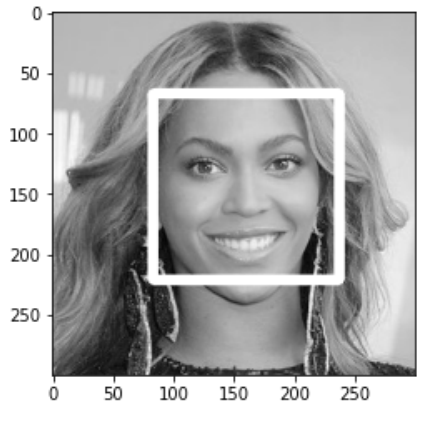
```
! pip install opencv-python
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dist-packages (4.1.2.30)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from opencv-python) (1.19.5)
```

```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
image1 = cv2.imread(r'/content/beyonce.jpg', 0)
image2 = cv2.imread(r'/content/dog.jpeg', 0)
plt.imshow(image1, cmap ='gray')
```

```
<matplotlib.image.AxesImage at 0x7fdd009b7f60>
```




```
cv2.data.haarcascades
```

```
'/usr/local/lib/python3.6/dist-packages/cv2/data/'
```

```
face_detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml
```

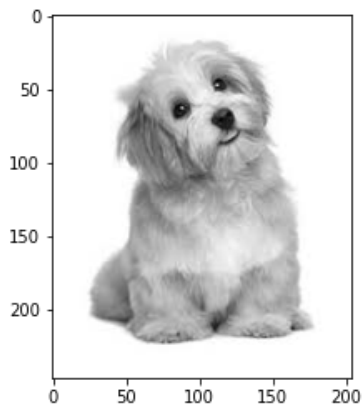
```
def detect_face (image):
  face_image = image.copy()
  face_rectangle = face_detector.detectMultiScale(face_image)
  face_rectangle = face_detector.detectMultiScale(face_image)
  for (x,y,width,height) in face_rectangle:
      cv2.rectangle(face_image, (x,y), (x + width, y+height), (255,255,255), 8)
  return face_image
detection_result = detect_face(image1)
plt.imshow(detection_result, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fdcff8e7588>
```



```
detection_result = detect_face(image2)
plt.imshow(detection_result, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fdcf6082e48>
```

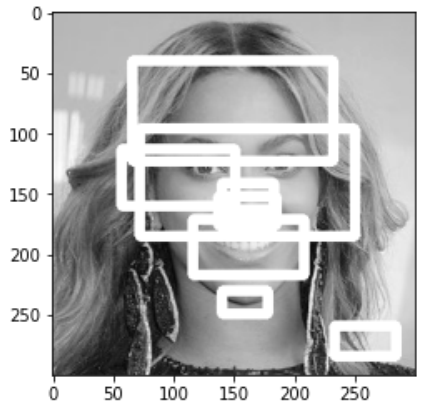



```
eye_detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
def detect_eye (image):
  face_image = image.copy()
  face_rectangle = eye_detector.detectMultiScale(face_image)
  face_rectangle = eye_detector.detectMultiScale(face_image)
  for (x,y,width,height) in face_rectangle:
      cv2.rectangle(face_image, (x,y), (x + width, y+height), (255,255,255), 8)
  return face_image
detection_result = detect_eye(image1)
plt.imshow(detection_result, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fdcf078b128>
```

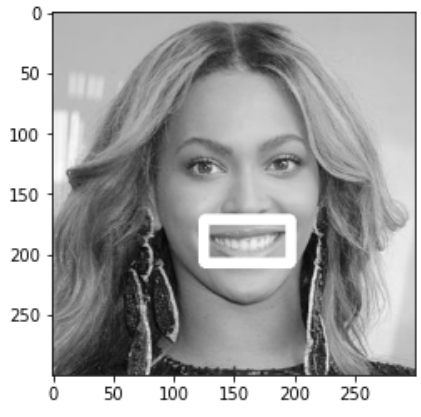

```
smile_detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_smile.xml')
def detect_smile (image):
  face_image = image.copy()
  face_rectangle = smile_detector.detectMultiScale(face_image)
  face_rectangle = smile_detector.detectMultiScale(face_image)
  for (x,y,width,height) in face_rectangle:
      cv2.rectangle(face_image, (x,y), (x + width, y+height), (255,255,255), 8)
  return face_image
detection_result = detect_smile(image1)
plt.imshow(detection_result, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fdcf0768278>
```




```
detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_smile.xml')
tect_smile (image):
_image = image.copy()
_rectangle = smile_detector.detectMultiScale(face_image, scaleFactor = 2.0, minNeighbors =20)
(x,y,width,height) in face_rectangle:
cv2.rectangle(face_image, (x,y), (x + width, y+height), (255,255,255), 8)
rn face_image
ion_result = detect_smile(image1)
show(detection_result, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fdcf06dca58>
```



✓ 0s completed at 8:16 PM