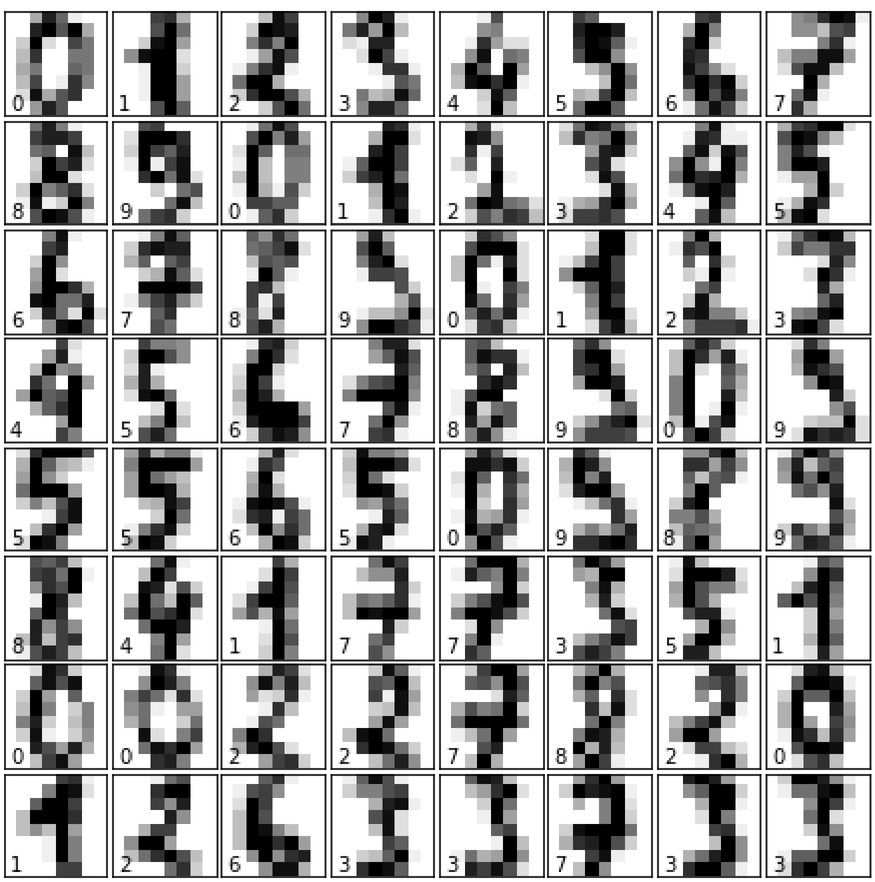
SETTING UP DEPENDENCIES NEEDED

In [2]:
```
#Import data and dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
```

CREATING A VISUALIZATION OF THE DATASET

In [3]:

```
#taken from https://www.datacamp.com/community/tutorials/machine-learning-python
digits = load_digits()

# Import matplotlib
import matplotlib.pyplot as plt

# Figure size (width, height) in inches
fig = plt.figure(figsize=(6, 6))

# Adjust the subplots
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

# For each of the 64 images
for i in range(64):
    # Initialize the subplots: add a subplot in the grid of 8 by 8, at the i+1-th position
    ax = fig.add_subplot(8, 8, i + 1, xticks=[], yticks=[])
    # Display an image at the i-th position
    ax.imshow(digits.images[i], cmap=plt.cm.binary, interpolation='nearest')
    # label the image with the target value
    ax.text(0, 7, str(digits.target[i]))

# Show the plot
plt.show()
```



