Car price prediction case study

By Vishal Kumar

Linkedin: https://www.linkedin.com/in/vishal-kumar-819585275/ (https://www.linkedin.com/in/vishal-kumar-819585275/)

The flow of the case study is as below:

1. Reading the data in python
2. Defining the problem statement
3. Identifying the Target variable
4. Looking at the distribution of Target variable
5. Basic Data exploration
6. Rejecting useless columns
7. Visual Exploratory Data Analysis for data distribution (Histogram and Barcharts)
8. Feature Selection based on data distribution
9. Outlier treatment
10. Missing Values treatment
11. Visual correlation analysis
12. Statistical correlation analysis (Feature Selection)
13. Converting data to numeric for ML
14. Sampling and K-fold cross validation
15. Trying multiple Regression algorithms
16. Selecting the best Model

Data description

The business meaning of each column in the data is as below

Price: The Price of the car in dollars

Age: The age of the car in months

KM: How many KMS did the car was used

FuelType: Petrol/Diesel/CNG car

HP: Horse power of the car

MetColor: Whether car has metallic color or not

Automatic: Whether car has automatic transmission or not

CC: The engine size of the car

Doors: The number of doors in the car

Weight: The weight of the car

Important Libraries

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: data=pd.read_csv('CarPricesData.csv')
        data
```

```
Out[2]:
         Price   Age     KM  FuelType   HP  MetColor  Automatic      CC  Doors  Weight
   0     13500  23.0  46986    Diesel   90         1          0  2000.0      3  1165.0
   1     13750  23.0  72937    Diesel   90         1          0  2000.0      3  1165.0
   2     13950  24.0  41711    Diesel   90         1          0  2000.0      3  1165.0
   3     14950  26.0  48000    Diesel   90         0          0  2000.0      3  1165.0
   4     13750  30.0  38500    Diesel   90         0          0  2000.0      3  1170.0
 ...       ...   ...    ...       ...  ...       ...        ...     ...    ...     ...
1431      7500  69.0  20544    Petrol   86         1          0  1300.0      3  1025.0
1432     10845  72.0  19000    Petrol   86         0          0  1300.0      3  1015.0
1433      8500  71.0  17016    Petrol   86         0          0  1300.0      3  1015.0
1434      7250  70.0  16916    Petrol   86         1          0  1300.0      3  1015.0
1435      6950  76.0      1    Petrol  110         0          0  1600.0      5  1114.0

1436 rows × 10 columns
```

```
In [3]: data.shape
```

```
Out[3]: (1436, 10)
```

In [4]:

```
data.head(10)
```

Out[4]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986	Diesel	90	1	0	2000.0	3	1165.0
1	13750	23.0	72937	Diesel	90	1	0	2000.0	3	1165.0
2	13950	24.0	41711	Diesel	90	1	0	2000.0	3	1165.0
3	14950	26.0	48000	Diesel	90	0	0	2000.0	3	1165.0
4	13750	30.0	38500	Diesel	90	0	0	2000.0	3	1170.0
5	12950	32.0	61000	Diesel	90	0	0	2000.0	3	1170.0
6	16900	27.0	94612	Diesel	90	1	0	2000.0	3	1245.0
7	18600	30.0	75889	Diesel	90	1	0	2000.0	3	1245.0
8	21500	27.0	19700	Petrol	192	0	0	1800.0	3	1185.0
9	12950	23.0	71138	Diesel	69	0	0	1900.0	3	1105.0

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Price      1436 non-null   int64  
 1   Age        1434 non-null   float64
 2   KM         1436 non-null   int64  
 3   FuelType   1432 non-null   object 
 4   HP         1436 non-null   int64  
 5   MetColor   1436 non-null   int64  
 6   Automatic  1436 non-null   int64  
 7   CC         1434 non-null   float64
 8   Doors      1436 non-null   int64  
 9   Weight     1434 non-null   float64
dtypes: float64(3), int64(6), object(1)
memory usage: 112.3+ KB
```

In [6]:
```
data.describe(include='all')
```

Out[6]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic
count	1436.000000	1434.000000	1436.000000	1432	1436.000000	1436.000000	1436.000000
unique	NaN	NaN	NaN	3	NaN	NaN	NaN
top	NaN	NaN	NaN	Petrol	NaN	NaN	NaN
freq	NaN	NaN	NaN	1260	NaN	NaN	NaN
mean	10730.824513	55.986750	68533.259749	NaN	101.502089	0.674791	0.055710
std	3626.964585	18.581796	37506.448872	NaN	14.981080	0.468616	0.229441
min	4350.000000	1.000000	1.000000	NaN	69.000000	0.000000	0.000000
25%	8450.000000	44.000000	43000.000000	NaN	90.000000	0.000000	0.000000
50%	9900.000000	61.000000	63389.500000	NaN	110.000000	1.000000	0.000000
75%	11950.000000	70.000000	87020.750000	NaN	110.000000	1.000000	0.000000
max	32500.000000	80.000000	243000.000000	NaN	192.000000	1.000000	1.000000

In [7]:
```
data.isna().sum()    #isnull()
```

Out[7]:
```
Price        0
Age          2
KM           0
FuelType     4
HP           0
MetColor     0
Automatic    0
CC           2
Doors        0
Weight       2
dtype: int64
```

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

Out[8]:
```
Price         236
Age            77
KM           1263
FuelType        3
HP             12
MetColor        2
Automatic       2
CC             12
Doors           4
Weight         59
dtype: int64
```

In [9]:
```
data['HP'].unique()
```

Out[9]:
```
array([ 90, 192,  69, 110,  97,  71, 116,  98,  86,  72, 107,  73],
      dtype=int64)
```

In [10]:
```
data
```

Out[10]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986	Diesel	90	1	0	2000.0	3	1165.0
1	13750	23.0	72937	Diesel	90	1	0	2000.0	3	1165.0
2	13950	24.0	41711	Diesel	90	1	0	2000.0	3	1165.0
3	14950	26.0	48000	Diesel	90	0	0	2000.0	3	1165.0
4	13750	30.0	38500	Diesel	90	0	0	2000.0	3	1170.0
...	...	...	...	...	...	...	...	...	...	...
1431	7500	69.0	20544	Petrol	86	1	0	1300.0	3	1025.0
1432	10845	72.0	19000	Petrol	86	0	0	1300.0	3	1015.0
1433	8500	71.0	17016	Petrol	86	0	0	1300.0	3	1015.0
1434	7250	70.0	16916	Petrol	86	1	0	1300.0	3	1015.0
1435	6950	76.0	1	Petrol	110	0	0	1600.0	5	1114.0

