In [1]:
```
pip install PyWavelets
```

```
Requirement already satisfied: PyWavelets in c:\users\dhanusha\anaconda3\lib
\site-packages (1.4.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\dhanusha\anaconda3\l
ib\site-packages (from PyWavelets) (1.23.5)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\dhanusha\anaconda3\l
ib\site-packages (4.9.0.80)
Requirement already satisfied: numpy>=1.21.2 in c:\users\dhanusha\anaconda3\l
ib\site-packages (from opencv-python) (1.23.5)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\dhanusha\anaconda3\lib\sit
e-packages (0.12.2)
Requirement already satisfied: pandas>=0.25 in c:\users\dhanusha\anaconda3\li
b\site-packages (from seaborn) (1.5.3)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\dhanusha\anac
onda3\lib\site-packages (from seaborn) (1.23.5)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\dhanusha\a
naconda3\lib\site-packages (from seaborn) (3.7.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\dhanusha\anacond
a3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: cycler>=0.10 in c:\users\dhanusha\anaconda3\li
b\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\dhanusha\anaconda
3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dhanusha\anac
onda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\dhanusha\anaconda3\l
ib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dhanusha\anacond
a3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\dhanusha\anaconda3
\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (22.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\dhanusha\anaconda
3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in c:\users\dhanusha\anaconda3\li
b\site-packages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\dhanusha\anaconda3\lib\si
te-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.
16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [43]:

```
import numpy as np
import cv2
import matplotlib
from matplotlib import pyplot as plt
%matplotlib inline
```

In [44]:

```
img = cv2.imread("D:/Dhanusha VIT/SET project/Sports person Classifier/Model/t
img.shape
```

Out[44]:

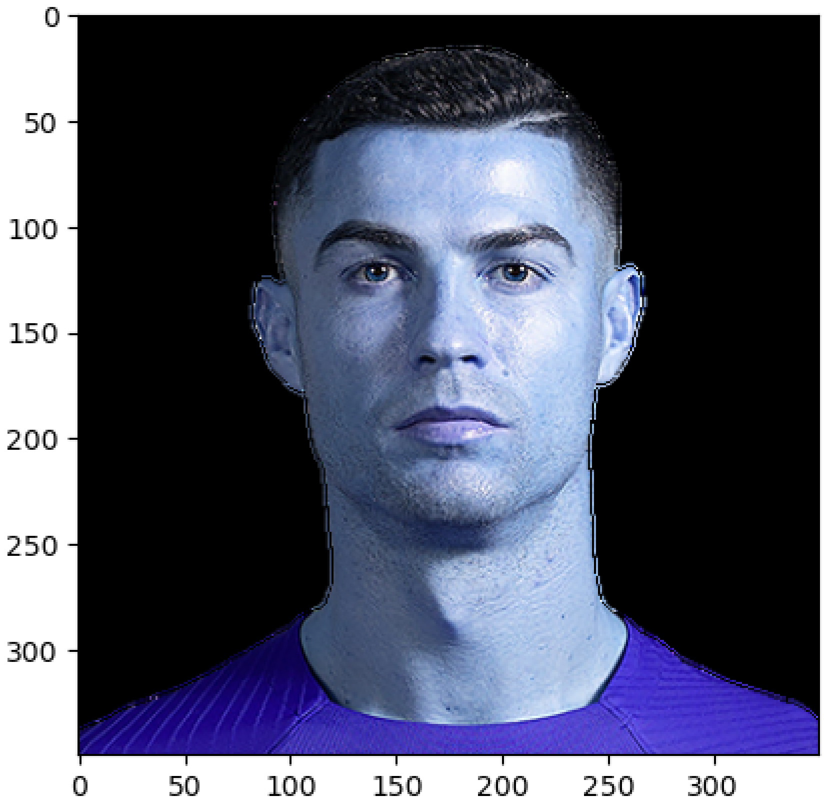
```
(350, 350, 3)
```

In [45]:

```
plt.imshow(img)
```

Out[45]:

```
<matplotlib.image.AxesImage at 0x20418073b80>
```



In [46]:

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray.shape
```

Out[46]:

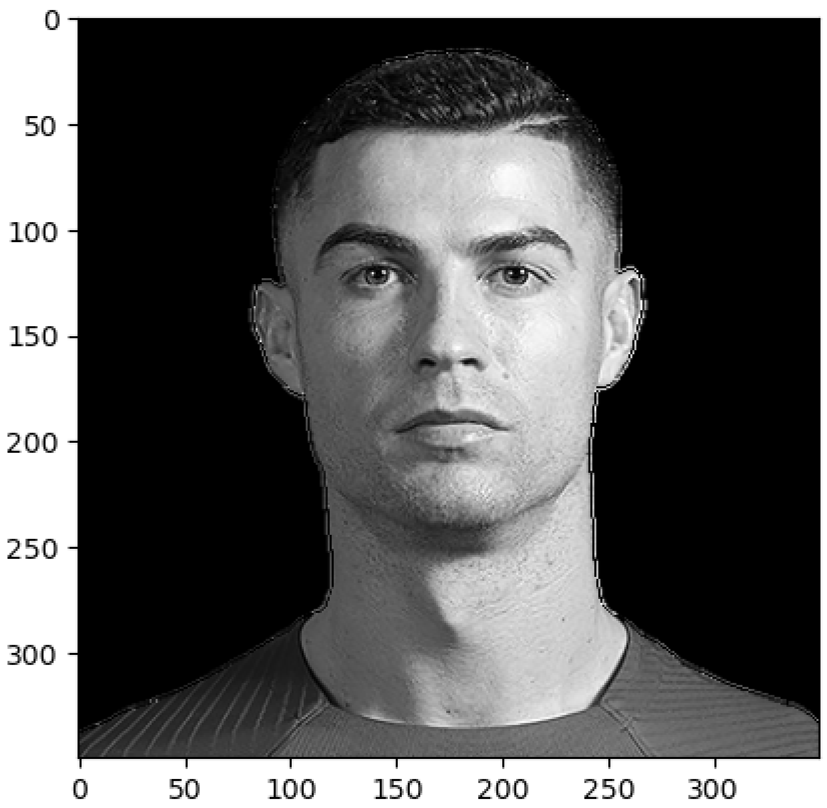
```
(350, 350)
```

In [47]:
```
gray
```

Out[47]:
```
array([[  0,   0,   0, ...,   0,   0,   0],
       [  0,   0,   0, ...,   0,   0,   0],
       [  0,   0,   0, ...,   0,   0,   0],
       ...,
       [ 42,  37,  29, ..., 106, 109, 104],
       [ 41,  30,  68, ..., 102, 101,  97],
       [ 32,  58,  86, ..., 106, 100, 104]], dtype=uint8)
```

In [48]:
```
 plt.imshow(gray, cmap ='gray')
```

Out[48]:
```
<matplotlib.image.AxesImage at 0x20418c4ad40>
```



In [49]:
```
face_cascade = cv2.CascadeClassifier("D:/Dhanusha VIT/SET project/Sports perso
eye_cascade = cv2.CascadeClassifier("D:/Dhanusha VIT/SET project/Sports person

faces = face_cascade.detectMultiScale(gray, 1.3, 5)
faces
```

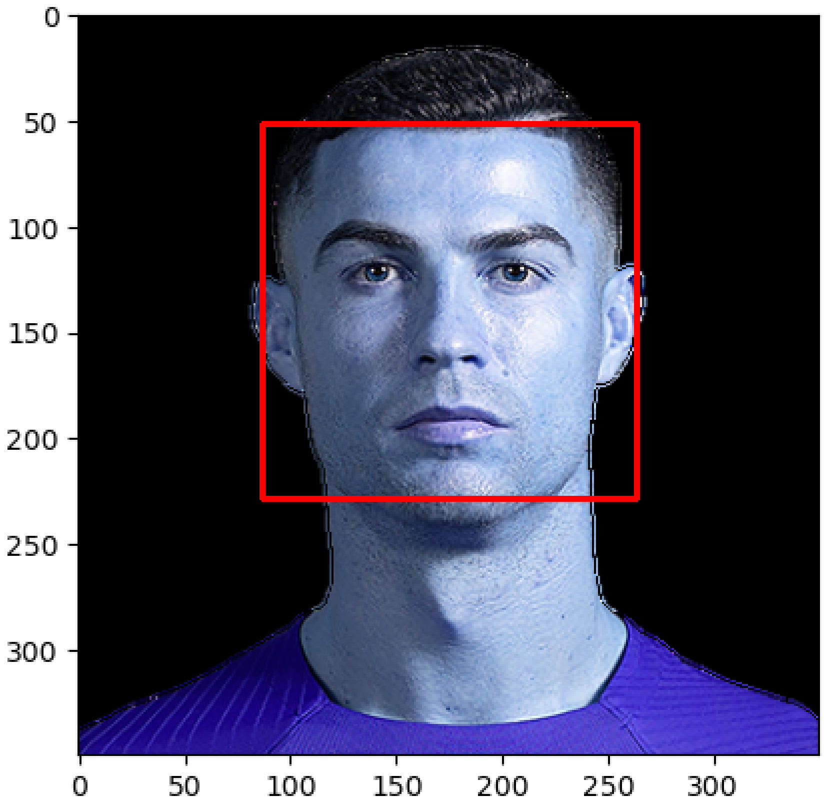
Out[49]:
```
array([[ 87,  52, 177, 177]])
```

