Data Science Project

on

Car Price Prediction using Machine Learning.

BACHELOR OF TECHNOLOGY DEGREE

Session 2022-23

in

CSE-Data Science

By:

Aditya Gupta

2100321540007

Under the guidance of:

Ms. Dimple Tiwari
Assistant Professor

DEPARTMENT OF CSE-DS
ABES ENGINEERING COLLEGE, GHAZIABAD

Estd. 2000

NAAC Accredited

AFFILIATED TO
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW
(Formerly UPTU)

Preparing Data

The selected dataset for study of car price contains labels that includes Car name, Year, Selling Price, Present Price, Kms Driven, Fuel Type, Seller Type, Transmission, and the number of previous owners. The dataset serves as a valuable resource for developing and training machine learning models to predict the selling price of a used car based on different factors. The data has been collected through various sources, including online car marketplaces and dealerships.

Overall, the car price dataset provides a valuable resource for researchers and data scientists interested in analyzing the used car market and developing predictive models to help buyers and sellers make informed decisions.

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
```

In [2]:
```
data = pd.read_csv('car data.xls')
```

In [3]:
```
data
```

Out[3]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0
...	...	...	...	...	...	...	...	...	...
296	city	2016	9.50	11.60	33988	Diesel	Dealer	Manual	0
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	Manual	0
298	city	2009	3.35	11.00	87934	Petrol	Dealer	Manual	0
299	city	2017	11.50	12.50	9000	Diesel	Dealer	Manual	0
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	Manual	0

301 rows × 9 columns

Labeled Data

The pandas library provides a useful method called describe() that can be used to calculate summary statistics for the numerical columns in a data frame. These statistics include measures such as the mean, standard deviation, minimum and maximum values, quartiles, and more. By using the describe() method, we can quickly get an overview of the distribution and range of values for each numerical column in the data frame.

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Seller_Type    301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

In [10]:
```
data.describe()
```

Out[10]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

Optimizing Data

Data preprocessing is an essential step in any data analysis or machine learning project. It involves cleaning, transforming, and preparing raw data into a format that can be easily analyzed or used to train machine learning models. Common data preprocessing techniques include handling missing or erroneous data, scaling and normalizing features, encoding categorical variables, and splitting the data into training and testing sets. Effective data preprocessing can improve the accuracy and efficiency of machine learning models and lead to more meaningful insights from data analysis.

7. Data Preprocessing

In [11]:
```
data.head(1)
```

Out[11]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0

In [12]:
```
import datetime
```

In [13]:
```
date_time = datetime.datetime.now()
```

In [14]:
```
data['Age']=date_time.year - data['Year']
```

In [15]:
```
data.head()
```

Out[15]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	9
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	10
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	6
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	12
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	9

In [16]:
```
data.drop('Year',axis=1,inplace=True)
```

In [17]:
```
data.head()
```

Out[17]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	0	9
1	sx4	4.75	9.54	43000	Diesel	Dealer	Manual	0	10
2	ciaz	7.25	9.85	6900	Petrol	Dealer	Manual	0	6
3	wagon r	2.85	4.15	5200	Petrol	Dealer	Manual	0	12
4	swift	4.60	6.87	42450	Diesel	Dealer	Manual	0	9

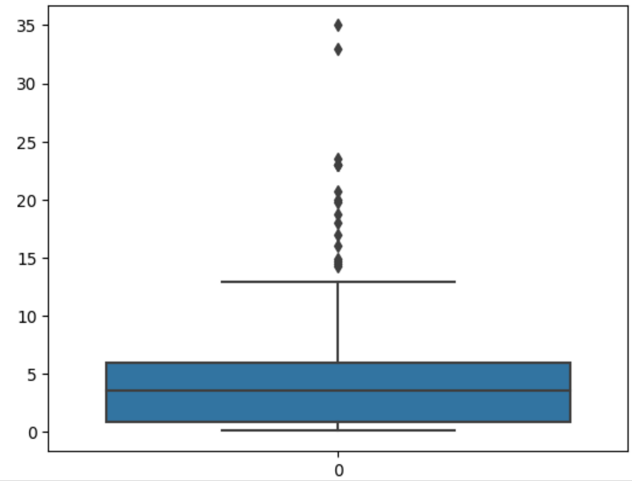
Data outliers are data points that significantly deviate from the expected or normal range of values in a dataset. Outliers can have a significant impact on statistical analysis and machine learning models, leading to inaccurate results and biased predictions. It's important to identify and handle outliers appropriately, either by removing them from the dataset or by transforming them to reduce their impact on the analysis.

Outlier Removal

```
In [18]: import seaborn as sns

In [19]: sns.boxplot(data['Selling_Price'])

Out[19]: <AxesSubplot: >
```




```
In [20]: sorted(data['Selling_Price'],reverse=True)

Out[20]: [35.0,
          33.0,
          23.5,
          23.0,
          23.0,
          23.0,
          20.75,
          19.99,
          19.75,
          18.75,
          18.0,
          17.0,
          16.0,
          14.9,
          14.73,
          14.5,
          14.25,
          12.9,
          12.5,

In [21]: data = data[~(data['Selling_Price']>=33.0) & (data['Selling_Price']<=35.0)]