1436 rows × 10 columns

EDA

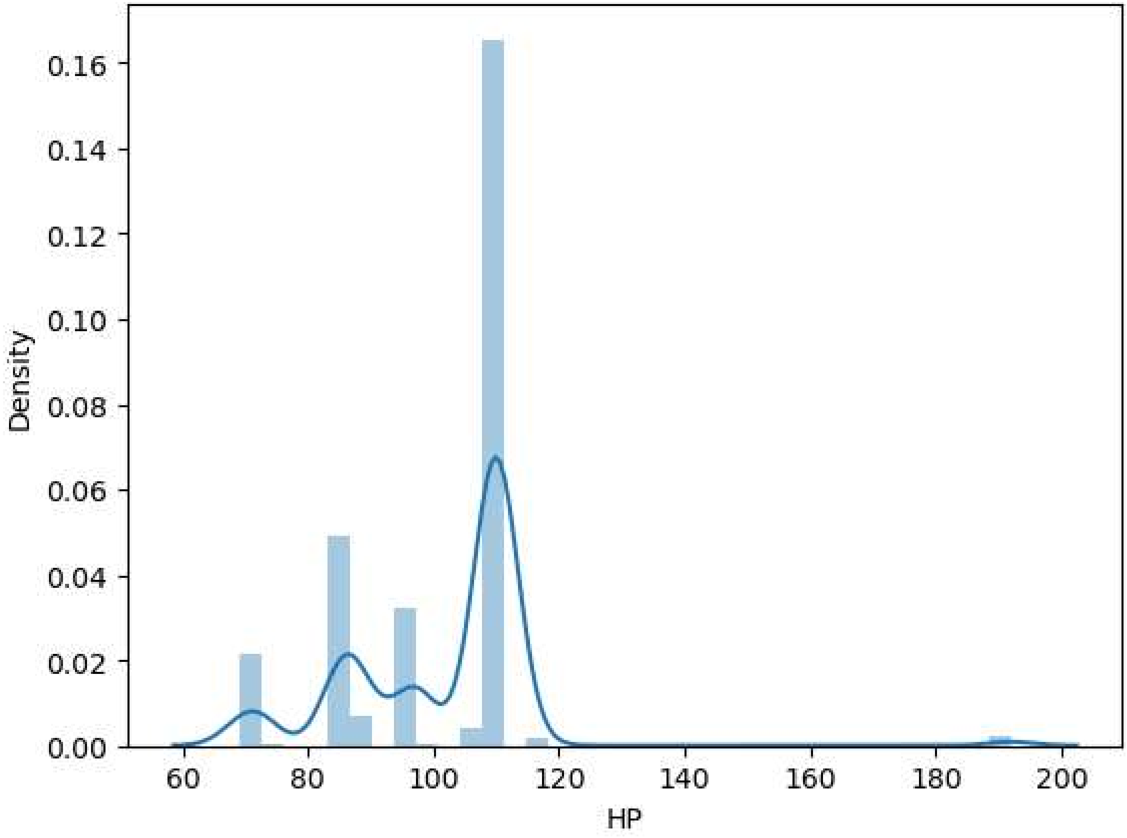
In [11]:
```
data.columns
```

Out[11]:
```
Index(['Price', 'Age', 'KM', 'FuelType', 'HP', 'MetColor', 'Automatic', 'CC',
       'Doors', 'Weight'],
      dtype='object')
```

In [12]:
```
sns.distplot(data.HP)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[12]:
```
<AxesSubplot:xlabel='HP', ylabel='Density'>
```



In [13]:
```
data.HP.value_counts()
```

Out[13]:
```
110    835
86     249
97     164
72      73
90      36
69      34
107     21
192     11
116      9
98       2
71       1
73       1
Name: HP, dtype: int64
```

In [14]:
```
# def plots(data,colstoplot):
#         fig,subplot=plt.subplot(nrows=1,ncols = len(colstoplot),figsize=(20,6))
#         fig.suptitle('Bar charts of:' +str(colstoplot))

#         for i,j in zip(colstoplot,range(len(colstoplot))):
#             data.groupby(i).size().plot(kind='bar',ax=subplot[j])
```

In [15]:
```
# plots(data=data,colstoplot=['FuelType','HP','Metcolor','Automatic','CC','Doors
```

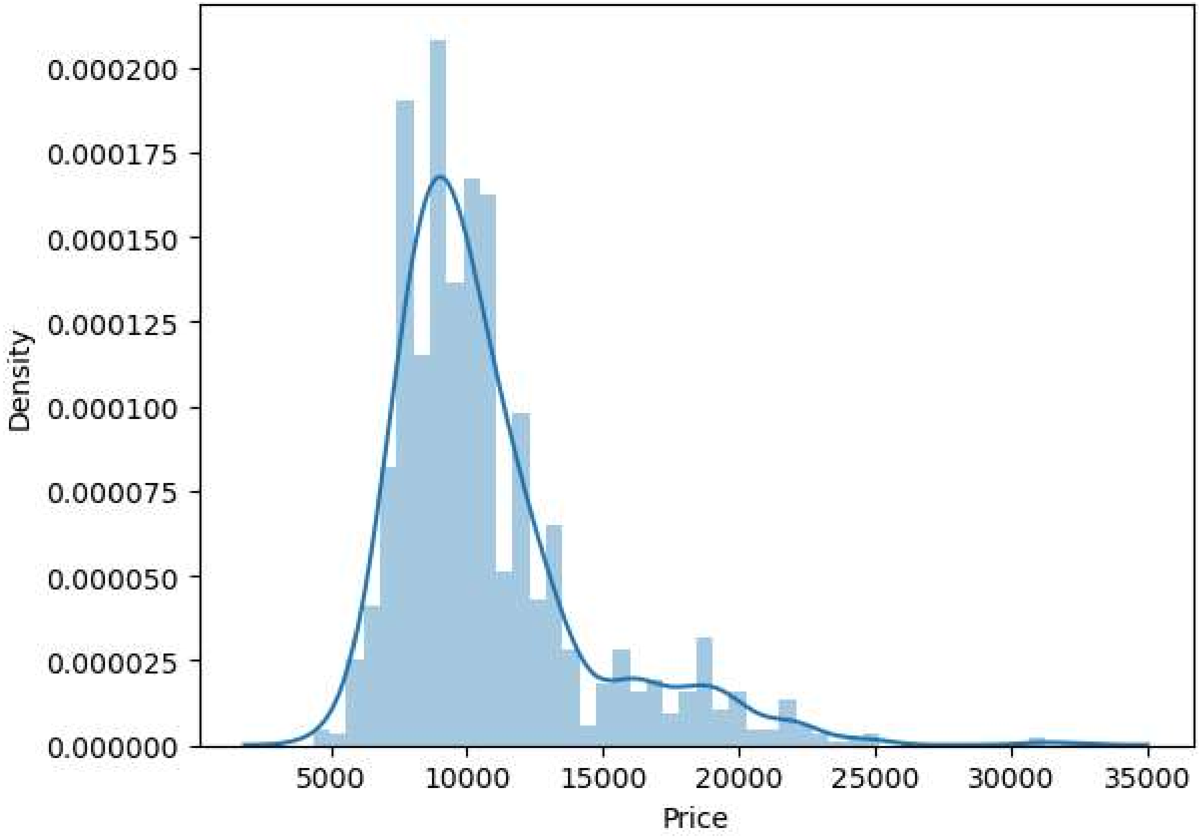
In [16]:
```
data.columns
```

Out[16]:
```
Index(['Price', 'Age', 'KM', 'FuelType', 'HP', 'MetColor', 'Automatic', 'CC',
       'Doors', 'Weight'],
      dtype='object')
```

In [17]:
```
sns.distplot(data.Price)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

