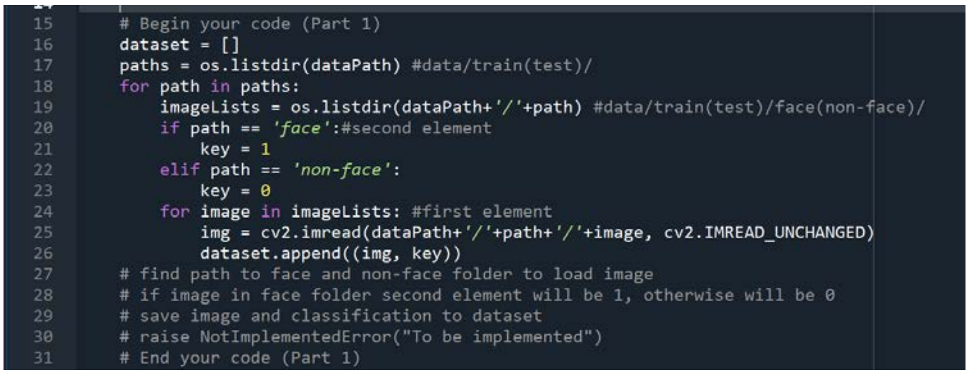
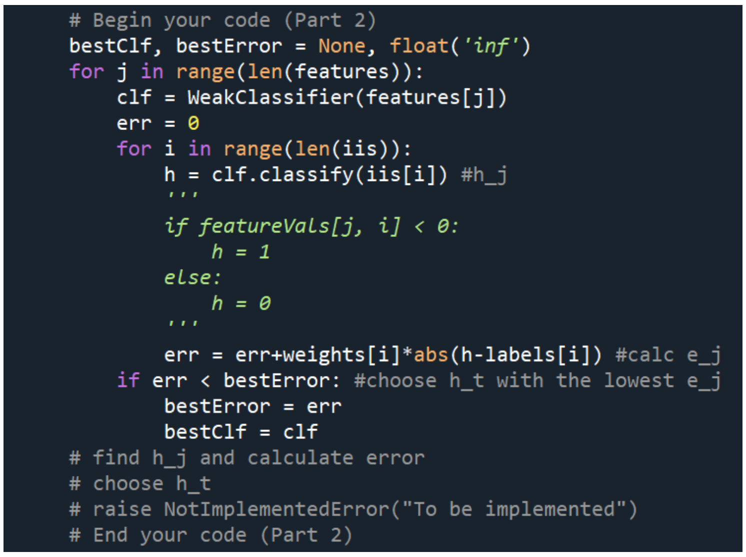
Homework 1: Face Detection

Part I. Implementation (6%):

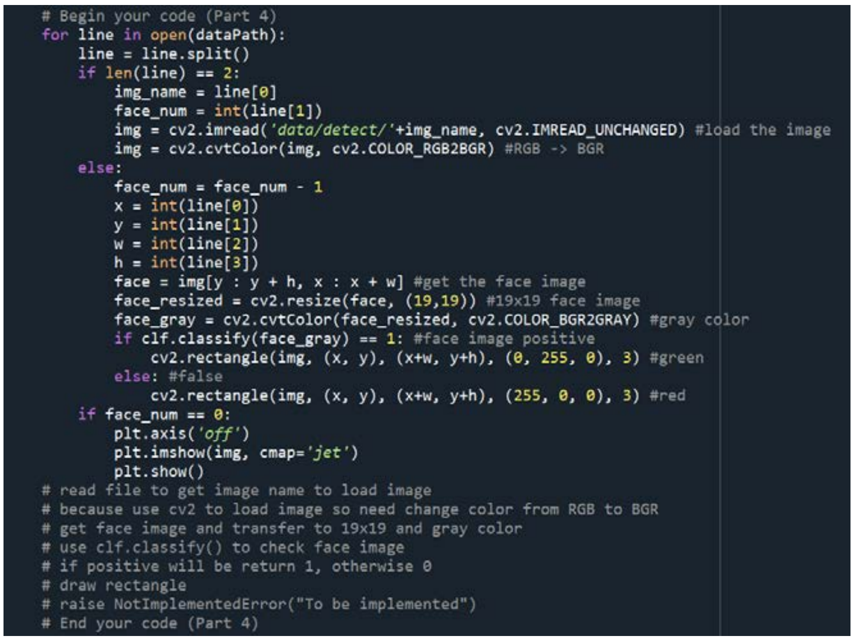
● Please screenshot your code snippets of Part 1, Part 2, Part 4, and explain your implementation.



```python
    # Begin your code (Part 1)
    dataset = []
    paths = os.listdir(dataPath) #data/train(test)/
    for path in paths:
        imageLists = os.listdir(dataPath+'/'+path) #data/train(test)/face(non-face)/
        if path == 'face':#second element
            key = 1
        elif path == 'non-face':
            key = 0
        for image in imageLists: #first element
            img = cv2.imread(dataPath+'/'+path+'/'+image, cv2.IMREAD_UNCHANGED)
            dataset.append((img, key))
    # find path to face and non-face folder to load image
    # if image in face folder second element will be 1, otherwise will be 0
    # save image and classification to dataset
    # raise NotImplementedError("To be implemented")
    # End your code (Part 1)
```



