```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib
        import cv2
        from matplotlib import pyplot as plt
        %matplotlib inline
```

Preprocessing: Detect face and eyes

```
In [3]: import os
        os.getcwd()

Out[3]: 'C:\\Users\\kumar'
```

from IPython import display
display.Image("D:\CelebrityFaceRecognition\model\test_images\sharapova1.jpg")

from IPython import display
display.Image("D:\CelebrityFaceRecognition\model\test_images\sharapova1.jpg")

```
In [4]: img=cv2.imread("D:\CelebrityFaceRecognition\sharapova1.jpg")
```

```
In [5]: img.shape

Out[5]: (555, 700, 3)
```

```
In [6]: plt.imshow(img)
```

```
Out[6]: <matplotlib.image.AxesImage at 0x1fac662ad10>
```

```
In [7]: gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```
In [8]: gray
```

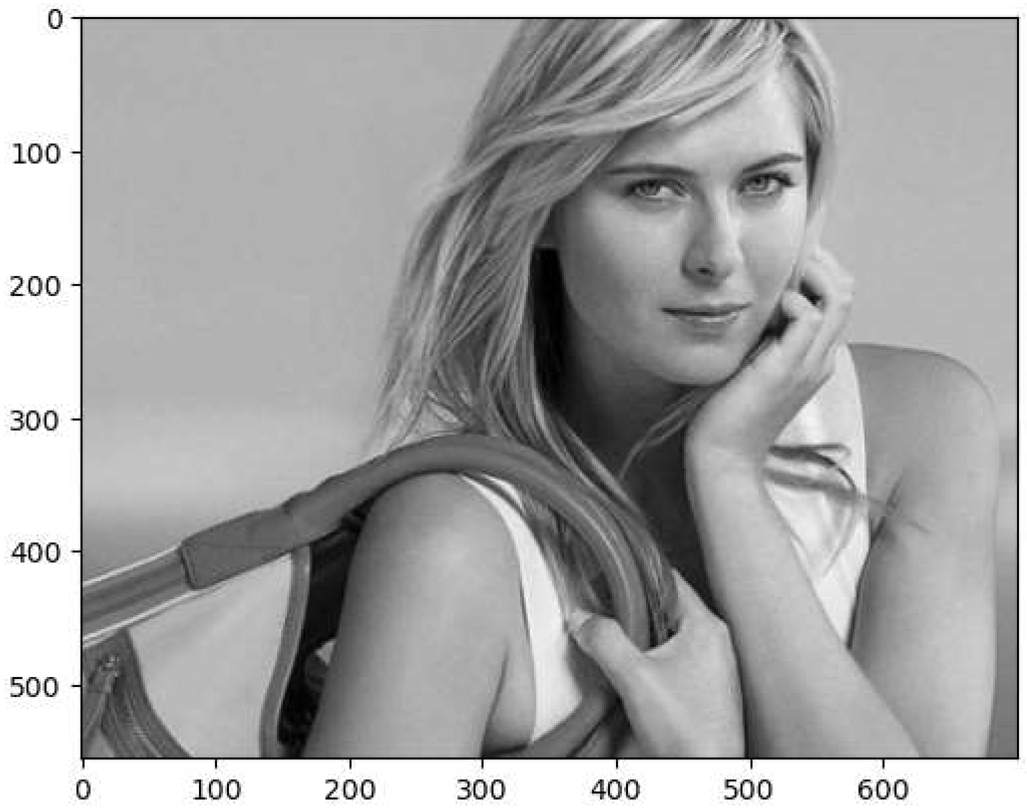
```
Out[8]: array([[175, 175, 175, ..., 176, 175, 174],
               [175, 175, 175, ..., 177, 175, 174],
               [175, 175, 175, ..., 177, 176, 174],
               ...,
               [ 84,  87,  88, ..., 113, 113, 113],
               [ 88,  89,  90, ..., 113, 113, 113],
               [ 93,  91,  91, ..., 112, 112, 112]], dtype=uint8)
```

In [9]:

```
plt.imshow(gray,cmap='gray')
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x1fac67bc3a0>
```



