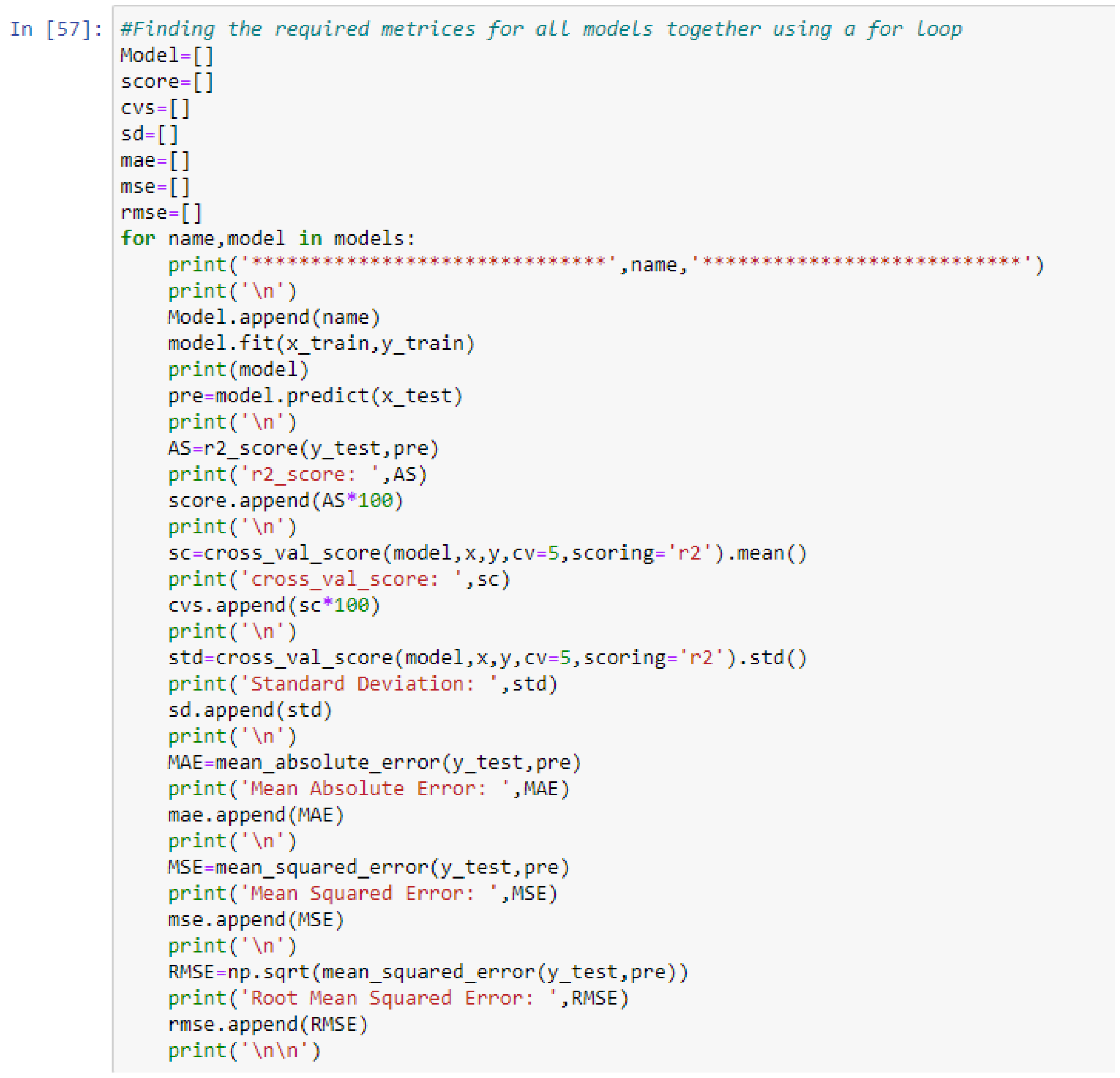
```
In [55]: LR=LinearRegression()
         l=Lasso()
         en=ElasticNet()
         rd=Ridge()
         dtr=DecisionTreeRegressor()
         knr=KNeighborsRegressor()
         rf=RandomForestRegressor()
         ab=AdaBoostRegressor()
         gb=GradientBoostingRegressor()
```

Evaluation of ML Models:

