FLIP ROBO

PROJECT ON USED CAR PRICE PREDICTION

Submitted by:
JAY PANDEY

FlipRobo SME:
KHUSHBOO GARG

ACKNOWLEDGMENT

I want to thank the Flip Robo Technologies team for giving me the chance to work with this dataset during my internship. It enabled me to develop my analytical abilities.The entire DataTrained team deserves a great thank you.

Reference used in this project:

- GitHub Notes & Repository.
- Various Kaggle and Github projects.
- Analytics Vidya's different papers on Data Science.
- SCIKIT Learn Library Documentation.
- Predicting from www.cardekho.com

Introduction

Business Problem Framing

The manufacturer sets the price of a new car in the market, with some additional expenses paid by the government in the form of taxes.

Therefore, a system that accurately assesses the value of the car utilising a range of features is urgently needed for used car price prediction. With the covid 19 impact in the market, we have seen lot of changes in the car market. The current system involves a procedure where a vendor chooses a price at random and the buyer is unaware of the car and its current market value. In actuality, neither the seller nor the price at which he ought to sell the car have any notion of the current value of the vehicle. In order to solve this issue, a model must be created that can accurately estimate a car's true price rather than just its price range.

Conceptual Background of the Domain Problem

Due to the numerous variables that affect a used automobile's market price, figuring out whether the quoted price of a used car is accurate is a difficult undertaking. The goal of this research is to create machine learning models that can precisely forecast a used car's price based on its attributes so that buyers can make educated decisions. Because the price of a car typically depends on a variety of unique features and factors, accurate car price prediction requires specialist expertise. Usually, brand and model, age, horsepower, and mileage are the most important ones. Due to the frequent changes in fuel prices, the kind of gasoline used in the automobile and fuel consumption per mile have a significant impact on the price of a car.

Review of Literature

The idea of predicting car prices using machine learning techniques.. By combining multiple machine learning methods including Support Vector Machine, Random Forest, and Artificial Neural Networks, the authors of this research suggested an ensemble model. To forecast the price of used automobiles in Bosnia and Herzegovina, they built this model using information from the website www.autopijaca.ba. Their model's accuracy is 87%. Using machine learning techniques, Kanwal Noor and Sadaqat Jan suggested a system for predicting vehicle prices. In this study, a model to forecast car prices using multiple linear regression was suggested. By using the feature selection technique, they selected the feature that had the most impact and eliminated the others. The proposed model's prediction accuracy was nearly 98%.

Motivation for the Problem Undertaken

This problem is a real world dataset. The exposure of this data gives me the opportunity to locate my skills in solving a real time problem. It is the primary motivation to solve this problem. The data required for this project must be extracted from the internet and processed. It can be challenging to decide whether a used car is worth the asking price while viewing listings online.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

The goal of this project is to predict used car price prediction. We make this dataset by web scraping from ***CarDekho Website***. The algorithms are used with their own mathematical equation on background. Here the problem is in two phase.

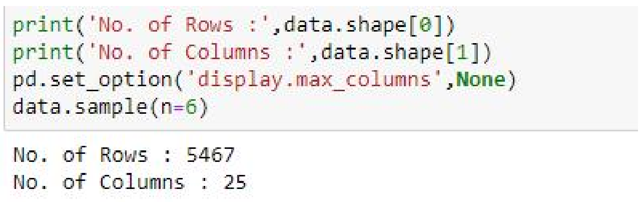
One is: ***Data Collection Phase-*** We have to scrape at least 5000 used cars data from websites (Olx, cardekho, Cars24 etc.), need web scraping for this.

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

This project have one big set of data. All the missing value are imputed. Then different steps of data pre-processing is performed over the dataset like data cleaning, data visualization, relationship between features with target. In model building. Final model is select based on R2 score, Mean absolute error, Root mean square error and Cross Validation score of different algorithms. Here 7 different algorithm are used.

