HW8 :

1[d] We are trying to minimize W which we had split into (purple w from lecture 14) w and b where is w0 hence as w has d parameters and b has 1 parameter then we have quadratic programming with d+1 variable hence [d] as we are still solving quadratic functions with d+1 variables.

(2) [a]

hw8.py > ...

```python
import numpy
from sklearn import svm
import random

train = numpy.loadtxt('./features.train')
test = numpy.loadtxt('./features.test')
digits_train = train[:, 0]
X_train = train[:, 1:]
digits_test = test[:, 0]
X_test = test[:, 1:]


def make_binary(digit_class, digits):
    scores = []
    for score in digits:
        if (score == digit_class):
            scores.append(1)
        else:
            scores.append(-1)
    return scores

def Ein(d_classifier, c, q):
    binary_train = make_binary(d_classifier, digits_train)
    classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
    classifier.fit(X_train, binary_train)
    return 1 - classifier.score(X_train, binary_train)
def Eout(d_classifier, c, q):…
def support_vectors(d_classifier, c, q):…
def Ecv(digits, X, folds, c, q):…
def find_best_c(d_classifier, runs):…
def q2():
    print("for 0, Ein is: ", Ein(0, 0.01, 2))
    print("for 2, Ein is: ", Ein(2, 0.01, 2))
    print("for 4, Ein is: ", Ein(4, 0.01, 2))
    print("for 6, Ein is: ", Ein(6, 0.01, 2))
    print("for 8, Ein is: ", Ein(8, 0.01, 2))
def q3():…
def q4():…
def q5(q):…
def q6():…
def q7():…
def q8():…
def q9():…
def q10():…
q2()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/hw8.py"
for 0, Ein is:  0.10588396653408316
for 2, Ein is:  0.10026059525442321
for 4, Ein is:  0.08942531888629812
for 6, Ein is:  0.09107118365107669
for 8, Ein is:  0.074338225209162
thierno@Thiernos-MacBook-Pro CS  156a % 
```

← Hence according to the above code the highest Ein is for 0 versus all hence the correct answer is [a]

Ln 129, Col 5 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit

(3) [a]

hw8.py > make_binary

```python
import numpy
from sklearn import svm
import random

train = numpy.loadtxt('./features.train')
test = numpy.loadtxt('./features.test')
digits_train = train[:, 0]
X_train = train[:, 1:]
digits_test = test[:, 0]
X_test = test[:, 1:]


def make_binary(digit_class, digits):
    scores = []
    for score in digits:
        if (score == digit_class):
            scores.append(1)
        else:
            scores.append(-1)
    return scores

def Ein(d_classifier, c, q):
    binary_train = make_binary(d_classifier, digits_train)
    classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
    classifier.fit(X_train, binary_train)
    return 1 - classifier.score(X_train, binary_train)
def Eout(d_classifier, c, q):…
def support_vectors(d_classifier, c, q):…
def Ecv(digits, X, folds, c, q):…
def find_best_c(d_classifier, runs):…
def q2():…
def q3():
    print("for 1, Ein is: ", Ein(1, 0.01, 2))
    print("for 2, Ein is: ", Ein(3, 0.01, 2))
    print("for 4, Ein is: ", Ein(5, 0.01, 2))
    print("for 6, Ein is: ", Ein(7, 0.01, 2))
    print("for 8, Ein is: ", Ein(9, 0.01, 2))
def q4():…
def q5(q):…
def q6():…
def q7():…
def q8():…
def q9():…
def q10():…
q3()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/hw8.py"
for 1, Ein is:  0.014401316691811772
for 2, Ein is:  0.09024825126868741
for 4, Ein is:  0.07625840076807022
for 6, Ein is:  0.08846523110684401
for 8, Ein is:  0.08832807570977919
thierno@Thiernos-MacBook-Pro CS  156a % 
```

← Hence according to the above code, the lowest Ein is for 1 v all hence the correct answer is [a]

