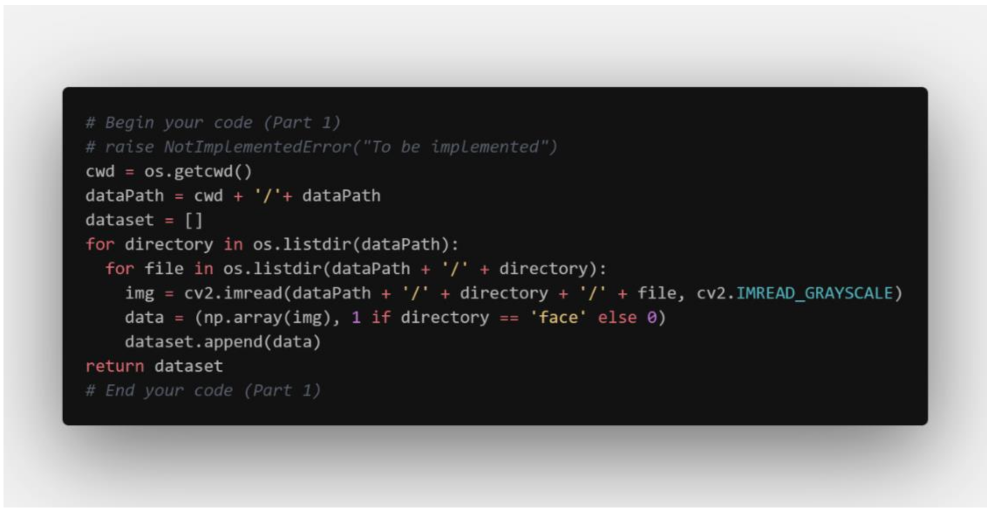
- Part I. Implementation
  - Part1

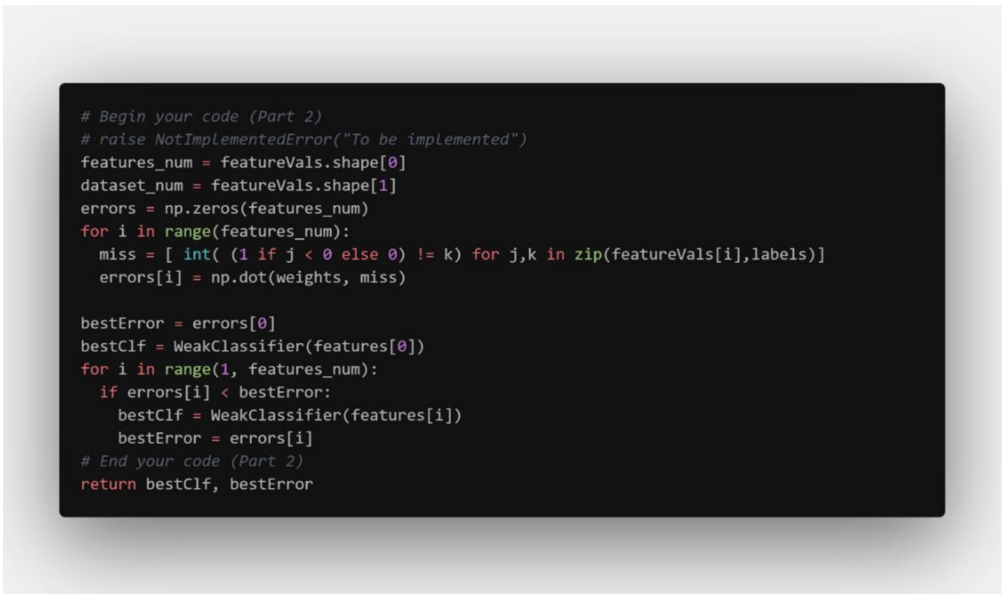
1. Get the path of the directory.
2. Go through every file in the directory.
3. Read the image in grayscale.
4. Collect the images and store into the dataset list.
5. Return the dataset.



```python
# Begin your code (Part 1)
# raise NotImplementedError("To be implemented")
cwd = os.getcwd()
dataPath = cwd + '/'+ dataPath
dataset = []
for directory in os.listdir(dataPath):
  for file in os.listdir(dataPath + '/' + directory):
    img = cv2.imread(dataPath + '/' + directory + '/' + file, cv2.IMREAD_GRAYSCALE)
    data = (np.array(img), 1 if directory == 'face' else 0)
    dataset.append(data)
return dataset
# End your code (Part 1)
```

