Sports Celebrity Image Classification

In [6]:

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

(1) Preprocessing: Detect face and eyes

When we look at any image, most of the time we identify a person using a face. An image might contain multiple faces, also the face can be obstructed and not clear. The first step in our pre-processing pipeline is to detect faces from an image. Once face is detected, we will detect eyes, if two eyes are detected then only we keep that image otherwise discard it.</h4>

Now how do you detect face and eyes?

We will use haar cascade from opencv for this. Here is an article on this: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html?highlight=haar

In [7]:

```
img = cv2.imread('./test_images/sharapova1.jpg')
img.shape
```

Out[7]:

```
(555, 700, 3)
```

In [8]:

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

Out[8]:

```
<matplotlib.image.AxesImage at 0x819ef48>
```

In [9]:

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

Out[9]:

Out[9]:

```
(555, 700)
```

In [10]:

```
gray
```

Out[10]:

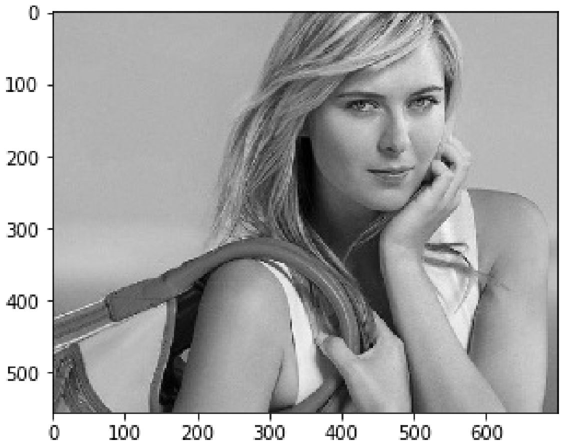
```
array([[175, 175, 175, ..., 176, 175, 174],
       [175, 175, 175, ..., 177, 175, 174],
       [175, 175, 175, ..., 177, 176, 174],
       ...,
       [ 84,  87,  88, ..., 113, 113, 113],
       [ 88,  89,  90, ..., 113, 113, 113],
       [ 93,  91,  91, ..., 112, 112, 112]], dtype=uint8)
```

In [11]:

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

Out[11]:

```
<matplotlib.image.AxesImage at 0x825bf08>
```



In [12]:

```
face_cascade = cv2.CascadeClassifier('./opencv/haarcascades/haarcascade_frontalface_defau
lt.xml')
eye_cascade = cv2.CascadeClassifier('./opencv/haarcascades/haarcascade_eye.xml')

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

Out[12]:

```
array([[352,  38, 233, 233]], dtype=int32)
```

In [13]:

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

Out[13]:

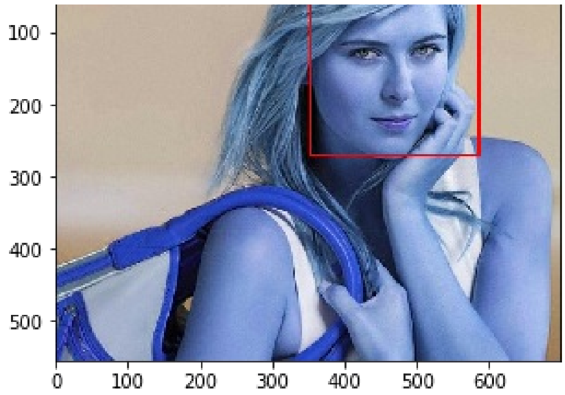
```
(352, 38, 233, 233)
```

In [14]:

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

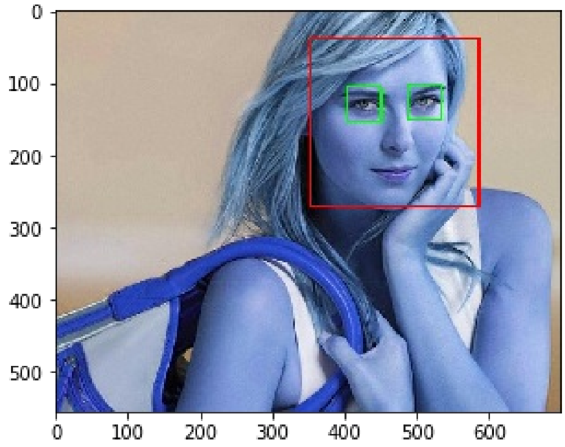
Out[14]:

```
<matplotlib.image.AxesImage at 0x8ba2448>
```



In [15]:

```
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()
```



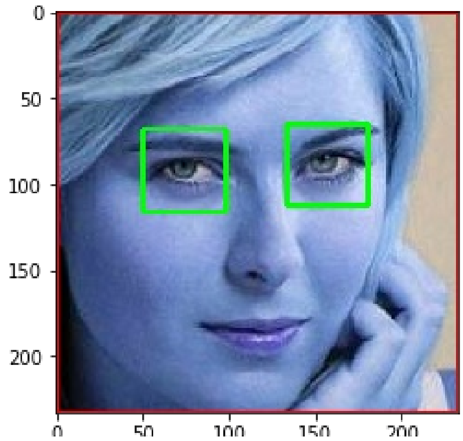
(2) Preprocessing: Crop the facial region of the image

In [16]:

```
%matplotlib inline
plt.imshow(roi_color, cmap='gray')
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x9080f08>
```



In [17]:

```
cropped_img = np.array(roi_color)
cropped_img.shape
```

Out[17]:

```
(233, 233, 3)
```

(3) 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

Wavelet transform

In [18]:

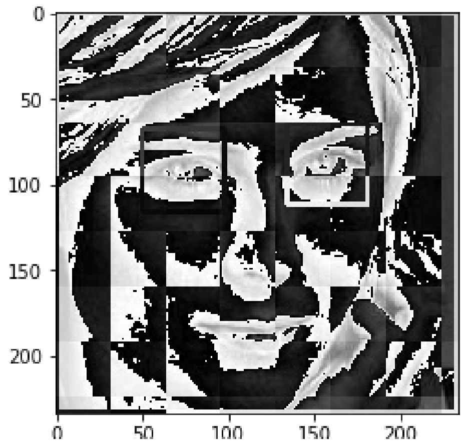
```
import numpy as np
import pywt
import cv2

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 [19]:

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

Out[19]:

```
<matplotlib.image.AxesImage at 0xc822348>
```



You can see above a wavelet transformed image that gives clues on facial features such as eyes, nose, lips etc. This along with raw pixel image can be used as an input for our classifier

(3) Preprocessing: Load image, detect face. If eyes >=2, then save and crop the face region

Lets write a python function that can take input image and returns cropped image (if face and eyes >=2 are detected)

In [20]:

```
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 [21]:

```
original_image = cv2.imread('./test_images/sharapova1.jpg')
plt.imshow(original_image)
```

Out[21]:

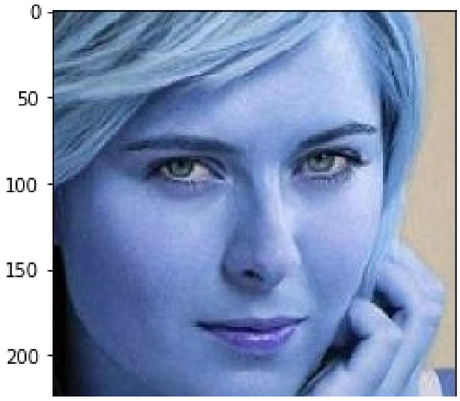
```
<matplotlib.image.AxesImage at 0xc881548>
```

In [22]:

```
cropped_image = get_cropped_image_if_2_eyes('./test_images/sharapova1.jpg')
plt.imshow(cropped_image)
```

Out[22]:

```
<matplotlib.image.AxesImage at 0xc8e88c8>
```



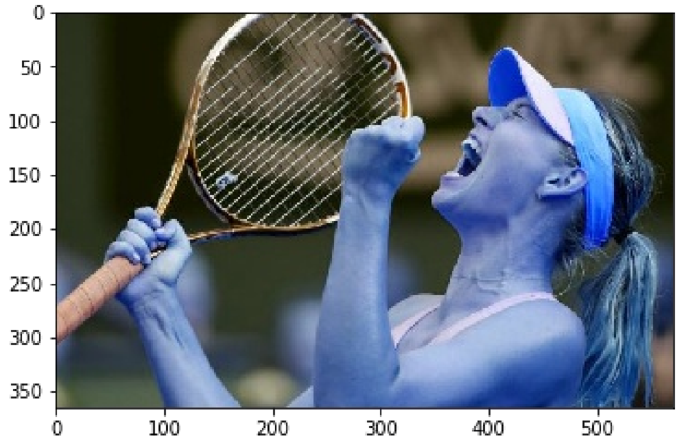
In below image face is not very clear and it doesn't have two eyes clearly visible

In [23]:

```
org_image_obstructed = cv2.imread('./test_images/sharapova2.jpg')
plt.imshow(org_image_obstructed)
```

Out[23]:

```
<matplotlib.image.AxesImage at 0xc9495c8>
```



In [24]:

```
cropped_image_no_2_eyes = get_cropped_image_if_2_eyes('./test_images/sharapova2.jpg')
cropped_image_no_2_eyes
```

Above cropped_image_no_2_eyes is None which means we should ignore this image and we will not use such image for model training

In [25]:

```
path_to_data = "./images_dataset/"
path_to_cr_data = "./images_dataset/cropped/"
```

In [26]:

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

In [27]:

```
img_dirs
```

Out[27]:

```
['./images_dataset/lionel_messi',
 './images_dataset/maria_sharapova',
 './images_dataset/roger_federer',
 './images_dataset/serena_williams',
 './images_dataset/virat_kohli']
```

Go through all images in dataset folder and create cropped images for them. There will be cropped folder inside dataset folder after you run this code

In [28]:

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

In [29]:

```
cropped_image_dirs = []
celebrity_file_names_dict = {}
for img_dir in img_dirs:
    count = 1
    celebrity_name = img_dir.split('/')[-1]
    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
```

```
Generating cropped images in folder:  ./images_dataset/cropped/lionel_messi
Generating cropped images in folder:  ./images_dataset/cropped/maria_sharapova
Generating cropped images in folder:  ./images_dataset/cropped/roger_federer
Generating cropped images in folder:  ./images_dataset/cropped/serena_williams
Generating cropped images in folder:  ./images_dataset/cropped/virat_kohli
```

Now you should have cropped folder under datasets folder that contains cropped images

Manually examine cropped folder and delete any unwanted images

In [30]:

```
celebrity_file_names_dict = {}
for img_dir in cropped_image_dirs:
    celebrity_name = img_dir.split('/')[-1]
    file_list = []
    for entry in os.scandir(img_dir):
        file_list.append(entry.path)
    celebrity_file_names_dict[celebrity_name] = file_list
celebrity_file_names_dict
```

Out[30]:

```
{'lionel_messi': ['./images_dataset/cropped/lionel_messi\\lionel_messi1.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi10.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi11.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi13.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi14.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi15.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi16.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi17.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi18.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi19.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi2.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi20.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi22.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi23.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi24.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi25.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi26.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi27.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi28.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi29.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi3.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi3.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi30.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi32.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi33.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi34.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi35.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi36.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi37.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi38.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi39.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi4.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi5.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi6.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi7.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi8.png',
  './images_dataset/cropped/lionel_messi\\lionel_messi9.png'],
 'maria_sharapova': ['./images_dataset/cropped/maria_sharapova\\maria_sharapova10.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova11.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova13.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova14.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova15.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova16.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova17.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova18.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova19.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova20.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova21.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova22.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova23.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova24.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova25.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova26.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova27.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova28.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova29.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova30.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova31.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova32.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova34.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova35.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova4.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova5.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova6.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova7.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova8.png',
  './images_dataset/cropped/maria_sharapova\\maria_sharapova9.png'],
 'roger_federer': ['./images_dataset/cropped/roger_federer\\roger_federer1.png',
  './images_dataset/cropped/roger_federer\\roger_federer10.png',
  './images_dataset/cropped/roger_federer\\roger_federer11.png',
  './images_dataset/cropped/roger_federer\\roger_federer12.png',
  './images_dataset/cropped/roger_federer\\roger_federer13.png',
  './images_dataset/cropped/roger_federer\\roger_federer14.png',
  './images_dataset/cropped/roger_federer\\roger_federer15.png',
  './images_dataset/cropped/roger_federer\\roger_federer16.png',
  './images_dataset/cropped/roger_federer\\roger_federer17.png',
  './images_dataset/cropped/roger_federer\\roger_federer18.png',
  './images_dataset/cropped/roger_federer\\roger_federer19.png',
  './images_dataset/cropped/roger_federer\\roger_federer2.png',
  './images_dataset/cropped/roger_federer\\roger_federer20.png',
  './images_dataset/cropped/roger_federer\\roger_federer21.png',
  './images_dataset/cropped/roger_federer\\roger_federer22.png',
  './images_dataset/cropped/roger_federer\\roger_federer23.png',
  './images_dataset/cropped/roger_federer\\roger_federer24.png',
  './images_dataset/cropped/roger_federer\\roger_federer25.png',
  './images_dataset/cropped/roger_federer\\roger_federer28.png',
  './images_dataset/cropped/roger_federer\\roger_federer29.png',
  './images_dataset/cropped/roger_federer\\roger_federer3.png',
  './images_dataset/cropped/roger_federer\\roger_federer30.png',
  './images_dataset/cropped/roger_federer\\roger_federer4.png',
  './images_dataset/cropped/roger_federer\\roger_federer5.png',
  './images_dataset/cropped/roger_federer\\roger_federer6.png',
  './images_dataset/cropped/roger_federer\\roger_federer7.png',
  './images_dataset/cropped/roger_federer\\roger_federer8.png',
  './images_dataset/cropped/roger_federer\\roger_federer8.png',
  './images_dataset/cropped/roger_federer\\roger_federer9.png'],
 'serena_williams': ['./images_dataset/cropped/serena_williams\\serena_williams10.png',
  './images_dataset/cropped/serena_williams\\serena_williams12.png',
  './images_dataset/cropped/serena_williams\\serena_williams13.png',
  './images_dataset/cropped/serena_williams\\serena_williams14.png',
  './images_dataset/cropped/serena_williams\\serena_williams15.png',
  './images_dataset/cropped/serena_williams\\serena_williams16.png',
  './images_dataset/cropped/serena_williams\\serena_williams17.png',
  './images_dataset/cropped/serena_williams\\serena_williams2.png',
  './images_dataset/cropped/serena_williams\\serena_williams20.png',
  './images_dataset/cropped/serena_williams\\serena_williams22.png',
  './images_dataset/cropped/serena_williams\\serena_williams23.png',
  './images_dataset/cropped/serena_williams\\serena_williams24.png',
  './images_dataset/cropped/serena_williams\\serena_williams26.png',
  './images_dataset/cropped/serena_williams\\serena_williams28.png',
  './images_dataset/cropped/serena_williams\\serena_williams29.png',
  './images_dataset/cropped/serena_williams\\serena_williams3.png',
  './images_dataset/cropped/serena_williams\\serena_williams30.png',
  './images_dataset/cropped/serena_williams\\serena_williams31.png',
  './images_dataset/cropped/serena_williams\\serena_williams32.png',
  './images_dataset/cropped/serena_williams\\serena_williams35.png',
  './images_dataset/cropped/serena_williams\\serena_williams4.png',
  './images_dataset/cropped/serena_williams\\serena_williams5.png',
  './images_dataset/cropped/serena_williams\\serena_williams6.png',
  './images_dataset/cropped/serena_williams\\serena_williams7.png',
  './images_dataset/cropped/serena_williams\\serena_williams8.png',
  './images_dataset/cropped/serena_williams\\serena_williams9.png'],
 'virat_kohli': ['./images_dataset/cropped/virat_kohli\\virat_kohli1.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli10.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli11.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli12.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli13.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli14.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli15.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli16.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli17.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli18.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli19.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli2.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli20.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli21.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli23.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli25.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli26.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli27.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli28.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli30.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli31.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli32.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli33.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli35.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli36.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli37.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli38.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli4.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli40.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli41.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli42.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli43.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli44.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli45.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli46.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli47.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli48.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli5.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli6.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli7.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli8.png',
  './images_dataset/cropped/virat_kohli\\virat_kohli9.png']}
```

In [31]:

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

Out[31]:

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

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 [32]:

```
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_img_har.res
hape(32*32,1)))
        X.append(combined_img)
        y.append(class_dict[celebrity_name])
```

In [33]:

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

Out[33]:

```
4096
```

In [34]:

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

Out[34]:

```
4096
```

In [35]:

```
X[0]
```

Out[35]:

```
array([[100],
       [129],
       [140],
       ...,
       [237],
       [234],
       [232]], dtype=uint8)
```

In [36]:

```
y[0]
```

Out[36]:

```
0
```

In [37]:

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

Out[37]:

```
(162, 4096)
```

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 [38]:

```
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 [39]:

```
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 = 10))])
pipe.fit(X_train, y_train)
pipe.score(X_test, y_test)
```

Out[39]:

```
0.8780487804878049
```

In [40]:

```
print(classification_report(y_test, pipe.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.75      0.86      0.80         7
           1       0.91      0.91      0.91        11
           2       0.75      0.75      0.75         4
           3       1.00      0.83      0.91         6
           4       0.92      0.92      0.92        13

    accuracy                           0.88        41
   macro avg       0.87      0.85      0.86        41
weighted avg       0.88      0.88      0.88        41
```

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 [41]:

```
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 [42]:

```
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 [43]:

```
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[43]:

	model	best_score	best_params
0	svm	0.826000	{'svc__C': 1, 'svc__kernel': 'linear'}
1	random_forest	0.628333	{'randomforestclassifier__n_estimators': 10}
2	logistic_regression	0.834000	{'logisticregression__C': 1}

In [44]:

```
best_estimators
```

Out[44]:

```
{'svm': Pipeline(memory=None,
          steps=[('standardscaler',
                   StandardScaler(copy=True, with_mean=True, with_std=True)),
                  ('svc',
                   SVC(C=1, break_ties=False, cache_size=200, class_weight=None,
                       coef0=0.0, decision_function_shape='ovr', degree=3,
                       gamma='auto', kernel='linear', max_iter=-1,
                       probability=True, random_state=None, shrinking=True,
                       tol=0.001, verbose=False))],
          verbose=False),
 'random_forest': Pipeline(memory=None,
          steps=[('standardscaler',
                   StandardScaler(copy=True, with_mean=True, with_std=True)),
                  ('randomforestclassifier',
                   RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                          class_weight=None, criterion='gini',
                                          max_depth=None, max_features='auto',
                                          max_leaf_nodes=None, max_samples=None,
                                          min_impurity_decrease=0.0,
                                          min_impurity_split=None,
                                          min_samples_leaf=1, min_samples_split=2,
                                          min_weight_fraction_leaf=0.0,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=10, n_jobs=None,
                                              oob_score=False, random_state=None,
                                              verbose=0, warm_start=False))],
          verbose=False),
 'logistic_regression': Pipeline(memory=None,
          steps=[('standardscaler',
                  StandardScaler(copy=True, with_mean=True, with_std=True)),
                 ('logisticregression',
                  LogisticRegression(C=1, class_weight=None, dual=False,
                                     fit_intercept=True, intercept_scaling=1,
                                     l1_ratio=None, max_iter=100,
                                     multi_class='auto', n_jobs=None,
                                     penalty='l2', random_state=None,
                                     solver='liblinear', tol=0.0001, verbose=0,
                                     warm_start=False))],
          verbose=False)}
```

In [45]:

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

Out[45]:

```
0.9512195121951219
```

In [46]:

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

Out[46]:

```
0.6829268292682927
```

In [47]:

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

Out[47]:

```
0.9512195121951219
```

In [48]:

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

In [49]:

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

Out[49]:

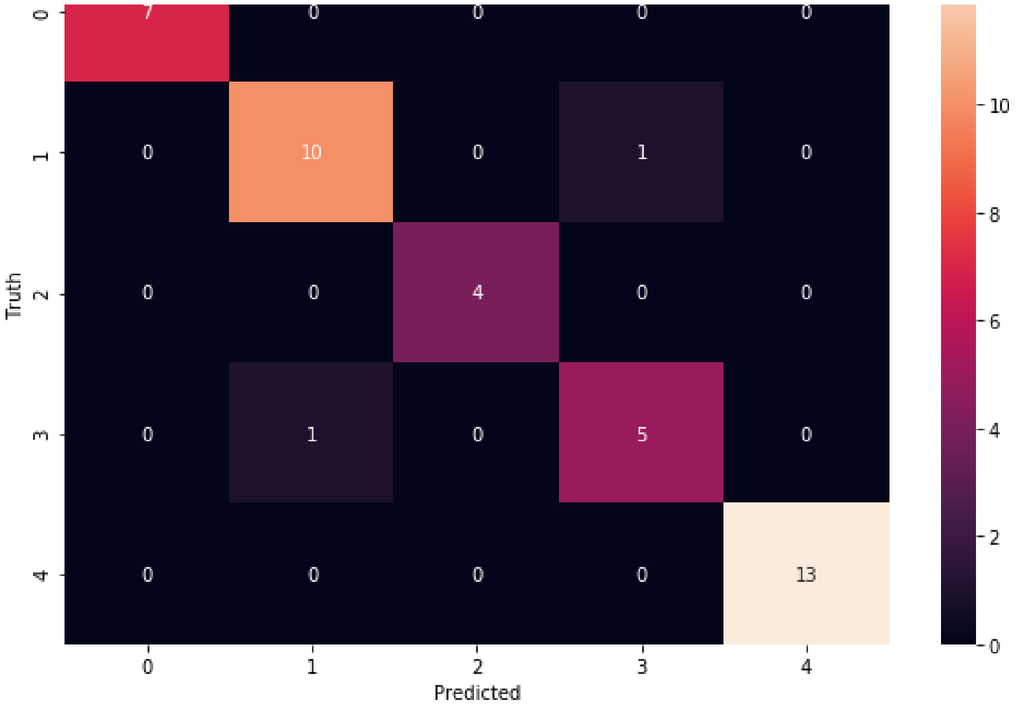
```
array([[ 7,  0,  0,  0,  0],
       [ 0, 10,  0,  1,  0],
       [ 0,  0,  4,  0,  0],
       [ 0,  1,  0,  5,  0],
       [ 0,  0,  0,  0, 13]], dtype=int64)
```

In [50]:

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

Out[50]:

```
Text(69.0, 0.5, 'Truth')
```



In [51]:

```
class_dict
```

Out[51]:

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

Save the trained model

In [52]:

```
!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 c:\users\s\anaconda3\lib\site-packages (0.14.1)
```

Out[52]:

```
['saved_model.pkl']
```

Save class dictionary

In [53]:

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

In [ ]: