Face Mask Detection with Convolutional Neural Network(CNN)

Machine Learning

American International University Bangladesh

Name: Zisan Ahmed
ID: 19-39294-1
Sec: A

Face Mask Detection with Convolutional Neural Network

Project Objective:

The objective of the facemask detection project is to develop an accurate and efficient model that can detect whether a person is wearing a facemask or not. With the ongoing COVID-19 pandemic, the use of facemasks has become a crucial measure to control the spread of the virus. The use of automated systems to detect facemasks can assist in enforcing regulations and ensuring public safety.

The project aims to use convolutional neural networks (CNN) to classify images as either containing a person wearing a facemask or not. This involves training a CNN model on a large dataset of images of people wearing or not wearing facemasks. The trained model can then be used to predict whether a person in a new image is wearing a facemask or not.

The objective of this project is not only to develop an accurate and efficient model but also to explore various approaches to improve the model's performance. This includes experimenting with different architectures of CNN, hyperparameter tuning, and data augmentation techniques. Furthermore, the project aims to compare the performance of different models and techniques to identify the most effective approach.

The final objective of this project is to develop a practical solution that can be deployed in real-world scenarios such as airports, hospitals, and public transport systems, among others. An accurate and efficient facemask detection system can assist in preventing the spread of COVID-19 and other contagious diseases, thereby ensuring public safety.

Project Methodology:

Data Collection: Collecting a large dataset of images that includes individuals both wearing and not wearing face masks. This dataset will be used to train and evaluate our deep learning model.

Data Preprocessing: Preprocessing the dataset by resizing images, converting images to grayscale or RGB format, and normalizing pixel values to improve the model's performance.

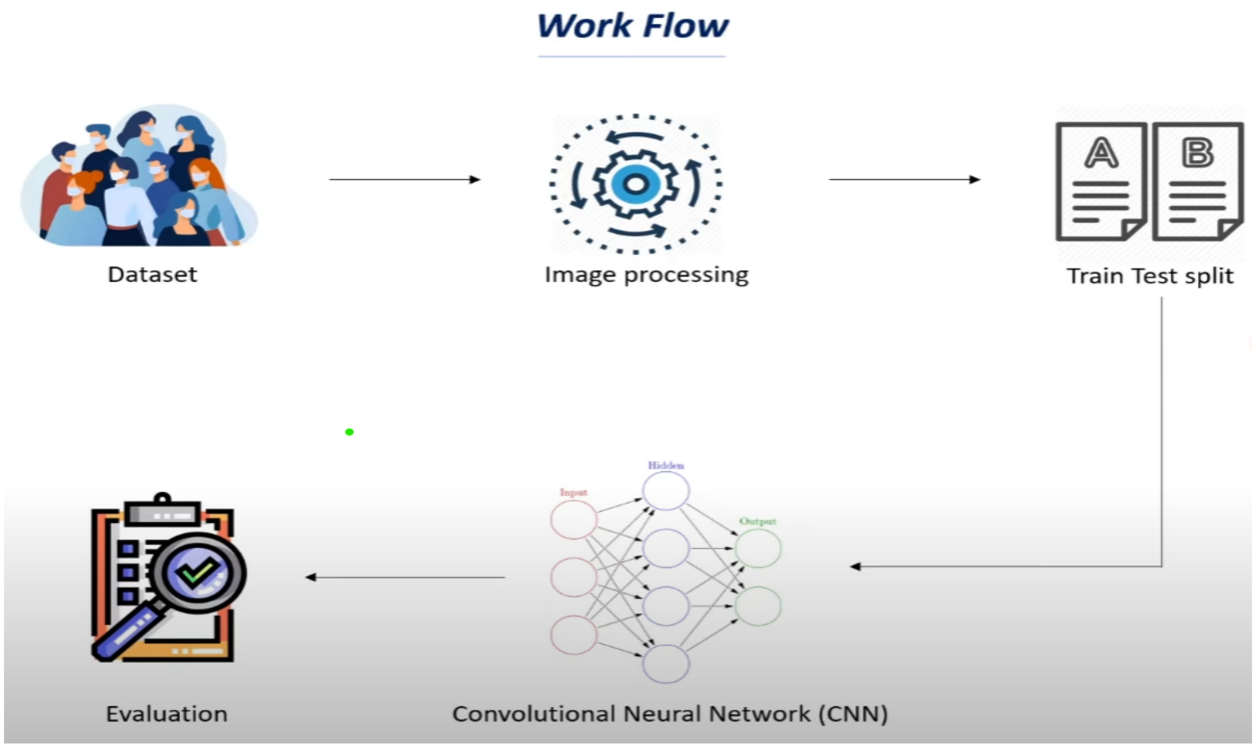
Model Selection: Selecting the appropriate deep learning model for this problem. Convolutional neural networks (CNNs) are commonly used for image classification tasks, and we will be exploring various CNN architectures to select the best one for our problem.

Model Training: Training the selected model on the preprocessed dataset. This involves splitting the dataset into training, validation, and testing sets, training the model on the training set, tuning the model's hyperparameters, and evaluating the model's performance on the validation and testing sets.

Model Evaluation: Evaluating the performance of the optimized model on the testing set. This involves calculating metrics such as accuracy, precision, recall, and F1 score.

Predictive System : Deploying the optimized model to a production environment where it can be used to detect face masks in real-time.

Block Diagram and Workflow Diagram of Proposed Model



The first step is to collecting the appropriate Dataset. The dataset contains two set of images. First set of images is people wearing mask and second set of images is people not wearing mask. We will be working on kaggle dataset for this porpuse.

The next step is Image Processing. We have to process such as resizing the image, converting this images into numpy arrays ets.

Next step is to split this images as training data and testing data. We using training set to train our neural network and test set to evaluate our models performance.

When this step is done we feed this data to our convolutional neural network which tries to find out how a person look like with or without a mask.

When this training is done we evaluate convolutional neural network with certain evaluation matrix. When all these thing will done we will build ta predictive system if you feed the image to convolutional neural network it will tell you whether that person is wearing a mask or not. So This are the steps we will follow.

Data Collection Procedure

This is the following URL that I have collected the data

https://www.kaggle.com/datasets/omkargurav/face-mask-dataset

I have access and downloaded the data through API token.

The procedure to configure the kaggle.json file in Visual Studio Code is slightly different. Ifollowed these steps:

Create a new folder named .kaggle in my home directory. I can do this by running the following command in the terminal:

```
mkdir ~/.kaggle
```

Copying the kaggle.json file to the .kaggle folder. I can do this by running the following command in the terminal:

```
cp /path/to/kaggle.json ~/.kaggle/
```

Replacing /path/to/kaggle.json with the actual path to the kaggle.json file on my computer.

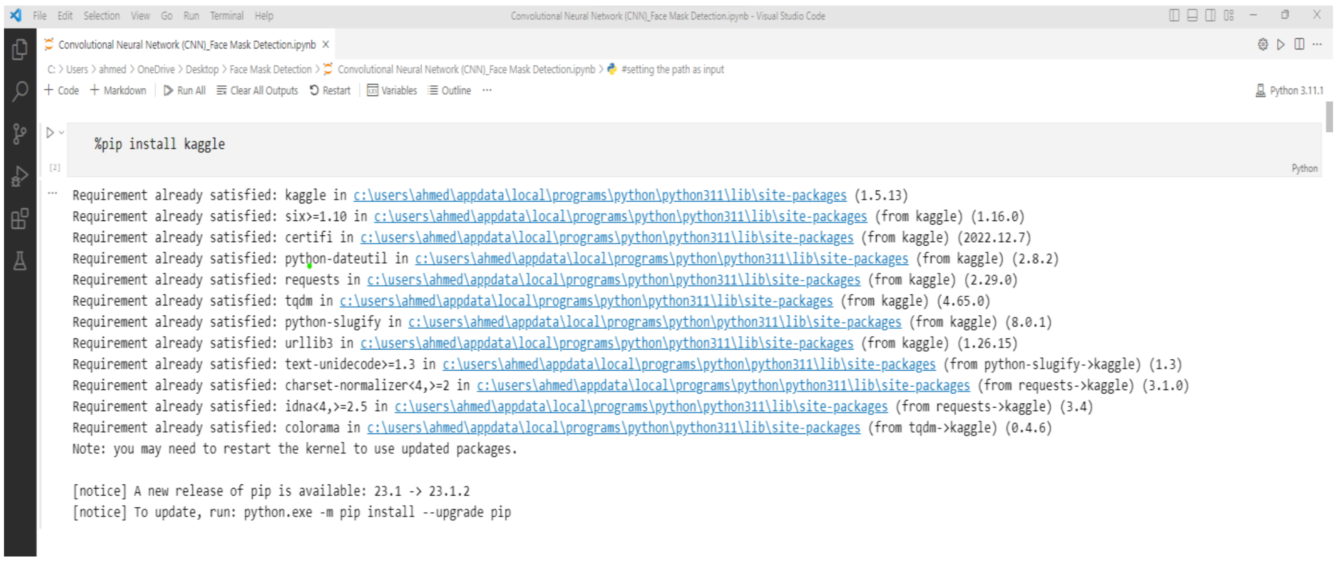
Need to change the permissions of the kaggle.json file. I can do this by running the following command in the terminal:

```
chmod 600 ~/.kaggle/kaggle.json
```

This command sets the file permissions to read and write for the owner of the file, and no permissions for anyone else.

Once I have completed these steps, you can use the Kaggle API in your Python code in Visual Studio Code. The API will automatically look for the kaggle.json file in the ~/.kaggle folder.

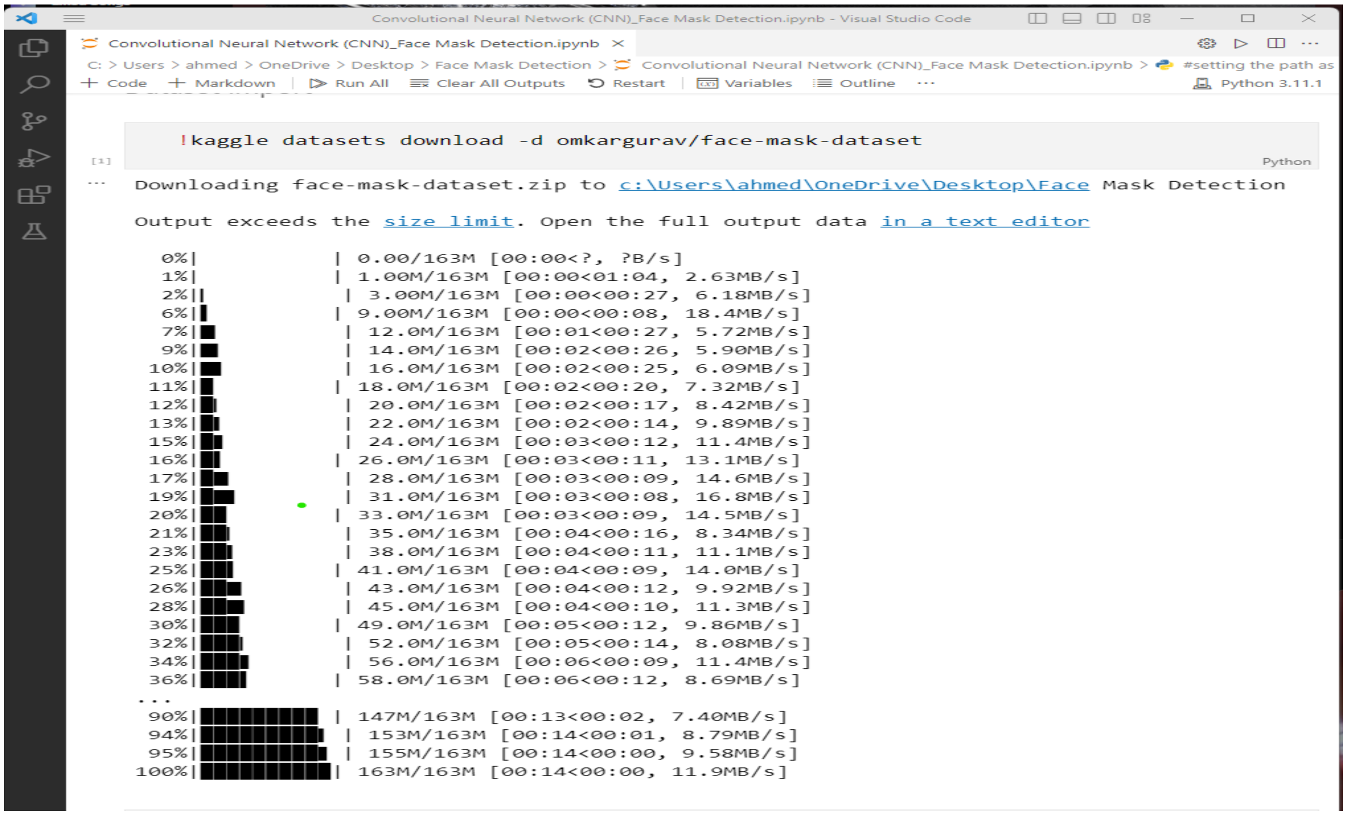
After that we need to install kaggle in visual studio code


