REAL TIME FACE MASK DETECTION USING PYTHON | OpenCV|MACHINE LEARNING

PRESENTED BY:-

Poonam Dubey
(222IS019)

ABSTRACT:-

In covid time,it was mandatory for all the citizens to wear a face mask to protect themselves from COVID-19. This application can be helpful for all the hospitals,shop owners, offices, banks or any public place because if anyone is not wearing a mask then he or she must not be allowed in that area. So, to take care of this problem we don't need any guard or person who keeps a watch on people. We can integrate a camera which continuously clicks pictures of humans and detect from there faces whether they are wearing a face mask or not.

NO MASK
MASK DETECTED

INTRODUCTION:

STEPS:-

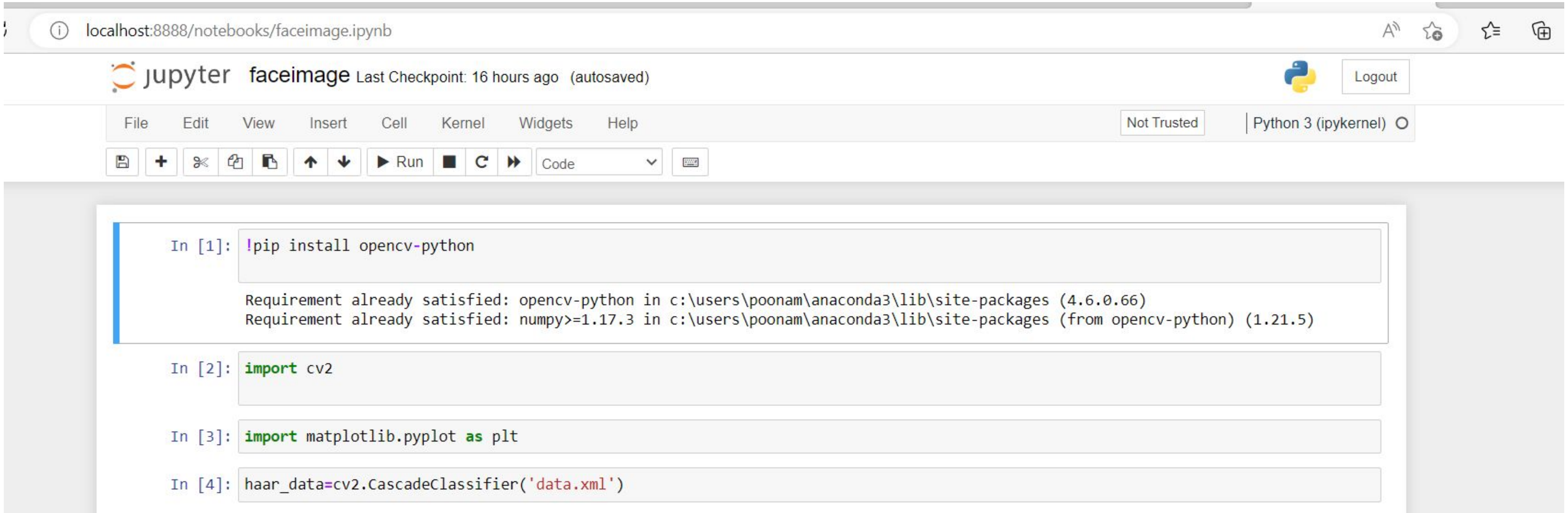
1:-COLLECT FACE DATA WITH AND WITHOUT MASK

2:-TRAIN DATA USING MACHINE LEARNING

3:-DO PREDICTION ON LIVE DATA USING CAMERA

Step 1:Collect face data with and without mask:

we have a library known as OpenCV which will help us to read the image and return array of color pixels. Command:- 1- !pip install opencv-python 2- import cv2


localhost:8888/notebooks/faceimage.ipynb
jupyter faceimage Last Checkpoint: 16 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]: !pip install opencv-python

        Requirement already satisfied: opencv-python in c:\users\poonam\anaconda3\lib\site-packages (4.6.0.66)
        Requirement already satisfied: numpy>=1.17.3 in c:\users\poonam\anaconda3\lib\site-packages (from opencv-python) (1.21.5)

In [2]: import cv2

In [3]: import matplotlib.pyplot as plt

In [4]: haar_data=cv2.CascadeClassifier('data.xml')
```

Introduction to OpenCv:

1. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
2. The library has more than 2500 optimized algorithms.
3. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.
4. Will help us to load images in Python and convert them into array.
5. Each index of array represents (red, green, blue) color pixel which ranges from 0 to 255.

Features of OpenCv:

- Face Detection
- Geometric Transformations
- Image Thresholding
- Smoothing Images
- Canny Edge Detection
- Background Removals
- Image Segmentation

Face Detection Using OpenCV:

So now we are going to see how to detect face from an image. Face detection algorithm was introduced by Viola and Jones in 2001. They divided this algorithm in four stages :

1. Haar Features Selection
2. Integral Images
3. AdaBoost
4. Cascading Classifier

we already have a XML file which is going to help us to detect faces from the image.

