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
```

In [5]:
```
df=pd.read_csv("https://github.com/YBI-Foundation/Dataset/raw/main/MPG.csv")
```

In [6]:
```
df.head()
```

Out[6]:

	mpg	cylinders	displacement	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

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
```

In [8]:
```
df.info()#missing value horse power has missing value
```

```
<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 [9]:

```
df.describe()#summery  statistics
#we have mean different value for features=>dataset has different scale
```

Out[9]:

	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

In [10]:

```
df.corr()
```

Out[10]:

	mpg	cylinders	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
acceleration	0.420289	-0.505419	-0.543684	-0.689196	-0.417457	1.000000	0.288137
model_year	0.579267	-0.348746	-0.370164	-0.416361	-0.306564	0.288137	1.000000

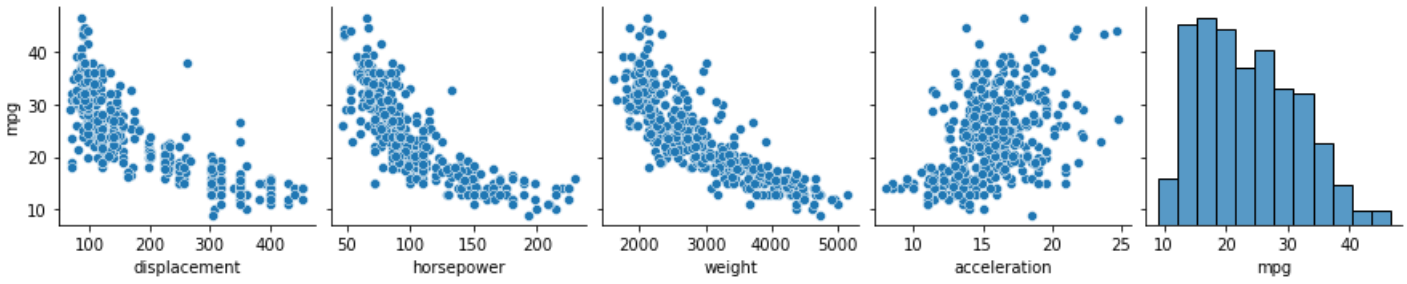
In [12]:
```
df=df.dropna()#drop missing value and store  in df, so all data reduced to 392
```

In [13]:
```
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
```

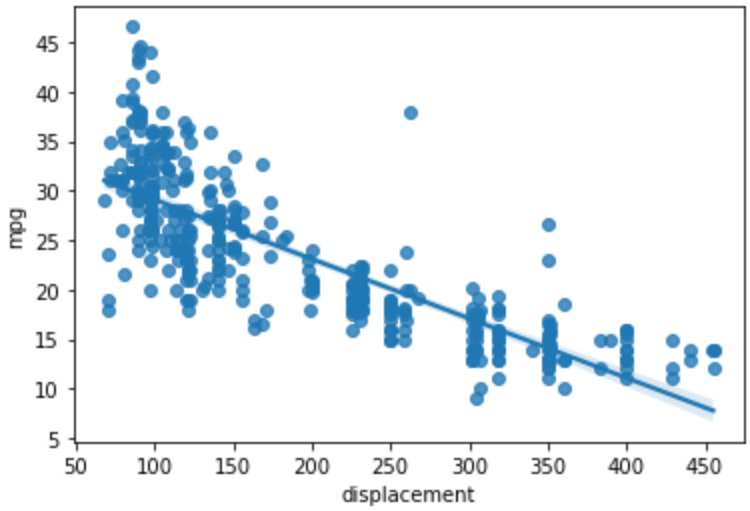
In [23]:
```
sns.pairplot(
    df,
    x_vars=['displacement','horsepower','weight','acceleration','mpg'],
    y_vars=['mpg'],
)#we are using only these features
```

Out[23]:
```
<seaborn.axisgrid.PairGrid  at  0x215cc56e460>
```



In [24]:
```
sns.regplot(x='displacement',y='mpg',data=df)#simple  regression  line
```

Out[24]:
```
<AxesSubplot:xlabel='displacement',  ylabel='mpg'>
```



In [25]:

Out[25]:
```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'name'],
      dtype='object')
```

In [26]:
```
y=df['mpg']
```

In [27]:
```
y.shape
```

Out[27]:
```
(392,)
```

In [29]:
```
x=df[['displacement','horsepower','weight','acceleration']]
```

In [30]:
```
x.shape
```

Out[30]:
```
(392, 4)
```

In [31]:
```
#scale the data
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
ss=StandardScaler()
```

In [33]:
```
X=ss.fit_transform(x)
```

In [35]:
```
X
```

Out[35]:
```
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 [36]:
```
pd.DataFrame(X).describe()
```