Data Sources and their formats

The data is used for both training and testing the model. It has 5467 rows and 25 columns. The model
will train with the help of this dataset. It has 24 independent features and one dependent or target variable (Price_Rs.)
Later the dataset is divided into two parts, training and testing. After determine the proper model, the model is applied to predict the target variable for the test data.



```
print('No. of Rows :',data.shape[0])
print('No. of Columns :',data.shape[1])
pd.set_option('display.max_columns',None)
data.sample(n=6)
```

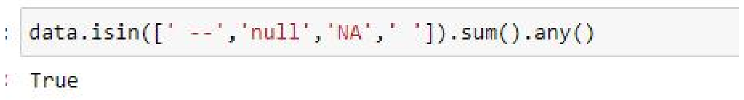
```
No. of Rows : 5467
No. of Columns : 25
```

1. To determine the data format, info() method is used. There are total 4 numarical columns among 25 columns.

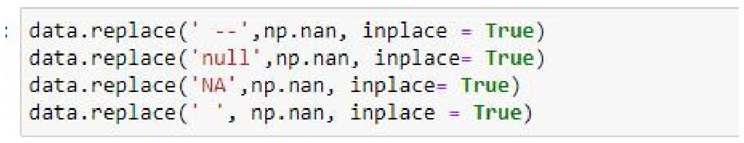
Data Pre-processing Done:

Feature Engineering:

' --', 'null', 'NA', ' ' are present in the total dataset.



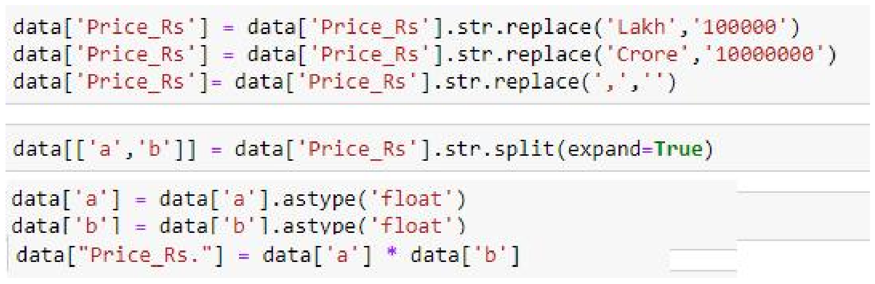
```
data.isin([' --','null','NA',' ']).sum().any()

True
```



```
data.replace(' --',np.nan, inplace = True)
data.replace('null',np.nan, inplace= True)
data.replace('NA',np.nan, inplace= True)
data.replace(' ', np.nan, inplace = True)
```

Change target variable (Price_Rs):

Need to convert in in date format.



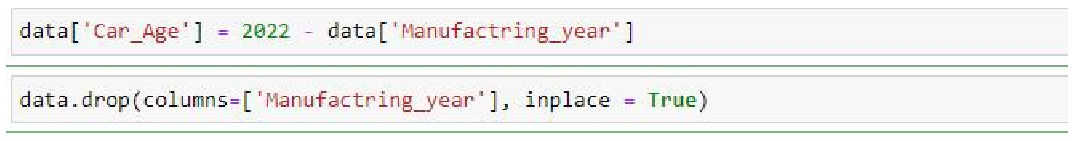
```
data['Price_Rs'] = data['Price_Rs'].str.replace('Lakh','100000')
data['Price_Rs'] = data['Price_Rs'].str.replace('Crore','10000000')
data['Price_Rs']= data['Price_Rs'].str.replace(',','')
```

```
data[['a','b']] = data['Price_Rs'].str.split(expand=True)
```

```
data['a'] = data['a'].astype('float')
data['b'] = data['b'].astype('float')
data["Price_Rs."] = data['a'] * data['b']
```

Transferring Manufacturing_year into Age column:

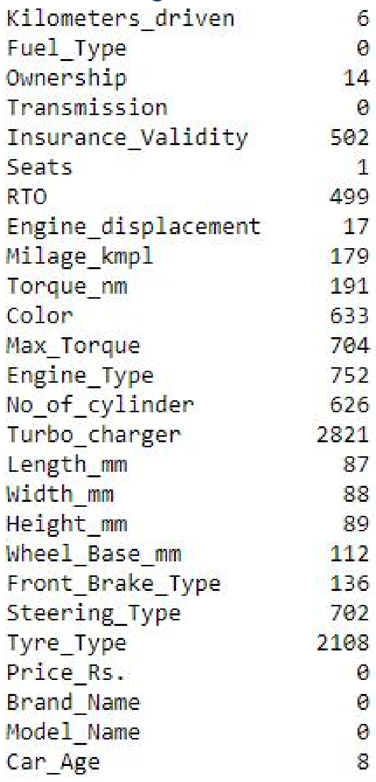
Let’s transfer the Manufacturing year column into corresponding age



```
data['Car_Age'] = 2022 - data['Manufactring_year']
```

```
data.drop(columns=['Manufactring_year'], inplace = True)
```