In [50]:
```
(x,y,w,h) = faces[0]
x,y,w,h
```

Out[50]:
```
(87, 52, 177, 177)
```

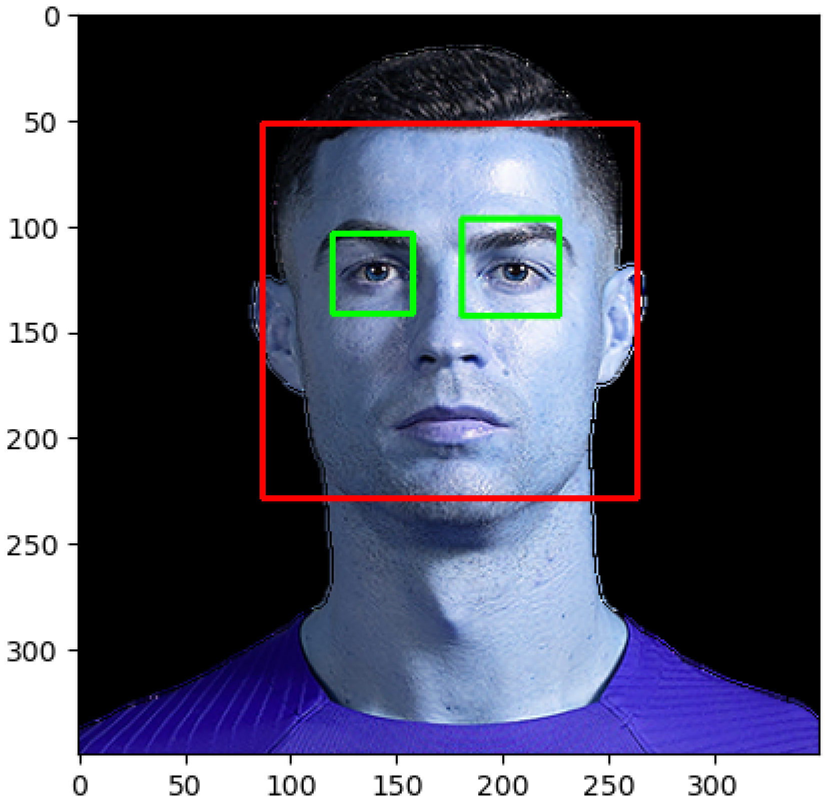
In [51]:
```
face_img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
plt.imshow(face_img)
```

Out[51]:
```
<matplotlib.image.AxesImage at 0x20418cc92d0>
```



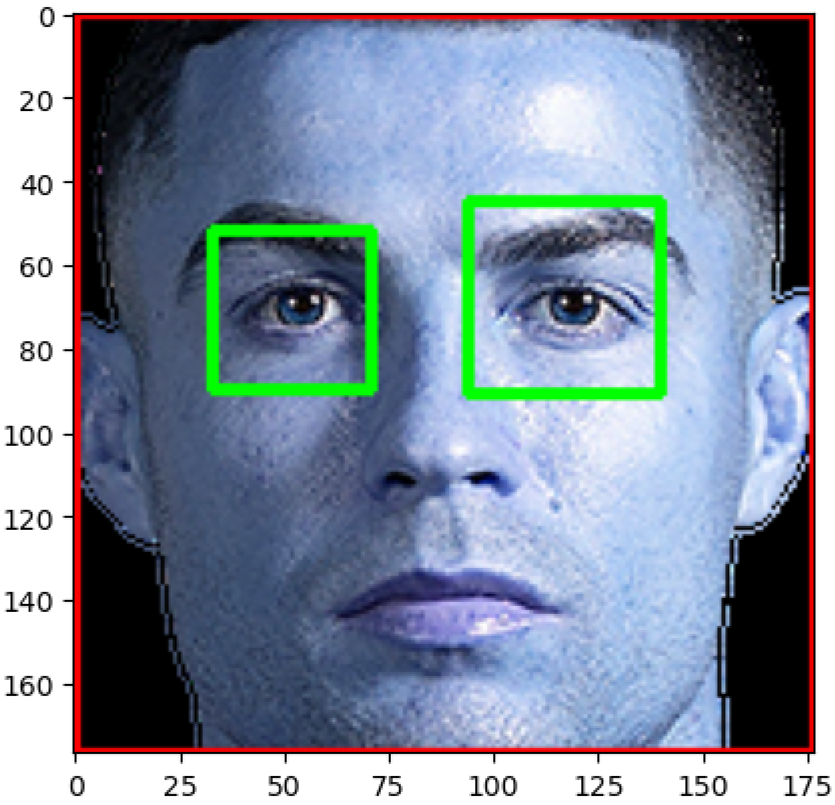
In [52]:

```
cv2.destroyAllWindows()
for (x,y,w,h) in faces:
    face_img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = face_img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)


plt.figure()
plt.imshow(face_img, cmap='gray')
plt.show()
```



```
In [53]: %matplotlib inline
         plt.imshow(roi_color, cmap='gray')
```

