Experiment 9-10

Student Name: Aditya Abhiram **UID: 21BCS9710**

Branch: CSE(AIML) **Section/Group: AIML-12(B)**

Semester: 3rd **Date of Performance: 05/11/22**

Subject Name: Python For Machine Learning **Subject Code: 21CSH-238**

Aim of the practical: Build a multiple linear regression model for the prediction of car prices.

Steps for experiment /practical:

Step1. Let us import our libraries. Numpy is a fast matrix computation library that most of the other libraries depend on and we might need it at some point. Pandas is our data manipulation library and one of the most important libraries in our pipeline. Matplotlib and Seaborn are used for plotting graphs.

Step2. Import the cvs file.

Step3. We can use head() to view first five records. We observe that there are a lot of variables and many of them are categorical. So, feature selection will play an important role going forward.
Let us check if there are any missing values.

```
[1] import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import seaborn as sns
    import matplotlib.pyplot as plt


    from sklearn.tree import DecisionTreeRegressor
    from sklearn import datasets

    import os
    for dirname, _, filenames in os.walk('/content/os.walk'):
        for filename in filenames:
            print(os.path.join(dirname, filename))

    /content/os.walk/Data Dictionery - carprices.xlsx
    /content/os.walk/CarPrice_Assignment.csv

[2] data = pd.read_csv('/content/os.walk/CarPrice_Assignment.csv')
    df = data.copy()

    df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg	price
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495.0
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500.0
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500.0
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950.0
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450.0

5 rows × 26 columns

```
[3] df.columns

    Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
           'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
           'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
           'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
           'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
           'price'],
          dtype='object')
```

```
[8]  df.shape

     (205, 26)

[9]  df.dtypes

     car_ID              int64
     symboling           int64
     CarName            object
     fueltype           object
     aspiration         object
     doornumber         object
     carbody            object
     drivewheel         object
     enginelocation     object
     wheelbase         float64
     carlength         float64
     carwidth          float64
     carheight         float64
     curbweight          int64
     enginetype         object
     cylindernumber     object
     enginesize          int64
     fuelsystem         object
     boreratio         float64
     stroke            float64
     compressionratio  float64
     horsepower          int64
     peakrpm             int64
     citympg             int64
     highwaympg          int64
     price             float64
     dtype: object
```

Step4. Turns out there aren't any. So we do not need to worry about filling any missing values.

```
df.info()
```

Python

```
Output exceeds the size limit. Open the full output data in a text editor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
```

Step5. Now, we shall do some processing of our data.

1) We only want the company name. So lets split the CarName and extract only company name. We will rename it to Company to avoid confusion.
2) We will calculate total miles per gallon and remove citympg and highwaympg
3) We do not require ID as well so lets remove that as well.
4) We will change the datatype of symboling to string since its a categorical variable and should not be confused to be continuous.

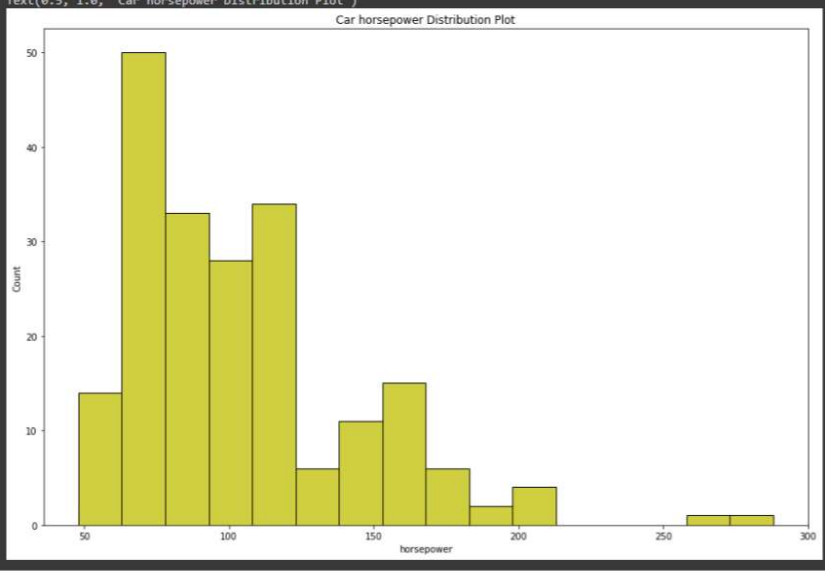
We can get descriptive statistical values using describe() of pandas.