- Part2

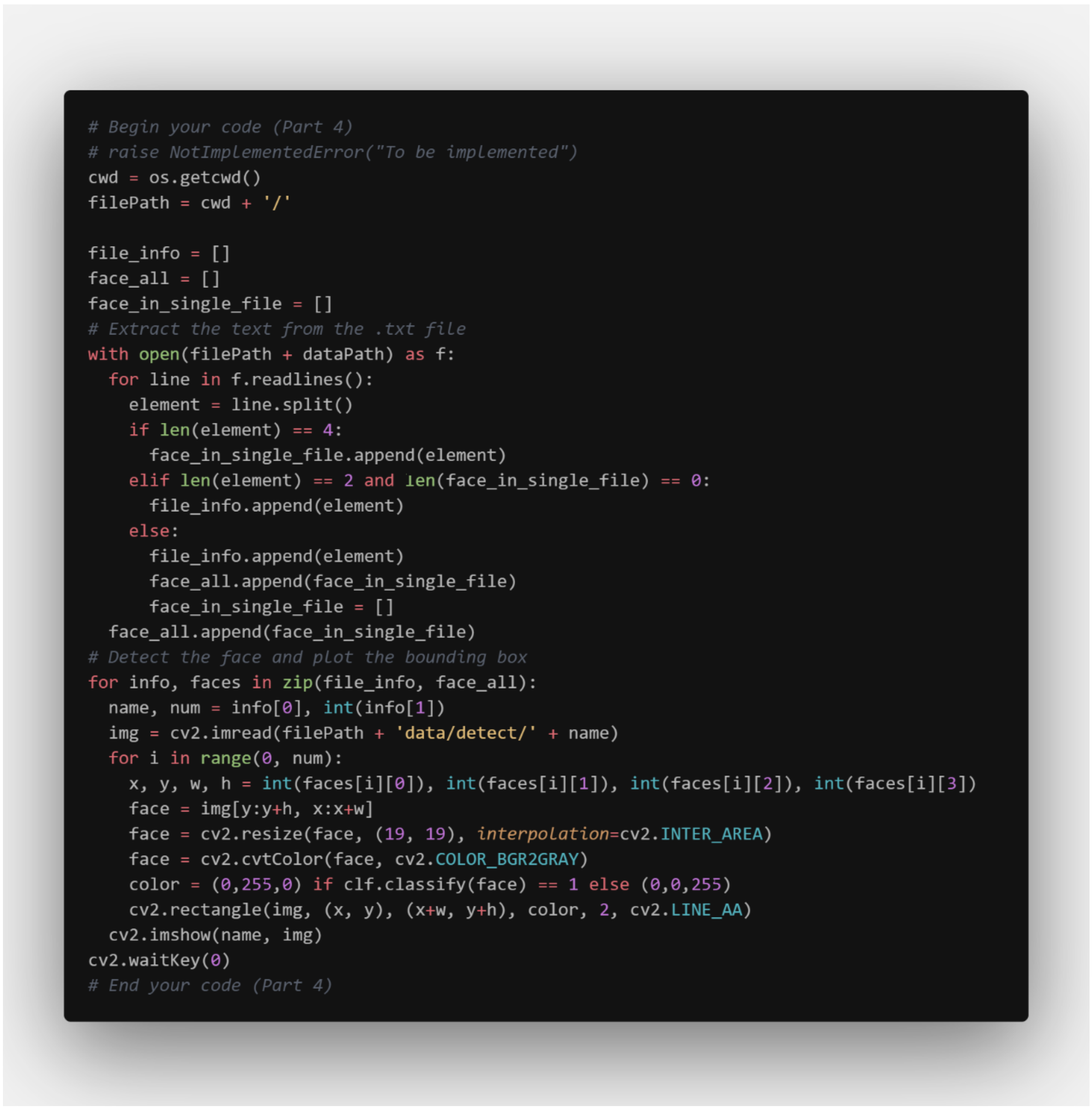
1. Set zero as threshold of the feature value, if it is negative, we consider it is the feature of face and therefore equals to one.
2. Calculate the amount of data that can be classified correctly by comparing the feature value with the corresponding label.
3. Calculate the score of error while taking the weight into consideration.
4. Select the best classifier which has the smallest error.



```python
# Begin your code (Part 2)
# raise NotImplementedError("To be implemented")
features_num = featureVals.shape[0]
dataset_num = featureVals.shape[1]
errors = np.zeros(features_num)
for i in range(features_num):
  miss = [ int( (1 if j < 0 else 0) != k) for j,k in zip(featureVals[i],labels)]
  errors[i] = np.dot(weights, miss)

bestError = errors[0]
bestClf = WeakClassifier(features[0])
for i in range(1, features_num):
  if errors[i] < bestError:
    bestClf = WeakClassifier(features[i])
    bestError = errors[i]
# End your code (Part 2)
return bestClf, bestError
```