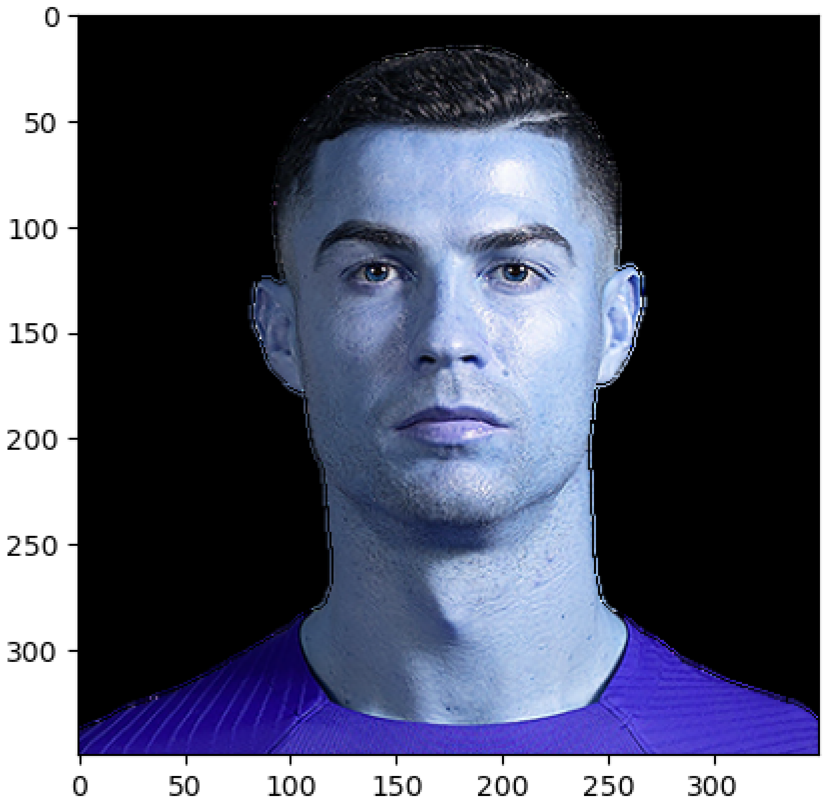
```
Out[53]: <matplotlib.image.AxesImage at 0x20418d32860>
```



```
In [54]: def get_cropped_image_if_2_eyes(image_path):
             img = cv2.imread(image_path)
             gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
             faces = face_cascade.detectMultiScale(gray, 1.3, 5)
             for (x,y,w,h) in faces:
                 roi_gray = gray[y:y+h, x:x+w]
                 roi_color = img[y:y+h, x:x+w]
                 eyes = eye_cascade.detectMultiScale(roi_gray)
                 if len(eyes) >= 2:
                     return roi_color
```