```
haar_data=cv2.CascadeClassifier('data.xml')
```

This code returns x, y, width and height of the face detected in the image. And we can draw a rectangle on the face using this code:

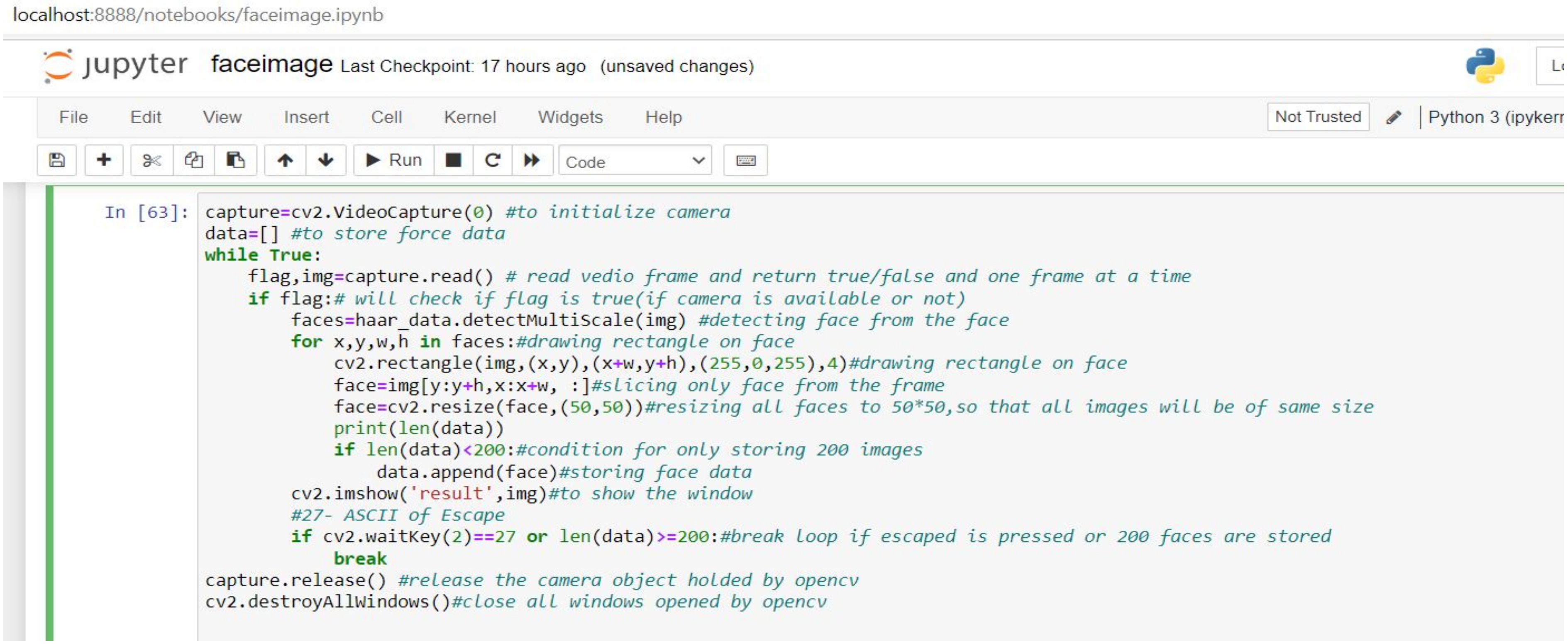
```
#cv2.rectangle(img,(x,y),(w,h),(b,g,r),border_thickness)
```

We will iterate over the array returned to us by detectMultiScale method and put x,y,w,h in cv2.rectangle

we need a lot of images of people wearing a mask and not wearing a mask.

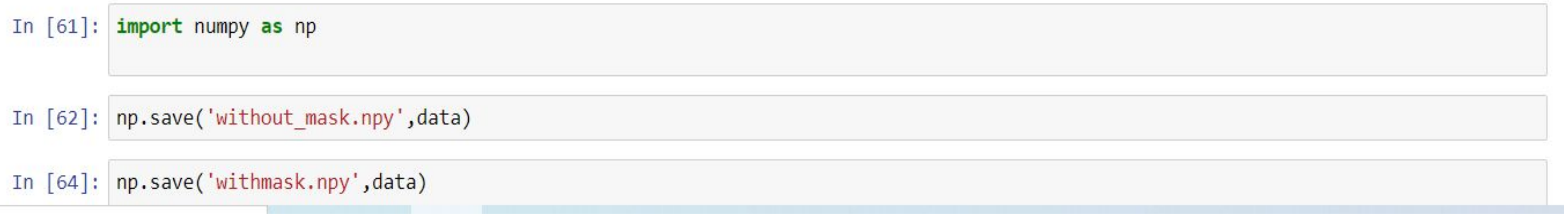
So first we need to collect data and we are going to collect data using our own camera. Here is the complete code to perform face detection using camera and storing face data only:

So first we need to collect data and we are going to collect data using our own camera. Here is the complete code to perform face detection using camera and storing face data only:


localhost:8888/notebooks/faceimage.ipynb
jupyter faceimage Last Checkpoint: 17 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykern
Run
Code
In [63]:

```
capture=cv2.VideoCapture(0) #to initialize camera
data=[] #to store force data
while True:
    flag,img=capture.read() # read vedio frame and return true/false and one frame at a time
    if flag:# will check if flag is true(if camera is available or not)
        faces=haar_data.detectMultiScale(img) #detecting face from the face
        for x,y,w,h in faces:#drawing rectangle on face
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)#drawing rectangle on face
            face=img[y:y+h,x:x+w, :]#slicing only face from the frame
            face=cv2.resize(face,(50,50))#resizing all faces to 50*50,so that all images will be of same size
            print(len(data))
            if len(data)<200:#condition for only storing 200 images
                data.append(face)#storing face data
        cv2.imshow('result',img)#to show the window
        #27- ASCII of Escape
        if cv2.waitKey(2)==27 or len(data)>=200:#break loop if escaped is pressed or 200 faces are stored
            break
capture.release() #release the camera object holded by opencv
cv2.destroyAllWindows()#close all windows opened by opencv
```

save the data in a numpy file and you can also plot the face data to check the data collected by OpenCV



```
In [61]: import numpy as np

In [62]: np.save('without_mask.npy',data)

In [64]: np.save('withmask.npy',data)
```

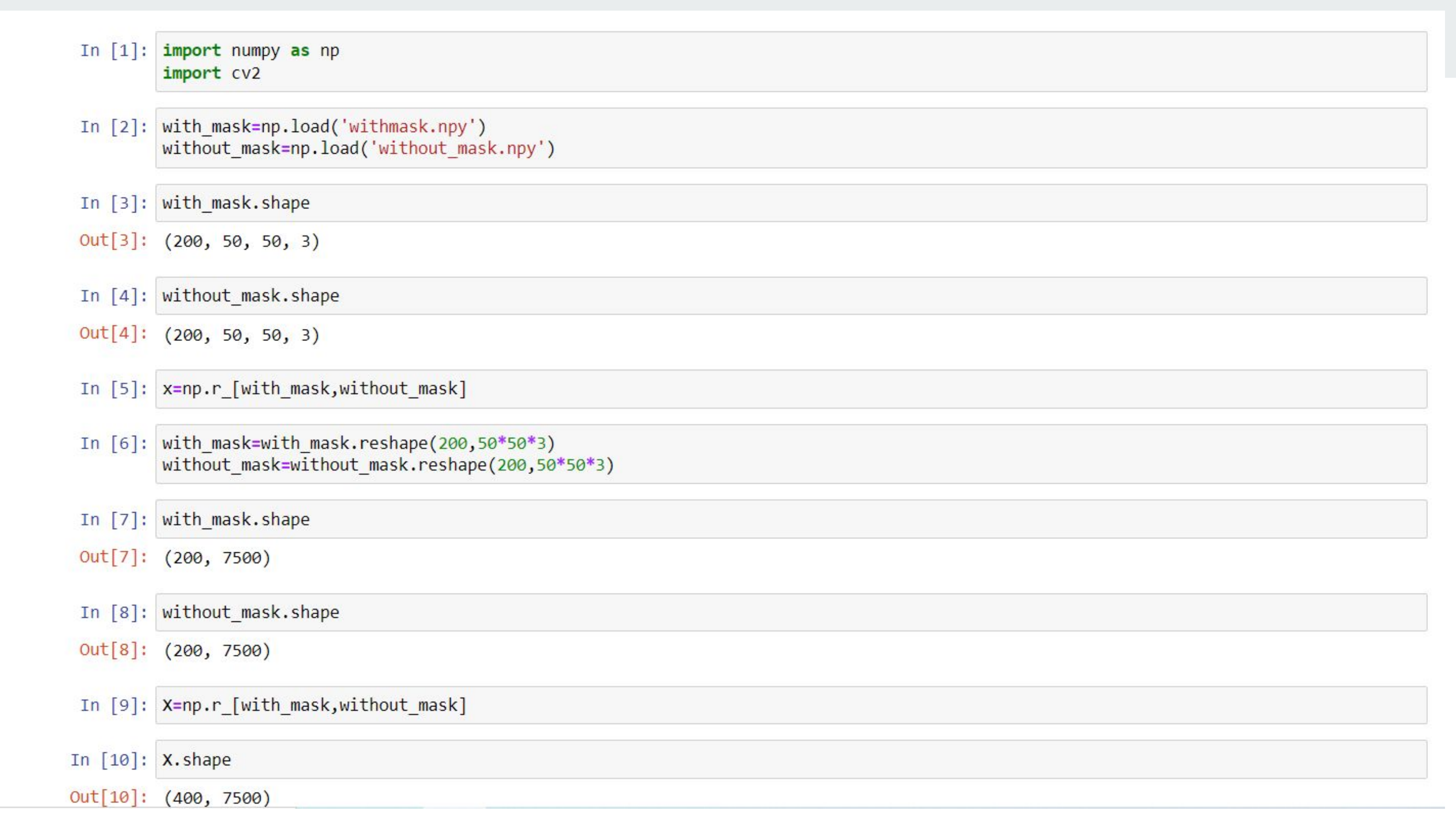
Now we can load the data anywhere and start processing it to apply machine learning on it

Step 2:-TRAIN DATA USING MACHINE LEARNING

Now we can see that data is loaded with the shape 200, 50, 50, 3.

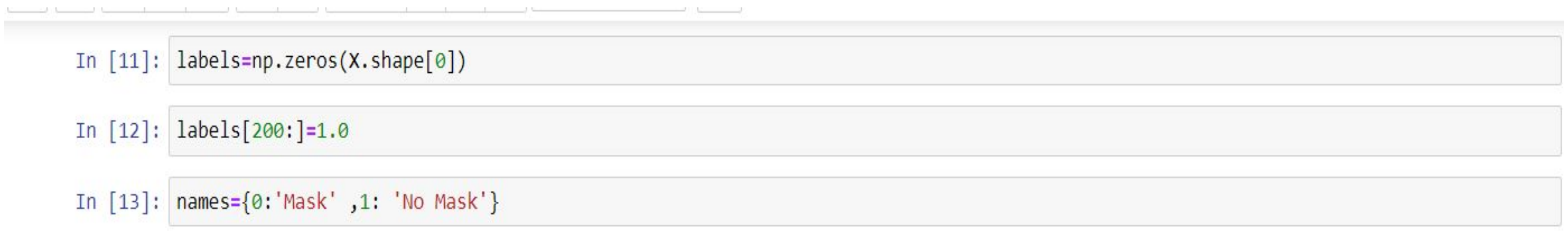
- Here 200 is the number of images we have collected
- 50, 50 is the size of each image
- 3 is the color channel (red, green, blue)

We can reshape the data to make it 2D :



```
In [1]: import numpy as np
        import cv2

In [2]: with_mask=np.load('withmask.npy')
        without_mask=np.load('without_mask.npy')

In [3]: with_mask.shape

Out[3]: (200, 50, 50, 3)

In [4]: without_mask.shape

Out[4]: (200, 50, 50, 3)

In [5]: x=np.r_[with_mask,without_mask]

In [6]: with_mask=with_mask.reshape(200,50*50*3)
        without_mask=without_mask.reshape(200,50*50*3)

In [7]: with_mask.shape

Out[7]: (200, 7500)

In [8]: without_mask.shape

Out[8]: (200, 7500)

In [9]: X=np.r_[with_mask,without_mask]

