FLIP ROBO

Car Price Prediction

Submitted by: **Haindavi Chakravarthi**

Internship: **23**

ACKNOWLEDGEMENT

The successful completion of any work would be always be incomplete unless we mention the valuable cooperation and assistance of those people who were a source of constant guidance and encouragement, they served as bacon light and crowned our efforts with success. First of all, I would like to thank all my mentors in Data Trained and FlipRobo Technologies for this opportunity. I wish to express our sincere thanks to the above people, without whom I would not have been able to complete this project. I thank that I got the chance to do this project because this project has given me a lot many thoughts whether in scrapping or in the preparation of the model, it helped me to regain my knowledge levels and also helped to smoothly handle the projects, finally this project has given a good idea of handling the projects.

INTRODUCTION

- 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.

- Predicting the price of used cars is both an important and interesting problem. The market value is based on a number of factors, including demand, supply, options, and incentives. The market value of a vehicle usually falls somewhere between the sticker price and the invoice price. Because the market value is an average, some people will pay more than that amount, while others will pay less.

- A car's value is determined by many factors: the popularity of the make and model of your car, vehicle specifications, trim levels, physical appearance, mileage, consistent maintenance and working condition.

- Using this as a base, I have collected the data from **“Cars24”** website and here I have collected and scrapped the information of the cars with definite and significant features required for predicting the better model.

- Once the data is collected, the data will be cleaned and pre-processed with all the necessary tools and the same will be used to build machine learning models in order to predict the price of the same.

Analytical Problem Framing:

- Here our dataset has 7039 rows and 16 columns, using this dataset we will be building the model followed by training the data and then finally the model is tested by using 70% of the training data and 30% of the testing data.

- As we have pre-processed the data by removing the null values there may be chances of getting outliers and certain unknown values.

- The following are the columns or features of our dataset:

```
Name
Brand
Model
Transmission
Year of Purchase
Kilometers Driven
Last Service
Fuel Type
Owner
Insurance
History
Location
EMI per month
Price
```

- These columns are named when the data is collected but during the pre-processing there are few newly created columns which are extracted from the existing data and thus the number of columns increased and the columns are renamed.

Importing libraries :

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

- Primarily these are the basic libraries which are imported and further the libraries necessary are imported.

Collecting the data :

```
data1 = pd.read_csv("cars1.csv")
data2 = pd.read_csv("cars2.csv")
data3 = pd.read_csv("cars3.csv")
data4 = pd.read_csv("cars4.csv")
data5 = pd.read_csv("cars5.csv")
data6 = pd.read_csv("cars6.csv")
data7 = pd.read_csv("cars7.csv")
data8 = pd.read_csv("cars8.csv")
data9 = pd.read_csv("cars9.csv")
data10 = pd.read_csv("cars10.csv")
data11 = pd.read_csv("cars11.csv")
data12 = pd.read_csv("cars12.csv")
data13 = pd.read_csv("cars13.csv")
```

➤ Here we have just collected the scraped data into different dataframes which needs to be concatenated for the complete data frame.

Combining the data into a dataframe :

```
data = pd.concat([data1,data2,data3,data4,data5,data6,data7,data8,data9,data10,data11,data12,data13], axis = 0, ignore_index=True
data.head()
```

	Name	Brand	Model	Transmission	Year of Purchase	Kilometers Driven	Last Service	Fuel Type	Owner	Insurance	History	Location	EMI per month	Price
0	Swift VDI ABS MANUAL	Maruti	['VDI', 'ABS']	MANUAL	Mar-15	76,264 km	76,264km (22 Nov 2021)	Diesel	1st Owner	Valid upto Mar 2023 3rd Party	Non-Accidental	DELHI	₹9,840/month	₹4,26,499
1	Swift ZDI MANUAL	Maruti	['ZDI']	MANUAL	Jul-14	92,088 km	92,088km (08 Mar 2022)	Diesel	1st Owner	Valid upto Mar 2023 3rd Party	Non-Accidental	DELHI	₹9,710/month	₹4,20,799
2	Baleno ALPHA DDIS 190 MANUAL	Maruti	['ALPHA', 'DDIS', '190']	MANUAL	Jan-16	67,332 km	67,332km (16 Nov 2021)	Diesel	1st Owner	Valid upto Jul 2022 Third Party	Non-Accidental	DELHI	₹13,612/month	₹5,92,199
3	Baleno ALPHA DDIS 190 MANUAL	Maruti	['ALPHA', 'DDIS', '190']	MANUAL	Nov-15	76,135 km	76,135km (07 Mar 2022)	Diesel	2nd Owner	Valid upto Mar 2023 3rd Party	Non-Accidental	DELHI	₹12,672/month	₹5,50,899
4	Baleno ZETA 1.2 K12 MANUAL	Maruti	['ZETA', '1.2', 'K12']	MANUAL	Mar-18	37,786 km	37,786km (09 Feb 2022)	Petrol	1st Owner	Valid upto Mar 2023 3rd Party	Non-Accidental	DELHI	₹14,546/month	₹6,33,199

Observation : Here we can see that there is a lot of preprocessing to be done in the data for the better model and accuracy.

➤ Here the data is concatenated and the dataframe is formed.

Information of the data :

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7039 entries, 0 to 7038
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                7039 non-null   object
 1   Brand               7039 non-null   object
 2   Model               7039 non-null   object
 3   Transmission        6956 non-null   object
 4   Year of Purchase    7039 non-null   object
 5   Kilometers Driven   7039 non-null   object
 6   Last Service        7039 non-null   object
 7   Fuel Type           7039 non-null   object
 8   Owner               7039 non-null   object
 9   Insurance           7039 non-null   object
 10  History             7039 non-null   object
 11  Location            6498 non-null   object
 12  EMI per month       7039 non-null   object
 13  Price               7039 non-null   object
dtypes: object(14)
memory usage: 770.0+ KB
```

Observation : Here we can see that all the columns are with Object datatype and also we can see that the data has null values in few columns which we have fill during the preprocessing of the data.

Null values of the data :

```
data.isnull().sum()
```

```
Name                   0
Brand                  0
Model                  0
Transmission          83
Year of Purchase       0
Kilometers Driven      0
Last Service           0
Fuel Type              0
Owner                  0
Insurance              0
History                0
Location             541
EMI per month          0
Price                  0
dtype: int64
```

Observation : Here we can see that the columns "Transmission" and "Location" have null values in them.

Description of the data :

```
pd.set_option("display.max_columns",None)
data.describe()
```

	Name	Brand	Model	Transmission	Year of Purchase	Kilometers Driven	Last Service	Fuel Type	Owner	Insurance	History	Location	EMI per month	Price
count	7039	7039	7039	6956	7039	7039	7039	7039	7039	7039	7039	6498	7039	7039
unique	781	24	669	2	256	4371	4473	4	4	96	1	14	2906	3170
top	Baleno DELTA 1.2 K12 MANUAL	Maruti	['VXI']	MANUAL	Jan-14	36,696 km	1,12,006km (03 Jan 2022)	Petrol	1st Owner	Valid upto Mar 2023 3rd Party	Non-Accidental	DELHI	₹0/month	₹3,44,999
freq	241	3636	816	6024	167	7	7	4501	5610	5032	7039	1853	541	14

Observation : Here we can see the unique values,frequently repeated values of the columns and also we can observe that the statistical information of the data is not presented as all the columns are with object type of the data.

- The dataset has only object datatype and so statistical description is not done over here.

- **Here the pre-processing of the data starts:** Here I have used splitting concept for the extraction of the required data from the extracted data and thus created new columns.

```
x = data["Name"]
```

```
list0 = []
for i in x:
    list0.append(i.split(' ')[0])
print(list0)
```

```
['Swift', 'Swift', 'Baleno', 'Baleno', 'Baleno', 'Swift', 'Swift', 'Swift', 'Ciaz', 'Ciaz', 'Baleno', 'Ertiga', 'Baleno', 'Sw
ift', 'Ciaz', 'Swift', 'Swift', 'Baleno', 'Ertiga', 'Alto', 'Swift', 'Swift', 'Swift', 'Ritz', 'Swift', 'Baleno', 'Baleno',
'Swift', 'Swift', 'Swift', 'Swift', 'Ciaz', 'Swift', 'Swift', 'Swift', 'Baleno', 'Swift', 'Baleno', 'Swift', 'Swift', 'Swif
t', 'Swift', 'Swift', 'Ciaz', 'Alto', 'Swift', 'Dzire', 'Baleno', 'Swift', 'Baleno', 'Swift', 'Baleno', 'Baleno', 'Swift', 'S
wift', 'Alto', 'Baleno', 'Swift', 'Swift', 'Swift', 'Swift', 'Swift', 'Swift', 'Ritz', 'Swift', 'Baleno', 'Baleno', 'Swift',
'Swift', 'Swift', 'Swift', 'Ciaz', 'Swift', 'Swift', 'Swift', 'Baleno', 'Swift', 'Baleno', 'Swift', 'Swift', 'IGNIS', 'Swif
t', 'Swift', 'Ertiga', 'Swift', 'Alto', 'Swift', 'Swift', 'Swift', 'Swift', 'IGNIS', 'Swift', 'Swift', 'Ertiga', 'Swift', 'Sw
ift', 'Swift', 'Ertiga', 'Baleno', 'Swift', 'Swift', 'Baleno', 'Swift', 'Swift', 'Baleno', 'Baleno', 'Swift', 'Swift', 'Swif
t', 'Ciaz', 'Swift', 'Swift', 'Baleno', 'Swift', 'Swift', 'Baleno', 'Swift', 'Swift', 'Swift', 'Swift', 'Baleno', 'Swift', 'S
wift', 'Swift', 'Swift', 'Dzire', 'Ritz', 'Dzire', 'Swift', 'Swift', 'Swift', 'Ritz', 'Baleno', 'Dzire', 'Dzire', 'Ertiga',
'Baleno', 'Ritz', 'Swift', 'Ertiga', 'Celerio', 'Swift', 'Swift', 'Swift', 'Celerio', 'Swift', 'Swift', 'Alto', 'Swift', 'Swi
ft', 'Swift', 'Ciaz', 'Dzire', 'Swift', 'Ciaz', 'Ertiga', 'Baleno', 'Swift', 'Baleno', 'Swift', 'Swift', 'Ciaz', 'Swift', 'Sw
ift', 'Baleno', 'Swift', 'Swift', 'Baleno', 'Swift', 'Swift', 'Swift', 'Swift', 'Swift', 'Ritz', 'Swift', 'Swift', 'Baleno',
'Baleno', 'Dzire', 'IGNIS', 'Baleno', 'Dzire', 'Celerio', 'Swift', 'Baleno', 'Baleno', 'Swift', 'Swift', 'Swift', 'Swift', 'S
wift', 'Swift', 'Baleno', 'Ciaz', 'Swift', 'Baleno', 'Swift', 'Swift', 'Ritz', 'IGNIS', 'Swift', 'Ertiga', 'Baleno', 'Swift',
'Swift', 'Swift', 'Swift', 'IGNIS', 'Swift', 'Swift', 'Ertiga', 'Swift', 'Ertiga', 'Swift', 'Ertiga', 'Swift', 'Swift', 'Alt
o', 'Swift', 'Dzire', 'Swift', 'Baleno', 'Swift', 'Swift', 'Alto', 'Swift', 'Swift', 'Swift', 'Swift', 'Ciaz', 'Alto', 'Alt
o', 'S', 'Swift', 'Alto', 'Alto', 'Swift', 'S', 'Vitara', 'Swift', 'Alto', 'Swift', 'Vitara', 'Vitara', 'Swift', 'Vitara', 'V
itara', 'Swift', 'Vitara', 'Alto', 'Alto', 'S', 'Alto', 'Celerio', 'Swift', 'Alto', 'S', 'Alto', 'Vitara', 'Swift', 'Alto',
```

```
Name_of_the_car = list0
```

```
df = pd.DataFrame()
df["Name_of_the_car"] = list0
df
```

	Name_of_the_car
0	Swift
1	Swift
2	Baleno
3	Baleno
4	Baleno
...	...
7034	Alto
7035	
7036	maze
7037	
7038	

7039 rows × 1 columns

```
Car_name = df
```

```
data.insert(len(data.columns),"Car_name",Car_name.values)
```

Observation : Here we have inserted the craeted column into the dataframe.