■ Part4

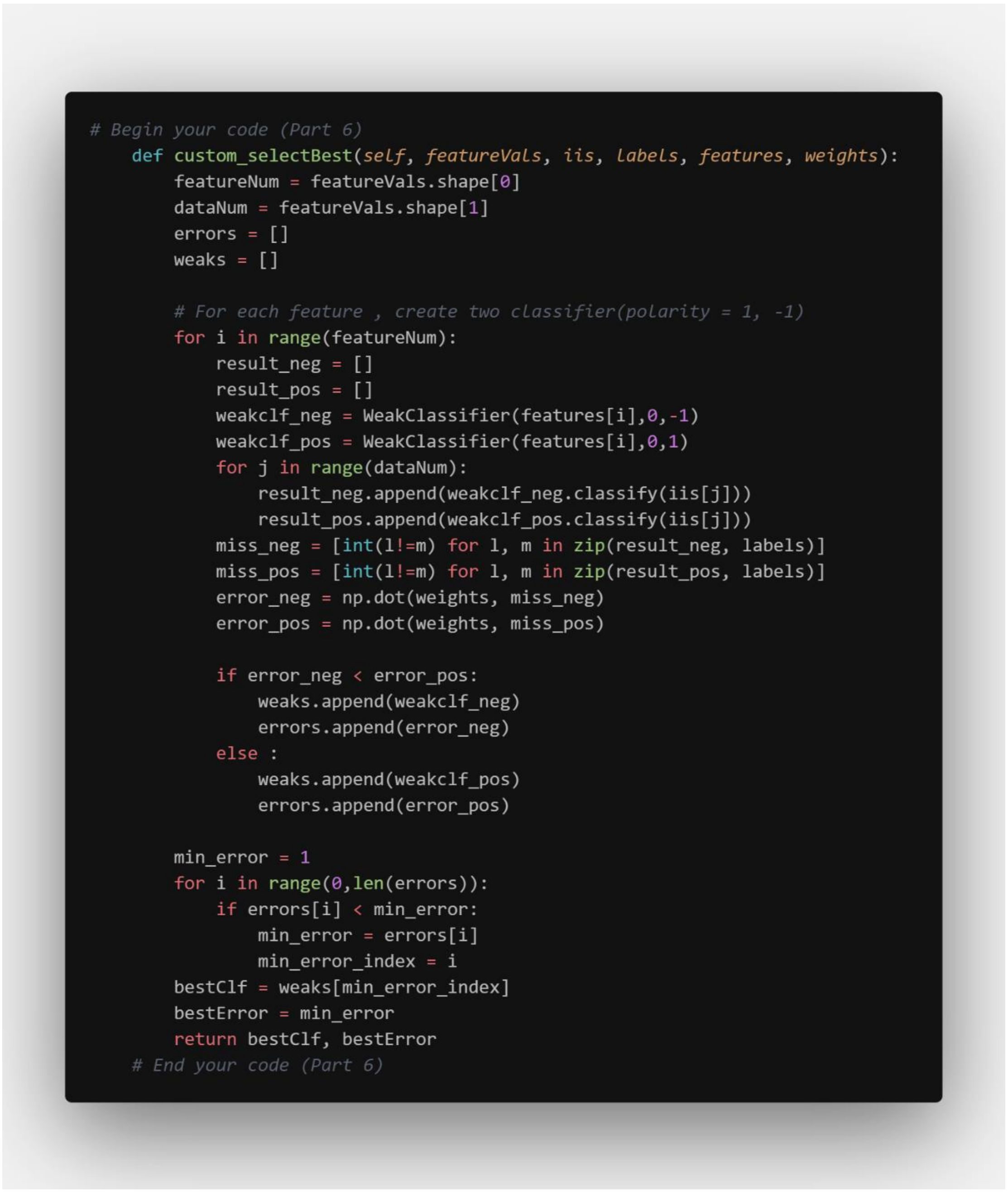
1. Read the text file in the folder.
2. Extract the name of the image file, coordinate, width, and length.
3. Read the image, resize to 19x19, and then convert into gray scale.
4. Use clf.classify() to detect whether the area is face.
5. Plot the bounding box on the face in color according to the result of step 4.
6. Show the image on screen.



```python
# Begin your code (Part 4)
# raise NotImplementedError("To be implemented")
cwd = os.getcwd()
filePath = cwd + '/'

file_info = []
face_all = []
face_in_single_file = []
# Extract the text from the .txt file
with open(filePath + dataPath) as f:
  for line in f.readlines():
    element = line.split()
    if len(element) == 4:
      face_in_single_file.append(element)
    elif len(element) == 2 and len(face_in_single_file) == 0:
      file_info.append(element)
    else:
      file_info.append(element)
      face_all.append(face_in_single_file)
      face_in_single_file = []
  face_all.append(face_in_single_file)
# Detect the face and plot the bounding box
for info, faces in zip(file_info, face_all):
  name, num = info[0], int(info[1])
  img = cv2.imread(filePath + 'data/detect/' + name)
  for i in range(0, num):
    x, y, w, h = int(faces[i][0]), int(faces[i][1]), int(faces[i][2]), int(faces[i][3])
    face = img[y:y+h, x:x+w]
    face = cv2.resize(face, (19, 19), interpolation=cv2.INTER_AREA)
    face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
    color = (0,255,0) if clf.classify(face) == 1 else (0,0,255)
    cv2.rectangle(img, (x, y), (x+w, y+h), color, 2, cv2.LINE_AA)
  cv2.imshow(name, img)
cv2.waitKey(0)
# End your code (Part 4)
```

