```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

#LOAD DATA###################################
digits= load_digits()

#Print to show there are 1797 images (8 by 8 images for a dimensionality of 64)
print("Image Data Shape" , digits.data.shape)

# Print to show there are 1797 labels (integers from 0-9)
print("Label Data Shape", digits.target.shape)

##############################################
#DISPLAY TRAINING DATA

plt.figure(figsize=(10,4)) #(width, height in inches based on default dpi)

#enumerate over list
#zip: in pairs
for index, (image, label) in enumerate(zip(digits.data[0:9],
 digits.target[0:9])):
    plt.subplot(2, 5, index + 1)
    plt.imshow(np.reshape(image, (8,8)), cmap=plt.cm.gray)
    plt.title('\n  Training: %i' % label, fontsize = 8)
plt.show()
##############################################
#SPLIT
x_train, x_test, y_train, y_test = train_test_split(digits.data,
 digits.target, test_size=0.25, random_state=0)

##############################################
# Solver
# LBFGS ,default liblinear
# C= reciprocal regulation magnitude, default C=1
##############################################
model = LogisticRegression(multi_class='auto',solver='lbfgs',max_iter=3000 )
status=model.fit(x_train, y_train)
print( 'status',status)
##############################################
##############################################
# classification
##############################################
W0 = model.intercept_
W  = model.coef_

nclass  = W.shape[0]
nfeature= W.shape[1]
npoints = x_test.shape[0]
```

```
Y=np.zeros((nclass));
Posterior=np.zeros((nclass));
Confusion=np.zeros((nclass,nclass));


for j in range(0,npoints,1):
    for i in range(0,nclass,1):
        Y[i]= W0[i] +np.dot(W[i][:],x_test[j,:])

    for k in range(0,nclass,1):
        Posterior[k]=0
        for i in range(0,nclass,1):
            Posterior[k] = Posterior[k] + np.exp(Y[i]-Y[k])
    Posterior=1/Posterior
    qr = np.argmax( Posterior)
    Confusion[qr,y_test[j]]= Confusion[qr,y_test[j]]+1
##########################################
# Confusion matrix
for i in range(0,nclass,1):
    print( Confusion[i,:])

exit()
```