```
data.drop(['Name'],axis = 1,inplace = True)
```

Observation : Here we have dropped the parent column "Name".

- Here the preprocessing the column "name" is done and the new column got created.

Year of Purchase :

```
yea = data["Year of Purchase"]
```

```
list1 = []
for j in yea:
    list1.append(j.strip(' ')[0:3])
print(list1)
```

```
'Apr', 'Jan', 'Jul', 'Apr', 'Dec', 'May', 'Apr', 'May', 'May', 'Aug', 'Nov', 'May', 'Feb', 'Mar', 'Oct', 'Feb', 'Mar', 'Jun',
'Mar', 'Oct', 'Jan', 'Apr', 'Nov', 'Jan', 'Aug', 'Jan', 'Nov', 'Sep', 'Oct', 'Jul', 'Dec', 'Nov', 'Jul', 'May', 'Jan', 'Nov',
'Sep', 'Mar', 'Feb', 'Apr', 'May', 'May', 'Mar', 'Aug', 'Apr', 'Apr', 'Mar', 'Oct', 'Mar', 'Jan', 'Jan', 'Nov', 'Feb', 'Mar',
'Jun', 'Feb', 'Mar', 'Oct', 'Aug', 'Apr', 'Feb', 'Jan', 'Aug', 'Feb', 'Apr', 'Jan', 'Nov', 'Dec', 'Feb', 'Nov', 'May', 'May',
'Sep', 'Jul', 'Nov', 'Jul', 'Feb', 'Aug', 'Sep', 'Nov', 'Aug', 'Jan', 'Mar', 'Jan', 'May', 'Jan', 'Aug', 'Jan', 'Jun', 'Jan',
'Feb', 'Jun', 'Nov', 'Nov', 'Jul', 'Feb', 'Jan', 'Aug', 'Apr', 'Oct', 'Jun', 'Nov', 'Jul', 'Oct', 'Feb', 'Apr', 'Feb', 'Feb',
'Mar', 'Oct', 'Aug', 'Apr', 'Feb', 'Jan', 'Aug', 'Feb', 'Apr', 'Jan', 'Nov', 'Feb', 'Sep', 'Nov', 'Dec', 'Nov', 'Oct', 'Jun',
'Jan', 'Dec', 'Jun', 'May', 'Mar', 'Jan', 'May', 'Jan', 'Jan', 'Aug', 'Jan', 'May', 'Apr', 'Aug', 'Jan', 'Jan', 'Mar', 'Mar',
'Jul', 'Oct', 'Jun', 'Mar', 'Jan', 'Aug', 'Jul', 'Nov', 'Jan', 'Jul', 'Nov', 'Mar', 'Mar', 'Sep', 'Apr', 'Jul', 'May', 'Aug',
'Dec', 'Nov', 'May', 'Jul', 'Apr', 'Oct', 'Dec', 'Sep', 'May', 'Apr', 'Feb', 'Oct', 'Nov', 'Jun', 'Dec', 'May', 'Jan', 'Jan',
'Aug', 'Jan', 'Mar', 'Mar', 'Jan', 'Jan', 'Apr', 'Aug', 'Aug', 'Aug', 'Jul', 'Feb', 'Jan', 'Feb', 'Aug', 'Feb', 'Mar', 'Jul',
'Oct', 'Apr', 'Jan', 'Mar', 'Mar', 'Nov', 'Jul', 'Feb', 'Apr', 'Jan', 'Mar', 'Mar', 'Jan', 'Jan', 'Apr', 'Jan', 'Mar', 'Mar',
'Jan', 'Sep', 'Apr', 'Apr', 'Jan', 'Mar', 'Jan', 'Mar', 'Sep', 'Apr', 'Jun', 'Nov', 'Jan', 'Feb', 'May', 'Oct', 'Aug', 'Apr',
'Apr', 'Jan', 'Jan', 'Jul', 'Nov', 'Feb', 'Feb', 'Oct', 'Jan', 'Feb', 'Jan', 'Jun', 'Apr', 'Apr', 'Jan', 'Jan', 'Feb', 'May',
'Oct', 'Oct', 'Jul', 'Jan', 'Jan', 'Aug', 'Jan', 'Aug', 'Feb', 'Jan', 'Nov', 'Feb', 'May', 'Feb', 'Mar', 'Mar', 'Mar', 'Nov',
'Aug', 'Oct', 'May', 'Apr', 'Jan', 'Aug', 'Nov', 'Dec', 'Jan', 'Dec', 'Sep', 'May', 'Feb', 'Sep', 'Jan', 'Oct', 'Aug', 'Apr',
'Jul', 'Mar', 'Jun', 'Jan', 'May', 'Mar', 'Aug', 'Dec', 'Feb', 'Mar', 'Aug', 'Jan', 'Jun', 'Jul', 'May', 'Mar', 'Sep', 'May',
'Jan', 'Oct', 'Mar', 'May', 'Jun', 'Dec', 'Apr', 'Mar', 'Oct', 'Jan', 'Nov', 'Jun', 'Sep', 'Feb', 'May', 'Aug', 'Apr', 'Jun',
'Mar', 'Jun', 'Mar', 'Jun', 'Jan', 'Apr', 'Mar', 'Sep', 'Mar', 'Feb', 'Feb', 'Oct', 'Dec', 'Oct', 'Dec', 'Nov', 'Jan', 'Apr',
```

```
Purchase_month = list1
```

```
df1 = pd.DataFrame()
df1["Purchase_month"] = list1
df1
```

	Purchase_month
0	Mar
1	Jul
2	Jan
3	Nov
4	Mar
...	...
7034	Feb
7035	Jan
7036	Sep
7037	Aug
7038	Feb

```
list2 = []
for k in yea:
    list2.append(k.strip('')[-2:])
print(list2)
```

```
'14', '10', '14', '17', '18', '13', '14', '15', '14', '17', '14', '15', '15', '15', '17', '17', '15', '14', '17', '15', '13',
'13', '13', '20', '13', '13', '14', '15', '18', '15', '15', '14', '14', '16', '18', '17', '16', '15', '13', '18', '17', '15',
'15', '17', '15', '14', '14', '17', '14', '13', '17', '13', '15', '17', '13', '15', '15', '14', '15', '15', '13', '13', '17',
'17', '15', '17', '14', '15', '14', '14', '15', '18', '18', '15', '17', '10', '14', '14', '19', '13', '13', '15', '16', '14',
'15', '19', '13', '12', '13', '17', '17', '17', '14', '13', '18', '14', '16', '12', '14', '17', '14', '13', '17', '13', '15',
'17', '13', '15', '15', '14', '15', '13', '12', '15', '17', '18', '17', '18', '18', '18', '18', '13', '19', '16', '15', '14',
'12', '14', '13', '13', '15', '20', '17', '13', '16', '13', '13', '14', '19', '15', '14', '19', '13', '13', '13', '14', '18',
'16', '15', '17', '13', '15', '18', '13', '13', '13', '20', '15', '17', '14', '19', '13', '15', '20', '14', '13', '17', '15',
'17', '20', '18', '20', '14', '18', '15', '14', '20', '17', '14', '19', '15', '18', '18', '12', '19', '17', '14', '16', '15',
'17', '16', '15', '18', '13', '12', '20', '19', '16', '14', '18', '15', '14', '20', '17', '14', '19', '18', '19', '15', '19',
'16', '19', '15', '19', '16', '14', '16', '19', '16', '17', '20', '13', '17', '16', '18', '18', '13', '16', '16', '09', '18',
'18', '16', '13', '11', '21', '19', '18', '18', '13', '17', '20', '13', '17', '16', '16', '16', '17', '16', '21', '18', '11',
'13', '09', '18', '18', '18', '18', '16', '18', '12', '15', '14', '13', '14', '13', '14', '16', '17', '19', '17', '18', '12',
'16', '18', '14', '17', '16', '17', '16', '14', '16', '17', '18', '12', '20', '12', '19', '15', '17', '17', '15', '14', '09',
'13', '17', '13', '18', '20', '18', '13', '13', '11', '10', '17', '18', '20', '20', '18', '14', '13', '18', '18', '14', '19',
'17', '19', '17', '13', '18', '09', '10', '19', '18', '17', '14', '19', '16', '16', '16', '18', '16', '12', '17', '18', '15',
'17', '18', '18', '18', '19', '16', '19', '19', '17', '18', '16', '16', '16', '19', '15', '16', '16', '12', '16', '18', '15',
'18', '16', '18', '18', '15', '14', '17', '15', '17', '14', '19', '17', '18', '17', '18', '16', '16', '16', '19', '15', '16',
'16', '12', '16', '18', '15', '18', '16', '16', '16', '16', '18', '19', '14', '17', '13', '17', '12', '18', '14', '17', '17',
```

```
purchase_year = list2
```

```
df2 = pd.DataFrame()
df2["purchase_year"] = list2
df2
```

	purchase_year
0	15
1	14
2	16
3	15
4	18
...	...
7034	19
7035	19
7036	20
7037	17
7038	18

```
data.insert(len(data.columns),"Purchase_month",Purchase_month.values)
```

```
data.insert(len(data.columns),"Purchase_year",Purchase_year.values)
```

Observation : Here we can see that we are inserting the created columns into the dataframe.

```
data.shape
```

```
(7039, 16)
```

```
data.drop(['Year of Purchase'],axis = 1,inplace = True)
```

Observation : Here we can see that we have dropped the parent column "Year of Purchase".

- Here the new columns like "Purchase month" and "purchase year" are created and after creating these new columns the parent columns are dropped.

Kilometers Driven :

```
kms = data["Kilometers Driven"]
```

```
list3 = []
for l in kms:
    list3.append(l.split(" ")[0].replace(",",""))
print(list3)
```

```
['76264', '92088', '67332', '76135', '37786', '74447', '70112', '68749', '121025', '81877', '18164', '27611', '80392', '7726
3', '76530', '82367', '83937', '44193', '94820', '8235', '52121', '14734', '71382', '77787', '41996', '48259', '9610', '7740
9', '54884', '30528', '39011', '36696', '72310', '46856', '112006', '57394', '97591', '41963', '56552', '49216', '76551', '76
608', '57256', '66396', '13235', '11026', '40587', '54588', '29449', '14221', '92086', '62781', '23091', '45750', '43064', '2
691', '38216', '107206', '65561', '27651', '52121', '14734', '71382', '77787', '41996', '48259', '9610', '77409', '54884', '3
0528', '39011', '36696', '72310', '46856', '112006', '57394', '97591', '41963', '56552', '49216', '31885', '33242', '122416',
'119146', '23673', '3219', '24652', '114506', '38070', '104093', '29026', '37925', '46168', '64013', '64127', '58818', '6713
9', '16930', '72334', '61499', '77409', '9610', '97591', '46856', '57394', '48259', '30528', '39011', '54884', '36696', '7231
0', '43732', '35148', '47329', '69806', '41963', '70638', '112006', '56552', '49216', '37205', '86430', '58990', '83143', '96
866', '91724', '76854', '55763', '73240', '104005', '38015', '42295', '68262', '34025', '56149', '66594', '38019', '33250',
'69579', '35605', '17419', '67003', '65080', '56789', '30512', '35607', '44638', '31370', '40547', '48080', '70981', '61778',
'34349', '17474', '93428', '112745', '25937', '76156', '42119', '70990', '54884', '36696', '72310', '43732', '35148', '4732
9', '69806', '41963', '70638', '112006', '56552', '49216', '83212', '55532', '120326', '18206', '45688', '68793', '35282', '3
1671', '8695', '43181', '27816', '167528', '19831', '49147', '84904', '79144', '70677', '78595', '99562', '78932', '122972',
'25954', '59162', '27428', '28614', '82843', '45868', '70124', '27398', '36469', '93340', '11311', '47530', '114506', '12006
0', '38070', '29026', '58818', '104093', '16930', '122416', '52870', '67139', '119146', '23673', '24652', '3219', '46168', '4
0587', '107206', '14221', '65561', '27651', '8235', '76551', '76608', '57256', '11026', '66396', '13235', '15498', '6637', '8
7334', '27027', '24515', '76013', '15282', '83988', '56467', '22096', '76039', '32860', '41056', '55409', '75243', '81726',
'40643', '70746', '25636', '54041', '117629', '51434', '46829', '40212', '21765', '15934', '44280', '72378', '87334', '2702
```

```
Kilometers_Driven = list3
```

```
df3 = pd.DataFrame()
df3["Kilometers_Driven"] = list3
df3
```

	Kilometers_Driven
0	76264
1	92088
2	67332
3	76135
4	37786
...	...
7034	16539
7035	31962
7036	15472
7037	6427
7038	35407

```
data.insert(len(data.columns),"Kilometers_Driven",Kilometers_Driven.values)
```

Observation : Here we have inserted the created column into the dataframe.