- As you can see , I had called the algorithms, then I called the empty list with the name models [ ], and calling all the model one by one and storing the result in that.



```
In [56]: models= []
         models.append(('Linear Regression',LR))
         models.append(('Lasso Regression',l))
         models.append(('Elastic Net Regression',en))
         models.append(('Ridge Regression',rd))
         models.append(('Decision Tree Regressor',dtr))
         models.append(('KNeighbors Regressor',knr))
         models.append(('RandomForestRegressor',rf))
         models.append(('AdaBoostRegressor',ab))
         models.append(('GradientBoostingRegressor',gb))
```



```
In [57]: #Finding the required metrices for all models together using a for loop
         Model=[]
         score=[]
         cvs=[]
         sd=[]
         mae=[]
         mse=[]
         rmse=[]
         for name,model in models:
             print('******************************',name,'***************************')
             print('\n')
             Model.append(name)
             model.fit(x_train,y_train)
             print(model)
             pre=model.predict(x_test)
             print('\n')
             AS=r2_score(y_test,pre)
             print('r2_score: ',AS)
             score.append(AS*100)
             print('\n')
             sc=cross_val_score(model,x,y,cv=5,scoring='r2').mean()
             print('cross_val_score: ',sc)
             cvs.append(sc*100)
             print('\n')
             std=cross_val_score(model,x,y,cv=5,scoring='r2').std()
             print('Standard Deviation: ',std)
             sd.append(std)
             print('\n')
             MAE=mean_absolute_error(y_test,pre)
             print('Mean Absolute Error: ',MAE)
             mae.append(MAE)
             print('\n')
             MSE=mean_squared_error(y_test,pre)
             print('Mean Squared Error: ',MSE)
             mse.append(MSE)
             print('\n')
             RMSE=np.sqrt(mean_squared_error(y_test,pre))
             print('Root Mean Squared Error: ',RMSE)
             rmse.append(RMSE)
             print('\n\n')
```

As you can observe, I made a for loop and called all the algorithms one by one and appending their result to models.

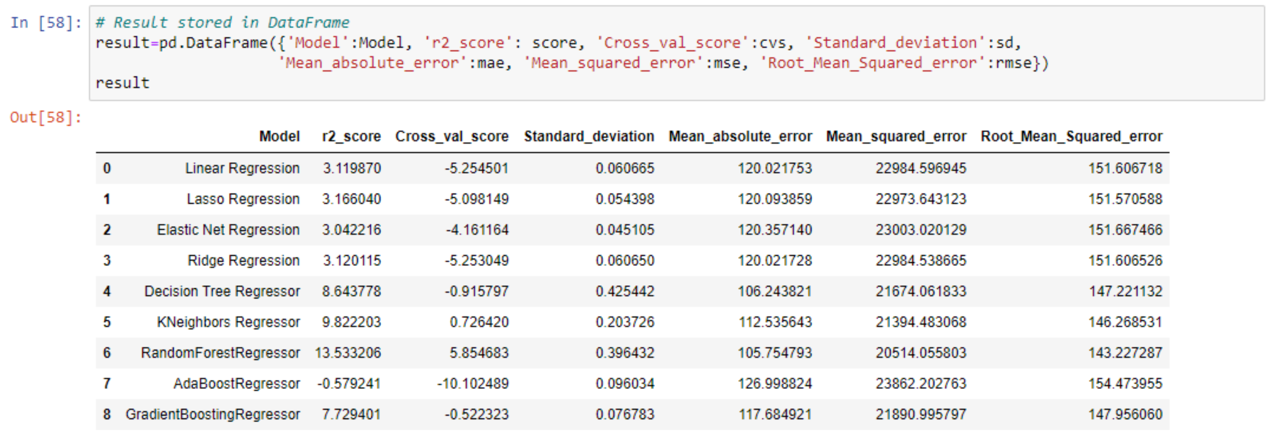
Let me show the output so that we can glance the result in more appropriate way.

Key Metrics for success in solving problem under consideration:

- r2_score
- Mean Absolute Error (MAE)
- Mean Square Error (MSE)
- Root Mean Square Error (RMSE)
- Cross-validation
- Hyperparameter Tuning using GridSearchCV

Result of ML Models:

We saved result of ML Models in DataFrame.



```
In [58]: # Result stored in DataFrame
         result=pd.DataFrame({'Model':Model, 'r2_score': score, 'Cross_val_score':cvs, 'Standard_deviation':sd,
                              'Mean_absolute_error':mae, 'Mean_squared_error':mse, 'Root_Mean_Squared_error':rmse})
         result
Out[58]:
```

	Model	r2_score	Cross_val_score	Standard_deviation	Mean_absolute_error	Mean_squared_error	Root_Mean_Squared_error
0	Linear Regression	3.119870	-5.254501	0.060665	120.021753	22984.596945	151.606718
1	Lasso Regression	3.166040	-5.098149	0.054398	120.093859	22973.643123	151.570588
2	Elastic Net Regression	3.042216	-4.161164	0.045105	120.357140	23003.020129	151.667466
3	Ridge Regression	3.120115	-5.253049	0.060650	120.021728	22984.538665	151.606526
4	Decision Tree Regressor	8.643778	-0.915797	0.425442	106.243821	21674.061833	147.221132
5	KNeighbors Regressor	9.822203	0.726420	0.203726	112.535643	21394.483068	146.268531
6	RandomForestRegressor	13.533206	5.854683	0.396432	105.754793	20514.055803	143.227287
7	AdaBoostRegressor	-0.579241	-10.102489	0.096034	126.998824	23862.202763	154.473955
8	GradientBoostingRegressor	7.729401	-0.522323	0.076783	117.684921	21890.995797	147.956060

We can see that Random Forest Regressor and KNeighbors Regressor are performing well compared to other algorithms. Now we will try Hyperparameter Tuning to find out the best parameters and try to increase the scores.