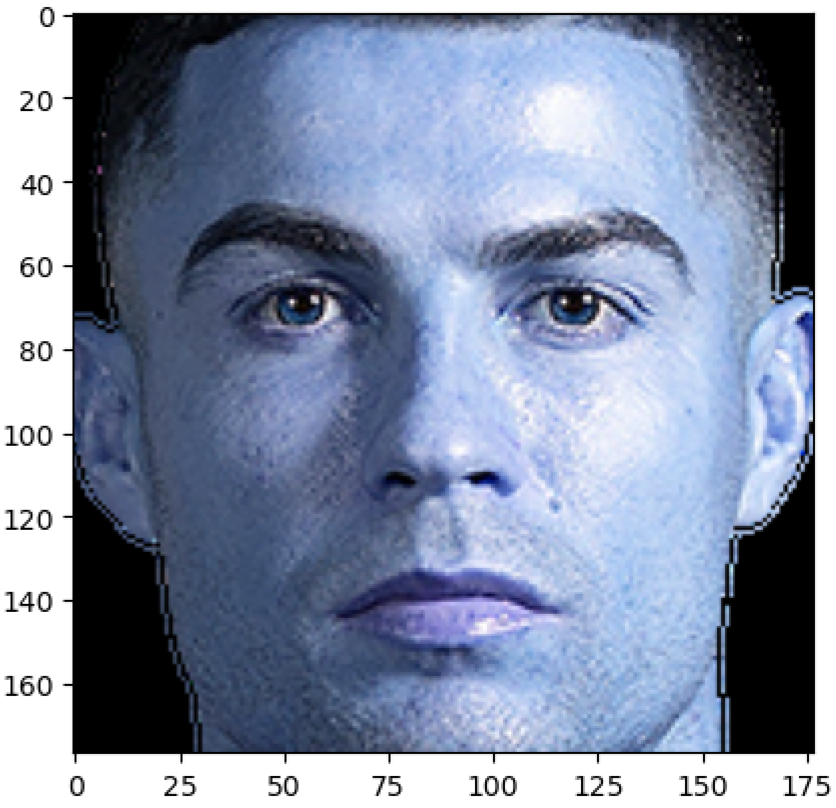
```
In [55]: original_image = cv2.imread('D:/Dhanusha VIT/SET project/Sports person Classif
         plt.imshow(original_image)
```

```
Out[55]: <matplotlib.image.AxesImage at 0x20418d15fc0>
```



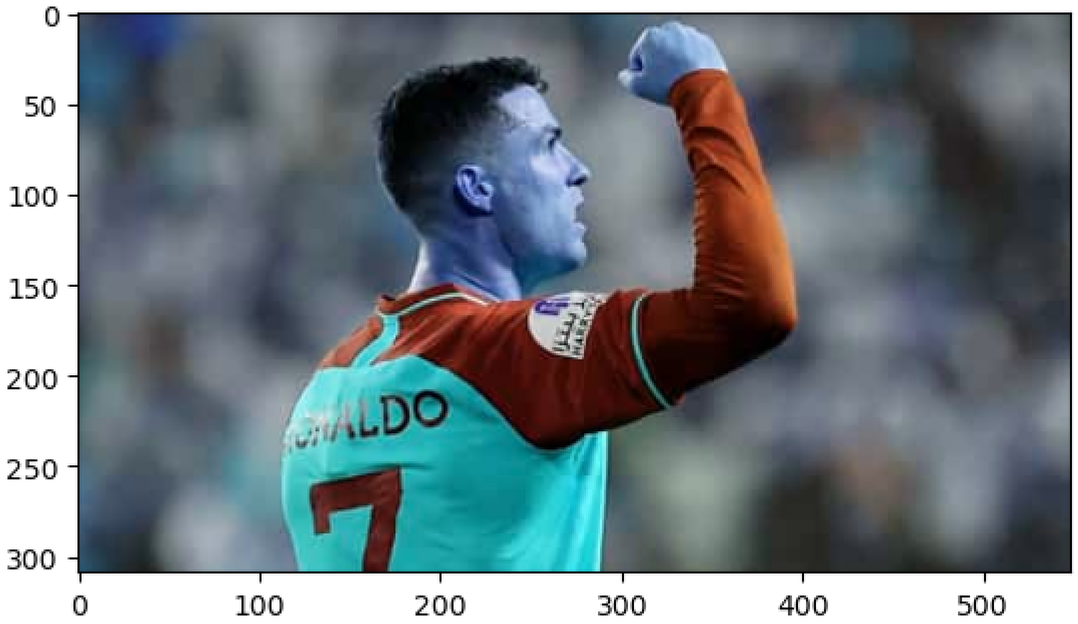
In [56]:
```
cropped_image = get_cropped_image_if_2_eyes('D:/Dhanusha VIT/SET project/Sport
plt.imshow(cropped_image)
```

Out[56]:
```
<matplotlib.image.AxesImage at 0x204193500d0>
```



In [58]:
```
org_image_obstructed = cv2.imread("D:/Dhanusha VIT/SET project/Sports person C
plt.imshow(org_image_obstructed)
```

Out[58]:
```
<matplotlib.image.AxesImage at 0x204193c0940>
```



In [16]:

```
cropped_image_no_2_eyes = get_cropped_image_if_2_eyes("D:/Dhanusha VIT/SET pro
cropped_image_no_2_eyes
```