Out[36]:

	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

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,train_size=0.7,random_state=0)
```

In [39]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[39]:
```
((274, 4), (118, 4), (274,), (118,))
```

In [41]:
```
from sklearn.linear_model import LinearRegression
```

In [42]:
```
lr=LinearRegression()
```

In [43]:
```
lr.fit(x_train,y_train)
```

Out[43]:
```
LinearRegression()
```

In [44]:
```
lr.intercept_#milege  value
```

Out[44]:
```
23.282609552321578
```

In [45]:
```
lr.coef_# displacement  horsepower  weight  acceleartion
```

Out[45]:
```
array([-1.14649316,  -1.2019826 ,  -4.47558771,  -0.02124705])
```

In [49]:
```
y_pred=lr.predict(x_test)
```

In [50]:
```
y_pred
```

Out[50]:
```
array([29.85557686, 24.84298931, 12.0553547 ,  30.06469152, 29.89641923,
       22.08370663, 31.13070583, 23.49978313, 22.875542  ,  28.52513069,
       31.74836437, 11.84176573, 25.52371763,  8.96144019, 12.70737613,
       29.22140984, 24.72074407, 14.71475857, 28.5524767 ,  25.83049124,
       20.35277629, 18.94056403, 27.30644203, 23.42210027, 29.63476531,
       13.21371562, 25.98923545, 25.75453248, 23.34193305, 17.0243314 ,
       24.07157236, 26.83776215, 27.42077405, 31.07336149, 22.38846104,
       25.80452984, 32.19907663, 11.78888241, 14.12630049,  9.6112525 ,
       15.93990182, 31.5509983 , 27.30331632, 14.86530129, 30.66101311,
       13.01899078, 26.95861374, 11.07828976, 16.0646135 , 24.00912358,
       28.93583311, 16.11897582, 11.68687403, 27.87311093, 29.34258158,
       23.7211364 , 23.28943189, 20.68360918, 32.25038135, 28.61230952,
       24.30245269, 29.93755482, 30.70076977, 10.63426328, 29.61271783,
       17.78271035, 10.146612  , 26.45908698, 26.35195028, 30.26599634,
       32.01361269, 29.81299018, 17.40861662, 14.54103139, 23.92401284,
       13.99230989, 30.8515683 , 18.45972034, 30.37790953, 30.99867893,
       20.68713991, 29.46169893, 23.75768817, 28.16300188,  9.06164988,
       20.63006165, 26.61443165, 26.16828571, 30.74082145, 25.73950869,
       25.53552639, 14.41261578, 25.92823135, 21.72851891, 22.94126972,
       26.82849712, 32.33666073, 24.71122436, 33.22418376, 30.32531943,
        7.39916159, 29.44819226, 27.59094406, 27.0864455 , 29.58832417,
       25.23939357, 29.11574968, 16.47764729, 30.39367377, 13.86676944,
       13.29414205, 32.05892851, 27.91693494, 18.53840297, 26.17311854,
       27.73455466, 24.343233  ,  21.72720868])
```

In [54]:
```
from  sklearn.metrics  import  mean_absolute_error,mean_absolute_percentage_error,r2_score
```

In [52]:
```
mean_absolute_error(y_test,y_pred)
```

Out[52]:
```
3.3697102176643154
```

In [53]:
```
mean_absolute_percentage_error(y_test,y_pred)
```

Out[53]:
```
0.1401214470334179
```

In [55]:
```
r2_score(y_test,y_pred)#variance
```

Out[55]:
```
0.6786796941789563
```

In [56]:

```
#change degree of features to 2
from sklearn.preprocessing import PolynomialFeatures
```

In [59]:
```
poly=PolynomialFeatures(degree=2,interaction_only=True,include_bias=False)
```

In [60]:
```
x_train2=poly.fit_transform(x_train)
```

In [64]:
```
x_test2=poly.fit_transform(x_test)
```

In [69]:
```
lr.fit(x_train2,y_train)
```

Out[69]:
```
LinearRegression()
```

In [70]:
```
lr.intercept_
```

Out[70]:
```
20.986328144855065
```

In [71]:
```
lr.coef_
```

Out[71]:
```
array([-2.64544189, -4.43406687, -2.024449  , -0.86053055,  1.33738429,
        1.00978736, -0.50835528,  0.19893889, -0.1737716 ,  0.88683992])
```

In [72]:
```
y_pred_poly=lr.predict(x_test2)
```

In [73]:
```
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error,r2_score
```

Out[73]:
```
0.6786796941789563
```

In [74]:
```
mean_absolute_error(y_test,y_pred_poly)
```

Out[74]:
```
2.959517622440975
```

In [77]:
```
mean_absolute_percentage_error(y_test,y_pred_poly)
```

Out[77]:
```
0.11837877022970053
```

In [78]:
```
r2_score(y_test,y_pred_poly)#variance
```

Out[78]:
```
0.7313735467550742
```

In [79]:
```
import pandas as pd
```

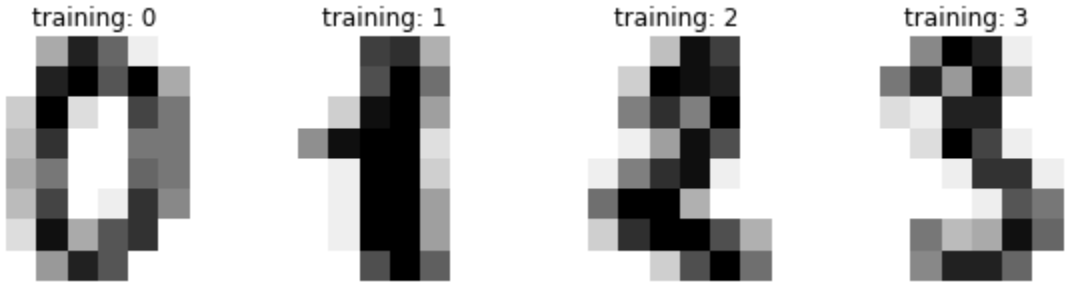
In [80]:
```
import numpy as np
```

In [82]:
```
import matplotlib.pyplot as plt
```

In [83]:
```
from sklearn.datasets import load_digits
```

In [84]:
```
df=load_digits()
```

In [86]:
```
_,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)
```



In [87]:
```
df.images.shape#1797  images
```

Out[87]:
```
(1797, 8, 8)
```

In [88]:
```
df.images[0]#data point in first image
```

Out[88]:
```
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 [89]:
```
df.images[0].shape
```

Out[89]:
```
(8, 8)
```

In [90]:
```
#flatten
```

```
n_samples=len(df.images)#64 columns or we can use -1
data=df.images.reshape((n_samples,-1))
```