```
data.drop(['Kilometers Driven'],axis = 1,inplace = True)
```

Observation : Here we have dropped the parent column "Kilometers Driven" .

- Here the column "Kilometres Driven" is deleted and the column with the same name but different extracted data is created.

```
service= data["Last Service"].str.split( '(' )
```

```
list4=[]
for m in range(len(service)):
    print(service[m][1][:-1])
    list4 += [service[m][1][:-1]]
print(list4)
```

```
18 Feb 2022
14 Mar 2022
23 Feb 2022
11 Mar 2022
07 Feb 2022
```

```
service = df4
service
```

	service
0	22 Nov 2021
1	08 Mar 2022
2	16 Nov 2021
3	07 Mar 2022
4	09 Feb 2022
...	...
7034	24 Dec 2021
7035	01 Mar 2022
7036	21 Dec 2021
7037	21 Dec 2021
7038	11 Jan 2022

```
data.insert(len(data.columns),"service",service.values)
```

Observation : Here we have inserted the column into the dataframe.

```
data.drop(['Last Service'],axis = 1,inplace = True)
```

Observation : Here we can see that we can see that we have dropped the parent column "Last Service".

Owner :

```
own = data["Owner"]
```

```
list5 = []
for n in own:
    list5.append(n.strip(" ")[0])
print(list5)
```

```
['1', '1', '1', '2', '1', '1', '1', '1', '2', '1'
'2', '2', '1', '1', '1', '1', '1', '1', '1', '1',
'1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
'1', '1', '1', '1', '2', '3', '1', '1', '3', '1',
```

```
owner = list5
```

```
df4 = pd.DataFrame()
df4["owner"] = list5
df4
```

	owner
0	1
1	1
2	1
3	2
4	1
...	...
7034	2
7035	1
7036	1
7037	1
7038	1

```
data.insert(len(data.columns),"owner",owner.values)
```

Observation : Here we have inserted the created column into the dataframe.

```
data.drop(['Owner'],axis = 1,inplace = True)
```

Observation : Here we have dropped the parent column "Owner".

- Here we have created the new column named “owner” and the parent column is deleted.

Insurance :

```python
ins = data["Insurance"]
```

```python
list5 = []   ## insurance year
for p in ins:
    list5.append(p.split(" ")[2:4][1])
print(list5)
```

```
['2023', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '2022', '202
3', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '2023', '2022', '2023',
'2023', '2023', '2023', '2023', '2022', '2023', '2022', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '202
2', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2022', '2023',
'2023', '2023', '2023', '2022', '2023', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '202
2', '2023', '2023', '2022', '2022', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023',
'2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2022', '2022', '2022', '2023', '202
3', '2023', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '2023',
'2023', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '202
2', '2022', '2022', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2022', '2023', '2023',
'2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '2023', '202
3', '2023', '2023', '2023', '2023', '2023', '2022', '2023', '2022', '2023', '2023', '2022', '2023', '2022', '2023', '2022',
'2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '202
3', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2022',
'2023', '2023', '2023', '2022', '2023', '2022', '2023', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2023', '202
3', '2022', '2023', '2023', '2022', '2023', '2023', '2023', '2023', '2022', '2023', '2022', '2023', '2023', '2023', '2023',
'2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2022', '202
3', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2022', '2023',
'2022', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '2023', '202
```

```python
Year_of_insurance = list5
```

```python
df5 = pd.DataFrame()
df5["Year_of_insurance"] = list5
df5
```

	Year_of_insurance
0	2023
1	2023
2	2022
3	2023
4	2023
...	...
7034	2023
7035	2023
7036	2023
7037	2023
7038	2022

```python
data.insert(len(data.columns),"Year_of_insurance",Year_of_insurance.values)
```

```python
list6 = []
for q in ins:
    list6.append(q.split(" ")[2:4][0])
print(list6)
```

```
['Mar', 'Mar', 'Jul', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Sept', 'Mar', 'Sept', 'Mar', 'Mar', 'Ma
r', 'Mar', 'Mar', 'Aug', 'Mar', 'Mar', 'Mar', 'Mar', 'Sept', 'Mar', 'Mar', 'Mar', 'Jul', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar',
'Dec', 'Mar', 'Nov', 'Mar', 'Mar', 'Dec', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Nov', 'Mar', 'Mar', 'Dec', 'Mar', 'Mar', 'Mar',
'Mar', 'Mar', 'Dec', 'Mar', 'Mar', 'Mar', 'Mar', 'Aug', 'Mar', 'Mar', 'Mar', 'Mar', 'Sept', 'Mar', 'Mar', 'Mar', 'Jul', 'Ma
r', 'Mar', 'Mar', 'Mar', 'Mar', 'Dec', 'Mar', 'Nov', 'Mar', 'Mar', 'Sept', 'Nov', 'Mar', 'Mar', 'Nov', 'Aug', 'Mar', 'Mar',
'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Dec', 'Sep
t', 'Jul', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Nov', 'Mar', 'Mar', 'Mar', 'Mar', 'Dec', 'Mar', 'Mar', 'M
ar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Aug', 'Mar', 'Mar', 'Nov', 'Jan', 'Mar', 'Feb', 'Mar', 'Mar', 'Aug', 'Jul',
'Aug', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Jul', 'Mar', 'Mar', 'Mar', 'Mar', 'Nov', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar',
'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Nov', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Nov',
'Mar', 'Sept', 'Mar', 'Mar', 'Jul', 'Mar', 'Jun', 'Mar', 'Jun', 'Mar', 'Mar', 'Jan', 'Mar', 'Mar', 'Mar', 'Mar', 'Jan', 'Fe
b', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Aug', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'M
ar', 'Mar', 'Mar', 'Mar', 'Nov', 'Mar', 'Aug', 'Mar', 'Nov', 'Mar', 'Dec', 'Mar', 'Mar', 'Mar', 'Dec', 'Mar', 'Mar', 'Mar',
'Mar', 'Mar', 'Mar', 'Oct', 'Mar', 'Mar', 'Jul', 'Mar', 'Jan', 'Mar', 'Mar', 'Oct', 'Mar', 'Aug', 'Mar', 'Mar', 'Mar', 'Mar',
'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Jul', 'Mar', 'Jan', 'Mar',
'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Dec', 'Mar', 'Jul', 'Mar', 'Mar', 'Mar', 'Mar',
'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Oct', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar',
'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Oct', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Dec', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar',
'Mar', 'Mar', 'Mar', 'Nov', 'Aug', 'Nov', 'Jan', 'Mar', 'Mar', 'Mar', 'Dec', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar', 'Mar',
```

```
Month_of_insurance = list6
```

```
df6 = pd.DataFrame()
df6["Month_of_insurance"] = list6
df6
```

	Month_of_insurance
0	Mar
1	Mar
2	Jul
3	Mar
4	Mar
...	...
7034	Mar
7035	Mar
7036	Mar
7037	Mar
7038	Oct

```
data.insert(len(data.columns),"Month_of_insurance",Month_of_insurance.values)
```

```
data.drop(['Insurance'],axis = 1,inplace = True)
```

- Here we have dropped the parent column after creating the new columns "Year of insurance" and "Month of insurance".

EMI per month :

```
em = data["EMI per month"]
```

```
list8 = []   #emi
for s in em:
    list8.append(s.split("₹")[1].split("/")[0].replace(",",""))
print(list8)
```

```
['9840', '9710', '13612', '12672', '14546', '7843', '10746', '7902', '11889', '13765', '13783', '16418', '11718', '8922', '13
476', '8357', '8945', '13501', '11955', '9109', '13193', '10099', '8858', '5707', '8838', '12927', '14489', '9188', '9184',
'10718', '10024', '16106', '9799', '10318', '9937', '12797', '11723', '13410', '11017', '9487', '9792', '8548', '9903', '1325
7', '8492', '10434', '13626', '13462', '8564', '15584', '9259', '12317', '13995', '9862', '10095', '9279', '15054', '9063',
'7704', '11135', '13193', '10099', '8858', '5707', '8838', '12927', '14489', '9188', '9184', '10718', '10024', '16106', '979
9', '10318', '9937', '12797', '11723', '13410', '11017', '9487', '10550', '10372', '7401', '10812', '8560', '8763', '9191',
'7654', '9655', '10124', '11431', '9434', '9612', '11315', '9967', '9163', '13173', '16306', '12169', '10097', '9188', '1448
9', '11723', '10318', '12797', '12927', '10718', '10024', '9184', '16106', '9799', '8685', '12658', '8599', '10789', '13410',
'9746', '9937', '11017', '9487', '12401', '9466', '7626', '7770', '11454', '13287', '9418', '14785', '9967', '9919', '9047',
'7713', '12096', '13155', '14933', '12460', '14724', '6089', '9396', '12977', '11634', '8403', '9480', '10113', '10208', '987
6', '9976', '8152', '7802', '7840', '7638', '14678', '13517', '9903', '11520', '10602', '13382', '8792', '13057', '9637', '91
84', '16106', '9799', '8685', '12658', '8599', '10789', '13410', '9746', '9937', '11017', '9487', '10878', '6224', '6051', '9
184', '12695', '12570', '14063', '10910', '13578', '13471', '11386', '7742', '15334', '11486', '10033', '8029', '9207', '905
2', '8405', '7426', '15552', '15914', '8357', '13617', '7829', '8262', '6565', '12558', '10336', '12169', '16292', '8977', '7
654', '7351', '9655', '11431', '9163', '10124', '16306', '7401', '14380', '13173', '10812', '8560', '9191', '8763', '9612',
'13626', '9063', '15584', '7704', '11135', '9109', '9792', '8548', '9903', '10434', '13257', '8492', '7572', '10500', '9259',
'8498', '6634', '8095', '10500', '15111', '8095', '9409', '11383', '17900', '17577', '8494', '18064', '16506', '9630', '1481
7', '7050', '6568', '12563', '5793', '10980', '8785', '5418', '9949', '8751', '15864', '9259', '8498', '6634', '8095', '1050
0', '15111', '9610', '18417', '9409', '9033', '10985', '9220', '16169', '18417', '6770', '9890', '14382', '7704', '16980', '9
```