Checking Null:



```
Kilometers_driven          6
Fuel_Type                  0
Ownership                 14
Transmission               0
Insurance_Validity       502
Seats                      1
RTO                      499
Engine_displacement       17
Milage_kmpl              179
Torque_nm                191
Color                    633
Max_Torque               704
Engine_Type              752
No_of_cylinder           626
Turbo_charger           2821
Length_mm                 87
Width_mm                  88
Height_mm                 89
Wheel_Base_mm            112
Front_Brake_Type         136
Steering_Type            702
Tyre_Type               2108
Price_Rs.                  0
Brand_Name                 0
Model_Name                 0
Car_Age                    8
```

There are a huge set of null data in most of the features.

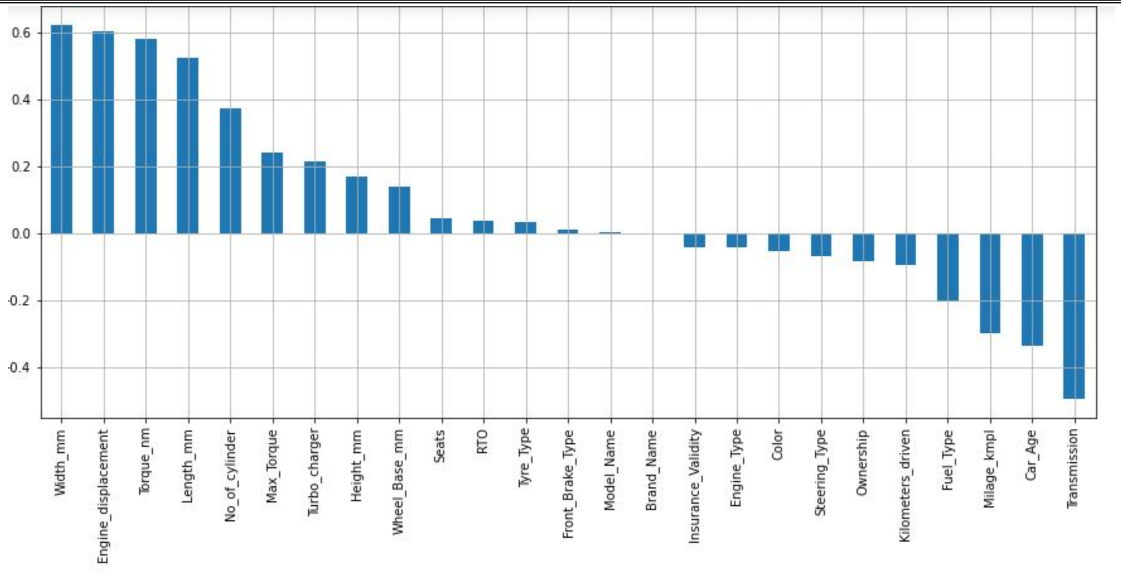
Now use describe() method to check the statistical features. Following observations are noted.

1. Null value is present.
2. Seems outliers are present as there is a difference between 75% and max of some features.
3. Minimum Engine_displacement is 0. Seems it is a error,
4. Minimum Length_mm is 4 where maximim 5453.
5. Minimum Car_age is 0 where maximum is 20 years.
6. In maximum cases Engine_Type is In-Line Engine

Correlation:

Observations of the correlation:

1. Brand_Name and Model_Name is very less correlated with target variable.
2. Maximum correlation observe in Width and Engine_displacement followed by torque & length.
3. Most of features are moderately & poorly correlated with each other.



Outliers Detection and Removal:

Let’s check Outliers. From the previous Boxplot , it is seen that there are some outiers in numarical featues columns. But it is a realistic dataset. So let's keep it.

Data Inputs- Logic- Output Relationships

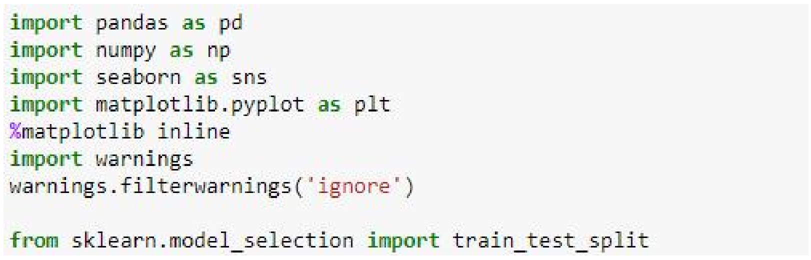
We can see in the correlation that every features are correlated with each other and also they are highly
correlated with target variable label.