CREATING PRE-PROCESSING DIGITS DATA

In [6]:
```
X = digits['data']
X
```

Out[6]:
```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

In [7]:
```
X = np.c_[np.ones(digits['data'].shape[0]),X]
```

SPLITTING DATA SET INTO TRAINING AND TESTING DATA SETS

In [633]:
```
#Splitting Data Set into Training and Testing Data Sets
X_train, X_test, y_train, y_test = train_test_split(X, digits['target'],
test_size = .80, random_state = 1)
```

CREATING THE SIGMOID FUNCTION, COST, AND GRADIENT DESCENT FUNCTIONS

In [10]:
```
#creating the sigmoid function for the prediction hypothesis

def sigmoid(z):
    return 1 / (1 + np.exp(-z))
```

In [114]:
```
#Calculating the regularized cost function

def cost_function(X_train, y_train, thetas_array, n, reg):
    prediction = sigmoid(np.dot(X_train,np.transpose(thetas_array)))
    prediction[prediction == 1] = 0.999
    cost = -(np.sum((y_train * np.log(prediction)) + ((1 - y_train) * np
.log(1-prediction)))/n) + ((reg/(2*n))*np.sum(thetas_array[1:]**2))
    return cost
```

In [115]:
```
def gradient_descent(X_train, y_train, alpha, thetas_array,n, reg):
    thetas = thetas_array[1:] - alpha * (((np.dot((sigmoid(np.dot(X_trai
n,np.transpose(thetas_array))) - y_train), X_train[:,1:]))))/n + ((reg/n
)*thetas_array[1:])
    return thetas
```

CREATING THE ONE-VS-ALL FUNCTION FOR

In [607]:
```
def training(X_train, y_train, alpha, iters, reg):
    n = len(X_train)
    thetas_array = np.zeros(65)
    thetas_j = 0
    theta_0 = 0
    cost = []
    #final_thetas = []
    for i in range(iters):
        theta_0 = thetas_array[0] - ((alpha/n)*(np.sum(np.dot(X_train,th
etas_array) - y_train) * 1))
        thetas_j = gradient_descent(X_train, y_train, alpha, thetas_arra
y,n, reg)
        thetas_array = np.insert(thetas_j, 0, theta_0)
        cost.append(cost_function(X_train, y_train, thetas_array, n, reg
))
    #final_thetas.append(thetas_array)

    #Plot cost function error per iteration
    """x = np.arange(0, len(cost), step=1)
    plt.plot(x, cost, "-b", label="Cost Function Curve")
    plt.title("Learning Curve")
    plt.xlabel("Number Of Iterations")
    plt.ylabel("Cost Function Value")
    plt.legend()
    plt.show()"""

    return thetas_array, cost
```

In [624]:
```
def one_vs_all(X_train, y_train, alpha, iters, reg):
    cost = []
    thetas_i = []
    for i in range(len(digits['target_names'])):
        thetas_i.append(training(X_train, np.where(y_train==i,1,0), alph
a, iters, reg)[0])
        cost.append(training(X_train, np.where(y_train==i,1,0), alpha, i
ters, reg)[1])