```
EMI = list8
```

```
df8 = pd.DataFrame()
df8["EMI"] = list8
df8
```

	EMI
0	9840
1	9710
2	13612
3	12672
4	14546
...	...
7034	0
7035	0
7036	0
7037	0
7038	0

```
data.insert(len(data.columns),"EMI",EMI.values)
```

Observation : here we have inserted the created column into the dataframe

```
data.drop(['EMI per month'],axis = 1,inplace = True)
```

Observation : Here we have seen that we have dropped the parent column.

- Here we have added the column into the dataframe and deleted the parent column.

Service :

```
ser = data["service"]
```

```
list9 = []   #service m
for t in ser:
    list9.append(t.split(" ")[1])
print(list9)
```

```
['Nov', 'Mar', 'Nov', 'Mar', 'Feb', 'Feb', 'Feb', 'Nov', 'Feb', 'Mar', 'Feb', 'Mar', 'Feb', 'Mar', 'Mar', 'Nov', 'Feb', 'Ma
r', 'Feb', 'Feb', 'Mar', 'Nov', 'Feb', 'Feb', 'Nov', 'Mar', 'Feb', 'Mar', 'Dec', 'Feb', 'Mar', 'Jan', 'Feb', 'Mar', 'Jan', 'M
ar', 'Mar', 'Jan', 'Jan', 'Jan', 'Feb', 'Jan', 'Mar', 'Nov', 'Feb', 'Jan', 'Feb', 'Feb', 'Feb', 'Feb', 'Feb', 'Mar', 'Feb',
'Feb', 'Feb', 'Feb', 'Feb', 'Dec', 'Feb', 'Feb', 'Mar', 'Nov', 'Feb', 'Feb', 'Nov', 'Mar', 'Feb', 'Mar', 'Dec', 'Feb', 'Mar',
'Jan', 'Feb', 'Mar', 'Jan', 'Mar', 'Mar', 'Jan', 'Jan', 'Jan', 'Mar', 'Jan', 'Nov', 'Mar', 'Jan', 'Feb', 'Jan', 'Jan', 'Jan',
'Mar', 'Feb', 'Jan', 'Feb', 'Feb', 'Jan', 'Jan', 'Feb', 'Feb', 'Feb', 'Jan', 'Mar', 'Feb', 'Mar', 'Mar', 'Mar', 'Mar', 'Feb',
'Mar', 'Dec', 'Jan', 'Feb', 'Jan', 'Mar', 'Jan', 'Feb', 'Jan', 'Feb', 'Jan', 'Jan', 'Jan', 'Feb', 'Feb', 'Oct', 'Dec', 'Jan',
'Dec', 'Feb', 'Feb', 'Feb', 'Feb', 'Dec', 'Dec', 'Feb', 'Feb', 'Feb', 'Mar', 'Mar', 'Feb', 'Feb', 'Feb', 'Feb', 'Feb', 'Jan',
'Mar', 'Mar', 'Feb', 'Jan', 'Mar', 'Jan', 'Feb', 'Nov', 'Jan', 'Jan', 'Mar', 'Feb', 'Feb', 'Jan', 'Feb', 'Mar', 'Feb', 'Dec',
'Jan', 'Feb', 'Jan', 'Mar', 'Jan', 'Feb', 'Jan', 'Feb', 'Jan', 'Jan', 'Jan', 'Mar', 'Feb', 'Nov', 'Jan', 'Jan', 'Feb', 'Jan',
'Feb', 'Feb', 'Feb', 'Jan', 'Feb', 'Jan', 'Feb', 'Jan', 'Jan', 'Mar', 'Nov', 'Feb', 'Dec', 'Jan', 'Mar', 'Mar', 'Oct', 'Mar',
'Mar', 'Jan', 'Feb', 'Feb', 'Feb', 'Jan', 'Feb', 'Jan', 'Jan', 'Dec', 'Jan', 'Feb', 'Jan', 'Mar', 'Feb', 'Nov', 'Dec', 'Feb',
'Mar', 'Jan', 'Jan', 'Feb', 'Feb', 'Feb', 'Dec', 'Feb', 'Feb', 'Feb', 'Feb', 'Feb', 'Jan', 'Mar', 'Jan', 'Nov', 'Feb', 'Mar',
'Jan', 'Mar', 'Jan', 'Mar', 'Jan', 'Mar', 'Jan', 'Dec', 'Mar', 'Feb', 'Jan', 'Jan', 'Feb', 'Jan', 'Feb', 'Feb', 'Feb', 'Feb',
'Feb', 'Nov', 'Mar', 'Jan', 'Jan', 'Mar', 'Jan', 'Feb', 'Jan', 'Mar', 'Jan', 'Mar', 'Jan', 'Mar', 'Jan', 'Jan', 'Jan', 'Feb',
'Jan', 'Feb', 'Nov', 'Nov', 'Jan', 'Feb', 'Feb', 'Feb', 'Feb', 'Feb', 'Mar', 'Feb', 'Feb', 'Mar', 'Oct', 'Aug', 'Nov', 'Feb',
'Mar', 'Mar', 'Dec', 'Feb', 'Feb', 'Feb', 'Jan', 'Oct', 'Mar', 'Jan', 'Jan', 'Mar', 'Feb', 'Mar', 'Mar', 'Dec', 'Mar', 'Oct',
'Aug', 'Nov', 'Mar', 'Feb', 'Feb', 'Mar', 'Feb', 'Mar', 'Feb', 'Jan', 'Jan', 'Feb', 'Jan', 'Feb', 'Oct', 'Mar', 'Jan', 'Dec',
'Feb', 'Mar', 'Jan', 'Feb', 'Feb', 'Jan', 'Feb', 'Mar', 'Oct', 'Oct', 'Feb', 'Mar', 'Feb', 'Jan', 'Mar', 'Feb', 'Jan', 'Mar',
```

```
service_month = list9
```

```
df9 = pd.DataFrame()
df9["service_month"] = list9
df9
```

	service_month
0	Nov
1	Mar
2	Nov
3	Mar
4	Feb
...	...
7034	Dec
7035	Mar
7036	Dec
7037	Dec
7038	Jan

```
data.insert(len(data.columns),"service_month",service_month.values)
```

Observation : Here we have inserted the column into the dataframe.

```
service_year = list10
```

```
df10 = pd.DataFrame()
df10["service_year"] = list10
df10
```

```
list10 = []
for t in ser:
    list10.append(t.split(" ")[-1])
print(list10)
```

```
['2021', '2022', '2021', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '202
1', '2022', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2021', '2022', '2022', '2022', '2021', '2022', '2022',
'2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '202
2', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2021',
'2022', '2022', '2021', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '202
2', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022',
'2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '202
1', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2021', '2021',
'2022', '2021', '2022', '2022', '2022', '2022', '2021', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '202
2', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022',
'2022', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '202
2', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022',
'2022', '2022', '2022', '2021', '2022', '2021', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '202
2', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2021', '2021', '2022', '2022', '2022',
'2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2021', '2022', '202
2', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022',
'2022', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '202
2', '2022', '2022', '2022', '2022', '2022', '2022', '2022', '2021', '2021', '2022', '2022', '2022', '2022', '2022', '2022',
'2022', '2022', '2022', '2022', '2021', '2021', '2021', '2022', '2022', '2022', '2021', '2022', '2022', '2022', '2022', '202
```

	service_year
0	2021
1	2022
2	2021
3	2022
4	2022
...	...
7034	2021
7035	2022
7036	2021
7037	2021
7038	2022

```
data.insert(len(data.columns),"service_year",service_year.values)
```

Observation : Here we have inserted the created column into the dataframe

```
data.drop(['service'],axis = 1,inplace = True)
```

Observation : here we can see that we have extracted 2 columns from the parent column "service" and now we have dropped the column.

- Here the new column is added and the parent column is deleted from the dataframe.

Price :

```
pric = data["Price"]
```

```
list11 = []   #service m
for u in pric:
    list11.append(u.replace("₹","").replace(",",""))
print(list11)
```

```
99', '478199', '410999', '424399', '369799', '429299', '576599', '367299', '452599', '592799', '585599', '370499', '678799',
'400999', '535299', '608999', '427499', '437699', '401899', '655499', '392399', '332699', '483399', '573799', '437899', '3833
99', '244999', '382499', '562099', '630699', '397899', '397699', '465099', '434599', '701699', '424699', '447499', '430799',
'556399', '509199', '583299', '478199', '410999', '457699', '449899', '319399', '469199', '370299', '379199', '397999', '3304
99', '418399', '438999', '496399', '408699', '416499', '491299', '432099', '396799', '572899', '710499', '528799', '437799',
'397899', '630699', '509199', '447499', '556399', '562099', '465099', '434599', '397699', '701699', '424699', '375799', '5502
99', '371999', '468199', '583299', '422399', '430799', '478199', '410999', '538999', '410099', '329299', '335599', '497399',
'577899', '407999', '643699', '432099', '429999', '391699', '333099', '525599', '572099', '650199', '541599', '640999', '2617
99', '406999', '564299', '505299', '363399', '410699', '438499', '442699', '428099', '432499', '352399', '336999', '338699',
'329799', '638999', '587999', '429299', '500299', '459999', '582099', '380499', '567799', '417599', '397699', '701699', '4246
99', '375799', '550299', '371999', '468199', '583299', '422399', '430799', '478199', '410999', '472099', '267699', '260099',
'397699', '551899', '546399', '611999', '473499', '590699', '585999', '494399', '334399', '667799', '498799', '434999', '3469
99', '398699', '391899', '363499', '320499', '677399', '693299', '361399', '592399', '338199', '357199', '282699', '545899',
'448299', '528799', '709899', '388599', '330499', '317199', '418399', '496399', '396799', '438999', '710499', '319399', '6258
99', '572899', '469199', '370299', '397999', '379199', '416499', '592799', '392399', '678799', '332699', '483399', '394399',
'424399', '369799', '429299', '452599', '576599', '367299', '326899', '455499', '400999', '367599', '285699', '349899', '4554
99', '657999', '349899', '407599', '494299', '780499', '766299', '367399', '787699', '719299', '417299', '645099', '303999',
'282799', '546099', '248799', '476599', '380199', '232299', '431299', '378699', '691099', '400999', '367599', '285699', '3498
99', '455499', '657999', '416399', '803199', '407599', '391099', '476799', '399299', '704499', '803199', '291699', '428699',
```

```
Car_Price = list11
```

```
df11 = pd.DataFrame()
df11["Car_Price"] = list11
df11
```

	Car_Price
0	426499
1	420799
2	592199
3	550899
4	633199
...	...
7034	365599
7035	638999
7036	774999
7037	631399
7038	533099

```
data.insert(len(data.columns),"Car_Price",Car_Price.values)
```

Observation : Here we have inserted the created column into dataframe

```
data.drop(['Price'],axis = 1,inplace = True)
```

Observation : Here we have seen that the parent column is deleted.

- Here I have dropped the parent column and inplace the extracted new column is replaced.

- Now here we drop the unnecessary column "History" in which we have the same data in all the records and is not useful for our prediction and so we delete the column.

```
data["History"].unique()
```

```
array(['Non-Accidental'], dtype=object)
```

```
data.drop(['History'],axis = 1,inplace = True)
```

Observation : Here we have seen that the column "History" is with single value "Non-Accidental" and is also of no effect in the model and so we can drop this column.

Filling the null-values :

```
data['Location'] = data['Location'].fillna(data['Location'].mode()[0])
```

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

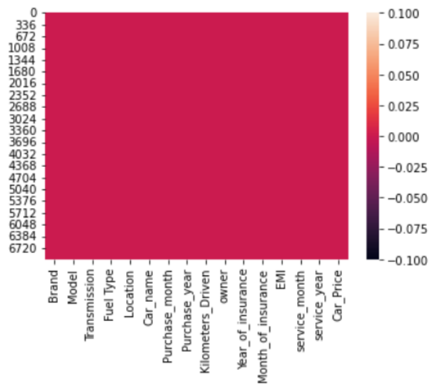
```
data.isnull().sum()
```

```
Brand                  0
Model                  0
Transmission           0
Fuel Type              0
Location               0
Car_name               0
Purchase_month         0
Purchase_year          0
Kilometers_Driven      0
owner                  0
Year_of_insurance      0
Month_of_insurance     0
EMI                    0
service_month          0
service_year           0
Car_Price              0
dtype: int64
```

➢ Here we have filled all the null values.