- Part6

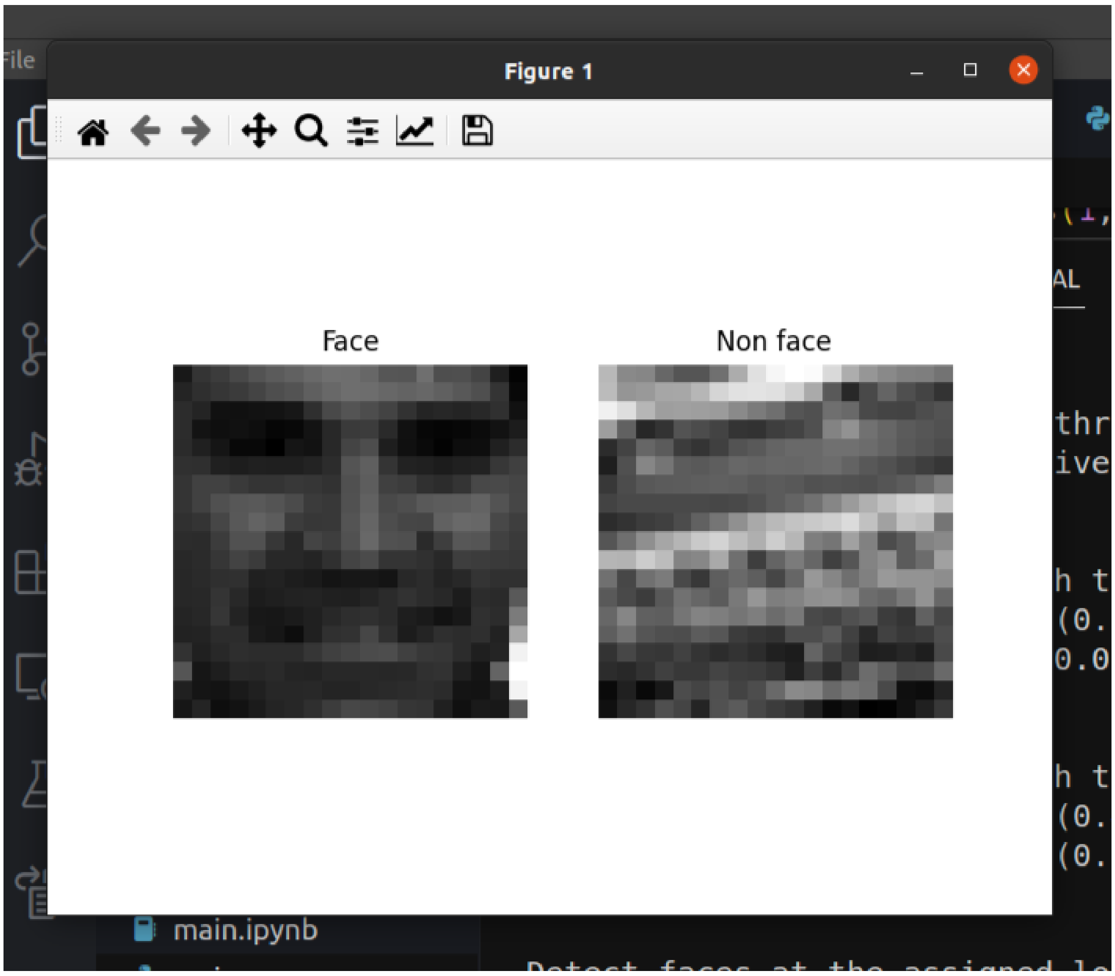
1. Create two weak classifiers for each feature.
2. Calculate the amount of data that can be classified correctly by comparing the return of the classifier with the corresponding label.
3. Calculate the score of error while taking the weight into consideration.
4. Select the best classifier which has the smallest error.



```python
# Begin your code (Part 6)
    def custom_selectBest(self, featureVals, iis, labels, features, weights):
        featureNum = featureVals.shape[0]
        dataNum = featureVals.shape[1]
        errors = []
        weaks = []

        # For each feature , create two classifier(polarity = 1, -1)
        for i in range(featureNum):
            result_neg = []
            result_pos = []
            weakclf_neg = WeakClassifier(features[i],0,-1)
            weakclf_pos = WeakClassifier(features[i],0,1)
            for j in range(dataNum):
                result_neg.append(weakclf_neg.classify(iis[j]))
                result_pos.append(weakclf_pos.classify(iis[j]))
            miss_neg = [int(l!=m) for l, m in zip(result_neg, labels)]
            miss_pos = [int(l!=m) for l, m in zip(result_pos, labels)]
            error_neg = np.dot(weights, miss_neg)
            error_pos = np.dot(weights, miss_pos)

            if error_neg < error_pos:
                weaks.append(weakclf_neg)
                errors.append(error_neg)
            else :
                weaks.append(weakclf_pos)
                errors.append(error_pos)

        min_error = 1
        for i in range(0,len(errors)):
            if errors[i] < min_error:
                min_error = errors[i]
                min_error_index = i
        bestClf = weaks[min_error_index]
        bestError = min_error
        return bestClf, bestError
    # End your code (Part 6)
```