In [10]:

```
face_cascade = cv2.CascadeClassifier("D:\CelebrityFaceRecognition\model\opencv
eye_cascade=cv2.CascadeClassifier("D:\CelebrityFaceRecognition\model\opencv\ha

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

Out[10]:

```
array([[352,  38, 233, 233]])
```

In [11]:

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

Out[11]:

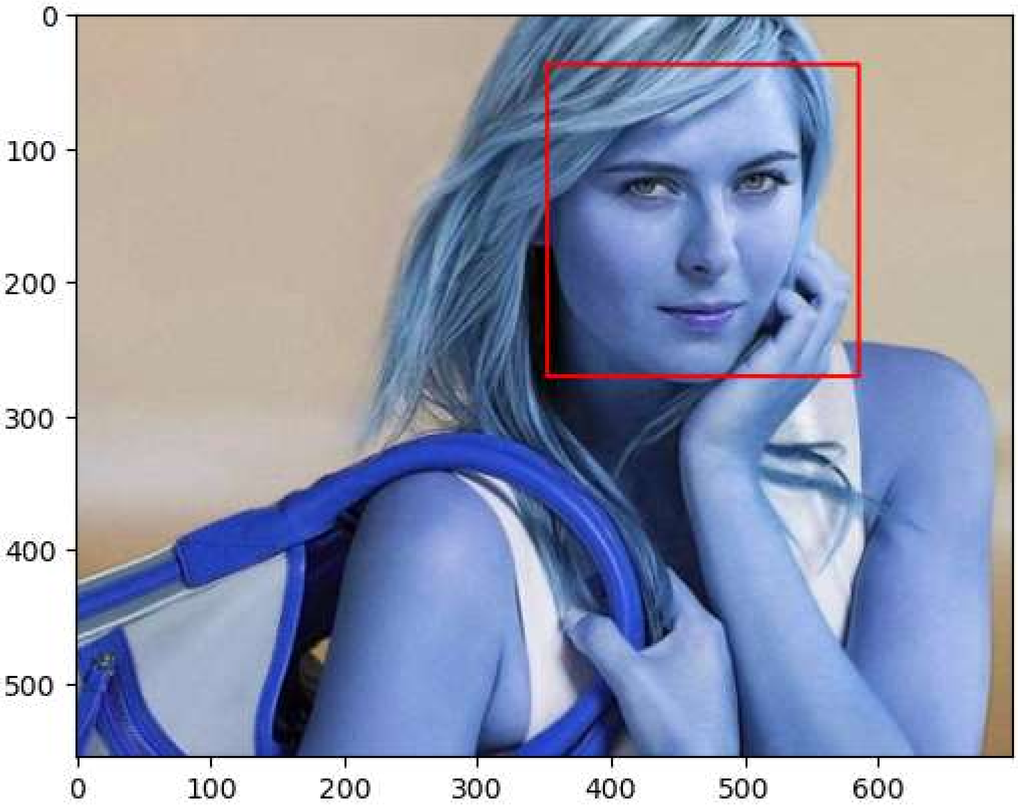
```
(352, 38, 233, 233)
```

In [12]:

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

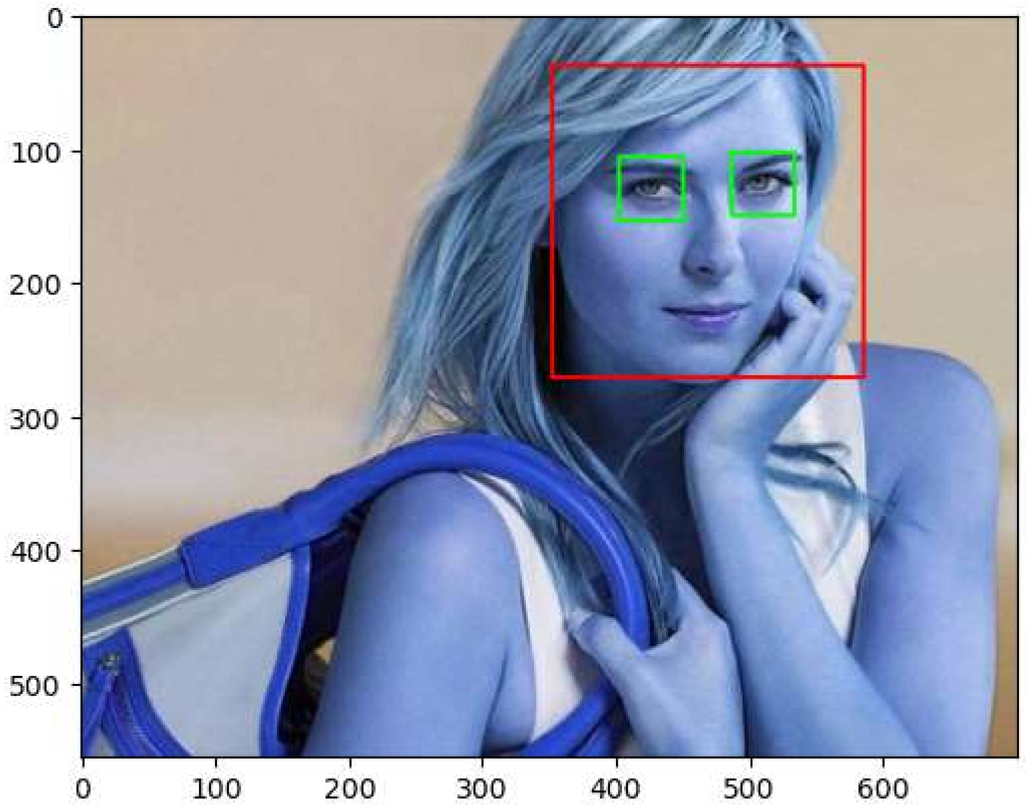
Out[12]:

```
<matplotlib.image.AxesImage at 0x1fac6aaa2c0>
```



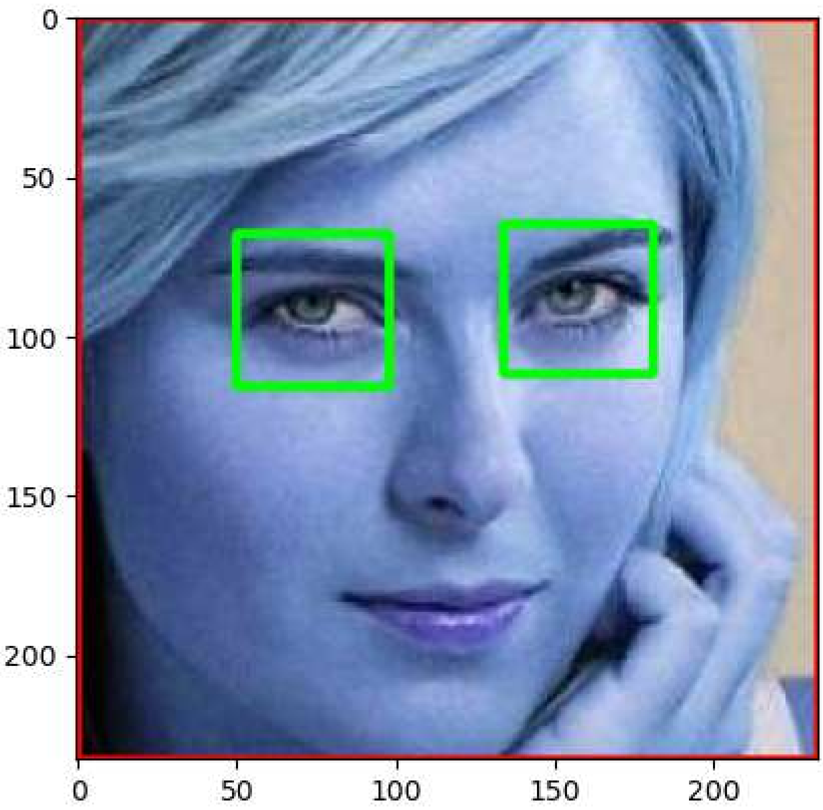
In [13]:

```
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()
```



Preprocessing: Crop the facial region of the image