In [91]:
```
data[0]#image has different value so it needs scaling
```

Out[91]:
```
array([ 0.,  0.,  5., 13.,  9.,  1.,  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 [92]:
```
data[0].shape
```

Out[92]:
```
(64,)
```

In [93]:
```
data.shape
```

Out[93]:
```
(1797, 64)
```

In [95]:
```
data.min()
```

Out[95]:
```
0.0
```

In [96]:
```
data.max()
```

Out[96]:
```
16.0
```

In [97]:
```
data=data/16
```

In [98]:
```
data.min()
```

Out[98]:
```
0.0
```

In [99]:
```
data.max()
```

Out[99]:
```
1.0
```

In [100…
```
data[0]
```

Out[100…
```
array([0.    , 0.    , 0.3125, 0.8125, 0.5625, 0.0625, 0.    , 0.    ,
       0.    , 0.    , 0.8125, 0.9375, 0.625 , 0.9375, 0.3125, 0.    ,
       0.    , 0.1875, 0.9375, 0.125 , 0.    , 0.6875, 0.5   , 0.    ,
       0.    , 0.25  , 0.75  , 0.    , 0.    , 0.5   , 0.5   , 0.    ,
       0.    , 0.3125, 0.5   , 0.    , 0.    , 0.5625, 0.5   , 0.    ,
       0.    , 0.25  , 0.6875, 0.    , 0.0625, 0.75  , 0.4375, 0.    ,
       0.    , 0.125 , 0.875 , 0.3125, 0.625 , 0.75  , 0.    , 0.    ,
       0.    , 0.    , 0.375 , 0.8125, 0.625 , 0.    , 0.    , 0.    ])
```

In [101…
```
from sklearn.model_selection import train_test_split
```

In [102…
```
x_train,x_test,y_train,y_test=train_test_split(data,df.target,train_size=0.7,random_state
```

In [103…
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[103… ((1257, 64), (540, 64), (1257,), (540,))

In [104…
```
from sklearn.ensemble import RandomForestClassifier
```

In [105…
```
rf=RandomForestClassifier()
```

In [106…
```
rf.fit(x_train,y_train)
```

Out[106… RandomForestClassifier()

In [107…
```
y_pred=rf.predict(x_test)
```

In [108…
```
y_pred
```

Out[108…
```
array([2, 8, 2, 6, 6, 7, 1, 9, 8, 5, 2, 8, 6, 6, 6, 6, 1, 0, 5, 8, 8, 7,
       8, 4, 7, 5, 4, 9, 2, 9, 4, 7, 6, 8, 9, 4, 3, 1, 0, 1, 8, 6, 7, 7,
       1, 0, 7, 6, 2, 1, 9, 6, 7, 9, 0, 0, 9, 1, 6, 3, 0, 2, 3, 4, 1, 9,
       2, 6, 9, 1, 8, 3, 5, 1, 2, 1, 2, 2, 9, 7, 2, 3, 6, 0, 5, 3, 7, 5,
       1, 2, 9, 9, 3, 1, 7, 7, 4, 8, 5, 8, 5, 5, 2, 5, 9, 0, 7, 1, 4, 7,
       3, 4, 8, 9, 7, 9, 8, 0, 6, 5, 2, 5, 8, 4, 1, 7, 0, 6, 1, 5, 5, 9,
       9, 5, 9, 9, 5, 7, 5, 6, 2, 8, 6, 9, 6, 1, 5, 1, 5, 9, 9, 1, 5, 3,
       6, 1, 8, 9, 8, 7, 6, 7, 6, 5, 6, 0, 8, 8, 9, 8, 6, 1, 0, 4, 1, 6,
       3, 8, 6, 7, 4, 3, 6, 3, 0, 3, 3, 3, 0, 7, 7, 5, 7, 8, 0, 7, 8, 9,
       6, 4, 5, 0, 1, 4, 6, 4, 3, 3, 0, 9, 5, 9, 2, 1, 4, 2, 1, 6, 8, 9,
       2, 4, 9, 3, 7, 6, 2, 3, 3, 1, 6, 9, 3, 6, 3, 2, 2, 0, 7, 6, 1, 1,
       9, 7, 2, 7, 8, 5, 5, 7, 5, 2, 3, 7, 2, 7, 5, 5, 7, 0, 9, 1, 6, 5,
       9, 7, 4, 3, 8, 0, 3, 6, 4, 6, 3, 1, 6, 8, 8, 8, 4, 6, 7, 5, 2, 4,
       5, 3, 2, 4, 6, 9, 4, 5, 4, 3, 4, 6, 2, 9, 0, 1, 7, 2, 0, 9, 6, 0,
       4, 2, 0, 7, 9, 8, 5, 7, 8, 2, 8, 4, 3, 7, 2, 6, 9, 1, 5, 1, 0, 8,
       2, 5, 9, 5, 6, 8, 2, 7, 2, 1, 5, 1, 6, 4, 5, 0, 9, 4, 1, 1, 7, 0,
       8, 9, 0, 5, 4, 3, 8, 8, 6, 5, 3, 4, 4, 4, 8, 8, 7, 0, 9, 6, 3, 5,
       2, 3, 0, 8, 8, 3, 1, 3, 3, 0, 0, 4, 6, 0, 7, 7, 6, 2, 0, 4, 4, 2,
       3, 7, 8, 9, 8, 6, 8, 5, 6, 2, 2, 3, 1, 7, 7, 8, 0, 3, 3, 2, 1, 5,
       5, 9, 1, 3, 7, 0, 0, 7, 0, 7, 5, 9, 3, 3, 4, 3, 1, 8, 9, 8, 3, 6,
       2, 1, 6, 2, 1, 7, 5, 5, 1, 9, 2, 8, 9, 7, 2, 1, 4, 9, 3, 2, 6, 2,
       5, 9, 6, 5, 8, 2, 0, 7, 8, 0, 6, 8, 4, 1, 8, 6, 4, 3, 4, 2, 0, 4,
       5, 8, 3, 9, 1, 8, 3, 4, 5, 0, 8, 5, 6, 3, 0, 6, 9, 1, 5, 2, 2, 1,
```

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

model accuracy

In [109…

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

In [110…

```
confusion_matrix(y_test,y_pred)# 45 0's predicted correctly
```

Out[110…

```
array([[45,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 51,  0,  0,  0,  1,  0,  0,  0,  0],
       [ 1,  1, 51,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 53,  0,  0,  0,  0,  1,  0],
       [ 0,  0,  0,  0, 46,  0,  0,  2,  0,  0],
       [ 0,  0,  0,  1,  0, 54,  1,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0, 60,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 53,  0,  0],
       [ 0,  2,  0,  0,  0,  0,  0,  0, 59,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0,  0, 56]], dtype=int64)
```

In [111…

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

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       0.94      0.98      0.96        52
           2       1.00      0.96      0.98        53
           3       0.98      0.98      0.98        54
           4       1.00      0.96      0.98        48
           5       0.96      0.95      0.96        57
           6       0.98      1.00      0.99        60
           7       0.96      1.00      0.98        53
           8       0.98      0.97      0.98        61
           9       0.98      0.98      0.98        57

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