Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Importing the dataset

```
df =
pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/MPG.cs
v')
```

Get Information of Dataframe

```
df.info()   #gives column name, count,  not null category, D-type(data
type)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64  
 7   origin        398 non-null    object 
 8   name          398 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB

df.describe()   #gives the linear relation of each column with another
column
                #if the mean of the columns not of same power ,we need
to standardize data

              mpg   cylinders  displacement  horsepower       
weight  \
count  398.000000  398.000000    398.000000  392.000000   398.000000   

mean    23.514573    5.454774    193.425879  104.469388  2970.424623   

std      7.815984    1.701004    104.269838   38.491160   846.841774   
```

min	9.000000	3.000000	68.000000	46.000000	1613.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000

	acceleration	model_year
count	398.000000	398.000000
mean	15.568090	76.010050
std	2.757689	3.697627
min	8.000000	70.000000
25%	13.825000	73.000000
50%	15.500000	76.000000
75%	17.175000	79.000000
max	24.800000	82.000000

```
df.head(13)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	\
0	18.0	8	307.0	130.0	3504	12.0	
1	15.0	8	350.0	165.0	3693	11.5	
2	18.0	8	318.0	150.0	3436	11.0	
3	16.0	8	304.0	150.0	3433	12.0	
4	17.0	8	302.0	140.0	3449	10.5	
5	15.0	8	429.0	198.0	4341	10.0	
6	14.0	8	454.0	220.0	4354	9.0	
7	14.0	8	440.0	215.0	4312	8.5	
8	14.0	8	455.0	225.0	4425	10.0	
9	15.0	8	390.0	190.0	3850	8.5	
10	15.0	8	383.0	170.0	3563	10.0	
11	14.0	8	340.0	160.0	3609	8.0	
12	15.0	8	400.0	150.0	3761	9.5	

	model_year	origin	name
0	70	usa	chevrolet chevelle malibu
1	70	usa	buick skylark 320
2	70	usa	plymouth satellite
3	70	usa	amc rebel sst
4	70	usa	ford torino
5	70	usa	ford galaxie 500
6	70	usa	chevrolet impala
7	70	usa	plymouth fury iii
8	70	usa	pontiac catalina
9	70	usa	amc ambassador dpl
10	70	usa	dodge challenger se

```
11          70    usa         plymouth 'cuda 340
12          70    usa      chevrolet monte carlo
```

```
df.isnull().sum() #(df.isna().sum() gives same result)
#gives the sum of all null values columns-wise
```

```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
name            0
dtype: int64
```

```
df.nunique()   #gives total no. of unique entries
```

```
mpg             129
cylinders         5
displacement     82
horsepower       93
weight          351
acceleration     95
model_year       13
origin            3
name            305
dtype: int64
```

```
df.columns  #give column names in the dataframe
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'name'],
      dtype='object')
```

```
df.shape
```

```
(392, 9)
```

```
df.corr()       #as mpg cylinders and displacement have very high
correlation we will use only one of those
                #in this case we are using displacement
```

```
                   mpg  cylinders  displacement  horsepower    weight
\
mpg           1.000000  -0.777618     -0.805127   -0.778427 -0.832244

cylinders    -0.777618   1.000000      0.950823    0.842983  0.897527

displacement -0.805127   0.950823      1.000000    0.897257  0.932994
```

```
horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538

weight       -0.832244   0.897527      0.932994    0.864538  1.000000

acceleration  0.423329  -0.504683     -0.543800   -0.689196 -0.416839

model_year    0.580541  -0.345647     -0.369855   -0.416361 -0.309120


              acceleration  model_year
mpg               0.423329    0.580541
cylinders        -0.504683   -0.345647
displacement     -0.543800   -0.369855
horsepower       -0.689196   -0.416361
weight           -0.416839   -0.309120
acceleration      1.000000    0.290316
model_year        0.290316    1.000000
```

#Remove Missing Values