File Edit Selection View Go Run Terminal Help
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as input
+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ...
Python 3.11.1

```
%pip install kaggle
```

```
Requirement already satisfied: kaggle in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (1.5.13)
Requirement already satisfied: six>=1.10 in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from kaggle) (2022.12.7)
Requirement already satisfied: python-dateutil in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from kaggle) (2.29.0)
Requirement already satisfied: tqdm in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from kaggle) (4.65.0)
Requirement already satisfied: python-slugify in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from kaggle) (1.26.15)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from requests->kaggle) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from requests->kaggle) (3.4)
Requirement already satisfied: colorama in c:\users\ahmed\appdata\local\programs\python\python311\lib\site-packages (from tqdm->kaggle) (0.4.6)
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

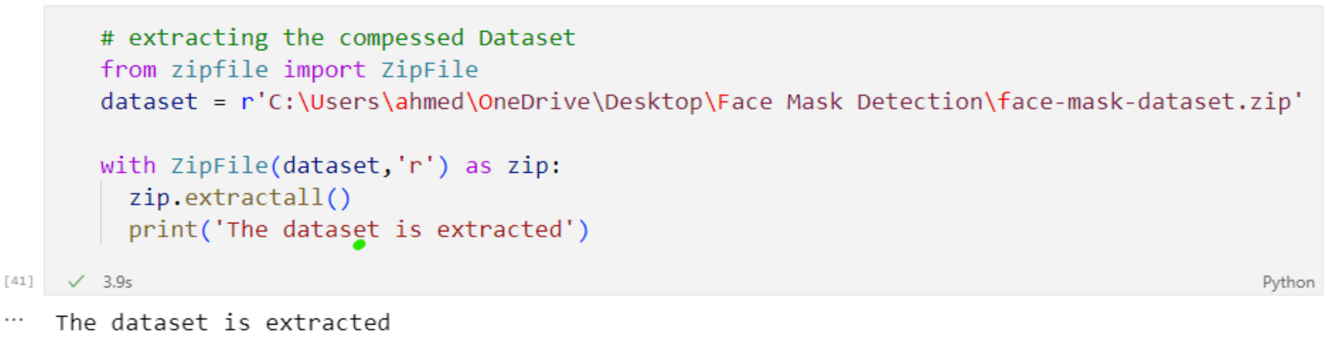
Now we have to import out dataset copying the api command through the link


Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ...
Python 3.11.1

```
!kaggle datasets download -d omkargurav/face-mask-dataset
```

```
Downloading face-mask-dataset.zip to c:\Users\ahmed\OneDrive\Desktop\Face Mask Detection

Output exceeds the size limit. Open the full output data in a text editor

  0%|           | 0.00/163M [00:00<?, ?B/s]
  1%|           | 1.00M/163M [00:00<01:04, 2.63MB/s]
  2%|▏          | 3.00M/163M [00:00<00:27, 6.18MB/s]
  6%|▌          | 9.00M/163M [00:00<00:08, 18.4MB/s]
  7%|▋          | 12.0M/163M [00:01<00:27, 5.72MB/s]
  9%|▊          | 14.0M/163M [00:02<00:26, 5.90MB/s]
 10%|▉          | 16.0M/163M [00:02<00:25, 6.09MB/s]
 11%|█          | 18.0M/163M [00:02<00:20, 7.32MB/s]
 12%|█▏         | 20.0M/163M [00:02<00:17, 8.42MB/s]
 13%|█▎         | 22.0M/163M [00:02<00:14, 9.89MB/s]
 15%|█▍         | 24.0M/163M [00:03<00:12, 11.4MB/s]
 16%|█▌         | 26.0M/163M [00:03<00:11, 13.1MB/s]
 17%|█▋         | 28.0M/163M [00:03<00:09, 14.6MB/s]
 19%|█▉         | 31.0M/163M [00:03<00:08, 16.8MB/s]
 20%|██         | 33.0M/163M [00:03<00:09, 14.5MB/s]
 21%|██▏        | 35.0M/163M [00:04<00:16, 8.34MB/s]
 23%|██▎        | 38.0M/163M [00:04<00:11, 11.1MB/s]
 25%|██▌        | 41.0M/163M [00:04<00:09, 14.0MB/s]
 26%|██▋        | 43.0M/163M [00:04<00:12, 9.92MB/s]
 28%|██▊        | 45.0M/163M [00:04<00:10, 11.3MB/s]
 30%|███        | 49.0M/163M [00:05<00:12, 9.86MB/s]
 32%|███▏       | 52.0M/163M [00:05<00:14, 8.08MB/s]
 34%|███▍       | 56.0M/163M [00:06<00:09, 11.4MB/s]
 36%|███▌       | 58.0M/163M [00:06<00:12, 8.69MB/s]
...
 90%|█████████  | 147M/163M [00:13<00:02, 7.40MB/s]
 94%|█████████▍ | 153M/163M [00:14<00:01, 8.79MB/s]
 95%|█████████▌ | 155M/163M [00:14<00:00, 9.58MB/s]
100%|██████████| 163M/163M [00:14<00:00, 11.9MB/s]
```

Now we need to extract the dataset



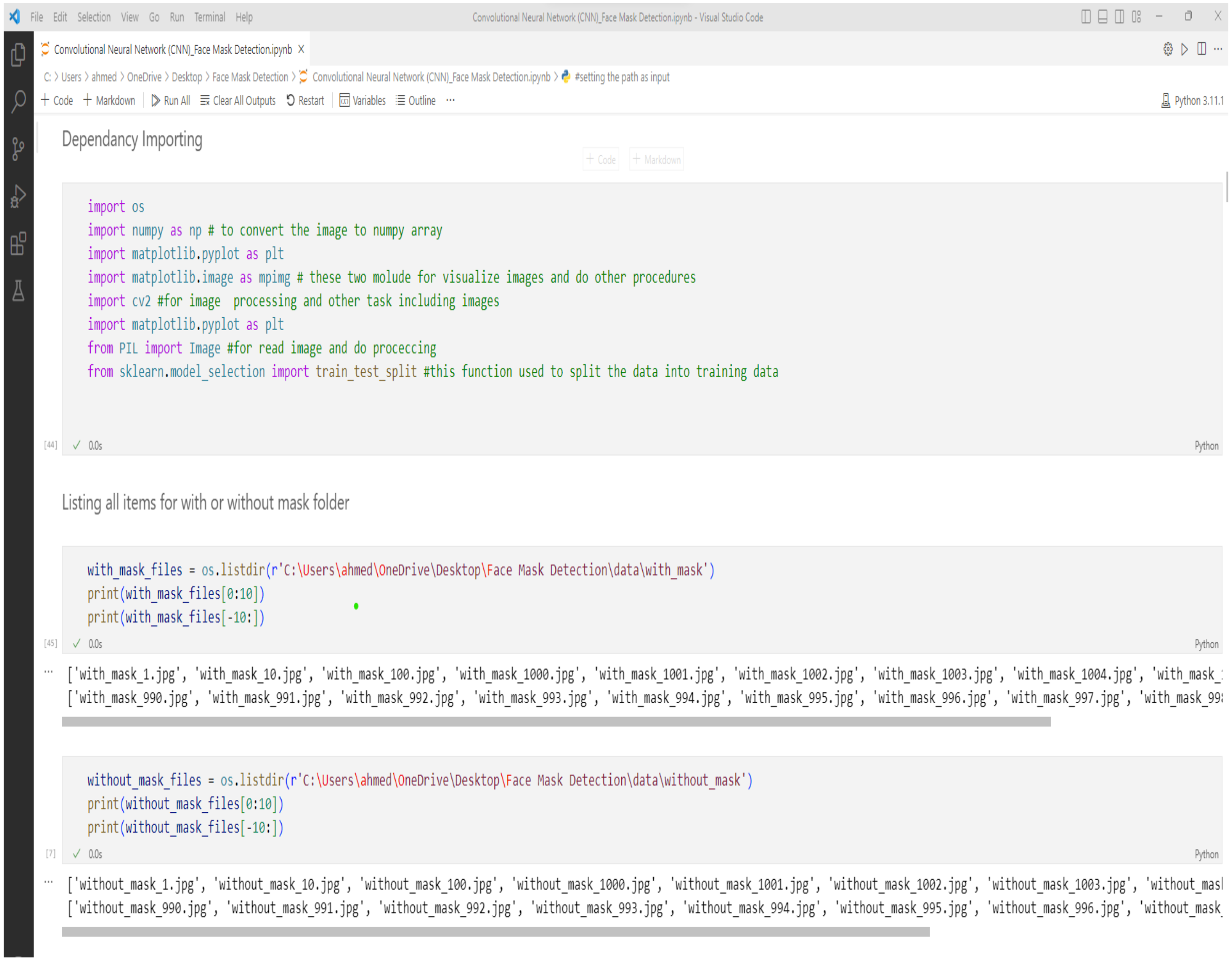
```
# extracting the compressed Dataset
from zipfile import ZipFile
dataset = r'C:\Users\ahmed\OneDrive\Desktop\Face Mask Detection\face-mask-dataset.zip'

with ZipFile(dataset,'r') as zip:
  zip.extractall()
  print('The dataset is extracted')
```

[41] ✓ 3.9s Python

```
The dataset is extracted
```

Automatically created a data folder having all the datast.

Now we will import all our dependacies that we need in our project and then we eill convert the dataset into list.


File Edit Selection View Go Run Terminal Help
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as input
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1

Dependancy Importing

```
import os
import numpy as np # to convert the image to numpy array
import matplotlib.pyplot as plt
import matplotlib.image as mpimg # these two molude for visualize images and do other procedures
import cv2 #for image  processing and other task including images
import matplotlib.pyplot as plt
from PIL import Image #for read image and do proceccing
from sklearn.model_selection import train_test_split #this function used to split the data into training data
```

[44] ✓ 0.0s Python

Listing all items for with or without mask folder

```
with_mask_files = os.listdir(r'C:\Users\ahmed\OneDrive\Desktop\Face Mask Detection\data\with_mask')
print(with_mask_files[0:10])
print(with_mask_files[-10:])
```

[45] ✓ 0.0s Python

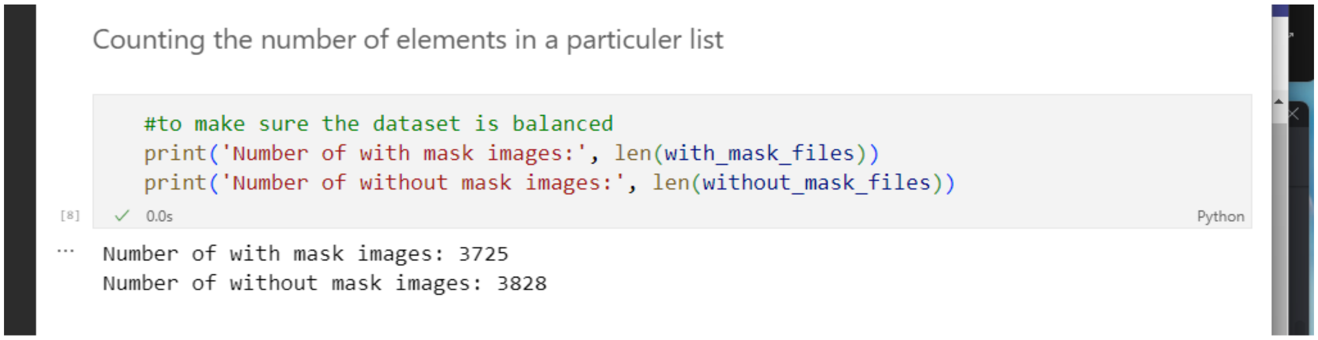
```
['with_mask_1.jpg', 'with_mask_10.jpg', 'with_mask_100.jpg', 'with_mask_1000.jpg', 'with_mask_1001.jpg', 'with_mask_1002.jpg', 'with_mask_1003.jpg', 'with_mask_1004.jpg', 'with_mask_
['with_mask_990.jpg', 'with_mask_991.jpg', 'with_mask_992.jpg', 'with_mask_993.jpg', 'with_mask_994.jpg', 'with_mask_995.jpg', 'with_mask_996.jpg', 'with_mask_997.jpg', 'with_mask_99
```

