Project Presentation (DS-640)

Predictive Analytics & Forecasting Modeling

Topic : Face Detection

Face Detection Model For Celebrities

Developed & Presented By:

Virendrasinh Rajput

❖What is Face Detection:

- Face Detection is AI Based Computer Technology Used to Find & Identify Human Face in Digital Image. It Uses Various Types of Data algorithms to Identify Particular individual.

❖Face Detection In Our Project:

- We Used & Applied Face Detection Model to Identify Celebrities. We Selected Sports As Our Domain and We Decided to Go with 5 Famous Celebrities in our Project.

1. Maria Sharapova
2. Virat Kohli
3. Lionel Messi
4. Serena Williams
5. Roger Federer

❖Application of Face Detection:

- Major Use of Face Detection That we are mostly Focusing on is in Security Domain. It can Identify & Track Moments Of Famous Celebrities in crowded Conditions. Also, in Mobile like Face ID & In Office for punch in & Punch Out. & To identify Criminals Etc.

❖ Introduction:

- In This Project We Used & Applied Face Detection Model to Identify Celebrities. We Selected Sports As Our Domain and We Decided to Go with 5 Famous Celebrities in our Project.

1. Maria Sharapova
2. Virat Kohli
3. Lionel Messi
4. Serena Williams
5. Roger Federer

- We Used Fatkun Extension For Bulk Download of Images Needed For Model Training & Testing.
- We Used Haar Casced Algorithm for OpenCV Library as Face and eye Detection Classifier.
- Our Project Will Detect Face of 5 Celebrities & also Follows Specific conditions like Both Eyes Visibility or Face Visibility etc.
- Our Deployment Aim is to Create a Website or UI That can detect Face by Dragging and dropping an image of the celebrities.

❖Fatkun Extension & Haar Cascade:

➢ Fatkun Extension:

- Fatkun extension is Chrome Extension, and it is used to do image scrapping for project. It is used to download bulk images for our need.
- Fatkun is most basic Chrome Extension which can download bulk image from webpage by just one click and we can also adjust height and width of the images.
- This images will be downloaded in one folder, and we can name it as Celebrity 1 or name of the celebrity.

➢ Haar Cascade:

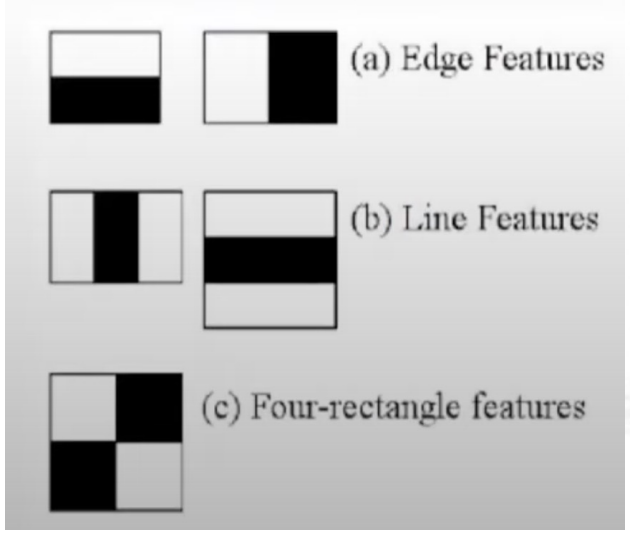
- Haar Cascade is a machine literacy- based approach where a lot of positive and negative images are used to train the classifier.
- Positive images – These images contain the images which we want our classifier to identify. Negative Images – Images of everything differently, which don't contain the object we want to detect.
- That's why we will select only image with face and two eyes for our model.
- This image classifier will save every face with two eyes in our cropped folder so we will have to manually clean the data to only have suitable images.

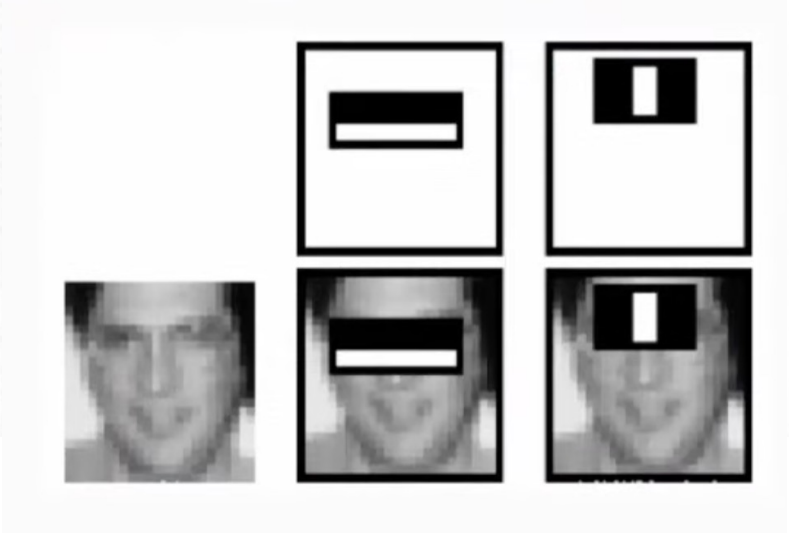
❖ Feature Engineering with SVM:

- Feature Engineering is the technique to Transforming Raw data into features. It uses domain knowledge to Extract Feature from Raw data.
- Support Vector Machines Are Supervised Learning models Which are associated with learning algorithm that can analyze data for classification and regression analysis.
- We are using combine feature engineering with SVM for our face detection model for better performance & more reliable machine learning model.
- We compare our model with SVM, Random Forest & Logistic Regression. Based of results we can see that our SVM Accuracy is more compared to Random Forest or logistic regression. That's why choose SVM for our Project.

