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

indata = np.loadtxt(r"C:\Users\Zhenc_000\Documents\Caltech\CS156\Set8\features.train.txt")
outdata = np.loadtxt(r"C:\Users\Zhenc_000\Documents\Caltech\CS156\Set8\features.test.txt")

def setx(source):
    source[:,0] = [1 for i in range(len(source))]
    return source

def setz(source):
    array = np.zeros([len(source),6])
    array[:,0] = [1 for i in range(len(source))]
    array[:,1] = [source[i,1] for i in range(len(source))]
    array[:,2] = [source[i,2] for i in range(len(source))]
    array[:,3] = [source[i,1]*source[i,2] for i in range(len(source))]
    array[:,4] = [source[i,1]**2 for i in range(len(source))]
    array[:,5] = [source[i,2]**2 for i in range(len(source))]
    return array

def length(number, source): #function to calculate number of data points for each digit
    return sum(1 for i in range (len(source)) if source[i,0] == number)

def populate(number, source): #create x-vector for a single digit
    array = np.zeros([length(number,source),3])
    array[:,0] = [1 for i in range(len(source)) if source[i,0] == number]
    array[:,1] = [source[i,1] for i in range(len(source)) if source[i,0] == number]
    array[:,2] = [source[i,2] for i in range(len(source)) if source[i,0] == number]
    return array

train_set = [populate(x, indata) for x in range(10)]
test_set = [populate(x, outdata) for x in range(10)]

def populate_all(number, source): #set classification of all other digits to -1 and target
digit to 1
    array = np.array([1 if source[i,0] == number else -1 for i in range(len(source))])
    return array

train_all = [populate_all(x, indata) for x in range(10)]
test_all = [populate_all(x, outdata) for x in range(10)]

def regreg(x,y,l): #function to perform linear regression with regularization
    return np.dot(np.dot(np.linalg.inv(np.dot(np.transpose(x),x) + l*np.identity(len(x[0]))),np.
    transpose(x)),y)

def error(w,z,y): #find the in-sample or out of sample error
    g = np.sign(np.dot(z,np.transpose(w)))
    return ((len(y)-np.sum(np.dot(y,g)))/2)/len(y)

trainx = setx(indata)
testx = setx(outdata)
trainz = setz(indata)
testz = setz(outdata)
```

```
trainz15 = np.append(setz(train_set[1]),setz(train_set[5]),axis=0) #set lists for 1 vs 5
classifier
y15in = np.append(np.ones([len(train_set[1])]),-1*np.ones([len(train_set[5])]),axis=0)
testz15 = np.append(setz(test_set[1]),setz(test_set[5]),axis=0)
y15out = np.append(np.ones([len(test_set[1])]),-1*np.ones([len(test_set[5])]),axis=0)

#print E_in and E_out for lambda = 1 and 0.01
print(error(regreg(trainz15,y15in,1),trainz15,y15in),error(regreg(trainz15,y15in,1),testz15,
y15out),error(regreg(trainz15,y15in,0.01),trainz15,y15in),error(regreg(trainz15,y15in,0.01),
testz15,y15out))

error_in = [error(regreg(trainx,train_all[i],1),trainx,train_all[i]) for i in range(0,10)]
error_out = [error(regreg(trainx,train_all[i],1),testx,test_all[i]) for i in range(0,10)]
zerror_in = [error(regreg(trainz,train_all[i],1),trainz,train_all[i]) for i in range(0,10)]
zerror_out = [error(regreg(trainz,train_all[i],1),testz,test_all[i]) for i in range(0,10)]

plt.plot(error_in) #plot Errors
plt.plot(error_out)
plt.plot(zerror_in)
plt.plot(zerror_out)
plt.xlabel('Digit')
plt.ylabel('Error')
plt.legend(['In-sample','Out-sample','In-sample-z','Out-sample-z'])
plt.show()
```