In [10]: X.shape

Out[10]: (400, 7500)
```

Using NPR will help you to store data row wise. So our features are ready. Now we need target variable. So let's create one array of zeros and assign first 200 indexes as zero and next 200 indexes as one. Because first 200 images belong to faces with mask and next 200 images belong to faces without mask.

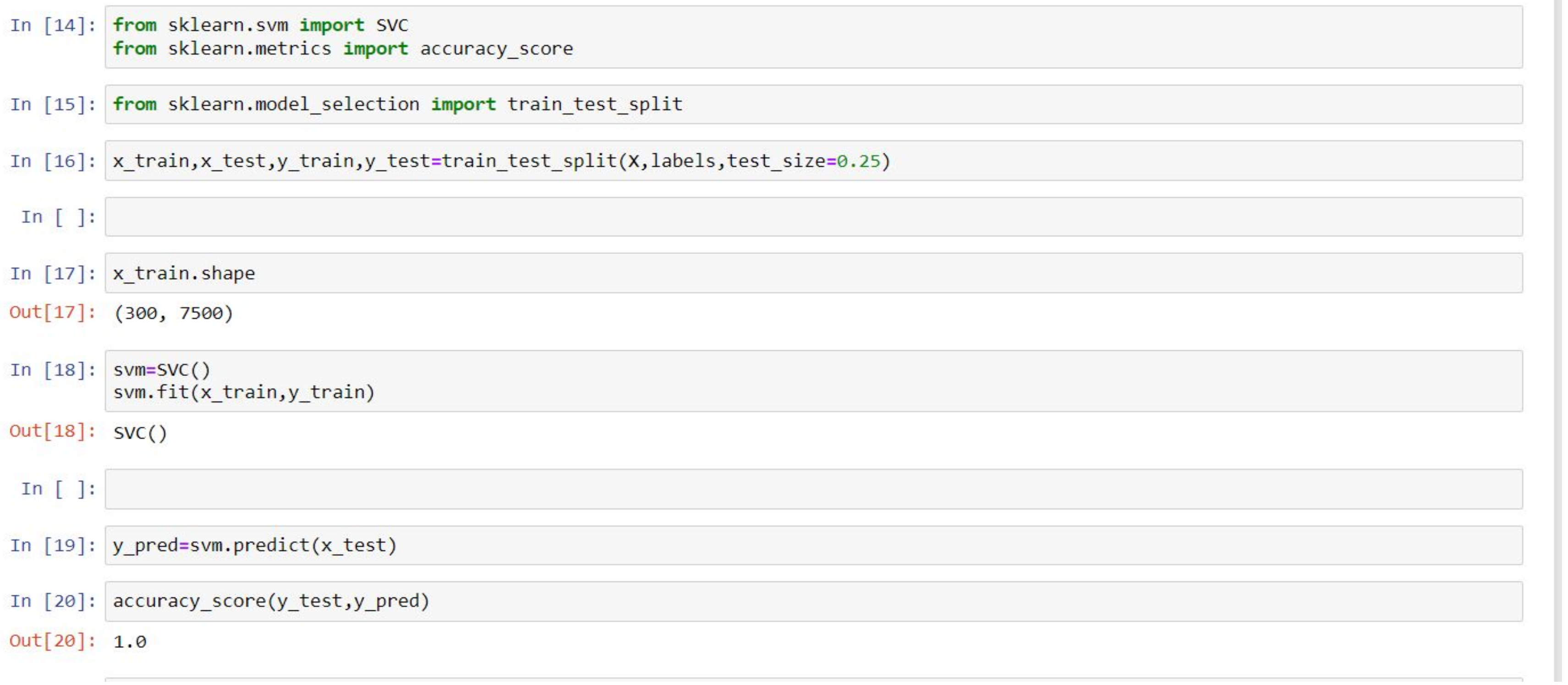


```
In [11]: labels=np.zeros(X.shape[0])

