```
!pip install mtcnn
```

```
    Collecting mtcnn
      Downloading mtcnn-0.1.1-py3-none-any.whl (2.3 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 9.5 MB/s eta 0:
    Requirement already satisfied: keras>=2.0.0 in /usr/local/lib/python3.10/dist-p
    Requirement already satisfied: opencv-python>=4.1.0 in /usr/local/lib/python3.1
    Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-
    Installing collected packages: mtcnn
    Successfully installed mtcnn-0.1.1
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call d
```

```
import cv2 as cv
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

```
img = cv.imread("/content/drive/MyDrive/dataset/vijay/v6.jpeg")
# opencv BGR channel format and plt reads images as RGB channel format
```

```
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img) # RGB
```

```
    <matplotlib.image.AxesImage at 0x79b9d82fb4c0>
```

```
from mtcnn.mtcnn import MTCNN

detector = MTCNN()
results = detector.detect_faces(img)
```

```
    1/1 [==============================] - 1s 711ms/step
    1/1 [==============================] - 0s 220ms/step
    1/1 [==============================] - 0s 70ms/step
    1/1 [==============================] - 0s 56ms/step
    1/1 [==============================] - 0s 44ms/step
    1/1 [==============================] - 0s 40ms/step
    1/1 [==============================] - 0s 38ms/step
    1/1 [==============================] - 0s 38ms/step
    1/1 [==============================] - 0s 35ms/step
    1/1 [==============================] - 0s 35ms/step
    1/1 [==============================] - 0s 44ms/step
```

```
      10/10 [==============================] - 0s 13ms/step
      1/1 [==============================] - 0s 236ms/step
```

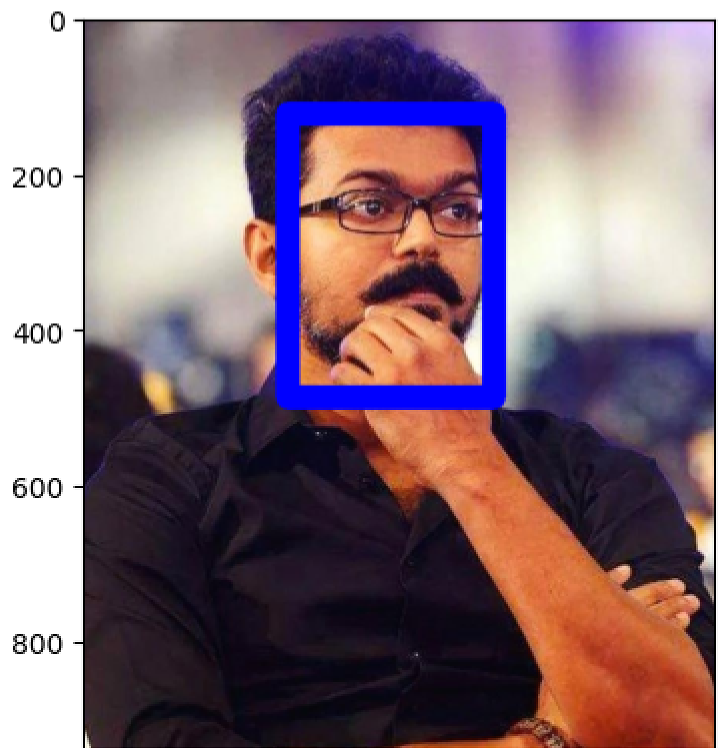
```
results
```

```
      [{'box': [262, 123, 263, 364],
        'confidence': 0.9997992515563965,
        'keypoints': {'left_eye': (368, 239),
         'right_eye': (483, 253),
         'nose': (438, 280),
         'mouth_left': (361, 367),
         'mouth_right': (466, 378)}}]
```

```
x,y,w,h = results[0]['box']
```

```
img = cv.rectangle(img, (x,y), (x+w, y+h), (0,0,255), 30)
plt.imshow(img)
```

```
      <matplotlib.image.AxesImage at 0x79b9c87c3460>
```



```
my_face = img[y:y+h, x:x+w]
#Facenet takes as input 160x160
my_face = cv.resize(my_face, (160,160))
plt.imshow(my_face)
```

```
      <matplotlib.image.AxesImage at 0x79b9d065ab60>
```

```
my_face
```

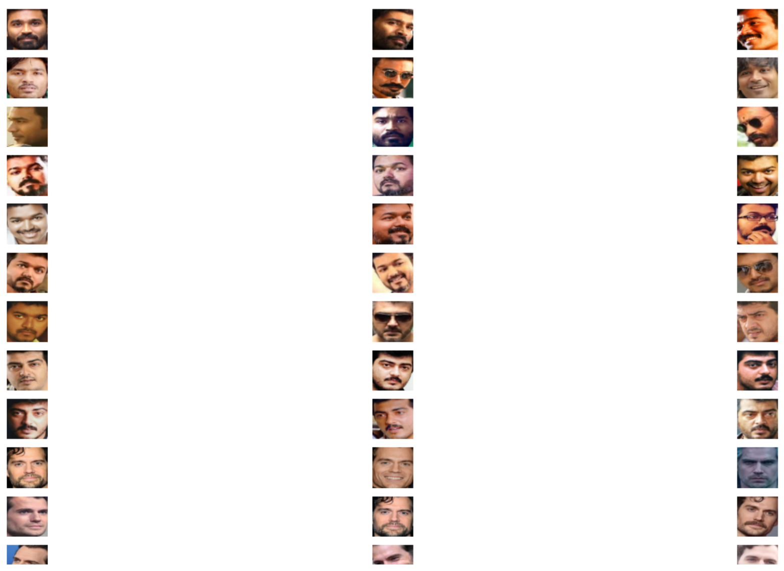
```
      array([[[  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255],
               ...,
               [  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255]],

              [[  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255],
               ...,
               [  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255]],

              [[  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255],
               ...,
               [  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255]],

              ...,

              [[  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255],
               ...,
               [  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255]],

              [[  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255],
               ...,
               [  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255]],

              [[  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255],
               ...,
               [  0,   0, 255],
               [  0,   0, 255],
               [  0,   0, 255]]], dtype=uint8)
```

```
class FACELOADING:
    def __init__(self, directory):
        self.directory = directory
        self.target_size = (160,160)
        self.X = []
        self.Y = []
        self.detector = MTCNN()


    def extract_face(self, filename):
        img = cv.imread(filename)
        img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
        x,y,w,h = self.detector.detect_faces(img)[0]['box']
        x,y = abs(x), abs(y)
        face = img[y:y+h, x:x+w]
        face_arr = cv.resize(face, self.target_size)
        return face_arr


    def load_faces(self, dir):
        FACES = []
        for im_name in os.listdir(dir):
            try:
                path = dir + im_name
                single_face = self.extract_face(path)
                FACES.append(single_face)
            except Exception as e:
                pass
        return FACES

    def load_classes(self):
        for sub_dir in os.listdir(self.directory):
            path = self.directory +'/'+ sub_dir+'/'
            FACES = self.load_faces(path)
            labels = [sub_dir for _ in range(len(FACES))]
            print(f"Loaded successfully: {len(labels)}")
            self.X.extend(FACES)
            self.Y.extend(labels)

        return np.asarray(self.X), np.asarray(self.Y)


    def plot_images(self):
        plt.figure(figsize=(18,16))
        for num,image in enumerate(self.X):
            ncols = 3
            nrows = len(self.Y)//ncols + 1
            plt.subplot(nrows,ncols,num+1)
            plt.imshow(image)
            plt.axis('off')


faceloading = FACELOADING("/content/drive/MyDrive/dataset")
X, Y = faceloading.load_classes()
```

```
1/1 [==============================] - 0s 213ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
9/9 [==============================] - 0s 11ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
4/4 [==============================] - 0s 9ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 604ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
```

```
1/1 [==============================] - 0s 22ms/step
163/163 [==============================] - 2s 10ms/step
3/3 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
6/6 [==============================] - 0s 9ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 77ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 37ms/step
```

```
plt.figure(figsize=(16,12))
for num,image in enumerate(X):
    ncols = 3
    nrows = len(Y)//ncols + 1
    plt.subplot(nrows,ncols,num+1)
    plt.imshow(image)
    plt.axis('off')
```



```
!pip install keras-facenet
```

```
    Collecting keras-facenet
      Downloading keras-facenet-0.3.2.tar.gz (10 kB)
      Preparing metadata (setup.py) ... done
    Requirement already satisfied: mtcnn in /usr/local/lib/python3.10/dist-packages
    Requirement already satisfied: keras>=2.0.0 in /usr/local/lib/python3.10/dist-p
    Requirement already satisfied: opencv-python>=4.1.0 in /usr/local/lib/python3.1
    Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-
    Building wheels for collected packages: keras-facenet
      Building wheel for keras-facenet (setup.py) ... done
      Created wheel for keras-facenet: filename=keras_facenet-0.3.2-py3-none-any.wh
      Stored in directory: /root/.cache/pip/wheels/1d/d8/a9/85cf04ea29321d2afcb82c0
    Successfully built keras-facenet
    Installing collected packages: keras-facenet
    Successfully installed keras-facenet-0.3.2
```

```
from keras_facenet import FaceNet
embedder = FaceNet()

def get_embedding(face_img):
    face_img = face_img.astype('float32') # 3D(160x160x3)
    face_img = np.expand_dims(face_img, axis=0)
    # 4D (Nonex160x160x3)
    yhat= embedder.embeddings(face_img)
    return yhat[0] # 512D image (1x1x512)
```

```
EMBEDDED_X = []

for img in X:
    EMBEDDED_X.append(get_embedding(img))

EMBEDDED_X = np.asarray(EMBEDDED_X)
```

```
    1/1 [==============================] - 4s 4s/step
    1/1 [==============================] - 0s 95ms/step
    1/1 [==============================] - 0s 96ms/step
    1/1 [==============================] - 0s 94ms/step
    1/1 [==============================] - 0s 105ms/step
    1/1 [==============================] - 0s 96ms/step
    1/1 [==============================] - 0s 97ms/step
    1/1 [==============================] - 0s 100ms/step
    1/1 [==============================] - 0s 95ms/step
    1/1 [==============================] - 0s 97ms/step
    1/1 [==============================] - 0s 95ms/step
    1/1 [==============================] - 0s 96ms/step
    1/1 [==============================] - 0s 108ms/step
    1/1 [==============================] - 0s 97ms/step
    1/1 [==============================] - 0s 93ms/step
    1/1 [==============================] - 0s 98ms/step
    1/1 [==============================] - 0s 98ms/step
    1/1 [==============================] - 0s 95ms/step
    1/1 [==============================] - 0s 97ms/step
    1/1 [==============================] - 0s 92ms/step
    1/1 [==============================] - 0s 99ms/step
    1/1 [==============================] - 0s 95ms/step
    1/1 [==============================] - 0s 95ms/step
    1/1 [==============================] - 0s 98ms/step
```

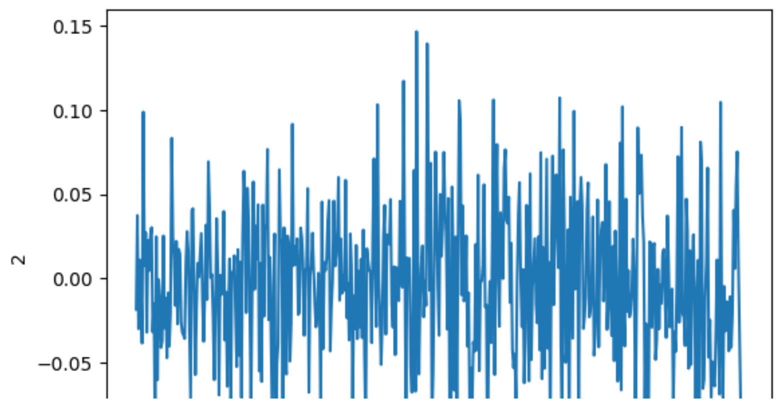
```
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 93ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 181ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 163ms/step
```

```
np.savez_compressed('faces_embeddings_done_4classes.npz', EMBEDDED_X, Y)
```

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
encoder.fit(Y)
Y = encoder.transform(Y)
```

```
plt.plot(EMBEDDED_X[0])
plt.ylabel(Y[0])
```

```
Text(0, 0.5, '2')
```



```
y
```

```
123
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(EMBEDDED_X, Y, shuffle=True, random_st
from sklearn.svm import SVC
model = SVC(kernel='linear', probability=True)
model.fit(X_train, Y_train)
```

```
▾                SVC
SVC(kernel='linear', probability=True)
```

```
ypreds_train = model.predict(X_train)
ypreds_test = model.predict(X_test)
```

```
from sklearn.metrics import accuracy_score

accuracy_score(Y_train, ypreds_train)
```

```
1.0
```

```
accuracy_score(Y_test,ypreds_test)
```

```
1.0
```

```
t_im = cv.imread("/content/d11.jpeg")
t_im = cv.cvtColor(t_im, cv.COLOR_BGR2RGB)
x,y,w,h = detector.detect_faces(t_im)[0]['box']
```

```
1/1 [==============================] - 0s 90ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 49ms/step
7/7 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 39ms/step
```

```
t_im = t_im[y:y+h, x:x+w]
t_im = cv.resize(t_im, (160,160))
test_im = get_embedding(t_im)
```

```
1/1 [==============================] - 0s 107ms/step
```

```
test_im = [test_im]
ypreds = model.predict(test_im)
```

```
plt.imshow(t_im)
encoder.inverse_transform(ypreds)
```

```
array(['dhanush'], dtype='<U12')
```