```
In [14]: %matplotlib inline
         plt.imshow(roi_color, cmap='gray')

Out[14]: <matplotlib.image.AxesImage at 0x1fac6aa84f0>
```

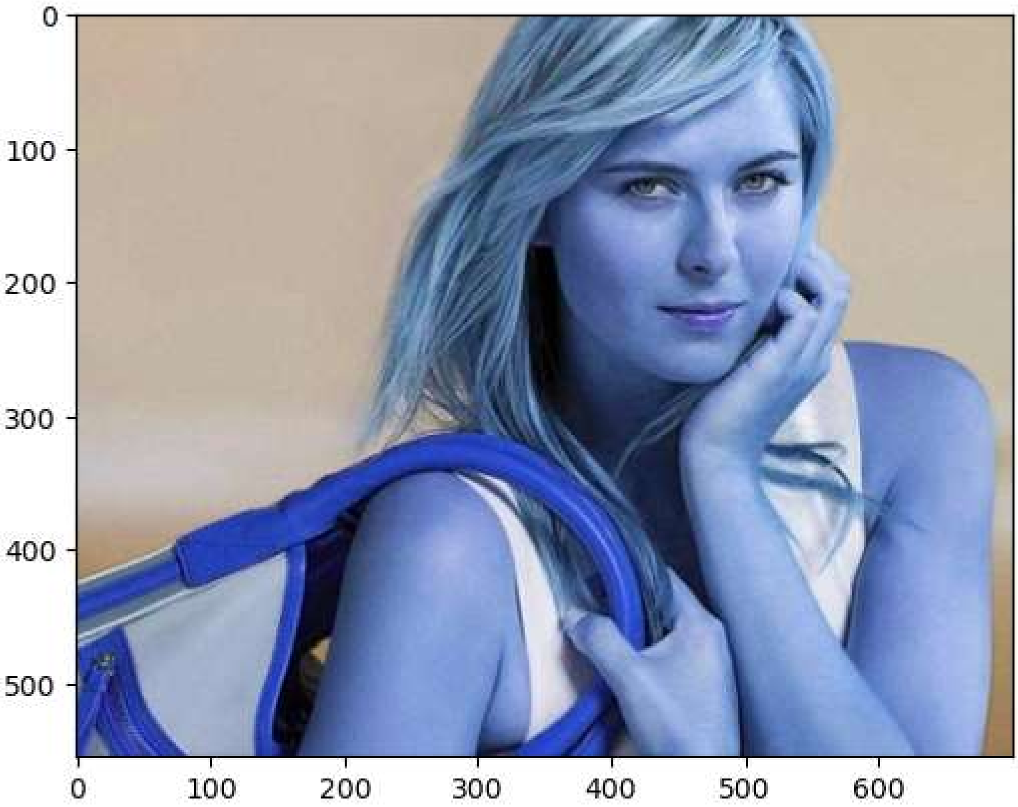