- Part II. Results & Analysis
  - Part1

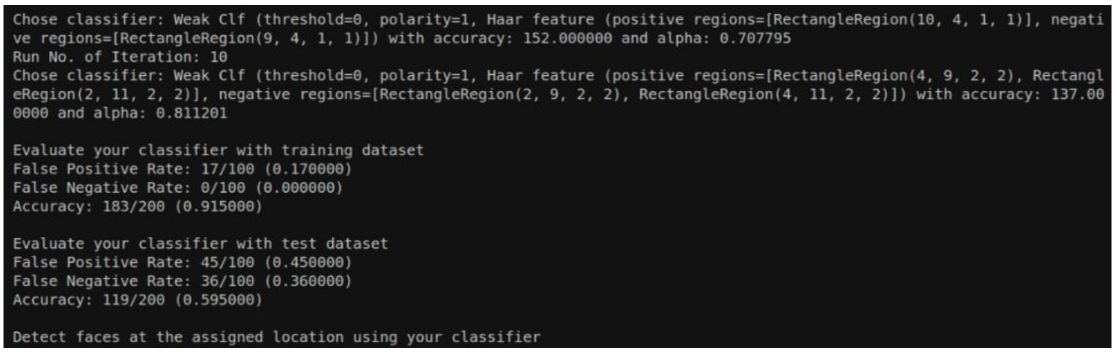
1. If Part1 is done correctly, we obtain the following Picture-1.



Picture-1

  - Part2

1. If Part2 is done correctly, we get the following Picture-2



```
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 4, 1, 1)], negati
ve regions=[RectangleRegion(9, 4, 1, 1)]) with accuracy: 152.000000 and alpha: 0.707795
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 9, 2, 2), Rectangl
eRegion(2, 11, 2, 2)], negative regions=[RectangleRegion(2, 9, 2, 2), RectangleRegion(4, 11, 2, 2)]) with accuracy: 137.00
0000 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
```

Picture-2

■ Part3

1. Testing T from 1 to 10, we obtain the following data.
2. The accuracies of both data are growing slightly in each generation and the accuracy of training data is higher about thirty percent than that of test data. In the first few generations, the accuracy of test data is about fifty percent, which means it can barely classify the data, and it reaches sixty percent in the end.