In [12]: labels[200:]=1.0

In [13]: names={0:'Mask' ,1: 'No Mask'}
```

Now we can apply machine learning on our data after dividing it into train and test.

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

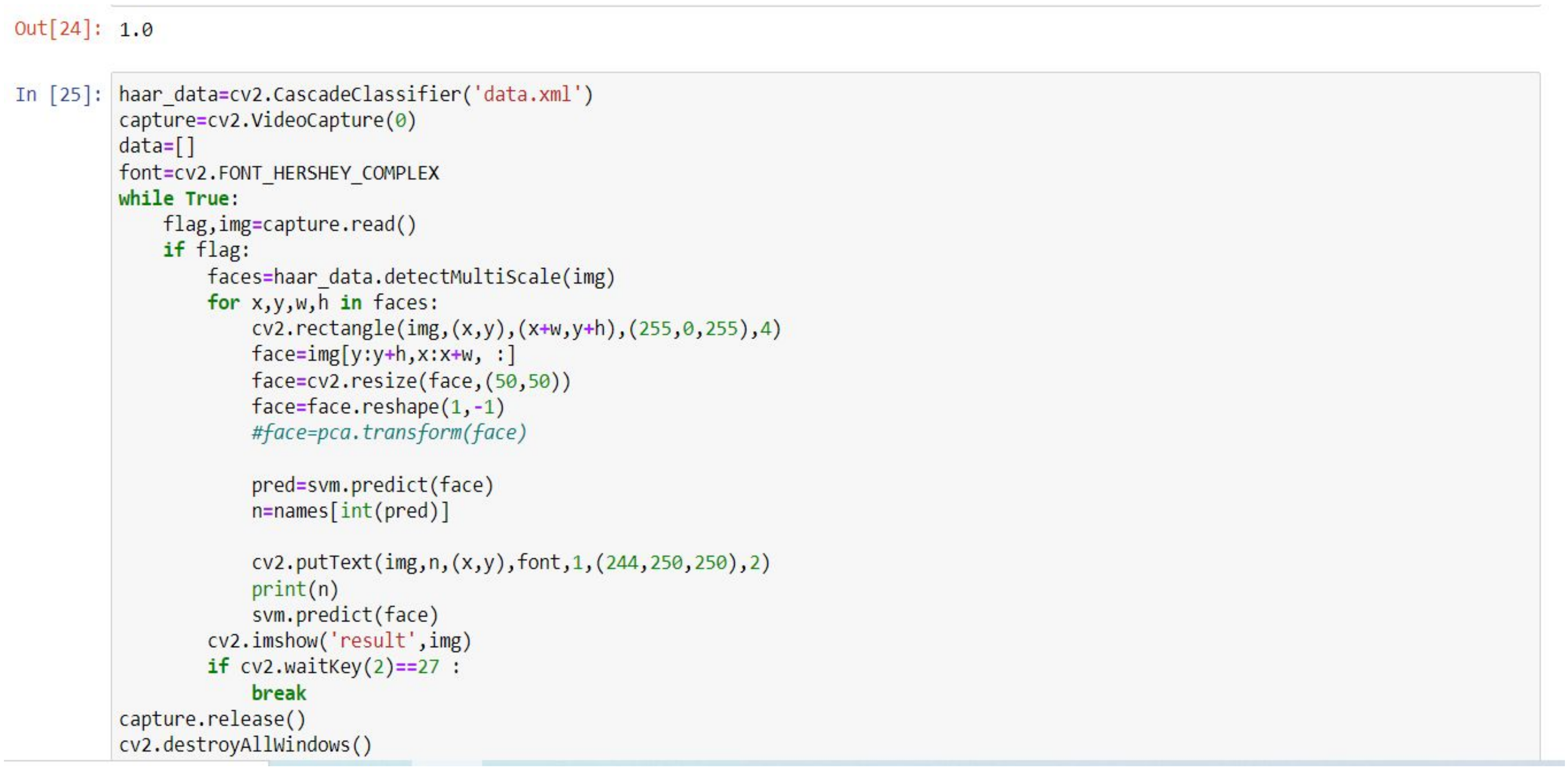
The algorithm we are using is SVM here. And after training this data on SVM we are getting accuracy of 100%.



```
In [14]: from sklearn.svm import SVC
         from sklearn.metrics import accuracy_score

In [15]: from sklearn.model_selection import train_test_split

In [16]: x_train,x_test,y_train,y_test=train_test_split(X,labels,test_size=0.25)

In [ ]:

In [17]: x_train.shape

Out[17]: (300, 7500)

In [18]: svm=SVC()
         svm.fit(x_train,y_train)

Out[18]: SVC()

In [ ]:

In [19]: y_pred=svm.predict(x_test)

In [20]: accuracy_score(y_test,y_pred)