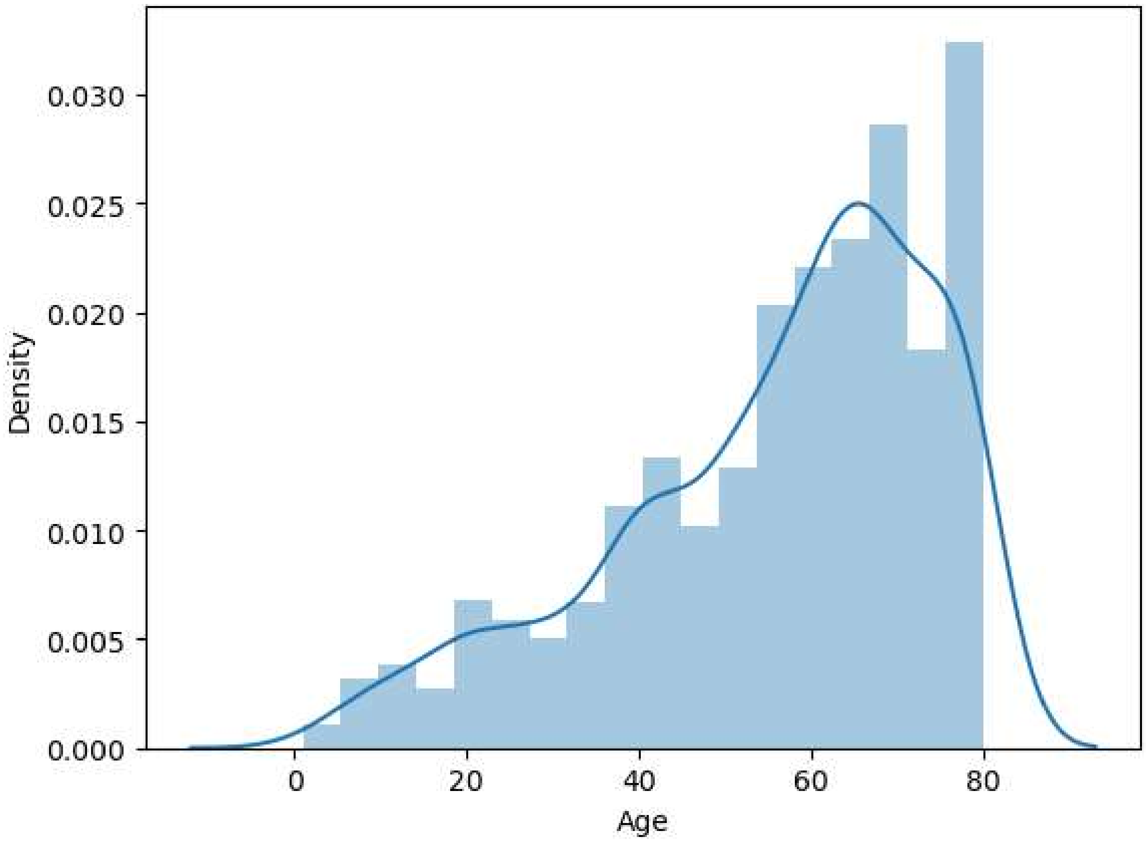
Out[17]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [18]:
```
sns.distplot(data.Age)
```

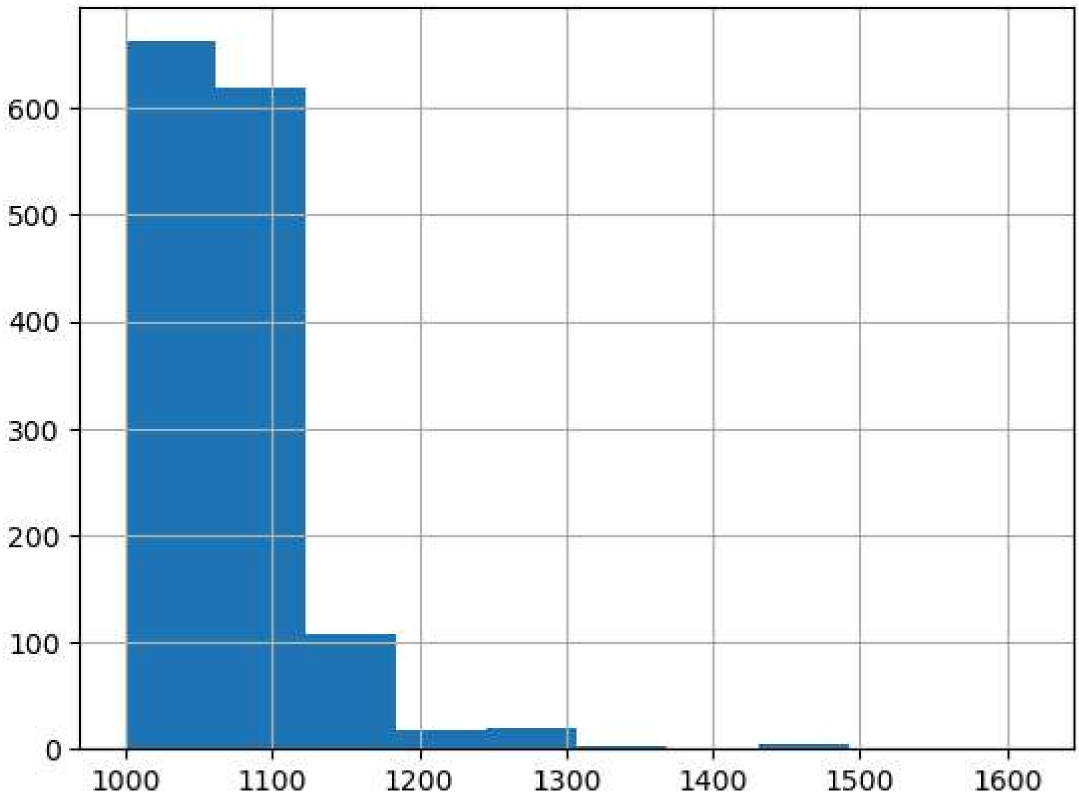
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[18]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