```
In [15]: cropped_img = np.array(roi_color)
         cropped_img.shape

Out[15]: (233, 233, 3)
```

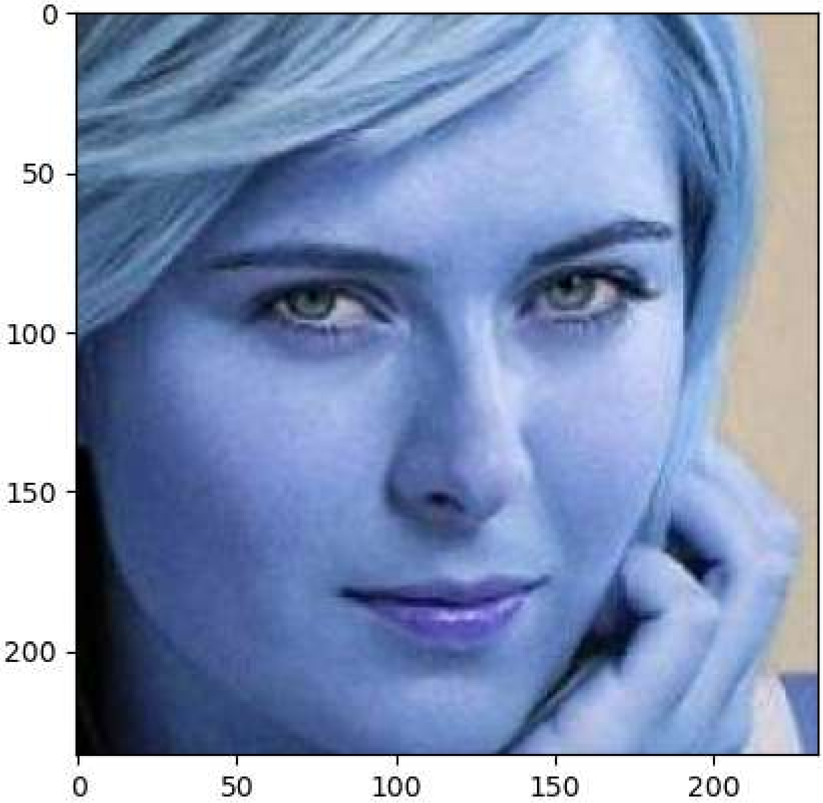
```
In [16]: def get_cropped_image_if_2_eyes(image_path):
             img=cv2.imread(image_path)
             gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
             faces=face_cascade.detectMultiScale(gray,1.3, 5)
             for(x,y,w,h) in faces:
                 roi_gray = gray[y:y+h, x:x+w]
                 roi_color = img[y:y+h, x:x+w]
                 eyes = eye_cascade.detectMultiScale(roi_gray)
                 if len(eyes)>=2:
                     return roi_color
```

```
In [17]: original_image=cv2.imread("D:\CelebrityFaceRecognition\sharapova1.jpg")
         plt.imshow(original_image)
```

```
Out[17]: <matplotlib.image.AxesImage at 0x1fad16992d0>
```



```
In [18]: cropped_image=get_cropped_image_if_2_eyes("D:\\CelebrityFaceRecognition\\shara
         plt.imshow(cropped_image)

Out[18]: <matplotlib.image.AxesImage at 0x1fad1835d50>
```



In [19]:

```
org_image_obstructed =cv2.imread("D:\CelebrityFaceRecognition\sharapova2.jpg")
plt.imshow(org_image_obstructed)
```

Out[19]:

```
<matplotlib.image.AxesImage at 0x1fad18a5ae0>
```

In [20]:

```
cropped_image_no_2_eyes=get_cropped_image_if_2_eyes("D:\CelebrityFaceRecogniti
cropped_image_no_2_eyes
```

In [21]:

```
path_to_data="D:\CelebrityFaceRecognition\model\dataset"
path_to_cropp_data="D:\CelebrityFaceRecognition\model\dataset\cropp"
```

In [22]:

```
import os
```

In [23]:

```
img_dirs=[]
for entry in os.scandir(path_to_data):
    if entry.is_dir(): #  check whether a given path is an existing directory
        img_dirs.append(entry.path)
```

In [24]:

```
img_dirs
```

Out[24]:

```
['D:\\CelebrityFaceRecognition\\model\\dataset\\lionel_messi',
 'D:\\CelebrityFaceRecognition\\model\\dataset\\maria_sharapova',
 'D:\\CelebrityFaceRecognition\\model\\dataset\\roger_federer',
 'D:\\CelebrityFaceRecognition\\model\\dataset\\serena_williams',
 'D:\\CelebrityFaceRecognition\\model\\dataset\\virat_kohli']
```

In [25]:

```
import shutil
if os.path.exists(path_to_cropp_data):
    shutil.rmtree(path_to_cropp_data) # delete the entire directory tree
os.mkdir(path_to_cropp_data)
```

In [26]:

```
{
    'lionel_messsi':[
        'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp\\messi\\messi1.pn
        'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp\\messi\messi2.png

    ],
     'virat_kohli':[
            'D:\CelebrityFaceRecognition\model\dataset\cropp\kohli\kohli1.png'
            'D:\CelebrityFaceRecognition\model\dataset\cropp\kohli\kohli2.png'