```
sns.heatmap(data.isnull())
```

<AxesSubplot:>



➢ Here we can see that there are no null values in our dataset now and thus we can continue for model building.

➢ Here we know that all our features are of the same object datatype even after extracting the required data from the parent columns and so here we are going to **change the datatype of the columns** where it is necessary.

Changing the datatypes :

```
data['Purchase_year'] = data['Purchase_year'].astype('int')
```

```
data['Purchase_year'].dtype
```

```
dtype('int32')
```

Observation : Here we have converted the created column "Purchase_year" into "int" datatype.

```
data['Kilometers_Driven'] = data['Kilometers_Driven'].astype('int')
```

```
data['Kilometers_Driven'].dtype
```

```
dtype('int32')
```

Observation : Here we have converted the created column "Kilometers_Driven" into "int" datatype.

```
data['owner'] = data['owner'].astype('int')
```

```
data['owner'].dtype
```

```
dtype('int32')
```

Observation : Here we have converted the created column "owner" into "int" datatype.

```
data['Year_of_insurance'] = data['Year_of_insurance'].astype('int')
```

```
data['Year_of_insurance'].dtype
```

```
dtype('int32')
```

Observation : Here we have converted the created column "Year_of_insurance" into "int" datatype.

```
data['EMI'] = data['EMI'].astype('int')
```

```
data['EMI'].dtype
```

```
dtype('int32')
```

Observation : Here we have converted the created column "EMI" into "int" datatype.

```
data['service_year'] = data['service_year'].astype('int')
```

```
data['service_year'].dtype
```

```
dtype('int32')
```

Observation : Here we have converted the created column "service_year" into "int" datatype.

```
data['Car_Price'] = data['Car_Price'].astype('int')
```

```
data['Car_Price'].dtype
```

```
dtype('int32')
```

Observation : Here we have converted the created column "Car_Price" into "int" datatype

Checking the datatypes of the columns again:

```
data.dtypes
```

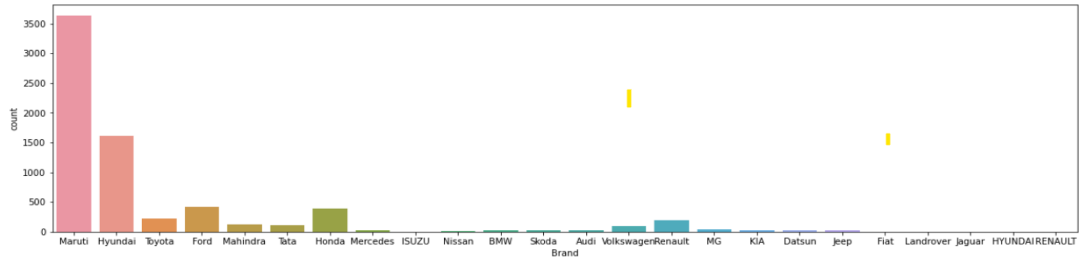
```
Brand                  object
Model                  object
Transmission           object
Fuel Type              object
Location               object
Car_name               object
Purchase_month         object
Purchase_year           int32
Kilometers_Driven       int32
owner                   int32
Year_of_insurance       int32
Month_of_insurance     object
EMI                     int32
service_month          object
service_year            int32
Car_Price               int32
dtype: object
```

Observation : Here we can see that the columsn have changed their datatypes.

- Now here we will move to **visualization** part where we will be going to visualize the features of our dataset and analyse them.

Brand :

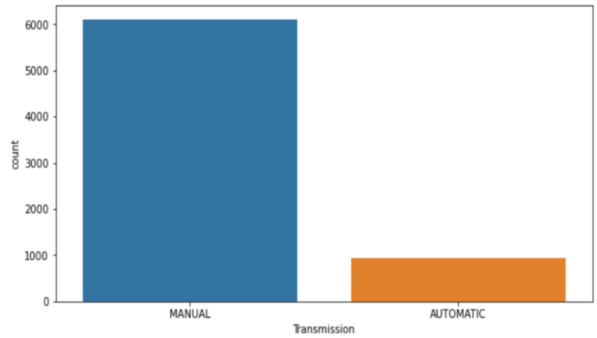
```
# Brand
plt.figure(figsize=(20,5))
sns.countplot(data.Brand);
```



Observation : Here we have seen that the column has the highest count for the attribute "Maruti" followed by "Hyundai" and the least count is for "Nissan".

Transmission :

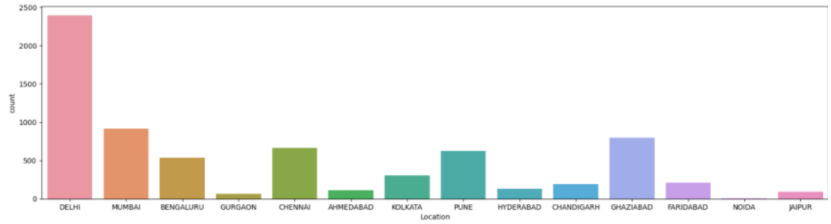
```
# Transmission
plt.figure(figsize=(10,5))
sns.countplot(data.Transmission);
```



Observation : here we can see that the column has the highest count for the attribute "Manual" and the least count is for "Automatic".

Location :

```
plt.figure(figsize=(20,5),dpi = 100)
sns.countplot(data.Location);
```

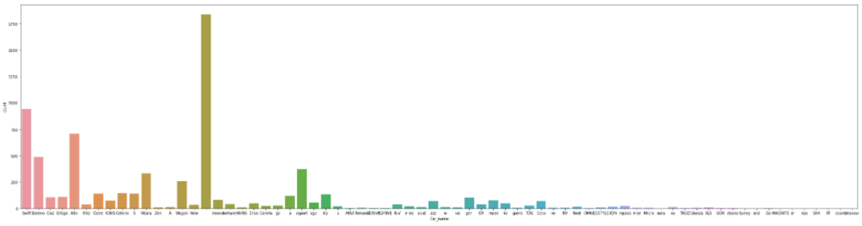


Observation : here we can see that the highest count is for the attribute "Delhi" followed by "Ghaziabad" and the least count is for the category "Noida".

Car_name :

```
Car_name

plt.figure(figsize=(40,10))
sns.countplot(data.Car_name);
```

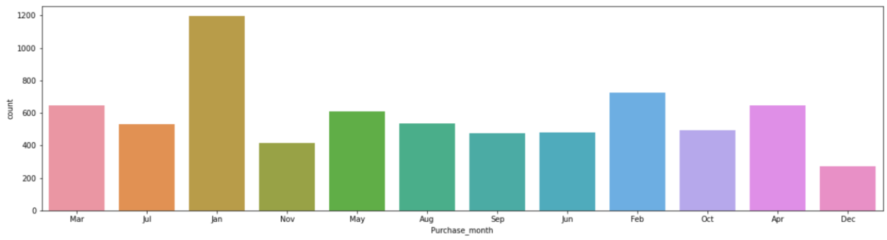


Observation : here we can see that the highest count is for the attribute "Innova" followed by "Swift" and the least count is for thye attribute "Aris".

Purchase_month :

```
Purchase_month

plt.figure(figsize=(20,5))
sns.countplot(data.Purchase_month);
```



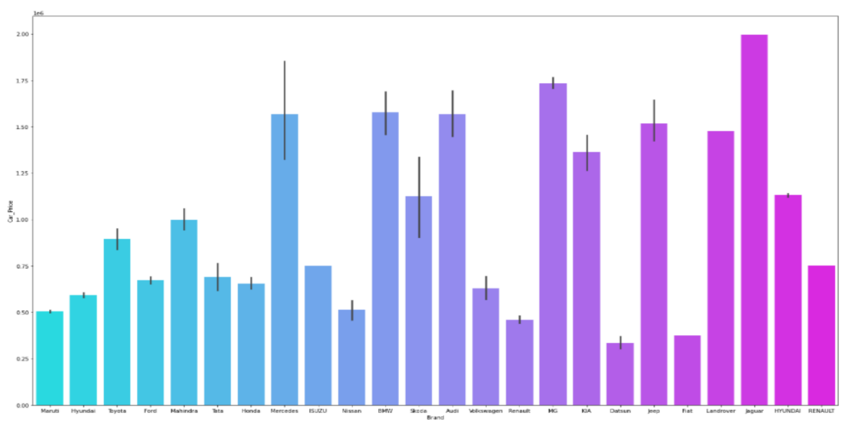
Observation : here we can see that the attribute with the highest count in the column is "Jan" followed by "Feb" and the least count is for the attribute "Dec".

- ➢ These are few features for which we have done **Univariate analysis**, now let's move to **Bivariate analysis** of the features of the data.

Brand with Car_price :

```
plt.figure(figsize = (25,15))
sns.barplot(x = 'Brand', y = 'Car_Price', data = data, palette = 'cool')

<AxesSubplot:xlabel='Brand', ylabel='Car_Price'>
```

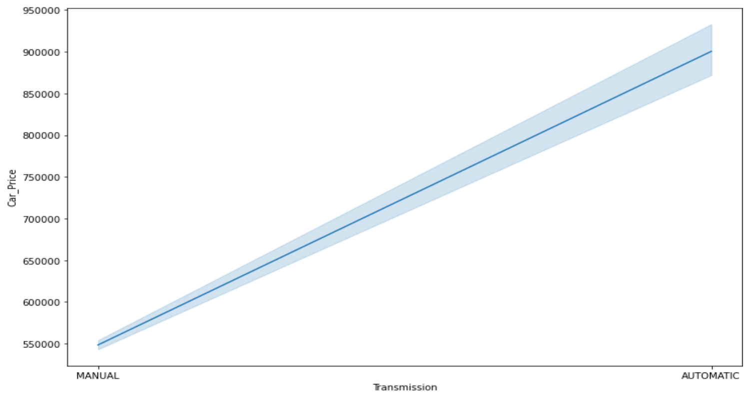


Observation : here we can see that the brand with highest price is "Jaguar" and the least price is "Datsun".

Transmission with car_price :

```
plt.figure(figsize = (12,8))
sns.lineplot(x = 'Transmission', y = 'Car_Price', data = data, palette = 'Blues')
```

```
<AxesSubplot:xlabel='Transmission', ylabel='Car_Price'>
```



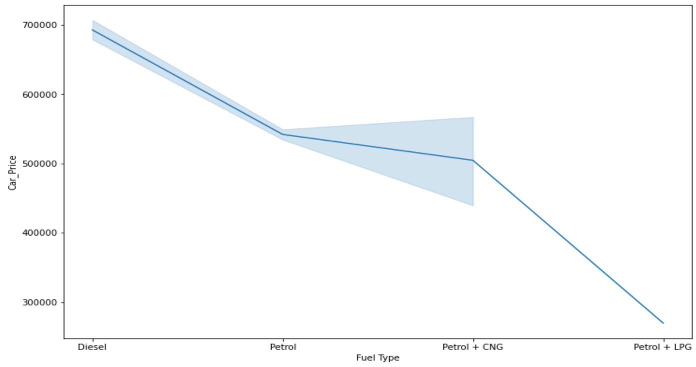
Observation : Here we can see that the highest price is seen for the Automatic category of the column " Transmission".

Fuel Type with car_price :