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

No such assumptions are taken for this case.

Hardware and Software Requirements and Tools Used

Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz
RAM: 4.00 GB
System Type: 64-bit operating system, x64-based processor
Window: Windows 10 Pro
Anaconda – Jupyter Notebook
Libraries Used



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

Except this, different libraries are used for machine learning model building from sklearn.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods):

In this problem regression-based machine learning algorithm like linear regression can be used. For that first data encoding and data scaling using standard scalar is done. For building an appropriate ML model before implementing classification algorithms, data is split in training & test data using train_test_split.

Testing of Identified Approaches (Algorithms)

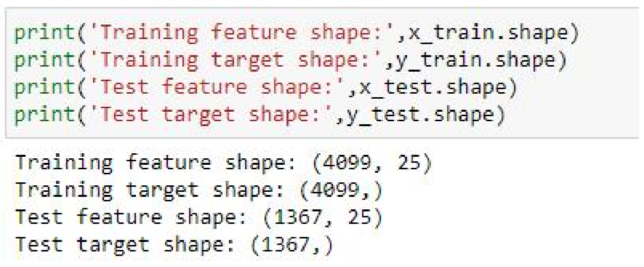
Total 7 algorithms used for the training and testing are:
1. Linear Regression
2. DecisionTree Regressor
3. KNeighbors Regressor
4. GradientBoosting Regressor
5. RandomForest Regressor
6. Support Vector Regression
7. AdaBoost Regressor

Key Metrics for success in solving problem under consideration:

From metrics module of sklearn library import mean_absolute_error, mean_squared_error, r2_score. From model_selection also, we use cross_val_score. Those are the matrices use to validate the model's quality. R2 Score: R-squared (R2) is a statistical measure that shows how much of a dependent variable's variance is explained by one or more independent variables in a regression model.

Run and Evaluate selected models

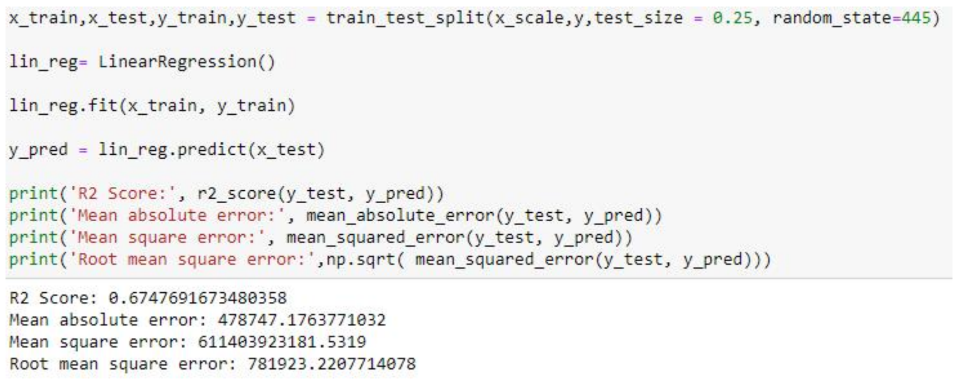
First find the best random state of train_test_split to get best accuracy. Here the random state is 445. Then after splitting the data into 4 different part and check the shape of the data.



```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

```
Training feature shape: (4099, 25)
Training target shape: (4099,)
Test feature shape: (1367, 25)
Test target shape: (1367,)
```

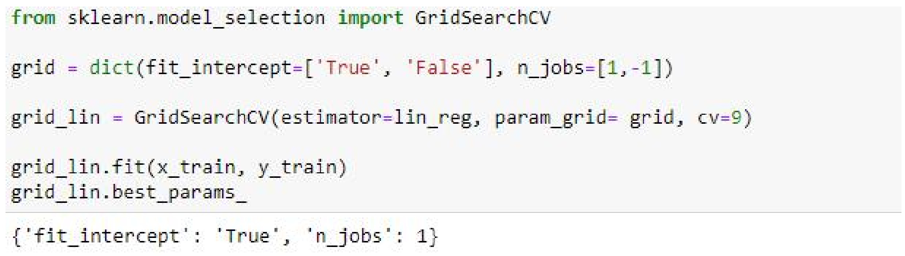
A Linear Regression:



```
x_train,x_test,y_train,y_test = train_test_split(x_scale,y,test_size = 0.25, random_state=445)

lin_reg= LinearRegression()

lin_reg.fit(x_train, y_train)

y_pred = lin_reg.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

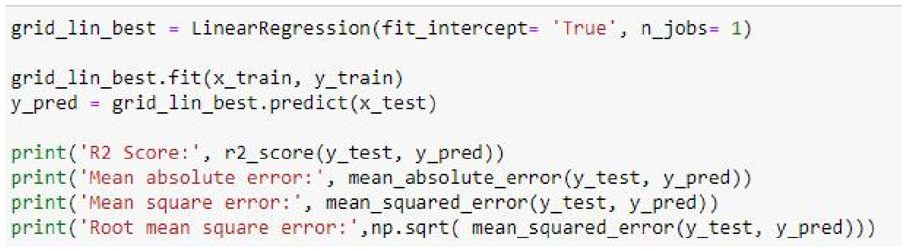
```
R2 Score: 0.6747691673480358
Mean absolute error: 478747.1763771032
Mean square error: 611403923181.5319
Root mean square error: 781923.2207714078
```

Then apply Hyper Parameter Tuning



```
from sklearn.model_selection import GridSearchCV

grid = dict(fit_intercept=['True', 'False'], n_jobs=[1,-1])

grid_lin = GridSearchCV(estimator=lin_reg, param_grid= grid, cv=9)

grid_lin.fit(x_train, y_train)
grid_lin.best_params_
```

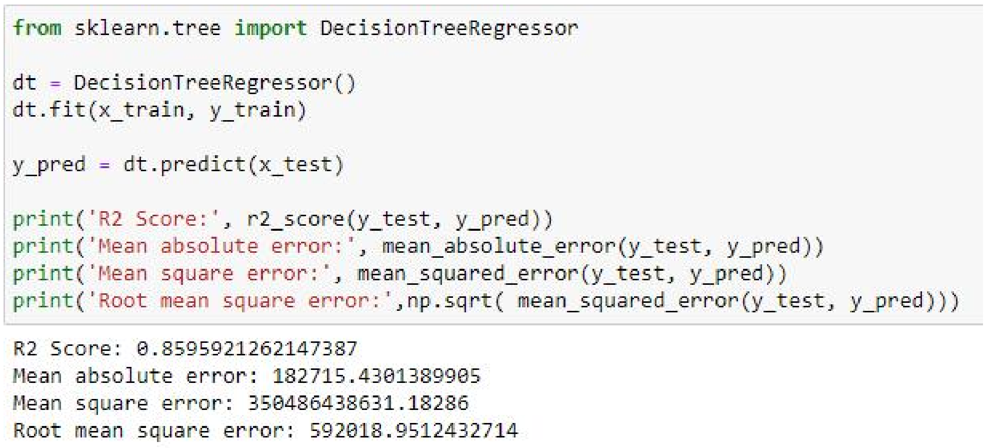
```
{'fit_intercept': 'True', 'n_jobs': 1}
```



```
grid_lin_best = LinearRegression(fit_intercept= 'True', n_jobs= 1)

grid_lin_best.fit(x_train, y_train)
y_pred = grid_lin_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

No such improvement seen after GridSearchCV.

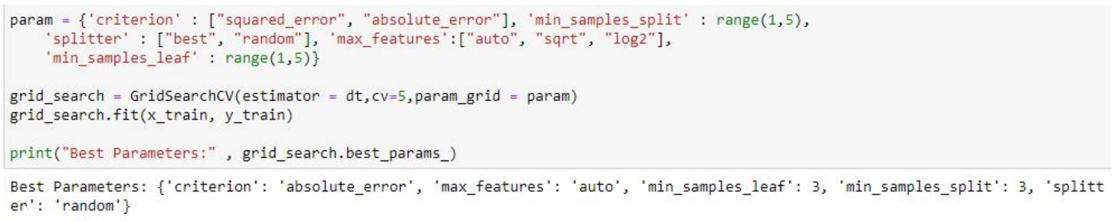
B Decision Tree Regressor:



```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)

y_pred = dt.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

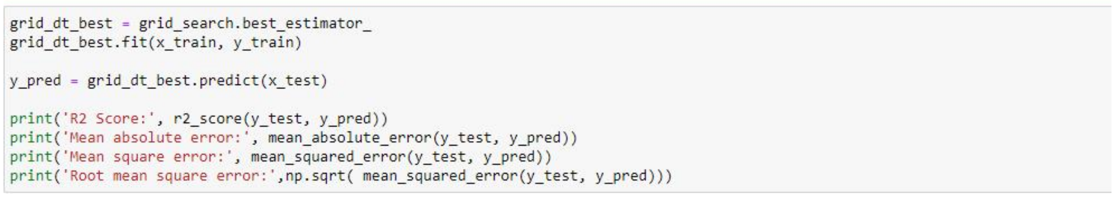
```
R2 Score: 0.8595921262147387
Mean absolute error: 182715.4301389905
Mean square error: 350486438631.18286
Root mean square error: 592018.9512432714
```

Then apply Hyper Parameter Tuning.



```
param = {'criterion' : ["squared_error", "absolute_error"], 'min_samples_split' : range(1,5),
    'splitter' : ["best", "random"], 'max_features':["auto", "sqrt", "log2"],
    'min_samples_leaf' : range(1,5)}

grid_search = GridSearchCV(estimator = dt,cv=5,param_grid = param)
grid_search.fit(x_train, y_train)

print("Best Parameters:" , grid_search.best_params_)
```

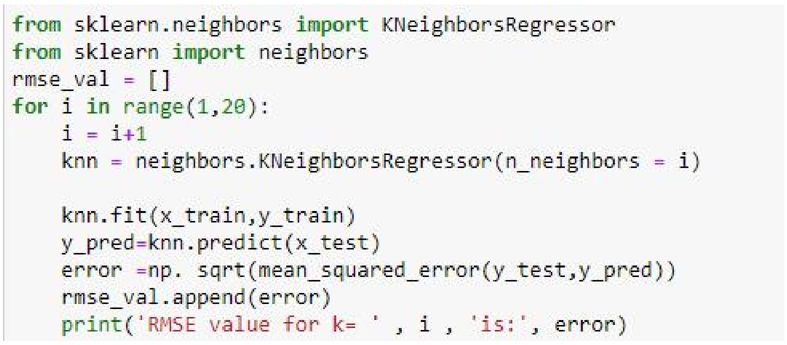
```
Best Parameters: {'criterion': 'absolute_error', 'max_features': 'auto', 'min_samples_leaf': 3, 'min_samples_split': 3, 'splitt
er': 'random'}
```



```
grid_dt_best = grid_search.best_estimator_
grid_dt_best.fit(x_train, y_train)

y_pred = grid_dt_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

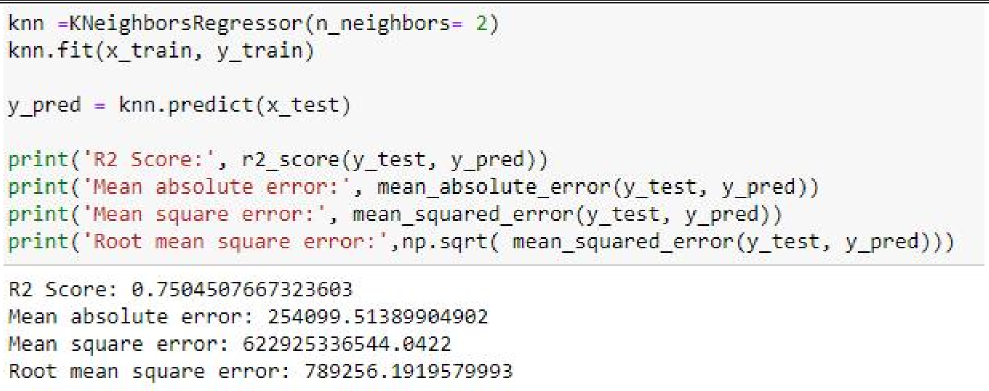
After using Gridseaech CV, R2 is not improved.

C KNeighbors Regressor:



```
from sklearn.neighbors import KNeighborsRegressor
from sklearn import neighbors
rmse_val = []
for i in range(1,20):
    i = i+1
    knn = neighbors.KNeighborsRegressor(n_neighbors = i)

