**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**M. Tech., Fall Semester 2021-2022**

In partial fulfillment of the requirements for the course of

**SWE 1010-Digital Image Processing**
**IN**
**INTEGRATED M. TECH CSE SPECIALIZATION IN BUSINESS ANALYTICS**

# Mask and ID card Detection

**Course Faculty**
Prof. Braveen

**Batch Mates**
Mohit More -19MIA1005
Gitansh Saharan -19MIA1060
Sayantan Nandy – 19MIA1049

# Acknowledgement

We sincerely thank our Chancellor - Dr. G. Viswanathan, VIT University, for giving us the opportunity to pursue this course of Digital Image Processing and introduce J-component as part of our academic curriculum. We also thank Dr. Braveen for giving us the opportunity to do this project. It was a wonderful experience working on this project which made us learn many things about mask and ID card detection.

# Certificate

This is to certify that the project work titled “Mask and ID card Detection” that is being submitted by Gitansh Saharan , Mohit More and Sayantan Nandy for Digital Image Processing  is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place : VIT Chennai

Date: 08/12/2021

SIGNATURE OF THE FACULTY Dr. Braveen

## Table of Contents

# Abstract

The rapid outbreak of COVID-19 has caused serious harm and infected tens ofmillions of people worldwide. Since there is no specific treatment, wearing masks has become an effective method to prevent the transmission of COVID-19 and is required in most public areas, which has also led to a growing demand for automatic real-time mask detection services to replace manual reminding.

Also to maintain the safety precautions and run the universities with the whole safety it’s necessary to maintain a bio-bubble which means there should not be any contact from out side and in order to maintain that gate security guards should be at a distant.
To contribute towards communal health, this paper aims to devise a highly accurate and real-time technique that can efficiently detect non-mask faces and people not wearing ID card in public and thus, enforce the wearing of masks and ID card. We have used deep learning and open CV to develop our face detector model.

# Introduction

Coronavirus Disease unexpectedly broke out in 2019 and has seriously affected the whole world. As of now, COVID-19 has infected more than 125 million people worldwide and caused over 2.7 million deaths. One of the transmission routes of COVID-19 is through droplets of saliva or nasal secretions when an infected person coughs or sneezes, which is highly infectious and could be worse in crowded places. Recently, a study on understanding measures to tackle the COVID-19 pandemic reveals that wearing a face mask or other covering over the nose and mouth cuts the risk of Coronavirus spread by avoiding forward distance traveled by a person's exhaled breath by more than 90%. To mandate the use of facemask, it becomes essential to devise some technique that enforce individuals to apply a mask before exposure to public places. Due to coronavirus all the universities started running online, but when everything got normal , all the universities need to maintain bio bubble,

Through this paper, we aim to build a computer vision system that can detect if a person is wearing a mask and id or not.

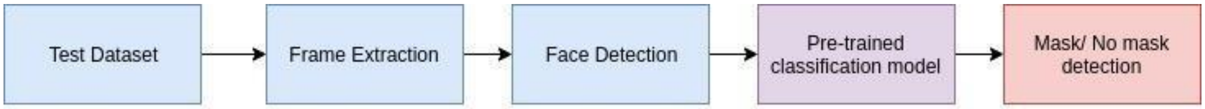
# Existing System

## Existing system is to Detect Mask

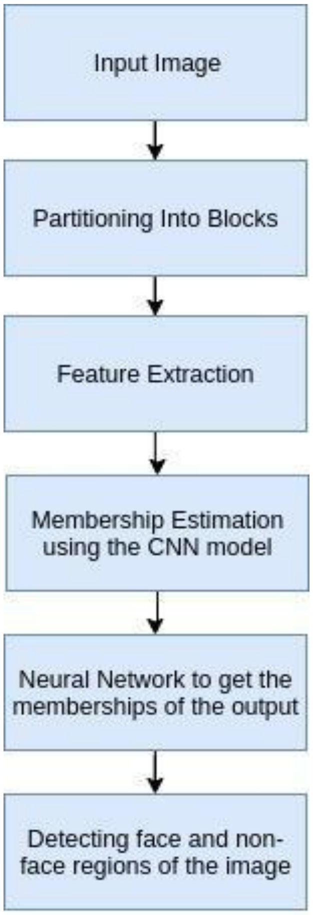
### *Methodology Being Used*

We have used the following steps for our project:

- The main task of our project is to recognize faces correctly and to do that we need tobuild a dataset of faces wearing face masks.
- In the dataset, we apply face detection to compute the bounding box location of theface in the image.
- Once we know *where* in the image the face is, we can extract the face Region ofInterest (ROI):
- And from there, we apply facial landmarks, allowing us to localize the eyes, nose,mouth, etc.
- Then we will again use facial landmarks to find if the person is wearing a mask ornot.



Phases for building a mask detector with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.


Input Image
Partitioning Into Blocks
Feature Extraction
Membership Estimation using the CNN model
Neural Network to get the memberships of the output
Detecting face and non-face regions of the image

CNN face mask detection

**Drawback :**

1) Less accuracy
2) Its only limited to mask detection using python

## Motivation

The COVID-19 pandemic has changed life as we know it. To handle the pandemic Government imposed night curfews, aware people of the importance of masks and social distancing, and with this masks became an important part of our lifestyle.

Also in order to handle the crises, VIT shifted its education system to online. But as the situation is improving, things are getting back to normal but still, we have to take precautions.

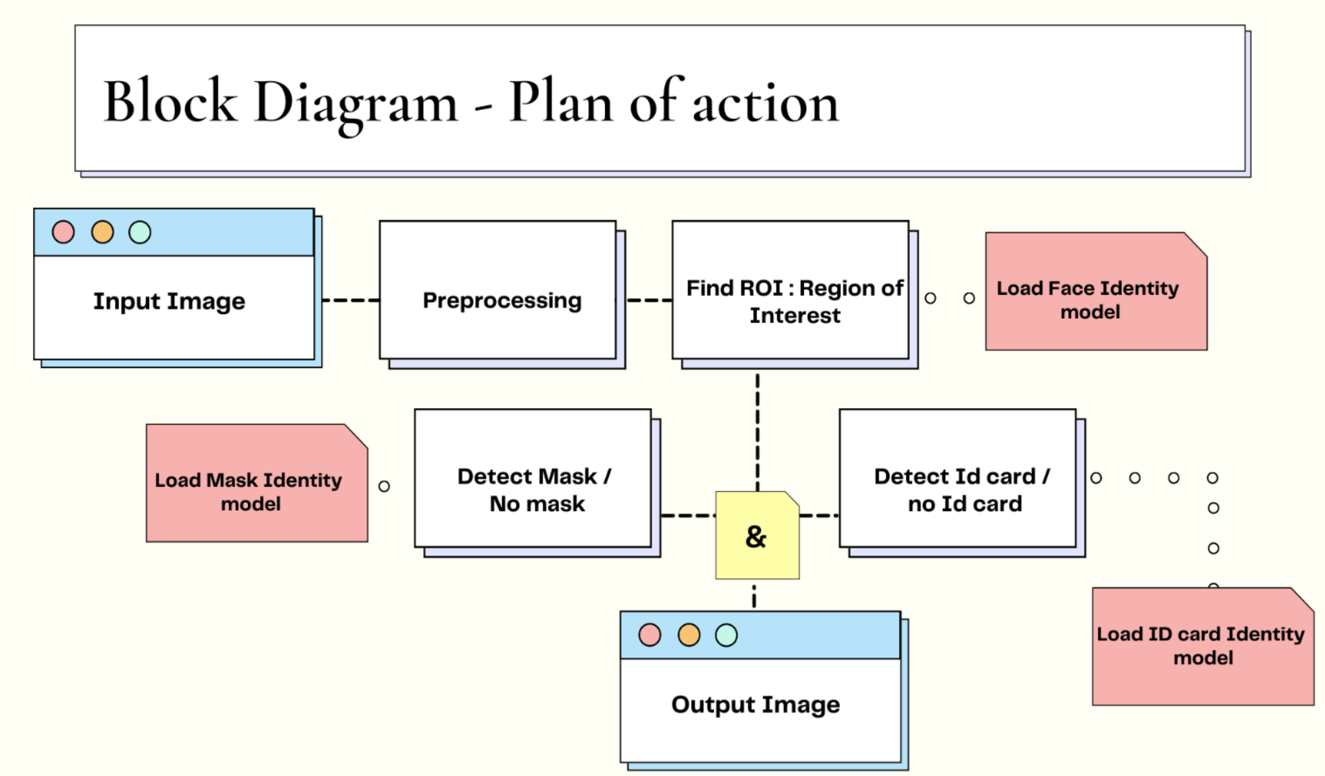
In VIT, an Id card is an important part of a VITian life, and as we all know we are not even allowed to enter any academic block without the ID card. If things get back to normal, and in order to maintain BIO BUBBLE, all the security guards need to stay inside the campus which is a bit difficult.

So in order to maintain discipline and ensure that students are wearing masks and carrying ID cards, we can deploy this software. With the help of this software, guards from an appropriate distance can keep a check on students.

## Proposed system

Our image recognition model is trained in such a manner that if a student is wearing only a mask or only id card and not even both, a red frame will appear on his/her face that will be visible on the security guard monitor in the control room and from a distance guard can instruct to wear the mask and ID card.

*Flow Diagram :*


Block Diagram - Plan of action
Input Image
Preprocessing
Find ROI : Region of Interest
Load Face Identity model
Load Mask Identity model
Detect Mask / No mask
&
Detect Id card / no Id card
Load ID card Identity model
Output Image

### Step of Implementation:

Load images using Python or any other programming you are working on.

- Convert images into an array

- And finally apply some algorithm on that array

Another good thing is that we have a library known as OpenCV which will help us to read the image and return an array of color pixels.

## Mask And Id Card Detection using open CV:

### Libraries Used:

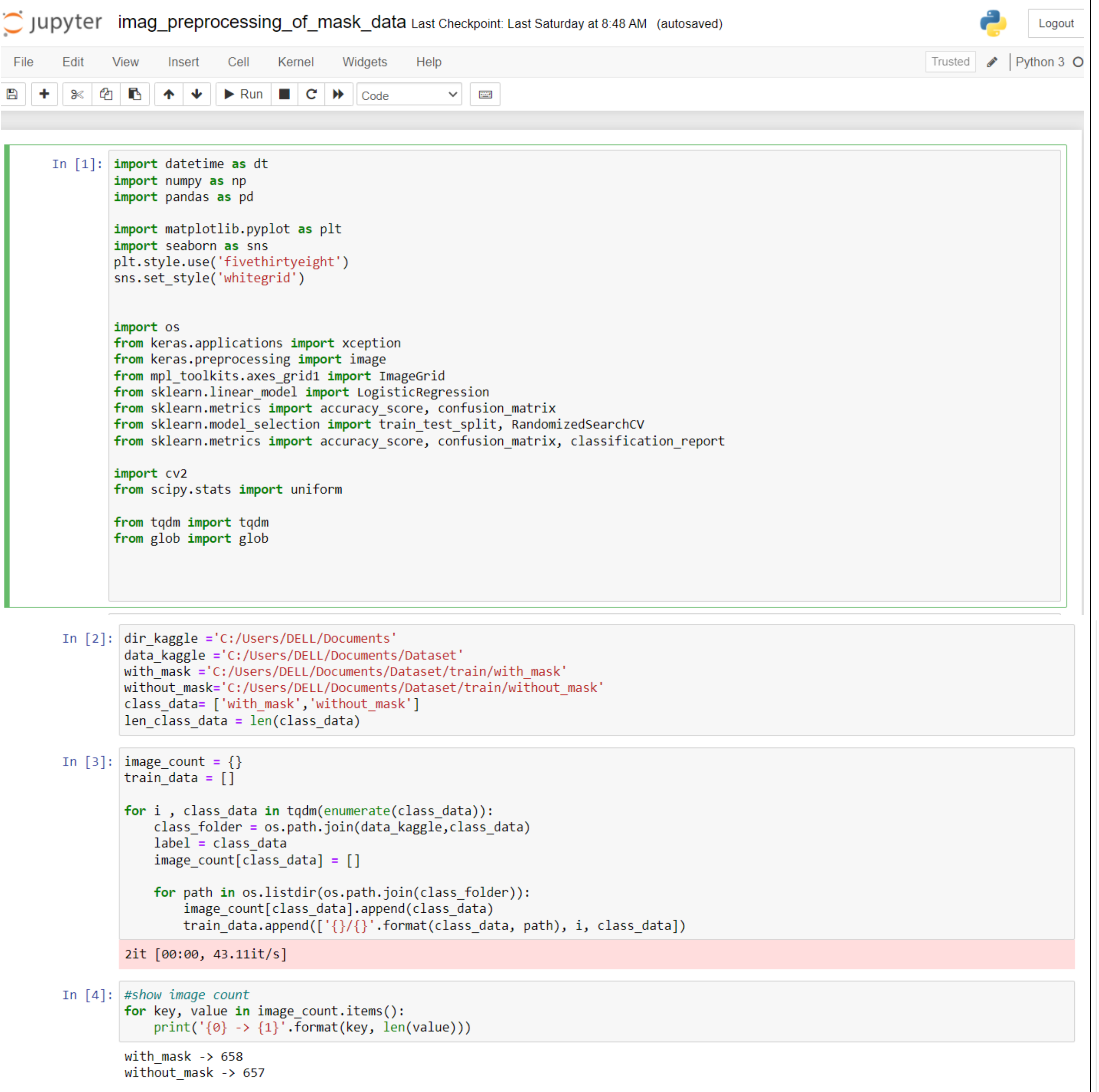
- Tensorflow
- Keras
- Imutils
- Numpy
- OpenCV
- Matplotlib
- Scipy
- Scikit

## Datasets Being Used

For the experiment, we use the dataset which consists of 600 masked faces with a minimum size of 32×32. The faces in this dataset have different orientation and occlusion degrees. We select 400 images that contain frontal faces from the dataset.The dataset is divided into 3 parts for training and validation and a test set with 400,120, and 80 images, respectively.