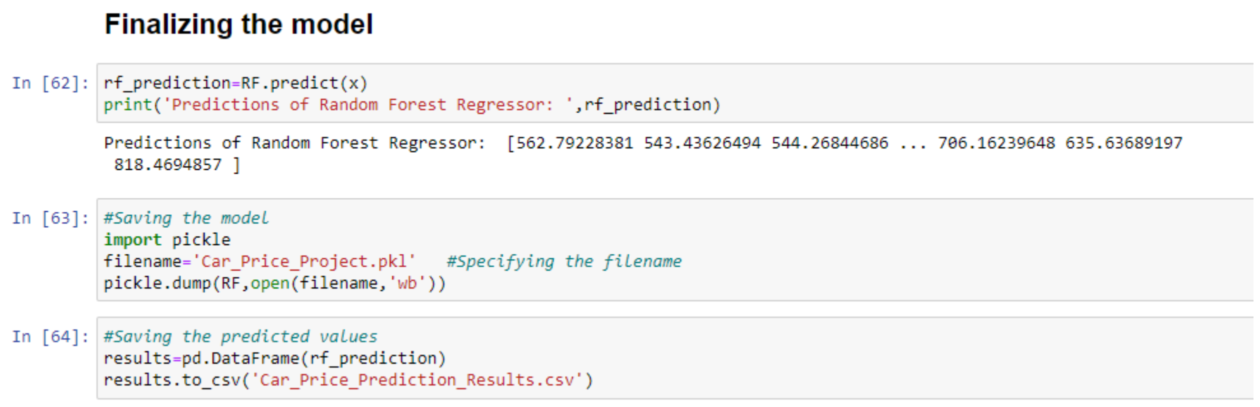
Hyperparameter Tuning:

In order to increase the accuracy score of the model we use hyperparameter tuning of the best model in order to find best parameters by using GridSearchCV() .

After hyperparameter tunning the accuracy score of Random Forest Regressor is increased, **13.53** to **16.84** and Cross validation score is **5.85** to **8.96.**

Saving the model

In order to dump the model which we have developed so that we can use it to make predictions in future, we have saved or dumped the best model.



Finalizing the model

```
In [62]: rf_prediction=RF.predict(x)
         print('Predictions of Random Forest Regressor: ',rf_prediction)

         Predictions of Random Forest Regressor:  [562.79228381 543.43626494 544.26844686 ... 706.16239648 635.63689197
          818.4694857 ]

In [63]: #Saving the model
         import pickle
         filename='Car_Price_Project.pkl'   #Specifying the filename
         pickle.dump(RF,open(filename,'wb'))

In [64]: #Saving the predicted values
         results=pd.DataFrame(rf_prediction)
         results.to_csv('Car_Price_Prediction_Results.csv')
```

Conclusion:

- ❖ After web scraping from cars24.com (using selenium) we making a dataset in csv format.

- ❖ First, we loaded the dataset and did the EDA process and other pre-processing techniques like skewness check and removal, handling the outliers present, filling the missing data, visualizing the distribution of data, etc.

- During the outlier’s analysis, we found that we are losing nearly 6% of data and it’s not a big loss.

- Then we did the model training, building the model and finding out the best model on the basis of different metrics scores we got like r2_score, Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), Cross-validation

- We used algorithms like Linear Regression, Lasso Regression, Ridge Regression, ElasticNet, Decision Tree Regressor, KNeighbors Regressor, Random Forest Regressor, AdaBoost Regressor, Gradient Boosting Regressor

- After performing the analysis, we got KNeighbors Regressor and Random Forest Regressor algorithm as the best algorithms among all. After that finding out the best parameter and improving the scores, we performed Hyperparameter Tuning.

- The problem while doing Hyperparameter Tuning is that it took nearly 2 hours to fetch the best parameters.

- ❖ After Tuning Random Forest Regressor is selected as the final model at R2 Score of **16.84** with Hyper-parameter tuning.

- ❖ We finalized the best model we obtained by saving the model in an pkl file. Also, we found the predictions obtained and saved it in a new data frame.

- ❖ We can improve the data by adding more features that are positively correlated with the target variable, having less outliers, normally distributed values, etc.