In [17]:

```
path_to_data = "D:/Dhanusha VIT/SET project/Sports person Classifier/Model/dat
path_to_cr_data = "D:/Dhanusha VIT/SET project/Sports person Classifier/Model/
```

In [18]:

```
import os
img_dirs = []
for entry in os.scandir(path_to_data):
    if entry.is_dir():
        img_dirs.append(entry.path)

img_dirs
```

In [110]:

```
import shutil
if os.path.exists(path_to_cr_data):
     shutil.rmtree(path_to_cr_data)
os.mkdir(path_to_cr_data)
```

In [ ]:

```
cropped_image_dirs = []
celebrity_file_names_dict = {}

for img_dir in img_dirs:
    count = 1
    celebrity_name = img_dir.split('/')[-1]
    print(celebrity_name)

    celebrity_file_names_dict[celebrity_name] = []

    for entry in os.scandir(img_dir):
        roi_color = get_cropped_image_if_2_eyes(entry.path)
        if roi_color is not None:
            cropped_folder = path_to_cr_data + celebrity_name
            if not os.path.exists(cropped_folder):
                os.makedirs(cropped_folder)
                cropped_image_dirs.append(cropped_folder)
                print("Generating cropped images in folder: ",cropped_folder)

            cropped_file_name = celebrity_name + str(count) + ".png"
            cropped_file_path = cropped_folder + "/" + cropped_file_name

            cv2.imwrite(cropped_file_path, roi_color)
            celebrity_file_names_dict[celebrity_name].append(cropped_file_path
            count += 1
```

In [36]:

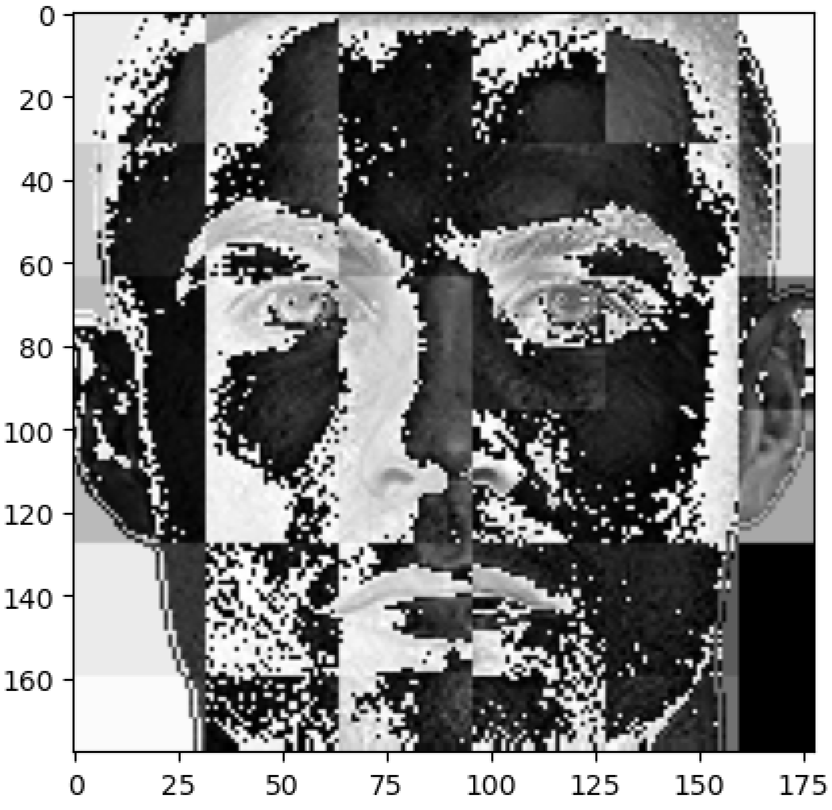
```
import numpy as np
import pywt
import cv2

def w2d(img, mode='haar', level=1):
    imArray = img
    #Datatype conversions
    #convert to grayscale
    imArray = cv2.cvtColor( imArray,cv2.COLOR_RGB2GRAY )
    #convert to float
    imArray =  np.float32(imArray)
    imArray /= 255;
    # compute coefficients
    coeffs=pywt.wavedec2(imArray, mode, level=level)

    #Process Coefficients
    coeffs_H=list(coeffs)
    coeffs_H[0] *= 0;

    # reconstruction
    imArray_H=pywt.waverec2(coeffs_H, mode);
    imArray_H *= 255;
    imArray_H =  np.uint8(imArray_H)

    return imArray_H
```

In [60]:

```
im_har = w2d(cropped_image,'db1',5)
plt.imshow(im_har, cmap='gray')
```