# Sample Source Code / Implementation

## Code and Output Image Processing of Mask Dataset :


jupyter
imag_preprocessing_of_mask_data Last Checkpoint: Last Saturday at 8:48 AM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

In [1]:

```
import datetime as dt
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
sns.set_style('whitegrid')


import os
from keras.applications import xception
from keras.preprocessing import image
from mpl_toolkits.axes_grid1 import ImageGrid
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import cv2
from scipy.stats import uniform

from tqdm import tqdm
from glob import glob
```

In [2]:

```
dir_kaggle ='C:/Users/DELL/Documents'
data_kaggle ='C:/Users/DELL/Documents/Dataset'
with_mask ='C:/Users/DELL/Documents/Dataset/train/with_mask'
without_mask='C:/Users/DELL/Documents/Dataset/train/without_mask'
class_data= ['with_mask','without_mask']
len_class_data = len(class_data)
```

In [3]:

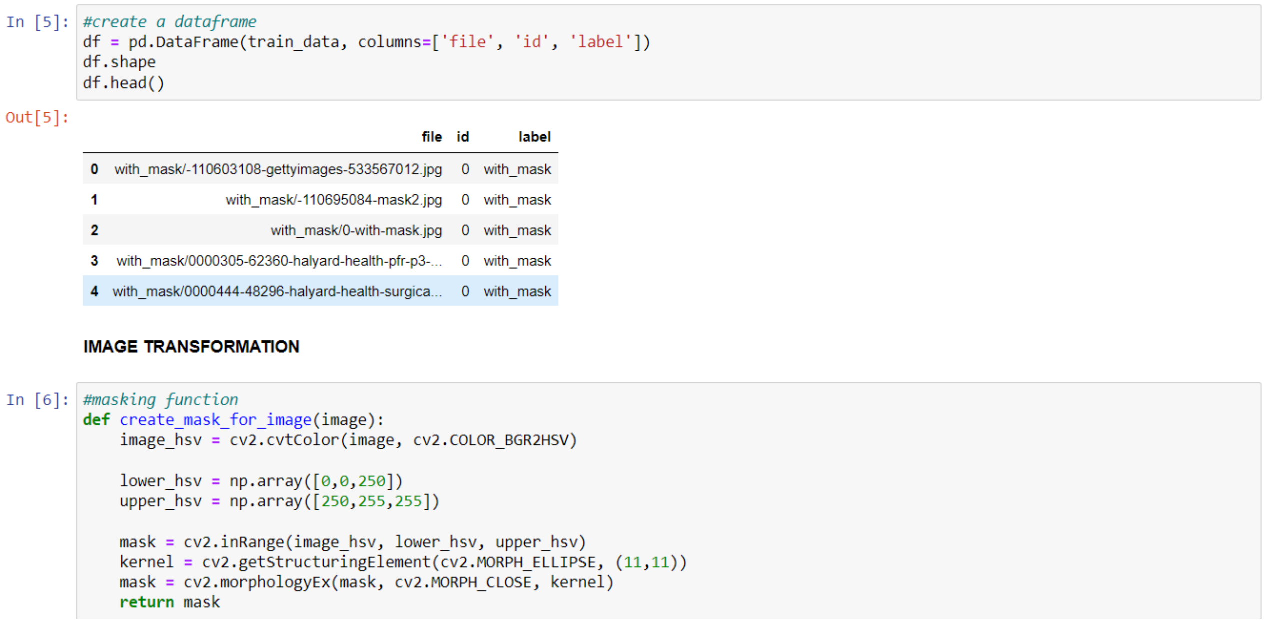
```
image_count = {}
train_data = []

for i , class_data in tqdm(enumerate(class_data)):
    class_folder = os.path.join(data_kaggle,class_data)
    label = class_data
    image_count[class_data] = []

    for path in os.listdir(os.path.join(class_folder)):
        image_count[class_data].append(class_data)
        train_data.append(['{}/{}'.format(class_data, path), i, class_data])
```

```
2it [00:00, 43.11it/s]
```

In [4]:

```
#show image count
for key, value in image_count.items():
    print('{0} -> {1}'.format(key, len(value)))
```

```
with_mask -> 658
without_mask -> 657
```



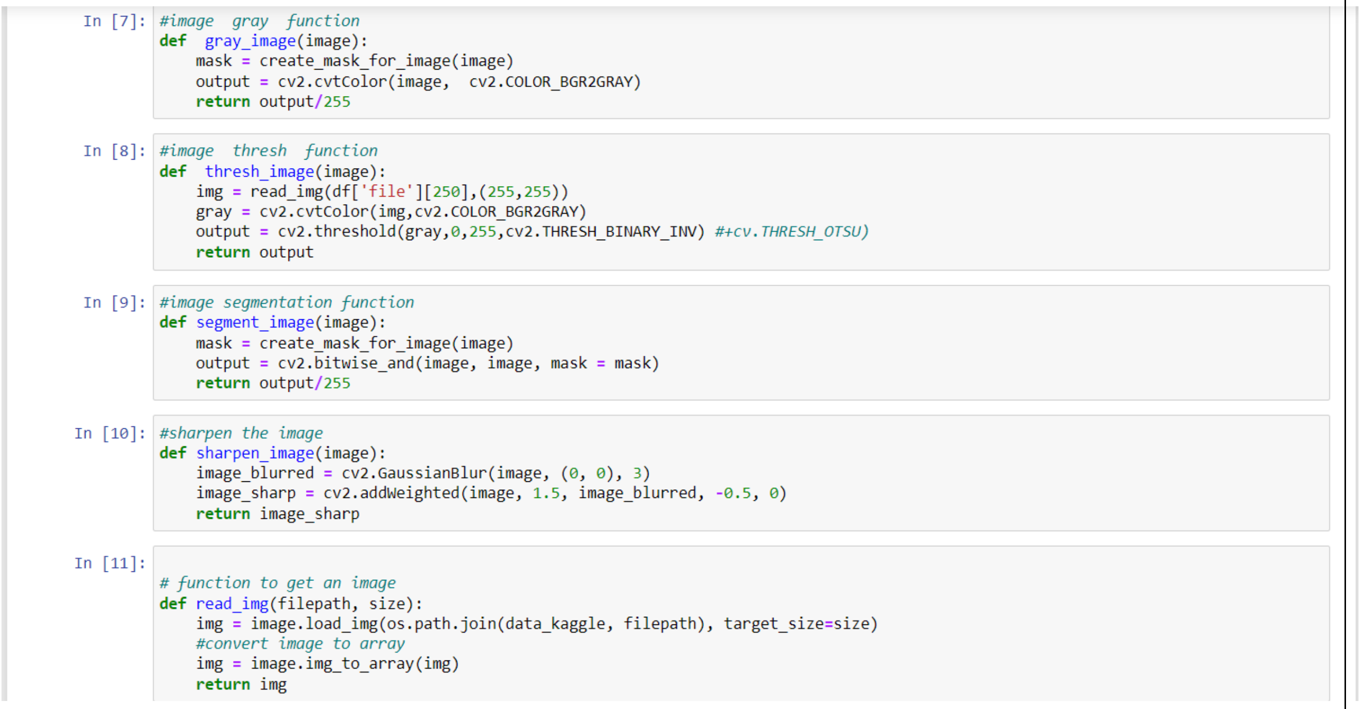
```
In [5]: #create a dataframe
        df = pd.DataFrame(train_data, columns=['file', 'id', 'label'])
        df.shape
        df.head()
```

Out[5]:

| | file | id | label |
|---|---|---|---|
| 0 | with_mask/-110603108-gettyimages-533567012.jpg | 0 | with_mask |
| 1 | with_mask/-110695084-mask2.jpg | 0 | with_mask |
| 2 | with_mask/0-with-mask.jpg | 0 | with_mask |
| 3 | with_mask/0000305-62360-halyard-health-pfr-p3-... | 0 | with_mask |
| 4 | with_mask/0000444-48296-halyard-health-surgica... | 0 | with_mask |

### IMAGE TRANSFORMATION

```
In [6]: #masking function
        def create_mask_for_image(image):
            image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

            lower_hsv = np.array([0,0,250])
            upper_hsv = np.array([250,255,255])

            mask = cv2.inRange(image_hsv, lower_hsv, upper_hsv)
            kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11))
            mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
            return mask
```



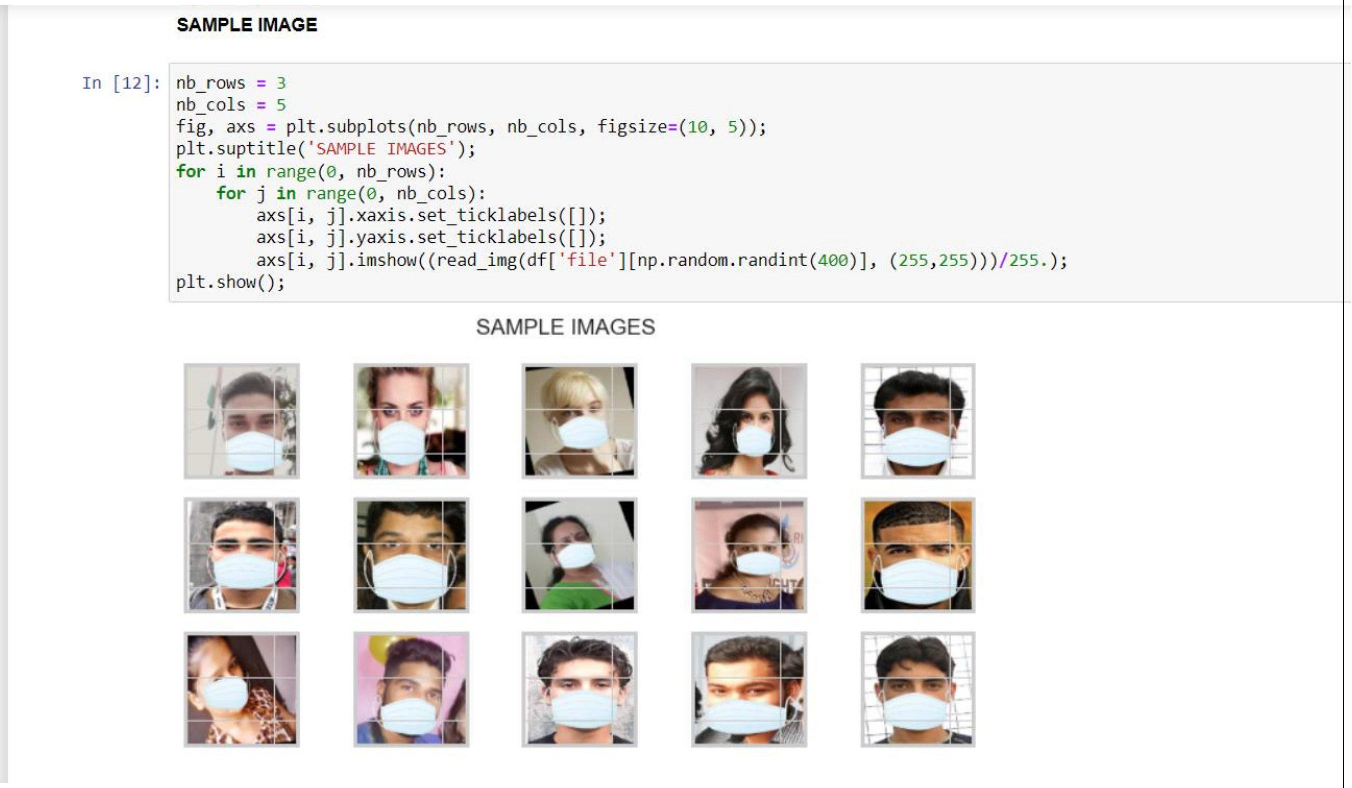
```
In [7]: #image  gray  function
        def  gray_image(image):
            mask = create_mask_for_image(image)
            output = cv2.cvtColor(image,  cv2.COLOR_BGR2GRAY)
            return output/255
```

```
In [8]: #image  thresh  function
        def  thresh_image(image):
            img = read_img(df['file'][250],(255,255))
            gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            output = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV) #+cv.THRESH_OTSU)
            return output
```

```
In [9]: #image segmentation function
        def segment_image(image):
            mask = create_mask_for_image(image)
            output = cv2.bitwise_and(image, image, mask = mask)
            return output/255
```

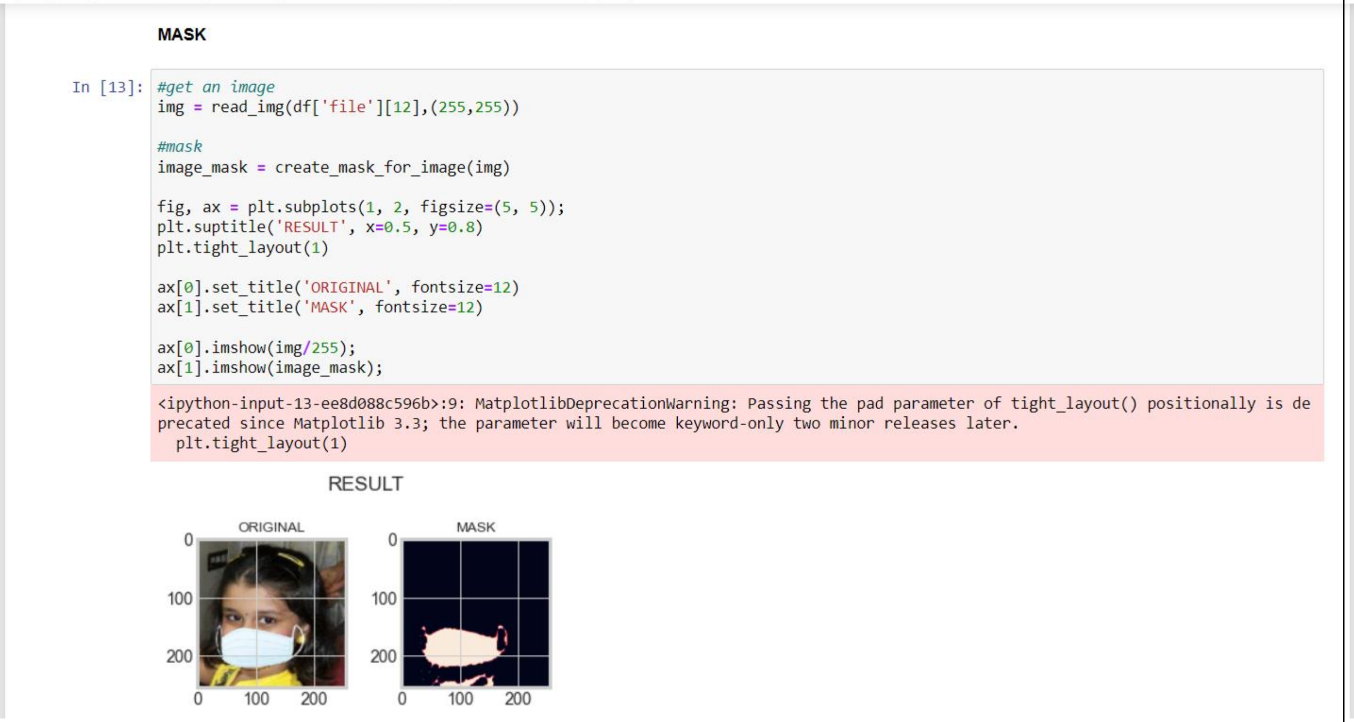
```
In [10]: #sharpen the image
         def sharpen_image(image):
             image_blurred = cv2.GaussianBlur(image, (0, 0), 3)
             image_sharp = cv2.addWeighted(image, 1.5, image_blurred, -0.5, 0)
             return image_sharp
```

```
In [11]:
         # function to get an image
         def read_img(filepath, size):
             img = image.load_img(os.path.join(data_kaggle, filepath), target_size=size)
             #convert image to array
             img = image.img_to_array(img)
             return img
```



SAMPLE IMAGE

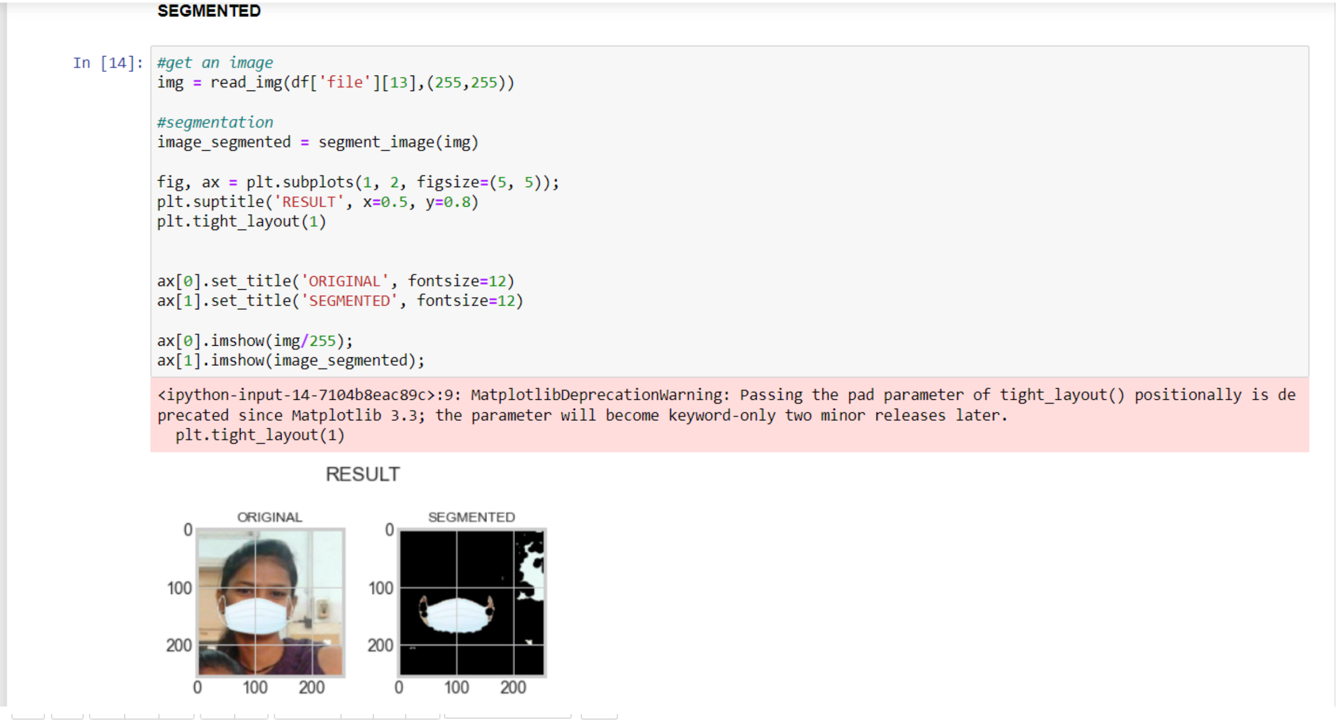
```
In [12]: nb_rows = 3
         nb_cols = 5
         fig, axs = plt.subplots(nb_rows, nb_cols, figsize=(10, 5));
         plt.suptitle('SAMPLE IMAGES');
         for i in range(0, nb_rows):
             for j in range(0, nb_cols):
                 axs[i, j].xaxis.set_ticklabels([]);
                 axs[i, j].yaxis.set_ticklabels([]);
                 axs[i, j].imshow((read_img(df['file'][np.random.randint(400)], (255,255)))/255.);
         plt.show();
```