❖Open CV & Haar Casced Working:

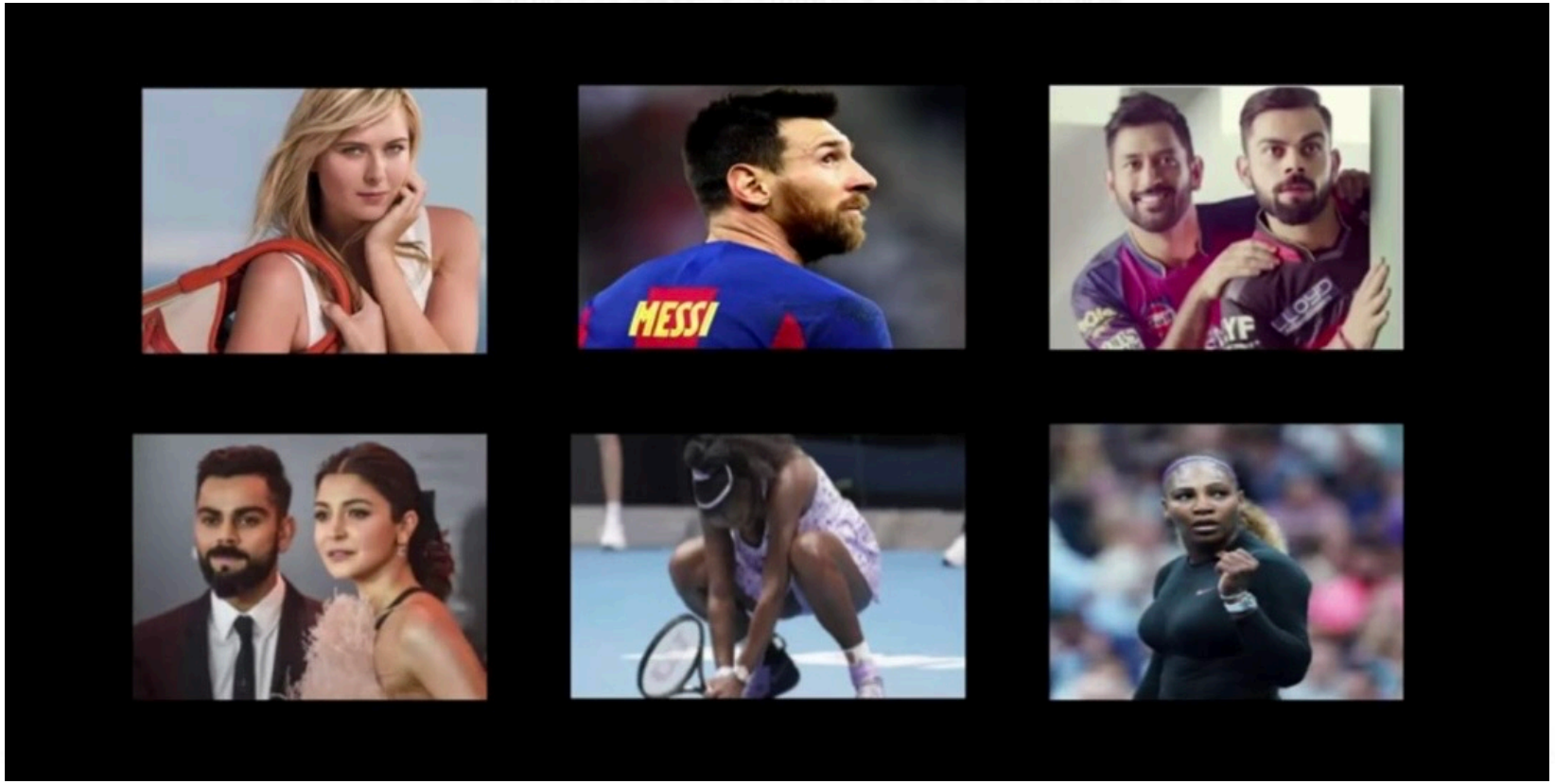
- We are not going to go too much into detail on what is hard cascade how it works because it's it requires a long discussion. But just to give a brief idea, you have this line and edge features which will use a moving window of this features to detect where your nose is and where is your eyes.
- For example, in this image when you have eyes, the area of Eye tend to be darker than the area below. Similarly, when you have nose the area of eyes since tend to be darker and the tip of the nose will be a little brighter.
- So, we can use all this mask to detect these areas. For Simplicity open CV documentation contains this ready-made API, which We can use to detect the face and image.



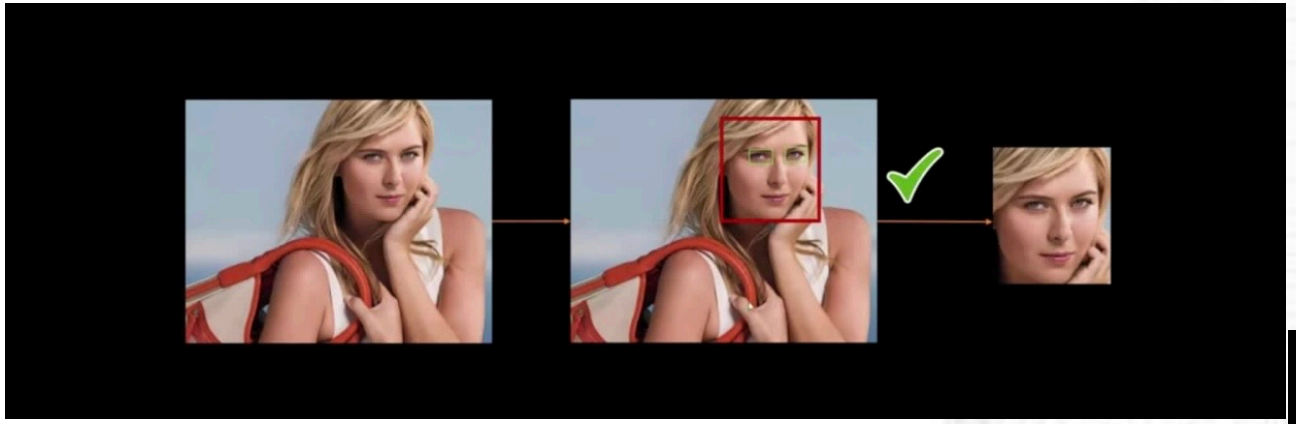


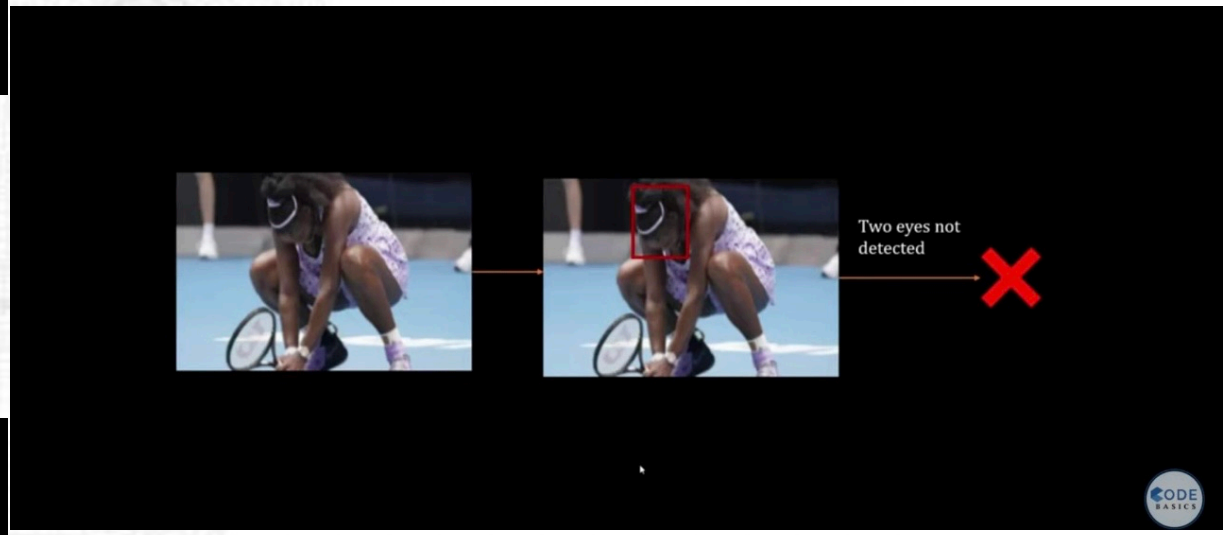
❖Raw Images From Bulk Downloads:

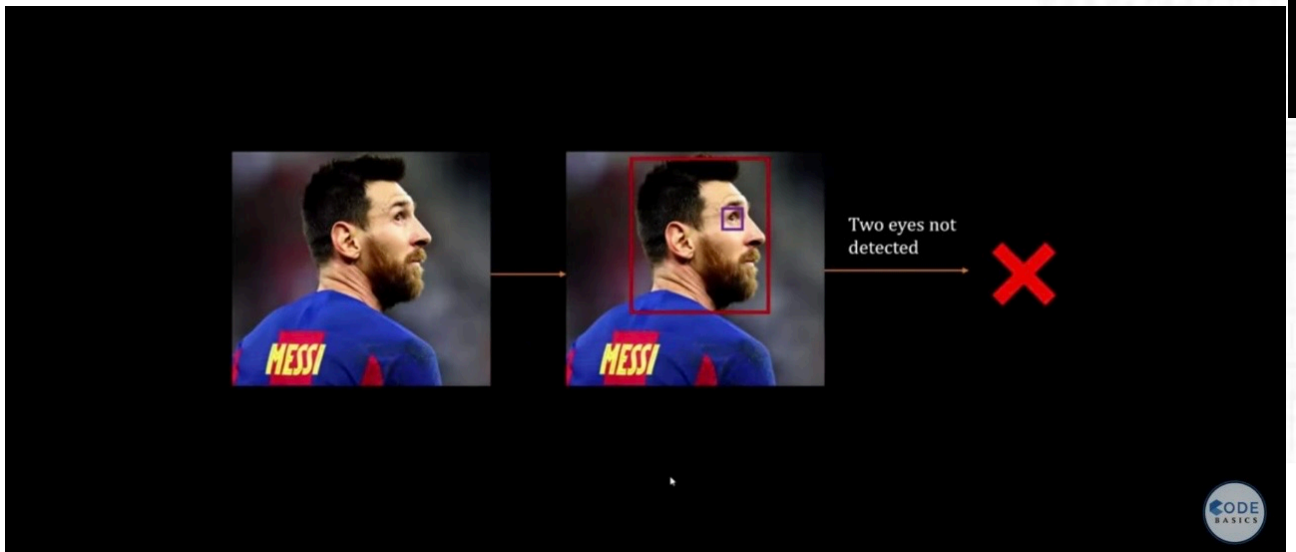
- Our Model Needs to Have Specific types of Image for its Training and testing Purpose. Now while doing Bulk Download of the Images using Fatkun Extension It will download Raw images & some of which is not suitable for model. so, we will have to manually Clean That data.
- Raw images can have data like face of different people, Both eyes not visible or face is not properly visible.



❖Suitable & Non-Suitable Images Example:

