```
c = 0.1
151 out of 169 correct. recall = 0.897436, precision = 0.714286, F1 = 0.795455

c = 0.000002
158 out of 169 correct. recall = 0.743590, precision = 0.966667, F1 = 0.840580

# make predictions using all of the data points in x
# print ‘success’ or ‘failure’ depending on whether the
# prediction is correct
#
# x is the data set
# y is the labels
# w is the current set of weights
#
def predict (x, y, w):
    correct = 0;
    pred_true_act_true = 0.0
    act_true = 0.0
    pred_true = 0.0

    for index in range (len (y)):

        if ((np.dot (x[index], w) > 0) and (y[index] > 0)):
            print ('success')
            correct = correct + 1
            pred_true_act_true += 1
        elif ((np.dot (x[index], w) < 0) and (y[index] < 0)):
            print ('success')
            correct = correct + 1
        else:
               print ('failure')

        if (y[index] > 0):
            act_true += 1
        if (np.dot (x[index], w) > 0):
            pred_true += 1


        recall = pred_true_act_true/act_true
        prec = pred_true_act_true/pred_true
        f1 = (2*prec * recall)/(recall + prec)
    print ('%d out of %d correct. recall = %f, precision = %f, F1 = %f' %
(correct, len(y),
                                                                    recall,
                                                                    prec,
                                                                    f1))
# evaluates and returns the gradient
#
# x is the data set
# y is the labels
# w is the current set of weights
# c is the weight of the slack variables
#
def gradient(x, y, w, c):
    # Note that the gradient has 30 dims because the data has 30 dims
    gradient = np.zeros (30)
    n = x.shape[0]
    lmd = 1/(n*c)
```

```
    for j in range(gradient.shape[0]):
        gradient[j] += lmd*w[j]

        for i in range(x.shape[0]):
            if(1-y[i]*(np.dot(x[i],w)) >= 0):
                gradient[j] += -y[i]*x[i][j]/float(n)


    return gradient


# evaluates the loss function and returns the loss
#
# x is the data set
# y is the labels
# w is the current set of weights
# c is the weight of the slack variables
#
def f (x, y, w, c):
    loss = 0
    # fill in missing code here!!
    n = x.shape[0]
    lmd = 1/(n*c)

    loss = lmd/2 * np.linalg.norm(w)**2

    for i in range(n):
        loss += max(0, 1-y[i]*(np.dot(w,x[i])))/float(n)


    return loss
```