In [22]: data.shape

Out[22]: (299, 9)
```

Encoding the Categorical Columns

In [23]:
```
data.head(1)
```

Out[23]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	0	9

In [24]:
```
data['Fuel_Type'].unique()
```

Out[24]:
```
array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

In [25]:
```
data['Fuel_Type'] = data['Fuel_Type'].map({'Petrol':0,'Diesel':1,'CNG':2})
```

In [26]:
```
data['Fuel_Type'].unique()
```

Out[26]:
```
array([0, 1, 2], dtype=int64)
```

In [27]:
```
data['Seller_Type'].unique()
```

Out[27]:
```
array(['Dealer', 'Individual'], dtype=object)
```

In [28]:
```
data['Seller_Type'] = data['Seller_Type'].map({'Dealer':0,'Individual':1})
```

In [29]:
```
data['Seller_Type'].unique()
```

Out[29]:
```
array([0, 1], dtype=int64)
```

In [30]:
```
data['Transmission'].unique()
```

Out[30]:
```
array(['Manual', 'Automatic'], dtype=object)
```

In [31]:
```
data['Transmission'] =data['Transmission'].map({'Manual':0,'Automatic':1})
```

In [32]:
```
data['Transmission'].unique()
```

Out[32]:
```
array([0, 1], dtype=int64)
```

In [33]:
```
data.head()
```

Out[33]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	ritz	3.35	5.59	27000	0	0	0	0	9
1	sx4	4.75	9.54	43000	1	0	0	0	10
2	ciaz	7.25	9.85	6900	0	0	0	0	6
3	wagon r	2.85	4.15	5200	0	0	0	0	12
4	swift	4.60	6.87	42450	1	0	0	0	9

8. Store Feature Matrix In X and Response(Target) In Vector y

In [34]:
```
X = data.drop(['Car_Name','Selling_Price'],axis=1)
y = data['Selling_Price']
```

In [35]:
```
y
```

Out[35]:
```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 299, dtype: float64
```

9. Splitting The Dataset Into The Training Set And Test Set

In [36]:
```
from sklearn.model_selection import train_test_split
```

In [37]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=42)
```

Comparing Models

10. Import The models

In [38]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

from sklearn.tree import DecisionTreeRegressor
```

11. Model Training

In [39]:
```
lr = LinearRegression()
lr.fit(X_train,y_train)

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

dt = DecisionTreeRegressor()
dt.fit(X_train,y_train)
```

Out[39]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

12. Prediction on Test Data

In [40]:
```
y_pred1 = lr.predict(X_test)
y_pred2 = rf.predict(X_test)
y_pred3 = dt.predict(X_test)
```

13. Evaluating the Algorithm

In [41]:
```
from sklearn import metrics
```

In [42]:
```
score1 = metrics.r2_score(y_test,y_pred1)
score2 = metrics.r2_score(y_test,y_pred2)
score3 = metrics.r2_score(y_test,y_pred3)
```

In [43]:
```
print(score1,score2,score3)
```

```
0.6790884983129397 0.745952792380107 0.6482064492647674
```

In [44]:
```
final_data = pd.DataFrame({'Models':['LR','RF','DT'],
             "R2_SCORE":[score1,score2,score3]})
```

In [45]:
```
final_data
```

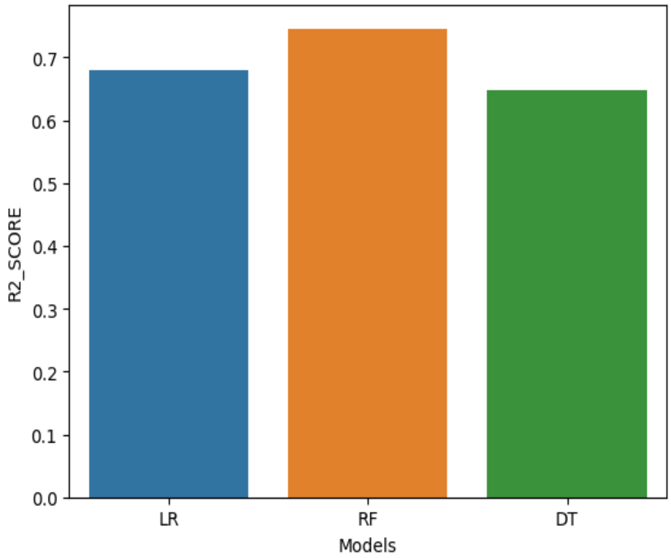
Out[45]:

	Models	R2_SCORE
0	LR	0.679088
1	RF	0.745953
2	DT	0.648206

Visualization of Result

```
In [46]: sns.barplot(x='Models', y='R2_SCORE', data=final_data)

Out[46]: <AxesSubplot: xlabel='Models', ylabel='R2_SCORE'>
```



Predicting New Model

```
In [60]: data_new = pd.DataFrame({
    'Present_Price':5.59,
    'Kms_Driven':27000,
    'Fuel_Type':0,
    'Seller_Type':0,
    'Transmission':0,
    'Owner':1,
    'Age':8
},index=[0])
y_new_pred = model.predict(data_new)

print("Predicted target variable for new data:", y_new_pred)

Predicted target variable for new data: [3.35]
```