    return np.array(thetas_i), np.array(cost)
```

In [625]:
```
one_vs_all(X_train, y_train, .001, 1000, .08)
```

Out[625]:
```
(array([[ 4.73722309e+00,  0.00000000e+00, -5.03012002e-03,
         -3.27872725e-02, -1.47327268e-02, -5.83496751e-02,
         -8.94011603e-02, -3.87765466e-02, -4.21101941e-03,
         -9.99696592e-05, -2.38004912e-02, -9.07233286e-03,
         -2.65147094e-02,  1.16204230e-02,  3.98905142e-02,
         -2.90055917e-02, -3.31653156e-03, -3.58844316e-05,
          8.37994303e-03, -2.01334339e-03, -5.38649951e-02,
         -9.17960627e-02,  7.76863714e-02,  5.57654613e-03,
         -6.78008531e-04,  0.00000000e+00,  2.67664589e-02,
         -1.33326644e-02, -1.15017942e-01, -1.89593333e-01,
         -1.23321619e-02,  3.99924588e-02,  0.00000000e+00,
          0.00000000e+00,  4.13529359e-02,  2.46707035e-02,
         -1.36326956e-01, -1.92179118e-01, -6.21298653e-02,
          3.98698662e-02,  0.00000000e+00,  0.00000000e+00,
         -1.42973147e-03,  7.08552784e-02, -9.67702177e-02,
         -1.31509725e-01, -1.84453357e-02,  1.74258743e-02,
         -4.45240149e-04,  0.00000000e+00, -2.16385841e-02,
          3.90985949e-02, -1.99774967e-03,  2.82453568e-02,
          1.28462439e-02, -5.12148677e-02, -2.70656729e-03,
          0.00000000e+00, -7.40035339e-03, -3.89621296e-02,
         -2.29336023e-02, -5.93190381e-02, -6.63748596e-02,
         -3.88899646e-02, -3.26930033e-03],
        [ 4.18809253e+00,  0.00000000e+00, -1.02818897e-02,
         -5.35655603e-02, -7.51647330e-02, -1.24405414e-01,
          2.15170995e-02, -2.72914883e-02, -1.11340184e-02,
         -4.03667044e-04, -6.41117030e-02, -1.55144600e-01,
         -4.87739862e-02,  1.75629311e-02,  2.72643303e-02,
         -4.31896135e-02, -7.72137134e-03, -4.86013194e-04,
         -4.73345429e-03, -2.52064325e-02,  1.25415578e-01,
          7.81864553e-02, -1.84464491e-02, -4.06863181e-02,
         -2.30655844e-03,  0.00000000e+00, -7.72172260e-03,
         -8.86078285e-03,  4.21576957e-02, -5.55696438e-03,
         -3.76124603e-02, -5.34568054e-02,  0.00000000e+00,
          0.00000000e+00, -5.79600964e-02, -6.26710790e-02,
          2.72244949e-03, -2.26405159e-02, -1.28853450e-01,
         -6.52394454e-02,  0.00000000e+00,  0.00000000e+00,
         -7.36019135e-02, -9.98607303e-02, -1.82242901e-03,
          4.62899835e-02, -8.26539186e-02, -8.19909901e-02,
         -1.08017330e-03,  0.00000000e+00, -2.53109796e-02,
         -5.22246669e-02, -4.67069046e-03,  5.08076964e-02,
         -8.43772663e-03, -3.96881693e-02,  2.44464384e-02,
          0.00000000e+00, -1.07775618e-02, -9.31348338e-02,
         -1.04410421e-01, -2.96070748e-02,  6.10153974e-02,
          6.45685518e-03,  1.71317109e-02],
        [ 4.39176959e+00,  0.00000000e+00,  8.52668870e-03,
          3.55391254e-02, -4.71145299e-02, -7.14372770e-02,
         -6.09481916e-02, -2.76536030e-02, -2.95582913e-03,
         -2.55185662e-04,  6.00071150e-02,  7.61236145e-03,
         -6.04307064e-02,  1.11624837e-02, -5.98214283e-02,
         -9.04232698e-03, -1.53955958e-03,  5.59484516e-04,
          1.81556722e-02, -8.09113737e-02, -7.64458310e-02,
          2.92327929e-02, -2.70887722e-02,  5.42996863e-03,
         -3.05009514e-04,  0.00000000e+00, -4.42599657e-02,
         -1.61262277e-01, -1.65956839e-01, -2.60588655e-02,
         -1.65075515e-02, -2.91693003e-02,  0.00000000e+00,
          0.00000000e+00, -4.39844627e-02, -1.02144680e-01,
         -6.87537952e-02, -5.13667488e-02, -1.06034324e-01,
```

