Mini Project 8: Predictive Analytics

Step 1: Import library

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt
```

Step 2: Import Data

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

df
```

Saved successfully!

			ment	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.0	2790	15.6	82	usa	ford mustang gl
394	44.0	4	97.0	52.0	2130	24.6	82	europe	vw pickup

```
df.nunique()
```

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

Data Preprocessing

```
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
```

```
df.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

```
df.corr()
```

	ers	displacement	horsepower	weight	acceleration	model_year
mpg	1.000000	-0.775396	-0.804203	-0.778427	-0.831741	0.420289	0.579267
cylinders	-0.775396	1.000000	0.950721	0.842983	0.896017	-0.505419	-0.348746
displacement	-0.804203	0.950721	1.000000	0.897257	0.932824	-0.543684	-0.370164
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361
weight	-0.831741	0.896017	0.932824	0.864538	1.000000	-0.417457	-0.306564

```
df.shape
```

```
(398, 9)
```

```
df.columns
```

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

Auxillary Step: Remove Missing Value Rows

```
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
                                   int64
```

Saved successfully!

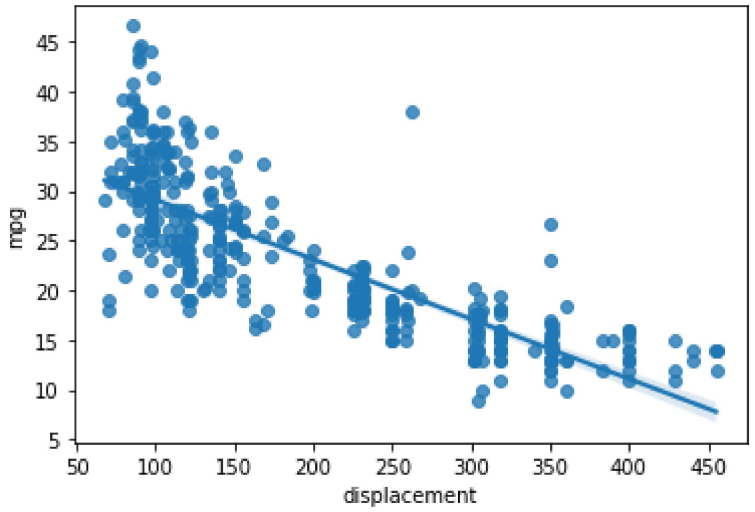
```
                                    float64
                                    float64
 4   weight        392 non-null     int64
 5   acceleration  392 non-null     float64
 6   model_year    392 non-null     int64
 7   origin        392 non-null     object
 8   name          392 non-null     object
dtypes: float64(4), int64(3), object(2)
memory usage: 30.6+ KB
```

Data Visualisation

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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f90a89cc090>
```



Step 3: Define y and X

Saved successfully!

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

```
y.shape
```

```
(392,)
```

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

```
X.shape
```

```
(392, 4)
```

Auxillary Step: Data Scaling

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

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

	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
75%	7.782764e-01	5.600800e-01	7.510927e-01	5.384714e-01

Step 4: Splitting Data

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7,random_state=2529)

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

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

Step 5: Create a Model

```
from sklearn.linear_model import LinearRegression

model=LinearRegression()
```

Step 6: Train a Model

```
model.fit(X_train,y_train)
```

Saved successfully!

```
model.intercept_
```

```
23.485738559737584
```

```
model.coef_
```

```
array([-1.05767743, -1.68734727, -4.10787617, -0.11495177])
```

Step 7: Predict Model

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

```
y_pred
```

```
array([18.51865637, 15.09305675, 14.30128789, 23.6753321 , 29.7546115 ,
       23.68796629, 26.61066644, 24.56692437, 15.06260986, 11.94312046,
       24.08050053, 27.96518468, 31.66130278, 31.01309132, 18.32428976,
       19.32795009, 28.08847536, 32.1506879 , 31.15859692, 27.15792144,
       18.82433097, 22.54580176, 26.15598115, 32.36393869, 20.74377679,
        8.78027518, 22.19699435, 18.20614294, 25.00052718, 15.26421552,
       23.13441082, 17.10542257,  9.87180062, 30.00790415, 20.41204655,
       29.11860245, 24.4305187 , 21.72601835, 10.51174626, 13.12426391,
       21.41938406, 19.96113872,  6.19146626, 17.79025345, 22.5493033 ,
       29.34765021, 13.4861847 , 25.88852083, 29.40406946, 22.41841964,
       22.07684766, 16.46575802, 24.06290693, 30.12890046, 10.11318121,
        9.85011438, 28.07543852, 23.41426617, 20.08501128, 30.68234133,
       20.92026393, 26.78370281, 22.9078744 , 14.15936872, 24.6439883 ,
       26.95515832, 15.25709393, 24.11272087, 30.80980589, 14.9770217 ,
       27.67836372, 24.2372919 , 10.92177228, 30.22858779, 30.88687365,
       27.33992044, 31.18447082, 10.8873597 , 27.63510608, 16.49231363,
       25.63229888, 29.49776285, 14.90393439, 32.78670687, 30.37325244,
       30.9262743 , 14.71702373, 27.09633246, 26.69933806, 29.06424799,
       32.45810182, 29.44846898, 31.61239999, 31.57891837, 21.46542321,
```

Saved successfully!

```
, 28.96419915, 31.09628395, 24.80549594,
, 23.04466919, 22.14143162, 15.95854367,
28.62870918, 25.58809869, 11.4040908 , 25.73334842, 30.83500051,
21.94176255, 15.34532941, 30.37399213, 28.7620624 , 29.3639931 ,
29.10476703, 20.44662365, 28.11466839])
```

Step 8: Accuracy

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

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

```
0.14713035779536746
```

```
mean_absolute_error(y_test,y_pred)
```

```
3.3286968643244106
```

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

```
0.7031250746717692
```

Polynomial Regression

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

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

Saved successfully!

```
X_test1=p.fit_transform(X_test)
```

```
model.fit(X_train1,y_train)
```

```
LinearRegression()
```

```
model.intercept_
```

```
21.27336450063766
```

```
model.coef_
```

```
array([-2.76070596, -5.00559628, -1.36884133, -0.81225214,  1.24596571,
       -0.12475017, -0.90542822,  1.35064048, -0.17337823,  1.41680398])
```

```
y_pred_p=model.predict(X_test1)
```

Accuracy

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

```
mean_absolute_percentage_error(y_test,y_pred_p)
```

```
0.1207401834293869
```

```
mean_absolute_error(y_test,y_pred_p)
```

```
2.7887147720295977
```

Saved successfully!

```
0.7461731314563803
```

Link of the same:

https://colab.research.google.com/drive/1J2SCi7MfIqCZxEZPoRpyh6KIf5EtKbsE?usp=sharing