```
without_mask_files = os.listdir(r'C:\Users\ahmed\OneDrive\Desktop\Face Mask Detection\data\without_mask')
print(without_mask_files[0:10])
print(without_mask_files[-10:])
```

[7] ✓ 0.0s Python

```
['without_mask_1.jpg', 'without_mask_10.jpg', 'without_mask_100.jpg', 'without_mask_1000.jpg', 'without_mask_1001.jpg', 'without_mask_1002.jpg', 'without_mask_1003.jpg', 'without_mas
['without_mask_990.jpg', 'without_mask_991.jpg', 'without_mask_992.jpg', 'without_mask_993.jpg', 'without_mask_994.jpg', 'without_mask_995.jpg', 'without_mask_996.jpg', 'without_mask_
```

Now we will count the list item



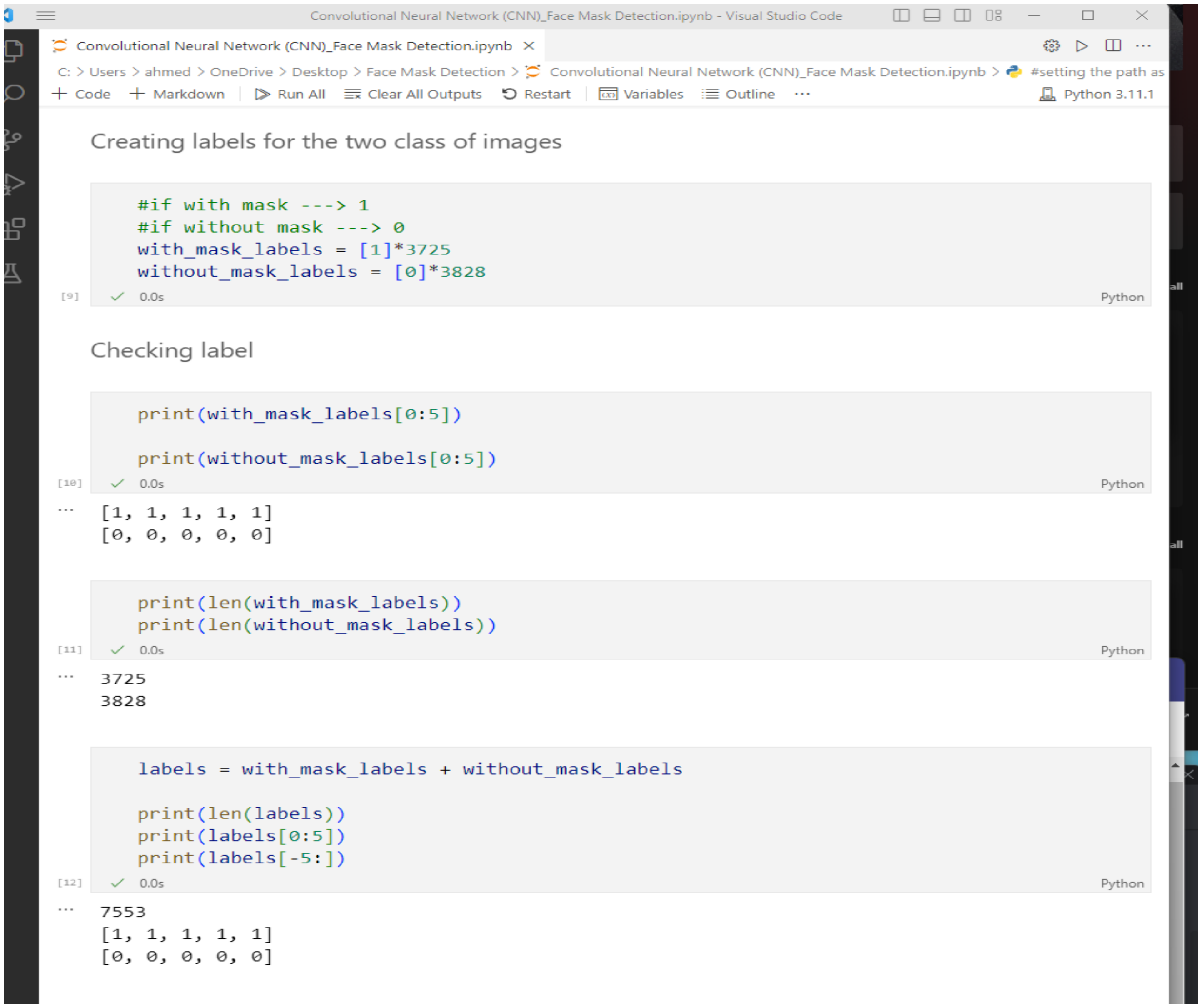
Counting the number of elements in a particuler list

```
#to make sure the dataset is balanced
print('Number of with mask images:', len(with_mask_files))
print('Number of without mask images:', len(without_mask_files))
```

[8] ✓ 0.0s Python

```
Number of with mask images: 3725
Number of without mask images: 3828
```

Now we label the data for two class of images. If a person wearing mask we will label as 1 and if not the 0. After that we will check the label and their length



Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline ...
Python 3.11.1

Creating labels for the two class of images

```
#if with mask ---> 1
#if without mask ---> 0
with_mask_labels = [1]*3725
without_mask_labels = [0]*3828
```

[9] ✓ 0.0s Python

Checking label

```
print(with_mask_labels[0:5])

print(without_mask_labels[0:5])
```

[10] ✓ 0.0s Python

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

```
print(len(with_mask_labels))
print(len(without_mask_labels))
```

[11] ✓ 0.0s Python

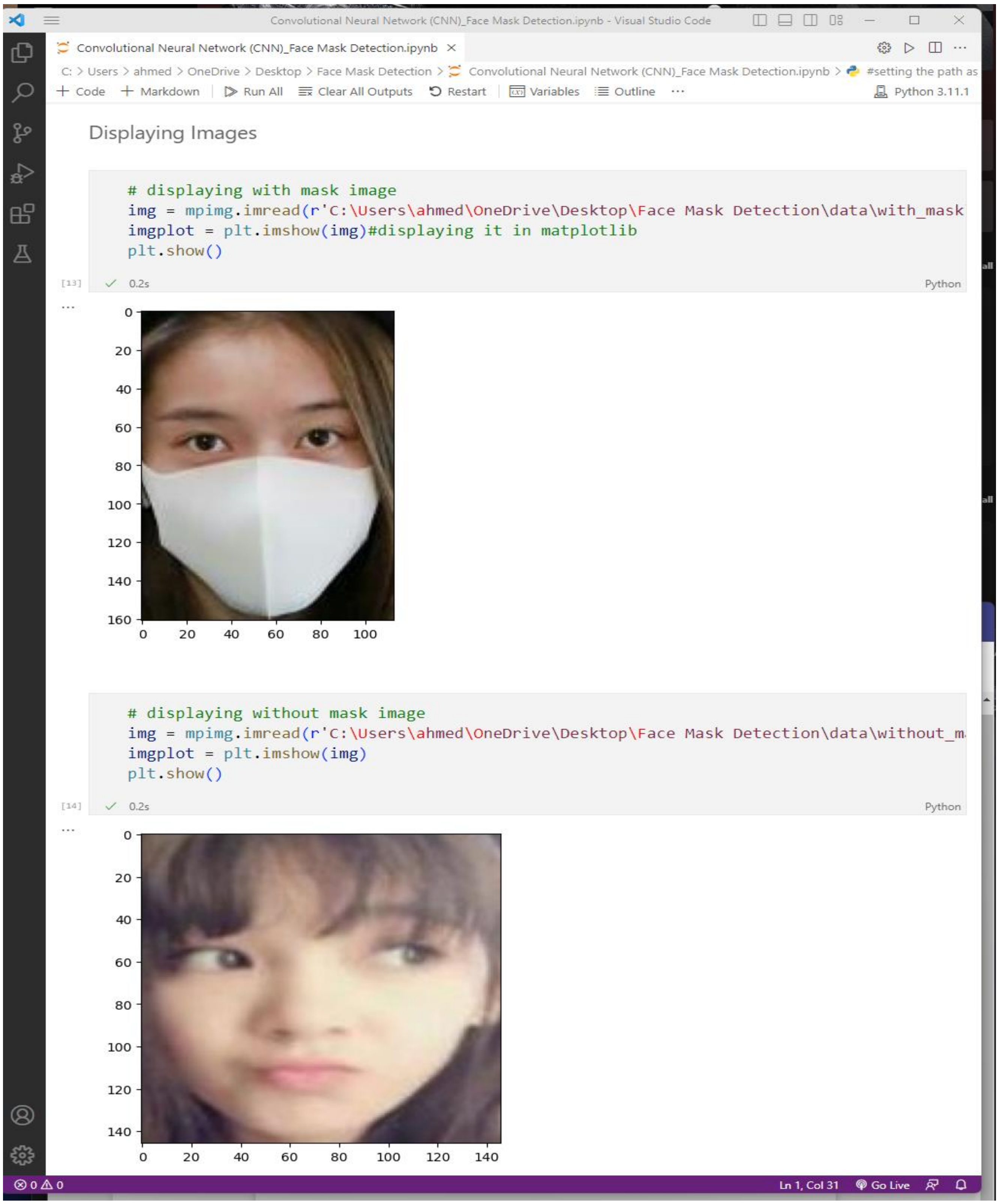
```
3725
3828
```

```
labels = with_mask_labels + without_mask_labels

print(len(labels))
print(labels[0:5])
print(labels[-5:])
```

[12] ✓ 0.0s Python

```
7553
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

Now we will display the images in matplotlib


Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1

Displaying Images

```
# displaying with mask image
img = mpimg.imread(r'C:\Users\ahmed\OneDrive\Desktop\Face Mask Detection\data\with_mask
imgplot = plt.imshow(img)#displaying it in matplotlib
plt.show()
```

[13] ✓ 0.2s Python

```
# displaying without mask image
img = mpimg.imread(r'C:\Users\ahmed\OneDrive\Desktop\Face Mask Detection\data\without_m
imgplot = plt.imshow(img)
plt.show()
```

[14] ✓ 0.2s Python

Now we need to do the image proceccing.

In our project, we have processed the images by resizing them to a uniform shape of 128x128 pixels using the resize() function from the PIL (Python Imaging Library) module. This step ensures that all images have the same size, which is required for training a convolutional neural network.

Next, we have converted the images to RGB format using the convert() method. This is necessary because some of the images in the dataset may be in black and white format, and we want all the images to have the same color format before training our model.