```
df=df.dropna()
```

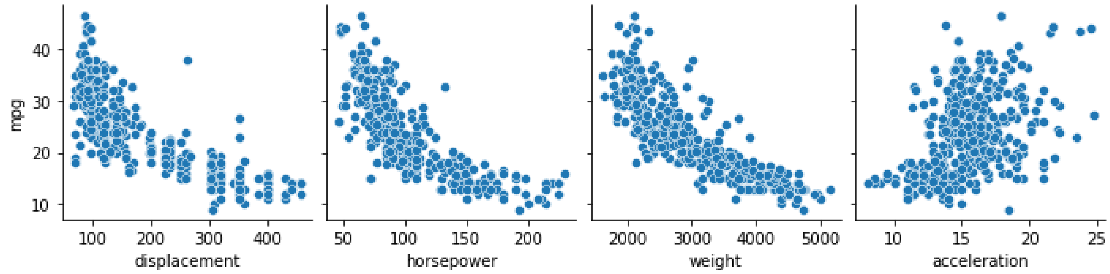
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64
 5   acceleration  392 non-null    float64
 6   model_year    392 non-null    int64
 7   origin        392 non-null    object
 8   name          392 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 30.6+ KB
```

#Data Visualization

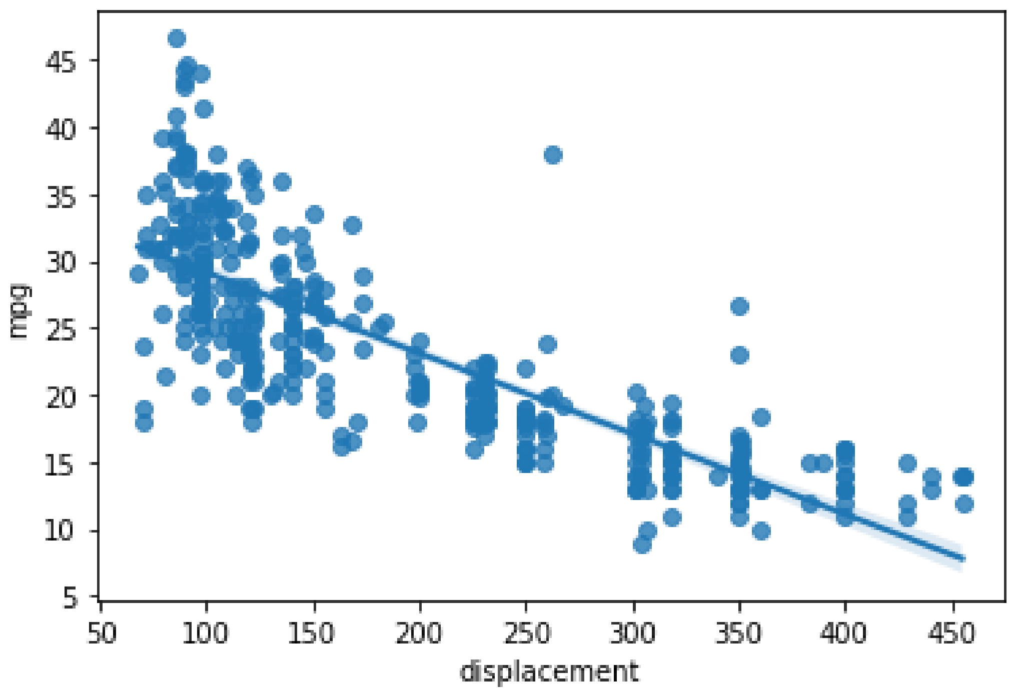
```
sns.pairplot(df,x_vars=['displacement','horsepower','weight','accelera
tion'],y_vars='mpg')
```

```
<seaborn.axisgrid.PairGrid at 0x7fef3f9fc850>
```




```
sns.regplot(x='displacement', y='mpg',data=df)    #regression line
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fef3d0464d0>



#Define Target Variable y and Feature X

```
df.columns
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'name'],
      dtype='object')
```

```
y=df['mpg']
```

```
y.shape
```

```
(392,)
```

```
X=df[['displacement','horsepower','weight','acceleration']]
```

```
X.shape
```

```
(392, 4)
```

#Scaling Data

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X= ss.fit_transform(X)
```

```
X
```

```
array([[ 1.07728956,  0.66413273,  0.62054034, -1.285258  ],
       [ 1.48873169,  1.57459447,  0.84333403, -1.46672362],
       [ 1.1825422 ,  1.18439658,  0.54038176, -1.64818924],
       ...,
       [-0.56847897, -0.53247413, -0.80463202, -1.4304305 ],
       [-0.7120053 , -0.66254009, -0.41562716,  1.11008813],
       [-0.72157372, -0.58450051, -0.30364091,  1.40043312]])
```

```
pd.DataFrame(X).describe()
```

```
                  0             1             2             3
count  3.920000e+02  3.920000e+02  3.920000e+02  3.920000e+02
mean  -2.537653e-16 -4.392745e-16  5.607759e-17  6.117555e-16
std    1.001278e+00  1.001278e+00  1.001278e+00  1.001278e+00
min   -1.209563e+00 -1.520975e+00 -1.608575e+00 -2.736983e+00
25%   -8.555316e-01 -7.665929e-01 -8.868535e-01 -6.410551e-01
50%   -4.153842e-01 -2.853488e-01 -2.052109e-01 -1.499869e-02
75%    7.782764e-01  5.600800e-01  7.510927e-01  5.384714e-01
max    2.493416e+00  3.265452e+00  2.549061e+00  3.360262e+00
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 1)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((274, 4), (118, 4), (274,), (118,))
```

LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
```

```
LinearRegression()
```

```
lr.intercept_
```

```
23.19931701979532
```

```
lr.coef_
```

```
array([-0.71568189, -1.1167925 , -4.51963105,  0.41172277])
```

mpg= 23.2 +(-0.71displacement) + (-1.11horsepower) + (-4.51weight) + (-0.41acceleration)

Model Prediction

```
y_pred=lr.predict(X_test)
```

```
y_pred.shape
```

```
(118,)
```

```
y_pred
```

```
array([26.28276367, 29.08833366, 28.17888013, 24.06092615,
30.32940196,
       29.34559658, 29.47244419, 14.16480917, 28.12054088,
28.62243268,
       21.19699253, 29.00202854, 17.08446147, 30.6393795 ,
26.68062515,
       18.16545075, 27.05814569, 30.19749225,  9.75469894,
23.47905709,
       28.25978712, 19.48402464, 18.1064714 , 13.28502247,
10.78371487,
       15.23539746, 30.32970958, 19.85446483, 22.37999549,
26.96328669,
       18.53096933, 25.30938579, 12.23913664, 23.35707834,
20.75265097,
       14.43741584, 19.4800578 , 20.51028037, 30.30992209,
28.70496355,
       11.62869958, 11.71491907, 23.55687348, 23.54831788,
23.99559915,
       19.11133191,  9.47947187, 30.64274232, 21.73778331,
8.26480713,
       16.82257963, 22.88819913, 25.65631731, 27.24852928,
31.15303313,
       23.38685254, 24.44456186, 25.87613663, 25.81319389,
31.60155856,
       23.32071279, 27.06454436, 31.01331394, 20.51927114,
23.09760801,
       22.33665492, 23.57958269, 16.09819394, 30.27109569,
8.5745448 ,
       26.89318004, 18.68219122, 16.65807253, 28.88072244,
26.96138898,
```

```
       30.06105048, 16.507612  , 12.02031005, 12.93921033,
26.82226838,
       19.87853884, 28.6680564 , 30.32376371, 28.69240544,
14.84426249,
       20.49630205, 30.74330121, 25.97075053, 31.06972194,
23.9671303 ,
       28.82409624, 21.11682718, 14.60616423, 11.49977764,
7.75259844,
       24.12423109, 14.48534835, 24.39258351, 23.40743093,
13.20487542,
       28.58355011, 26.42326644, 23.31389675, 28.91626441,
28.95276995,
       29.56630042, 23.43332526, 31.38563943, 14.14444937,
18.34692392,
       30.74367965, 23.9993707 , 30.32374274, 27.61841915,
15.3921527 ,
       25.62422581, 20.97391756, 12.8109788 ])
```

Model Evaluation

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
mean_absolute_percentage_error, r2_score

mean_absolute_error(y_test,y_pred)

3.1259768036439923

mean_absolute_percentage_error(y_test,y_pred)

0.13268990973331238

r2_score(y_test,y_pred)         #regressor.score(x_test,y_test)   ---
gives same result

0.7209702318140772
```

#Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures
poly=PolynomialFeatures(degree=2,
interaction_only=True,include_bias=False)

X_train2=poly.fit_transform(X_train)

X_test2=poly.fit_transform(X_test)

lr.fit(X_train2,y_train)

LinearRegression()

lr.intercept_

20.98947080512586
```

```
lr.coef_
```

```
array([-2.90070836, -4.57930091, -1.18400444, -0.48222393,
2.10945694,
        0.59137383, -0.46726197, -0.32012722, -0.02876052,
0.56719103])
```

```
y_pred_poly=lr.predict(X_test2)
```

```
mean_absolute_error(y_test,y_pred_poly)
```

```
2.8067976869672058
```

```
mean_absolute_percentage_error(y_test,y_pred_poly)
```

```
0.11485347284477058
```

```
r2_score(y_test,y_pred_poly)
```

```
0.7657248979487836
```