In [19]:
```
data.Weight.hist()
```

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



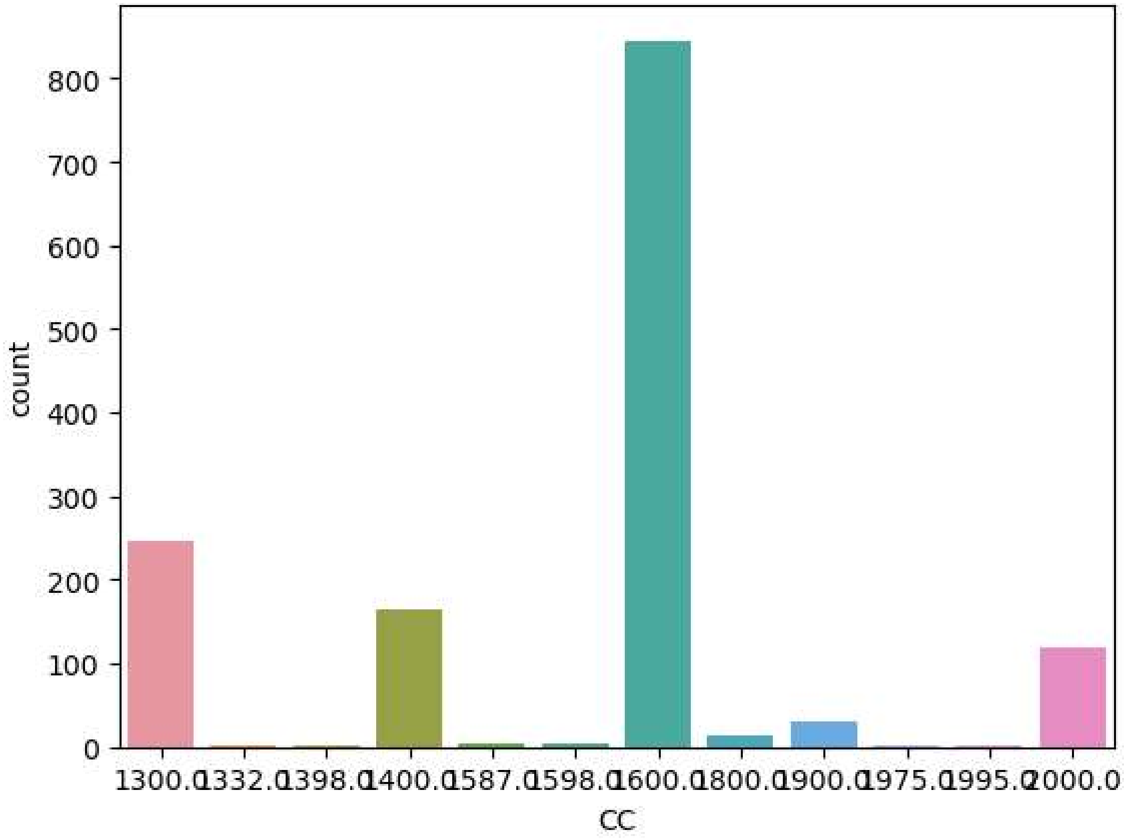
In [20]:

```
sns.countplot(data.CC)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[20]:

```
<AxesSubplot:xlabel='CC', ylabel='count'>
```



In [21]:

```
# outliers

data.Weight.describe()
```

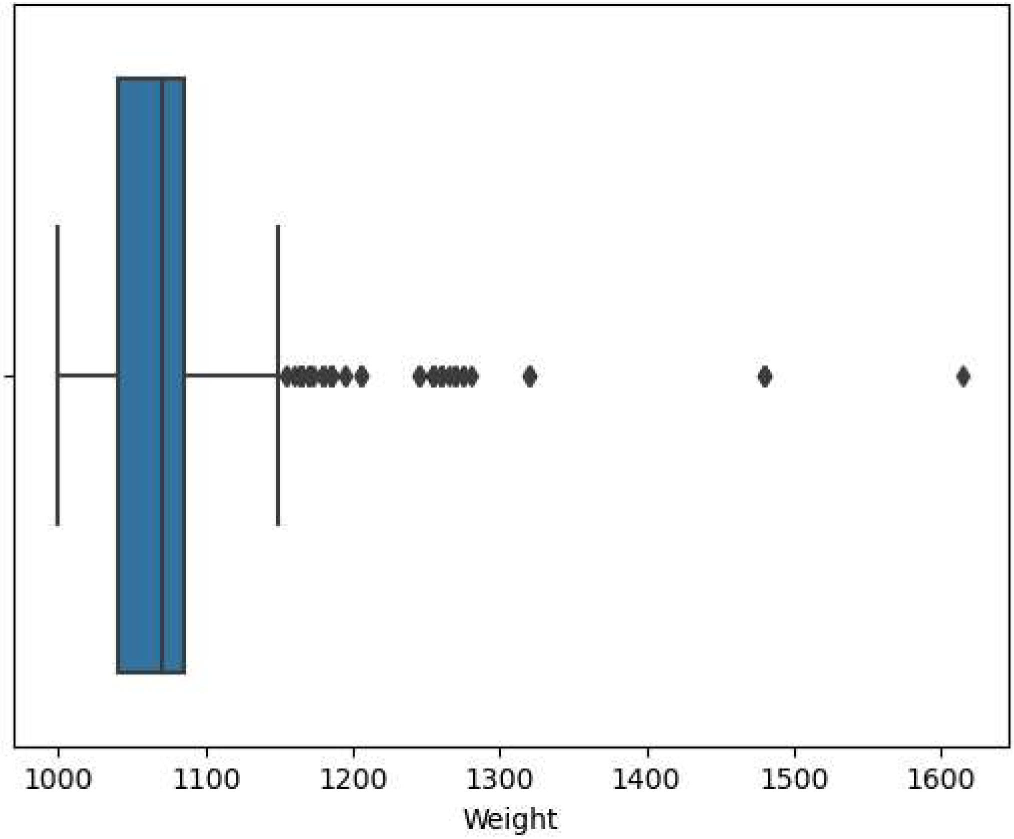
Out[21]:

```
count    1434.000000
mean     1072.487448
std        52.672475
min      1000.000000
25%      1040.000000
50%      1070.000000
75%      1085.000000
max      1615.000000
Name: Weight, dtype: float64
```

