```
# for questions #7-10

import numpy as np

def importData(filename):
    data = open(filename, 'r')
    lines = data.read().splitlines()

    matx = np.zeros(shape=(len(lines),3))
    vecty = np.zeros(shape=(len(lines),1))

    for i in range(len(lines)):
        point = lines[i].strip(' ').split('  ')
        matx[i] = np.matrix([1, float(point[1]), float(point[2])])
        vecty[i] = float(point[0])

    return matx, vecty


# change the y vectors to one_v_all binary classifiers
def one_v_all(vecty, digit):
    for i in range(len(vecty)):
        if vecty[i] == digit:
            vecty[i] = 1
        else:
            vecty[i] = -1

    return vecty


# generate a new dataset reflecting one-versus-one binary classifier
def one_v_one(dataset, d1, d2):
    matx, vecty = dataset
    newMatx = np.empty((0, 3))
    newVecty = np.empty(0)
    for i in range(len(vecty)):
        if vecty[i] == d1:
            newMatx = np.append(newMatx, [matx[i]], 0)
            newVecty = np.append(newVecty, 1)
        elif vecty[i] == d2:
            newMatx = np.append(newMatx, [matx[i]], 0)
            newVecty = np.append(newVecty, -1)
    return newMatx, newVecty


# feature transformation
def transform(matx):
    matz = np.zeros(shape=(len(matx), 6))
    for i in range(len(matx)):
        x1 = matx[i, 1]
        x2 = matx[i, 2]

        matz[i] = np.matrix([1, x1, x2, x1 * x2, x1 ** 2, x2 ** 2])

    return matz


def calculateError(w, dataset):
    matz, vecty = dataset
    errorCount = 0
    for i in range(len(matz)):
        if np.sign(w.transpose().dot(matz[i, :])) != np.sign(vecty[i]):
            errorCount += 1

    return errorCount / len(matz)
```

```
# input parameter digit represents the one digit with class +1 for
# one-versus-all binary classifier
def weightDecay(digit):
    dataset = importData('features.train.txt')
    matx, vecty = dataset
    vecty = one_v_all(vecty, digit)

    #find w_reg
    lam = 1
    w = (np.linalg.inv(matx.transpose().dot(matx) + lam * np.identity(3))
        .dot(matx.transpose()).dot(vecty))

    #calculate E_in
    inError = calculateError(w, (matx, vecty))

    #calculate E_out
    testMatx, testVecty = importData('features.test.txt')
    testVecty = one_v_all(testVecty, digit)
    outError = calculateError(w, (testMatx, testVecty))

    print(digit, 'versus all without transform')
    print('E_in:', inError, 'E_out:', outError, '\n')

# 7
# for i in range(5, 10):
#     weightDecay(i)


# input parameter digit represents the one digit with class +1 for
# one-versus-all binary classifier
def transformWeightDecay(digit):
    matx, vecty = importData('features.train.txt')
    vecty = one_v_all(vecty, digit)
    matx = transform(matx)

    #find w_reg
    lam = 1
    w = (np.linalg.inv(matx.transpose().dot(matx) + lam * np.identity(6))
        .dot(matx.transpose()).dot(vecty))

    #calculate E_in
    inError = calculateError(w, (matx, vecty))

    #calculate E_out
    testMatx, testVecty = importData('features.test.txt')
    testVecty = one_v_all(testVecty, digit)
    testMatx = transform(testMatx)
    outError = calculateError(w, (testMatx, testVecty))

    print(digit, 'versus all with transform')
    print('E_in:', inError, 'E_out:', outError, '\n')

# 8
# for i in range(0, 5):
#     transformWeightDecay(i)


# 9
# for i in range(0, 10):
#     weightDecay(i)
#     transformWeightDecay(i)


# input param lam is regularization constant lamda
def transformWeightDecay(lam):
```

```
    dataset = importData('features.train.txt')
    dataset = one_v_one(dataset, 1, 5)
    matx, vecty = dataset
    matx = transform(matx)

    #find w_reg
    w = (np.linalg.inv(matx.transpose().dot(matx) + lam * np.identity(6))
        .dot(matx.transpose()).dot(vecty))

    #calculate E_in
    inError = calculateError(w, (matx, vecty))

    #calculate E_out
    testData = importData('features.test.txt')
    testData = one_v_one(testData, 1, 5)
    testMatx, testVecty = testData
    testMatx = transform(testMatx)
    outError = calculateError(w, (testMatx, testVecty))

    print('lamda:', lam)
    print('E_in:', inError, 'E_out:', outError, '\n')

# 10
# transformWeightDecay(0.01)
# transformWeightDecay(1)
```