```
#Fuel Type

plt.figure(figsize = (12,8))
sns.lineplot(x = 'Fuel Type', y = 'Car_Price', data = data, palette = 'Blues')
```

```
<AxesSubplot:xlabel='Fuel Type', ylabel='Car_Price'>
```



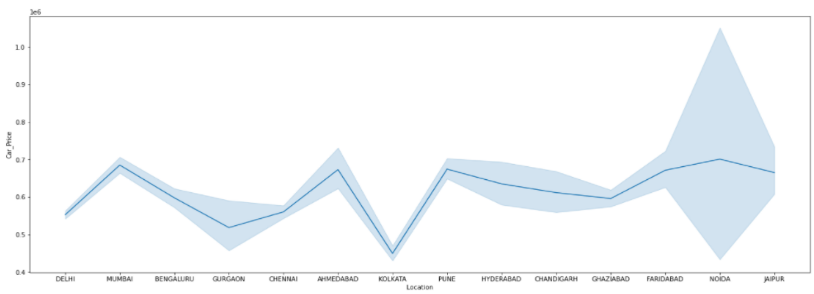
Observation : here we can see that the highest price is for the category "Diesel" and the least price is for the category "Petrol + LPG"

Location with car_price :

```
#Location

plt.figure(figsize = (22,8))
sns.lineplot(x = 'Location', y = 'Car_Price', data = data, palette = 'Blues')
```

```
<AxesSubplot:xlabel='Location', ylabel='Car_Price'>
```



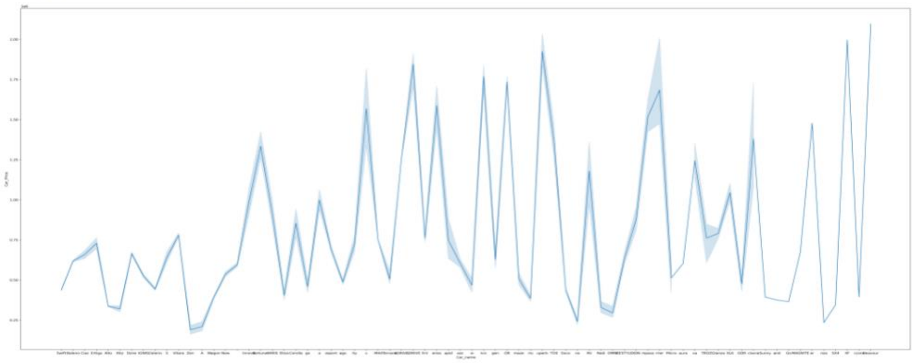
Observation : here we can see that all the locations the car_price is almost similar .

Car_name with car_price :

```
# Car_name

plt.figure(figsize = (42,20),dpi = 500)
sns.lineplot(x = 'Car_name', y = 'Car_Price', data = data, palette = 'Blues')
```

<AxesSubplot:xlabel='Car_name', ylabel='Car_Price'>



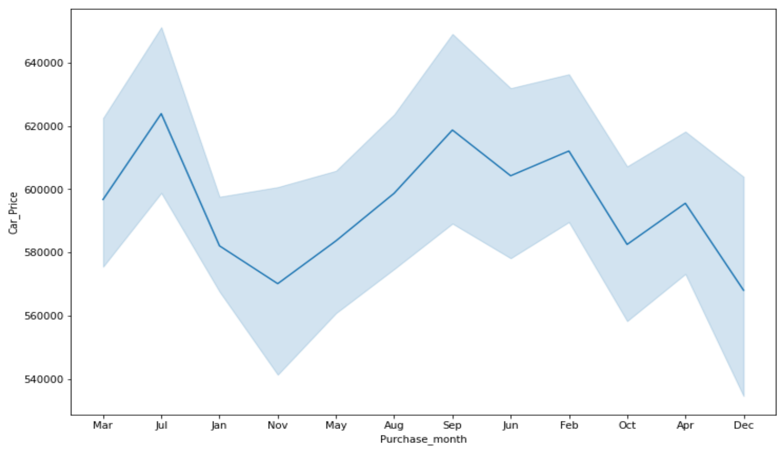
Observation : Here we can see that the highest car_price is for the category XF,Cardeavour and most of them have similar price ranges.

Purchase_month with car_price :

```
# Purchase_month

plt.figure(figsize = (12,8))
sns.lineplot(x = 'Purchase_month', y = 'Car_Price', data = data, palette = 'Blues')
```

<AxesSubplot:xlabel='Purchase_month', ylabel='Car_Price'>

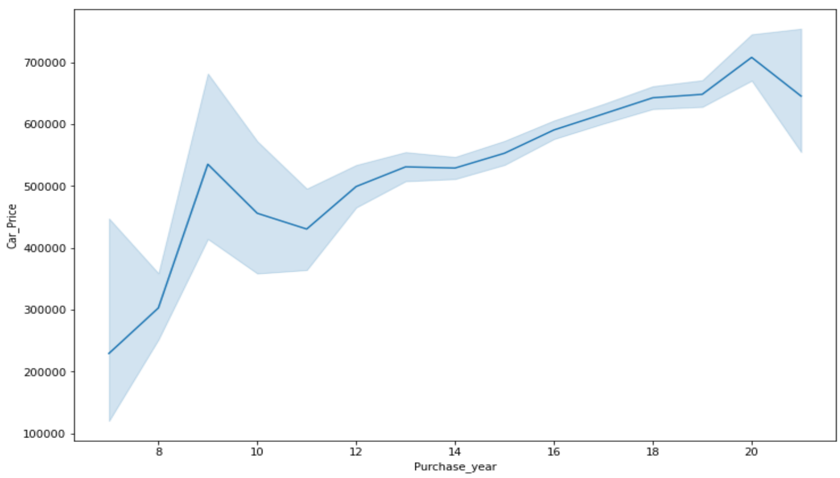


Observation : here we can see that the column has the highest price for the category "jul" with 620000 and the least is for the category "Dec".

Purchase_year with Car_Price :

```
plt.figure(figsize = (12,8))
sns.lineplot(x = 'Purchase_year', y = 'Car_Price', data = data, palette = 'Blues')
```

<AxesSubplot:xlabel='Purchase_year', ylabel='Car_Price'>

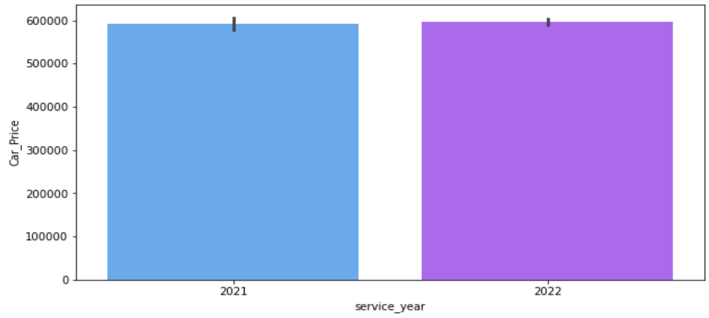


Observation ; here we can see that the column has the highest price for the category "20" and the least price for the category below "8".

service_year with Car_Price :

```
plt.figure(figsize = (10,5))
sns.barplot(x = 'service_year', y = 'Car_Price', data = data, palette = 'cool')
```

```
<AxesSubplot:xlabel='service_year', ylabel='Car_Price'>
```

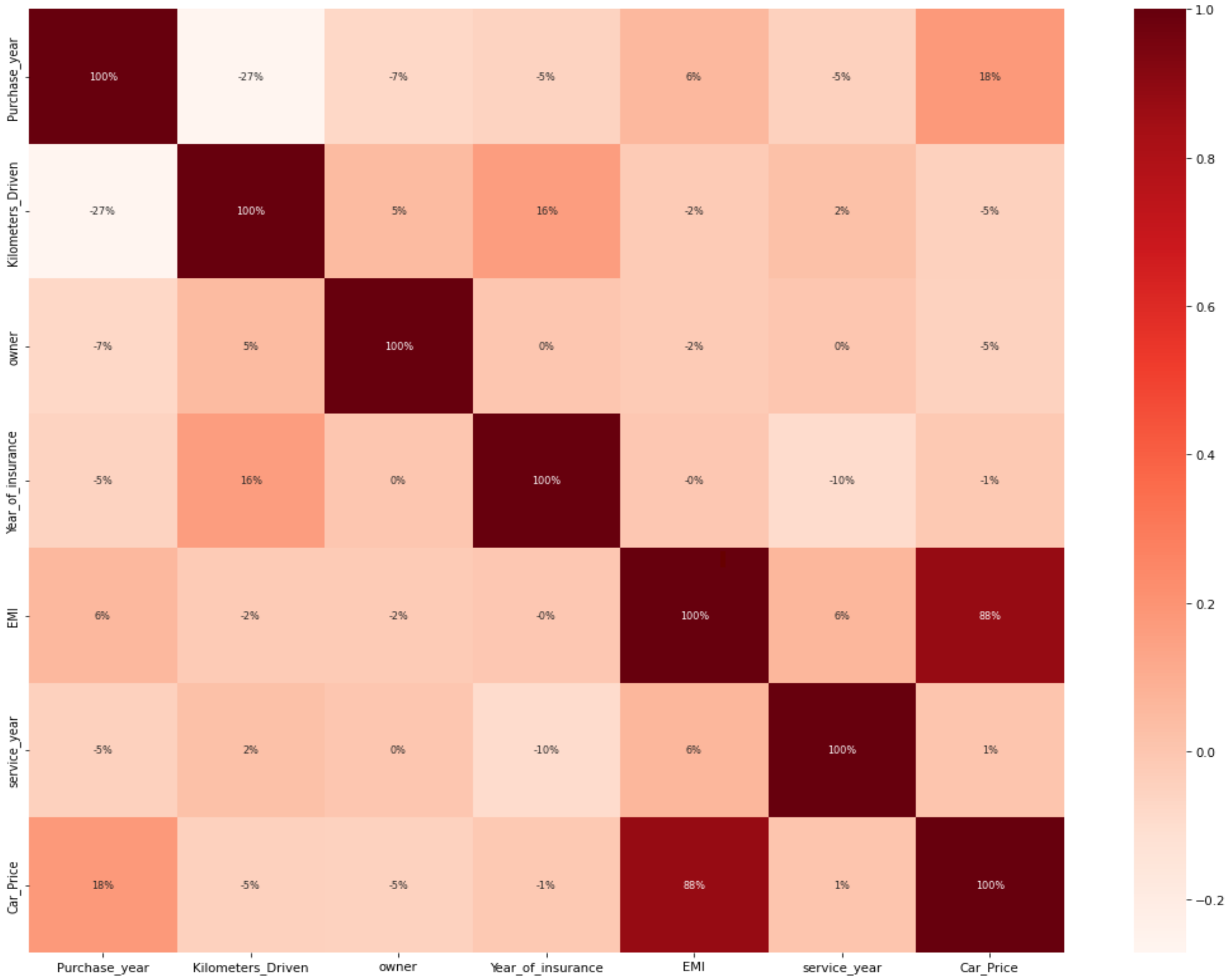


Observation : Here we can see that the categories present in the column are almost at the same price range almost at 600000

Correlation :

```
corr = data.corr()
```

```
plt.figure(figsize = (20,15))
sns.heatmap(corr,annot = True,fmt = ".0%",cbar = True,square = True,annot_kws = {'size':8}, cmap = 'Reds')
plt.show()
```



➢ Encoding the data through **Label Encoder:**

```
from sklearn.preprocessing import LabelEncoder
```

Observation : Here we have imported the "LabelEncoder" for encoding the data.