     ]
}
```

Out[26]:

```
{'lionel_messsi': ['D:\\CelebrityFaceRecognition\\model\\dataset\\cropp\\mess
i\\messi1.pngD:\\CelebrityFaceRecognition\\model\\dataset\\cropp\\messi\\mess
i2.png'],
 'virat_kohli': ['D:\\CelebrityFaceRecognition\\model\\dataset\\cropp\\kohli
\\kohli1.pngD:\\CelebrityFaceRecognition\\model\\dataset\\cropp\\kohli\\kohli
2.png']}
```

In [27]:

```
print(path_to_cropp_data)
```

```
D:\CelebrityFaceRecognition\model\dataset\cropp
```

In [28]:

```python
cropped_image_dirs=[]
celebrity_file_names_dict={}
import os
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_cropp_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
              print(cropped_file_path)
              cv2.imwrite(cropped_file_path, roi_color) # save an image on the
              celebrity_file_names_dict[celebrity_name].append(cropped_file_pa
              count +=1
```

```
lionel_messi
Generating cropped images in folder: D:\CelebrityFaceRecognition\model\dat
aset\cropp/lionel_messi
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
1.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
2.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
3.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
4.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
5.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
6.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
7.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
8.png
D:\CelebrityFaceRecognition\model\dataset\cropp/lionel_messi/lionel_messi
```

In [29]:

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

Out[29]:

```
{'lionel_messi': 0,
 'maria_sharapova': 1,
 'roger_federer': 2,
 'serena_williams': 3,
 'virat_kohli': 4}
```

Preprocessing: Use wavelet transform as a feature for traning our model

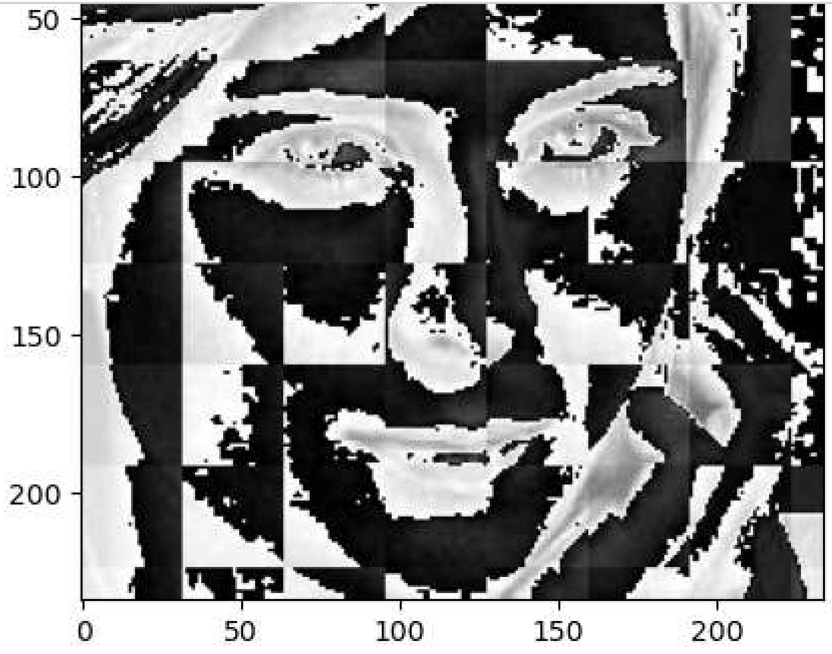
```
In wavelet transformed image, you can see edges clearly and that can give us
clues on various facial features such as eyes, nose, lips etc
```

In [30]:

```
import numpy as np
import pywt
import cv2

def w2d(img,mode='haar',level=1):
    imArray=img
    #datatype conversion
    #convert to grayscale
    imArray=cv2.cvtColor(imArray,cv2.COLOR_RGB2GRAY)
    #convert to float
    imArray=np.float32(imArray)
    imArray /=255;
    # compute coefficents
    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) #An 8-bit unsigned integer whose values exis

    return imArray_H
```

In [31]:

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



In [32]:

```
celebrity_file_names_dict
```

Out[32]:

```
{'lionel_messi': ['D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lio
nel_messi/lionel_messi1.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi2.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi3.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi4.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi5.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi6.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi7.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi8.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi9.png',
  'D:\\CelebrityFaceRecognition\\model\\dataset\\cropp/lionel_messi/lionel
_messi10.png',
```

Images in cropped folder can be used for model training. We will use these raw images along with wavelet transformed images to train our classifier. Let's prepare X and y now

In [33]:

```
X,y=[], []
for celeberity_name, training_file in celebrity_file_names_dict.items():
    for training_image in training_file:
        img=cv2.imread(training_image)
        if img is None:
            continue
        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_im
        X.append(combined_img)
        y.append(class_dict[celeberity_name])
```

In [34]:

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

Out[34]:

```
4096
```

In [35]:

```
len(X)
```

Out[35]:

```
187
```

In [36]:

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

Out[36]:

```
4096
```

In [37]:

```
X = np.array(X).reshape(len(X),4096).astype(float)
X.shape
```

Out[37]:

```
(187, 4096)
```

In [38]:

```
X[0]
```

Out[38]:

```
array([100., 129., 140., ..., 237., 234., 232.])
```

In [39]:

```
y[0]
```

Out[39]:

```
0
```

Data cleaning process is done. Now we are ready to train our model

```
We will use SVM with rbf kernel tuned with heuristic finetuning
```

In [40]:
```
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
```

In [41]:
```
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[41]:
```
0.8723404255319149
```

In [42]:
```
len(X_test)
```

Out[42]:
```
47
```

In [43]:
```
len(y_test)
```

Out[43]:
```
47
```

In [44]:
```
from sklearn.metrics import classification_report
```

In [45]:
```
print(classification_report(y_test, pipe.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.88      0.70      0.78        10
           1       1.00      1.00      1.00         8
           2       0.80      0.67      0.73         6
           3       0.91      0.91      0.91        11
           4       0.80      1.00      0.89        12

    accuracy                           0.87        47
   macro avg       0.88      0.86      0.86        47
weighted avg       0.88      0.87      0.87        47
```

Let's use GridSearch to try out different models with different paramets. Goal is to come up with best modle with best fine tuned parameters

In [46]:

```
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 [47]:

```
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 [48]:

```
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[48]:

	model	best_score	best_params
0	svm	0.692857	{'svc__C': 1, 'svc__kernel': 'linear'}
1	random_forest	0.614286	{'randomforestclassifier__n_estimators': 10}
2	logistic_regression	0.728571	{'logisticregression__C': 1}

```
In [49]: best_estimators
```

```
Out[49]: {'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 [50]: best_estimators['svm'].score(X_test,y_test)
```