Ln 19, Col 30 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit

(4) [c]

hw8.py > make_binary

```python
import numpy
from sklearn import svm
import random

train = numpy.loadtxt('./features.train')
test = numpy.loadtxt('./features.test')
digits_train = train[:, 0]
X_train = train[:, 1:]
digits_test = test[:, 0]
X_test = test[:, 1:]


def make_binary(digit_class, digits):
    scores = []
    for score in digits:
        if (score == digit_class):
            scores.append(1)
        else:
            scores.append(-1)
    return scores

def Ein(d_classifier, c, q):…
def Eout(d_classifier, c, q):…
def support_vectors(d_classifier, c, q):
    binary_train = make_binary(d_classifier, digits_train)
    classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
    classifier.fit(X_train, binary_train)
    return sum(classifier.n_support_)
def Ecv(digits, X, folds, c, q):…
def find_best_c(d_classifier, runs):…
def q2():…
def q3():…
def q4():
    print("for 0, number of support vectors are : ", support_vectors(0, 0.01, 2))
    print("for 1, number of support vectors are : ", support_vectors(1, 0.01, 2))
    print("Their differnce in number of support vectors is: ", support_vectors(0, 0.01, 2) - support_vectors(1, 0.01, 2))
def q5(q):…
def q6():…
def q7():…
def q8():…
def q9():…
def q10():…
q4()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

zsh

Python

```
for 6, Ein is:  0.08846523110684401
for 8, Ein is:  0.08832807570977919
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/hw8.py"
for 0, number of support vectors are :  2179
for 1, number of support vectors are :  386
Their differnce in number of support vectors is:  1793
thierno@Thiernos-MacBook-Pro CS  156a %
```

← Hence according to the above code, the correct answer is [c] as it is closest.

Ln 18, Col 14 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit

Instead of screenshotting the same code, the below code is the same for all functions and variables for questions 5 through 10:

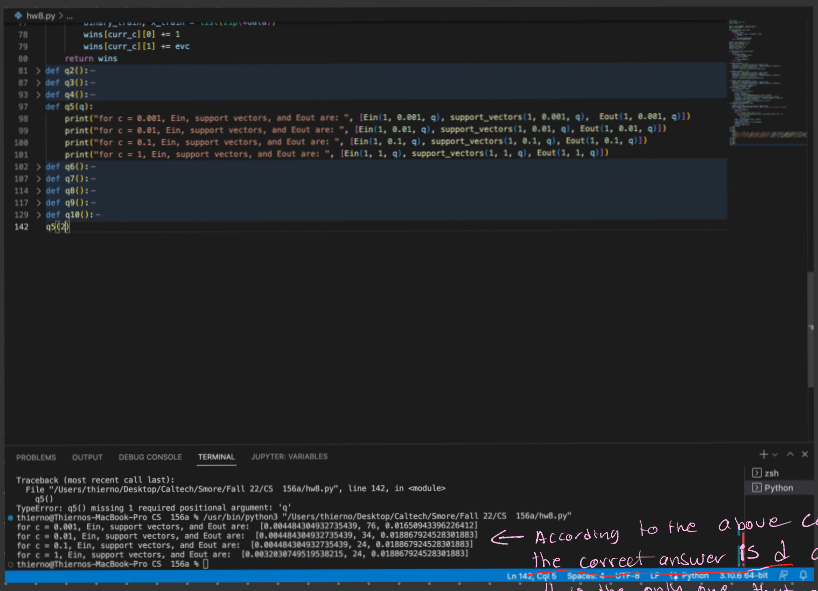
hw8.py > ...

```python
import numpy
from sklearn import svm
import random

train = numpy.loadtxt('./features.train')
test = numpy.loadtxt('./features.test')
#One_versus_One
def one_v_one(data, num1, num2):
    new_data = []
    for thing in data:
        if thing[0] != num1 and thing[0] != num2:
            continue
        else:
            new_data.append(thing)
    return numpy.array(new_data)

otrain = one_v_one(train, 1, 5)
otest = one_v_one(test, 1, 5)
digits_train = otrain[:, 0]
X_train = otrain[:, 1:]
digits_test = otest[:, 0]
X_test = otest[:, 1:]

def make_binary(digit_class, digits):
    scores = []
    for score in digits:
        if (score == digit_class):
            scores.append(1)
        else:
            scores.append(-1)
    return scores

def Ein(d_classifier, c, q):
    binary_train = make_binary(d_classifier, digits_train)
    classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
    classifier.fit(X_train, binary_train)
    return 1 - classifier.score(X_train, binary_train)