```
       -7.26553068e-02,  0.00000000e+00,  0.00000000e+00,
       -3.90520296e-03,  1.16182880e-02,  7.71818448e-02,
       -3.76424617e-02, -1.36941472e-01, -5.77154568e-02,
        2.56445894e-03,  0.00000000e+00,  2.67673415e-02,
        2.35759759e-02,  5.92762535e-02,  7.02672409e-02,
        5.17287370e-02,  5.81915200e-02,  1.55719857e-02,
        0.00000000e+00,  1.14076348e-02,  4.59597281e-02,
       -5.34676707e-02, -5.04901025e-02,  3.34814864e-02,
        9.84629184e-02,  2.33890261e-02],
      [ 4.45279760e+00,  0.00000000e+00,  1.84735540e-03,
       -2.63854654e-02, -2.95364830e-02,  2.60497604e-02,
        1.57824483e-02, -8.01164683e-03, -6.08162885e-03,
       -7.47309630e-05,  3.32773586e-02, -1.41010243e-02,
       -7.53168488e-02, -2.63990061e-02,  2.28067711e-03,
        1.09340220e-04, -4.00887216e-03, -2.94207084e-04,
       -3.09087789e-02, -1.60241992e-01, -8.32101935e-02,
        8.97983801e-02, -7.89726681e-02, -4.10519077e-02,
       -8.43517241e-04,  0.00000000e+00, -3.84022569e-02,
       -1.63578567e-01,  1.84533578e-02,  3.30671594e-02,
       -1.40988234e-01, -6.95877767e-02,  0.00000000e+00,
        0.00000000e+00, -4.13322727e-02, -1.28226438e-01,
       -7.38681307e-02,  3.35973631e-02,  3.50078874e-02,
       -1.10084230e-02,  0.00000000e+00,  0.00000000e+00,
       -4.68432909e-03, -9.14909881e-02, -1.66884467e-01,
       -7.15443131e-02,  6.93996189e-02,  7.67479455e-02,
       -7.57015555e-04,  0.00000000e+00,  1.26195908e-02,
       -1.94697071e-02, -7.13260643e-02, -2.47080573e-02,
        1.94270528e-02,  1.06193677e-02, -1.45917010e-02,
        0.00000000e+00, -5.68567186e-03, -8.85422673e-03,
       -2.54320100e-03, -1.20742781e-02, -2.70477534e-02,
       -6.00155665e-02, -1.79358371e-02],
      [ 4.65284707e+00,  0.00000000e+00, -3.79238336e-03,
       -4.25073343e-02, -9.99324713e-02, -7.85641476e-02,
       -1.34477074e-01, -6.99784874e-02, -1.00534047e-02,
       -1.82876778e-04, -1.68261917e-02, -6.91602659e-02,
       -7.05058075e-02, -1.03795266e-01, -1.14286419e-01,
