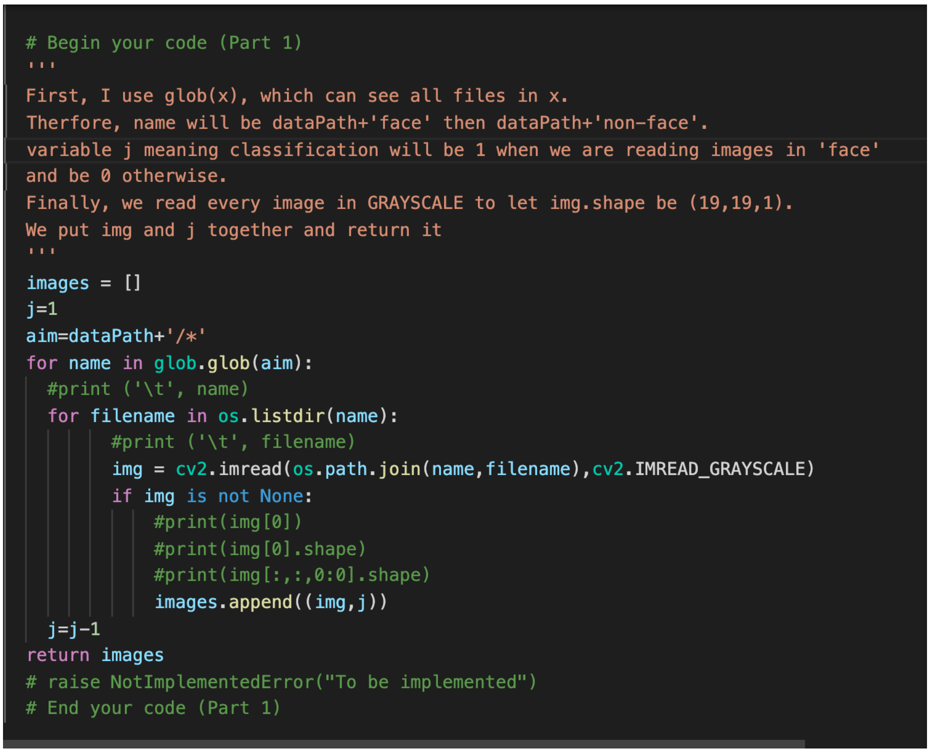
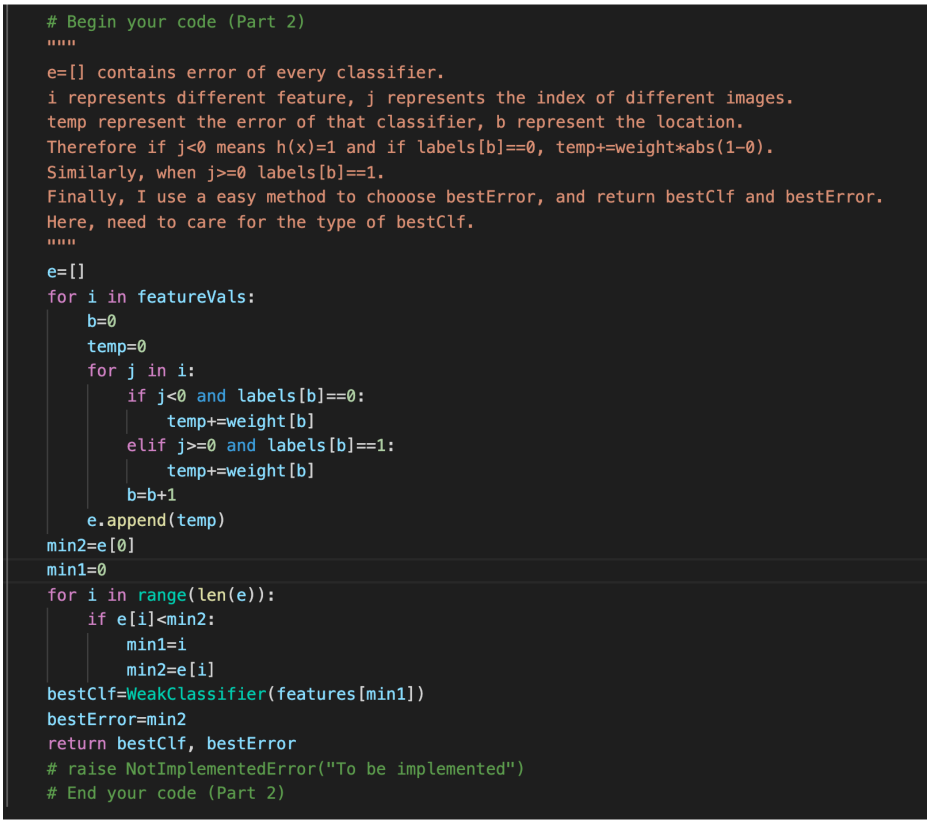
Homework 1: Face Detection Report