def Eout(d_classifier, c, q):
    binary_train = make_binary(d_classifier, digits_train)
    binary_test = make_binary(d_classifier, digits_test)
    classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
    classifier.fit(X_train, binary_train)
    return 1 - classifier.score(X_test, binary_test)
```

hw8.py > ...

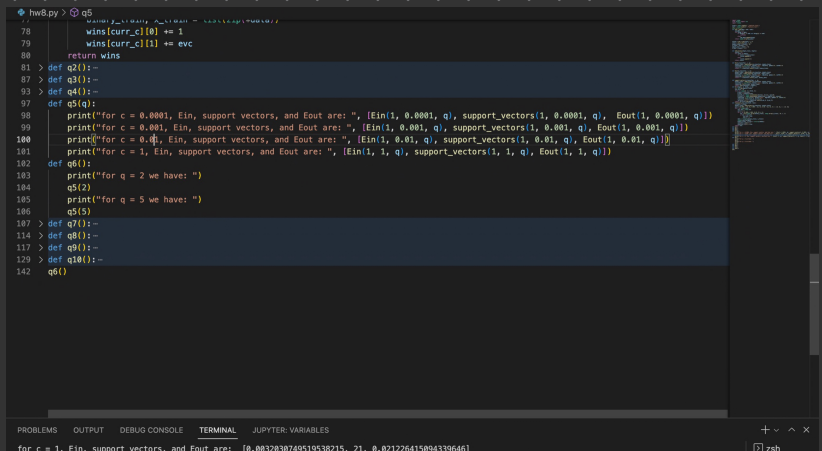
```python
def Eout(d_classifier, c, q):
    binary_train = make_binary(d_classifier, digits_train)
    binary_test = make_binary(d_classifier, digits_test)
    classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
    classifier.fit(X_train, binary_train)
    return 1 - classifier.score(X_test, binary_test)

def support_vectors(d_classifier, c, q):
    binary_train = make_binary(d_classifier, digits_train)
    classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
    classifier.fit(X_train, binary_train)
    return sum(classifier.n_support_)
def Ecv(digits, X, folds, c, q):
    k = int(len(digits)/folds)
    tot_error = 0
    for n in range(0, k * folds, k):
        D_val_X = X[n:n+k]
        D_val_Y = digits[n:n+k]
        D_train_X = numpy.concatenate((X[n+k:], X[:n]), axis=0)
        D_train_Y = numpy.concatenate((digits[n+k:], digits[:n]), axis=0)
        classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
        classifier.fit(D_train_X, D_train_Y)
        tot_error += (1 - classifier.score(D_val_X, D_val_Y))
    return tot_error/len(digits)
def find_best_c(d_classifier, runs):
    x_train = X_train
    binary_train = make_binary(d_classifier, digits_train)
    wins = {0.0001 : [0, 0], 0.001 : [0, 0], 0.01 : [0, 0], 0.1 : [0, 0], 1 : [0, 0]}
    for i in range(runs):
        curr_min = numpy.Inf
        curr_c = 0
        for c in [0.0001, 0.001, 0.01, 0.1, 1]:
            evc = Ecv(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
            if evc < curr_min:
                curr_min = evc
                curr_c = c
        data = list(zip(binary_train, x_train))
        random.shuffle(data)
        binary_train, x_train = list(zip(*data))
        wins[curr_c][0] += 1
        wins[curr_c][1] += evc
    return wins
```

(5) [d]



← According to the above code the correct answer is d as it is the only one that matches the outputed behaviour of our code.

(6) [b]



← According to the above code, the correct answer is [b] as it is the only one that matches the behaviour of our output.

(7) [b]

hw8.py > ...

```
        for c in [0.0001, 0.001, 0.01, 0.1, 1]:
            evc = Ecv(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
            if evc < curr_min:
                curr_min = evc
                curr_c = c
        data = list(zip(binary_train, x_train))
        random.shuffle(data)
        binary_train, x_train = list(zip(*data))
        wins[curr_c][0] += 1
        wins[curr_c][1] += evc
    return wins
def q2():...
def q3():...
def q4():...
def q5(q):...
def q6():...
def q7():
    score_board = find_best_c(1, 100)
    print("c= 0.0001 won this many times: ", score_board[0.0001][0])
    print("c= 0.001 won this many times: ", score_board[0.001][0])
    print("c= 0.01 won this many times: ", score_board[0.01][0])
    print("c= 0.1 won this many times: ", score_board[0.1][0])
    print("c= 1 won this many times: ", score_board[1][0])
def q8():...
def q9():...
def q10():...
q7()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
for c = 0.1, Ein, support vectors, and Eout are:  [0.004484304932735439, 24, 0.018867924528301883]
for c = 1, Ein, Eout, support vectors, and Eout are:  [0.003203074951953821 5, 24, 0.018867924528301883]
for q = 5 we have:
for c = 0.001, Ein, support vectors, and Eout are:  [0.004484304932735439, 25, 0.021226415094339646]
for c = 0.01, Ein, support vectors, and Eout are:  [0.003843689942344630 2, 23, 0.021226415094339646]
for c = 0.1, Ein, support vectors, and Eout are:  [0.003203074951953821 5, 25, 0.018867924528301883]
for c = 1, Ein, Eout, support vectors, and Eout are:  [0.003203074951953821 5, 21, 0.021226415094339646]
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/hw8.py"
c= 0.0001 won this many times:  0
c= 0.001 won this many times:  26
c= 0.01 won this many times:  27
c= 0.1 won this many times:  9
c= 1 won this many times:  8
thierno@Thiernos-MacBook-Pro CS  156a %
```

← According to the above code the correct answer is [b].

Ln 142, Col 5 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit

(8) [c]

hw8.py > ...