       -5.78730902e-02, -8.71822251e-03, -4.18237693e-05,
       -2.91372897e-02,  1.73294684e-02,  3.55696441e-02,
       -1.78302053e-02,  6.72072685e-03,  1.28480278e-02,
        1.48926676e-04,  0.00000000e+00,  2.47396022e-02,
        7.08631038e-02, -3.92214142e-02, -2.33953410e-02,
       -3.14098699e-03,  3.65172496e-02,  0.00000000e+00,
        0.00000000e+00,  7.32190225e-02,  2.11558404e-02,
       -3.86978640e-02,  1.76460937e-02,  1.24955264e-02,
        2.84586292e-02,  0.00000000e+00,  0.00000000e+00,
        6.66129499e-02, -1.25143730e-03,  7.66818350e-02,
        8.87896089e-02, -8.28717194e-03, -4.69046719e-02,
       -5.90421972e-04,  0.00000000e+00,  2.42839420e-02,
       -8.83229530e-02, -3.71014545e-02,  3.18850631e-03,
       -1.10000635e-01, -8.63015371e-02, -4.41317425e-03,
        0.00000000e+00, -2.01245834e-03, -3.45417427e-02,
       -9.06763530e-02, -7.30991224e-02, -1.17303157e-01,
       -3.76955407e-02, -3.41306626e-03],
      [ 4.31468041e+00,  0.00000000e+00,  7.29804424e-03,
        6.66766039e-02, -3.55910600e-02, -4.74033725e-05,
        1.11047712e-01,  3.88927387e-02, -1.83624957e-02,
        5.36996339e-04,  4.75579633e-03,  2.13132662e-02,
```

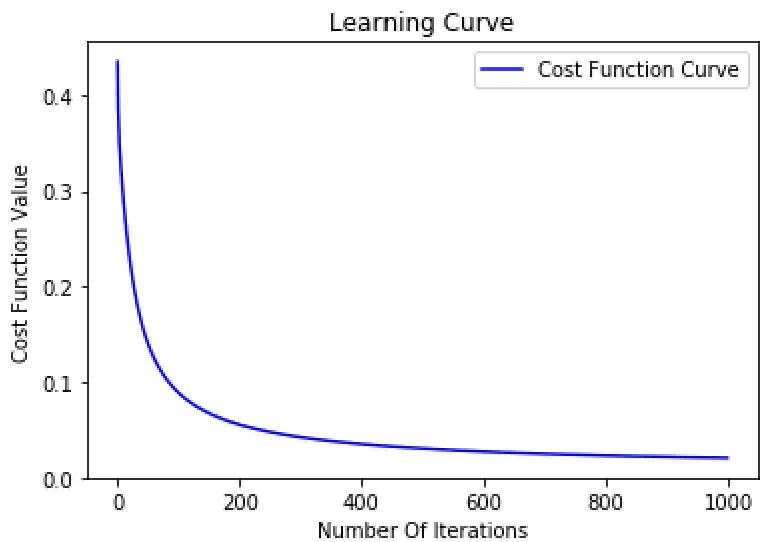
```
 -3.47508098e-02, -4.65106466e-02, -8.53747295e-02,
 -1.50917363e-02, -1.27970745e-02, -1.37027529e-04,
  2.21845689e-02,  5.35252113e-02, -3.94928191e-02,
 -1.40465219e-01, -2.13060392e-01, -7.39006360e-02,
 -2.20889239e-03,  0.00000000e+00,  5.56322807e-02,
  6.65033403e-02,  4.32946329e-02, -2.83440935e-02,
 -1.15565663e-01, -5.89438708e-02,  0.00000000e+00,
  0.00000000e+00, -9.41720482e-03, -6.71622663e-03,
 -2.25097610e-02, -3.23531113e-02, -4.98140840e-02,
 -3.16720722e-02,  0.00000000e+00,  0.00000000e+00,
 -3.82760024e-02, -1.43943181e-01, -1.04267767e-01,
 -4.17800209e-02, -3.80660981e-03, -2.63707062e-02,
 -3.95379358e-04,  0.00000000e+00,  4.10961299e-03,
 -6.88286697e-02, -7.43034282e-02,  3.47876578e-03,
 -2.48136620e-02, -7.14840397e-02, -4.41085101e-03,
  0.00000000e+00,  1.07723686e-02,  6.50686127e-02,