    knn.fit(x_train,y_train)
    y_pred=knn.predict(x_test)
    error =np. sqrt(mean_squared_error(y_test,y_pred))
    rmse_val.append(error)
    print('RMSE value for k= ' , i , 'is:', error)
```

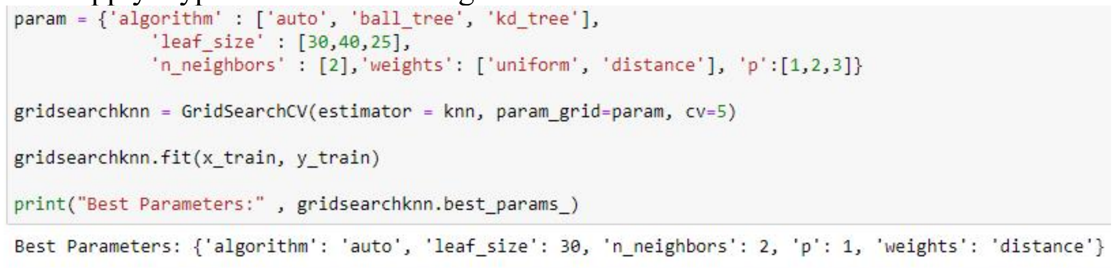
For k=2 we get the best RMSE value for KNeighborsRegressor()



```
knn =KNeighborsRegressor(n_neighbors= 2)
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

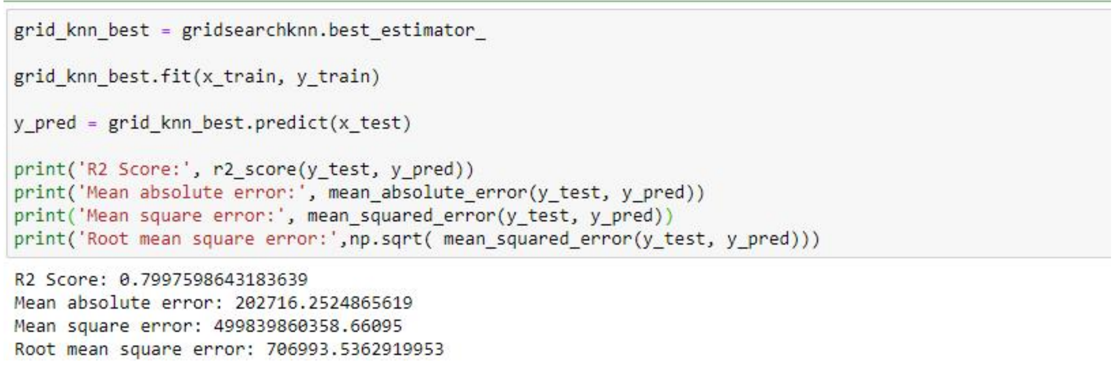
```
R2 Score: 0.7504507667323603
Mean absolute error: 254099.51389904902
Mean square error: 622925336544.0422
Root mean square error: 789256.1919579993
```

Then apply Hyper Parameter Tuning.



```
param = {'algorithm' : ['auto', 'ball_tree', 'kd_tree'],
             'leaf_size' : [30,40,25],
             'n_neighbors' : [2],'weights': ['uniform', 'distance'], 'p':[1,2,3]}

gridsearchknn = GridSearchCV(estimator = knn, param_grid=param, cv=5)

gridsearchknn.fit(x_train, y_train)

print("Best Parameters:" , gridsearchknn.best_params_)
```

```
Best Parameters: {'algorithm': 'auto', 'leaf_size': 30, 'n_neighbors': 2, 'p': 1, 'weights': 'distance'}
```

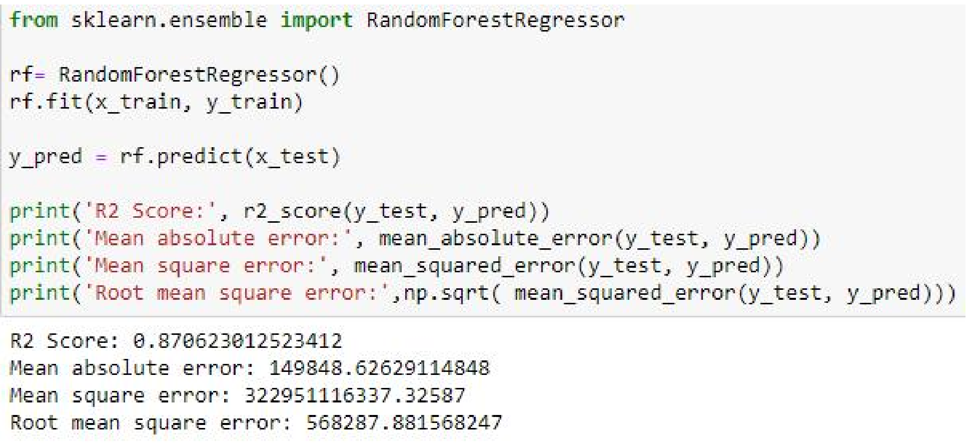


```
grid_knn_best = gridsearchknn.best_estimator_

grid_knn_best.fit(x_train, y_train)

y_pred = grid_knn_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.7997598643183639
Mean absolute error: 202716.2524865619
Mean square error: 499839860358.66095
Root mean square error: 706993.5362919953
```