109550116 楊傑宇

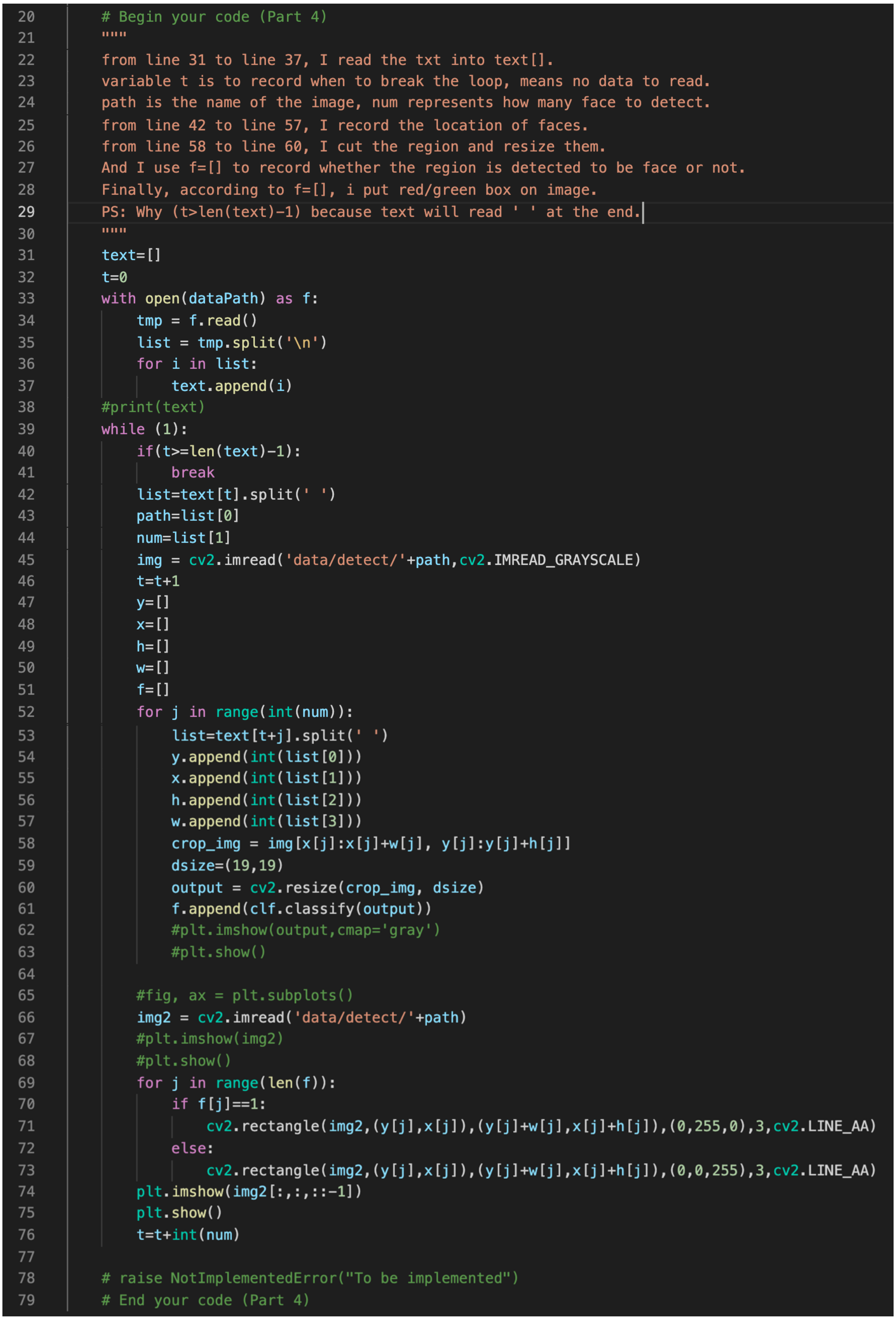
Part I. Implementation (6%):



```
# Begin your code (Part 1)
'''
First, I use glob(x), which can see all files in x.
Therfore, name will be dataPath+'face' then dataPath+'non-face'.
variable j meaning classification will be 1 when we are reading images in 'face'
and be 0 otherwise.
Finally, we read every image in GRAYSCALE to let img.shape be (19,19,1).
We put img and j together and return it
'''
images = []
j=1
aim=dataPath+'/*'
for name in glob.glob(aim):
  #print ('\t', name)
  for filename in os.listdir(name):
        #print ('\t', filename)
        img = cv2.imread(os.path.join(name,filename),cv2.IMREAD_GRAYSCALE)
        if img is not None:
            #print(img[0])
            #print(img[0].shape)
            #print(img[:,:,0:0].shape)
            images.append((img,j))
  j=j-1
return images
# raise NotImplementedError("To be implemented")
# End your code (Part 1)
```