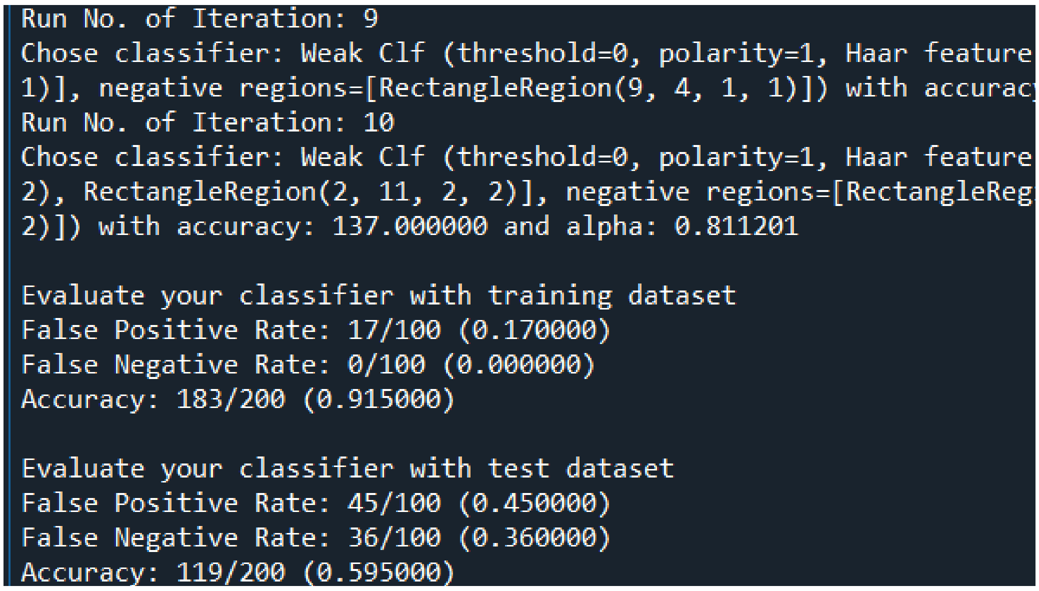
```python
    # Begin your code (Part 2)
    bestClf, bestError = None, float('inf')
    for j in range(len(features)):
        clf = WeakClassifier(features[j])
        err = 0
        for i in range(len(iis)):
            h = clf.classify(iis[i]) #h_j
            '''
            if featureVals[j, i] < 0:
                h = 1
            else:
                h = 0
            '''
            err = err+weights[i]*abs(h-labels[i]) #calc e_j
        if err < bestError: #choose h_t with the lowest e_j
            bestError = err
            bestClf = clf
    # find h_j and calculate error
    # choose h_t
    # raise NotImplementedError("To be implemented")
    # End your code (Part 2)
```



```python
    # Begin your code (Part 4)
    for line in open(dataPath):
        line = line.split()
        if len(line) == 2:
            img_name = line[0]
            face_num = int(line[1])
            img = cv2.imread('data/detect/'+img_name, cv2.IMREAD_UNCHANGED) #load the image
            img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #RGB -> BGR
        else:
            face_num = face_num - 1
            x = int(line[0])
            y = int(line[1])
            w = int(line[2])
            h = int(line[3])
            face = img[y : y + h, x : x + w] #get the face image
            face_resized = cv2.resize(face, (19,19)) #19x19 face image
            face_gray = cv2.cvtColor(face_resized, cv2.COLOR_BGR2GRAY) #gray color
            if clf.classify(face_gray) == 1: #face image positive
                cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 3) #green
            else: #false
                cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 3) #red
        if face_num == 0:
            plt.axis('off')
            plt.imshow(img, cmap='jet')
            plt.show()
    # read file to get image name to load image
    # because use cv2 to load image so need change color from RGB to BGR
    # get face image and transfer to 19x19 and gray color
    # use clf.classify() to check face image
    # if positive will be return 1, otherwise 0
    # draw rectangle
    # raise NotImplementedError("To be implemented")
    # End your code (Part 4)
```