MASK

```
In [13]: #get an image
         img = read_img(df['file'][12],(255,255))

         #mask
         image_mask = create_mask_for_image(img)

         fig, ax = plt.subplots(1, 2, figsize=(5, 5));
         plt.suptitle('RESULT', x=0.5, y=0.8)
         plt.tight_layout(1)

         ax[0].set_title('ORIGINAL', fontsize=12)
         ax[1].set_title('MASK', fontsize=12)

         ax[0].imshow(img/255);
         ax[1].imshow(image_mask);
```

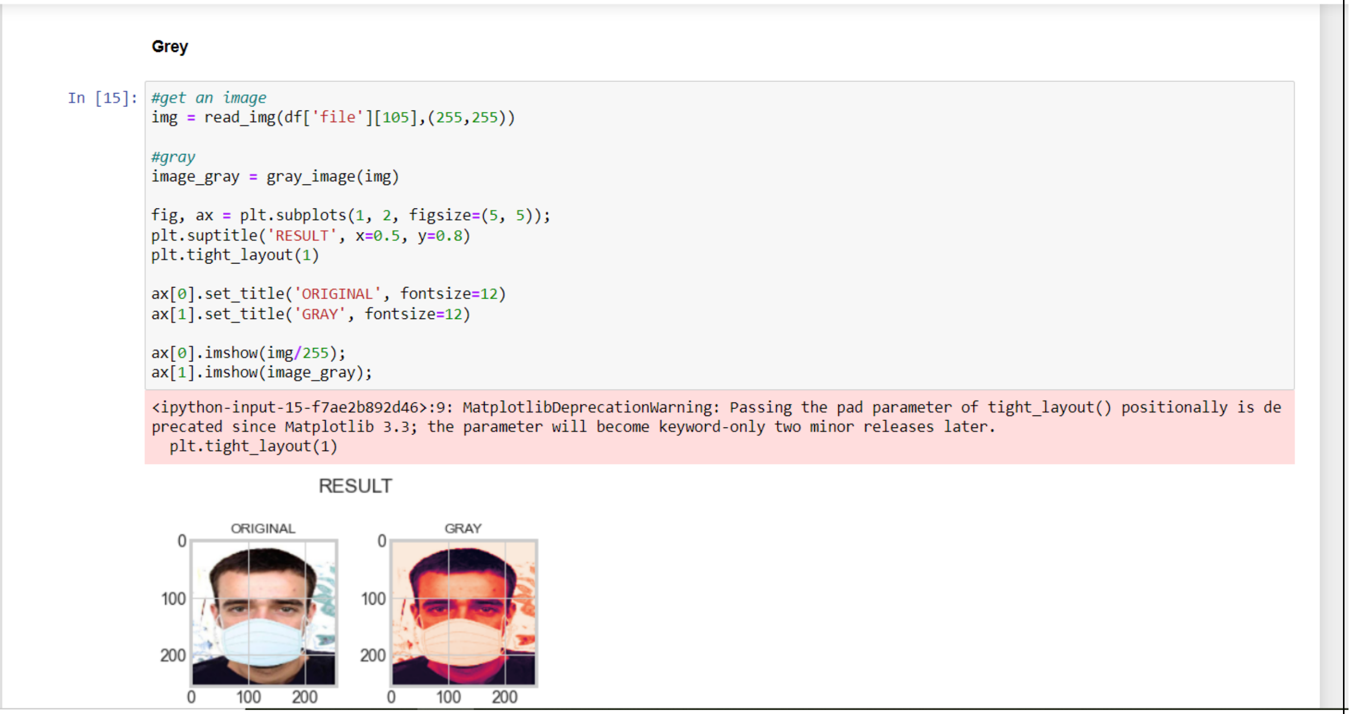
```
<ipython-input-13-ee8d088c596b>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```



**SEGMENTED**

In [14]:
```
#get an image
img = read_img(df['file'][13],(255,255))

#segmentation
image_segmented = segment_image(img)

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)


ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('SEGMENTED', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(image_segmented);
```

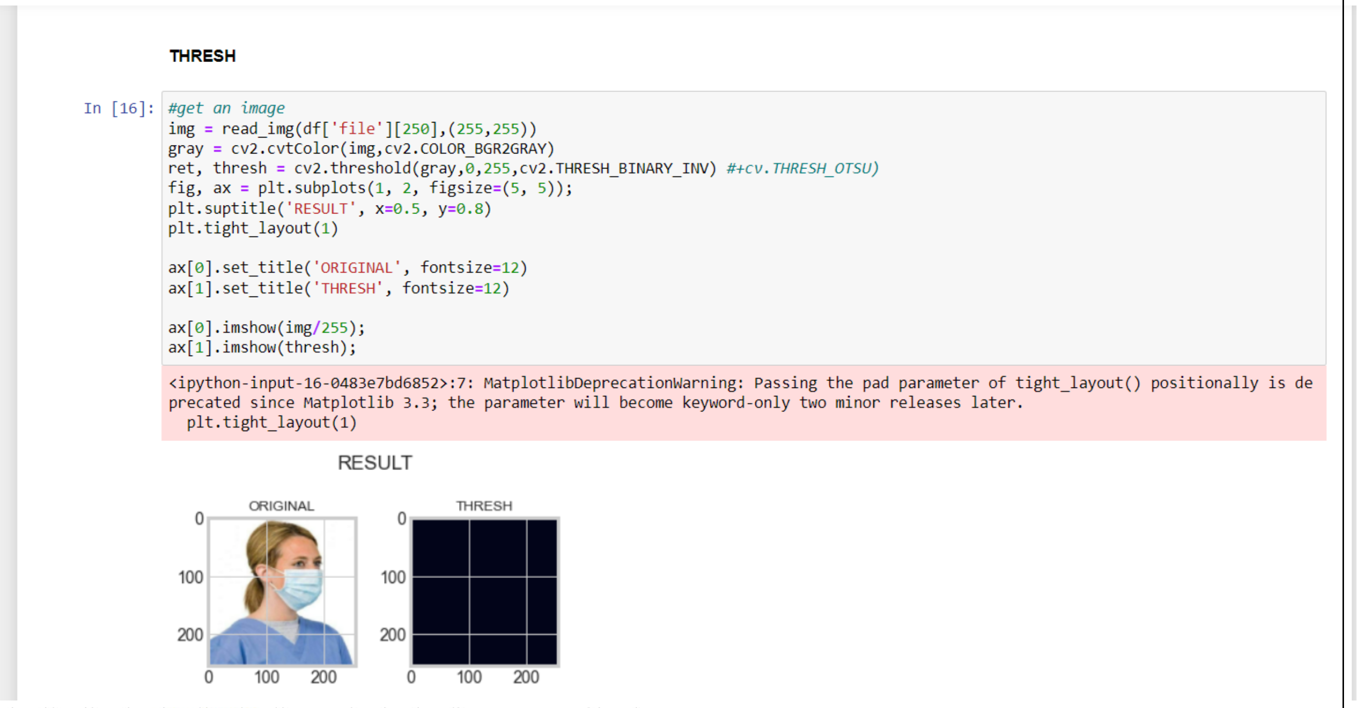
```
<ipython-input-14-7104b8eac89c>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```



**Grey**

In [15]:
```
#get an image
img = read_img(df['file'][105],(255,255))

#gray
image_gray = gray_image(img)

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('GRAY', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(image_gray);
```

```
<ipython-input-15-f7ae2b892d46>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```

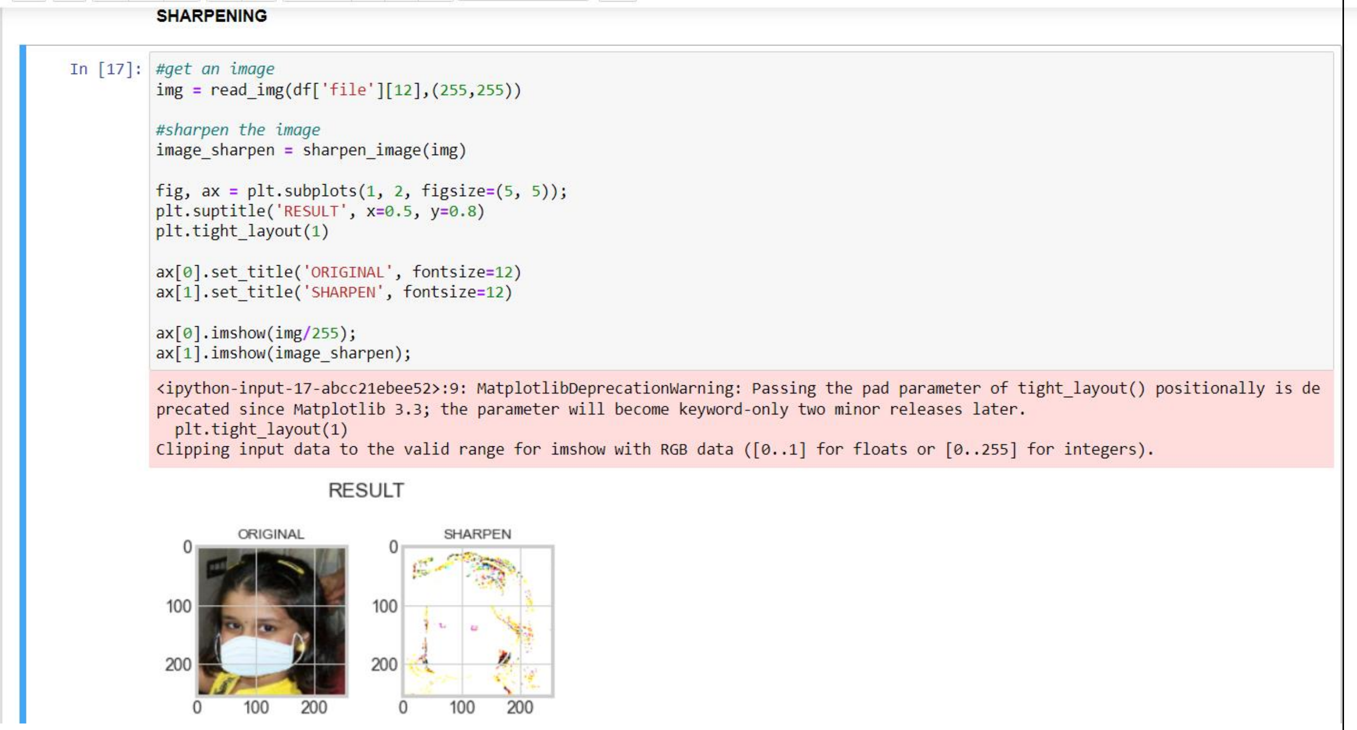


THRESH

In [16]:

```
#get an image
img = read_img(df['file'][250],(255,255))
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV) #+cv.THRESH_OTSU)
fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('THRESH', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(thresh);
```

```
<ipython-input-16-0483e7bd6852>:7: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```



SHARPENING

In [17]:

```
#get an image
img = read_img(df['file'][12],(255,255))

#sharpen the image
image_sharpen = sharpen_image(img)

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('SHARPEN', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(image_sharpen);
```

```
<ipython-input-17-abcc21ebee52>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

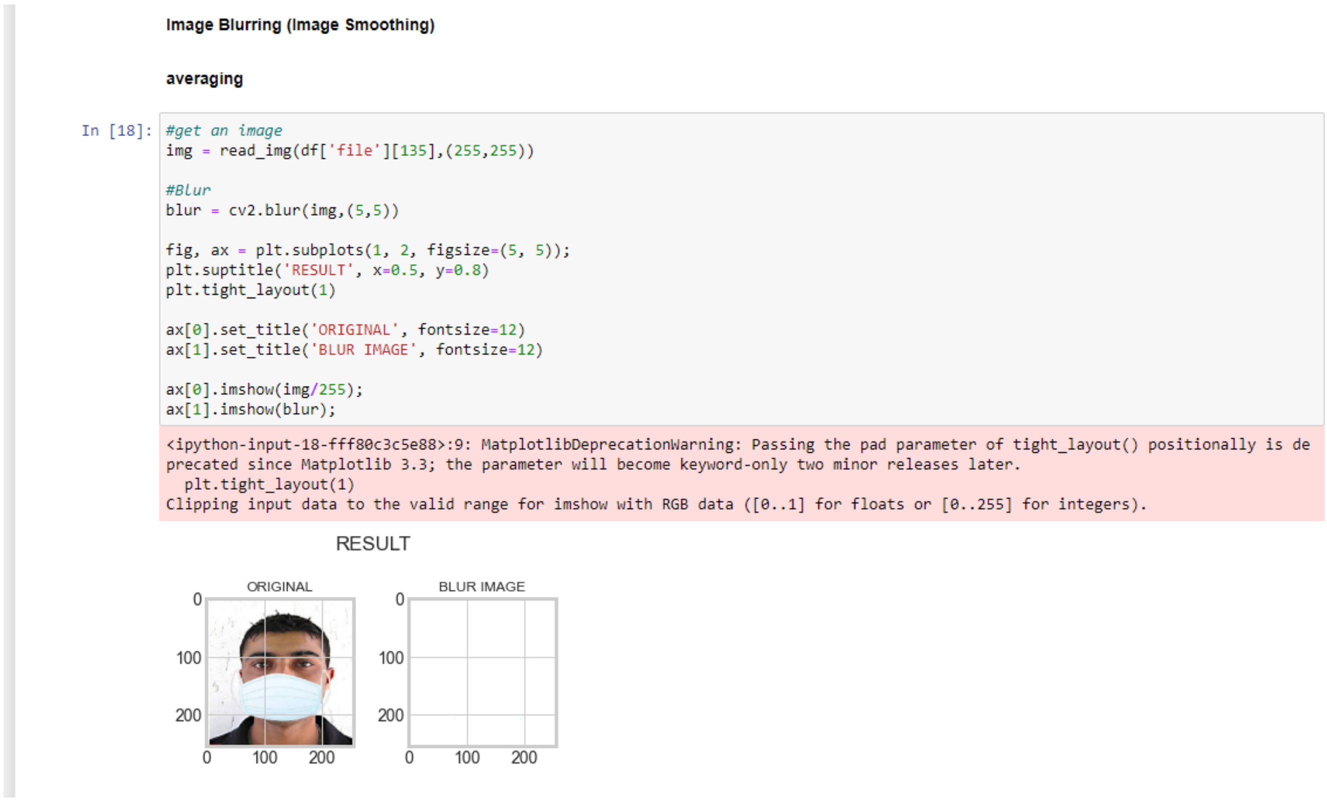


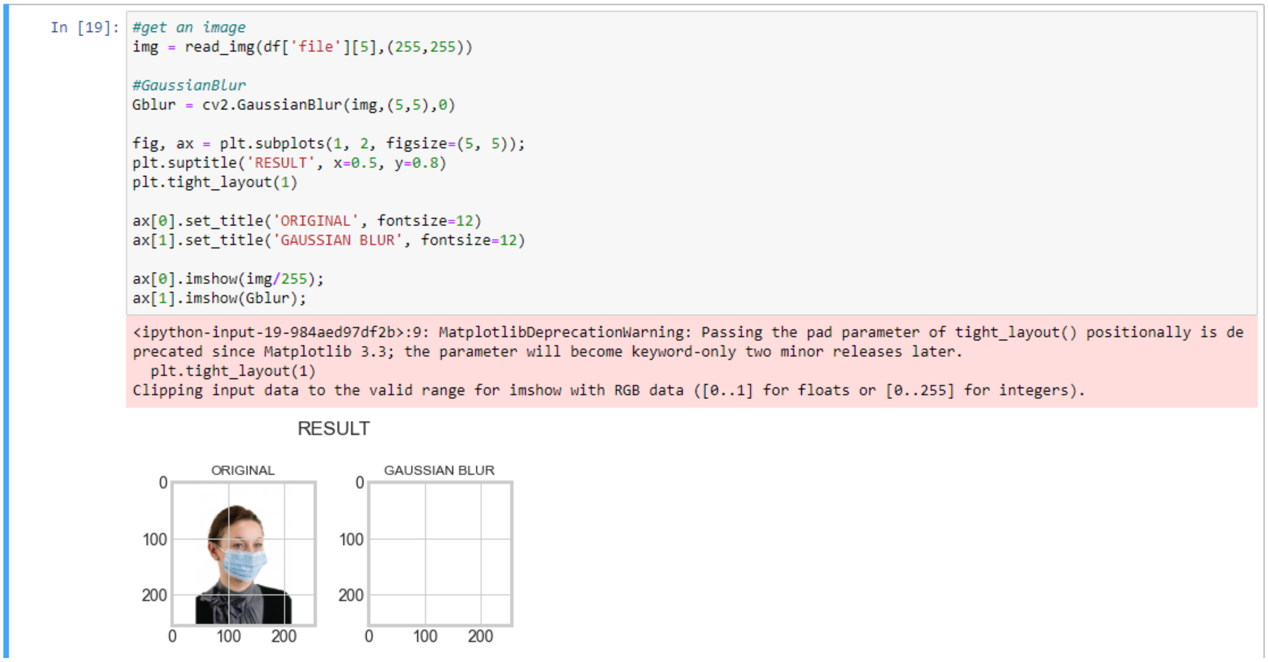
**Image Blurring (Image Smoothing)**

**averaging**

In [18]:
```
#get an image
img = read_img(df['file'][135],(255,255))