Out[60]:

```
<matplotlib.image.AxesImage at 0x20419450520>
```



In [ ]:

```
celebrity_file_names_dict
```

In [64]:

```
class_dict = {}
count = 0
for celebrity_name in celebrity_file_names_dict.keys():
    class_dict[celebrity_name] = count
    count = count + 1
class_dict
```

Out[64]:

```
{'Cristiano_Ronaldo': 0,
 'Kobe_Bryant': 1,
 'Maria_Sharapova': 2,
 'Novak_Djokovic': 3,
 'Virat_Kohli': 4}
```

In [65]:

```
X, y = [], []
for celebrity_name, training_files in celebrity_file_names_dict.items():
    for training_image in training_files:
        img = cv2.imread(training_image)
        scalled_raw_img = cv2.resize(img, (32, 32))
        img_har = w2d(img,'db1',5)
        scalled_img_har = cv2.resize(img_har, (32, 32))
        combined_img = np.vstack((scalled_raw_img.reshape(32*32*3,1),scalled_i
        X.append(combined_img)
        y.append(class_dict[celebrity_name])
```

In [66]:

```
len(X[0])
```

Out[66]:

```
4096
```

In [67]:

```
32*32*3 + 32*32
```

Out[67]:

```
4096
```

In [68]:

```
X[0]
```

Out[68]:

```
array([[172],
       [157],
       [140],
       ...,
       [  4],
       [  4],
       [  0]], dtype=uint8)
```

In [69]:

```
y[0]
```

Out[69]:

```
0
```

In [70]:
```
X = np.array(X).reshape(len(X),4096).astype(float)
X.shape
```

Out[70]:
```
(160, 4096)
```

In [71]:
```
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
```

In [72]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC(kernel = 'rbf', C =
pipe.fit(X_train, y_train)
pipe.score(X_test, y_test)
```

Out[72]:
```
0.625
```

In [73]:
```
print(classification_report(y_test, pipe.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.43      0.90      0.58        10
           1       0.83      0.56      0.67         9
           2       0.75      0.67      0.71         9
           3       1.00      0.33      0.50         9
           4       1.00      0.67      0.80         3

    accuracy                           0.62        40
   macro avg       0.80      0.62      0.65        40
weighted avg       0.76      0.62      0.63        40
```

In [74]:
```
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
```

In [75]:

```
model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto',probability=True),
        'params' : {
            'svc__C': [1,10,100,1000],
            'svc__kernel': ['rbf','linear']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'randomforestclassifier__n_estimators': [1,5,10]
        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='auto'),
        'params': {
            'logisticregression__C': [1,5,10]
        }
    }
}
```

In [76]:

```
scores = []
best_estimators = {}
import pandas as pd
for algo, mp in model_params.items():
    pipe = make_pipeline(StandardScaler(), mp['model'])
    clf =  GridSearchCV(pipe, mp['params'], cv=5, return_train_score=False)
    clf.fit(X_train, y_train)
    scores.append({
        'model': algo,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
    best_estimators[algo] = clf.best_estimator_

df = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df
```

Out[76]:

	model	best_score	best_params
0	svm	0.783333	{'svc__C': 1, 'svc__kernel': 'linear'}
1	random_forest	0.600000	{'randomforestclassifier__n_estimators': 10}
2	logistic_regression	0.758333	{'logisticregression__C': 1}

In [77]:

```
best_estimators
```

Out[77]:

```
{'svm': Pipeline(steps=[('standardscaler', StandardScaler()),
                 ('svc',
                  SVC(C=1, gamma='auto', kernel='linear', probability=Tru
e))]),
 'random_forest': Pipeline(steps=[('standardscaler', StandardScaler()),
                 ('randomforestclassifier',
                  RandomForestClassifier(n_estimators=10))]),
 'logistic_regression': Pipeline(steps=[('standardscaler', StandardScaler()),
                 ('logisticregression',
                  LogisticRegression(C=1, solver='liblinear'))])}
```

In [78]:

```
best_estimators['svm'].score(X_test,y_test)
```

Out[78]:

```
0.75
```

In [79]:

```
best_estimators['random_forest'].score(X_test,y_test)
```

Out[79]:

```
0.575
```

In [80]:

```
best_estimators['logistic_regression'].score(X_test,y_test)
```

Out[80]:

```
0.775
```

In [81]:

```
best_clf = best_estimators['svm']
```

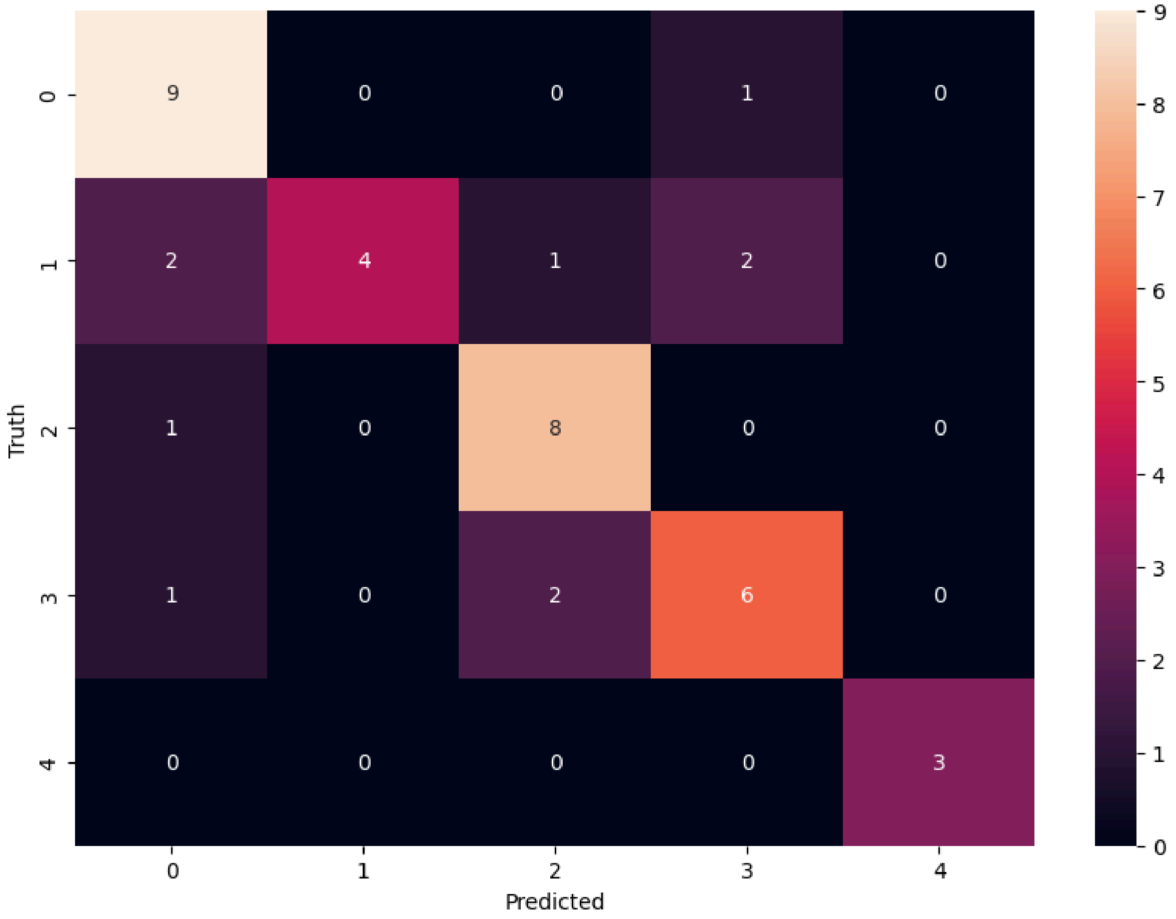
In [82]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, best_clf.predict(X_test))
cm
```

Out[82]:

```
array([[9, 0, 0, 1, 0],
       [2, 4, 1, 2, 0],
       [1, 0, 8, 0, 0],
       [1, 0, 2, 6, 0],
       [0, 0, 0, 0, 3]], dtype=int64)
```

```
In [83]: import seaborn as sn
         plt.figure(figsize = (10,7))
         sn.heatmap(cm, annot=True)
         plt.xlabel('Predicted')
         plt.ylabel('Truth')

Out[83]: Text(95.72222222222221, 0.5, 'Truth')
```



```
In [84]: class_dict

Out[84]: {'Cristiano_Ronaldo': 0,
          'Kobe_Bryant': 1,
          'Maria_Sharapova': 2,
          'Novak_Djokovic': 3,
          'Virat_Kohli': 4}
```

```
In [85]: !pip install joblib
         import joblib
         # Save the model as a pickle in a file
         joblib.dump(best_clf, 'saved_model.pkl')

         Requirement already satisfied: joblib in d:\josiah vit\anaconda3 app\lib\site
         -packages (1.1.1)

Out[85]: ['saved_model.pkl']
```

In [86]:

```
import json
with open("class_dictionary.json","w") as f:
    f.write(json.dumps(class_dict))
```