3. For the false positive rate, the rates of two data are decreasing slightly in each generation. And for the false negative rate, the rate of training data decrease to about zero after generation three. However, for the false negative rate of test data, it fluctuated and has a higher false rate compare to that of training data.
4. From the data, we can anticipate that the model trained is not good at distinguish the none face data.

200張	train data accuracy	test data accuracy
method 1 T = 1	81.0%	48.0%
method 1 T = 2	81.0%	48.0%
method 1 T = 3	88.0%	53.0%
method 1 T = 4	86.0%	47.5%
method 1 T = 5	88.5%	54.0%
method 1 T = 6	89.0%	51.0%
method 1 T = 7	90.0%	54.5%
method 1 T = 8	91.0%	55.0%
method 1 T = 9	90.0%	57.5%
method 1 T = 10	91.5%	59.5%

Chart-1

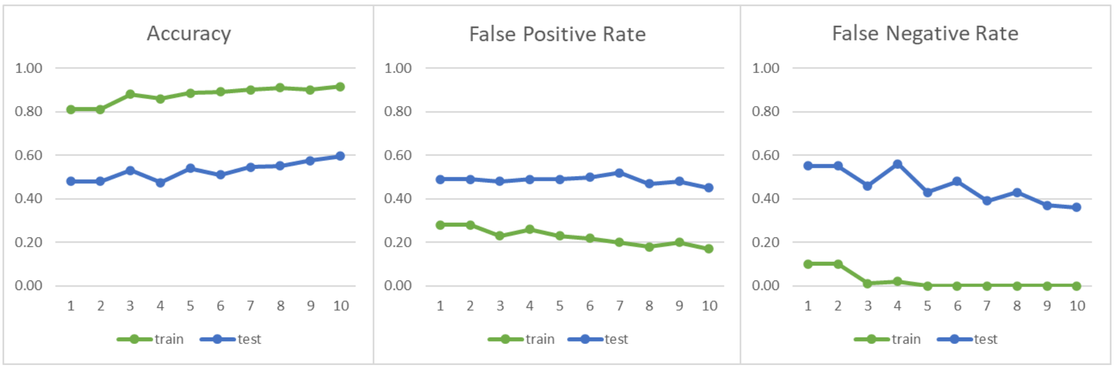
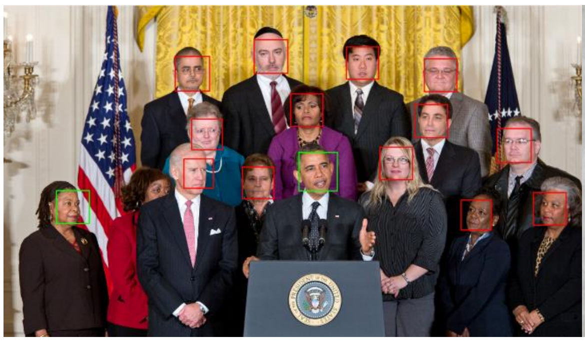
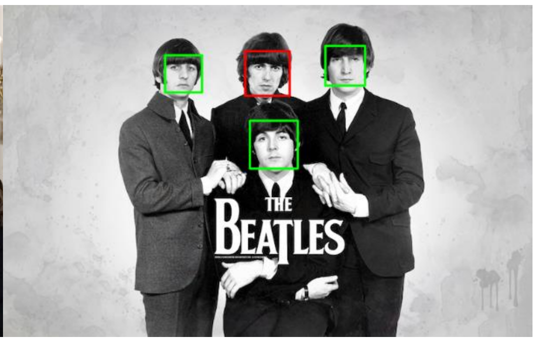