#Blur
blur = cv2.blur(img,(5,5))

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('BLUR IMAGE', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(blur);
```

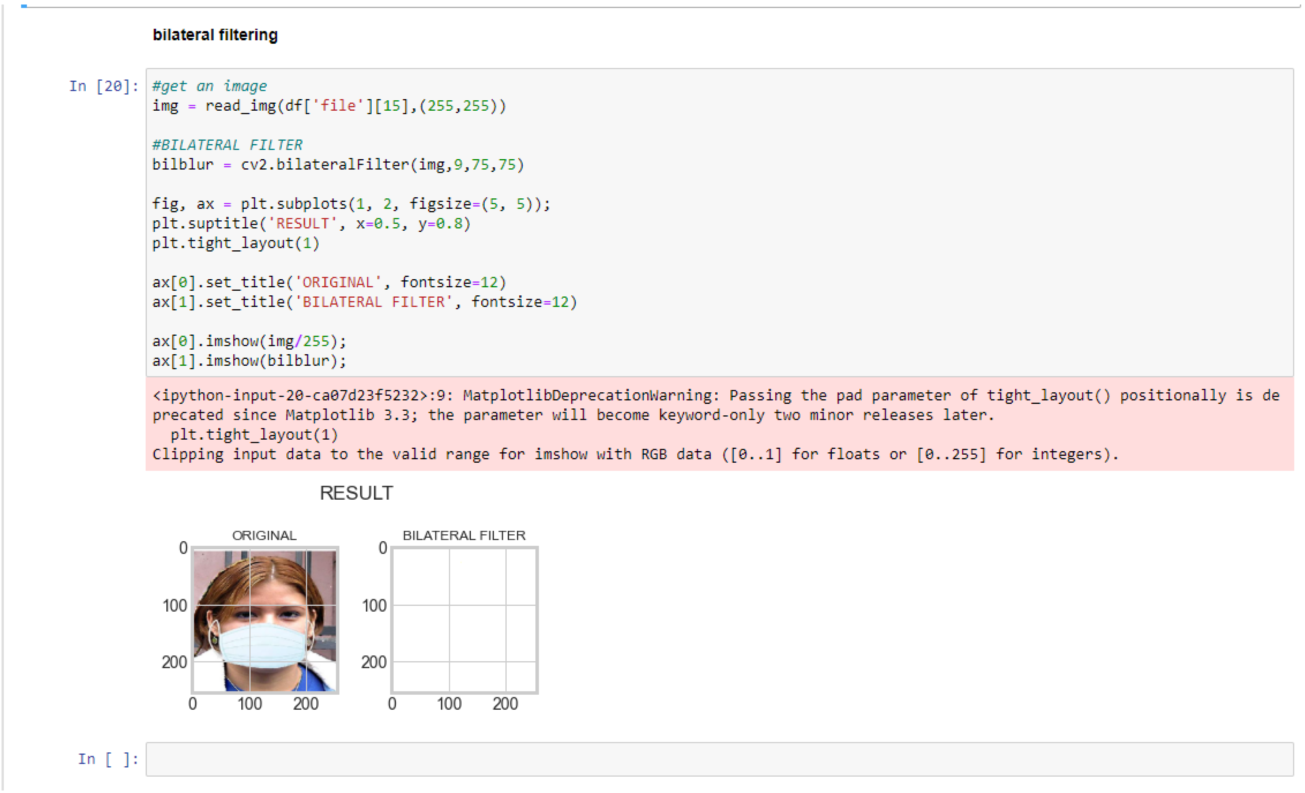
```
<ipython-input-18-fff80c3c5e88>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

**gaussian filter**



In [19]:
```
#get an image
img = read_img(df['file'][5],(255,255))

#GaussianBlur
Gblur = cv2.GaussianBlur(img,(5,5),0)

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('GAUSSIAN BLUR', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(Gblur);
```

```
<ipython-input-19-984aed97df2b>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



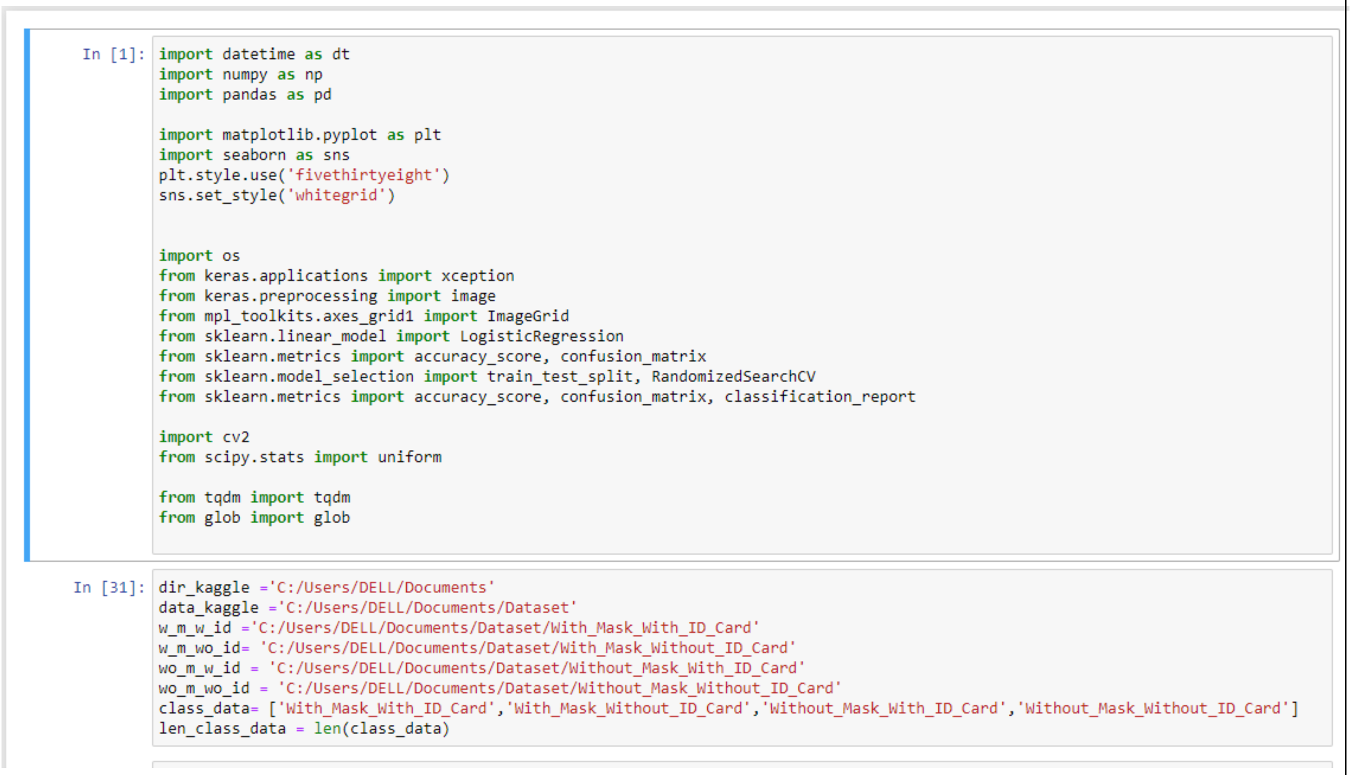
bilateral filtering

```
In [20]: #get an image
         img = read_img(df['file'][15],(255,255))

         #BILATERAL FILTER
         bilblur = cv2.bilateralFilter(img,9,75,75)

         fig, ax = plt.subplots(1, 2, figsize=(5, 5));
         plt.suptitle('RESULT', x=0.5, y=0.8)
         plt.tight_layout(1)

         ax[0].set_title('ORIGINAL', fontsize=12)
         ax[1].set_title('BILATERAL FILTER', fontsize=12)

         ax[0].imshow(img/255);
         ax[1].imshow(bilblur);
```

```
<ipython-input-20-ca07d23f5232>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

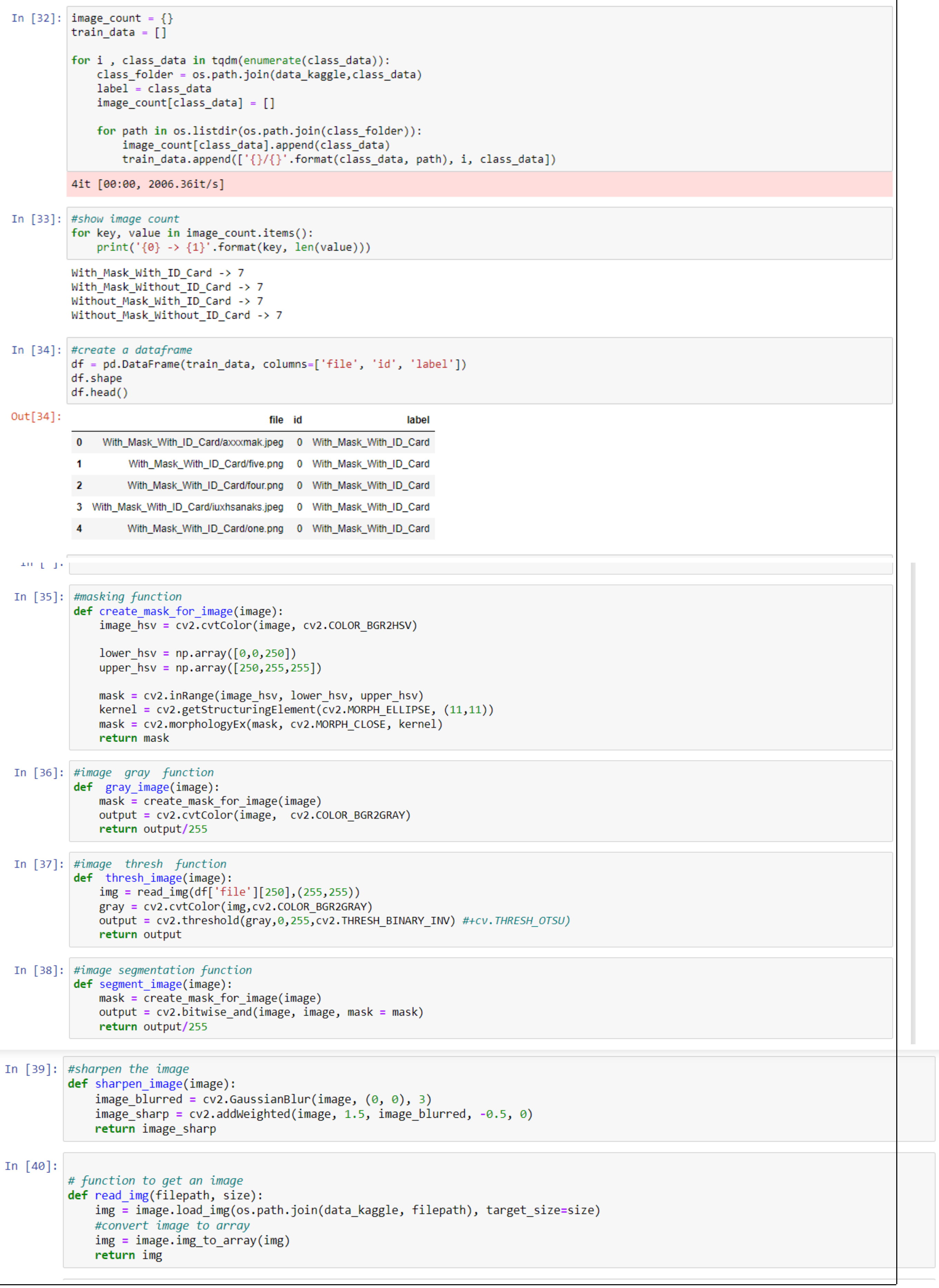
```
In [ ]:
```

# Code and Output for Image processing for all images:



```
In [1]: import datetime as dt
        import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        import seaborn as sns
        plt.style.use('fivethirtyeight')
        sns.set_style('whitegrid')


        import os
        from keras.applications import xception
        from keras.preprocessing import image
        from mpl_toolkits.axes_grid1 import ImageGrid
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score, confusion_matrix
        from sklearn.model_selection import train_test_split, RandomizedSearchCV
        from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

        import cv2
        from scipy.stats import uniform

        from tqdm import tqdm
        from glob import glob
```

```
In [31]: dir_kaggle ='C:/Users/DELL/Documents'
         data_kaggle ='C:/Users/DELL/Documents/Dataset'
         w_m_w_id ='C:/Users/DELL/Documents/Dataset/With_Mask_With_ID_Card'
         w_m_wo_id= 'C:/Users/DELL/Documents/Dataset/With_Mask_Without_ID_Card'
         wo_m_w_id = 'C:/Users/DELL/Documents/Dataset/Without_Mask_With_ID_Card'
         wo_m_wo_id = 'C:/Users/DELL/Documents/Dataset/Without_Mask_Without_ID_Card'
         class_data= ['With_Mask_With_ID_Card','With_Mask_Without_ID_Card','Without_Mask_With_ID_Card','Without_Mask_Without_ID_Card']
         len_class_data = len(class_data)
```



In [32]:
```
image_count = {}
train_data = []

for i , class_data in tqdm(enumerate(class_data)):
    class_folder = os.path.join(data_kaggle,class_data)
    label = class_data
    image_count[class_data] = []

    for path in os.listdir(os.path.join(class_folder)):
        image_count[class_data].append(class_data)
        train_data.append(['{}/{}'.format(class_data, path), i, class_data])
```

```
4it [00:00, 2006.36it/s]
```

In [33]:
```
#show image count
for key, value in image_count.items():
    print('{0} -> {1}'.format(key, len(value)))
```

```
With_Mask_With_ID_Card -> 7
With_Mask_Without_ID_Card -> 7
Without_Mask_With_ID_Card -> 7
Without_Mask_Without_ID_Card -> 7
```

In [34]:
```
#create a dataframe
df = pd.DataFrame(train_data, columns=['file', 'id', 'label'])
df.shape
df.head()
```