```
for column in data.columns:
    if data[column].dtype == np.number:
        continue
    data[column] = LabelEncoder().fit_transform(data[column])
```

Here we have used for loop to encode the complete data ie., all the column of the data.

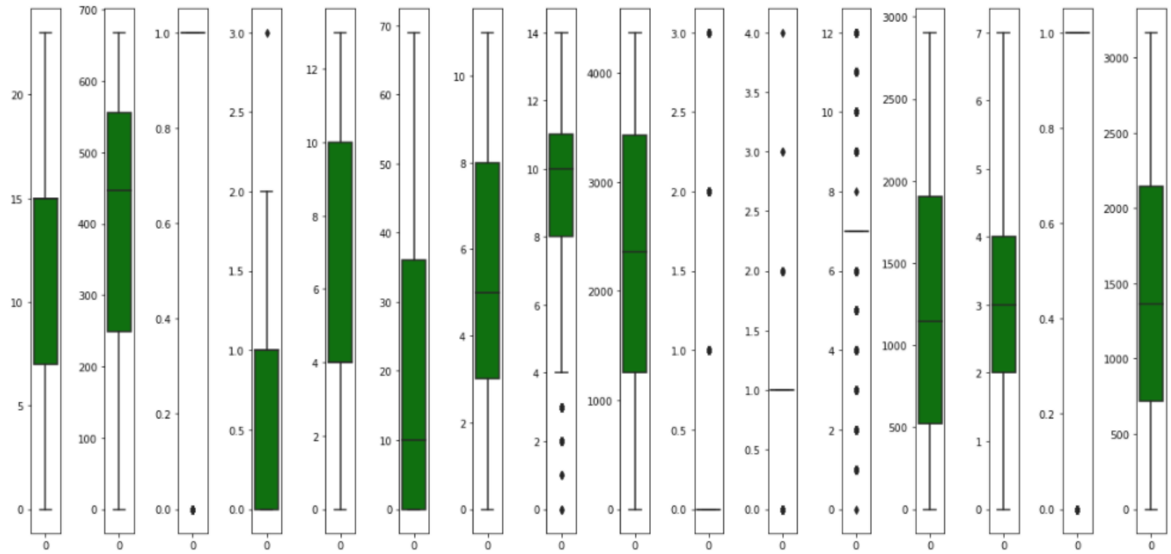
```
data.head()
```

	Brand	Model	Transmission	Fuel Type	Location	Car_name	Purchase_month	Purchase_year	Kilometers_Driven	owner	Year_of_insurance	Month_of_insurance
0	15	528	1	0	4	32	7	8	3712	0	1	7
1	15	626	1	0	4	32	5	7	4006	0	1	7
2	15	227	1	0	4	4	4	9	3463	0	0	5
3	15	227	1	0	4	4	9	8	3706	1	1	7
4	15	633	1	1	4	4	7	11	2122	0	1	7

Observation : Here we can see that we have encoded all the columns of the data.

➢ Now we will Check the **outliers** present in the data:

```
col_list = data.columns.values
ncol = 30
nrows = 12
plt.figure(figsize = (ncol,3*ncol))
for i in range (0, len(col_list)):
    plt.subplot(nrows,ncol,i+1)
    sns.boxplot(data = data[col_list[i]],color = 'green', orient = 'v')
    plt.tight_layout()
```



Observation : Here we can see that there are few columns which have outliers which are to be treated further for better model accuracy.

Removing the Outliers :

```
from scipy.stats import zscore
z = np.abs(zscore(data))
z.shape
```

```
(7039, 16)
```

```
threshold = 3
print(np.where(z>3))
```

```
(array([  47,   80,   83,  178,  215,  291,  315,  346,  355,  370,  382,
         533,  550,  550,  622,  623,  646,  735,  788,  819,  854,  854,
         855,  862,  905,  962, 1001, 1011, 1016, 1091, 1116, 1125, 1135,
        1198, 1199, 1210, 1212, 1300, 1319, 1324, 1379, 1420, 1430, 1468,
        1765, 1795, 1797, 1839, 1846, 1889, 1900, 1901, 1902, 1905, 1937,
        1944, 1945, 1962, 1964, 1965, 1974, 2008, 2009, 2022, 2055, 2066,
        2079, 2088, 2102, 2122, 2142, 2159, 2159, 2224, 2230, 2304, 2309,
        2327, 2413, 2641, 2716, 2825, 2832, 2847, 2865, 2870, 2899, 2903,
        2948, 2961, 2971, 2974, 3012, 3016, 3023, 3063, 3080, 3095, 3095,
        3177, 3234, 3264, 3271, 3311, 3318, 3333, 3333, 3355, 3363, 3406,
        3449, 3471, 3508, 3519, 3570, 3584, 3648, 3850, 4049, 4192, 4215,
        4219, 4261, 4264, 4269, 4273, 4283, 4325, 4341, 4361, 4393, 4397,
        4409, 4428, 4451, 4465, 4481, 4488, 4490, 4498, 4542, 4561, 4562,
        4605, 4657, 4666, 4692, 4725, 4727, 4735, 4786, 4838, 4839, 4914,
        4927, 4964, 4982, 4986, 4992, 5085, 5107, 5127, 5130, 5167, 5170,
        5213, 5253, 5278, 5287, 5297, 5360, 5370, 5371, 5382, 5383, 5479,
        5499, 5504, 5564, 5609, 5619, 5663, 5823, 5929, 5958, 5961, 6004,
        6065, 6074, 6084, 6143, 6152, 6153, 6164, 6165, 6253, 6271, 6276,
        6333, 6377, 6386, 6429, 6583, 6701, 6702, 6707, 6741, 6742, 6747,
        6752, 6757, 6759, 6824, 6862, 6896, 6904, 6907, 6932, 6948, 6962,
        7028], dtype=int64), array([ 9,  9,  9,  9,  9,  7,  7, 13,  7,  9,  7
         9,  9,  7,  7,  9,  9,  9,  9,  9,  9,  7,  9,  9,  9,  9,  9,  7,
         7,  9,  9,  9,  9,  9, 13,  9,  9,  7,  9,  9,  9,  7,  7,  9,  7,
         7,  9,  9,  9,  9,  7,  9,  7,  9,  7,  9, 13, 13,  9,  7,  9,  9,
         7,  9,  9,  7,  9, 13, 10,  9,  9,  7, 13,  9,  9,  9,  9, 13,  9,
        13,  9, 13,  9,  9,  9, 13,  9,  9,  9,  9,  9,  7,  9,  9, 13, 13,
        13,  9,  9,  9, 13,  9,  9,  9,  9,  9,  9,  9,  9,  9,  9, 11,  9,
         9,  9,  9,  9, 13,  9, 13,  9,  9,  9,  9, 13, 13,  9,  9,  9,  9,
         9,  9,  7,  9,  9,  9,  9,  9,  9, 13,  9, 13,  9,  9,  9,  7,  7,
         9,  9,  9,  9,  9,  9,  9,  9,  7,  9,  7,  9, 10,  9,  9,  9,  9,
         9,  7,  7,  9,  9,  9,  9,  9, 13,  9,  9,  7,  9,  9,  7,  9, 10,
         9,  9,  9,  9,  7,  7,  9,  9,  9,  9,  9, 13,  9,  9,  7,  9, 13,
         9, 13, 13,  9, 13, 13, 13, 13,  3, 13, 13, 13, 13, 13, 13,  9,  9],
```

```
data_new = data[(z<3).all(axis = 1)]
print(data.shape)
print(data_new.shape)
```

```
(7039, 16)
(6823, 16)
```

Observation : Here we can see that the data has reduced because the number of records got reduced from 7039 to 6823 that means we nhave been successful in treating the outliers.

```
data_new
```

	Brand	Model	Transmission	Fuel Type	Location	Car_name	Purchase_month	Purchase_year	Kilometers_Driven	owner	Year_of_insurance	Month_of_ins
0	15	528	0.386809	0	-0.399267	0.832017	7	8	3712	-0.477769	0.528474	0.
1	15	626	0.386809	0	-0.399267	0.832017	5	7	4006	-0.477769	0.528474	0.
2	15	227	0.386809	0	-0.399267	-0.473861	4	9	3463	-0.477769	-1.662682	-0.
3	15	227	0.386809	0	-0.399267	-0.473861	9	8	3706	2.093064	0.528474	0.
4	15	633	0.386809	1	-0.399267	-0.473861	7	11	2122	-0.477769	0.528474	0.
...	...	...	...	...	...	...	...	...	...	...	...	
7034	15	372	0.386809	1	-0.399267	-0.605666	3	12	747	2.093064	0.528474	0.
7035	7	33	0.386809	1	-0.399267	-1.334339	4	12	1768	-0.477769	0.528474	0.
7036	6	65	0.386809	1	-0.399267	1.327245	11	13	683	-0.477769	0.528474	0.
7037	7	380	0.386809	1	-0.399267	-1.334339	1	10	195	-0.477769	0.528474	0.
7038	7	493	0.386809	1	-0.399267	-1.334339	3	11	1995	-0.477769	-1.662682	1.

6823 rows × 16 columns

Checking the data loss :

```
data_loss = (7039-6823)/7039*100
```

```
data_loss
```

```
3.0686177013780367
```

Observation : Here we can see that the loss% is 3% which means that we have lost 3% of the data which is negligible.

Checking the Skewness :

```
data_new.skew()
```

```
Brand                -0.211438
Model                -0.517917
Transmission         -2.198933
Fuel Type            -0.488661
Location              0.627048
Car_name              0.690795
Purchase_month        0.116671
Purchase_year        -0.263006
Kilometers_Driven    -0.106703
owner                 1.615650
Year_of_insurance    -1.125474
Month_of_insurance   -0.722223
EMI                   0.233340
service_month         0.489505
service_year         -1.395127
Car_Price             0.225423
dtype: float64
```

Observation : Here we can see that we have columns which have the combination of positive skewness and negative skewness which have to be treated for better model accuracy.

Removing the skewness :

```
features = ["Transmission","service_year","Year_of_insurance","Month_of_insurance","Location","owner","Car_name"]
```

Observation : Here we have mentioned the features which have skewness and assigned them to a variable which is to be passed during tranforming method

```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
'''
parameters:
method = 'box-cox' or 'yeo-johnson'
'''
```

```
"\nparameters:\nmethod = 'box-cox' or 'yeo-johnson'\n"
```

```
data_new[features] = scaler.fit_transform(data_new[features].values)
data_new[features]
```

	Transmission	service_year	Year_of_insurance	Month_of_insurance	Location	owner	Car_name
0	0.386809	-1.916582	0.528474	0.060240	-0.399267	-0.477769	0.832017
1	0.386809	0.521762	0.528474	0.060240	-0.399267	-0.477769	0.832017
2	0.386809	-1.916582	-1.662682	-0.929743	-0.399267	-0.477769	-0.473861
3	0.386809	0.521762	0.528474	0.060240	-0.399267	2.093064	-0.473861
4	0.386809	0.521762	0.528474	0.060240	-0.399267	-0.477769	-0.473861
...	...	...	...	...	...	...	...
7034	0.386809	-1.916582	0.528474	0.060240	-0.399267	2.093064	-0.605666
7035	0.386809	0.521762	0.528474	0.060240	-0.399267	-0.477769	-1.334339
7036	0.386809	-1.916582	0.528474	0.060240	-0.399267	-0.477769	1.327245
7037	0.386809	-1.916582	0.528474	0.060240	-0.399267	-0.477769	-1.334339
7038	0.386809	0.521762	-1.662682	1.791497	-0.399267	-0.477769	-1.334339

6823 rows × 7 columns

```
data_new.skew()
```

```
Brand                -0.211438
Model                -0.517917
Transmission         -2.198933
Fuel Type            -0.488661
Location             -0.025383
Car_name             -0.118866
Purchase_month        0.116671
Purchase_year        -0.263006
Kilometers_Driven    -0.106703
owner                 1.615650
Year_of_insurance    -0.142321
Month_of_insurance    0.241816
EMI                   0.233340
service_month         0.489505
service_year         -1.395127
Car_Price             0.225423
dtype: float64
```

Observation : Here we can see that the skewness of the columns are treated and Ithe skewness values have changed ie., which are decreased

Separating the variables into independent and target variables :

```
x = data_new.drop("Car_Price", axis=1)
y = data_new["Car_Price"]
```

Here we have separated the data among the 2 variables x and y in which the label column is assigned with the variable "y" and the rest all the features are assigned to the variable "x".