Chart-2

- Part4

1. If Part4 is done correctly, we will obtain the following Picture-3,4

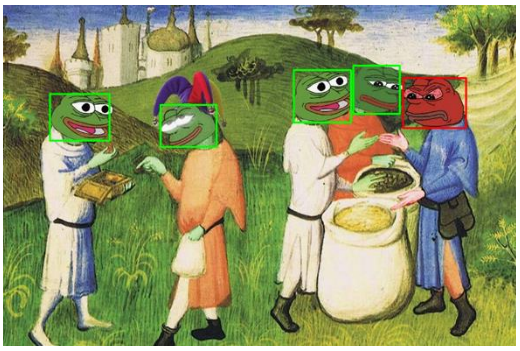

Picture-3

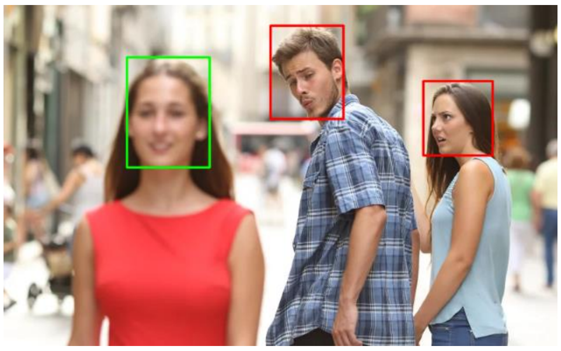


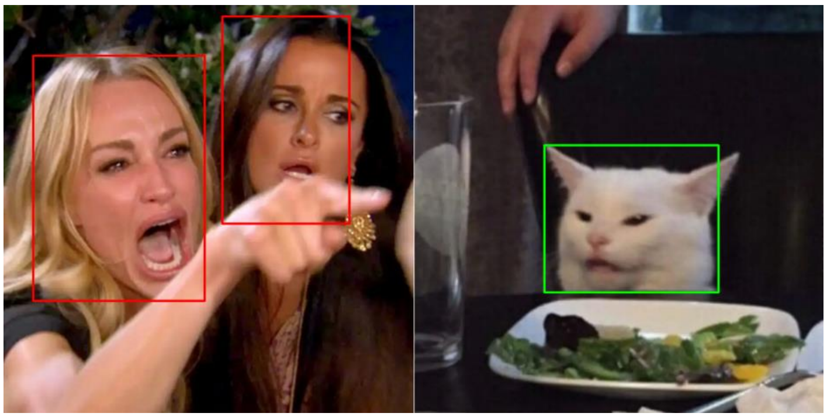
Picture-4

- Part5

1. If Part5 is done correctly, we will obtain the following Picture-5,6,7
2. From Picture-6, we found that it cannot distinguish side face, any little angle could cause the error. But if the face is in front view, it can be detected, even if it is a little obscure.