After that, we have converted the images to numpy arrays using the np.array() function. This is because most machine learning frameworks, including TensorFlow and Keras, work with numpy arrays as input data.

we have then appended the numpy arrays of the images to a list called data. This list contains all the images in the dataset, with and without masks. Finally, we will use this data to train your convolutional neural network to detect face masks in images

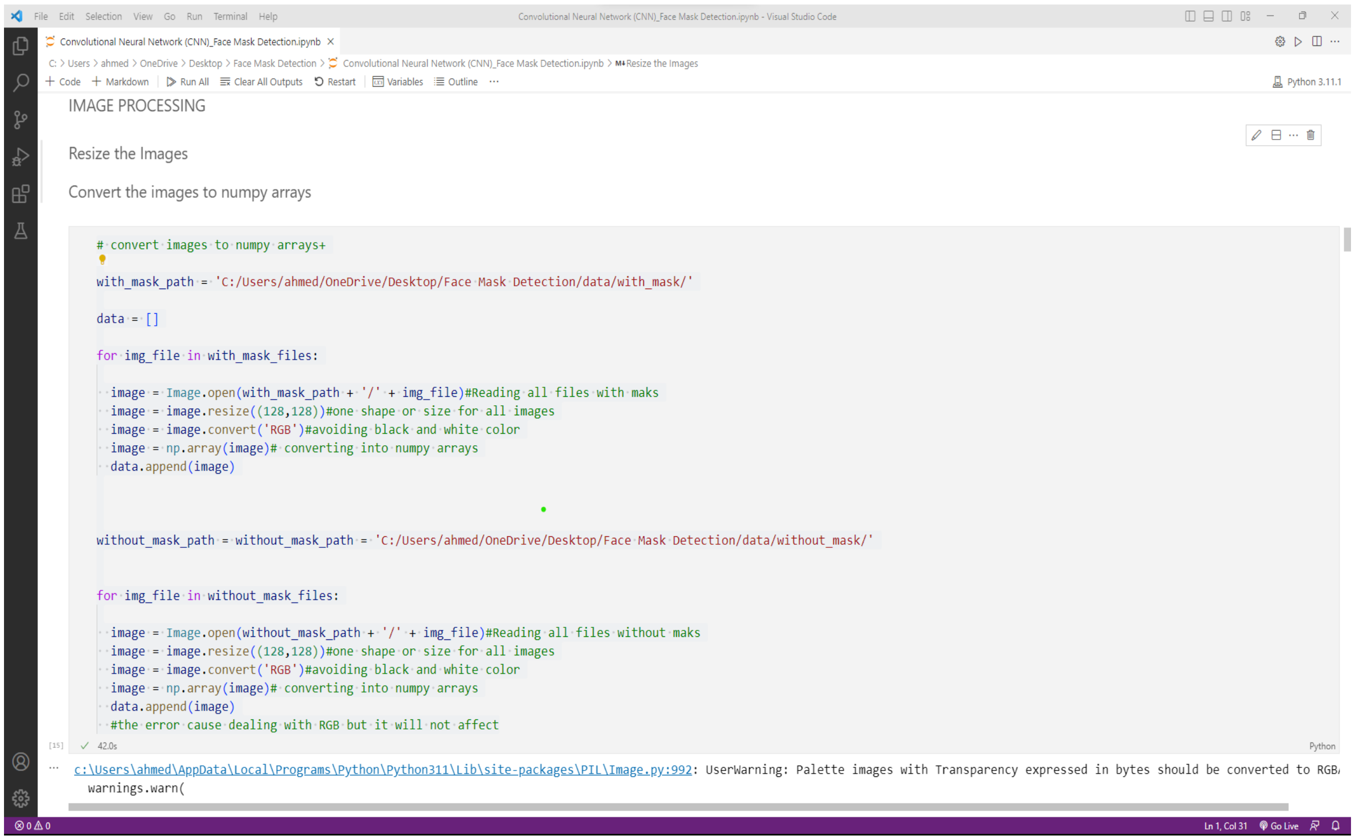

File Edit Selection View Go Run Terminal Help
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > Resize the Images
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1
Ln 1, Col 31 Go Live

IMAGE PROCESSING

Resize the Images

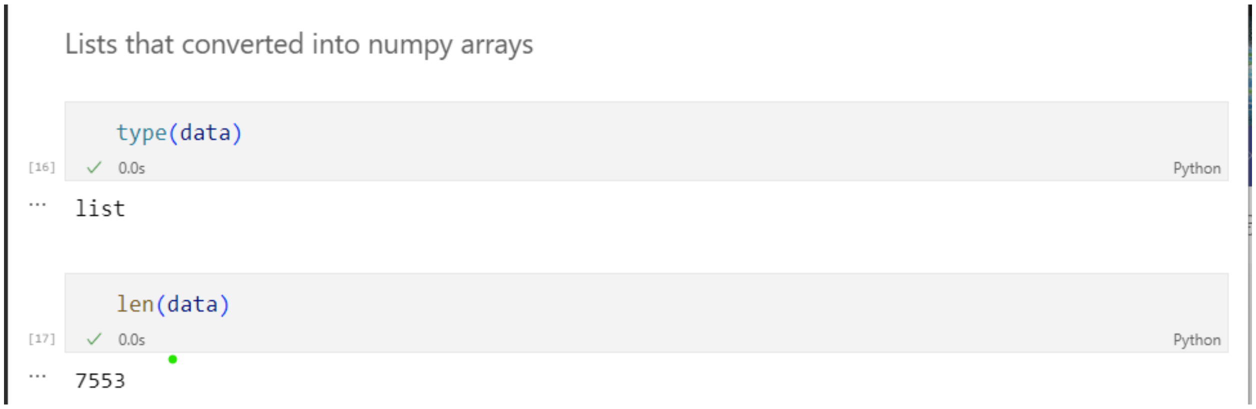
Convert the images to numpy arrays

```
# convert images to numpy arrays+

with_mask_path = 'C:/Users/ahmed/OneDrive/Desktop/Face Mask Detection/data/with_mask/'

data = []

for img_file in with_mask_files:

  image = Image.open(with_mask_path + '/' + img_file)#Reading all files with maks
  image = image.resize((128,128))#one shape or size for all images
  image = image.convert('RGB')#avoiding black and white color
  image = np.array(image)# converting into numpy arrays
  data.append(image)


without_mask_path = without_mask_path = 'C:/Users/ahmed/OneDrive/Desktop/Face Mask Detection/data/without_mask/'


for img_file in without_mask_files:

  image = Image.open(without_mask_path + '/' + img_file)#Reading all files without maks
  image = image.resize((128,128))#one shape or size for all images
  image = image.convert('RGB')#avoiding black and white color
  image = np.array(image)# converting into numpy arrays
  data.append(image)
  #the error cause dealing with RGB but it will not affect
```

[15] ✓ 42.0s Python

```
c:\Users\ahmed\AppData\Local\Programs\Python\Python311\Lib\site-packages\PIL\Image.py:992: UserWarning: Palette images with Transparency expressed in bytes should be converted to RGB
  warnings.warn(
```

Checking the data again



Lists that converted into numpy arrays

```
type(data)
```

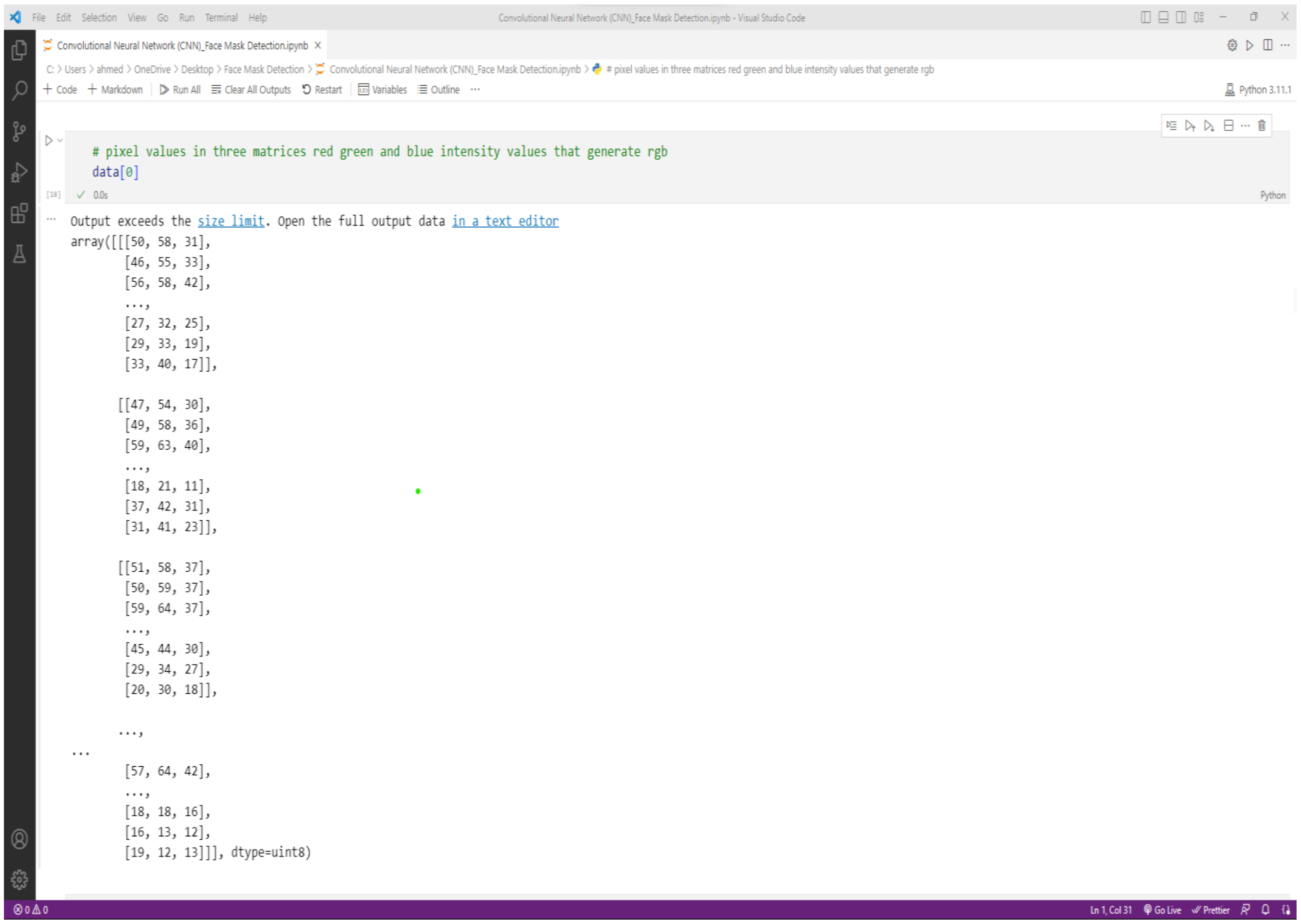
[16] ✓ 0.0s Python

```
list
```

```
len(data)
```

[17] ✓ 0.0s Python

```
7553
```

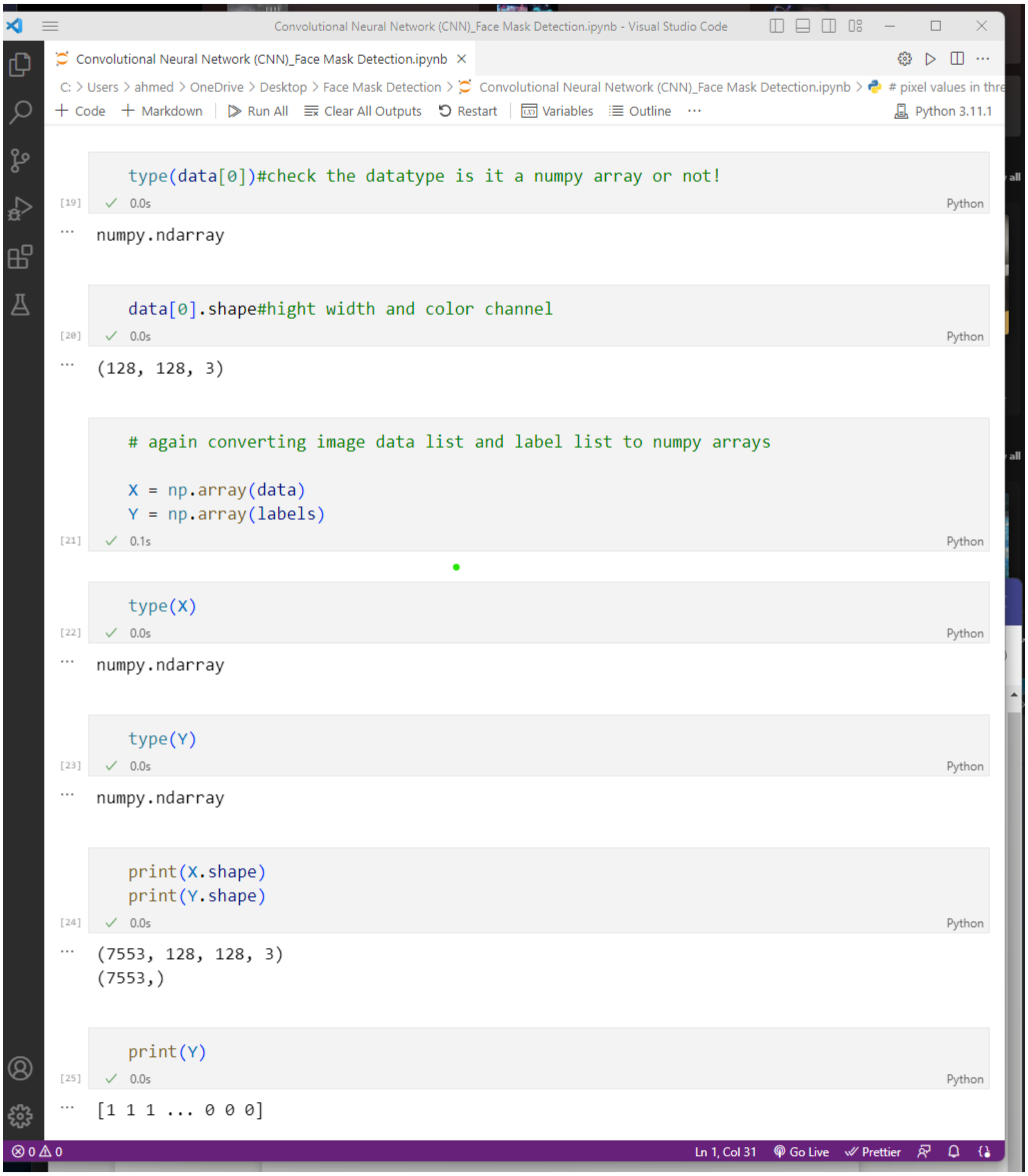


```
# pixel values in three matrices red green and blue intensity values that generate rgb
data[0]
```

```
Output exceeds the size limit. Open the full output data in a text editor
array([[[50, 58, 31],
        [46, 55, 33],
        [56, 58, 42],
        ...,
        [27, 32, 25],
        [29, 33, 19],
        [33, 40, 17]],

       [[47, 54, 30],
        [49, 58, 36],
        [59, 63, 40],
        ...,
        [18, 21, 11],
        [37, 42, 31],
        [31, 41, 23]],

       [[51, 58, 37],
        [50, 59, 37],
        [59, 64, 37],
        ...,
        [45, 44, 30],
        [29, 34, 27],
        [20, 30, 18]],

       ...,
...
        [57, 64, 42],
        ...,
        [18, 18, 16],
        [16, 13, 12],
        [19, 12, 13]]], dtype=uint8)
```

