```
#for #5-7

import numpy as np
import random

def createData(f, n):
    matx = np.zeros(shape=(n,3))
    vecty = np.zeros(shape=(n,1))
    for i in range(0, n):
        x = np.matrix([1, random.uniform(-1, 1), random.uniform(-1, 1)])
        matx[i] = x
        vecty[i] = 1 if (x[0,1] > f(x[0,0])) else -1

    return (matx, vecty)

def setup(n):
    #Set up the target function f
    x = random.uniform(-1, 1), random.uniform(-1, 1)
    y = random.uniform(-1, 1), random.uniform(-1, 1)
    z = np.polyfit(x, y, 1)
    f = np.poly1d(z)

    #create data
    dataset = createData(f, n)

    return f, dataset


#5
def linRegIn(dataset, n):
    matx, vecty = dataset
    w = np.linalg.pinv(matx).dot(vecty)
    errorCount = 0
    for i in range(0, n):
        if np.sign(w.transpose().dot(matx[i,:])) != np.sign(vecty[i]):
            errorCount += 1
    return (errorCount / n)


#6
def linRegOut(data, n):
    f, dataset = data
    matx, vecty = dataset
    w = np.linalg.pinv(matx).dot(vecty)
    newmatx, newvecty = createData(f, 1000)
    errorCount = 0
    for i in range(0, 1000):
        if np.sign(w.transpose().dot(newmatx[i,:])) != np.sign(newvecty[i]):
            errorCount += 1
    return (errorCount / 1000)


#7
def pla():
    f, dataset = setup(10)
    matx, vecty = dataset
    w = np.linalg.pinv(matx).dot(vecty).transpose()
    misclassified = [0]
    iterCount = 0
    while misclassified:
```

```
        iterCount += 1
        misclassified = []
        for i in range(0, 10):
            x = matx[i,:]
            y = vecty[i]
            if np.sign(w.dot(x))[0] != np.sign(y)[0]:
                misclassified.append(i)
        if misclassified:
            index = random.choice(misclassified)
            w = w + vecty[index] * matx[index,:]
    return iterCount


#5: Calculate E_in
totalErrorFreq = 0
for i in range(0, 1000):
    totalErrorFreq += linRegIn(setup(100)[1], 100)
print(totalErrorFreq / 1000)

#6: Calculate E_out
totalErrorFreq = 0
for i in range(0, 1000):
    totalErrorFreq += linRegOut(setup(100), 100)
print(totalErrorFreq / 1000)

#7: PLA
totalIterCount = 0
for i in range(0, 1000):
    totalIterCount += pla()
print(totalIterCount/1000)
```