```
[11] df.describe().T # T method for transpose
```

	count	mean	std	min	25%	50%	75%	max
car_ID	205.0	103.000000	59.322565	1.00	52.00	103.00	154.00	205.00
symboling	205.0	0.834146	1.245307	-2.00	0.00	1.00	2.00	3.00
wheelbase	205.0	98.756585	6.021776	86.60	94.50	97.00	102.40	120.90
carlength	205.0	174.049268	12.337289	141.10	166.30	173.20	183.10	208.10
carwidth	205.0	65.907805	2.145204	60.30	64.10	65.50	66.90	72.30
carheight	205.0	53.724878	2.443522	47.80	52.00	54.10	55.50	59.80
curbweight	205.0	2555.565854	520.680204	1488.00	2145.00	2414.00	2935.00	4066.00
enginesize	205.0	126.907317	41.642693	61.00	97.00	120.00	141.00	326.00
boreratio	205.0	3.329756	0.270844	2.54	3.15	3.31	3.58	3.94
stroke	205.0	3.255415	0.313597	2.07	3.11	3.29	3.41	4.17
compressionratio	205.0	10.142537	3.972040	7.00	8.60	9.00	9.40	23.00
horsepower	205.0	104.117073	39.544167	48.00	70.00	95.00	116.00	288.00
peakrpm	205.0	5125.121951	476.985643	4150.00	4800.00	5200.00	5500.00	6600.00
citympg	205.0	25.219512	6.542142	13.00	19.00	24.00	30.00	49.00
highwaympg	205.0	30.751220	6.886443	16.00	25.00	30.00	34.00	54.00
price	205.0	13276.710571	7988.852332	5118.00	7788.00	10295.00	16503.00	45400.00

Step6. Now, let us explore our data. We will look at how our data is distributed by plotting a histogram. We've plotted using both matplotlib and seaborn but both are the same while interpreting.

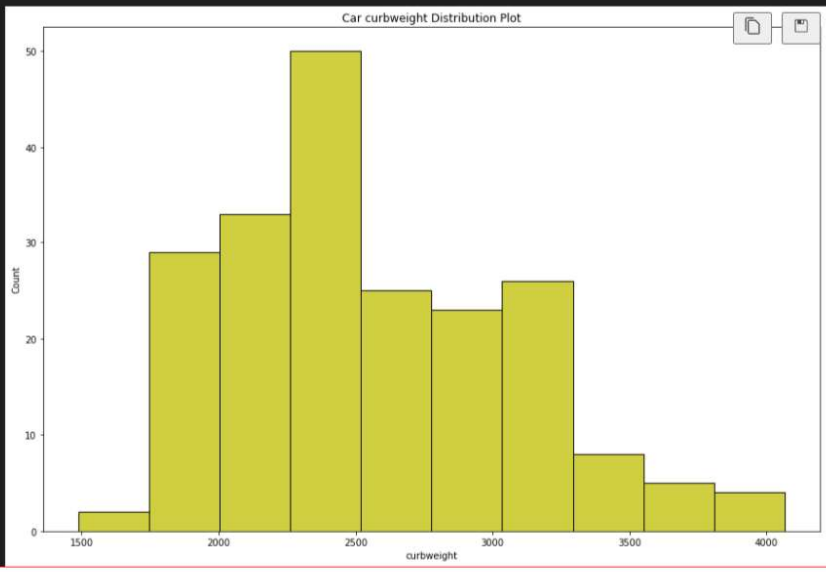
```
[12] plt.figure(figsize=(15,10))
     sns.histplot(data['horsepower'],color="y")
     plt.title('Car horsepower Distribution Plot')