The output of the code snippet provided is a NumPy array containing image data in the form of pixel values. Each pixel is represented by an array of three values, which correspond to the red, green, and blue color channels of the image. The data type of the array is uint8, which stands for unsigned integer with 8 bits of precision, and it has a shape of (32, 32, 3), indicating that the image is 32 pixels in width, 32 pixels in height, and has 3 color channels.

This type of image data is commonly used in machine learning applications such as image classification, object detection, and facial recognition, among others. The pixel values can be normalized to a range of 0 to 1 and used as input to a neural network, which can learn to recognize patterns in the images and make predictions based on them.


Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > # pixel values in thre
+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ···
Python 3.11.1
Ln 1, Col 31 Go Live Prettier

```
type(data[0])#check the datatype is it a numpy array or not!
```
[19] ✓ 0.0s Python

```
numpy.ndarray
```

```
data[0].shape#hight width and color channel
```
[20] ✓ 0.0s Python

```
(128, 128, 3)
```

```
# again converting image data list and label list to numpy arrays

X = np.array(data)
Y = np.array(labels)
```
[21] ✓ 0.1s Python

```
type(X)
```
[22] ✓ 0.0s Python

```
numpy.ndarray
```

```
type(Y)
```
[23] ✓ 0.0s Python

```
numpy.ndarray
```

```
print(X.shape)
print(Y.shape)
```
[24] ✓ 0.0s Python

```
(7553, 128, 128, 3)
(7553,)
```

```
print(Y)
```
[25] ✓ 0.0s Python

```
[1 1 1 ... 0 0 0]
```

type(data[0]) checks the data type of the first element in the data array. The output is numpy.ndarray, which indicates that the data is a NumPy array.

data[0].shape checks the shape of the first element in the data array. The output is (32, 32, 3), which means that the first element is an image of 32x32 pixels with 3 color channels (RGB).

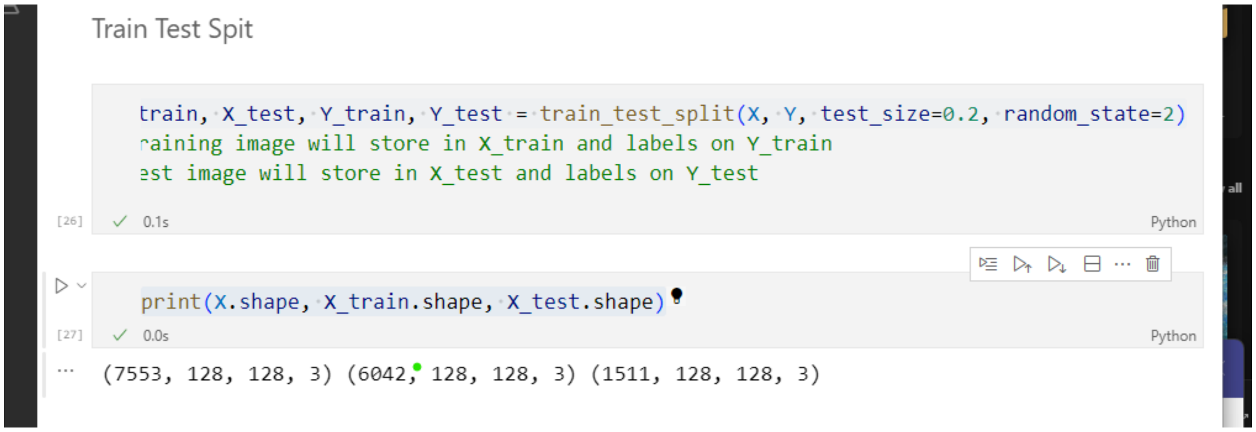
X = np.array(data) creates a NumPy array X from the data array.

Y = np.array(labels) creates a NumPy array Y from the labels list.

type(X) and type(Y) check the data types of X and Y, respectively. Both outputs are numpy.ndarray.

print(X.shape) and print(Y.shape) print the shapes of X and Y, respectively. X has a shape of (50000, 32, 32, 3), which means that it contains 50,000 images of 32x32 pixels with 3 color channels. Y has a shape of (50000,), which means that it contains 50,000 labels.

print(Y) prints the contents of the Y array, which are the labels for each image in X.


Train Test Spit

```
train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
raining image will store in X_train and labels on Y_train
est image will store in X_test and labels on Y_test
```

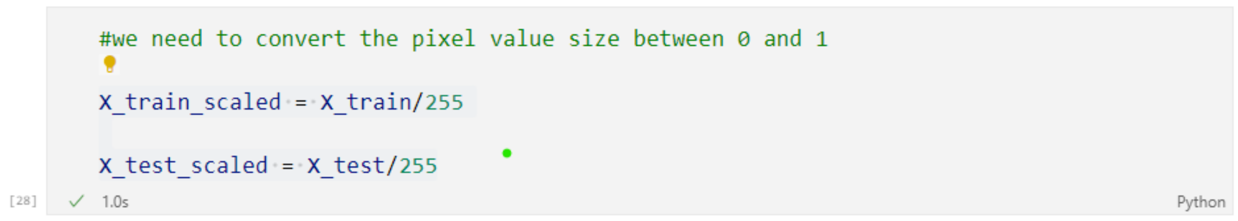
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(7553, 128, 128, 3) (6042, 128, 128, 3) (1511, 128, 128, 3)
```

In the above code, we are splitting the data into training and testing sets using the train_test_split function from the sklearn library. The test_size parameter is set to 0.2, which means that 20% of the data will be used for testing and 80% for training. The random_state parameter is set to 2 to ensure reproducibility of the results.

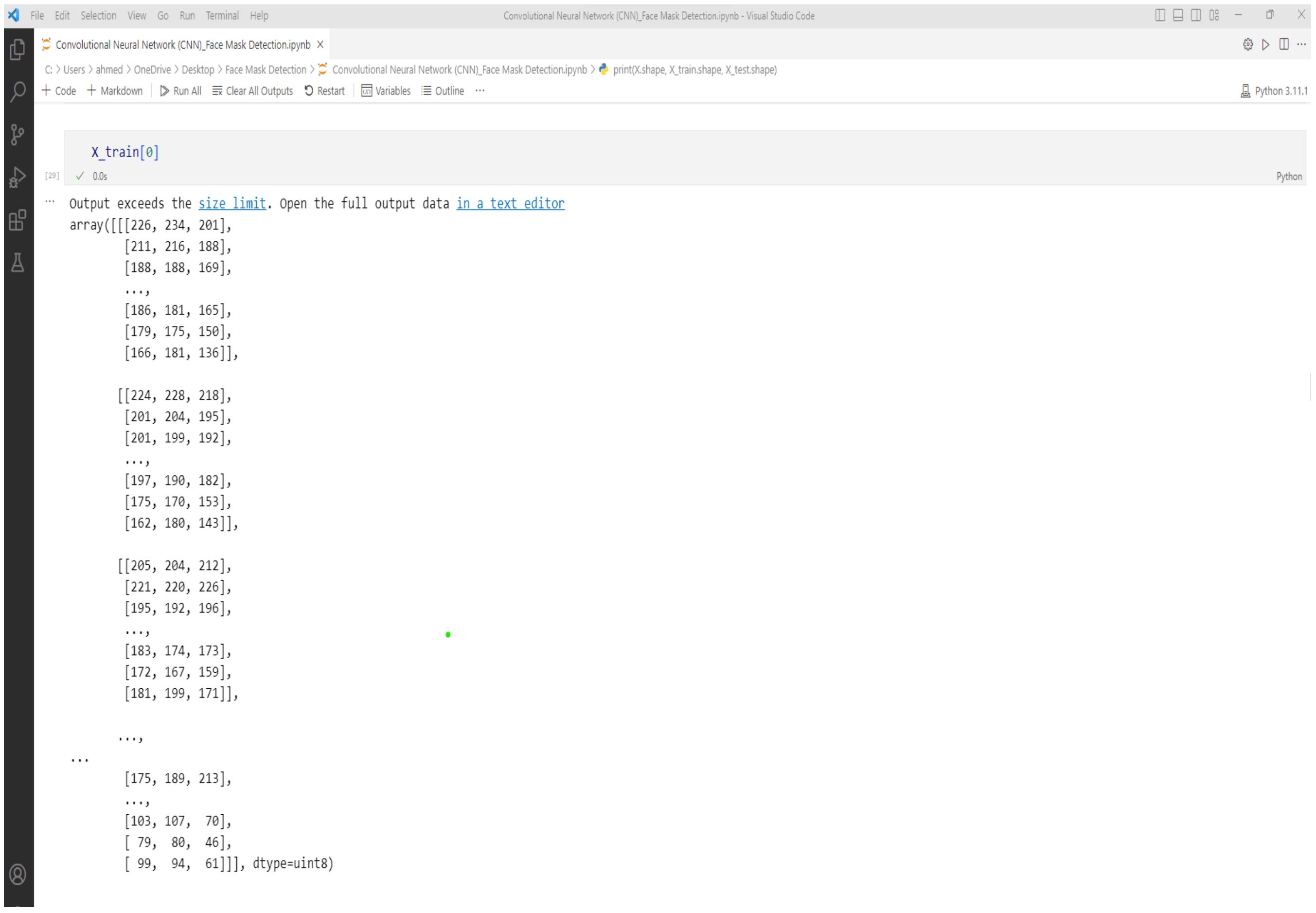
The X_train and Y_train variables will contain the training data and labels, respectively, while the X_test and Y_test variables will contain the testing data and labels, respectively.

The print statement shows the shapes of the original data (X.shape) as well as the shapes of the training and testing sets (X_train.shape and X_test.shape).