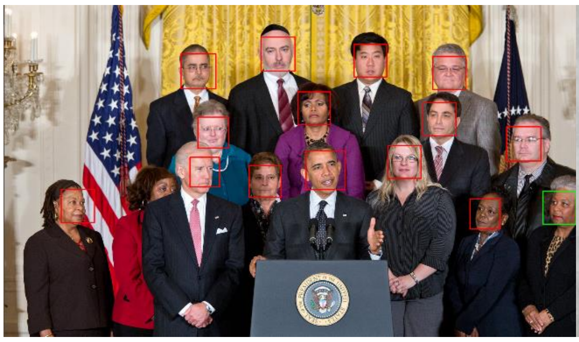

Picture-5


Picture-6


Picture-7

■ Part6

1. Test T from 1 to 10, using the custom select method, we obtain the following Picture-8,9 and Chart-3,4.
2. The accuracy of this method is higher than method about ten percent, however the performance on test picture is worse than method 1 and has a extreme low accuracy in Picture-8.
3. The false negative rate of test dataset of both method is fluctuated and is much higher than that of test data.
4. I think this problem is raised from how we select the best classifier in each generation. To optimize the method might be able to reduce the false rate.


Picture-8

Picture-9

200張	train data accuracy	test data accuracy
method 2 T = 1	81%	51%
method 2 T = 2	81%	58%
method 2 T = 3	85%	62%
method 2 T = 4	88%	59%
method 2 T = 5	94%	57%
method 2 T = 6	94%	60%
method 2 T = 7	96%	69%
method 2 T = 8	97%	61%
method 2 T = 9	99%	72%
method 2 T = 10	98%	72%

Chart-3

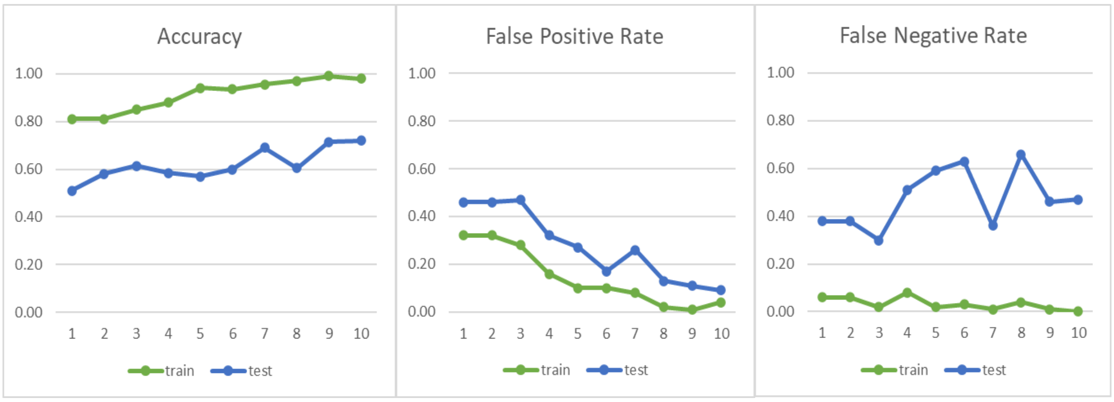


Chart-4

- Part III. Answer the questions (12%):

1. Please describe a problem you encountered and how you solved it.
    1. While calculating the score in part2, I did not realize that the weight is necessary to be taken into consideration. I struggle with the problem that the selectBest() return the same classifier in each generation which make the training process not taking effort for a long time.
    2. Until I decide to study the entire process of adaboost again, I finally realize that the adaboost update the weight depend on the best classifier, and the weight will be used in calculating the error.

2. What are the limitations of the Viola-Jones' algorithm?
    1. The Haar feature is based on changes in light, so detection accuracy can be affected by changes in light. The same object may have different detection results in different lighting environments.
    2. When a target is partially blocked or has a small area, it may not perform normally.
    3. The system is designed primarily for frontal faces, so the accuracy for data from other angles may be lower.

3. Based on Viola-Jones' algorithm, how to improve the accuracy except changing the training dataset and parameter T?
    1. Design a better method to choose classifier in each round, namely optimize part2.

4. Other than Viola-Jones' algorithm, 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.
    1. Yolo is an algorithm that uses convolutional neural networks for object detection, which has the advantages of high speed and high accuracy, but requires a large amount of training data.