     Text(0.5, 1.0, 'Car horsepower Distribution Plot')
```



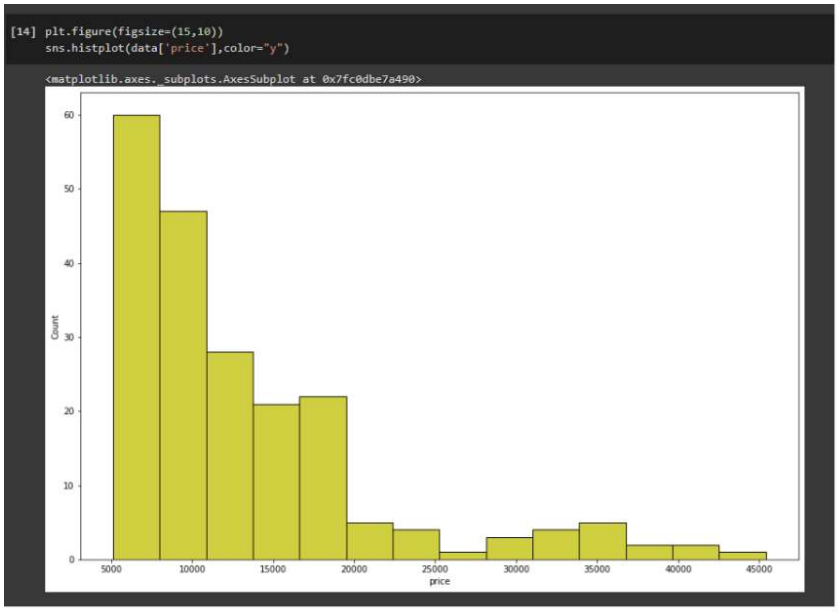

```
plt.figure(figsize=(15,10))
sns.histplot(data['curbweight'],color="y")
plt.title('Car curbweight Distribution Plot')
```

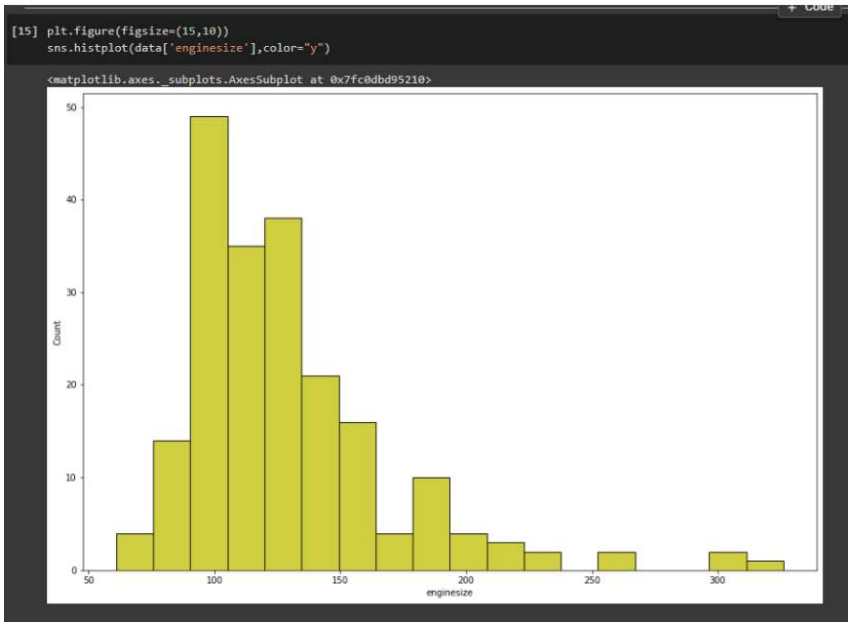
Python

```
Text(0.5, 1.0, 'Car curbweight Distribution Plot')
```



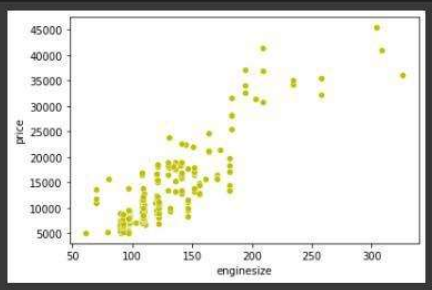
We can see that our data is skewed by looking at the above plots. What this means is that there are more cheaper cars in our dataset than expensive cars.



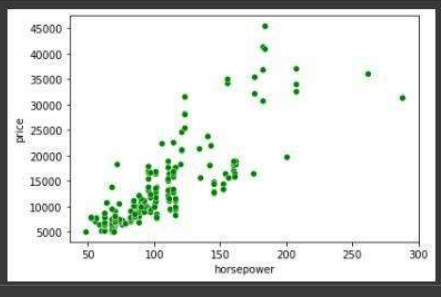


The plot below gives relation between enginesize and price.