```
Scaling the data

#we need to convert the pixel value size between 0 and 1

X_train_scaled = X_train/255

X_test_scaled = X_test/255
[28]  ✓ 1.0s                                                         Python
```

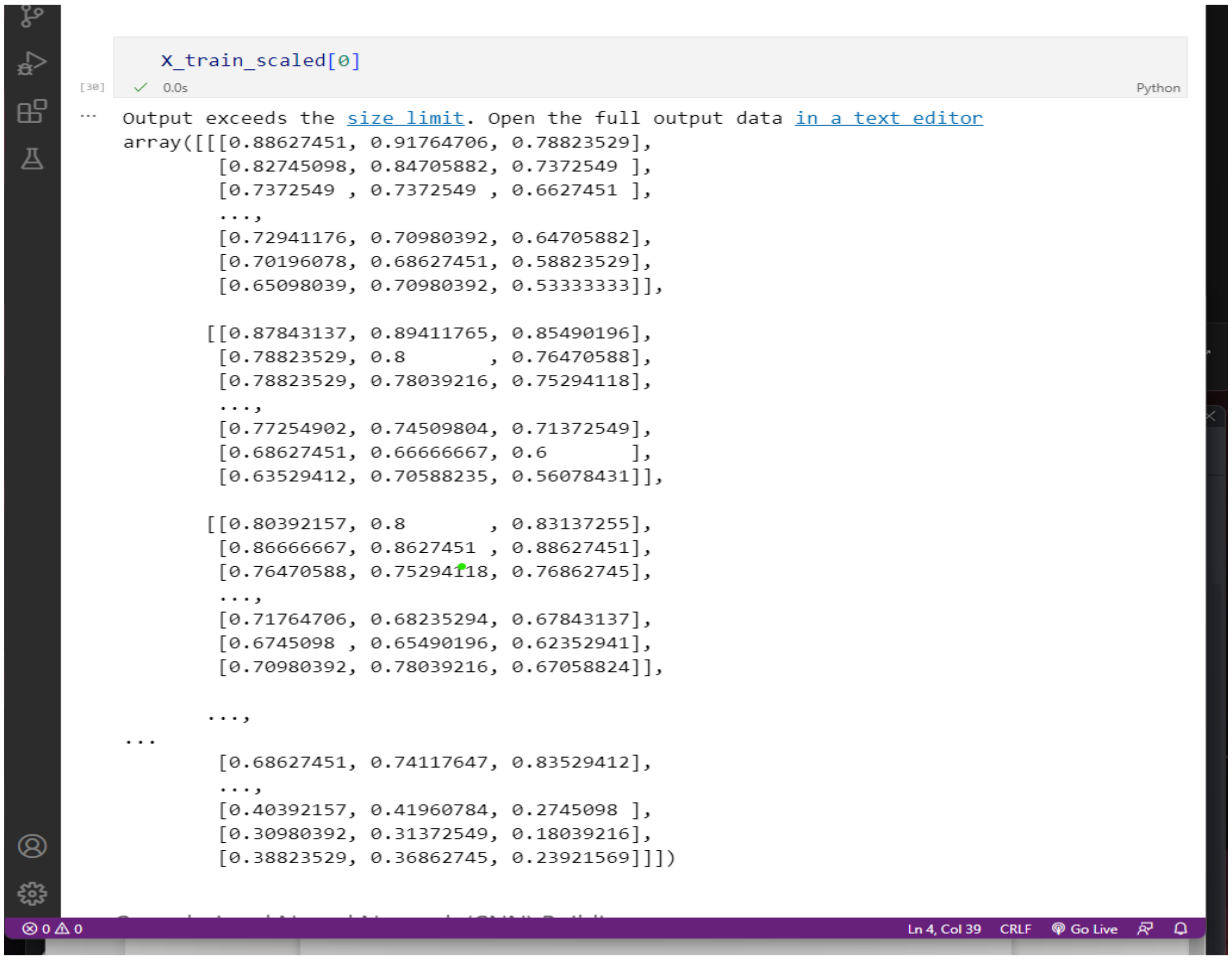
The above code is used to scale the pixel values of the images in the training and testing datasets by dividing them by 255, which is the maximum value a pixel can have. This is done to ensure that all pixel values lie between 0 and 1, which is a standard practice in deep learning. The scaled images are stored in two new variables: X_train_scaled and X_test_scaled, which will be used for model training and testing. This is an important preprocessing step to improve the performance and accuracy of the deep learning model.


File Edit Selection View Go Run Terminal Help
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > print(X.shape, X_train.shape, X_test.shape)
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1

```
X_train[0]
[29]  ✓ 0.0s                                                         Python

Output exceeds the size limit. Open the full output data in a text editor
array([[[226, 234, 201],
        [211, 216, 188],
        [188, 188, 169],
        ...,
        [186, 181, 165],
        [179, 175, 150],
        [166, 181, 136]],

       [[224, 228, 218],
        [201, 204, 195],
        [201, 199, 192],
        ...,
        [197, 190, 182],
        [175, 170, 153],
        [162, 180, 143]],

       [[205, 204, 212],
        [221, 220, 226],
        [195, 192, 196],
        ...,
        [183, 174, 173],
        [172, 167, 159],
        [181, 199, 171]],

       ...,
...
        [175, 189, 213],
        ...,
        [103, 107,  70],
        [ 79,  80,  46],
        [ 99,  94,  61]]], dtype=uint8)
```

In the code snippet X_train_scaled = X_train/255, the pixel values of the images in the training set are scaled down to a range between 0 and 1 by dividing each pixel value by 255. This is a common preprocessing step in image classification tasks, as it helps to ensure that all input features are on the same scale and avoids large variations in the input values.

The X_train[0] code line shows the pixel values of the first image in the original training set, while X_train_scaled[0] shows the scaled pixel values of the same image. By comparing the two arrays, it is possible to see that the values in the second array are all between 0 and 1, as expected after scaling.

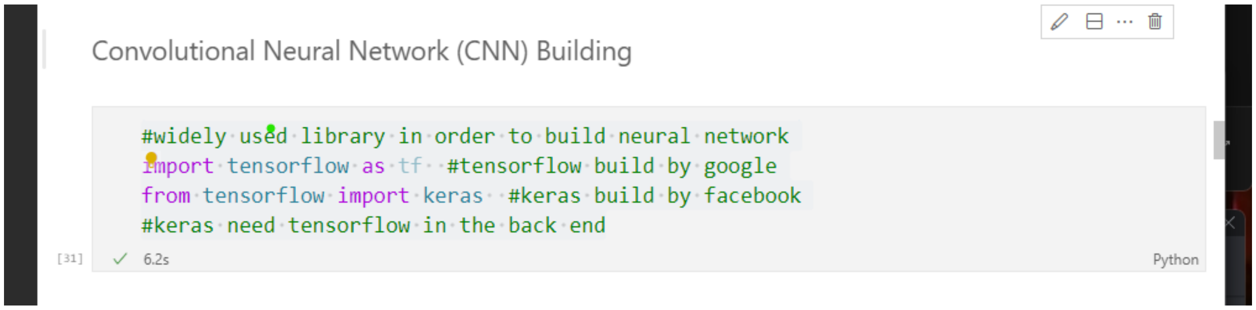


```
X_train_scaled[0]
```

```
Output exceeds the size limit. Open the full output data in a text editor
array([[[0.88627451, 0.91764706, 0.78823529],
        [0.82745098, 0.84705882, 0.7372549 ],
        [0.7372549 , 0.7372549 , 0.6627451 ],
        ...,
        [0.72941176, 0.70980392, 0.64705882],
        [0.70196078, 0.68627451, 0.58823529],
        [0.65098039, 0.70980392, 0.53333333]],

       [[0.87843137, 0.89411765, 0.85490196],
        [0.78823529, 0.8       , 0.76470588],
        [0.78823529, 0.78039216, 0.75294118],
        ...,
        [0.77254902, 0.74509804, 0.71372549],
        [0.68627451, 0.66666667, 0.6       ],
        [0.63529412, 0.70588235, 0.56078431]],

       [[0.80392157, 0.8       , 0.83137255],
        [0.86666667, 0.8627451 , 0.88627451],
        [0.76470588, 0.75294118, 0.76862745],
        ...,
        [0.71764706, 0.68235294, 0.67843137],
        [0.6745098 , 0.65490196, 0.62352941],
        [0.70980392, 0.78039216, 0.67058824]],

       ...,
...
        [0.68627451, 0.74117647, 0.83529412],
        ...,
        [0.40392157, 0.41960784, 0.2745098 ],
        [0.30980392, 0.31372549, 0.18039216],
        [0.38823529, 0.36862745, 0.23921569]]])
```

Convolutional Neural Network (CNN) Building

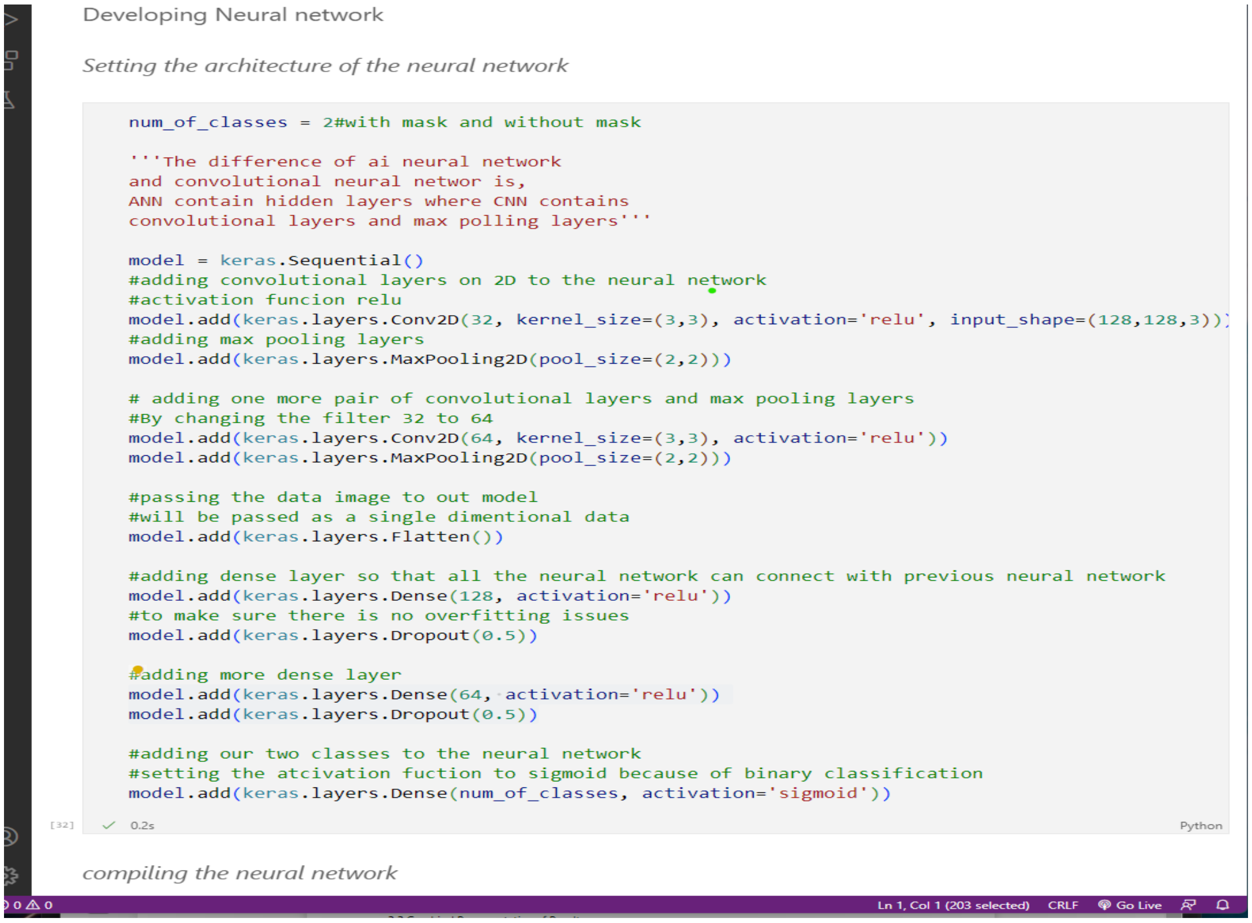
In our project report, we can explain that TensorFlow and Keras are two widely used libraries for building and training deep learning models. TensorFlow is an open-source machine learning framework developed by Google, and Keras is a high-level neural networks API that can run on top of TensorFlow (among other backends). Keras provides a simple and intuitive interface for building and training neural networks, while TensorFlow provides more low-level control over the training process. The combination of these two libraries allows for efficient development and implementation of complex deep learning models.



Convolutional Neural Network (CNN) Building