Out[20]: 1.0
```

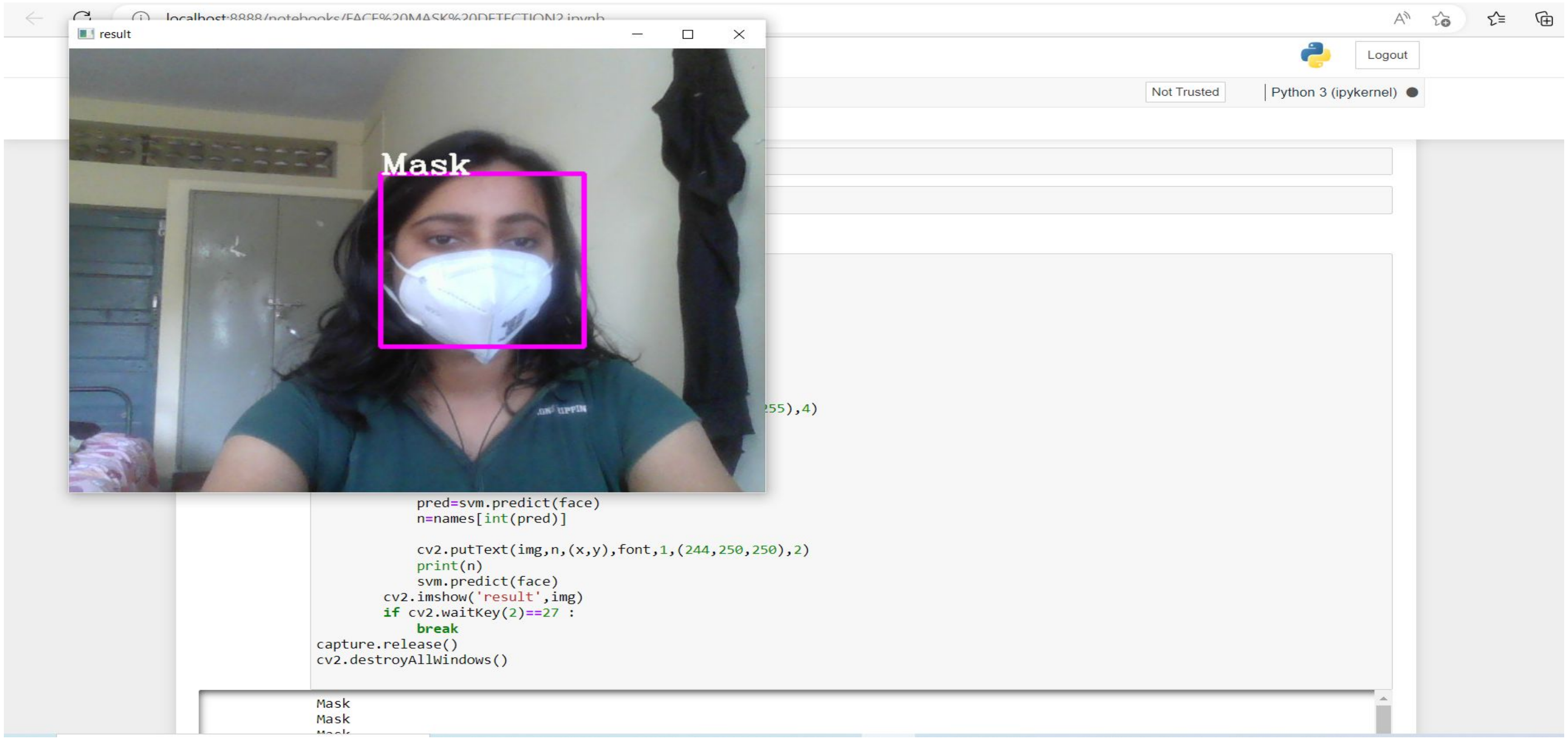
Step 3:-DO PREDICTION ON LIVE DATA USING CAMERA

we can test our faces with or without mask and check whether this algorithm is able to identity you that you are wearing a mask or not.



```
Out[24]: 1.0

In [25]: haar_data=cv2.CascadeClassifier('data.xml')
capture=cv2.VideoCapture(0)
data=[]
font=cv2.FONT_HERSHEY_COMPLEX
while True:
    flag,img=capture.read()
    if flag:
        faces=haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)
            face=img[y:y+h,x:x+w, :]
            face=cv2.resize(face,(50,50))
            face=face.reshape(1,-1)
            #face=pca.transform(face)

            pred=svm.predict(face)
            n=names[int(pred)]