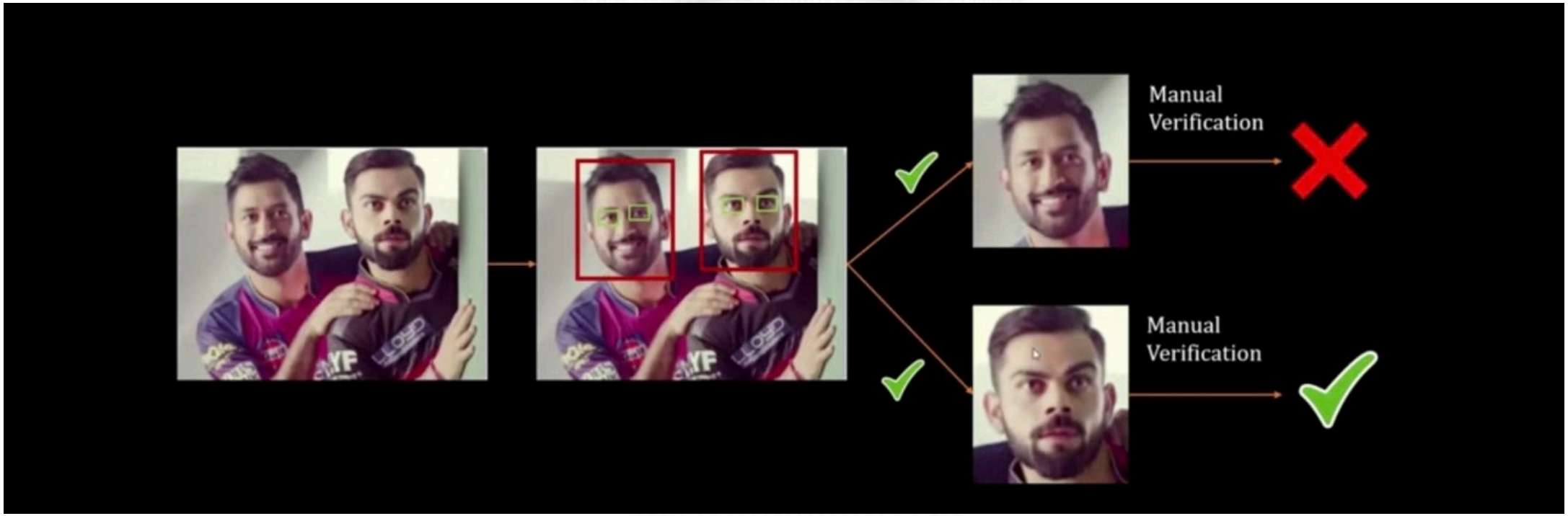




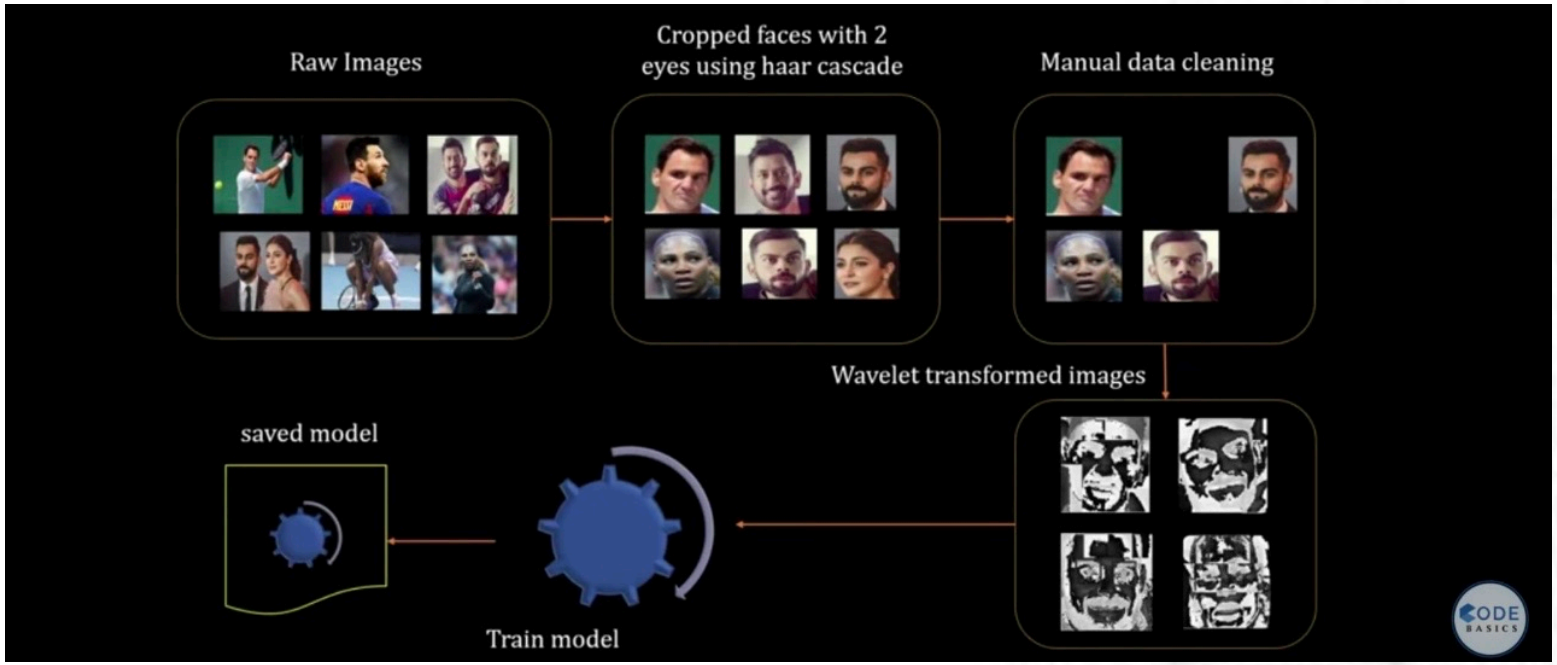


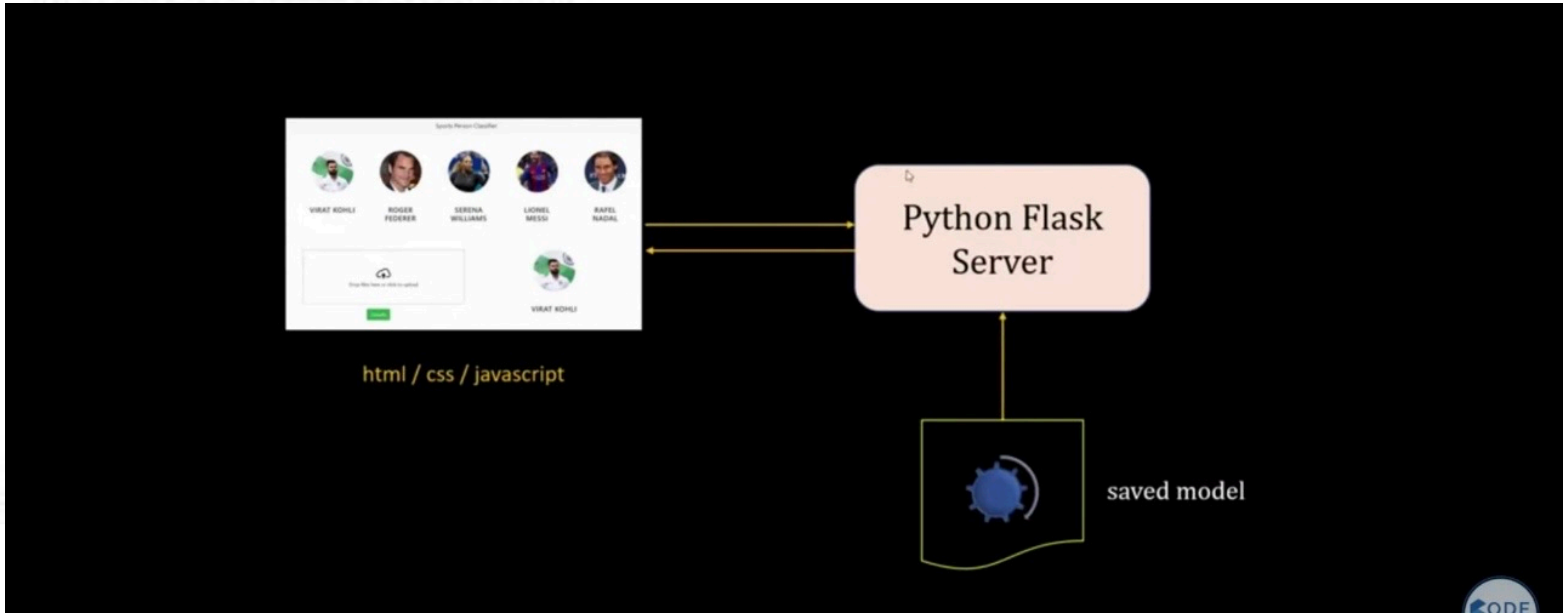
❖Manual Verification:

- Haar Casced will Cropped any face with two eyes without any Filter and store it in the image folder, but we manually did the Cleaning of data & only keep useful images. In this case our Algorithm has Clicked and stored two faced from a single picture. One of the Face is of M.S Dhoni & Other Face is Of Virat Kohli. As we only Need Virat Kohli's Face we have to Manually Delete MS Dhoni's Face.