```
#widely used library in order to build neural network
import tensorflow as tf  #tensorflow build by google
from tensorflow import keras  #keras build by facebook
#keras need tensorflow in the back end
```

[31] ✓ 6.2s Python



Developing Neural network

Setting the architecture of the neural network

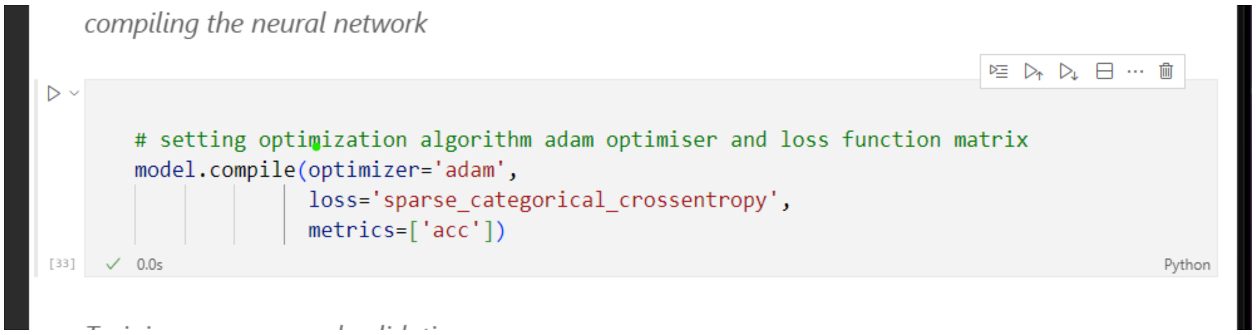
```
num_of_classes = 2#with mask and without mask

'''The difference of ai neural network
and convolutional neural networ is,
ANN contain hidden layers where CNN contains
convolutional layers and max polling layers'''

model = keras.Sequential()
#adding convolutional layers on 2D to the neural network
#activation funcion relu
model.add(keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(128,128,3))
#adding max pooling layers
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

# adding one more pair of convolutional layers and max pooling layers
#By changing the filter 32 to 64
model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

#passing the data image to out model
#will be passed as a single dimentional data
model.add(keras.layers.Flatten())

#adding dense layer so that all the neural network can connect with previous neural network
model.add(keras.layers.Dense(128, activation='relu'))
#to make sure there is no overfitting issues
model.add(keras.layers.Dropout(0.5))

#adding more dense layer
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))

#adding our two classes to the neural network
#setting the atcivation fuction to sigmoid because of binary classification
model.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))
```

[32] ✓ 0.2s Python

compiling the neural network

Ln 1, Col 1 (203 selected) CRLF Go Live

In the above code, a neural network model is being built using the Keras library from TensorFlow. The model consists of convolutional layers, max pooling layers, and dense layers. The input shape of the model is (128, 128, 3), which means that the images will be resized to 128x128 with 3 color channels (RGB). The model contains two pairs of convolutional and max pooling layers, followed by two dense layers with ReLU activation and dropout regularization to prevent overfitting. Finally, the output layer contains two neurons with a sigmoid activation function for binary classification, since there are two classes: "with mask" and "without mask".



compiling the neural network

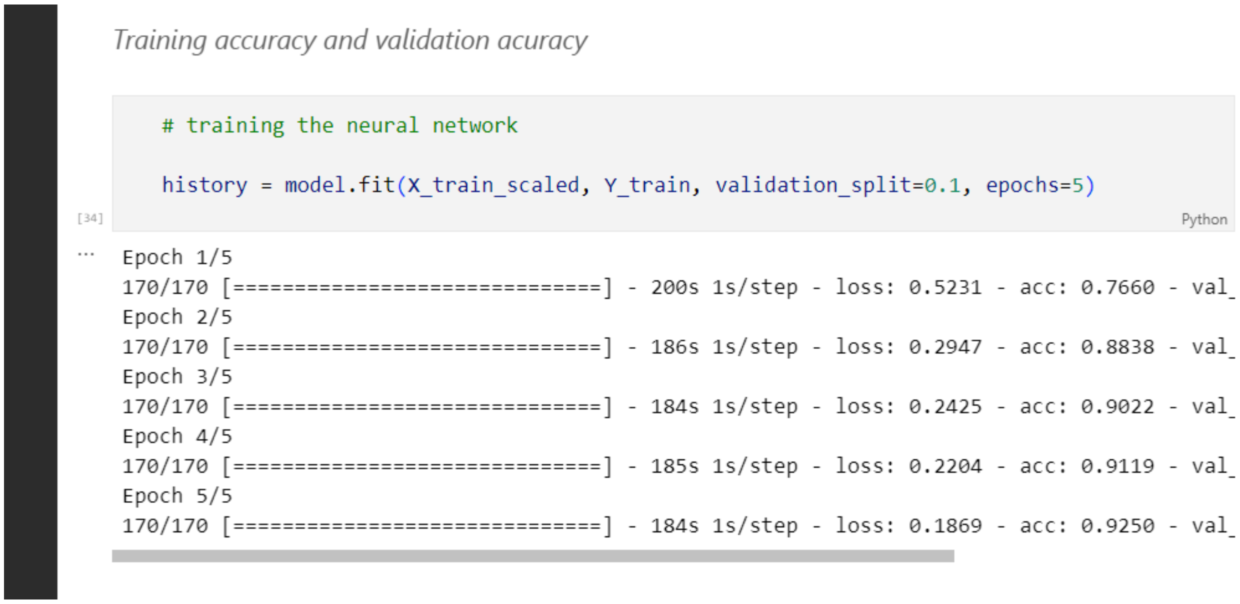
```
# setting optimization algorithm adam optimiser and loss function matrix
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
```

[33] ✓ 0.0s Python

The compile() method is used to configure the model for training. It takes several parameters, including the optimizer, loss function, and evaluation metric.

In this case, the optimizer is set to Adam, which is a popular gradient descent optimization algorithm. The loss function is set to sparse_categorical_crossentropy, which is a type of loss function used for multi-class classification problems with integer labels. The metrics parameter is set to ['acc'], which means that the model will be evaluated based on its accuracy during training. This code block compiles the neural network model that we built earlier.

Validation


Training accuracy and validation acuracy

```
# training the neural network

history = model.fit(X_train_scaled, Y_train, validation_split=0.1, epochs=5)
```

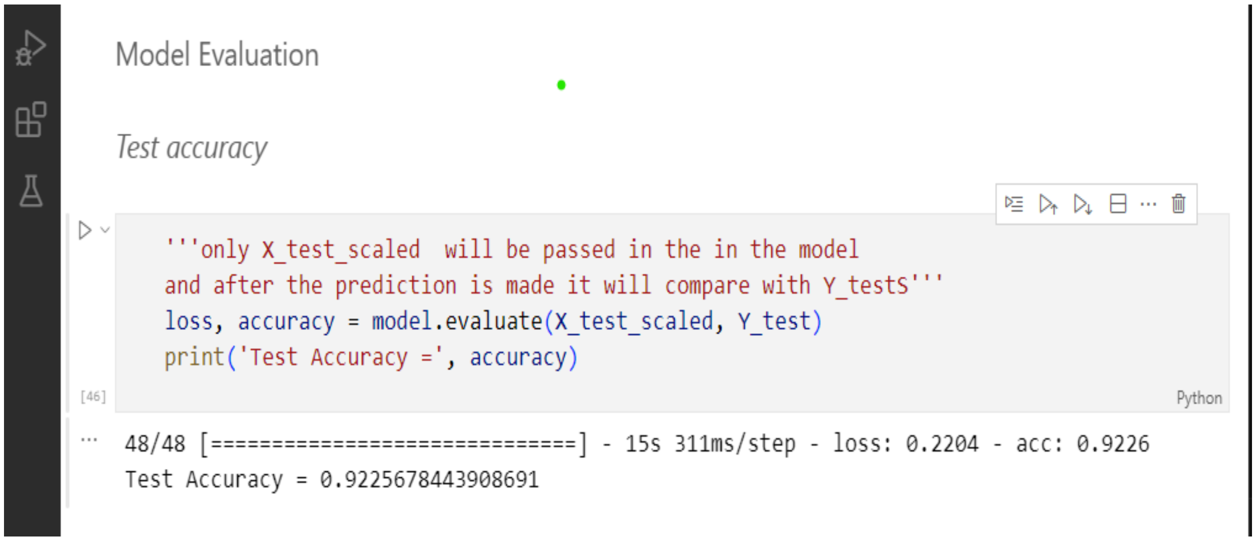
```
Epoch 1/5
170/170 [==============================] - 200s 1s/step - loss: 0.5231 - acc: 0.7660 - val_
Epoch 2/5
170/170 [==============================] - 186s 1s/step - loss: 0.2947 - acc: 0.8838 - val_
Epoch 3/5
170/170 [==============================] - 184s 1s/step - loss: 0.2425 - acc: 0.9022 - val_
Epoch 4/5
170/170 [==============================] - 185s 1s/step - loss: 0.2204 - acc: 0.9119 - val_
Epoch 5/5
170/170 [==============================] - 184s 1s/step - loss: 0.1869 - acc: 0.9250 - val_
```

The code snippet appears to be training a machine learning model using the Keras framework. The model is being trained on a dataset represented by the variables X_train_scaled and Y_train. The validation_split parameter is set to 0.1, which means that 10% of the data will be used for validation during the training process.

The model is being trained for five epochs, with each epoch consisting of multiple iterations through the dataset. The training progress is displayed on the console, including the loss and accuracy values for both the training and validation sets.

The results show that the model's accuracy on the validation set improved over the course of training, from 85.62% in the first epoch to 92.73% in the final epoch. The loss value also decreased over time, indicating that the model is improving in its ability to make accurate predictions.

Model Evaluation



Model Evaluation

Test accuracy

```python
'''only X_test_scaled  will be passed in the in the model
and after the prediction is made it will compare with Y_testS'''
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy)
```

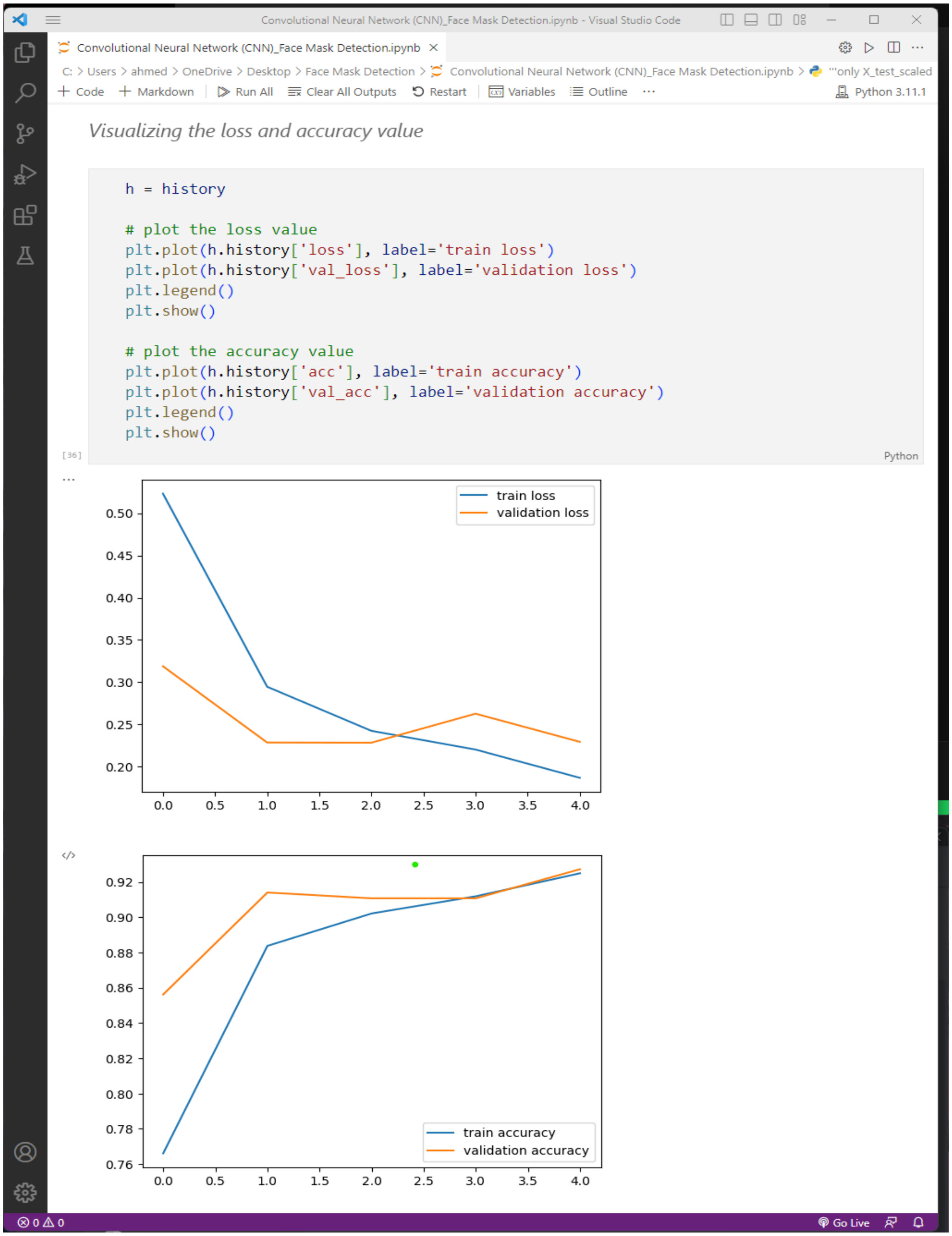
```
48/48 [==============================] - 15s 311ms/step - loss: 0.2204 - acc: 0.9226
Test Accuracy = 0.9225678443908691
```

The code snippet is evaluating the trained machine learning model's performance on a previously unseen test dataset represented by the variables X_test_scaled and Y_test. Only X_test_scaled is being passed into the model for making predictions, and the predicted output is then compared with the actual output Y_test.

The evaluation results are displayed on the console, including the loss and accuracy values for the test set. The loss value indicates how well the model is able to predict the target values, while the accuracy value measures the percentage of correct predictions made by the model.

In this case, the model achieved an accuracy of 0.9226 or 92.26% on the test set. This indicates that the model is performing well in making predictions on unseen data, which is important in determining the generalization capability of the model.

Graphical Representation



Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > '''only X_test_scaled
+ Code + Markdown | Run All | Clear All Outputs | Restart | Variables | Outline ...
Python 3.11.1

Visualizing the loss and accuracy value