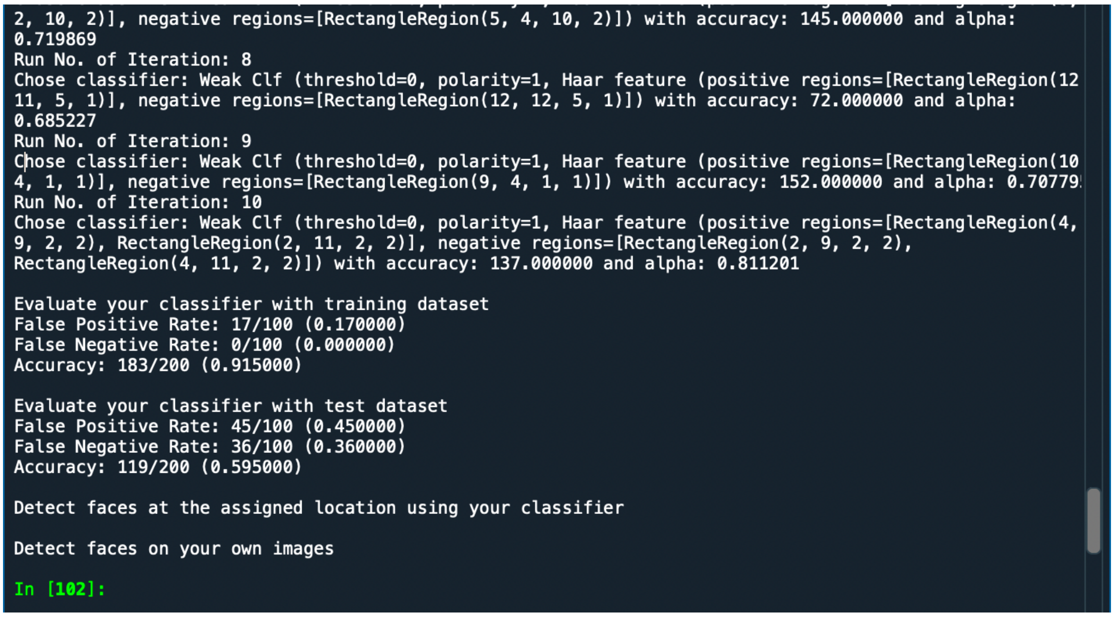


```
# Begin your code (Part 2)
""""
e=[] contains error of every classifier.
i represents different feature, j represents the index of different images.
temp represent the error of that classifier, b represent the location.
Therefore if j<0 means h(x)=1 and if labels[b]==0, temp+=weight*abs(1-0).
Similarly, when j>=0 labels[b]==1.
Finally, I use a easy method to chooose bestError, and return bestClf and bestError.
Here, need to care for the type of bestClf.
""""
e=[]
for i in featureVals:
    b=0
    temp=0
    for j in i:
        if j<0 and labels[b]==0:
            temp+=weight[b]
        elif j>=0 and labels[b]==1:
            temp+=weight[b]
        b=b+1
    e.append(temp)
min2=e[0]
min1=0
for i in range(len(e)):
    if e[i]<min2:
        min1=i
        min2=e[i]
bestClf=WeakClassifier(features[min1])
bestError=min2
return bestClf, bestError
# raise NotImplementedError("To be implemented")
# End your code (Part 2)
```



