import libraries

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

```
from sklearn.datasets import load_digits
digits=load_digits()
```

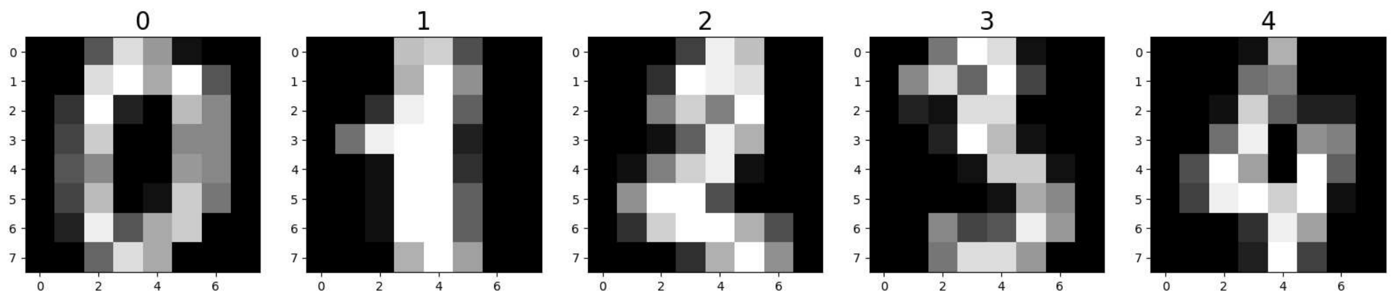
```
x=digits.data
x.shape
```

```
    (1797, 64)
```

```
y=digits.target
y.shape
```

```
    (1797,)
```

```
plt.figure(figsize=(20,4))
for index, (image ,label) in enumerate(zip(digits.data[0:5], digits.target[0:5])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image, (8,8)),cmap=plt.cm.gray)
    plt.title(label,fontsize=20)
```



```
#split the data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print("train input data", x_train.shape)
print("test input data",x_test.shape)
print("train output data ", y_train.shape)
print("test output data", y_test.shape)
```

```
    train input data (1437, 64)
    test input data (360, 64)
    train output data  (1437,)
    test output data (360,)
```

```
#model train
from sklearn.linear_model import LogisticRegression
model= LogisticRegression().fit(x_train,y_train)
model
```

```
    /usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
      n_iter_i = _check_optimize_result(
    ▾ LogisticRegression
    LogisticRegression()
```

```
#prediction
prediction=model.predict(x_test)
prediction
```

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

```
#confusion matrix
from sklearn import metrics
cm=metrics.confusion_matrix(y_test, prediction)
cm
```

```
    array([[27,  0,  0,  0,  0,  0,  0,  0,  0,  0],
           [ 0, 34,  0,  0,  0,  0,  0,  0,  1,  0],
           [ 0,  0, 35,  1,  0,  0,  0,  0,  0,  0],
           [ 0,  0,  0, 29,  0,  0,  0,  0,  0,  0],
           [ 0,  0,  0,  0, 29,  0,  0,  1,  0,  0],
           [ 0,  0,  0,  0,  0, 37,  0,  0,  0,  3],
           [ 0,  1,  0,  0,  0,  0, 43,  0,  0,  0],
           [ 0,  0,  0,  0,  1,  0,  0, 38,  0,  0],
           [ 0,  2,  1,  0,  0,  0,  0,  0, 36,  0],
           [ 0,  0,  0,  0,  0,  1,  0,  0,  0, 40]])
```