```
Out[50]: 0.8723404255319149
```

```
In [51]: best_estimators['random_forest'].score(X_test,y_test)
```

```
Out[51]: 0.7446808510638298
```

```
In [52]: best_estimators['logistic_regression'].score(X_test,y_test)
```

```
Out[52]: 0.851063829787234
```

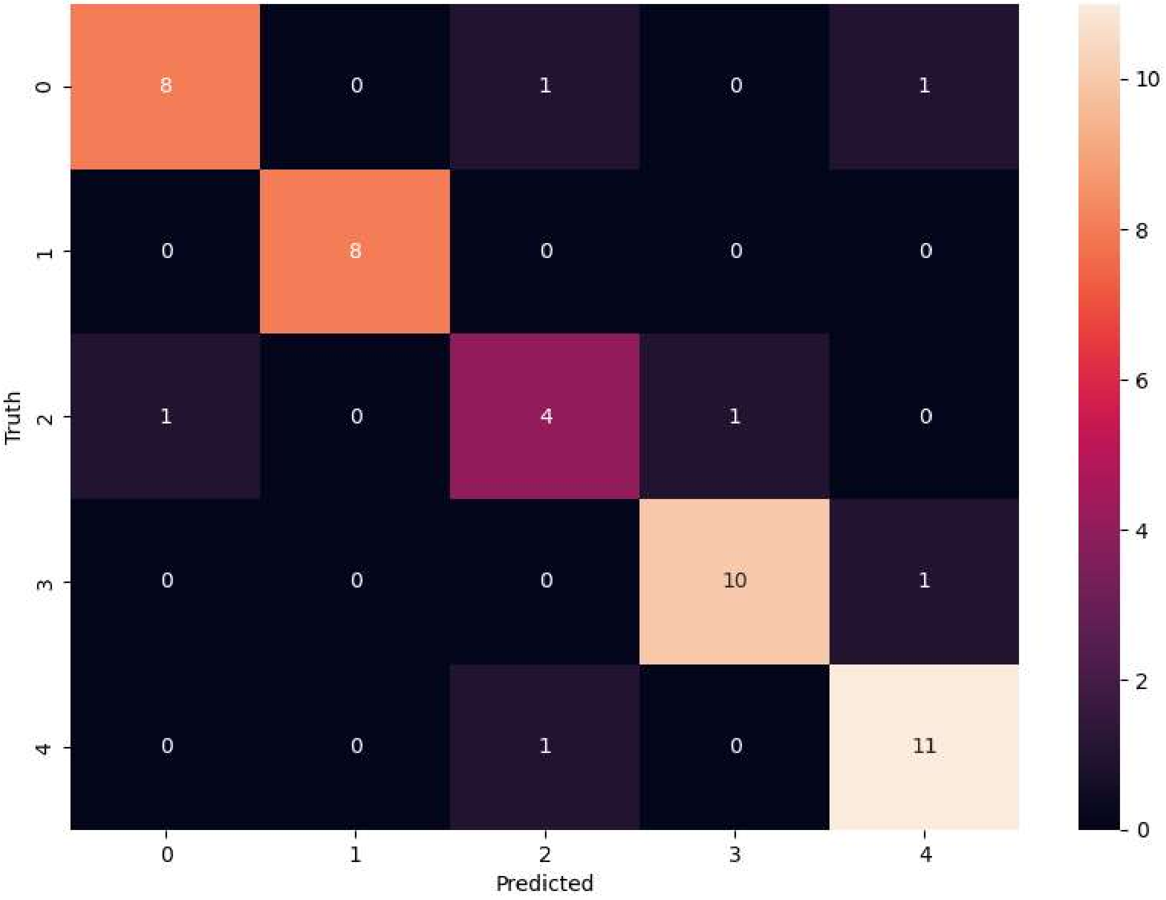
now draw confusion matrix

```
In [53]: best_clf=best_estimators['svm']
```

```
In [54]: from sklearn.metrics import confusion_matrix
         cm=confusion_matrix(y_test, best_clf.predict(X_test))
         cm
```

```
Out[54]: array([[ 8,  0,  1,  0,  1],
                [ 0,  8,  0,  0,  0],
                [ 1,  0,  4,  1,  0],
                [ 0,  0,  0, 10,  1],
                [ 0,  0,  1,  0, 11]], dtype=int64)
```

```
In [55]: import seaborn as sn
         plt.figure(figsize=(10,7))
         sn.heatmap(cm,annot=True)
         plt.xlabel('Predicted')
         plt.ylabel('Truth')

Out[55]: Text(95.72222222222221, 0.5, 'Truth')
```



```
In [56]: class_dict

Out[56]: {'lionel_messi': 0,
          'maria_sharapova': 1,
          'roger_federer': 2,
          'serena_williams': 3,
          'virat_kohli': 4}

In [ ]:
```

Save the trained model

In [57]:

```
!pip install joblib
import joblib
# Save the model as a pickle in a file
joblib.dump(best_clf, 'D:/CelebrityFaceRecognition/server/artifact/saved_model
```

```
Requirement already satisfied: joblib in c:\users\kumar\anaconda3\lib\site-pa
ckages (1.1.1)
```

Out[57]:

```
['D:/CelebrityFaceRecognition/server/artifact/saved_model.pkl']
```

Save class dictionary

In [58]:

```
import json
with open("D:/CelebrityFaceRecognition/server/artifact/class_dictionary.json",
    f.write(json.dumps(class_dict))
```

In [ ]:

```
```