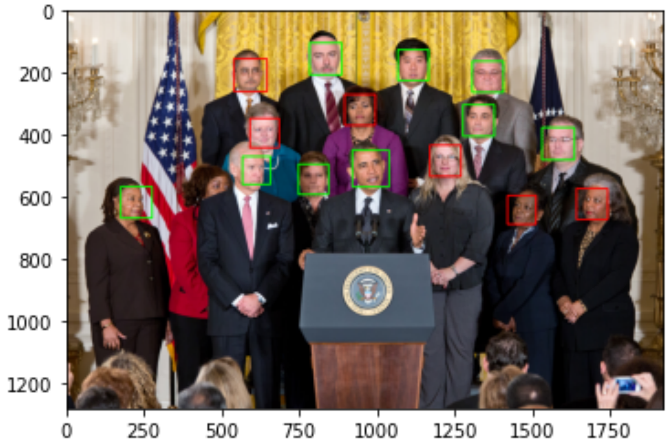
```
20      # Begin your code (Part 4)
21      """
22      from line 31 to line 37, I read the txt into text[].
23      variable t is to record when to break the loop, means no data to read.
24      path is the name of the image, num represents how many face to detect.
25      from line 42 to line 57, I record the location of faces.
26      from line 58 to line 60, I cut the region and resize them.
27      And I use f=[] to record whether the region is detected to be face or not.
28      Finally, according to f=[], i put red/green box on image.
29      PS: Why (t>len(text)-1) because text will read ' ' at the end.
30      """
31      text=[]
32      t=0
33      with open(dataPath) as f:
34          tmp = f.read()
35          list = tmp.split('\n')
36          for i in list:
37              text.append(i)
38      #print(text)
39      while (1):
40          if(t>=len(text)-1):
41              break
42          list=text[t].split(' ')
43          path=list[0]
44          num=list[1]
45          img = cv2.imread('data/detect/'+path,cv2.IMREAD_GRAYSCALE)
46          t=t+1
47          y=[]
48          x=[]
49          h=[]
50          w=[]
51          f=[]
52          for j in range(int(num)):
53              list=text[t+j].split(' ')
54              y.append(int(list[0]))
55              x.append(int(list[1]))
56              h.append(int(list[2]))
57              w.append(int(list[3]))
58              crop_img = img[x[j]:x[j]+w[j], y[j]:y[j]+h[j]]
59              dsize=(19,19)
60              output = cv2.resize(crop_img, dsize)
61              f.append(clf.classify(output))
62              #plt.imshow(output,cmap='gray')
63              #plt.show()
64
65          #fig, ax = plt.subplots()
66          img2 = cv2.imread('data/detect/'+path)
67          #plt.imshow(img2)
68          #plt.show()
69          for j in range(len(f)):
70              if f[j]==1:
71                  cv2.rectangle(img2,(y[j],x[j]),(y[j]+w[j],x[j]+h[j]),(0,255,0),3,cv2.LINE_AA)
72              else:
73                  cv2.rectangle(img2,(y[j],x[j]),(y[j]+w[j],x[j]+h[j]),(0,0,255),3,cv2.LINE_AA)
74          plt.imshow(img2[:,:,::-1])
75          plt.show()
76          t=t+int(num)
77
78      # raise NotImplementedError("To be implemented")
79      # End your code (Part 4)
```

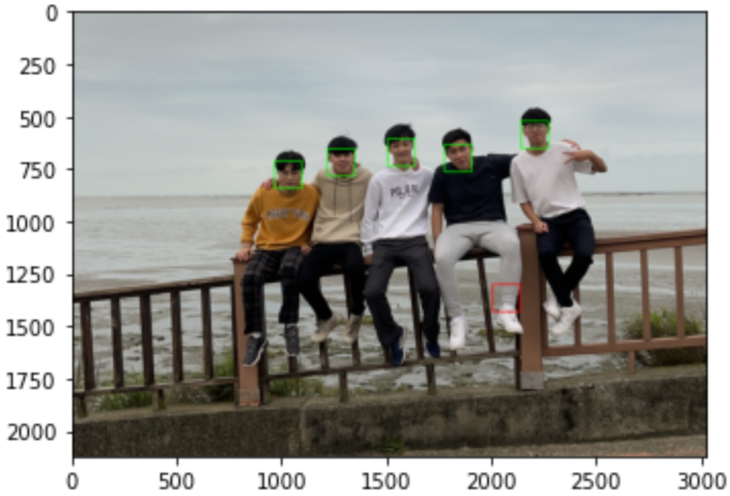
Part II. Results & Analysis (12%):