❖Project Working Flowchart:





❖Our Code: Image Read

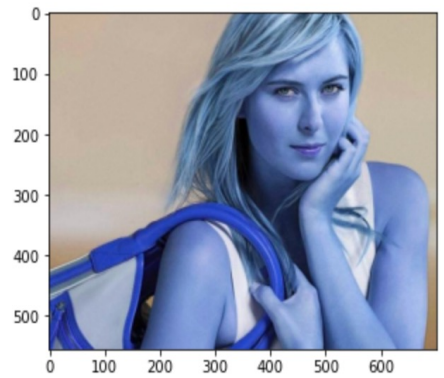
In [1]:
```
import numpy as np
import cv2
import matplotlib
from matplotlib import pyplot as plt
%matplotlib inline
```

In [2]:
```
img = cv2.imread('./test_images/sharapova1.jpg')
img.shape
```

Out[2]:
```
(555, 700, 3)
```

In [3]:
```
plt.imshow(img)
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x7fb5d0607430>
```



In [4]:
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray.shape
```

Out[4]:
```
(555, 700)
```

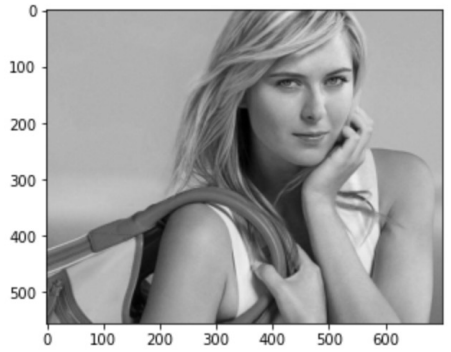
In [5]:
```
gray
```

Out[5]:
```
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)
```

❖Our Code: Image Read (Outline Face)

In [6]:
```
plt.imshow(gray, cmap='gray')
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x7fb5d143cd60>
```



In [7]:
```
face_cascade = cv2.CascadeClassifier('./opencv/haarcascades/haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('./opencv/haarcascades/haarcascade_eye.xml')

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

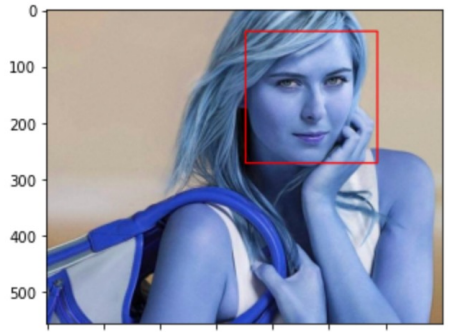
Out[7]:
```
array([[352,  38, 233, 233]], dtype=int32)
```

In [8]:
```
(x,y,w,h) = faces[0]
x,y,w,h
```

Out[8]:
```
(352, 38, 233, 233)
```

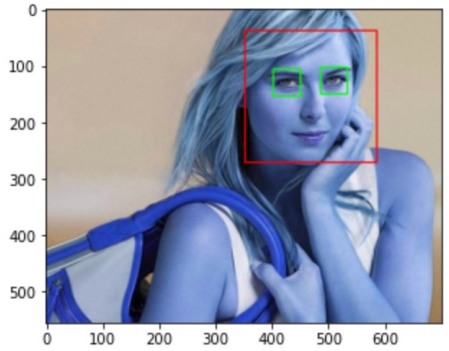
In [9]:
```
face_img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
plt.imshow(face_img)
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x7fb5d14d55b0>
```



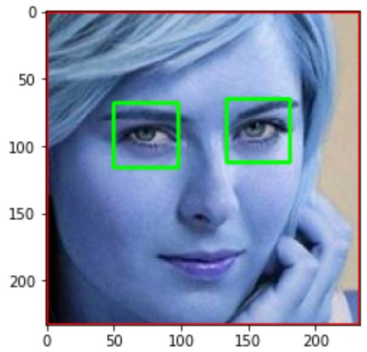
❖Our Code: Image Read (Outline Eyes)

In [10]:
```
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 [11]:
```
%matplotlib inline
plt.imshow(roi_color, cmap='gray')
```

Out[11]:
```
<matplotlib.image.AxesImage at 0x7fb5d28b97f0>
```



❖Our Code: Wavelet Transform

In [12]:
```
cropped_img = np.array(roi_color)
cropped_img.shape
```

Out[12]:
```
(233, 233, 3)
```

Wavelet transform

In [13]:
```
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 [14]:
```
im_har = w2d(cropped_img,'db1',5)
plt.imshow(im_har, cmap='gray')
```

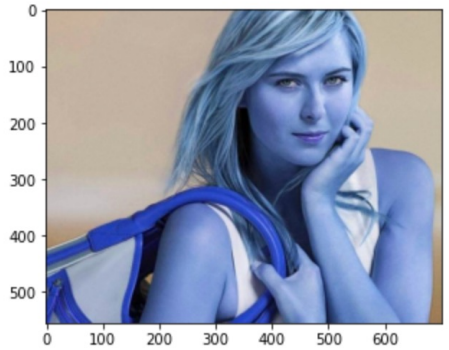
Out[14]:
```
<matplotlib.image.AxesImage at 0x7fb5d2d99550>
```

❖ Our Code: Preprocessing: Load image, detect face. If eyes >=2, then save and crop the face region

In [15]:
```
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
```

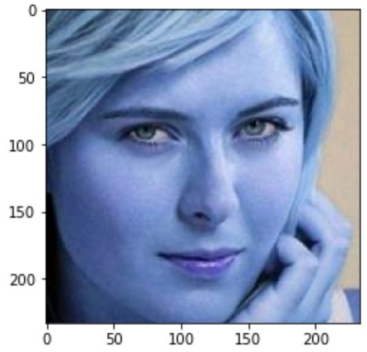
In [16]:
```
original_image = cv2.imread('./test_images/sharapova1.jpg')
plt.imshow(original_image)
```

Out[16]: <matplotlib.image.AxesImage at 0x7fb5d2f4aac0>



In [17]:
```
cropped_image = get_cropped_image_if_2_eyes('./test_images/sharapova1.jpg')
plt.imshow(cropped_image)
```

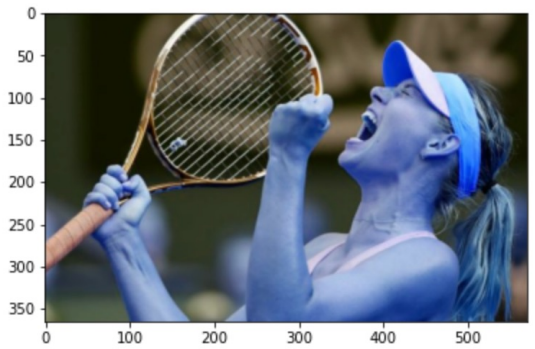
Out[17]: <matplotlib.image.AxesImage at 0x7fb5d34442e0>



