Q.12

```
from liblinear.liblinearutil import *
import numpy as np
import random
import math

def getdata(a):
        text =[]
        if a ==1:
            path = 'hw4_train.dat.txt'
        elif a ==2:
            path = 'hw4_test.dat.txt'
        with open(path) as f:
           for line in f:
               text.append([float(i) for i in line.split()])
        mm=np.asarray(text)
        X=mm[:,:-1]
        Y=mm[:,-1]

        return X,Y
def Transform(X):
    """
    Feature transform x into \Phi(x) = [1, x_i, x_ix_j , ], for all i != j
    """
    #Q = X.shape[1]
    function = []
    func_temp =[1]
    row = X.shape[1]
    a = 0
    b = 0
    c = 0
    for i in X:

            for x in i[0:]:
                func_temp.append(x)
            for x in i[0:]:
                for _ in range(a , row):
                    func_temp.append(x*X[b][_])
                a+=1
            a=0

            for x in i[0:]:
                for _ in range(a , row):
                    for j in range(c , row):
                        func_temp.append(x*X[b][_]*X[b][j])

                    c+=1
                a+=1
                c=a

            function.append(func_temp)
            func_temp = [1]
            b+=1
            a=0
            c=0

    function_array = np.asarray(function)
    return function_array


trainX ,trainY  = getdata(1)[0],getdata(1)[1]
testX ,testY = getdata(2)[0],getdata(2)[1]
trainX = Transform(trainX)
testX = Transform(testX)
Eout = []


for Lambda in [0.0001, 0.01, 1, 100, 10000]:
        C = 1 / (2 * Lambda)

        prob = problem(trainY, trainX)
        param = parameter('-s 0 -c {} -e 0.000001'.format(C))
        m = train(prob, param)
        p_label, p_acc, p_val = predict(testY, testX, m)
        Eout.append(np.mean(testY != p_label))
print(Eout)
```

Q13

上面都一樣我就只擷取主程式碼的部分

```
trainX ,trainY  = getdata(1)[0],getdata(1)[1]
testX ,testY = getdata(2)[0],getdata(2)[1]
trainX = Transform(trainX)
testX = Transform(testX)
Ein = []


for Lambda in [0.0001, 0.01, 1, 100, 10000]:
        C = 1 / (2 * Lambda)

        prob = problem(trainY, trainX)
        param = parameter('-s 0 -c {} -e 0.000001'.format(C))
        m = train(prob, param)
        p_label, p_acc, p_val = predict(trainY, trainX, m)
        Ein.append(np.mean(trainY != p_label))
print(Ein)
```

Q14

```
X ,Y  = getdata(1)[0],getdata(1)[1]

X = Transform(X)
trainX = X[:120,:]
trainY = Y[:120]
testX = X[120:,:]
testY = Y[120:]


Eval= []
for Lambda in [0.0001, 0.01, 1, 100, 10000]:
        C = 1 / (2 * Lambda)

        prob = problem(trainY, trainX)
        param = parameter('-s 0 -c {} -e 0.000001'.format(C))
        m = train(prob, param)
        p_label, p_acc, p_val = predict(testY, testX, m)
        Eval.append(np.mean(testY != p_label))
print(Eval)
print('#####################################################')
tX ,tY = getdata(2)[0],getdata(2)[1]
tX = Transform(tX)
Eout=[]
for Lambda in [0.0001, 0.01, 1, 100, 10000]:
        C = 1 / (2 * Lambda)

        prob = problem(trainY, trainX)
        param = parameter('-s 0 -c {} -e 0.000001'.format(C))
        m = train(prob, param)
        p_label, p_acc, p_val = predict(tY, tX, m)
        Eout.append(np.mean(tY != p_label))
print(Eout)
```

Q15

```
trainX ,trainY  = getdata(1)[0],getdata(1)[1]
testX ,testY = getdata(2)[0],getdata(2)[1]
trainX = Transform(trainX)
testX = Transform(testX)
Eout = []


for Lambda in [0.0001, 0.01, 1, 100, 10000]:
        C = 1 / (2 * Lambda)

        prob = problem(trainY, trainX)
        param = parameter('-s 0 -c {} -e 0.000001'.format(C))
        m = train(prob, param)
        p_label, p_acc, p_val = predict(testY, testX, m)
        Eout.append(np.mean(testY != p_label))
print(Eout)
```

Q16

```
X ,Y  = getdata(1)[0],getdata(1)[1]
X = Transform(X)
X = [X[i:i+40,:] for i in range(0,200,40)]
Y = [Y[i:i+40] for i in range(0,200,40)]
Ecv=[]
for Lambda in [0.0001, 0.01, 1, 100, 10000]:
        C = 1 / (2 * Lambda)
        for i in range(5):
            testX = X[i]
            testY = Y[i]
            trainX = np.vstack([X[j] for j in range(5) if i != j])
            trainY = np.hstack([Y[j] for j in range(5) if i != j])
            prob = problem(trainY, trainX)
            param = parameter('-s 0 -c {} -e 0.000001'.format(C))
            m = train(prob, param)
            p_label, p_acc, p_val = predict(testY, testX, m)
            Ecv.append(np.mean(testY != p_label))
        print(Ecv)
        print('average Ecv = ', np.mean(Ecv))
        print('#####################################')
        Ecv = []
```