In [1]:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [3]:

```
import matplotlib.pyplot as plt
```

In [4]:

```
import seaborn as sns
```

import data

In [5]:

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

Type *Markdown* and LaTeX: α^2

In [6]:

```
df.head()
```

Out[6]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	nam
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrole chevell malib
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buic skylar 32
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymout satellit
3	16.0	8	304.0	150.0	3433	12.0	70	usa	am rebel s:
4	17.0	8	302.0	140.0	3449	10.5	70	usa	for torin

In [7]:

```
df.nunique()
```

Out[7]:

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

DATA PREPROCESSING

In [11]:

```
df.info()
```

```
<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
```

In [12]:

```
df.describe()
```

Out[12]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_yea
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.00000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.01005
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.69762
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.00000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.00000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.00000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.00000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.00000

In [13]:

```
df.corr()
```

Out[13]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_y
mpg	1.000000	-0.775396	-0.804203	-0.778427	-0.831741	0.420289	0.579
cylinders	-0.775396	1.000000	0.950721	0.842983	0.896017	-0.505419	-0.348
displacement	-0.804203	0.950721	1.000000	0.897257	0.932824	-0.543684	-0.370
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416
weight	-0.831741	0.896017	0.932824	0.864538	1.000000	-0.417457	-0.306
acceleration	0.420289	-0.505419	-0.543684	-0.689196	-0.417457	1.000000	0.288
model_year	0.579267	-0.348746	-0.370164	-0.416361	-0.306564	0.288137	1.000

REMOVE MISSING VALUES

In [14]:

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

In [15]:

```
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
```

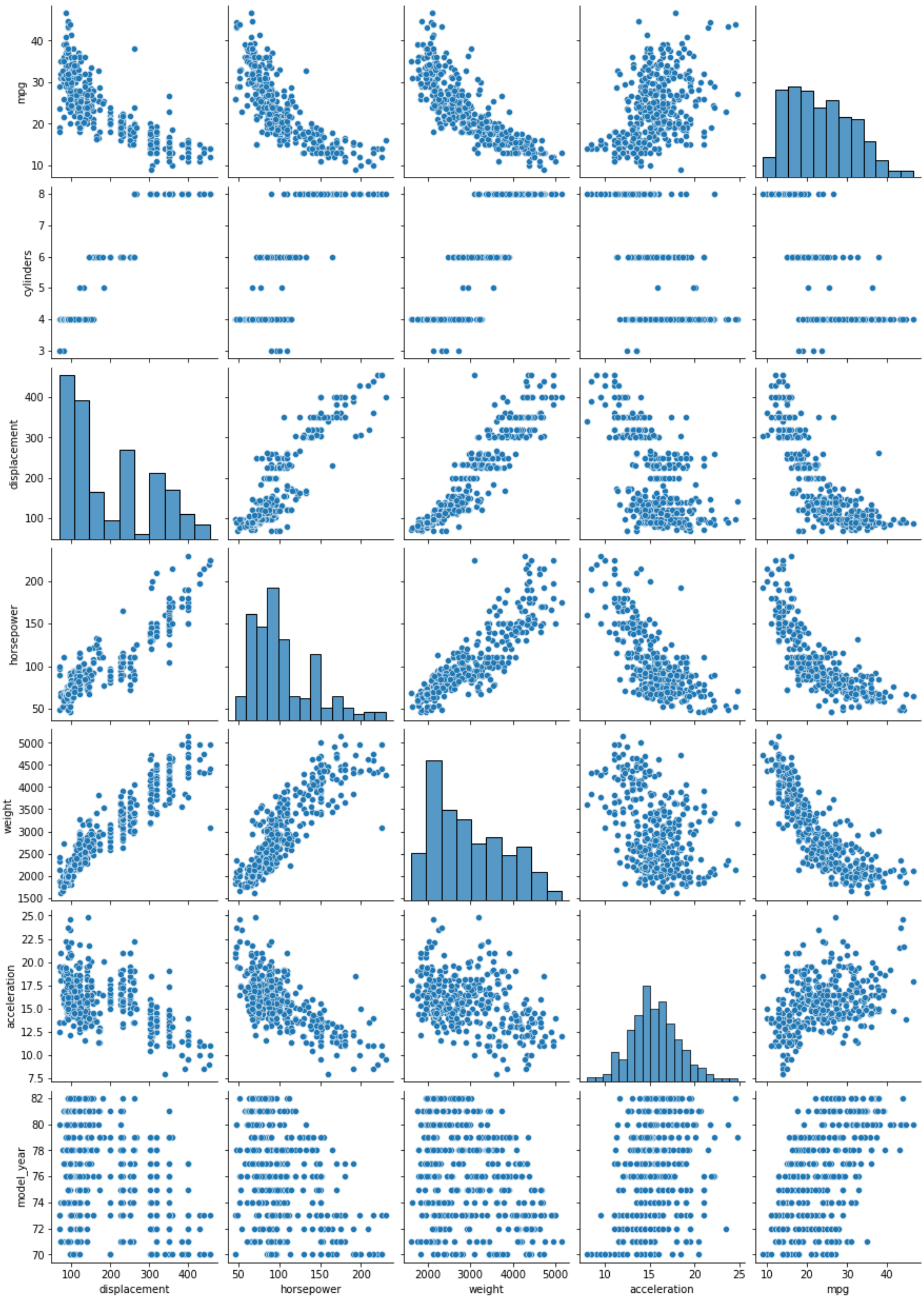
DATA VISUALIZATION

In [16]:

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

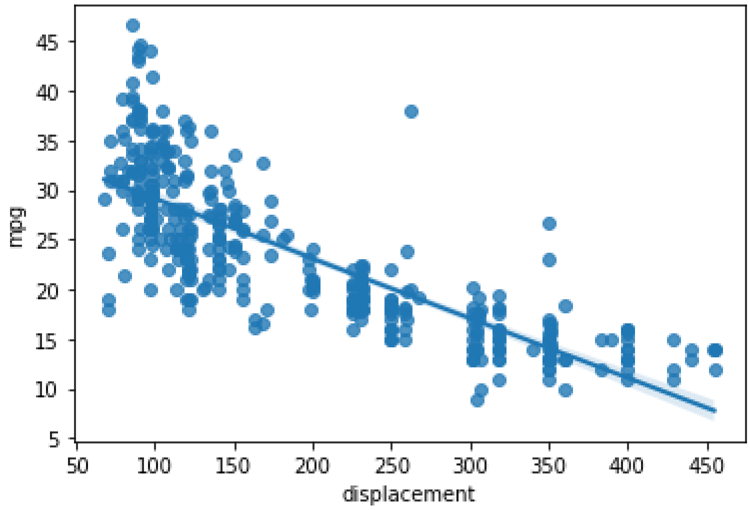
Out[16]:

```
<seaborn.axisgrid.PairGrid at 0x260d8072d08>
```



In [17]:

```
sns.regplot(x='displacement', y= 'mpg', data =df);
```



Define target variable y and feature x

In [18]:

```
df.columns
```

Out[18]:

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

In [19]:

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

In [20]:

```
y.shape
```

Out[20]:

```
(392,)
```

In [21]:

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

In [22]:

```
x.shape
```

Out[22]:

```
(392, 4)
```

Scaling data

In [23]:

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

In [24]:

```
ss=StandardScaler()
```

In [25]:

```
x=ss.fit_transform(x)
```

Train Teat Split Data

In [26]:

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

In [27]:

```
x_train, x_test, y_train, y_test=train_test_split(x,y,train_size=0.7,random_state=2525)
```

In [28]:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[28]:

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

Linear refgression model

In [29]:

```
from sklearn.linear_model import LinearRegression
```

In [30]:

```
lr= LinearRegression()
```

In [31]:

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

Out[31]:

```
LinearRegression()
```

In [32]:

```
lr.intercept_
```

Out[32]:

```
23.688921610685803
```

In [33]:

```
lr.coef_
```

Out[33]:

```
array([-0.13510042, -1.4297211 , -5.23891463,  0.22436094])
```

Predict test data

In [34]:

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

In [35]:

```
y_pred
```

Out[35]:

```
array([25.24954801, 26.85525431, 26.58882904, 29.48052754, 23.91216916,
       14.9529791 , 30.0607685 , 34.07634195, 30.550342  , 11.31024173,
       18.14067535, 18.75305197, 29.80678264, 33.19954312, 17.23635872,
       16.06983768, 25.94812038, 21.15777548, 29.92508087, 25.05587641,
       22.85575427, 30.96630956, 22.82202336, 24.04513247, 25.95102384,
       26.21136844, 14.91805111, 31.85928917, 21.95227216, 26.85446824,
        8.94214825, 26.21244694, 30.20552304,  7.15733458, 26.31771126,
       30.54356872, 14.13603243, 31.02810818, 33.19140036, 31.74995879,
       11.07428823, 30.50398808, 29.36195486, 31.022648  , 23.53384962,
       22.87821543, 11.03531446, 14.3757476 , 31.44484893, 26.64255441,
       27.96470623, 21.80486111, 20.32272978, 31.27632871, 24.83127389,
       19.13391479, 28.2786737 , 25.21468804, 26.89045676, 28.76603057,
       19.03600671, 29.49310219, 28.42147856, 26.6112997 ,  7.384747  ,
       20.13152225, 22.77931428, 20.50765035, 32.81875326, 27.92430623,
       13.34341223,  8.03767139, 25.34229398, 17.23635872, 33.03710336,
       31.07878627, 21.58700058, 24.53266643, 30.38829664, 17.84737111,
       31.30622407, 30.1021144 , 22.81248978, 20.01904445,  9.12644754,
       24.50457451, 29.57695629, 29.45235437, 31.59169567, 26.49442535,
       30.32795983, 12.36145993, 16.48933189, 15.27329229, 32.77989962,
       27.25863029, 11.07878871, 25.72147567, 12.57968624, 30.4363069 ,
       27.56306784, 24.92600083, 16.21791725, 23.89776551, 18.63499966,
       10.21748386, 21.60970196, 23.01257072, 27.30850629, 30.45961552,
       29.43254102, 27.21176721, 24.2365775 , 28.87030773, 21.16703179,
       27.97152628, 24.54560958, 32.23487944])
```

```
# Model Accuracy
```

In [38]:

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

In [39]:

```
mean_absolute_percentage_error(y_test,y_pred)
```

Out[39]:

```
0.16282215595698374
```

In [40]:

```
r2_score(y_test, y_pred)
```

Out[40]:

```
0.6767436309121444
```

Polynomial Regression

In [41]:

```
from sklearn.preprocessing import PolynomialFeatures
```

In [42]:

```
poly= PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
```

In [43]:

```
X_train2= poly.fit_transform(x_train)
```

In [44]:

```
x_test2= poly.fit_transform(x_test)
```

In [47]:

```
lr.fit(X_train2,y_train)
```

Out[47]:

```
LinearRegression()
```

In [48]:

```
lr.intercept_
```

Out[48]:

```
21.457120355191677
```

In [49]:

```
lr.coef_
```

Out[49]:

```
array([-1.97594907e+00, -5.50639326e+00, -1.82341405e+00, -8.04049934e-01,
        1.55534517e+00, -4.40583099e-01, -5.33735335e-01,  1.29466895e+00,
        2.61553723e-03,  5.86761939e-01])
```

In [50]:

```
y_pred_poly = lr.predict(x_test2)
```

Model Accuracy

In [51]:

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

In [52]:

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

Out[52]:

```
2.924007242447458
```

In [55]:

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

Out[55]:

```
0.12874881331071994
```

In [56]:

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

Out[56]:

```
0.7198303534964864
```

In [ ]:

```

```