❖ Our Code: Preprocessing (2 eyes Detections)

In [18]:
```
org_image_obstructed = cv2.imread('./test_images/sharapova2.jpg')
plt.imshow(org_image_obstructed)
```

Out[18]: <matplotlib.image.AxesImage at 0x7fb5d35649a0>



In [19]:
```
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 [20]:
```
path_to_data = "./dataset/"
path_to_cr_data = "./dataset/cropped/"
```

In [21]:
```
import os
img_dirs = []
for entry in os.scandir(path_to_data):
    if entry.is_dir():
        img_dirs.append(entry.path)
```

In [22]:
```
img_dirs
```

Out[22]:
```
['./dataset/maria_sharapova',
 './dataset/virat_kohli',
 './dataset/lionel_messi',
 './dataset/serena_williams',
 './dataset/roger_federer']
```

❖ Our Code: Preprocessing (Saving Results)

In [23]:
```
import shutil
if os.path.exists(path_to_cr_data):
     shutil.rmtree(path_to_cr_data)
os.mkdir(path_to_cr_data)
```

In [24]:
```
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:  ./dataset/cropped/maria_sharapova
Generating cropped images in folder:  ./dataset/cropped/virat_kohli
Generating cropped images in folder:  ./dataset/cropped/lionel_messi
Generating cropped images in folder:  ./dataset/cropped/serena_williams
Generating cropped images in folder:  ./dataset/cropped/roger_federer
```

❖ Our Code: Preprocessing (Create Class Dictionary)

In [25]:
```
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[25]:
```
{'maria_sharapova': ['./dataset/cropped/maria_sharapova/maria_sharapova2.png',
  './dataset/cropped/maria_sharapova/maria_sharapova3.png',
  './dataset/cropped/maria_sharapova/maria_sharapova1.png',
  './dataset/cropped/maria_sharapova/.DS_Store',
  './dataset/cropped/maria_sharapova/maria_sharapova5.png',
  './dataset/cropped/maria_sharapova/maria_sharapova7.png',
  './dataset/cropped/maria_sharapova/maria_sharapova6.png',
  './dataset/cropped/maria_sharapova/maria_sharapova17.png',
  './dataset/cropped/maria_sharapova/maria_sharapova16.png',
  './dataset/cropped/maria_sharapova/maria_sharapova28.png',
  './dataset/cropped/maria_sharapova/maria_sharapova14.png',
  './dataset/cropped/maria_sharapova/maria_sharapova15.png',
  './dataset/cropped/maria_sharapova/maria_sharapova29.png',
  './dataset/cropped/maria_sharapova/maria_sharapova11.png',
  './dataset/cropped/maria_sharapova/maria_sharapova10.png',
  './dataset/cropped/maria_sharapova/maria_sharapova12.png',
  './dataset/cropped/maria_sharapova/maria_sharapova22.png',
  './dataset/cropped/maria_sharapova/maria_sharapova35.png',
  './dataset/cropped/maria_sharapova/maria_sharapova34.png',
```

In [26]:
```
class_dict = {}
count = 0
for celebrity_name in celebrity_file_names_dict.keys():
    class_dict[celebrity_name] = count
    count = count + 1
class_dict
```

Out[26]:
```
{'maria_sharapova': 0,
 'virat_kohli': 1,
 'lionel_messi': 2,
 'serena_williams': 3,
 'roger_federer': 4}
```

❖ Our Code: Training & Testing

In [33]:
```
X, y = [], []
for celebrity_name, training_files in celebrity_file_names_dict.items():
    for training_image in training_files:
        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_img_har.reshape(32*32,1)))
        X.append(combined_img)
        y.append(class_dict[celebrity_name])
```

```
[ WARN:0@1369.434] global /Users/runner/work/opencv-python/opencv-python/opencv/modules/imgcodecs/src/loadsave.cpp
(239) findDecoder imread_('./dataset/cropped/virat_kohli/virat_kohli8.png'): can't open/read file: check file path/
integrity
```

In [34]:
```
len(X)
```

Out[34]:
```
162
```

In [35]:
```
len(X[0])
```

Out[35]:
```
4096
```

In [36]:
```
32*32*3 + 32*32
```

Out[36]:
```
4096
```

In [37]:
```
X[0]
```

Out[37]:
```
array([[43],
       [43],
       [43],
       ...,
       [ 0],
       [ 0],
       [18]], dtype=uint8)
```

In [38]:
```
X = np.array(X).reshape(len(X),4096).astype(float)
X.shape
```

Out[38]:
```
(162, 4096)
```

In [39]:
```
X[0]
```

Out[39]:
```
array([43., 43., 43., ...,  0.,  0., 18.])
```

❖ Our Code: SVM Model

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
from sklearn.metrics import classification_report
```

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

Out[41]: 0.8780487804878049

In [42]:
```
print(classification_report(y_test, pipe.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.71      0.83      0.77         6
           1       0.86      0.92      0.89        13
           2       0.83      1.00      0.91         5
           3       1.00      0.67      0.80         9
           4       1.00      1.00      1.00         8

    accuracy                           0.88        41
   macro avg       0.88      0.88      0.87        41
weighted avg       0.89      0.88      0.88        41
```

In [43]:
```
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 [44]:
```
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]
        }
    }
}
```

❖ Our Code: SVM Model (Grid Searching)

