Mode detection from Image

In [22]:

```
#

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import matplotlib.pylab as plt
import tensorflow as tf
import numpy as np
import cv2
import os
```

In [23]:

```
#image load
img=image.load_img("/Users/myyntiimac/Desktop/CNN-Happy or Sad/training/happy/IM
img
```

Out[23]:

In [24]:

```
#image converted to array
i1= cv2.imread("/Users/myyntiimac/Desktop/CNN-Happy or Sad/training/happy/5.jpg"
i1
```

Out[24]:

```
array([[[213, 224,  38],
        [216, 227,  40],
        [220, 231,  39],
        ...,
        [145, 177, 183],
        [145, 177, 183],
        [145, 177, 183]],

       [[207, 223,  46],
        [209, 225,  47],
        [213, 228,  46],
        ...,
        [145, 177, 183],
        [145, 177, 183],
        [145, 177, 183]],

       [[197, 218,  65],
        [198, 219,  63],
        [198, 220,  61],
        ...,
        [145, 177, 182],
        [145, 177, 182],
        [145, 177, 182]],

       ...,

       [[126, 104,  63],
        [133, 111,  70],
        [145, 123,  82],
        ...,
        [ 71,  46,  42],
        [ 71,  46,  44],
        [ 70,  44,  44]],

       [[136, 114,  73],
        [143, 121,  80],
        [153, 131,  90],
        ...,
        [ 77,  52,  50],
        [ 78,  52,  52],
        [ 78,  51,  54]],

       [[145, 123,  82],
        [151, 129,  88],
        [158, 136,  95],
        ...,
        [ 79,  53,  53],
        [ 80,  53,  56],
        [ 80,  53,  57]]], dtype=uint8)
```

In [25]:

```
#shape of image(height, width and RGB)
i1.shape
```

Out[25]:

```
(1280, 959, 3)
```

In [26]:

```
train=ImageDataGenerator(rescale=1/255)
validation=ImageDataGenerator(rescale=1/255)
```

In [27]:

```
train_dataset=train.flow_from_directory("/Users/myyntiimac/Desktop/CNN-Happy or
                                        target_size=(200,200),
                                        batch_size=3,
                                        class_mode='binary')

```

```
Found 12 images belonging to 2 classes.
```

In [28]:

```
validation_dataset=validation.flow_from_directory("/Users/myyntiimac/Desktop/CNN
                                                  target_size=(200,200),
                                                  batch_size=3,
                                                  class_mode='binary')
```

```
Found 0 images belonging to 2 classes.
```

In [29]:

```
 train_dataset.class_indices
```

Out[29]:

```
{'happy': 0, 'not happy': 1}
```

In [30]:

```
train_dataset.classes
```

Out[30]:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1], dtype=int32)
```

In [31]:

```
#define the CNN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200,
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

In [32]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 198, 198, 16)      448
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 99, 99, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 97, 97, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               17334784
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 17,358,881
Trainable params: 17,358,881
Non-trainable params: 0
_________________________________________________________________
```

In [33]:

```
#comile the model
model.compile(loss='binary_crossentropy',
              optimizer=tf.keras.optimizers.RMSprop(lr=0.001),
              metrics=['acc'])
```

In [34]:

```
#Train the model
model_fit=model.fit(train_dataset,
                    epochs=5,
                    validation_data=validation_dataset)

```

```
Train for 4 steps
Epoch 1/5
4/4 [==============================] - 4s 1s/step - loss: 10.8576 - ac
c: 0.5000
Epoch 2/5
4/4 [==============================] - 3s 626ms/step - loss: 1.8091 -
acc: 0.5833
Epoch 3/5
4/4 [==============================] - 2s 577ms/step - loss: 0.5153 -
acc: 0.7500
Epoch 4/5
4/4 [==============================] - 2s 575ms/step - loss: 0.3106 -
acc: 0.9167
Epoch 5/5
4/4 [==============================] - 2s 575ms/step - loss: 0.4749 -
acc: 0.9167
```

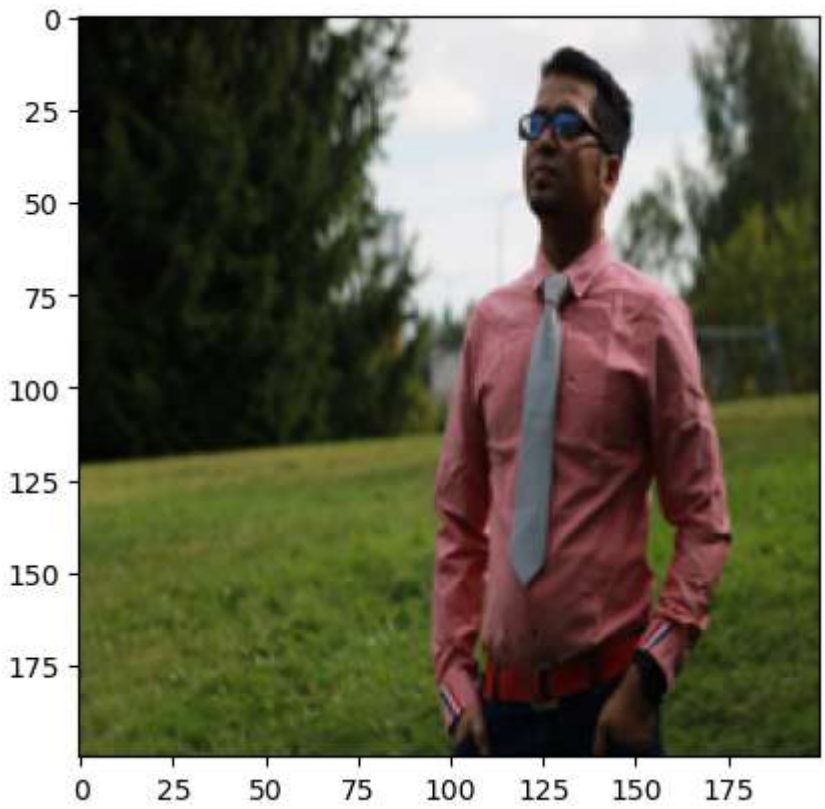
In [47]:

```
#Preprocess the testing image in size 200,200
dir_path = "/Users/myyntiimac/Desktop/CNN-Happy or Sad/testing"
for i in os.listdir(dir_path ):
    print(i)

```

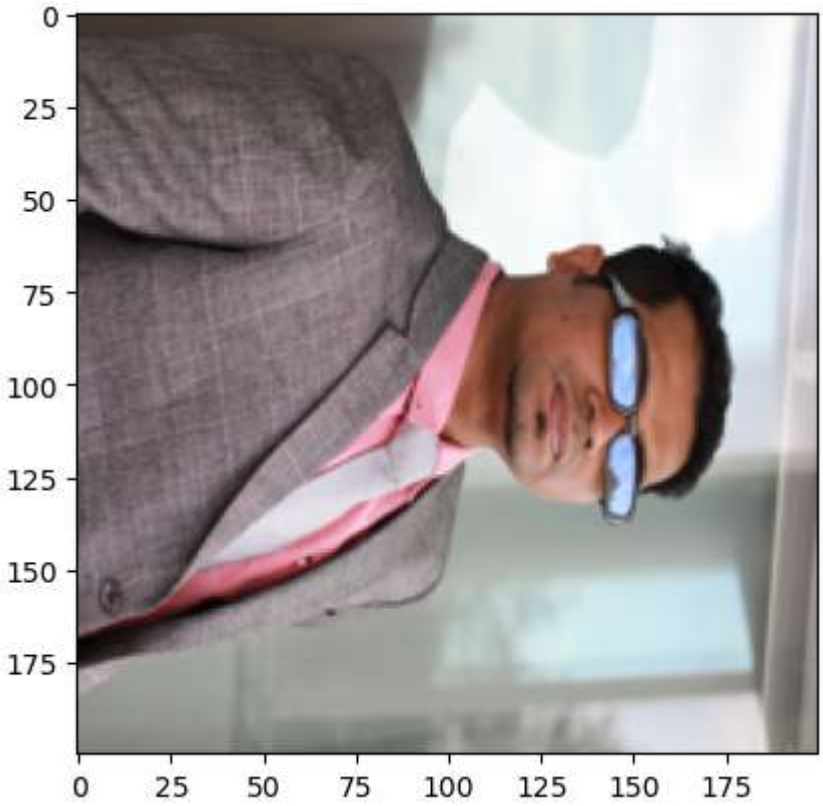
```
IMG_4512.JPG
.DS_Store
IMG_4498.JPG
IMG_4513.JPG
IMG_4497.JPG
```

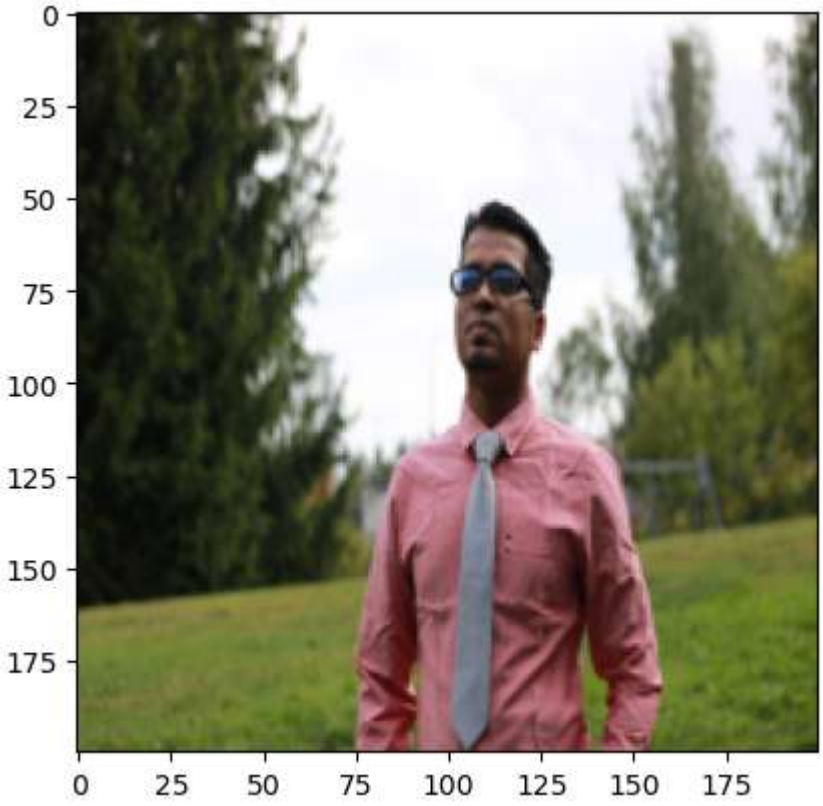
In [49]:

```
from PIL import Image
from PIL import Image as pil_image
dir_path = "/Users/myyntiimac/Desktop/CNN-Happy or Sad/testing"

for i in os.listdir(dir_path):
    # Create the full file path
    file_path = os.path.join(dir_path, i)

    # Check if the file is an image
    if os.path.isfile(file_path) and i.lower().endswith(('.png', '.jpg', '.jpeg'
        img = Image.open(file_path)
        img = img.resize((200, 200))
        plt.imshow(img)
        plt.show()
    else:
        print(f"Skipped non-image file: {i}")
```

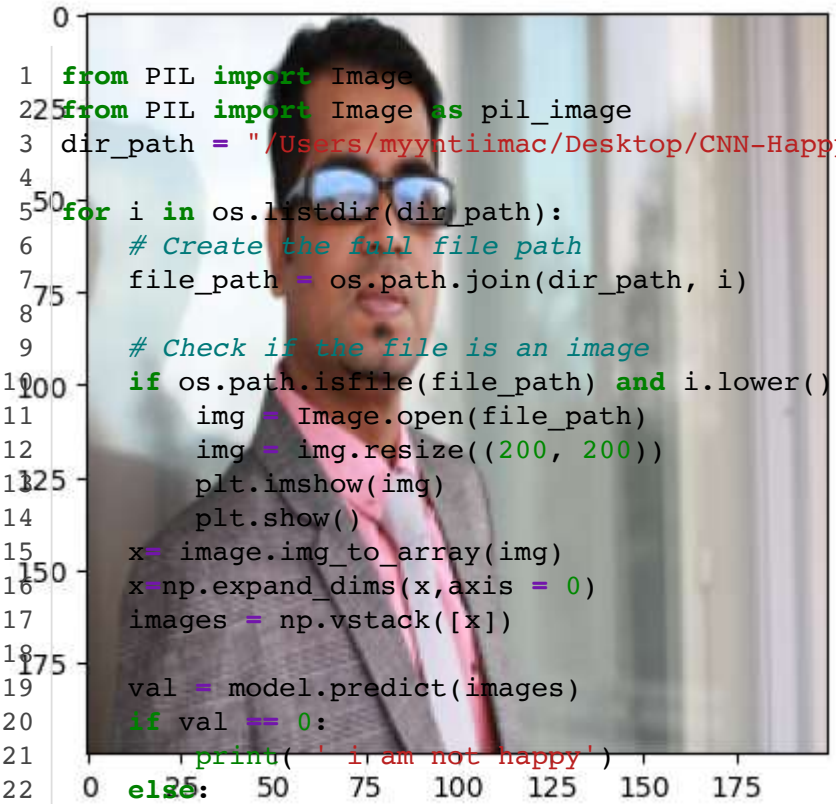



```
Skipped non-image file: .DS_Store
```


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

```
from PIL import Image
from PIL import Image as pil_image
dir_path = "/Users/myyntiimac/Desktop/CNN-Happy or Sad/testing"

for i in os.listdir(dir_path):
    # Create the full file path
    file_path = os.path.join(dir_path, i)

    # Check if the file is an image
    if os.path.isfile(file_path) and i.lower().endswith(('.png', '.jpg', '.jpeg'
        img = Image.open(file_path)
        img = img.resize((200, 200))
        plt.imshow(img)
        plt.show()
    x= image.img_to_array(img)
    x=np.expand_dims(x,axis = 0)
    images = np.vstack([x])

    val = model.predict(images)
    if val == 0:
        print( ' i am not happy')
    else:
        print('i am happy')
```



In [ ]:

```
1
```