```
[16] sns.scatterplot(x="enginesize", y="price", data=df,color='y');
```

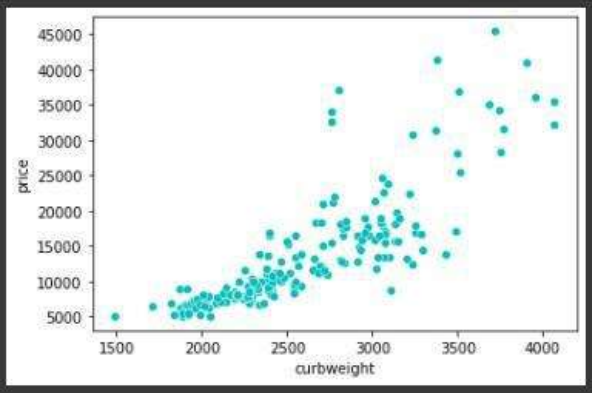



```
[17] sns.scatterplot(x="horsepower", y="price", data=df,color='g');
```



The plot below gives relation between curbweight and price.

```
[18] sns.scatterplot(x="curbweight", y="price", data=df,color='c');
```



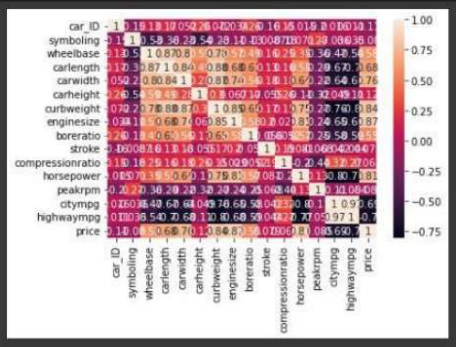
Step7. The Numerical Values having highest correlation are:
Engine Size, Curb Weight, Horsepower, Car Width, Car Length….
Let us drop all other variables