```
In [22]: sns.boxplot(data.Weight)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[22]: <AxesSubplot:xlabel='Weight'>
```



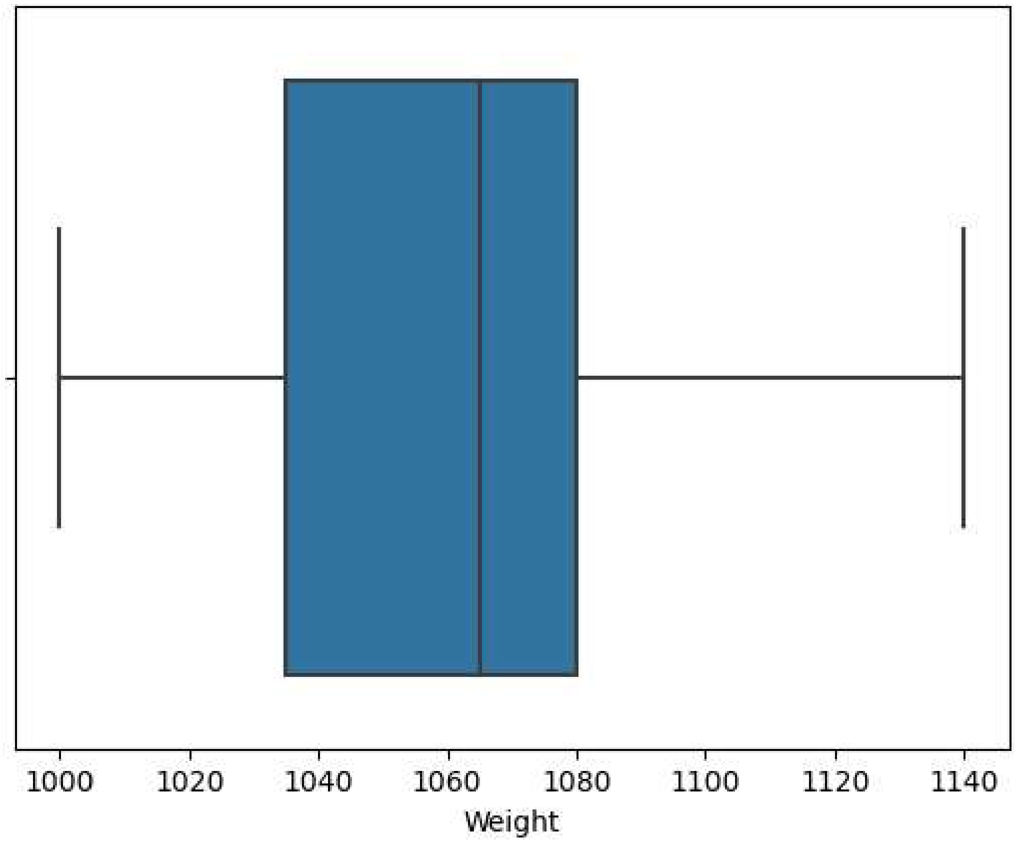
In [23]:

```
data=data[data['Weight']<1150]
sns.boxplot(data.Weight)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

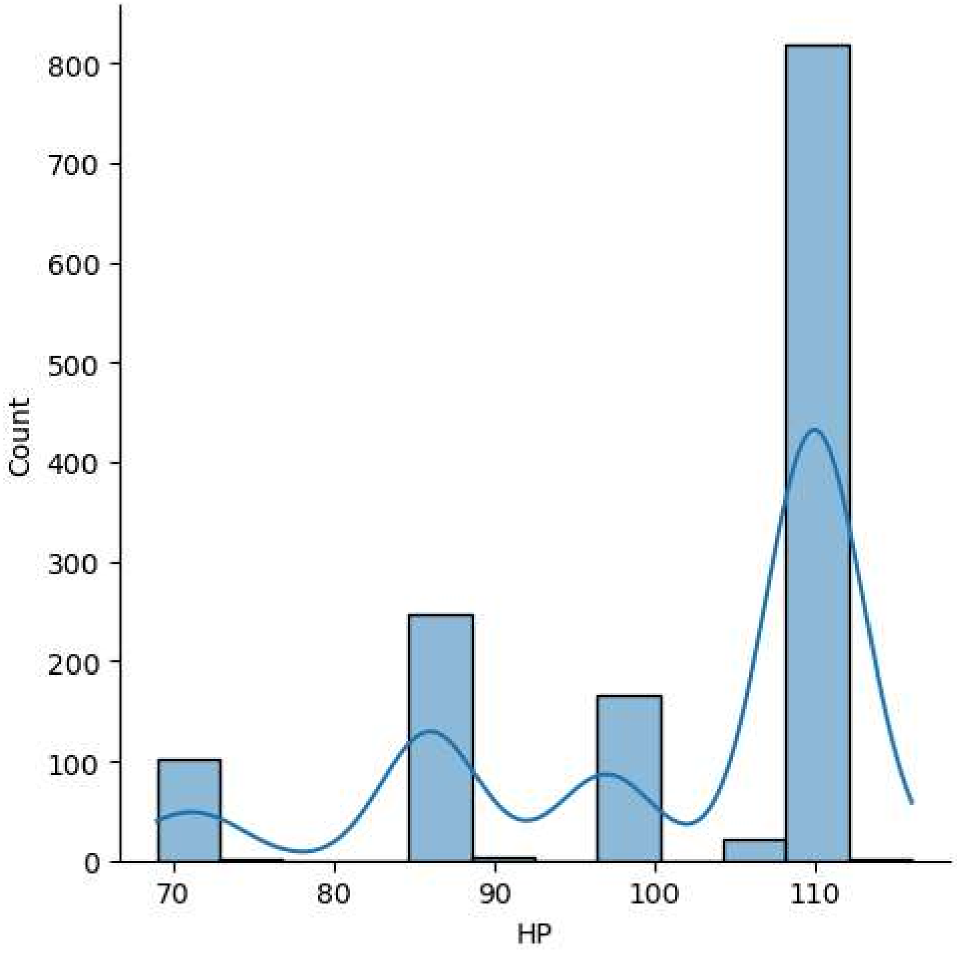
Out[23]:

```
<AxesSubplot:xlabel='Weight'>
```



In [24]:
```
sns.displot(data.HP, kde=True)
```

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x2d3ee1be880>
```



missing value treatment

In [25]:
```
data.isna().sum()
```

Out[25]:
```
Price        0
Age          2
KM           0
FuelType     3
HP           0
MetColor     0
Automatic    0
CC           2
Doors        0
Weight       0
dtype: int64
```

In [26]:

```
data[data.Age.isna()]
```

Out[26]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
38	15750	NaN	25329	Petrol	97	1	0	1400.0	3	1100.0
73	15750	NaN	28227	Petrol	97	1	0	1400.0	5	1110.0

In [27]:

```
data.Age.fillna(0,inplace=True)
```

```
C:\Users\visha\AppData\Local\Temp\ipykernel_24900\66375761.py:1: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  data.Age.fillna(0,inplace=True)
```

In [28]:

```
data.isna().sum()
```

Out[28]:

```
Price        0
Age          0
KM           0
FuelType     3
HP           0
MetColor     0
Automatic    0
CC           2
Doors        0
Weight       0
dtype: int64
```

In [29]:

```
data.FuelType.fillna('NA',inplace=True)
```

```
C:\Users\visha\AppData\Local\Temp\ipykernel_24900\2093027534.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  data.FuelType.fillna('NA',inplace=True)
```

In [30]:

```
data[data.CC.isna()]
```

Out[30]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
1392	7500	79.0	49827	Petrol	86	1	0	NaN	5	1035.0
1401	8950	71.0	47633	Petrol	110	1	0	NaN	5	1075.0

In [31]:

```
data['CC']=np.where(data['HP']==110,1600.0,data['CC'])
data['CC']=np.where(data['HP']==86,1300.0,data['CC'])
```

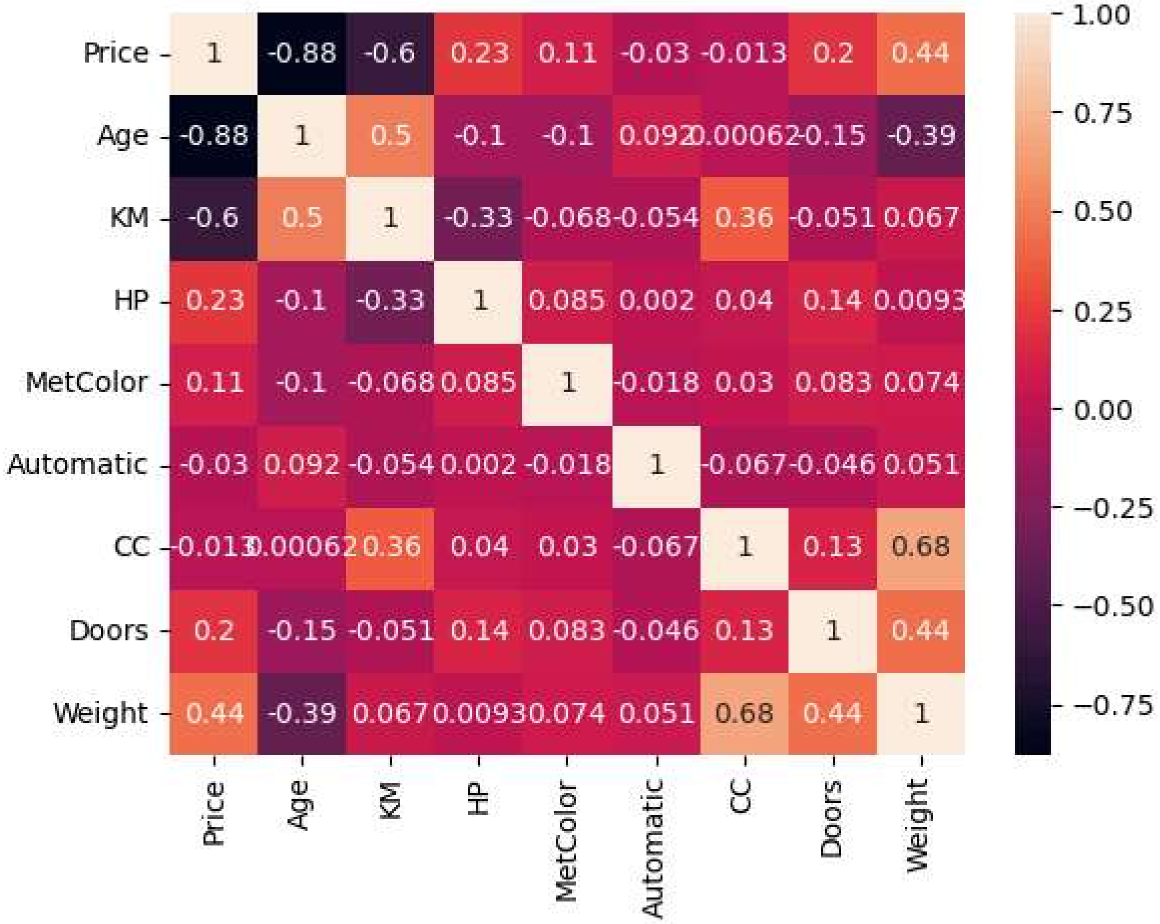
```
C:\Users\visha\AppData\Local\Temp\ipykernel_24900\3237792382.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  data['CC']=np.where(data['HP']==110,1600.0,data['CC'])
C:\Users\visha\AppData\Local\Temp\ipykernel_24900\3237792382.py:2: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  data['CC']=np.where(data['HP']==86,1300.0,data['CC'])
```

In [32]:
```
sns.heatmap(data.corr(),annot=True)
```

Out[32]:
```
<AxesSubplot:>
```



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

Out[33]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
9	12950	23.0	71138	Diesel	69	0	0	1900.0	3	1105.0
17	17950	24.0	21716	Petrol	110	1	0	1600.0	3	1105.0
18	16750	24.0	25563	Petrol	110	0	0	1600.0	3	1065.0
19	16950	30.0	64359	Petrol	110	1	0	1600.0	3	1105.0
20	15950	30.0	67660	Petrol	110	1	0	1600.0	3	1105.0

In [34]:

```python
data=pd.get_dummies(data)
data.head()
```

Out[34]:

	Price	Age	KM	HP	MetColor	Automatic	CC	Doors	Weight	FuelType_CNG	FuelType
9	12950	23.0	71138	69	0	0	1900.0	3	1105.0	0	
17	17950	24.0	21716	110	1	0	1600.0	3	1105.0	0	
18	16750	24.0	25563	110	0	0	1600.0	3	1065.0	0	
19	16950	30.0	64359	110	1	0	1600.0	3	1105.0	0	
20	15950	30.0	67660	110	1	0	1600.0	3	1105.0	0	

In [35]:

```python
x=data.drop(columns='Price')
y=data['Price']
```

In [36]:

```python
x.head(2)
```

Out[36]:

	Age	KM	HP	MetColor	Automatic	CC	Doors	Weight	FuelType_CNG	FuelType_Diesel
9	23.0	71138	69	0	0	1900.0	3	1105.0	0	1
17	24.0	21716	110	1	0	1600.0	3	1105.0	0	0

Train and test split

In [37]:

```python
from sklearn.preprocessing import MinMaxScaler
```

In [38]:

```python
pred = MinMaxScaler()
fit= pred.fit(x)
x= fit.transform(x)
x
```

Out[38]:

```
array([[0.2875    , 0.29274606, 0.        , ..., 1.        , 0.        ,
        0.        ],
       [0.3       , 0.08936251, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.3       , 0.10519385, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.8875    , 0.07002086, 0.36170213, ..., 0.        , 0.        ,
        1.        ],
       [0.875     , 0.06960934, 0.36170213, ..., 0.        , 0.        ,
        1.        ],
       [0.95      , 0.        , 0.87234043, ..., 0.        , 0.        ,
        1.        ]])
```

In [39]:

```
from sklearn.model_selection import train_test_split
```

In [40]:

```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=0)
```

In [41]:

```
x_train
```

Out[41]:

```
array([[0.8625    , 0.30688192, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.9       , 0.36419491, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.925     , 0.65805621, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.975     , 0.29666377, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.7625    , 0.56360314, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.7125    , 0.33114951, 0.87234043, ..., 0.        , 0.        ,
        1.        ]])
```

In [42]:

```
x_test
```

Out[42]:

```
array([[0.875     , 0.35802205, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.775     , 0.36213729, 0.36170213, ..., 0.        , 0.        ,
        1.        ],
       [0.925     , 0.50648768, 0.36170213, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.8125    , 0.34797263, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.9875    , 0.40634324, 0.87234043, ..., 0.        , 0.        ,
        1.        ],
       [0.7625    , 0.18352339, 0.36170213, ..., 0.        , 0.        ,
        1.        ]])
```

In [43]:

```
##Modeling

from sklearn.linear_model import LinearRegression

RegModel = LinearRegression()
fit= RegModel.fit(x_train,y_train)
y_pred=fit.predict(x_test)
```

In [44]:

```
pd.DataFrame(y_pred)
```

Out[44]:

	0
0	7919.508048
1	8768.834842
2	6743.689488
3	10702.649756
4	9466.531229
...	...
268	12777.311554
269	9004.252894
270	10903.365087
271	7928.360325
272	9430.499338

273 rows × 1 columns

In [45]:

```
pd.DataFrame(y_test)
```

Out[45]:

	Price
1208	7950
724	6900
1103	7950
989	9950
867	9750
...	...
313	11650
659	10500
734	10950
1159	8750
987	9950

273 rows × 1 columns

In [46]:

```
from sklearn import metrics
```

In [47]:

```
print('R2 score',metrics.r2_score(y_test,y_pred) )
```

```
R2 score 0.847187007709763
```

In [48]:

```
## Model 2
from sklearn.tree import DecisionTreeRegressor


RegModel = DecisionTreeRegressor()
fit= RegModel.fit(x_train,y_train)
y_pred=fit.predict(x_test)
print('R2 score',metrics.r2_score(y_test,y_pred) )
pd.DataFrame(y_pred)
```

```
R2 score 0.7787571241977113
```

Out[48]:

	0
0	7500.0
1	8750.0
2	7350.0
3	10495.0
4	9950.0
...	...
268	13950.0
269	9750.0
270	10500.0
271	9500.0
272	8950.0

273 rows × 1 columns

In [49]:
```
pd.DataFrame(y_test)
```

Out[49]:

	Price
1208	7950
724	6900
1103	7950
989	9950
867	9750
...	...
313	11650
659	10500
734	10950
1159	8750
987	9950

273 rows × 1 columns

In [50]:
```
### Model3
from sklearn.ensemble import RandomForestRegressor
RegModel = RandomForestRegressor()
fit= RegModel.fit(x_train,y_train)
y_pred=fit.predict(x_test)
print('R2 score',metrics.r2_score(y_test,y_pred) )
pd.DataFrame(y_pred)
```

```
R2 score 0.8735473593697775
```

Out[50]:

	0
0	7766.50
1	9103.90
2	7127.80
3	9682.50
4	10076.95
...	...
268	12282.90
269	9475.50
270	9878.35
271	8067.40
272	9156.95

273 rows × 1 columns

In [51]:

```
## model4
from sklearn.ensemble import AdaBoostRegressor
DTR = RandomForestRegressor()
model = AdaBoostRegressor(n_estimators=100,base_estimator=DTR, learning_rate =0.


fit= model.fit(x_train,y_train)
y_pred=fit.predict(x_test)
print('R2 score',metrics.r2_score(y_test,y_pred) )
pd.DataFrame(y_pred)
```

```
R2 score 0.8765790362093672
```

Out[51]:

	0
0	7938.523333
1	9047.500000
2	7132.050000
3	9950.500000
4	10074.000000
...	...
268	12269.950000
269	9411.000000
270	9982.800000
271	8019.400000
272	9252.000000

273 rows × 1 columns

In [52]:

```
## Model5
from xgboost import XGBRegressor

model = XGBRegressor(max_depth=  5, learning_rate=0.1, n_estimators= 100, objecti


fit= model.fit(x_train,y_train)
y_pred=fit.predict(x_test)
print('R2 score',metrics.r2_score(y_test,y_pred) )
pd.DataFrame(y_pred)
```

```
[20:02:41] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling
-group-i-0fc7796c793e6356f-1/xgboost/xgboost-ci-windows/src/objective/regressio
n_obj.cu:213: reg:linear is now deprecated in favor of reg:squarederror.
R2 score 0.8561114228092584
```

Out[52]:

	0
0	7879.320801
1	9060.098633
2	6767.004395
3	10225.208984
4	10327.085938
...	...
268	11888.777344
269	9338.845703
270	9944.996094
271	7893.187988
272	9003.925781

273 rows × 1 columns

In [53]:

```
### Model3
from sklearn.ensemble import RandomForestRegressor
final_model = RandomForestRegressor()
fit= final_model.fit(x_train,y_train)
y_pred_final=fit.predict(x_test)

print('R2 score',metrics.r2_score(y_train,fit.predict(x_train)) )
print('R2 score',metrics.r2_score(y_test,y_pred_final) )