Nothing is improved after GridSearchCV

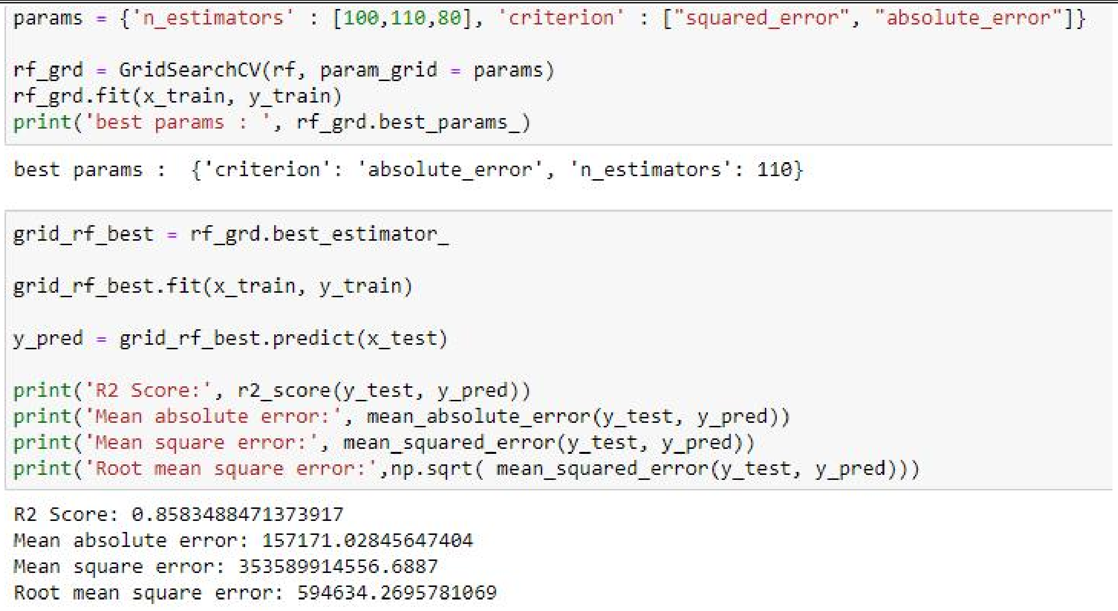
D Random Forest Regressor:



```
from sklearn.ensemble import RandomForestRegressor

rf= RandomForestRegressor()
rf.fit(x_train, y_train)

y_pred = rf.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.870623012523412
Mean absolute error: 149848.62629114848
Mean square error: 322951116337.32587
Root mean square error: 568287.881568247
```

Then apply Hyper Parameter Tuning.



```python
params = {'n_estimators' : [100,110,80], 'criterion' : ["squared_error", "absolute_error"]}

rf_grd = GridSearchCV(rf, param_grid = params)
rf_grd.fit(x_train, y_train)
print('best params : ', rf_grd.best_params_)
```

```
best params :  {'criterion': 'absolute_error', 'n_estimators': 110}
```

```python
grid_rf_best = rf_grd.best_estimator_

grid_rf_best.fit(x_train, y_train)

y_pred = grid_rf_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.8583488471373917
Mean absolute error: 157171.02845647404
Mean square error: 353589914556.6887
Root mean square error: 594634.2695781069
```

R2 score, RMSE is not improved after GridSearchCV.

CONCLUSION

Key Findings and Conclusions of the Study

• As car model get old eventually its price reduces with time.
• In terms of Avg. Price as number of cylinders increases the average price increases.
• Electric cars are a very tiny market and also relatively expensive when compared to gasoline powered vehicles.

Learning Outcomes of the Study in respect of Data Science

• Null removal is an important part of any problem.
• Scaling and standardization of data is mandatory.
• Data needs to be much precise and detailed for much better score.

Limitations of this work and Scope for Future Work

• We can scrape more information from many internet marketplaces like olx and car24. Clearly, more information leads to more accurate forecasting.
• R2 score can increase with hyper parameter tuning with several different parameter. As it takes a lot of time, I am not able to use lot of parameters here for tuning.