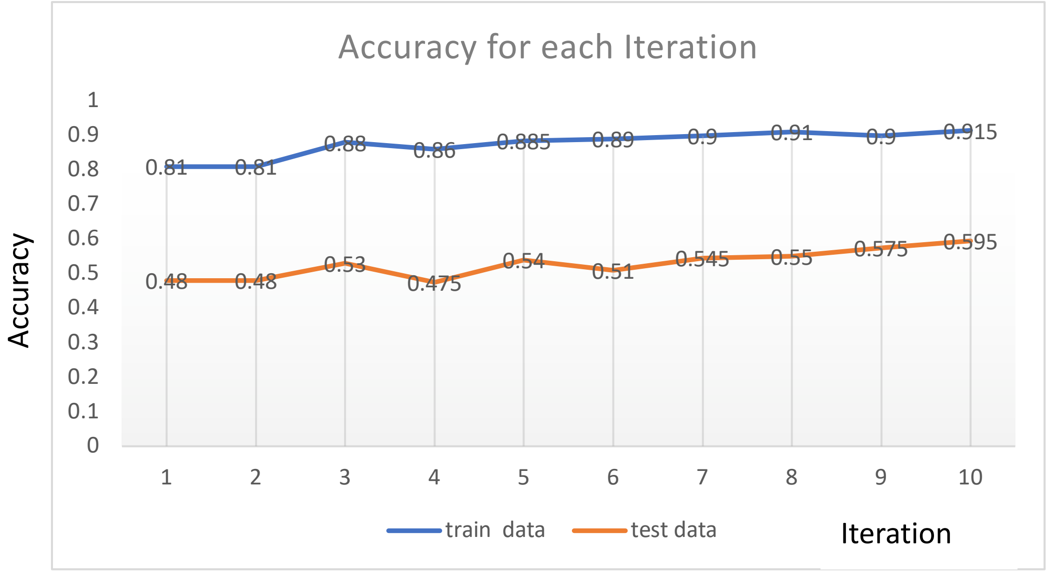
```
2, 10, 2)], negative regions=[RectangleRegion(5, 4, 10, 2)]) with accuracy: 145.000000 and alpha:
0.719869
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(12
11, 5, 1)], negative regions=[RectangleRegion(12, 12, 5, 1)]) with accuracy: 72.000000 and alpha:
0.685227
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10
4, 1, 1)], negative regions=[RectangleRegion(9, 4, 1, 1)]) with accuracy: 152.000000 and alpha: 0.70779
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4,
9, 2, 2), RectangleRegion(2, 11, 2, 2)], negative regions=[RectangleRegion(2, 9, 2, 2),
RectangleRegion(4, 11, 2, 2)]) with accuracy: 137.000000 and alpha: 0.811201

Evaluate your classifier with training dataset
False Positive Rate: 17/100 (0.170000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 183/200 (0.915000)

Evaluate your classifier with test dataset
False Positive Rate: 45/100 (0.450000)
False Negative Rate: 36/100 (0.360000)
Accuracy: 119/200 (0.595000)

Detect faces at the assigned location using your classifier

Detect faces on your own images

In [102]:
```

For T=4





200 張	Train data accuracy	Test data accuracy
T=1	0.81%	0.48%
T=2	0.81%	0.48
T=3	0.88%	0.53%
T=4	0.86%	0.475%
T=5	0.885%	0.54%
T=6	0.89%	0.51%
T=7	0.9%	0.545%
T=8	0.91%	0.55%
T=9	0.9%	0.575%
T=10	0.915%	0.595%



Discussion:

From the table above, I observe that the higher Iteration is, the more accuracy it perform. I think it is because the weight will become more and more accuracy with more round done. Furthermore, I observe that the train data's accuracy is higher than test data's. I think it is because we use train data to calculate clf. It is reasonable that train data will be more accuracy than test data.

Part III. Answer the questions (12%):

1. Please describe a problem you encountered and how you solved it.

A: The first problem I met is that my error will be bigger than 1, which makes alpha become uncountable. After asking friends, we modified the function then solve it. Another problem is that I can't make img.shape become (19,19,1) in part1. After realizing the meaning, I just turn read image from RGB to Grayscale and solve it.

2. What are the limitations of the Viola-Jones' algorithm?

A: If there are so many people and their face are close or overlap to each other, Viola-Jones' algorithm might not work well.

3. Based on Viola-Jones' algorithm, how to improve the accuracy except increasing the training dataset and changing the parameter T?

A: We can increase the number of features, which means to reduce the region of rectangle to make more features.

4. Please propose another possible face detection method (no matter how good or bad, please come up with an idea). Please discuss the pros and cons of the idea you proposed, compared to the Adaboost algorithm.

A: My idea is that using the colored difference to do face detection. For example, we can choose one point and record the color difference around the point and we get difference from a lot of face images and get the average value. Then, compare unknown images to the data. If unknown image's difference is similar to our data, it may be face image. For pros, it may be more accuracy than adaboost. For cons, it may be more time-consuming.