Out[34]:

| | file | id | label |
|---|---|---|---|
| 0 | With_Mask_With_ID_Card/axxxmak.jpeg | 0 | With_Mask_With_ID_Card |
| 1 | With_Mask_With_ID_Card/five.png | 0 | With_Mask_With_ID_Card |
| 2 | With_Mask_With_ID_Card/four.png | 0 | With_Mask_With_ID_Card |
| 3 | With_Mask_With_ID_Card/iuxhsanaks.jpeg | 0 | With_Mask_With_ID_Card |
| 4 | With_Mask_With_ID_Card/one.png | 0 | With_Mask_With_ID_Card |

In [ ]:

In [35]:
```
#masking function
def create_mask_for_image(image):
    image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    lower_hsv = np.array([0,0,250])
    upper_hsv = np.array([250,255,255])

    mask = cv2.inRange(image_hsv, lower_hsv, upper_hsv)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11))
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
    return mask
```

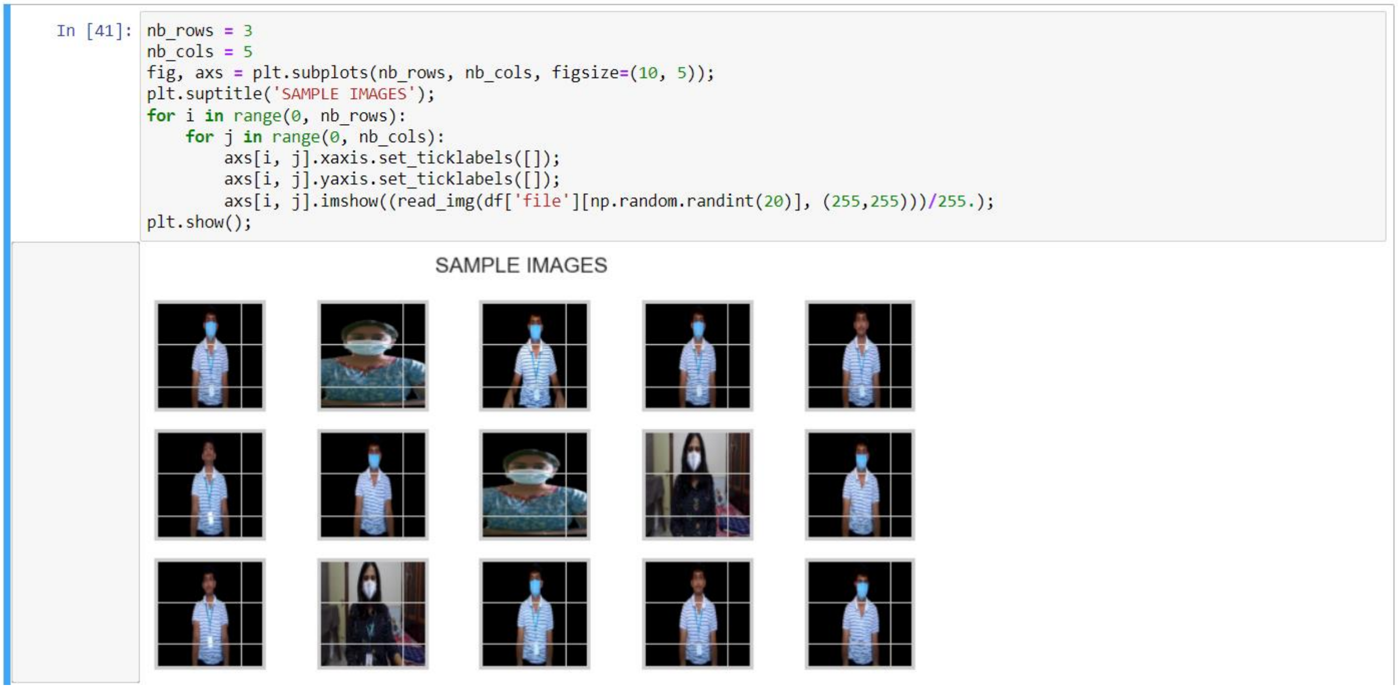
In [36]:
```
#image  gray  function
def  gray_image(image):
    mask = create_mask_for_image(image)
    output = cv2.cvtColor(image,  cv2.COLOR_BGR2GRAY)
    return output/255
```

In [37]:
```
#image  thresh  function
def  thresh_image(image):
    img = read_img(df['file'][250],(255,255))
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    output = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV) #+cv.THRESH_OTSU)
    return output
```

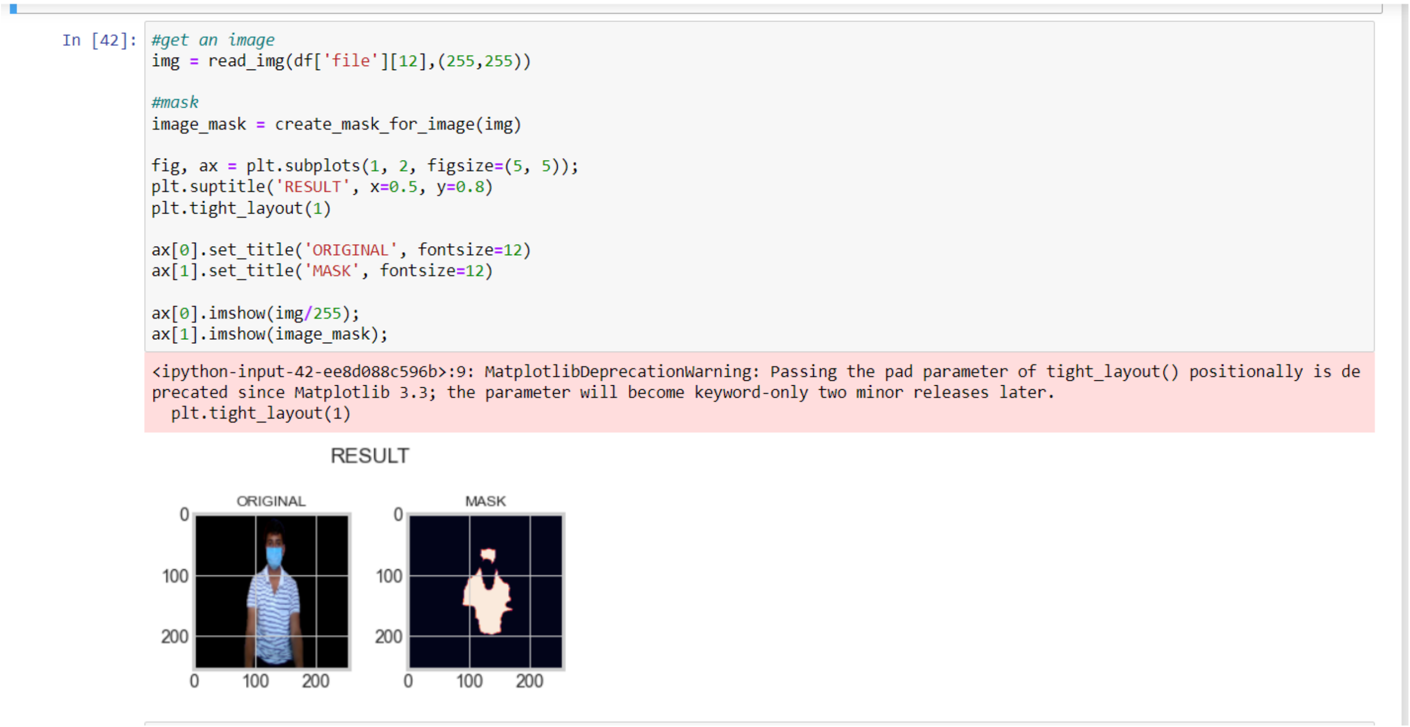
In [38]:
```
#image segmentation function
def segment_image(image):
    mask = create_mask_for_image(image)
    output = cv2.bitwise_and(image, image, mask = mask)
    return output/255
```

In [39]:
```
#sharpen the image
def sharpen_image(image):
    image_blurred = cv2.GaussianBlur(image, (0, 0), 3)
    image_sharp = cv2.addWeighted(image, 1.5, image_blurred, -0.5, 0)
    return image_sharp
```

In [40]:
```

# function to get an image
def read_img(filepath, size):
    img = image.load_img(os.path.join(data_kaggle, filepath), target_size=size)
    #convert image to array
    img = image.img_to_array(img)
    return img
```

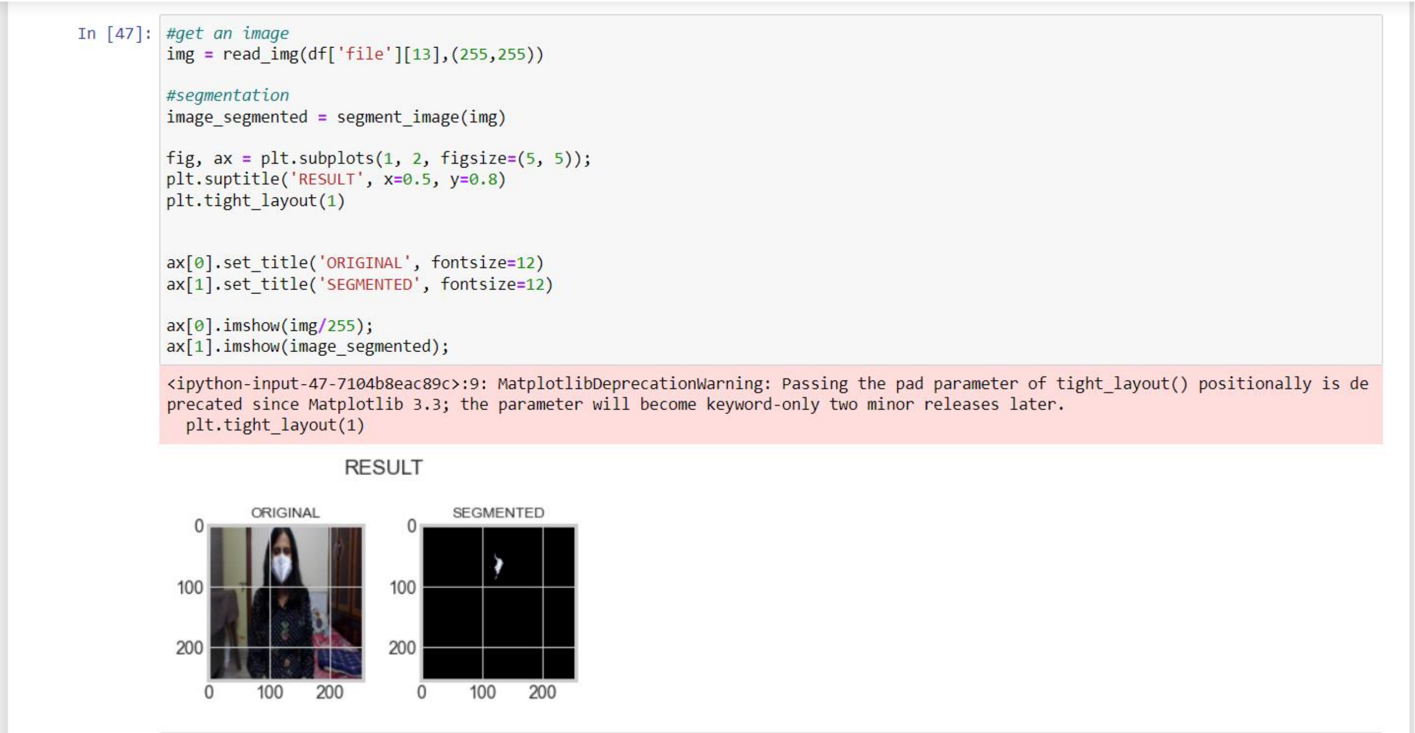


```
In [41]: nb_rows = 3
         nb_cols = 5
         fig, axs = plt.subplots(nb_rows, nb_cols, figsize=(10, 5));
         plt.suptitle('SAMPLE IMAGES');
         for i in range(0, nb_rows):
             for j in range(0, nb_cols):
                 axs[i, j].xaxis.set_ticklabels([]);
                 axs[i, j].yaxis.set_ticklabels([]);
                 axs[i, j].imshow((read_img(df['file'][np.random.randint(20)], (255,255)))/255.);
         plt.show();
```



```
In [42]: #get an image
         img = read_img(df['file'][12],(255,255))

         #mask
         image_mask = create_mask_for_image(img)

         fig, ax = plt.subplots(1, 2, figsize=(5, 5));
         plt.suptitle('RESULT', x=0.5, y=0.8)
         plt.tight_layout(1)

         ax[0].set_title('ORIGINAL', fontsize=12)
         ax[1].set_title('MASK', fontsize=12)

         ax[0].imshow(img/255);
         ax[1].imshow(image_mask);
```

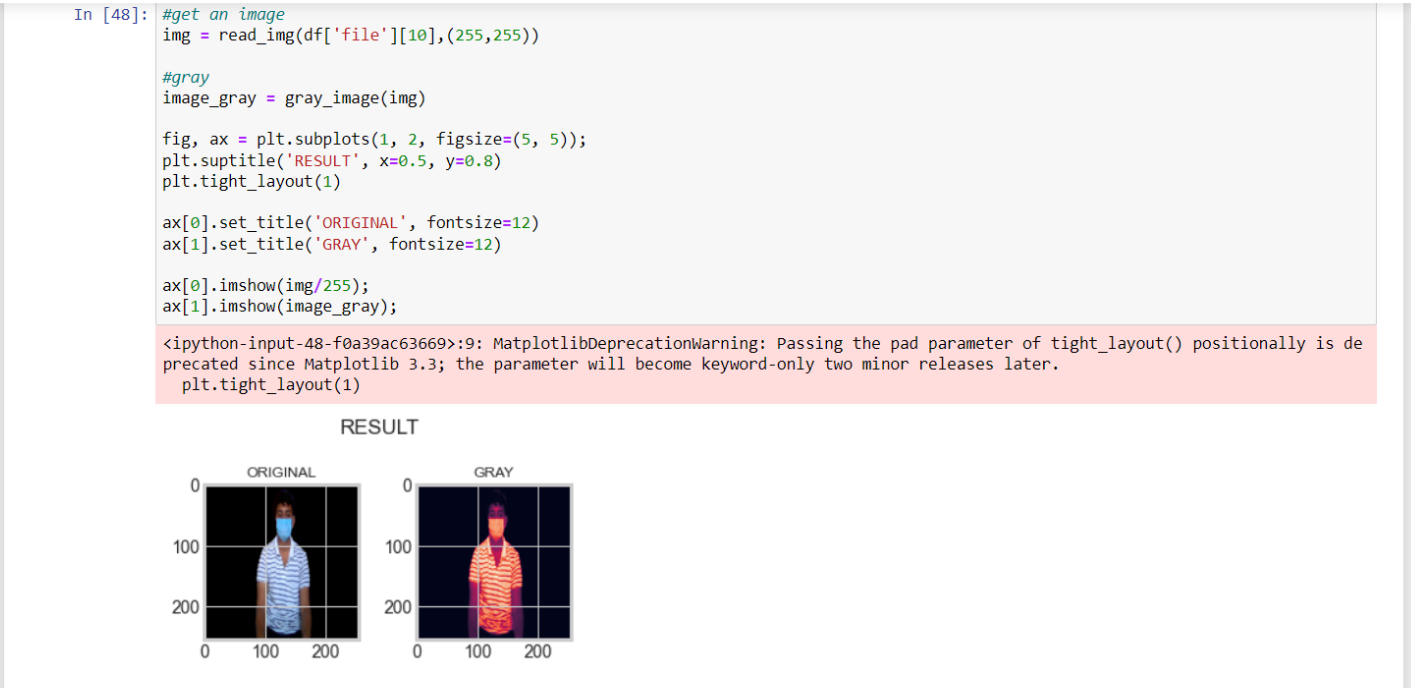
```
<ipython-input-42-ee8d088c596b>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```



In [47]:

```
#get an image
img = read_img(df['file'][13],(255,255))

#segmentation
image_segmented = segment_image(img)

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)


ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('SEGMENTED', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(image_segmented);
```

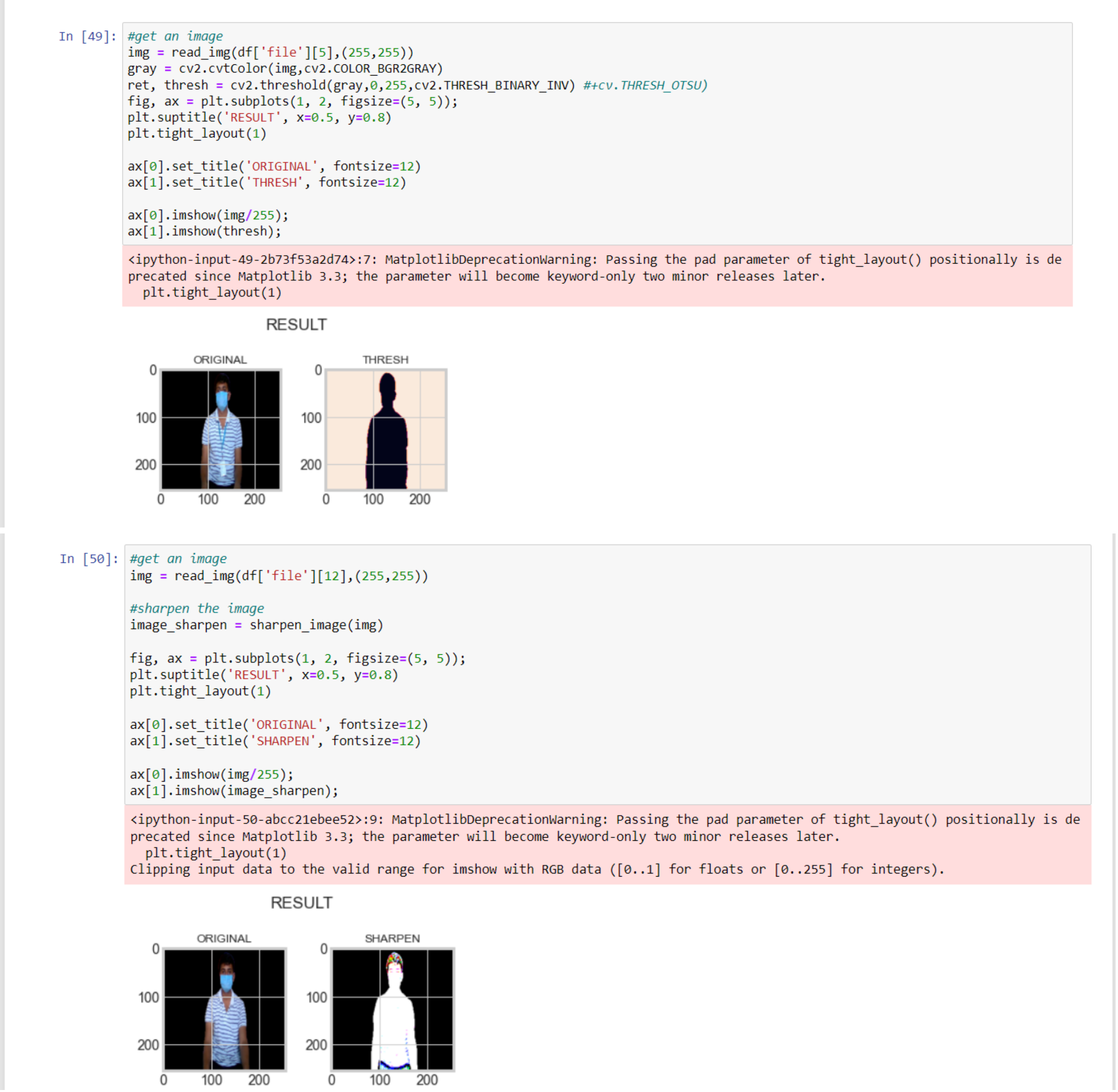
```
<ipython-input-47-7104b8eac89c>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```



In [48]:

```
#get an image
img = read_img(df['file'][10],(255,255))

#gray
image_gray = gray_image(img)

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('GRAY', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(image_gray);
```

```
<ipython-input-48-f0a39ac63669>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```

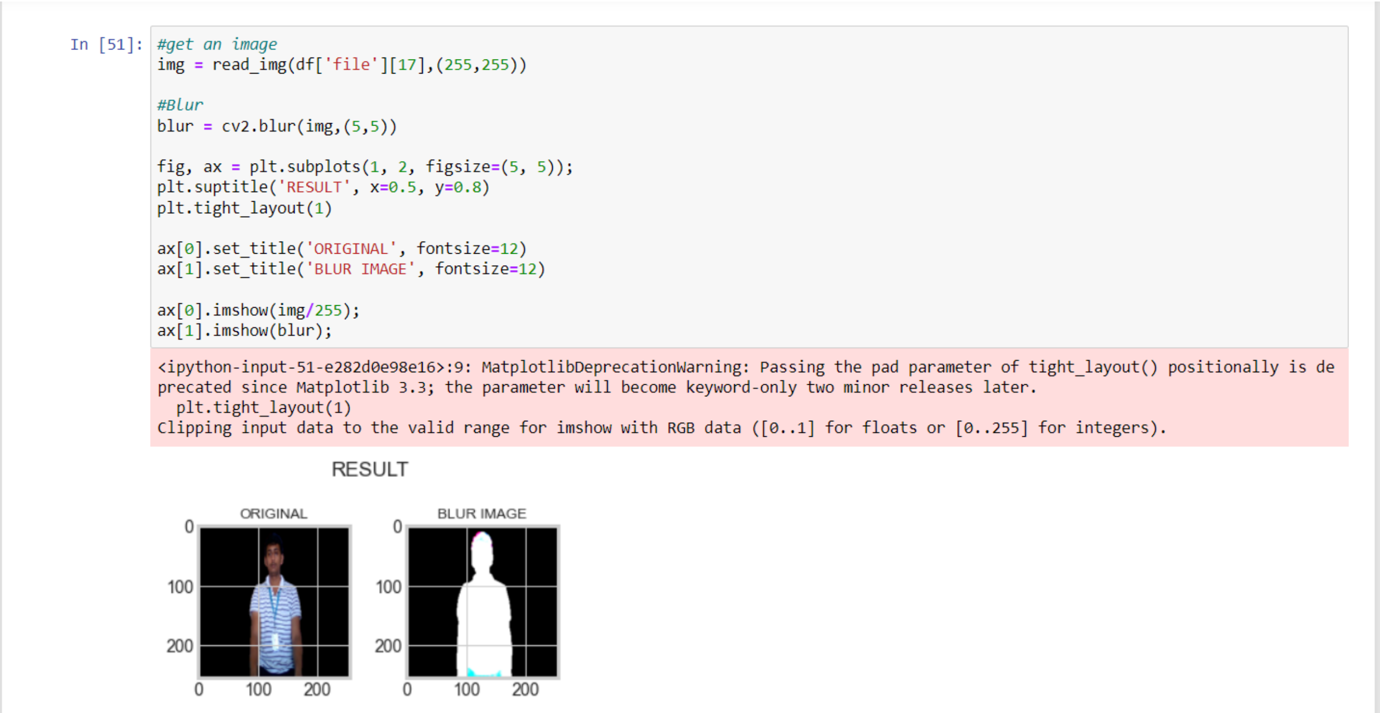


```
In [49]: #get an image
         img = read_img(df['file'][5],(255,255))
         gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
         ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV) #+cv.THRESH_OTSU)
         fig, ax = plt.subplots(1, 2, figsize=(5, 5));
         plt.suptitle('RESULT', x=0.5, y=0.8)
         plt.tight_layout(1)

         ax[0].set_title('ORIGINAL', fontsize=12)
         ax[1].set_title('THRESH', fontsize=12)

         ax[0].imshow(img/255);
         ax[1].imshow(thresh);
```

```
<ipython-input-49-2b73f53a2d74>:7: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
```

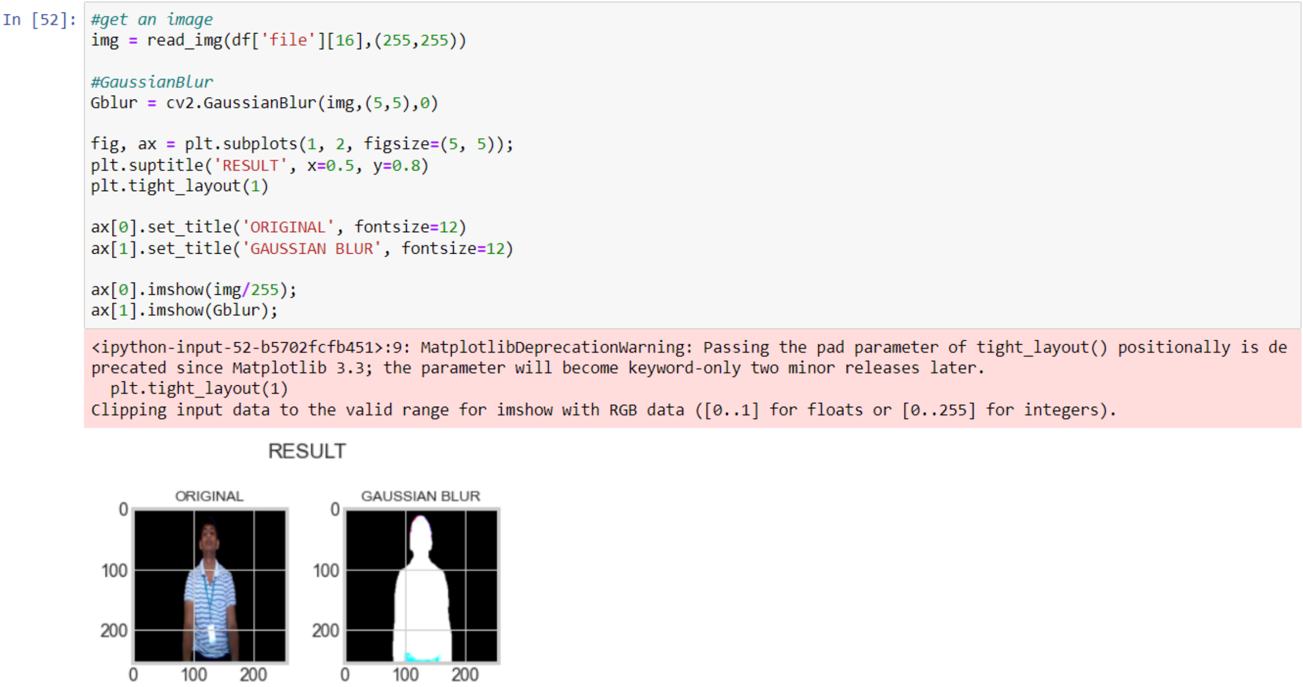
```
In [50]: #get an image
         img = read_img(df['file'][12],(255,255))

         #sharpen the image
         image_sharpen = sharpen_image(img)

         fig, ax = plt.subplots(1, 2, figsize=(5, 5));
         plt.suptitle('RESULT', x=0.5, y=0.8)
         plt.tight_layout(1)

         ax[0].set_title('ORIGINAL', fontsize=12)
         ax[1].set_title('SHARPEN', fontsize=12)

         ax[0].imshow(img/255);
         ax[1].imshow(image_sharpen);
```

```
<ipython-input-50-abcc21ebee52>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



In [51]:
```
#get an image
img = read_img(df['file'][17],(255,255))

#Blur
blur = cv2.blur(img,(5,5))

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('BLUR IMAGE', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(blur);
```

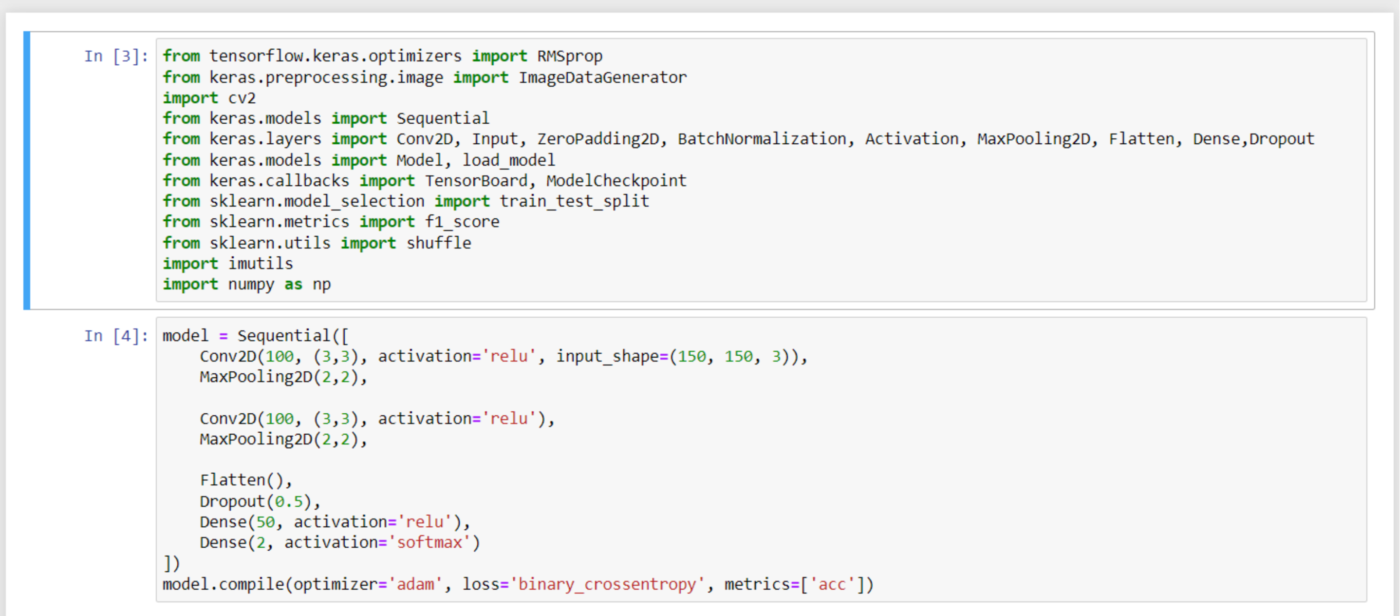
```
<ipython-input-51-e282d0e98e16>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



In [52]:
```
#get an image
img = read_img(df['file'][16],(255,255))

#GaussianBlur
Gblur = cv2.GaussianBlur(img,(5,5),0)

fig, ax = plt.subplots(1, 2, figsize=(5, 5));
plt.suptitle('RESULT', x=0.5, y=0.8)
plt.tight_layout(1)

ax[0].set_title('ORIGINAL', fontsize=12)
ax[1].set_title('GAUSSIAN BLUR', fontsize=12)

ax[0].imshow(img/255);
ax[1].imshow(Gblur);
```

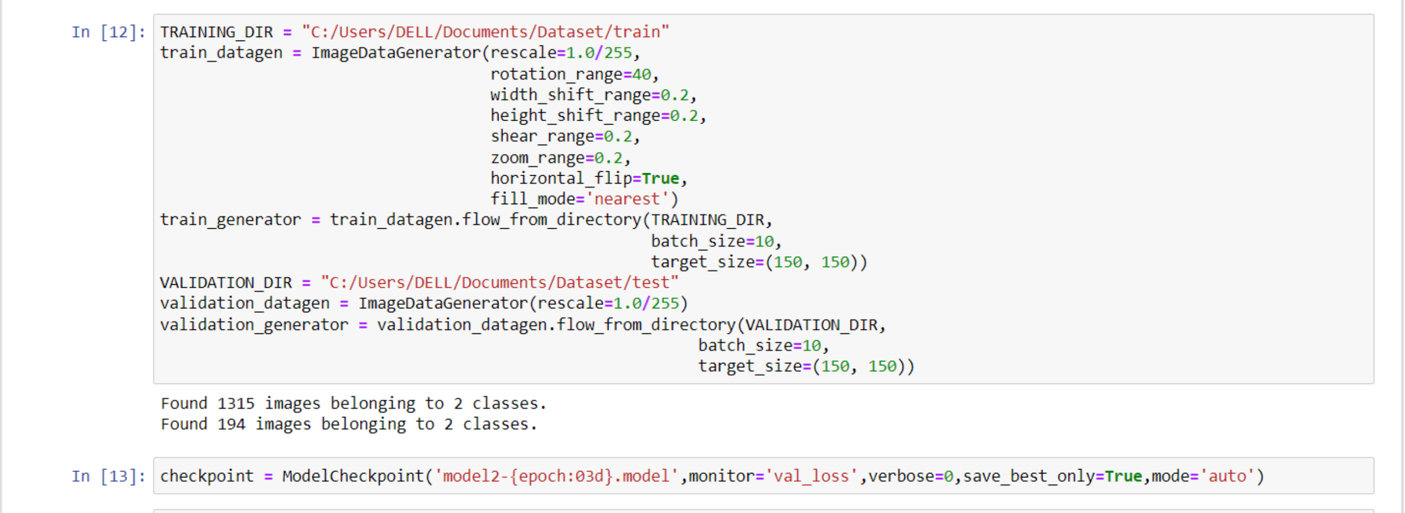
```
<ipython-input-52-b5702fcfb451>:9: MatplotlibDeprecationWarning: Passing the pad parameter of tight_layout() positionally is de
precated since Matplotlib 3.3; the parameter will become keyword-only two minor releases later.
  plt.tight_layout(1)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