 -1.26422751e-02, -5.59689068e-02, -9.80033511e-02,
 -4.74319830e-02, -1.25125729e-02],
[ 4.72239032e+00,  0.00000000e+00, -5.11327227e-03,
 -8.05489182e-02, -5.25027622e-02, -8.79753741e-02,
 -5.06085905e-02, -1.99176432e-02, -1.96190529e-03,
 -7.38583274e-05, -3.33975694e-02, -7.24452272e-02,
 -3.93088700e-02, -1.19448041e-01, -1.15274480e-01,
 -1.43006721e-02, -1.70290766e-03, -4.27334932e-05,
 -4.72569995e-02,  5.63261778e-03, -1.65427398e-02,
 -1.08724313e-01, -1.76482738e-01, -4.21466833e-02,
 -1.18734605e-03,  0.00000000e+00, -3.07873775e-02,
  1.30297294e-02,  3.20103584e-03, -2.87987737e-02,
 -7.99898456e-02, -7.18086404e-02,  0.00000000e+00,
  0.00000000e+00, -5.42342372e-03,  6.44984286e-02,
  2.52371967e-02, -1.36924306e-02,  2.03544998e-02,
 -2.37529485e-02,  0.00000000e+00,  0.00000000e+00,
 -1.52976203e-02,  9.60920792e-02, -7.23904086e-03,
 -1.22053087e-01,  1.22999831e-02,  7.88170888e-02,
  6.99143155e-03,  0.00000000e+00, -1.83178262e-02,
  2.23067331e-02,  2.42036326e-02, -1.18481751e-01,
  8.16236554e-03,  6.24481274e-02, -1.37868718e-02,
  0.00000000e+00, -7.18450833e-03, -8.98028646e-02,
 -6.95421670e-02, -4.95868710e-02,  3.64822086e-02,
  3.92272770e-03, -1.40449691e-02],
[ 4.75396625e+00,  0.00000000e+00,  5.00378115e-03,
 -8.15565024e-03,  3.95760936e-03,  1.13310966e-02,
  6.26730381e-02,  5.83069337e-02,  2.38448193e-02,
 -6.02567849e-05, -2.59163416e-02, -2.37534945e-02,
 -2.43755514e-02,  6.73752654e-03,  4.81569709e-02,
  4.34328991e-02,  1.64274398e-02, -6.79003560e-05,
 -4.91790368e-02, -1.01725311e-01, -1.14622847e-01,
 -5.15331166e-02, -3.56495982e-03,  7.54259591e-03,
  4.24626108e-03,  0.00000000e+00, -6.15767039e-02,
 -1.05139706e-01, -1.25782104e-01, -3.73143368e-02,
  4.56826275e-02,  3.46076113e-02,  0.00000000e+00,
  0.00000000e+00, -3.41915857e-02,  5.98153164e-03,
  2.30863567e-02,  4.65210515e-02,  2.81522928e-02,
  2.86812879e-02,  0.00000000e+00,  0.00000000e+00,
 -1.19891247e-02, -2.44300896e-02,  3.23727382e-02,
  1.76153951e-03, -6.38902453e-02, -2.55246272e-02,
 -2.75884628e-04,  0.00000000e+00, -5.72812179e-03,
```

```
        -6.85360040e-02, -1.89807332e-02, -1.35689283e-01,
        -1.41824553e-01, -4.78615223e-02, -2.31599485e-03,
         0.00000000e+00,  8.46295869e-03, -5.99837070e-03,
        -6.51847103e-02, -1.87808049e-01, -1.08683291e-01,
        -3.83494621e-02, -3.57929291e-03],
       [ 3.07065935e+00,  0.00000000e+00, -1.91189903e-02,
        -6.64764571e-02, -1.20095392e-01, -4.26617375e-02,
        -3.38118641e-02, -8.96690335e-02, -1.15073707e-02,
         3.55980681e-03, -2.34255219e-02, -7.32779988e-04,
        -4.98352803e-02, -1.17543322e-01,  5.98368076e-02,
         1.28535905e-02, -5.70426667e-03, -1.26929005e-04,
         1.24405159e-02,  6.46157247e-02,  4.38679119e-04,
        -1.66796416e-02,  5.51656543e-02,  8.90779428e-03,
        -1.64150669e-03,  0.00000000e+00, -5.22272187e-02,
        -1.44009277e-02,  9.85573722e-02, -2.29713848e-03,
        -4.44025648e-02, -9.89255099e-02,  0.00000000e+00,
         0.00000000e+00, -7.23761414e-02, -7.43249855e-02,
         1.25450695e-01,  7.00205443e-03, -1.88643769e-01,
        -1.33075582e-01,  0.00000000e+00,  0.00000000e+00,
        -1.39669779e-02,  1.04508215e-01,  3.77136099e-03,
        -5.33932791e-02,  4.42593991e-02, -8.29929874e-02,
        -1.11437074e-03,  0.00000000e+00,  2.01773996e-02,
         6.48545198e-02, -1.25046771e-01, -8.93499166e-02,
         2.19210103e-02, -5.49084855e-02, -9.60956324e-03,
         0.00000000e+00, -1.70285790e-02, -1.01386533e-01,
        -3.91590830e-02,  3.44661910e-02, -7.22911214e-02,
        -7.52393949e-02, -1.12113202e-02],
       [ 4.24764117e+00,  0.00000000e+00, -8.18174019e-03,
        -4.09195720e-02, -4.35970689e-02, -7.41376448e-02,
        -1.33645074e-01, -1.82026270e-02,  1.73239021e-02,
        -6.95179904e-05, -2.76610087e-02,  6.35486175e-03,
        -5.85164146e-02, -2.29702256e-02, -3.12872853e-02,
        -1.10787067e-02,  1.21706338e-02, -9.37735306e-05,
        -1.78904633e-02,  2.50172087e-02, -3.19416060e-02,
        -2.51705549e-02,  9.30578250e-02,  3.24325235e-02,
         4.97862446e-04,  0.00000000e+00, -2.53395300e-02,
         2.27420709e-02,  5.19726582e-02,  5.05984836e-02,
         1.41021763e-01,  2.93056563e-02,  0.00000000e+00,
         0.00000000e+00, -7.90698608e-02, -8.45924803e-02,
         3.15301727e-02, -1.10086972e-01, -1.08496963e-02,
         2.44155282e-02,  0.00000000e+00,  0.00000000e+00,
        -2.69255207e-02, -1.68336073e-01, -1.33552549e-01,
        -1.28204324e-01, -3.31004469e-02,  9.95908327e-04,
        -6.15246856e-04,  0.00000000e+00, -5.84926944e-03,
        -2.38770588e-02, -8.67004826e-02, -9.99042582e-02,
        -7.30864267e-02,  1.76451747e-02, -3.12369353e-03,
         0.00000000e+00, -7.86374229e-03, -4.81988679e-02,
        -6.01906578e-02, -1.12998313e-02, -1.92764864e-03,
        -2.53774864e-02, -5.00346649e-03]]),
 array([[0.43484911, 0.38524676, 0.36393725, ..., 0.02068795, 0.0206784
8,
        0.02066902],
       [0.39396703, 0.3406495 , 0.32121488, ..., 0.08884629, 0.0888460
1,
        0.08884578],
       [0.39419662, 0.33923507, 0.31791476, ..., 0.027304  , 0.0272837
7,
```

```
         0.02726357],
        ...,
        [0.3995314 , 0.34496806, 0.32324576, ..., 0.01944164, 0.0194258
1,
         0.01941   ],
        [0.43774161, 0.39519345, 0.3813745 , ..., 0.12302737, 0.1229978
2,
         0.12296833],
        [0.36665805, 0.30788297, 0.28634522, ..., 0.06140076, 0.0613822
1,
         0.06136372]]))
```

CHECKING IF GRADIENT DESCENT WORKS AS INTENDED

In [626]:
```python
cost = one_vs_all(X_train, y_train, .001, 1000, .08)[1]
x = np.arange(0, len(cost[0]), step=1)
plt.plot(x, cost[0], "-b", label="Cost Function Curve")
plt.title("Learning Curve")
plt.xlabel("Number Of Iterations")
plt.ylabel("Cost Function Value")
plt.legend()
plt.show()
```



CREATING ARRAY OF PREDICTED VALUES

In [628]:
```python
def predict_one_vs_all(X_train, all_thetas):
    prediction = sigmoid(np.dot(X_train,np.transpose(all_thetas)))
    pred_argmax = np.argmax(prediction,axis = 1)
    return pred_argmax
```

In [629]:
```
predict_one_vs_all(X_train, all_thetas)
```

Out[629]:
```
array([4, 2, 4, 6, 0, 7, 8, 1, 3, 3, 8, 9, 4, 9, 7, 1, 0, 0, 6, 7, 4,
9,
       7, 6, 7, 5, 8, 8, 7, 0, 0, 8, 8, 0, 7, 2, 8, 3, 5, 0, 6, 9, 2,
2,
       2, 9, 5, 7, 7, 6, 9, 8, 8, 3, 5, 7, 7, 2, 6, 0, 8, 7, 0, 9, 7,
6,
       7, 4, 3, 0, 5, 0, 0, 6, 5, 8, 0, 1, 4, 1, 2, 4, 7, 5, 0, 7, 0,
4,
       0, 0, 9, 7, 7, 4, 6, 6, 5, 8, 0, 1, 5, 7, 9, 0, 7, 7, 0, 3, 8,
7,
       2, 0, 4, 4, 0, 1, 0, 2, 5, 5, 8, 7, 5, 3, 6, 7, 3, 4, 8, 0, 6,
8,
       8, 7, 2, 0, 9, 3, 7, 8, 6, 3, 7, 9, 8, 0, 3, 7, 6, 8, 0, 0, 8,
5,
       5, 1, 8, 2, 7, 3, 1, 5, 1, 8, 1, 6, 3, 9, 4, 4, 7, 5, 6, 2, 5,
0,
       7, 5, 6, 0, 9, 2, 3, 9, 0, 7, 8, 1, 0, 1, 4, 1, 2, 8, 0, 8, 0,
9,
       0, 1, 5, 5, 3, 6, 2, 4, 0, 1, 0, 0, 3, 3, 2, 5, 3, 8, 1, 3, 7,
0,
       1, 2, 2, 9, 3, 5, 3, 8, 1, 5, 8, 8, 6, 9, 3, 6, 6, 9, 0, 6, 2,
7,
       0, 9, 6, 8, 5, 3, 0, 2, 5, 5, 6, 6, 0, 8, 3, 9, 2, 1, 1, 2, 2,
9,
       2, 1, 7, 5, 2, 5, 0, 2, 8, 5, 1, 7, 6, 8, 3, 7, 1, 2, 9, 9, 6,
5,
       6, 7, 5, 2, 6, 1, 9, 9, 1, 9, 8, 4, 6, 3, 8, 6, 1, 3, 4, 4, 6,
1,
       5, 3, 4, 0, 6, 1, 0, 3, 8, 2, 3, 8, 3, 2, 2, 1, 3, 4, 7, 8, 4,
6,
       4, 5, 2, 2, 1, 0, 2, 6, 9, 0, 4, 3, 7, 8, 8, 8, 3, 4, 1, 4, 6,
5,
       2, 9, 1, 4, 9, 1, 5])
```

CALCULATING THE TRAINING ACCURACY

In [631]:
```
def training_accuracy(X_train, y_train, all_thetas):
    prediction_train = predict_one_vs_all(X_train, all_thetas)
    correct = 0
    incorrect = 0

    for i in range(len(prediction_train)):
        if prediction_train[i] == y_train[i]:
            correct += 1
        else:
            incorrect += 1
    accuracy = round((correct/len(prediction_train)) * 100, 2)
    print(f'Training Accuracy: {accuracy}%')
    #return correct
```

In [632]:
```
training_accuracy(X_train, y_train, all_thetas)
```

```
Training Accuracy: 96.94%
```

CALCULATIN THE TESTING ACCURACY

In [635]:
```
def test_accuracy(X_test, y_test, all_thetas):
    prediction_train = predict_one_vs_all(X_test, all_thetas)
    correct = 0
    incorrect = 0

    for i in range(len(prediction_train)):
        if prediction_train[i] == y_test[i]:
            correct += 1
        else:
            incorrect += 1
    accuracy = round((correct/len(prediction_train)) * 100, 2)
    print(f'Training Accuracy: {accuracy}%')
```

In [636]:
```
test_accuracy(X_test, y_test, all_thetas)
```

```
Training Accuracy: 92.84%
```

In [ ]:
```

```