            cv2.putText(img,n,(x,y),font,1,(244,250,250),2)
            print(n)
            svm.predict(face)
        cv2.imshow('result',img)
        if cv2.waitKey(2)==27 :
            break
capture.release()
cv2.destroyAllWindows()
```

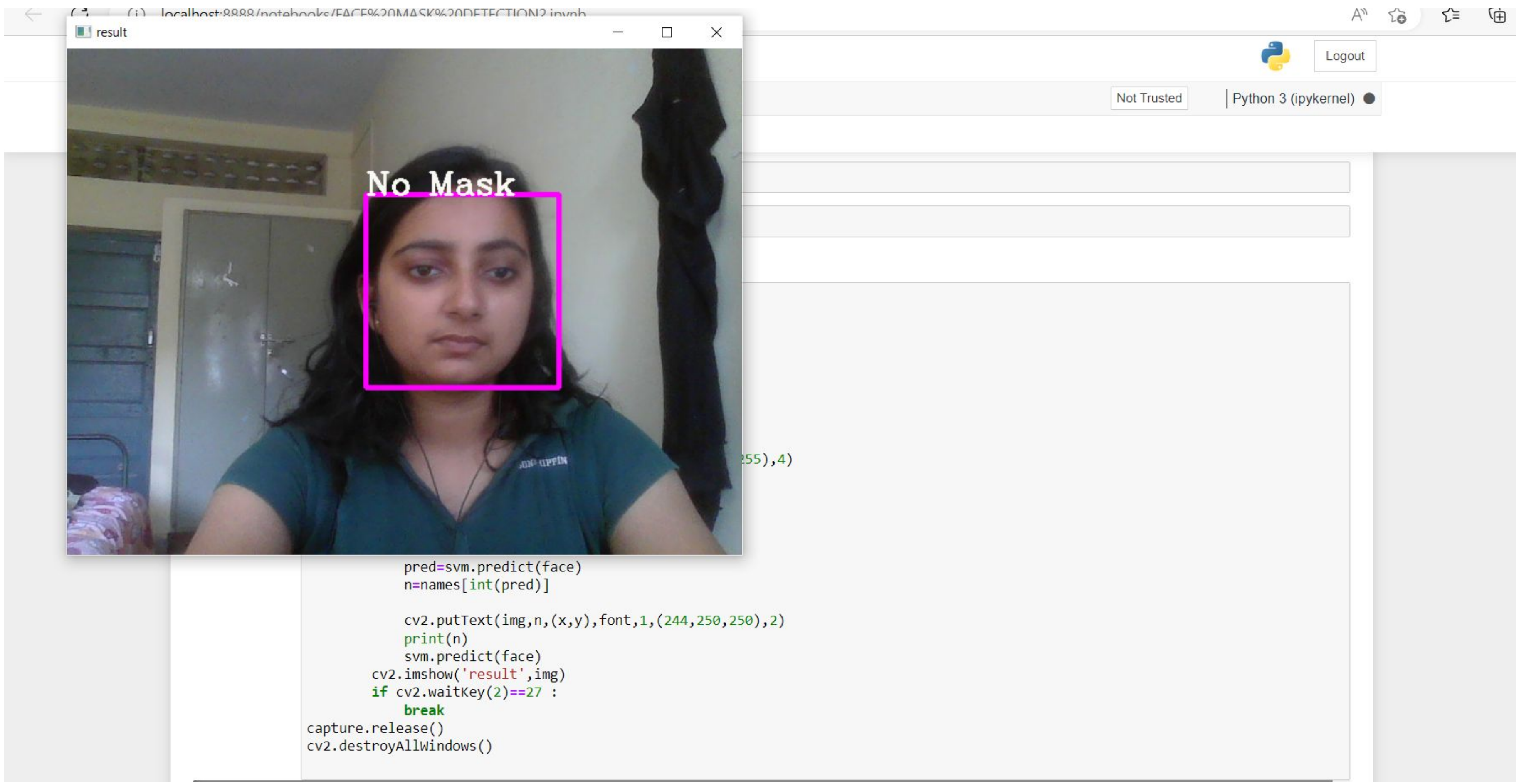
Result:-



```
255),4)

            pred=svm.predict(face)
            n=names[int(pred)]

            cv2.putText(img,n,(x,y),font,1,(244,250,250),2)
            print(n)
            svm.predict(face)
        cv2.imshow('result',img)
        if cv2.waitKey(2)==27 :
            break
capture.release()
cv2.destroyAllWindows()
```

```
Mask
Mask
```

No mask:-



```
255),4)

            pred=svm.predict(face)
            n=names[int(pred)]

            cv2.putText(img,n,(x,y),font,1,(244,250,250),2)
            print(n)
            svm.predict(face)
        cv2.imshow('result',img)
        if cv2.waitKey(2)==27 :
            break
capture.release()
cv2.destroyAllWindows()
```

TOOLS USED:-

1:-LOAD IMAGES USING PYTHON ,WE HAVE A LIBRARY KNOWN AS OpenCV WHICH WILL HELP US TO READ THE IMAGES AND RETURN ARRAY OF COLOR PIXELS.

2:-WE ARE GOING TO DETECT FACE FROM AN IMAGE.BY FACE DETECTION ALGORITHM WAS INTRODUCED BY VIOLA AND JONES IN 2001.THEY DIVIDED THIS ALGORITHM IN FOUR STAGES:

1. HEAR FEATURES SELECTION
2. INTEGRAL IMAGES
3. AdaBoost
4. cascading classifier

3:-USE SVM(SUPPORT VECTOR MACHINE ALGORITHM) TO TRAIN AND TEST THE DATA.

4:-CHECK BY USING CAMERA,WEATHER ALGORITHM ABLE TO DETECT PERSON IS WEARING MASK OR NOT.

THANK YOU