In [45]:
```
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[45]:

	model	best_score	best_params
0	svm	0.801333	{'svc__C': 1, 'svc__kernel': 'linear'}
1	random_forest	0.685667	{'randomforestclassifier__n_estimators': 10}
2	logistic_regression	0.810333	{'logisticregression__C': 1}

In [46]:
```
best_estimators
```

Out[46]:
```
{'svm': Pipeline(steps=[('standardscaler', StandardScaler()),
                        ('svc',
                         SVC(C=1, gamma='auto', kernel='linear', probability=True))]),
 'random_forest': Pipeline(steps=[('standardscaler', StandardScaler()),
                                  ('randomforestclassifier',
                                   RandomForestClassifier(n_estimators=10))]),
 'logistic_regression': Pipeline(steps=[('standardscaler', StandardScaler()),
                                        ('logisticregression',
                                         LogisticRegression(C=1, solver='liblinear'))])}
```

In [47]:
```
best_estimators['svm'].score(X_test,y_test)
```

Out[47]:
```
0.9024390243902439
```

In [48]:
```
best_estimators['random_forest'].score(X_test,y_test)
```

Out[48]:
```
0.6097560975609756
```

In [49]:
```
best_estimators['logistic_regression'].score(X_test,y_test)
```

Out[49]:
```
0.8780487804878049
```

❖ Our Code: Visualization

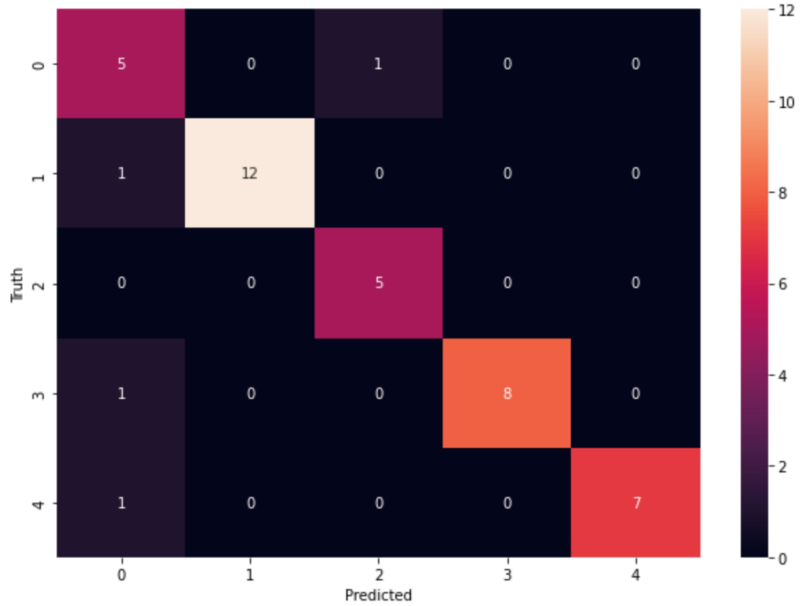
In [50]:
```
best_clf = best_estimators['svm']
```

In [51]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, best_clf.predict(X_test))
cm
```

Out[51]:
```
array([[ 5,  0,  1,  0,  0],
       [ 1, 12,  0,  0,  0],
       [ 0,  0,  5,  0,  0],
       [ 1,  0,  0,  8,  0],
       [ 1,  0,  0,  0,  7]])
```