# Training Model:



```
In [3]:
from tensorflow.keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout
from keras.models import Model, load_model
from keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import numpy as np
```

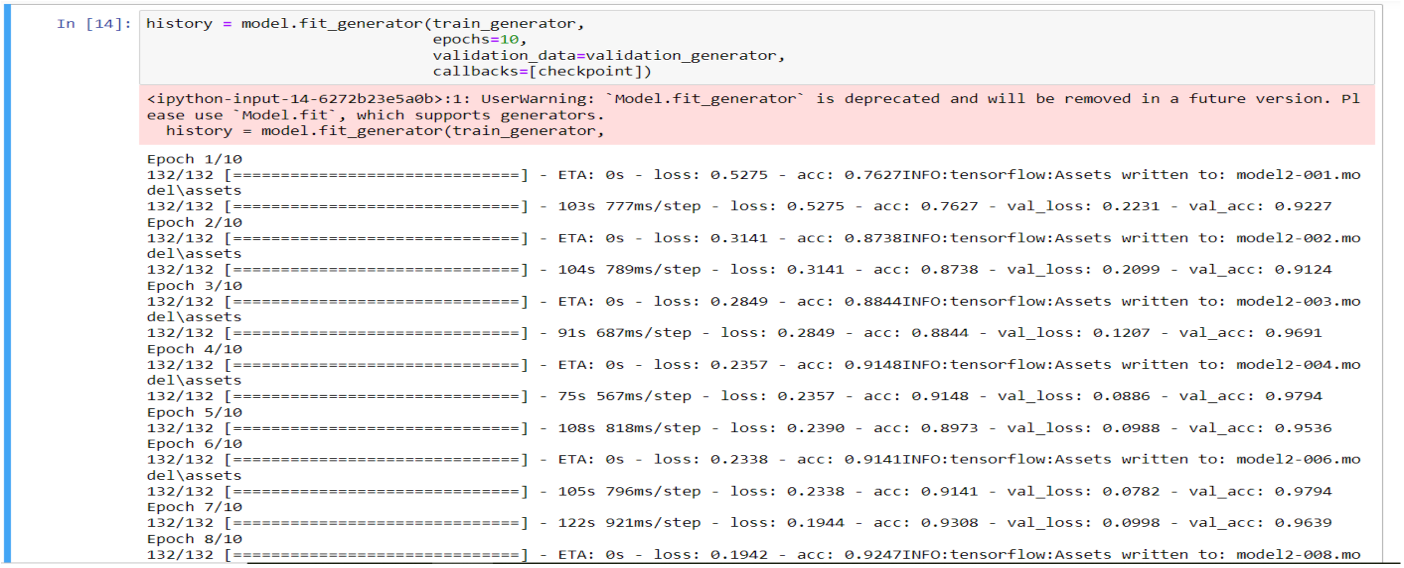
```
In [4]:
model = Sequential([
    Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2,2),

    Conv2D(100, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dropout(0.5),
    Dense(50, activation='relu'),
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```



```
In [12]:
TRAINING_DIR = "C:/Users/DELL/Documents/Dataset/train"
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=10,
                                                    target_size=(150, 150))
VALIDATION_DIR = "C:/Users/DELL/Documents/Dataset/test"
validation_datagen = ImageDataGenerator(rescale=1.0/255)
validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                         batch_size=10,
                                                         target_size=(150, 150))
```

```
Found 1315 images belonging to 2 classes.
Found 194 images belonging to 2 classes.
```

```
In [13]:
checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
```



```
In [14]:
history = model.fit_generator(train_generator,
                              epochs=10,
                              validation_data=validation_generator,
                              callbacks=[checkpoint])
```

```
<ipython-input-14-6272b23e5a0b>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Pl
ease use `Model.fit`, which supports generators.
  history = model.fit_generator(train_generator,
```

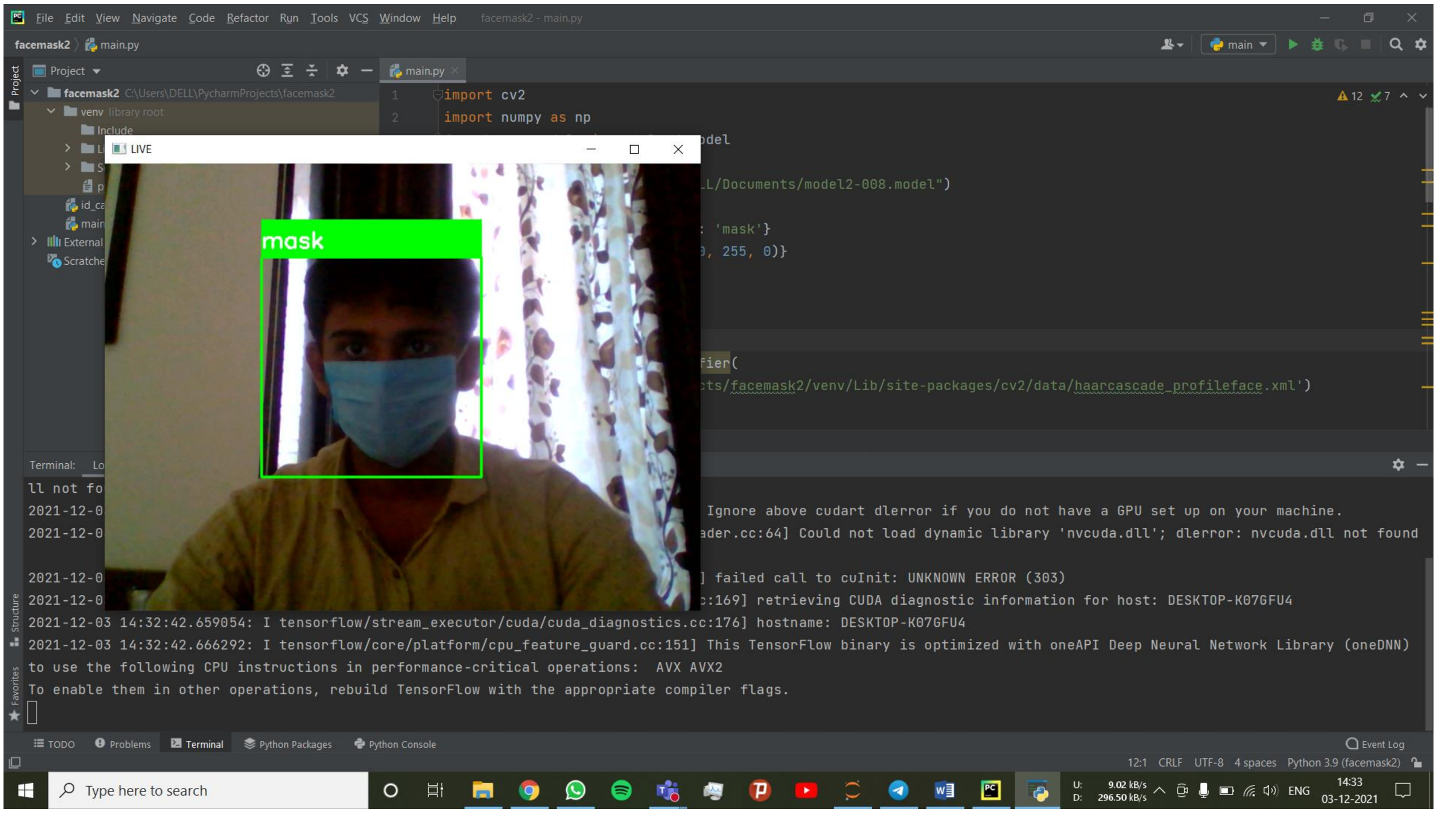
```
Epoch 1/10
132/132 [==============================] - ETA: 0s - loss: 0.5275 - acc: 0.7627INFO:tensorflow:Assets written to: model2-001.mo
del\assets
132/132 [==============================] - 103s 777ms/step - loss: 0.5275 - acc: 0.7627 - val_loss: 0.2231 - val_acc: 0.9227
Epoch 2/10
132/132 [==============================] - ETA: 0s - loss: 0.3141 - acc: 0.8738INFO:tensorflow:Assets written to: model2-002.mo
del\assets
132/132 [==============================] - 104s 789ms/step - loss: 0.3141 - acc: 0.8738 - val_loss: 0.2099 - val_acc: 0.9124
Epoch 3/10
132/132 [==============================] - ETA: 0s - loss: 0.2849 - acc: 0.8844INFO:tensorflow:Assets written to: model2-003.mo
del\assets
132/132 [==============================] - 91s 687ms/step - loss: 0.2849 - acc: 0.8844 - val_loss: 0.1207 - val_acc: 0.9691
Epoch 4/10
132/132 [==============================] - ETA: 0s - loss: 0.2357 - acc: 0.9148INFO:tensorflow:Assets written to: model2-004.mo
del\assets
132/132 [==============================] - 75s 567ms/step - loss: 0.2357 - acc: 0.9148 - val_loss: 0.0886 - val_acc: 0.9794
Epoch 5/10
132/132 [==============================] - 108s 818ms/step - loss: 0.2390 - acc: 0.8973 - val_loss: 0.0988 - val_acc: 0.9536
Epoch 6/10
132/132 [==============================] - ETA: 0s - loss: 0.2338 - acc: 0.9141INFO:tensorflow:Assets written to: model2-006.mo
del\assets
132/132 [==============================] - 105s 796ms/step - loss: 0.2338 - acc: 0.9141 - val_loss: 0.0782 - val_acc: 0.9794
Epoch 7/10
132/132 [==============================] - 122s 921ms/step - loss: 0.1944 - acc: 0.9308 - val_loss: 0.0998 - val_acc: 0.9639
Epoch 8/10
132/132 [==============================] - ETA: 0s - loss: 0.1942 - acc: 0.9247INFO:tensorflow:Assets written to: model2-008.mo
```

## Testing Model

```
import cv2
import numpy as np
from keras.models import load_model

model = load_model("C:/Users/DELL/Documents/model2-008.model")

results = {0: 'without mask', 1: 'mask'}
GR_dict = {0: (0, 0, 255), 1: (0, 255, 0)}

rect_size = 4
cap = cv2.VideoCapture(0)

haarcascade = cv2.CascadeClassifier(
    'C:/Users/DELL/PycharmProjects/facemask2/venv/Lib/site-
packages/cv2/data/haarcascade_profileface.xml')

while True:
    (rval, im) = cap.read()
    im = cv2.flip(im, 1, 1)

    rerect_size = cv2.resize(im, (im.shape[1] // rect_size, im.shape[0] //
rect_size))
    faces = haarcascade.detectMultiScale(rerect_size)
    for f in faces:
        (x, y, w, h) = [v * rect_size for v in f]

        face_img = im[y:y + h, x:x + w]
        rerect_sized = cv2.resize(face_img, (150, 150))
        normalized = rerect_sized / 255.0
        reshaped = np.reshape(normalized, (1, 150, 150, 3))
        reshaped = np.vstack([reshaped])
        result = model.predict(reshaped)

        label = np.argmax(result, axis=1)[0]

        cv2.rectangle(im, (x, y), (x + w, y + h), GR_dict[label], 2)
        cv2.rectangle(im, (x, y - 40), (x + w, y), GR_dict[label], -1)
        cv2.putText(im, results[label], (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(255, 255, 255), 2)

    cv2.imshow('LIVE', im)
    key = cv2.waitKey(10)

    if key == 27:
        break

cap.release()