```
def find_best_c(d_classifier, runs):
    x_train = X_train
    binary_train = make_binary(d_classifier, digits_train)
    wins = {0.0001 : [0, 0], 0.001 : [0, 0], 0.01 : [0, 0], 0.1 : [0, 0], 1 : [0, 0]}
    for i in range(runs):
        curr_min = numpy.Inf
        curr_c = 0
        for c in [0.0001, 0.001, 0.01, 0.1, 1]:
            evc = Ecv(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
            if evc < curr_min:
                curr_min = evc
                curr_c = c
        data = list(zip(binary_train, x_train))
        random.shuffle(data)
        binary_train, x_train = list(zip(*data))
        wins[curr_c][0] += 1
        wins[curr_c][1] += evc
    return wins
def q2():...
def q3():...
def q4():...
def q5(q):...
def q6():...
def q7():...
def q8():
    score_board = find_best_c(1, 100)
    print("c= 0.001 had an average Evc of: ", (score_board[0.001][1]) * (100 + (score_board[0.001][0])) / (score_board[0.001][0]))
def q9():...
def q10():...
q8()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
c= 1 won this many times:  8
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/hw8.py"
c= 0.001 had an average Evc of:  0.004675126419234704
thierno@Thiernos-MacBook-Pro CS  156a %
```

← Hence according to the above code, the correct answer is [c]

(9) [e]

```python
    for i in range(runs):
        curr_min = numpy.Inf
        curr_c = 0
        for c in [0.0001, 0.001, 0.01, 0.1, 1]:
            evc = Ecv(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
            if evc < curr_min:
                curr_min = evc
                curr_c = c
        data = list(zip(binary_train, x_train))
        random.shuffle(data)
        binary_train, x_train = list(zip(*data))
        wins[curr_c][0] += 1
        wins[curr_c][1] += evc
    return wins
def q2():…
def q3():…
def q4():…
def q5(q):…
def q6():…
def q7():…
def q8():…
def q9():
    binary_train = make_binary(1, digits_train)
    curr_min = numpy.inf
    curr_c = 0
    for c in [0.01, 1, 100, 10 ** 4, 10 ** 6]:
        classifier = svm.SVC(C=c, kernel='rbf', degree=2, gamma=1.0, coef0=1.0)
        classifier.fit(X_train, binary_train)
        Ein = 1 - classifier.score(X_train, binary_train)
        if (Ein < curr_min):
            curr_min = Ein
            curr_c = c
    return curr_c
def q10():…
print(q9())
```

```
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/hw8.py"
1000000
thierno@Thiernos-MacBook-Pro CS  156a %
```

Hence according to the above code the correct answer is [e]

(10) [c]

```python
    for i in range(runs):
        curr_min = numpy.Inf
        curr_c = 0
        for c in [0.0001, 0.001, 0.01, 0.1, 1]:
            evc = Ecv(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
            if evc < curr_min:
                curr_min = evc
                curr_c = c
        data = list(zip(binary_train, x_train))
        random.shuffle(data)
        binary_train, x_train = list(zip(*data))
        wins[curr_c][0] += 1
        wins[curr_c][1] += evc
    return wins
def q2():…
def q3():…
def q4():…
def q5(q):…
def q6():…
def q7():…
def q8():…
def q9():…
def q10():
    binary_train = make_binary(1, digits_train)
    binary_test = make_binary(1, digits_test)
    curr_min = numpy.inf
    curr_c = 0
    for c in [0.01, 1, 100, 10 ** 4, 10 ** 6]:
        classifier = svm.SVC(C=c, kernel='rbf', degree=2, gamma=1.0, coef0=1.0)
        classifier.fit(X_train, binary_train)
        Eout = 1 - classifier.score(X_test, binary_test)
        if (Eout < curr_min):
            curr_min = Eout
            curr_c = c
    return curr_c
print(q10())
```

```
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/hw8.py"
100
thierno@Thiernos-MacBook-Pro CS  156a %
```

Hence according to the above code, the correct answer is [c].