```
[19] df.corr()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg	price
car_ID	1.000000	-0.151621	0.129729	0.170636	0.052387	0.255960	0.071962	-0.033930	0.260064	-0.160824	0.150276	-0.015006	-0.203789	0.015940	0.011255	-0.109093
symboling	-0.151621	1.000000	-0.531954	-0.357612	-0.232919	-0.541038	-0.227691	-0.105790	-0.130051	-0.008735	-0.178515	0.070873	0.273606	-0.035823	0.034606	-0.079978
wheelbase	0.129729	-0.531954	1.000000	0.874587	0.795144	0.589435	0.776386	0.569329	0.488750	0.160959	0.249786	0.353294	-0.360469	-0.470414	-0.544082	0.577816
carlength	0.170636	-0.357612	0.874587	1.000000	0.841118	0.491029	0.877728	0.683360	0.606454	0.129533	0.158414	0.552623	-0.287242	-0.670909	-0.704662	0.682920
carwidth	0.052387	-0.232919	0.795144	0.841118	1.000000	0.279210	0.867032	0.735433	0.559150	0.182942	0.181129	0.640732	-0.220012	-0.642704	-0.677218	0.759325
carheight	0.255960	-0.541038	0.589435	0.491029	0.279210	1.000000	0.295572	0.067149	0.171071	-0.055307	0.261214	-0.108802	-0.320411	-0.048640	-0.107358	0.119336
curbweight	0.071962	-0.227691	0.776386	0.877728	0.867032	0.295572	1.000000	0.850594	0.648480	0.168790	0.151362	0.750739	-0.266243	-0.757414	-0.797465	0.835305
enginesize	-0.033930	-0.105790	0.569329	0.683360	0.735433	0.067149	0.850594	1.000000	0.583774	0.203129	0.028971	0.809769	-0.244660	-0.653658	-0.677470	0.874145
boreratio	0.260064	-0.130051	0.488750	0.606454	0.559150	0.171071	0.648480	0.583774	1.000000	-0.055909	0.005197	0.573677	-0.254976	-0.584532	-0.587012	0.553173
stroke	-0.160824	-0.008735	0.160959	0.129533	0.182942	-0.055307	0.168790	0.203129	-0.055909	1.000000	0.186110	0.080940	-0.067964	-0.042145	-0.043931	0.079443
compressionratio	0.150276	-0.178515	0.249786	0.158414	0.181129	0.261214	0.151362	0.028971	0.005197	0.186110	1.000000	-0.204326	-0.435741	0.324701	0.265201	0.067984
horsepower	-0.015006	0.070873	0.353294	0.552623	0.640732	-0.108802	0.750739	0.809769	0.573677	0.080940	-0.204326	1.000000	0.131073	-0.801456	-0.770544	0.808139
peakrpm	-0.203789	0.273606	-0.360469	-0.287242	-0.220012	-0.320411	-0.266243	-0.244660	-0.254976	-0.067964	-0.435741	0.131073	1.000000	-0.113544	-0.054275	-0.085267
citympg	0.015940	-0.035823	-0.470414	-0.670909	-0.642704	-0.048640	-0.757414	-0.653658	-0.584532	-0.042145	0.324701	-0.801456	-0.113544	1.000000	0.971337	-0.685751
highwaympg	0.011255	0.034606	-0.544082	-0.704662	-0.677218	-0.107358	-0.797465	-0.677470	-0.587012	-0.043931	0.265201	-0.770544	-0.054275	0.971337	1.000000	-0.697599
price	-0.109093	-0.079978	0.577816	0.682920	0.759325	0.119336	0.835305	0.874145	0.553173	0.079443	0.067984	0.808139	-0.085267	-0.685751	-0.697599	1.000000

Step8. Now, we need to select features using which we can model our price using linear regression. So, we will look at how much correlation each feature has with price. Correlation explains how much two things are related to each other. For example, the amount of rainfall is correlated with how wet your garden is. But, correlation doesn't always mean causation. Just because your garden is wet doesn't mean it was due to rain. It can also be the sprinkler or any other source of water. In general, correlation helps us the choose the most important variables to model after.

```
[20] sns.heatmap(df[num_col].corr(),annot = True);
```




```
[21] y = df.price
     print(y)

     0      13495.0
     1      16500.0
     2      16500.0
     3      13950.0
     4      17450.0
             ...
     200    16845.0
     201    19045.0
     202    21485.0
     203    22470.0
     204    22625.0
     Name: price, Length: 205, dtype: float64
```

We shall plot a boxplot to see how our price is distributed. Boxplot shows minimum, first quartile (25%), median, third quartile (75%), maximum and outliers (represented using dots).

```
[22] df_features = ['curbweight','enginesize','horsepower']

[23] X = df[df_features]

[24] X.describe()
```

	curbweight	enginesize	horsepower
count	205.000000	205.000000	205.000000
mean	2555.565854	126.907317	104.117073
std	520.680204	41.642693	39.544167
min	1488.000000	61.000000	48.000000
25%	2145.000000	97.000000	70.000000
50%	2414.000000	120.000000	95.000000
75%	2935.000000	141.000000	116.000000
max	4066.000000	326.000000	288.000000

```
[25] df_model = DecisionTreeRegressor(random_state = 1)

    # fit
    df_model.fit(X,y)

    DecisionTreeRegressor(random_state=1)

    print(X.head())
    print(df_model.predict(X.head()))

       curbweight  enginesize  horsepower
    0        2548         130         111
    1        2548         130         111
    2        2823         152         154
    3        2337         109         102
    4        2824         136         115
    [14997.5 14997.5 16500.  13950.  17450. ]
```

Learning outcome:

1: How to collect the data.
2: Fit a regression model to the data.
3: Verify that the model fits the data well.
4: Use the fitted regression equation to predict the values of new Observations

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			