In [52]:
```
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[52]:
```
Text(69.0, 0.5, 'Truth')
```



❖ Our Code:

In [53]:
```
class_dict
```

Out[53]:
```
{'maria_sharapova': 0,
 'virat_kohli': 1,
 'lionel_messi': 2,
 'serena_williams': 3,
 'roger_federer': 4}
```

Save the trained model

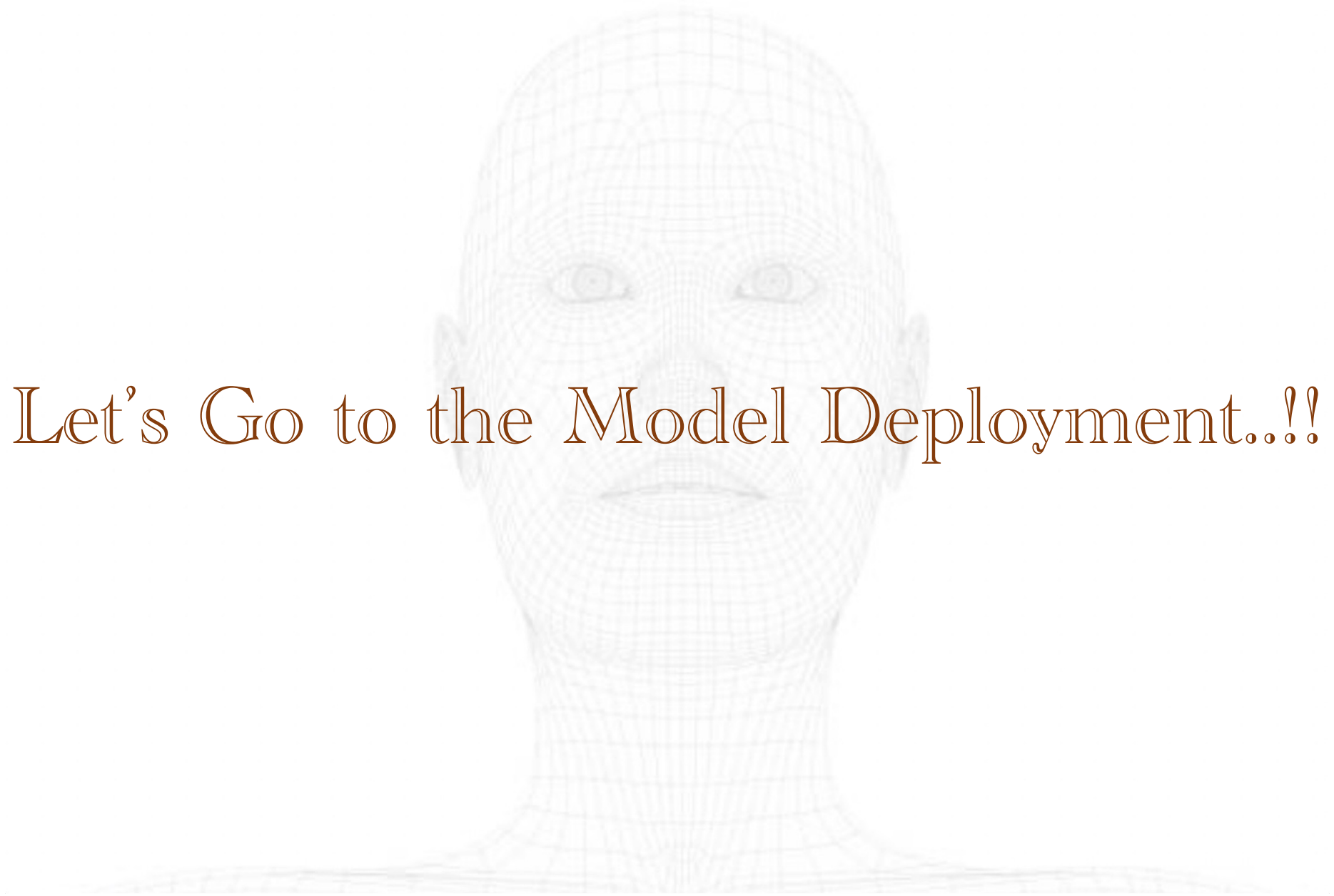
In [54]:
```
import joblib
# Save the model as a pickle in a file
joblib.dump(best_clf, 'saved_model.pkl')
```

Out[54]:
```
['saved_model.pkl']
```

Save class dictionary

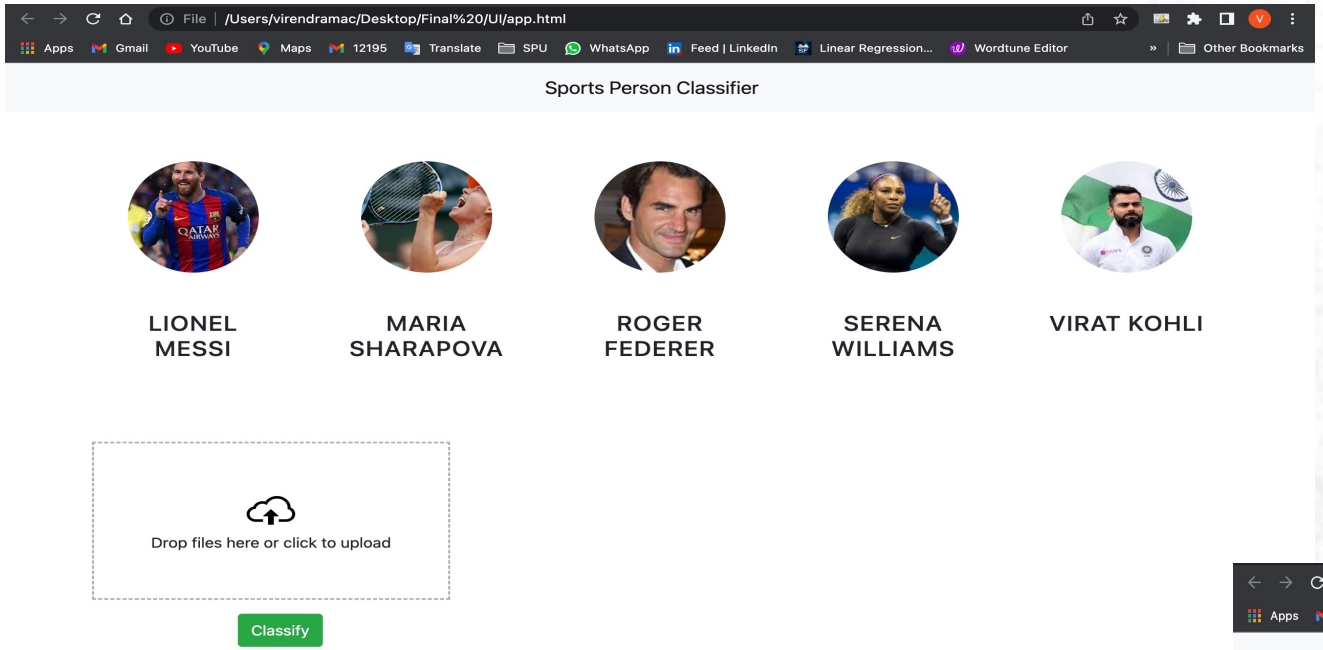
In [55]:
```
import json
with open("class_dictionary.json","w") as f:
    f.write(json.dumps(class_dict))
```

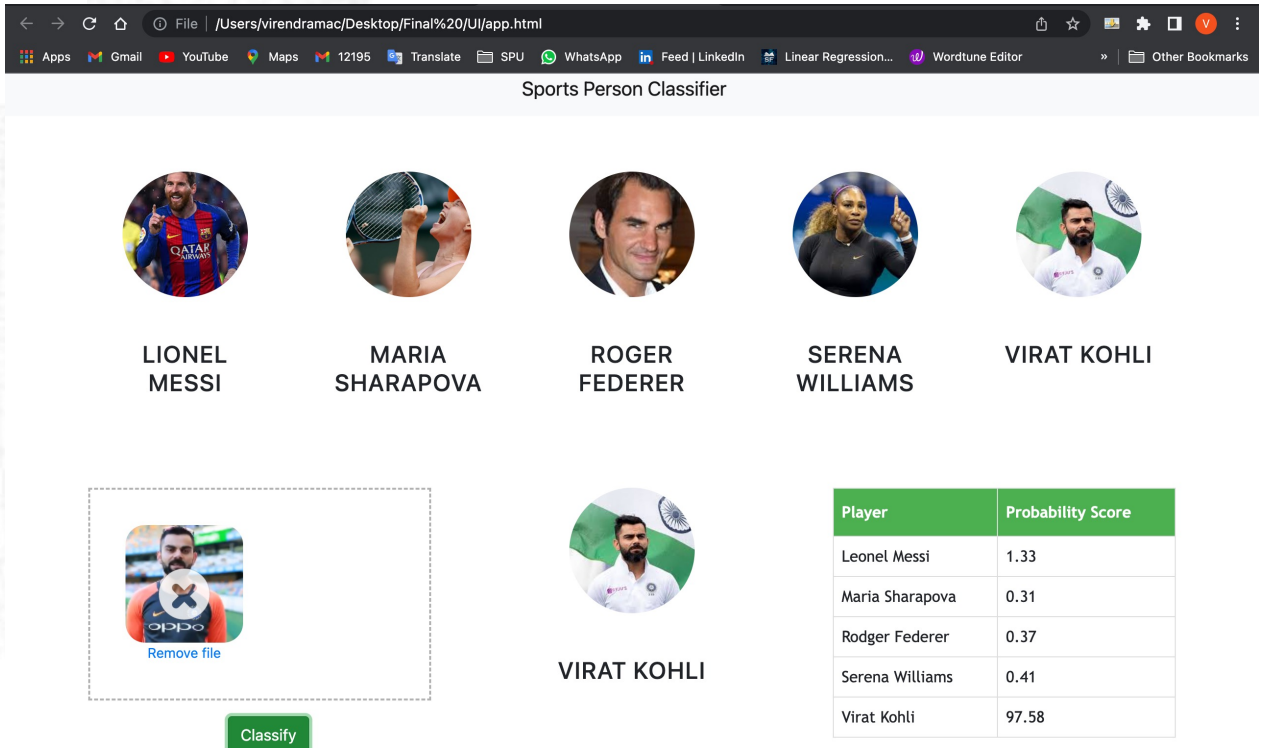
In [ ]:
```

```



Let's Go to the Model Deployment..!!

❖ Project Model Deployment :





Player	Probability Score
Leonel Messi	1.33
Maria Sharapova	0.31
Rodger Federer	0.37
Serena Williams	0.41
Virat Kohli	97.58

THANK YOU VERY MUCH..!!