cv2.destroyAllWindows()
```

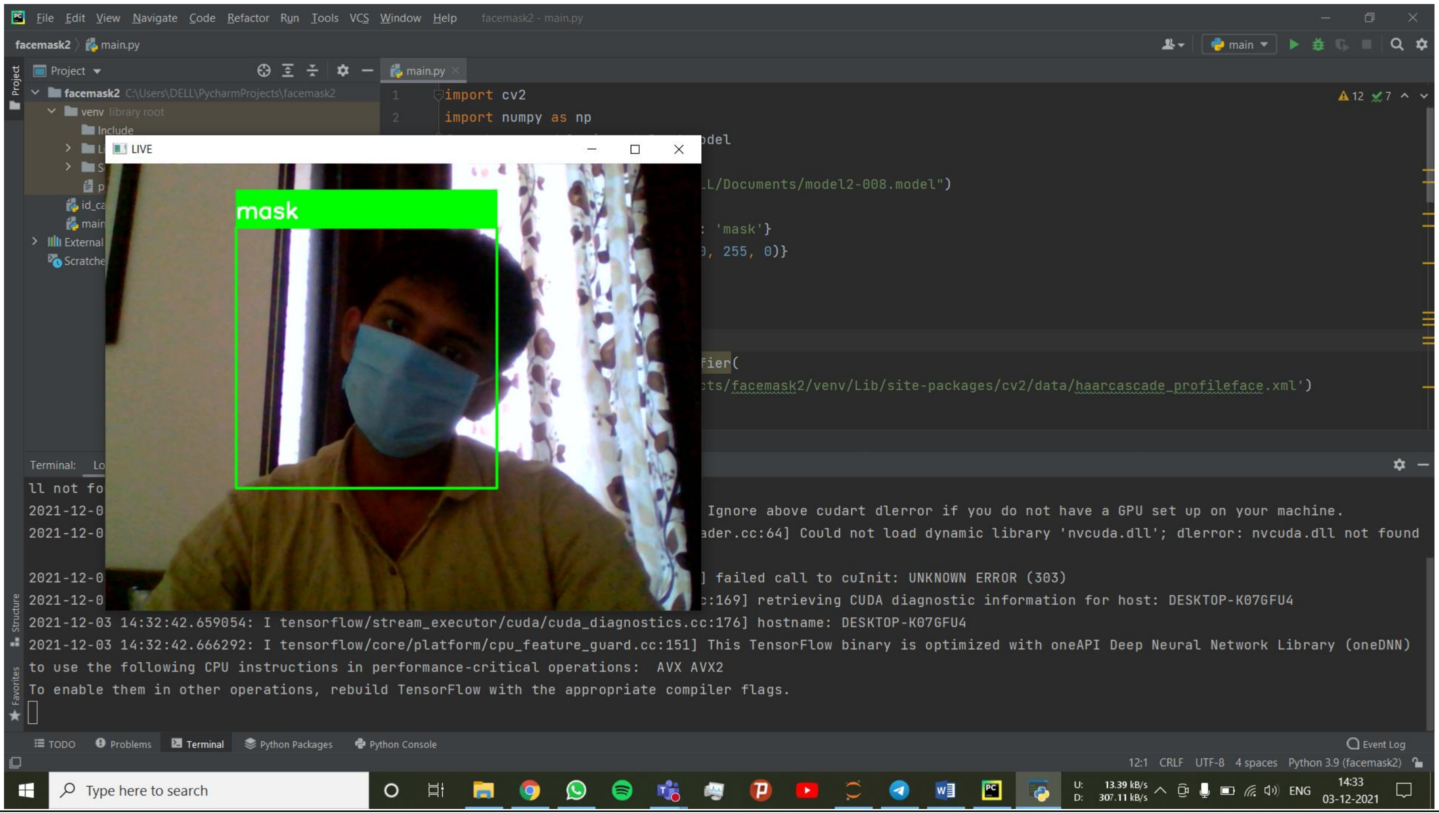
# Results / Output:

## With Mask:



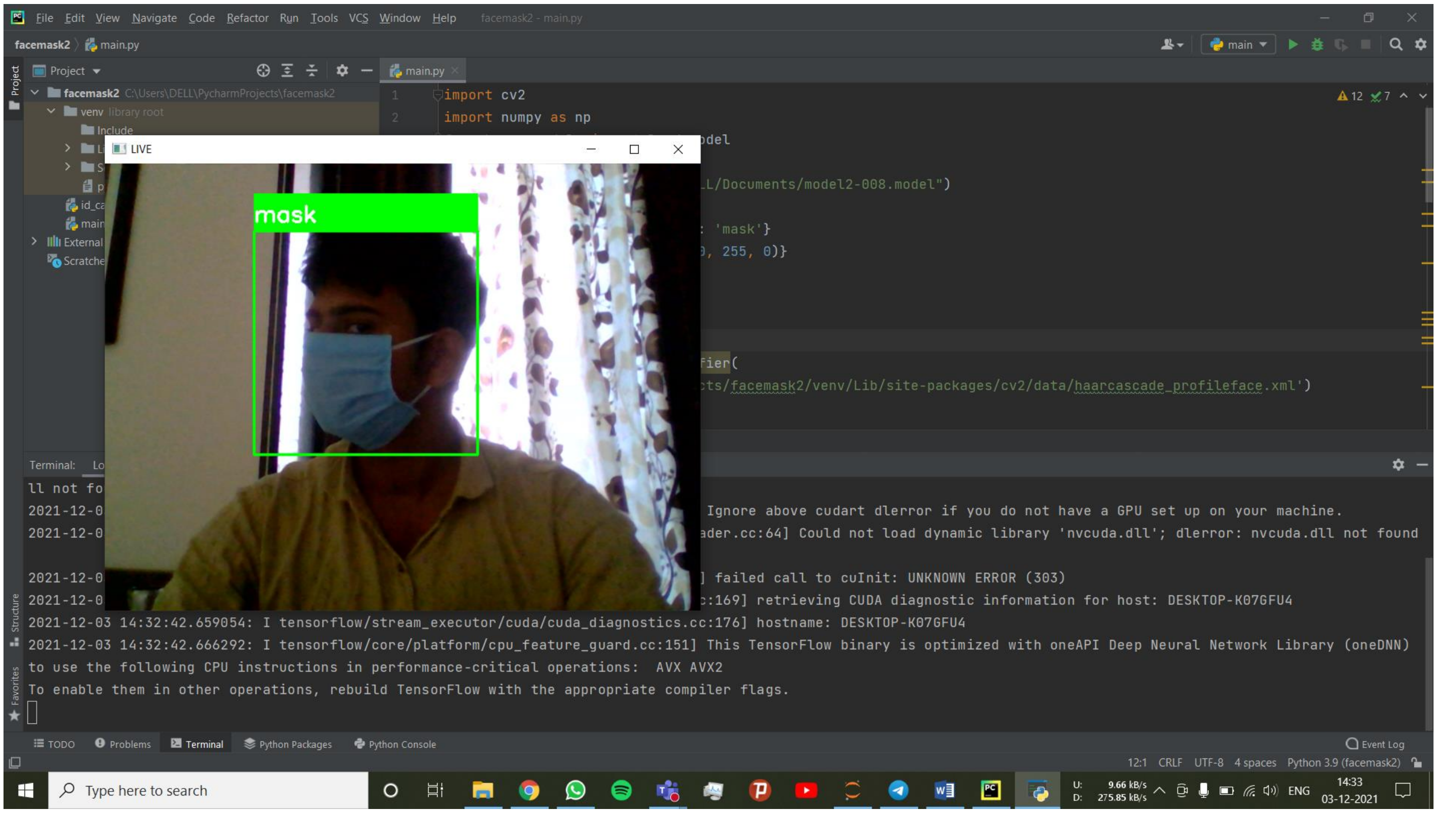
```
import cv2
import numpy as np
odel

L/Documents/model2-008.model")

: 'mask'}
), 255, 0)}




fier(
cts/facemask2/venv/Lib/site-packages/cv2/data/haarcascade_profileface.xml')
```

```
ll not fo
2021-12-0                                         Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2021-12-0                                        ader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2021-12-0                                        ] failed call to cuInit: UNKNOWN ERROR (303)
2021-12-0                                        c:169] retrieving CUDA diagnostic information for host: DESKTOP-K07GFU4
2021-12-03 14:32:42.659054: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-K07GFU4
2021-12-03 14:32:42.666292: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```



```
import cv2
import numpy as np
odel

L/Documents/model2-008.model")

: 'mask'}
), 255, 0)}




fier(
cts/facemask2/venv/Lib/site-packages/cv2/data/haarcascade_profileface.xml')
```

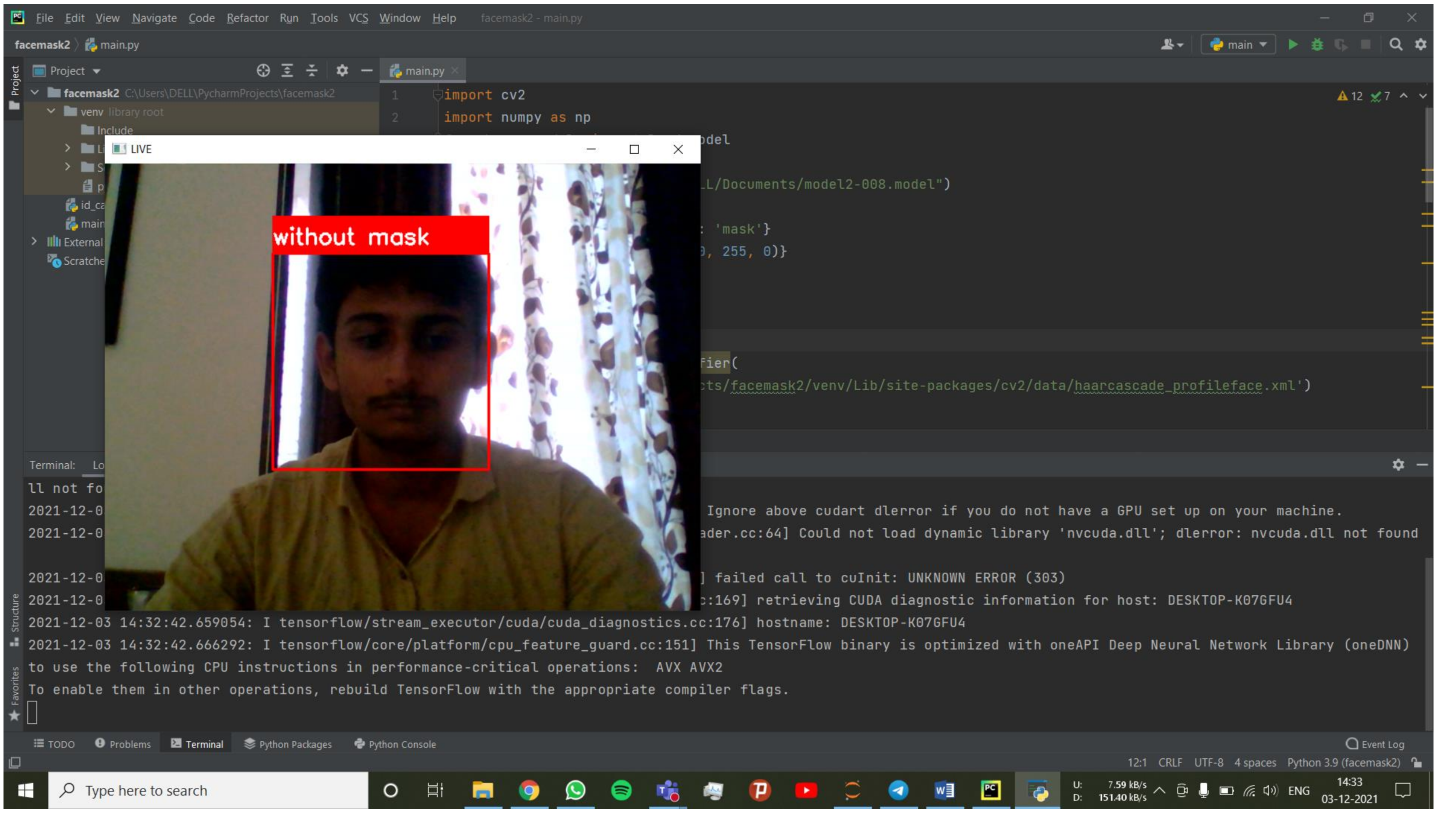
```
ll not fo
2021-12-0                                         Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2021-12-0                                        ader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2021-12-0                                        ] failed call to cuInit: UNKNOWN ERROR (303)
2021-12-0                                        c:169] retrieving CUDA diagnostic information for host: DESKTOP-K07GFU4
2021-12-03 14:32:42.659054: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-K07GFU4
2021-12-03 14:32:42.666292: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```



```
import cv2
import numpy as np
                                  odel

                                  L/Documents/model2-008.model")

                                  : 'mask'}
                                  , 255, 0)}

                                  fier(
                                  cts/facemask2/venv/Lib/site-packages/cv2/data/haarcascade_profileface.xml')
```

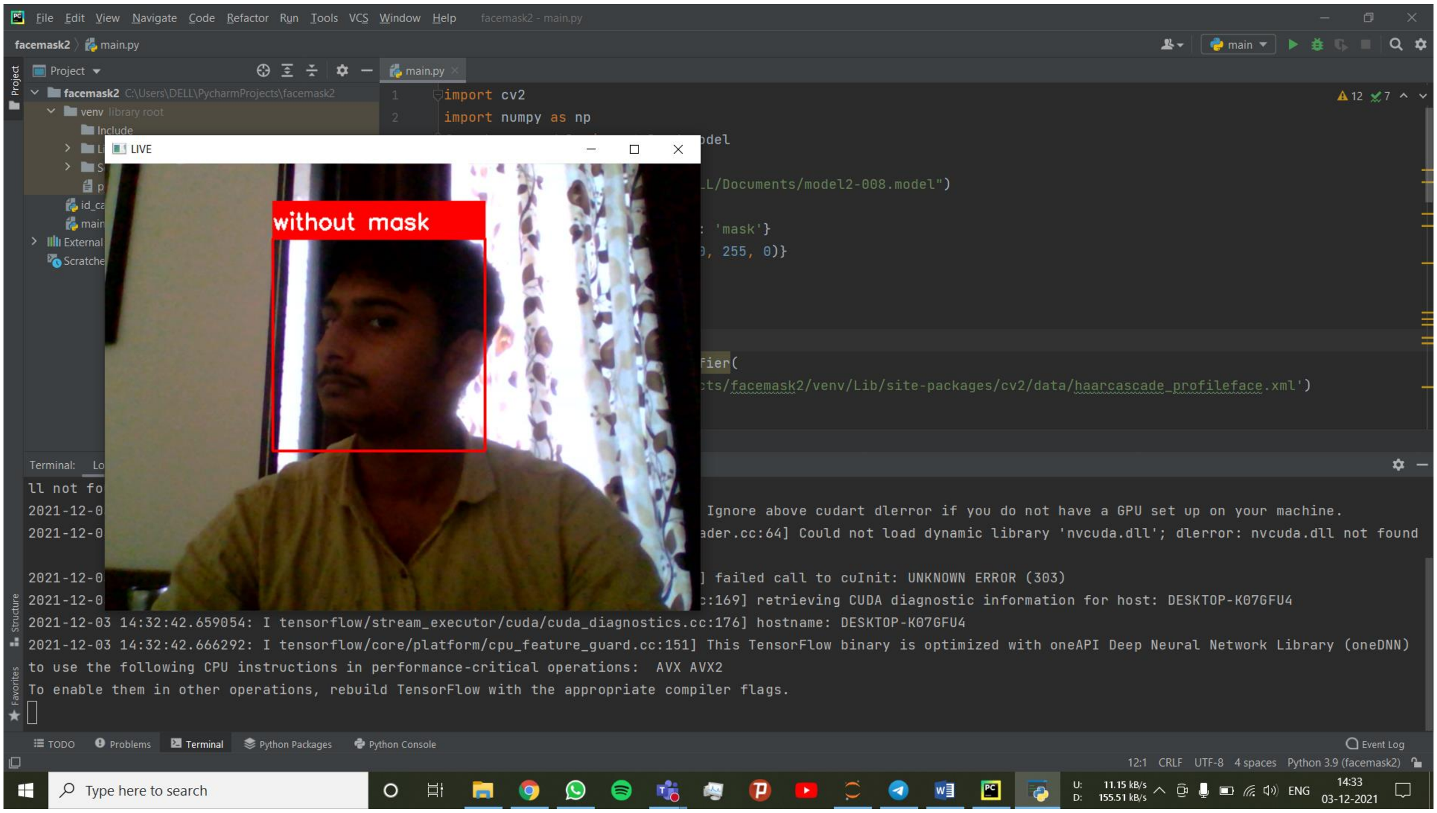
```
ll not fo
2021-12-0                          Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2021-12-0                          ader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2021-12-0                          ] failed call to cuInit: UNKNOWN ERROR (303)
2021-12-0                          :169] retrieving CUDA diagnostic information for host: DESKTOP-K07GFU4
2021-12-03 14:32:42.659054: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-K07GFU4
2021-12-03 14:32:42.666292: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

## Without Mask:



```
import cv2
import numpy as np
                                  odel

                                  L/Documents/model2-008.model")

                                  : 'mask'}
                                  , 255, 0)}

                                  fier(
                                  cts/facemask2/venv/Lib/site-packages/cv2/data/haarcascade_profileface.xml')
```

```
ll not fo
2021-12-0                          Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2021-12-0                          ader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2021-12-0                          ] failed call to cuInit: UNKNOWN ERROR (303)
2021-12-0                          :169] retrieving CUDA diagnostic information for host: DESKTOP-K07GFU4
2021-12-03 14:32:42.659054: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-K07GFU4
2021-12-03 14:32:42.666292: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```



```
import cv2
import numpy as np
odel

L/Documents/model2-008.model")

: 'mask'}
0, 255, 0)}



fier(
cts/facemask2/venv/Lib/site-packages/cv2/data/haarcascade_profileface.xml')
```

```
ll not fo
2021-12-0                                                Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2021-12-0                                               ader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2021-12-0                                               ] failed call to cuInit: UNKNOWN ERROR (303)
2021-12-0                                               c:169] retrieving CUDA diagnostic information for host: DESKTOP-K07GFU4
2021-12-03 14:32:42.659054: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-K07GFU4
2021-12-03 14:32:42.666292: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

**Conclusion**

In this project, we have developed a deep learning model for face mask detection using Python, Keras, and OpenCV. We developed the face mask detector model for detecting whether a person is wearing a mask and ID card or not. We have trained the model using Keras with network architecture. Training the model is the first part of this project and testing using a webcam using OpenCV is the second part.

## References

1) According to Fares Jalled and Ilia Voronkov, Object detection can be done using OpenCV easily. They also said that the face detection program can be implemented to detect and follow people in case of surveillance and other domains. The process of OD is classified into five major categories namely Sliding window-based, contour-based, graph-based, fuzzy-based and context-based. Apart from this, other approaches that are used for detecting objects like the shape-based detection and Steiner tree-based are also summarised.

References :

1. Jalled, F., & Voronkov, I. (2016). Object Detection using Image Processing. ArXiv, abs/1611.07791.

2). According to the Rupesh Kumar Rout(NIT R), training images of the mask gives an accuracy of 92 percent on the training set and 78 percent on the test set. which clearly resembles the model is overfitted.

The data set images should be of good shape and size otherwise it will corrupt the information of boundary. leads to inaccuracy for machine learning model

References :

Hemalatha, C., Muruganand, S., & Maheswaran, R. (2014). A Survey on Real-Time Object Detection, Tracking, and Recognition in Image Processing. International Journal of Computer Applications, 91, 38-42.