Part II. Results & Analysis (12%):

- Please screenshot the results.

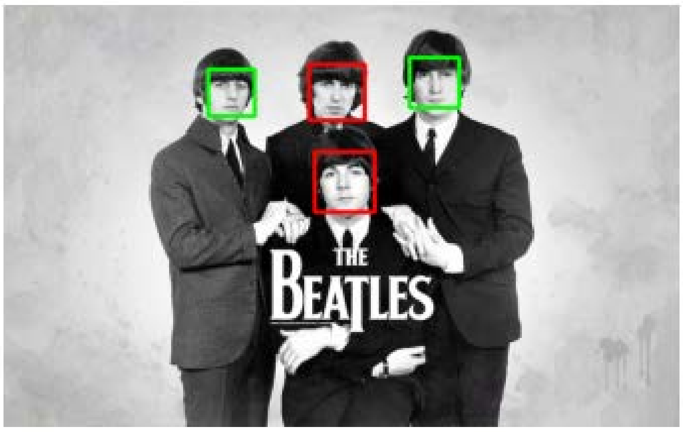


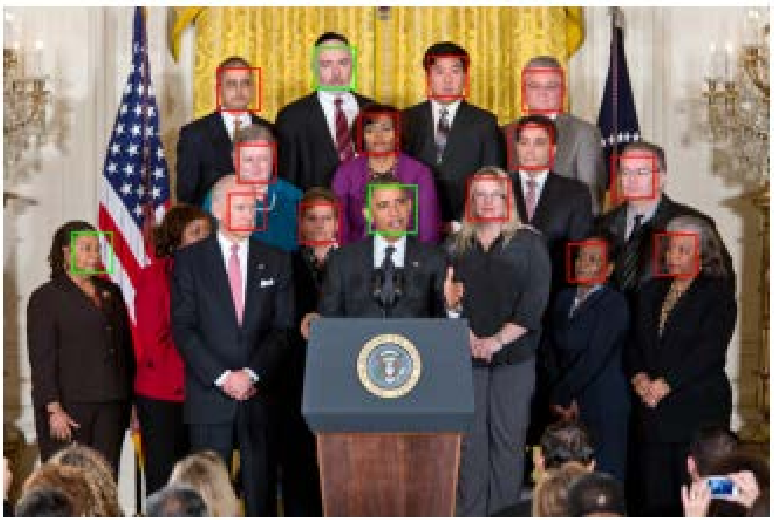
```
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature
1)], negative regions=[RectangleRegion(9, 4, 1, 1)]) with accurac
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature
2), RectangleRegion(2, 11, 2, 2)], negative regions=[RectangleReg
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)
```

T = 10:

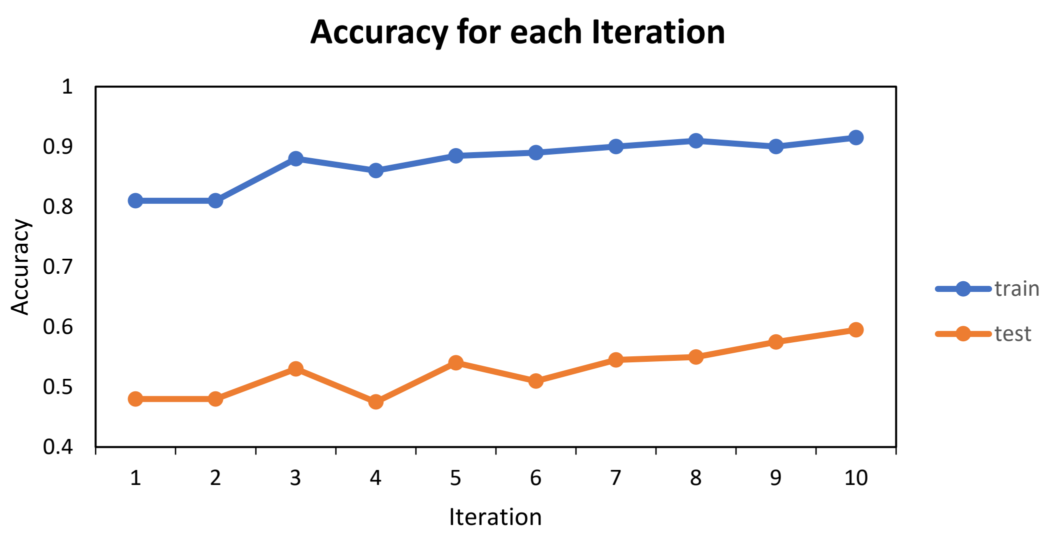




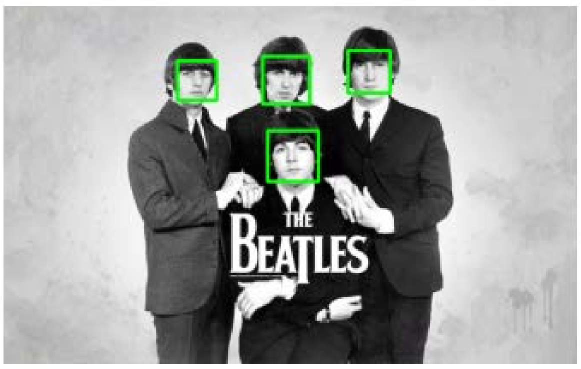
- Your analysis or observation.

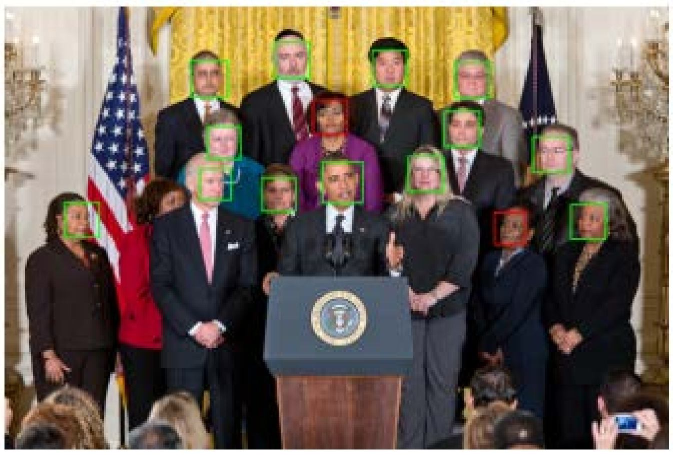
Please discuss the performance difference between the training and testing dataset, and present the results using a table or chart as follows.

200 張	Train data accuracy	Test data accuracy
method 1 t=1	81.00%	48.00%
method 1 t=2	81.00%	48.00%
method 1 t=3	88.00%	53.00%
method 1 t=4	86.00%	47.50%
method 1 t=5	88.50%	54.00%
method 1 t=6	89.00%	51.00%
method 1 t=7	90.00%	54.50%
method 1 t=8	91.00%	55.00%
method 1 t=9	90.00%	57.50%
method 1 t=10	91.50%	59.50%



T=1:





With T=1, face images are positive more than T=10.

Part III. Answer the questions (12%):

1. *Please describe a problem you encountered and how you solved it.*
- Use cv2 load image get wrong color: change RGB to BGR.

2. *What are the limitations of the Viola-Jones' algorithm?*
- It isn't as effective in detecting tilted or turned faces, affected by light condition.

3. *Based on Viola-Jones' algorithm, how to improve the accuracy except increasing the training dataset and changing the parameter T?*
- Reduce the size of the face need to be detected
- Increase the resolution of the image

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.*
- Random Forests algorithm.
- Random Forest is created using a bunch of decision trees which make use of different variables or features and makes use of bagging techniques for data sample.
- Random Forest has a better data sampling, Adaboost data sets get sampled repeatedly in the new data sample.
- Random Forest use decision trees make use multiple variables to make a final classification decision.