##gridsearch cv or randomize search cv
```

```
R2 score 0.9830181638584639
R2 score 0.8740755046220299
```

In [54]:
```
data.corr()
```

Out[54]:

	Price	Age	KM	HP	MetColor	Automatic	CC	Doo
Price	1.000000	-0.879118	-0.598447	0.229229	0.107950	-0.029610	-0.012986	0.2016
Age	-0.879118	1.000000	0.503338	-0.102598	-0.104782	0.091600	0.000617	-0.1511
KM	-0.598447	0.503338	1.000000	-0.326902	-0.067687	-0.053640	0.358502	-0.0514
HP	0.229229	-0.102598	-0.326902	1.000000	0.084814	0.001982	0.040460	0.1408
MetColor	0.107950	-0.104782	-0.067687	0.084814	1.000000	-0.018161	0.030212	0.0830
Automatic	-0.029610	0.091600	-0.053640	0.001982	-0.018161	1.000000	-0.066563	-0.0455
CC	-0.012986	0.000617	0.358502	0.040460	0.030212	-0.066563	1.000000	0.1294
Doors	0.201637	-0.151186	-0.051427	0.140813	0.083052	-0.045524	0.129484	1.0000
Weight	0.435891	-0.390328	0.067037	0.009341	0.073556	0.051311	0.676797	0.4378
FuelType_CNG	-0.034446	-0.006060	0.150121	0.079857	0.021651	0.004166	0.033298	0.0106
FuelType_Diesel	-0.154968	0.045091	0.484407	-0.672941	-0.032168	-0.067162	0.706243	0.0029
FuelType_NA	-0.019086	0.018746	-0.002797	0.003781	-0.000786	0.060554	-0.013486	-0.0178
FuelType_Petrol	0.159572	-0.042403	-0.504956	0.590993	0.021573	0.050704	-0.663573	-0.0039

In [55]:
```
data
```

Out[55]:

	Price	Age	KM	HP	MetColor	Automatic	CC	Doors	Weight	FuelType_CNG	FuelTy
9	12950	23.0	71138	69	0	0	1900.0	3	1105.0	0	
17	17950	24.0	21716	110	1	0	1600.0	3	1105.0	0	
18	16750	24.0	25563	110	0	0	1600.0	3	1065.0	0	
19	16950	30.0	64359	110	1	0	1600.0	3	1105.0	0	
20	15950	30.0	67660	110	1	0	1600.0	3	1105.0	0	
...	...	...	...	...	...	...	...	...	...	...	
1431	7500	69.0	20544	86	1	0	1300.0	3	1025.0	0	
1432	10845	72.0	19000	86	0	0	1300.0	3	1015.0	0	
1433	8500	71.0	17016	86	0	0	1300.0	3	1015.0	0	
1434	7250	70.0	16916	86	1	0	1300.0	3	1015.0	0	
1435	6950	76.0	1	110	0	0	1600.0	5	1114.0	0	

1361 rows × 13 columns

In [56]:

```
## feature engineering
x=data.drop(columns=['Price','CC','FuelType_NA'])
y=data['Price']

x.head(2)
```

Out[56]:

	Age	KM	HP	MetColor	Automatic	Doors	Weight	FuelType_CNG	FuelType_Diesel	FuelTy
9	23.0	71138	69	0	0	3	1105.0	0	1	
17	24.0	21716	110	1	0	3	1105.0	0	0	

In [57]:

```
pred = MinMaxScaler()
fit= pred.fit(x)
x= fit.transform(x)
x

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=0

x_train

x_test

### Model3
from sklearn.ensemble import RandomForestRegressor
final_model = RandomForestRegressor()
fit= final_model.fit(x_train,y_train)
y_pred_final=fit.predict(x_test)

print('Base model accuracies',0.8744112420928829)

# print('R2 score',metrics.r2_score(y_train,fit.predict(x_train)) )
print('R2 score',metrics.r2_score(y_test,y_pred_final) )


##gridsearch cv or randomize search cv
```

```
Base model accuracies 0.8744112420928829
R2 score 0.8745785677723594
```

In [58]:

```
#deploy
# 1. flask api
# 2. prediction
# 3. final model
```

In [59]:

```
# save
import joblib
joblib.dump(fit, "car_price_final_model.joblib")
```

Out[59]:

```
['car_price_final_model.joblib']
```

In [60]:

```
# load, no need to initialize the loaded_rf
loaded_rf = joblib.load("car_price_final_model.joblib")
```

In [61]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1361 entries, 9 to 1435
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Price            1361 non-null   int64  
 1   Age              1361 non-null   float64
 2   KM               1361 non-null   int64  
 3   HP               1361 non-null   int64  
 4   MetColor         1361 non-null   int64  
 5   Automatic        1361 non-null   int64  
 6   CC               1361 non-null   float64
 7   Doors            1361 non-null   int64  
 8   Weight           1361 non-null   float64
 9   FuelType_CNG     1361 non-null   uint8  
 10  FuelType_Diesel  1361 non-null   uint8  
 11  FuelType_NA      1361 non-null   uint8  
 12  FuelType_Petrol  1361 non-null   uint8  
dtypes: float64(3), int64(6), uint8(4)
memory usage: 111.6 KB
```

In [62]:

```
### prediction code
import pandas as pd
Age= float(input('Enter age of your car'))
KM= int(input('Enter number of km'))
FuelType= input('fuel type of your car:-Diesel, Petrol or CNG ')
HP= int(input('Enter value of HP'))
MetColor= int(input('Enter the value for metcolor'))
Automatic= int(input('Enter the value for Automatic'))
Doors= int(input('Enter the value for Doors'))
Weight= float(input('Enter the value for weight'))

# # input_series= pd.Series([Age,KM,HP,MetColor,Automatic,Doors,Weight,FuelType])
# input_indexes = pd.Series['Age','KM','HP','MetColor','Automatic','Doors','Weigh

#Creating a dictionary by passing Series objects as values
frame = {'Age':[Age],'KM':[KM],'HP':[HP],'MetColor':[MetColor],'Automatic':[Autor
#Creating DataFrame by passing Dictionary
Test_data = pd.DataFrame.from_dict(frame)
# #Printing elements of Dataframe
# print(result)
Test_data
```

```
Enter age of your car26
Enter number of km46666
fuel type of your car:-Diesel, Petrol or CNG Petrol
Enter value of HP120
Enter the value for metcolor1
Enter the value for Automatic0
Enter the value for Doors4
Enter the value for weight1200
```

Out[62]:

	Age	KM	HP	MetColor	Automatic	Doors	Weight	FuelType
0	26.0	46666	120	1	0	4	1200.0	Petrol

```
# prediction code

# import pandas as pd

# Age= 25.2
# KM= 45785
# FuelType= 'Diesel'
# HP= 90
# MetColor= 1
# Automatic= 0
# Doors= 3
# Weight= 1165.2


# frame = {'Age':[Age],'KM':[KM],'HP':[HP],'MetColor':[MetColor],'Automatic':
[Automatic],'Doors':[Doors],'Weight':[Weight],'FuelType':[FuelType]}
# #Creating DataFrame by passing Dictionary
# Test_data = pd.DataFrame.from_dict(frame)
# # #Printing elements of Dataframe
# # print(result)
# Test_data
```

Prediction

In [63]:

```
def predciction_code(Test_data):
    test=pd.get_dummies(Test_data)
    if 'FuelType_Diesel' not in test.columns:
        test['FuelType_Diesel'] = 0
    if 'FuelType_Petrol' not in test.columns:
        test['FuelType_Petrol'] = 0
    if 'FuelType_CNG' not in test.columns:
        test['FuelType_CNG'] = 0

    pred = MinMaxScaler()
    fit= pred.fit(x)
    test= fit.transform(test)
    pred_new=loaded_rf.predict(test)
    return pred_new

predciction_code(Test_data)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X
has feature names, but MinMaxScaler was fitted without feature names
  warnings.warn(
```

Out[63]:

```
array([5957.3])
```