Import Library

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 [7]:

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

In [8]:

```
df.head()
```

Out[8]:

	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 [9]:

```
df.nunique()
```

Out[9]:

```
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 [13]:

```
df.describe
```

Out[13]:

```
<bound method NDFrame.describe of       mpg  cylinders  displacement  horsep
ower  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
..    ...        ...           ...         ...     ...           ...
393  27.0          4         140.0        86.0    2790          15.6
394  44.0          4          97.0        52.0    2130          24.6
395  32.0          4         135.0        84.0    2295          11.6
396  28.0          4         120.0        79.0    2625          18.6
397  31.0          4         119.0        82.0    2720          19.4

     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
..          ...     ...                        ...
393          82     usa            ford mustang gl
394          82  europe                  vw pickup
395          82     usa              dodge rampage
396          82     usa                ford ranger
397          82     usa                 chevy s-10

[398 rows x 9 columns]>
```

In [15]:

```
df.corr()
```

Out[15]:

	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 [17]:

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

In [19]:

```
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
```

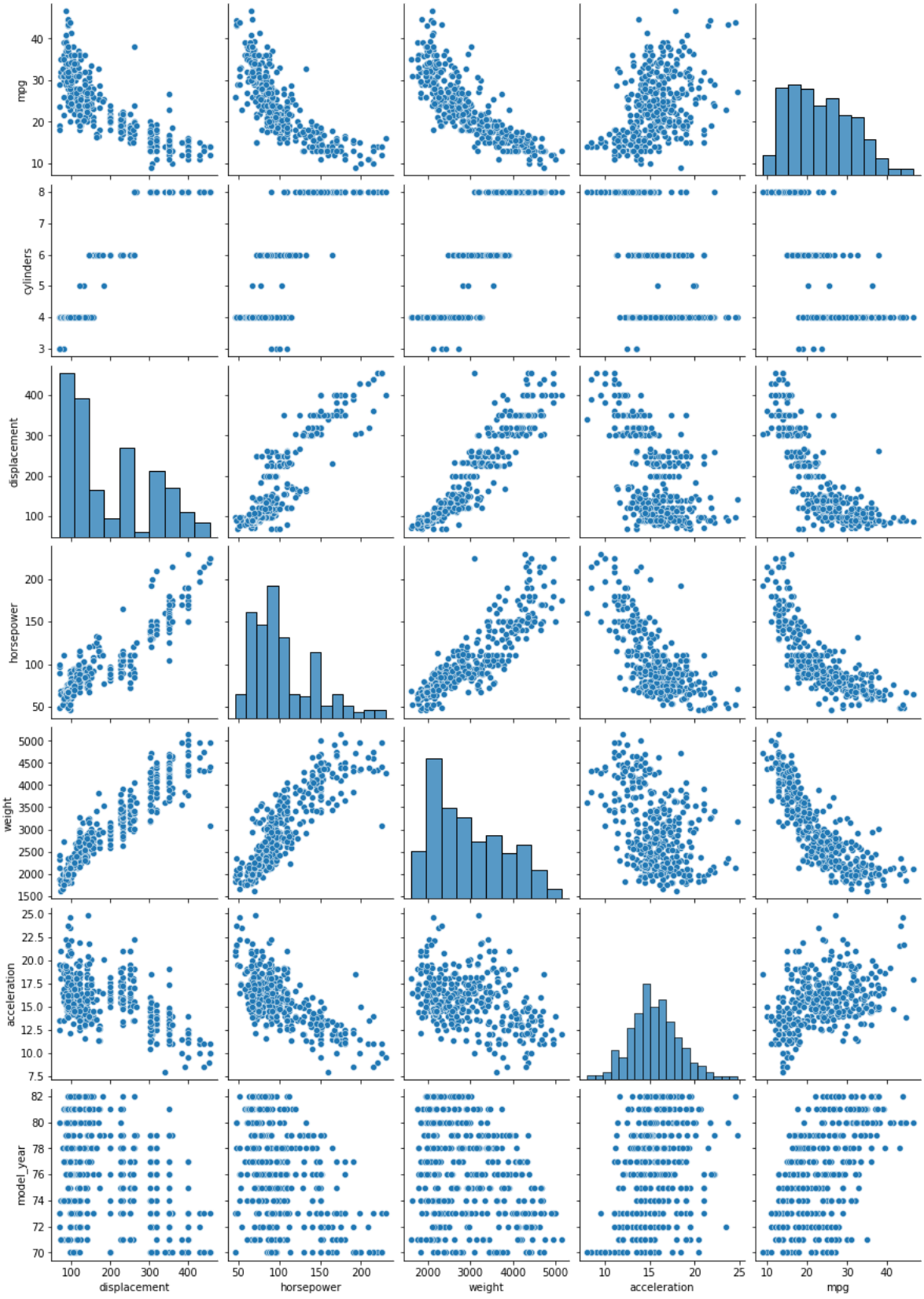
Data Visualization

In [23]:

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

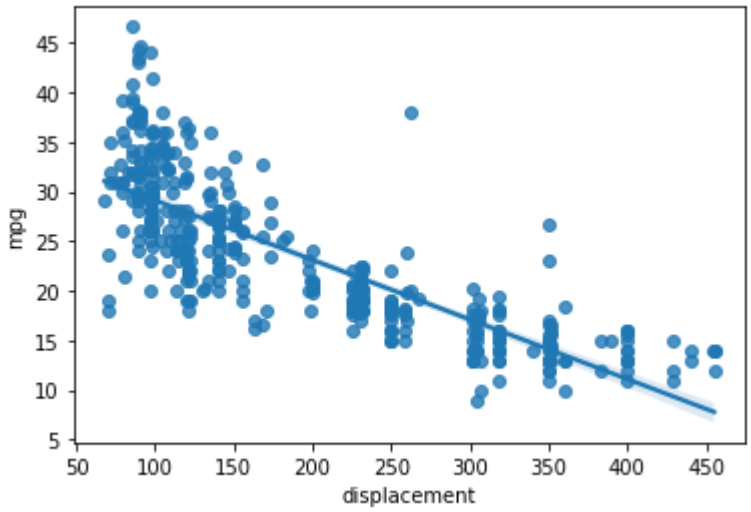
Out[23]:

```
<seaborn.axisgrid.PairGrid at 0x7f1d93e41450>
```



In [25]:

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



Define Target variable y and feature x

In [26]:

```
df.columns
```

Out[26]:

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

In [28]:

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

In [29]:

```
y.shape
```

Out[29]:

```
(392,)
```

In [67]:

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

In [68]:

```
X.shape
```

Out[68]:

```
(392, 4)
```

In [69]:

```
X
```

Out[69]:

	displacement	horsepower	weight	acceleration
0	307.0	130.0	3504	12.0
1	350.0	165.0	3693	11.5
2	318.0	150.0	3436	11.0
3	304.0	150.0	3433	12.0
4	302.0	140.0	3449	10.5
...	...	...	...	...
393	140.0	86.0	2790	15.6
394	97.0	52.0	2130	24.6
395	135.0	84.0	2295	11.6
396	120.0	79.0	2625	18.6
397	119.0	82.0	2720	19.4

392 rows × 4 columns

In [70]:

```
X.describe()
```

Out[70]:

	displacement	horsepower	weight	acceleration
count	392.000000	392.000000	392.000000	392.000000
mean	194.411990	104.469388	2977.584184	15.541327
std	104.644004	38.491160	849.402560	2.758864
min	68.000000	46.000000	1613.000000	8.000000
25%	105.000000	75.000000	2225.250000	13.775000
50%	151.000000	93.500000	2803.500000	15.500000
75%	275.750000	126.000000	3614.750000	17.025000
max	455.000000	230.000000	5140.000000	24.800000

Scaling Data

In [35]:

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

In [36]:

```
ss=StandardScaler()
```

In [38]:

```
X=ss.fit_transform(X)
```

In [40]:

```
X
```

Out[40]:

```
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]])
```

In [41]:

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

Out[41]:

	0	1	2	3
count	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02
mean	-1.393443e-16	-3.293850e-16	5.607759e-17	1.608691e-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

Train test split data

In [43]:

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

In [45]:

```
X_train, X_test, y_train, y_test=train_test_split(X,y, train_size=0.7, random_state=2525)
```

In [49]:

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

Out[49]:

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

Linear Regression Model

In [50]:

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

In [52]:

```
lr=LinearRegression()
```

In [53]:

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

Out[53]:

```
LinearRegression()
```

In [54]:

```
lr.intercept_
```

Out[54]:

```
23.688921610685803
```

In [56]:

```
lr.coef_
```

Out[56]:

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

Predict test data

In [57]:

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

In [59]:

```
y_pred
```

Out[59]:

```
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 [71]:

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

In [72]:

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

Out[72]:

```
3.417654680078564
```

In [73]:

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

Out[73]:

```
0.16282215595698368
```

In [74]:

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

Out[74]:

```
0.6767436309121445
```

Polynomial Regression

In [75]:

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

In [78]:

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

In [79]:

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

In [80]:

```
X_test2=poly.fit_transform(X_test)
```

In [82]:

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

Out[82]:

```
LinearRegression()
```

In [84]:

```
lr.intercept_
```

Out[84]:

```
21.457120355191684
```

In [85]:

```
lr.coef_
```

Out[85]:

```
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 [87]:

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

Model accuracy

In [89]:

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

In [90]:

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

Out[90]:

```
2.924007242447459
```

In [92]:

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

Out[92]:

```
0.12874881331071994
```

In [93]:

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

Out[93]:

```
0.7198303534964864
```

Hand Written Digit Prediction - Classification Analysis

Import Library

In [94]:

```
import pandas as pd
```

In [95]:

```
import numpy as np
```

In [98]:

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

Import Data

In [99]:

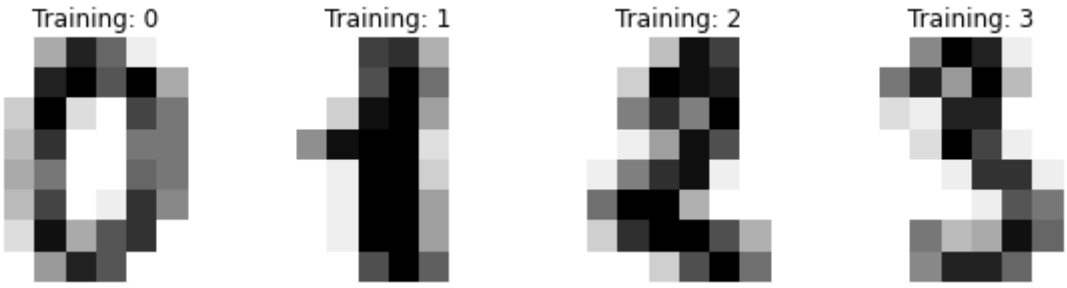
```
from sklearn.datasets import load_digits
```

In [100]:

```
df=load_digits()
```

In [102]:

```
_, axes=plt.subplots(nrows=1, ncols=4, figsize=(10,3))
for ax, image, label in zip(axes,df.images, df.target):
  ax.set_axis_off()
  ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
  ax.set_title("Training: %i" % label)
```



Data Preprocessing

In [103]:

```
df.images.shape
```

Out[103]:

```
(1797, 8, 8)
```

In [104]:

```
df.images[0]
```

Out[104]:

```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

In [105]:

```
df.images[0].shape
```

Out[105]:

```
(8, 8)
```

In [107]:

```
n_samples=len(df.images)
data=df.images.reshape((n_samples, -1))
```

In [109]:

```
data[0]
```

Out[109]:

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

In [110]:

```
data[0].shape
```

Out[110]:

```
(64,)
```

In [112]:

```
data.shape
```

Out[112]:

```
(1797, 64)
```

Scaling Data

In [113]:

```
data.min()
```

Out[113]:

```
0.0
```

In [114]:

```
data.max()
```

Out[114]:

```
16.0
```

In [116]:

```
data=data/16
```

In [117]:

```
data.min()
```

Out[117]:

```
0.0
```

In [119]:

```
data.max()
```

Out[119]:

```
0.0625
```

In [121]:

```
data[0]
```

Out[121]:

```
array([0.        , 0.        , 0.01953125, 0.05078125, 0.03515625,
       0.00390625, 0.        , 0.        , 0.        , 0.        ,
       0.05078125, 0.05859375, 0.0390625 , 0.05859375, 0.01953125,
       0.        , 0.        , 0.01171875, 0.05859375, 0.0078125 ,
       0.        , 0.04296875, 0.03125   , 0.        , 0.        ,
       0.015625  , 0.046875  , 0.        , 0.        , 0.03125   ,
       0.03125   , 0.        , 0.        , 0.01953125, 0.03125   ,
       0.        , 0.        , 0.03515625, 0.03125   , 0.        ,
       0.        , 0.015625  , 0.04296875, 0.        , 0.00390625,
       0.046875  , 0.02734375, 0.        , 0.        , 0.0078125 ,
       0.0546875 , 0.01953125, 0.0390625 , 0.046875  , 0.        ,
       0.        , 0.        , 0.        , 0.0234375 , 0.05078125,
       0.0390625 , 0.        , 0.        , 0.        ])
```

Train test split data

In [122]:

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

In [127]:

```
X_train, X_test, y_train, y_test=train_test_split(data, df.target, test_size=0.3)
```

In [128]:

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

Out[128]:

```
((1257, 64), (540, 64), (1257,), (540,))
```

Random Forest Model

In [130]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [132]:

```
rf=RandomForestClassifier()
```

In [133]:

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

Out[133]:

```
RandomForestClassifier()
```

Predict Test Data

In [134]:

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

In [135]:

```
y_pred
```

Out[135]:

```
array([7, 4, 1, 3, 1, 5, 9, 3, 4, 7, 6, 4, 4, 8, 8, 2, 6, 0, 3, 7, 2, 8,
       0, 5, 9, 6, 5, 1, 2, 0, 0, 5, 3, 7, 0, 2, 4, 7, 4, 3, 0, 1, 8, 2,
       2, 7, 9, 9, 0, 6, 1, 4, 6, 3, 8, 8, 3, 3, 8, 9, 1, 3, 6, 1, 0, 1,
       0, 2, 6, 6, 5, 2, 5, 7, 9, 9, 9, 4, 2, 3, 3, 5, 6, 5, 7, 9, 3, 4,
       1, 7, 4, 3, 5, 8, 1, 0, 8, 2, 6, 1, 2, 6, 6, 1, 3, 6, 7, 4, 9, 1,
       9, 9, 0, 1, 5, 8, 9, 2, 8, 8, 3, 8, 2, 8, 9, 1, 6, 0, 2, 3, 1, 9,
       6, 3, 2, 8, 4, 2, 4, 7, 4, 0, 3, 2, 5, 5, 6, 1, 8, 3, 9, 9, 7, 2,
       5, 7, 7, 1, 8, 8, 5, 5, 9, 5, 8, 9, 2, 1, 0, 6, 0, 8, 3, 4, 3, 5,
       9, 5, 4, 3, 0, 4, 1, 0, 2, 6, 7, 1, 1, 6, 0, 5, 5, 9, 4, 0, 2, 5,
       7, 7, 4, 0, 7, 4, 3, 0, 8, 5, 2, 3, 4, 2, 2, 3, 9, 0, 6, 7, 1, 0,
       2, 7, 1, 1, 4, 1, 9, 8, 1, 9, 0, 6, 7, 6, 0, 1, 6, 9, 8, 7, 7, 2,
       5, 2, 9, 2, 9, 2, 6, 9, 0, 9, 0, 1, 8, 4, 8, 1, 4, 2, 5, 1, 5, 0,
       6, 1, 5, 5, 1, 6, 6, 3, 0, 8, 1, 1, 1, 1, 1, 1, 7, 1, 0, 3, 2, 0,
       6, 4, 6, 9, 7, 2, 9, 6, 4, 9, 7, 3, 0, 3, 3, 2, 3, 3, 6, 6, 2, 4,
       9, 1, 2, 1, 1, 3, 4, 3, 7, 5, 6, 0, 1, 5, 1, 1, 6, 5, 0, 3, 4, 8,
       1, 3, 2, 2, 1, 0, 3, 8, 0, 6, 4, 2, 4, 0, 9, 5, 8, 9, 5, 9, 6, 6,
       0, 6, 2, 2, 9, 7, 5, 5, 7, 8, 7, 5, 5, 4, 1, 7, 2, 5, 0, 7, 9, 1,
       6, 4, 7, 1, 1, 2, 6, 9, 0, 8, 8, 7, 9, 6, 4, 3, 1, 9, 3, 4, 2, 6,
       7, 9, 1, 6, 8, 8, 2, 5, 9, 7, 0, 1, 2, 5, 3, 8, 0, 8, 6, 6, 5, 9,
       5, 7, 0, 5, 5, 9, 8, 8, 1, 2, 5, 1, 8, 4, 6, 7, 0, 2, 8, 8, 9, 4,
       2, 4, 3, 4, 2, 3, 6, 9, 9, 8, 4, 4, 9, 6, 6, 4, 1, 6, 7, 9, 5, 5,
       5, 9, 8, 0, 4, 5, 0, 5, 4, 1, 1, 3, 2, 2, 7, 8, 2, 8, 2, 2, 9, 1,
       7, 4, 8, 7, 2, 1, 4, 9, 7, 5, 2, 8, 4, 1, 8, 3, 6, 3, 3, 6, 7, 0,
       7, 1, 7, 3, 2, 3, 0, 2, 5, 7, 7, 6, 5, 9, 4, 1, 8, 2, 3, 1, 8, 4,
       0, 0, 1, 7, 6, 7, 5, 9, 1, 3, 4, 0])
```

Model Accuracy

In [136]:

```
from sklearn.metrics import confusion_matrix, classification_report
```

In [137]:

```
confusion_matrix(y_test, y_pred)
```

Out[137]:

```
array([[51,  0,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0, 66,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 59,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 49,  0,  0,  0,  2,  0,  0],
       [ 0,  0,  0,  0, 48,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 53,  0,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0, 54,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 47,  0,  0],
       [ 0,  2,  0,  0,  1,  0,  0,  0, 49,  0],
       [ 0,  0,  0,  1,  0,  0,  0,  0,  1, 54]])
```

In [138]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        52
           1       0.97      1.00      0.99        66
           2       1.00      1.00      1.00        59
           3       0.98      0.96      0.97        51
           4       0.96      0.98      0.97        49
           5       1.00      0.98      0.99        54
           6       1.00      1.00      1.00        54
           7       0.94      1.00      0.97        47
           8       0.98      0.94      0.96        52
           9       0.98      0.96      0.97        56

    accuracy                           0.98       540
   macro avg       0.98      0.98      0.98       540
weighted avg       0.98      0.98      0.98       540
```