```
x.head()
```

	Brand	Model	Transmission	Fuel Type	Location	Car_name	Purchase_month	Purchase_year	Kilometers_Driven	owner	Year_of_insurance	Month_of_insura
0	15	528	0.386809	0	-0.399267	0.832017	7	8	3712	-0.477769	0.528474	0.060
1	15	626	0.386809	0	-0.399267	0.832017	5	7	4006	-0.477769	0.528474	0.060
2	15	227	0.386809	0	-0.399267	-0.473861	4	9	3463	-0.477769	-1.662682	-0.929
3	15	227	0.386809	0	-0.399267	-0.473861	9	8	3706	2.093064	0.528474	0.060
4	15	633	0.386809	1	-0.399267	-0.473861	7	11	2122	-0.477769	0.528474	0.060

```
y.head()
```

```
0     850
1     814
2    1729
3    1542
4    1917
Name: Car_Price, dtype: int64
```

Here we use the Standard Scaler for Scaling the data present in the variable "x"

Scaling the data using the standard Scaler :

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
x = pd.DataFrame(scaler.fit_transform(x), columns=x.columns)
x
```

	Brand	Model	Transmission	Fuel Type	Location	Car_name	Purchase_month	Purchase_year	Kilometers_Driven	owner	Year_of_insurance	Mo
0	0.523717	0.712809	0.386809	-1.326706	-0.399267	0.832017	0.541421	-0.595678	1.127095	-0.477769	0.528474	
1	0.523717	1.249793	0.386809	-1.326706	-0.399267	0.832017	-0.069227	-1.047017	1.361512	-0.477769	0.528474	
2	0.523717	-0.936500	0.386809	-1.326706	-0.399267	-0.473861	-0.374551	-0.144339	0.928558	-0.477769	-1.662682	
3	0.523717	-0.936500	0.386809	-1.326706	-0.399267	-0.473861	1.152069	-0.595678	1.122311	2.093064	0.528474	
4	0.523717	1.288149	0.386809	0.722211	-0.399267	-0.473861	0.541421	0.758340	-0.140671	-0.477769	0.528474	
...	...	...	...	...	...	...	...	...	...	...	...	
6818	0.523717	-0.141983	0.386809	0.722211	-0.399267	-0.605666	-0.679875	1.209679	-1.237010	2.093064	0.528474	
6819	-1.067519	-1.999510	0.386809	0.722211	-0.399267	-1.334339	-0.374551	1.209679	-0.422929	-0.477769	0.528474	
6820	-1.266424	-1.824168	0.386809	0.722211	-0.399267	1.327245	1.762718	1.661018	-1.288039	-0.477769	0.528474	
6821	-1.067519	-0.098147	0.386809	0.722211	-0.399267	-1.334339	-1.290523	0.307001	-1.677140	-0.477769	0.528474	
6822	-1.067519	0.521029	0.386809	0.722211	-0.399267	-1.334339	-0.679875	0.758340	-0.241933	-0.477769	-1.662682	

6823 rows × 15 columns

Observation : Here we have scaled the data of our "x" ie., all our features are scaled.

Model Building :

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

Observation : Here we have imported the libraries required .

Checking the random_state:

```
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state=i)
    mod = LinearRegression()
    mod.fit(x_train, y_train)
    pred = mod.predict(x_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Maximum r2 score is ",maxAccu," on Random_state ",maxRS)
```

```
Maximum r2 score is  0.8569831256498824  on Random_state  7
```

Observation : Here we have checked the random state from 1 to 200 where we got the R2 Score as 85.6% at the random_state 7.

➢ Here we have imported the necessary libraries for modelbuilding and splitting the data into train and test sets and now we will split the data at the maximum random state which we got ie., 7.

splitting the data at the "maxRs" = 7 :

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=maxRS)
```

Observation : Here we can see that we splitted the data into train and test data with test_size as 30% and the max_random_state as 7

➢ Now here we import all the necessary model libraries.

➢ I'm testing the results with the below algorithms. 1. Linear Regression 2. Random Forest Regressor 3. Decision Trees Regressor 4. KNN Regressor In order to test the model, I'm using r2 score and RMSE (Root Mean Squared Error), further in order to verify the model's fit, I'm using cross validation score to identify the best model.

Regression Algorithms :

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor as KNN
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score
from sklearn import metrics
```

Observation : Here we can see that we have imported the necessary libraries to train and test the model.

Linear Regression :

```
# Checking r2score for Linear Regression
LR = LinearRegression()
LR.fit(x_train,y_train)

# prediction
predLR=LR.predict(x_test)
print('R2_score:',metrics.r2_score(y_test,predLR))
```

```
R2_score: 0.8569831256498824
```

```
# Mean Absolute Error (MAE)

print(metrics.mean_absolute_error(y_test, predLR))

# Mean Squared Error (MSE)
print(metrics.mean_squared_error(y_test, predLR))

# Root Mean Squared Error (RMSE)
print(np.sqrt(metrics.mean_squared_error(y_test, predLR)))
```

```
188.33710470714834
106638.76215756004
326.5559096962724
```

Observation : Here we can see that we have achieved 85.6% accuracy with Linear regression model

Random Forest Regressor:

```
#Checking R2 score for Random Forest Regressor:

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

# prediction
predRFR=RFR.predict(x_test)
print('R2_Score:',metrics.r2_score(y_test,predRFR))
```

```
R2_Score: 0.984098786719313
```

```
# Mean Absolute Error (MAE)
print(metrics.mean_absolute_error(y_test, predRFR))

# Mean Squared Error (MSE)
print(metrics.mean_squared_error(y_test, predRFR))

# Root Mean Squared Error (RMSE)
print(np.sqrt(metrics.mean_squared_error(y_test, predRFR)))
```

```
19.886629213483147
11856.542864338055
108.88775350946521
```

Observation : Here we can see that we have achieved 98.4% for Random Forest Regressor model

Decision Tree Regressor :

```
# Checking R2 score for Decision Tree Regressor
DTR=DecisionTreeRegressor()
DTR.fit(x_train,y_train)

# prediction
predDTR=DTR.predict(x_test)
print('R2_Score:',metrics.r2_score(y_test,predDTR))
```

```
R2_Score: 0.9739385316014878
```

```
# Mean Absolute Error (MAE)
print(metrics.mean_absolute_error(y_test, predDTR))

# Mean Squared Error (MSE)
print(metrics.mean_squared_error(y_test, predDTR))

# Root Mean Squared Error (RMSE)
print(np.sqrt(metrics.mean_squared_error(y_test, predDTR)))
```

```
24.37420615534929
19432.411333659013
139.400184123476
```

Observation : Here we have seen that we have achieved 97.3% accuracy with Decision Tree Regressor model.

KNN regressor :

```
# Checking R2 score for KNN regressor

knn=KNN()
knn.fit(x_train,y_train)

#prediction
predknn=knn.predict(x_test)
print('R2_Score:',metrics.r2_score(y_test,predknn))
```

```
R2_Score: 0.781359340265938
```

```
# Mean Absolute Error (MAE)
print(metrics.mean_absolute_error(y_test, predknn))

# Mean Squared Error (MSE)
print(metrics.mean_squared_error(y_test, predknn))

# Root Mean Squared Error (RMSE)
print(np.sqrt(metrics.mean_squared_error(y_test, predknn)))
```

```
289.6653639472399
163026.70169027845
403.76565194463785
```

Observation : here we can see that we have achieved 78% accuracy with KNN Regressor model

Checking the Cross Validation Score :

```
from sklearn.model_selection import cross_val_score
```

```
# Checking cv score for Linear Regression
print(cross_val_score(LR,x,y,cv=5).mean())
```

```
0.7836850757481711
```

```
# Checking cv score for Random Forest Regression
print(cross_val_score(RFR,x,y,cv=5).mean())
```

```
0.9709442195306274
```

```
# Checking cv score for Decision Tree Regression
print(cross_val_score(DTR,x,y,cv=5).mean())
```

```
0.9616810327158174
```

```
# Checking cv score for KNN Regression
print(cross_val_score(knn,x,y,cv=5).mean())
```

```
0.7384108581236906
```

➢ Observation: Here we can see that among all the models **"Random Forest Regressor"** has high CV Score.

Hyper parameter Tuning :

```
from sklearn.model_selection import GridSearchCV
```

```
#RandomForestRegressor

parameters = {'criterion':['mse', 'mae'],
             'max_features':['auto', 'sqrt', 'log2'],
             'n_estimators':[0,200],
             'max_depth':[2,4,6]}
```

```
GCV=GridSearchCV(RandomForestRegressor(),parameters,cv=5)
```

```
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'criterion': ['mse', 'mae'], 'max_depth': [2, 4, 6],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [0, 200]})
```

```
GCV.best_params_
```

```
{'criterion': 'mse',
 'max_depth': 6,
 'max_features': 'auto',
 'n_estimators': 200}
```

```
Cars_model = RandomForestRegressor(criterion='mse', max_depth=6, max_features='auto', n_estimators=200)
Cars_model.fit(x_train, y_train)
pred = Cars_model.predict(x_test)
print("RMSE value:",np.sqrt(metrics.mean_squared_error(y_test, predRFR)))
print('R2_Score:',r2_score(y_test,pred)*100)
```

```
RMSE value: 108.88775350946521
R2_Score: 97.2158311996941
```

Observation : Here we can see that our best model is Random Forest model and the model is with 98.4% accuracy before hyper parameter tuning and got reduced after hyper parameter tuning.

Saving the model :

```
# Saving the model using .pkl
import pickle
filename='Cars.pkl'
pickle.dump(RFR,open(filename,'wb'))
```

Observation : Here we have saved our model with ".pkl" method

```
loaded_model=pickle.load(open('Cars.pkl','rb'))
result=loaded_model.score(x_test,y_test)
print(result)
```

```
0.984098786719313
```

```
conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],pred[:]],index=["Predicted","Original"])
```

```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	10	11
Predicted	2425.020000	69.710000	3149.900000	139.430000	336.400000	235.020000	728.640000	2412.190000	129.690000	394.730000	1385.07000	2455.570000
Original	2421.658752	67.948279	3139.124703	137.966309	337.625356	208.709865	708.647486	2409.178936	104.423996	376.550029	1388.81832	2460.830035

Observation : Here we have finally presented the Predicted and original values of our data.

So, our best model is Random Forest model with accuracy% of 98.4%

CONCLUSION

- We have successfully built a model using multiple models and found that the Random Forest Regressor model is the best model with a good accuracy.

- Below are the details of the model's metrics predicting the dataset • R2- score of 0.98 • RMSE of 108.88.

- You can view the same from the visualizations on the correlation of independent variable over dependent variable (target) As we can see from the boxplot, I couldn't remove all the outliers yet since the data is expensive, I have to proceed with the dataset with outliers.

- Further, I couldn't get skewness under control for few variables through couple of transformation techniques, yet I have proceeded with building the model. Looking at the heat map for correlation, I could see there were few independent variables which were correlated with each other, yet I have not removed any variable based on their correlation because multi-collinearity will not affect prediction.

Limitations of this work and Scope for Future Work:

- ✓ Due to the presence of lot of outliers, we are unsure whether the model is going to perform well to a completely new dataset.
- ✓ During data-collection, there are certain websites that do not provide the necessary information on the used car due to which the data collected was not precise which had to pre-processed for building a better model.

- Other than these above limitations, I couldn't find more scope for improvement.


Thank You