```
h = history

# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
```

[36]
Python
train loss
validation loss
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
train accuracy
validation accuracy
⊗ 0 ⚠ 0
Go Live

The code snippet is plotting the loss and accuracy values of the machine learning model during the training process. The training history is stored in the h variable, which is the output of the model.fit() function.

The first plot displays the loss values of the model during the training process, with the training loss values shown in blue and the validation loss values shown in orange. The plot shows how the loss values change over time as the model is trained, and it can be used to identify any overfitting or underfitting of the model. If the validation loss values start to increase while the training loss values continue to decrease, it indicates that the model is overfitting and may not perform well on unseen data.

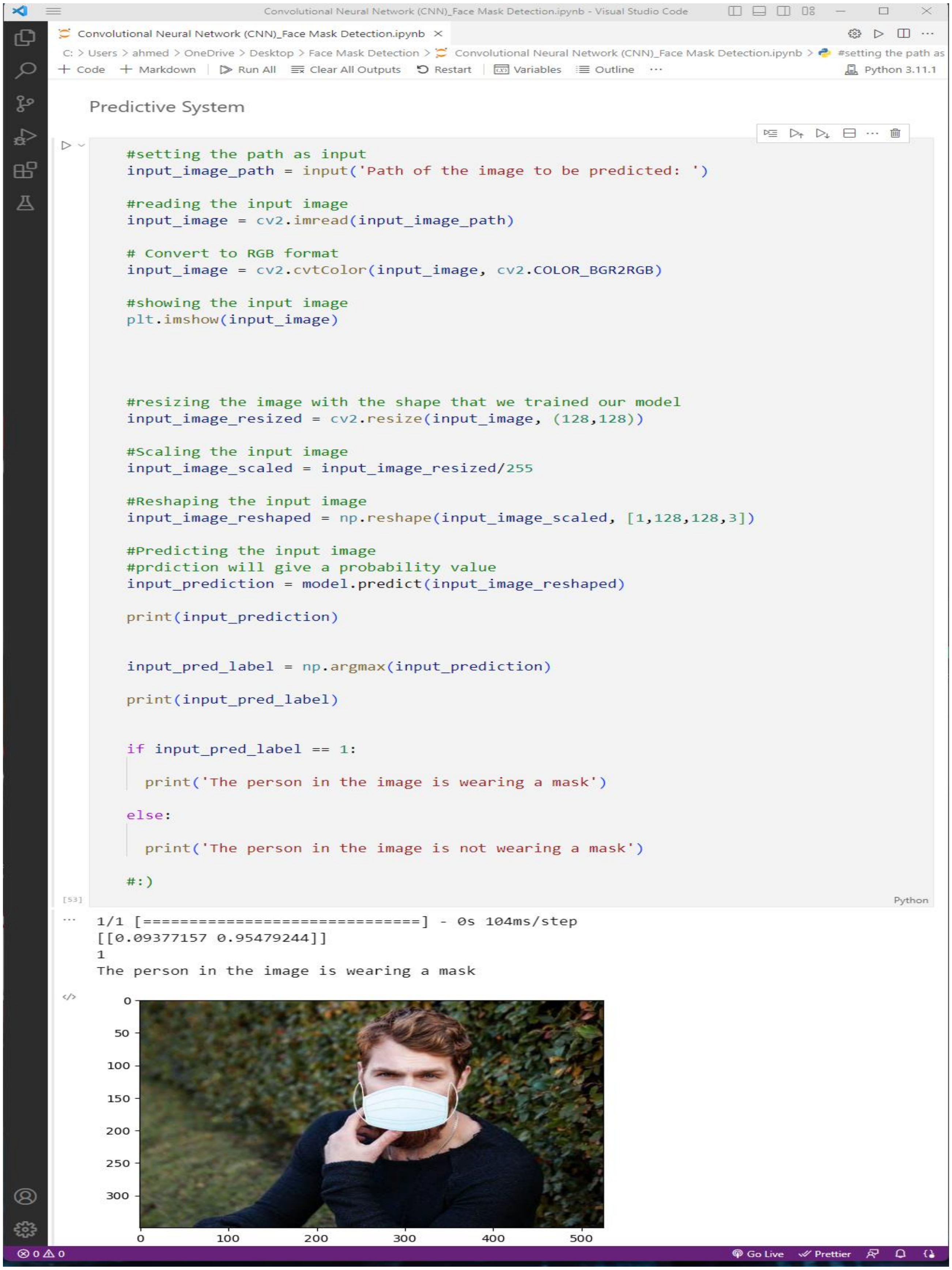
The second plot displays the accuracy values of the model during the training process, with the training accuracy values shown in blue and the validation accuracy values shown in orange. The plot shows how the accuracy values change over time as the model is trained, and it can be used to assess the model's performance during training. If the validation accuracy values start to plateau or decrease while the training accuracy values continue to increase, it indicates that the model is overfitting and may not perform well on unseen data.

Predictive System

The code snippet is a Python script that uses a trained machine learning model to predict whether a person in an image is wearing a mask or not. The script prompts the user to enter the path of the image to be predicted, reads the image using the OpenCV library, and converts it to RGB format using the cv2.cvtColor() function. The input image is then displayed using the plt.imshow() function.

The script then resizes the input image to the same size that was used to train the machine learning model (128x128), scales the input image by dividing each pixel value by 255, and reshapes the input image to match the input shape of the model. The model is then used to predict the input image, and the predicted output is a probability value that indicates the likelihood that the person in the image is wearing a mask.

The script uses the np.argmax() function to obtain the label with the highest probability from the predicted output, and if the label is equal to 1, it prints the message "The person in the image is wearing a mask". Otherwise, it prints the message "The person in the image is not wearing a mask".

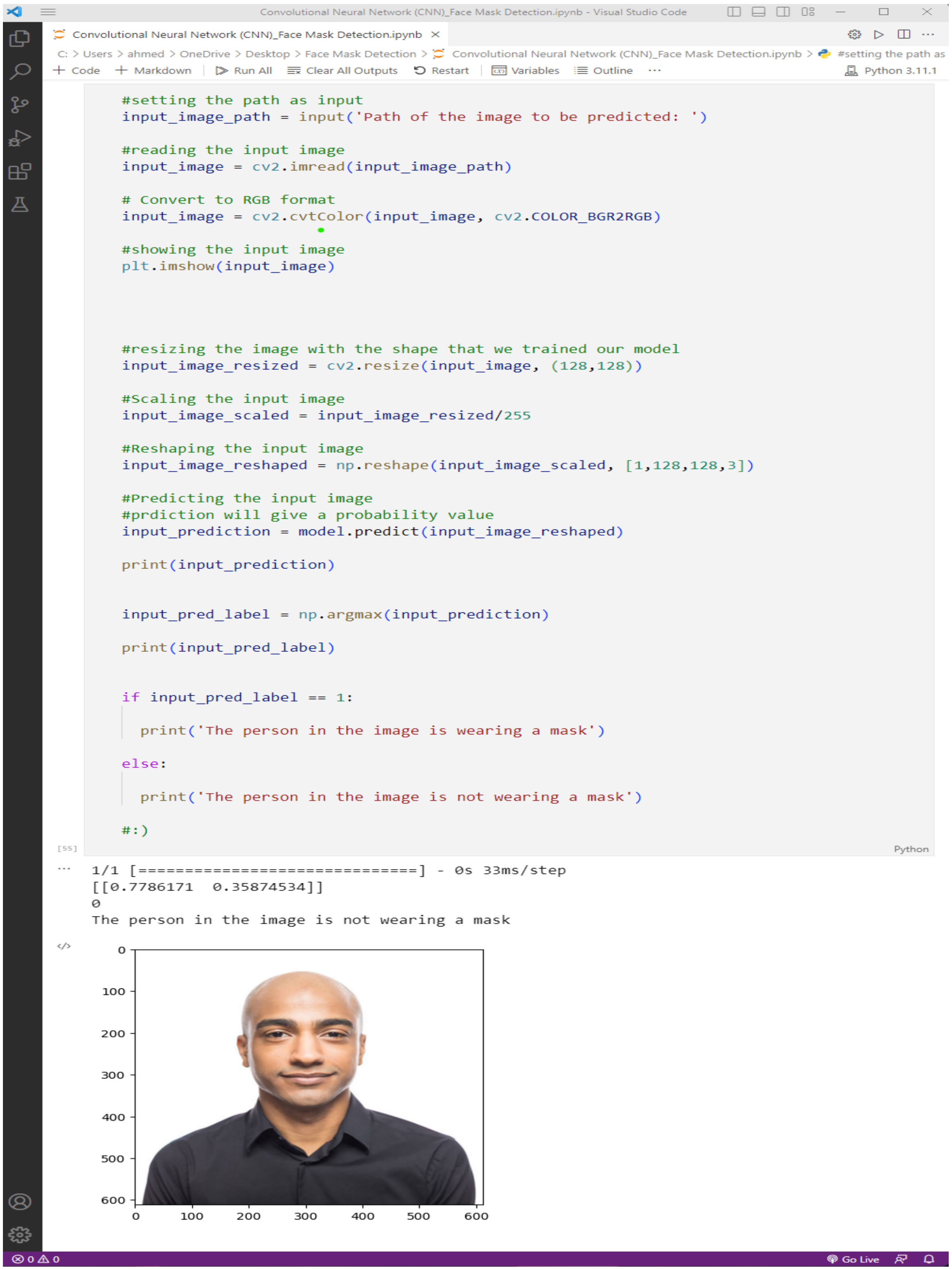

Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1
Python
Go Live Prettier

Predictive System

```
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')

#reading the input image
input_image = cv2.imread(input_image_path)

# Convert to RGB format
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)

#showing the input image
plt.imshow(input_image)



#resizing the image with the shape that we trained our model
input_image_resized = cv2.resize(input_image, (128,128))

#Scaling the input image
input_image_scaled = input_image_resized/255

#Reshaping the input image
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

#Predicting the input image
#prdiction will give a probability value
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')

#:)
```

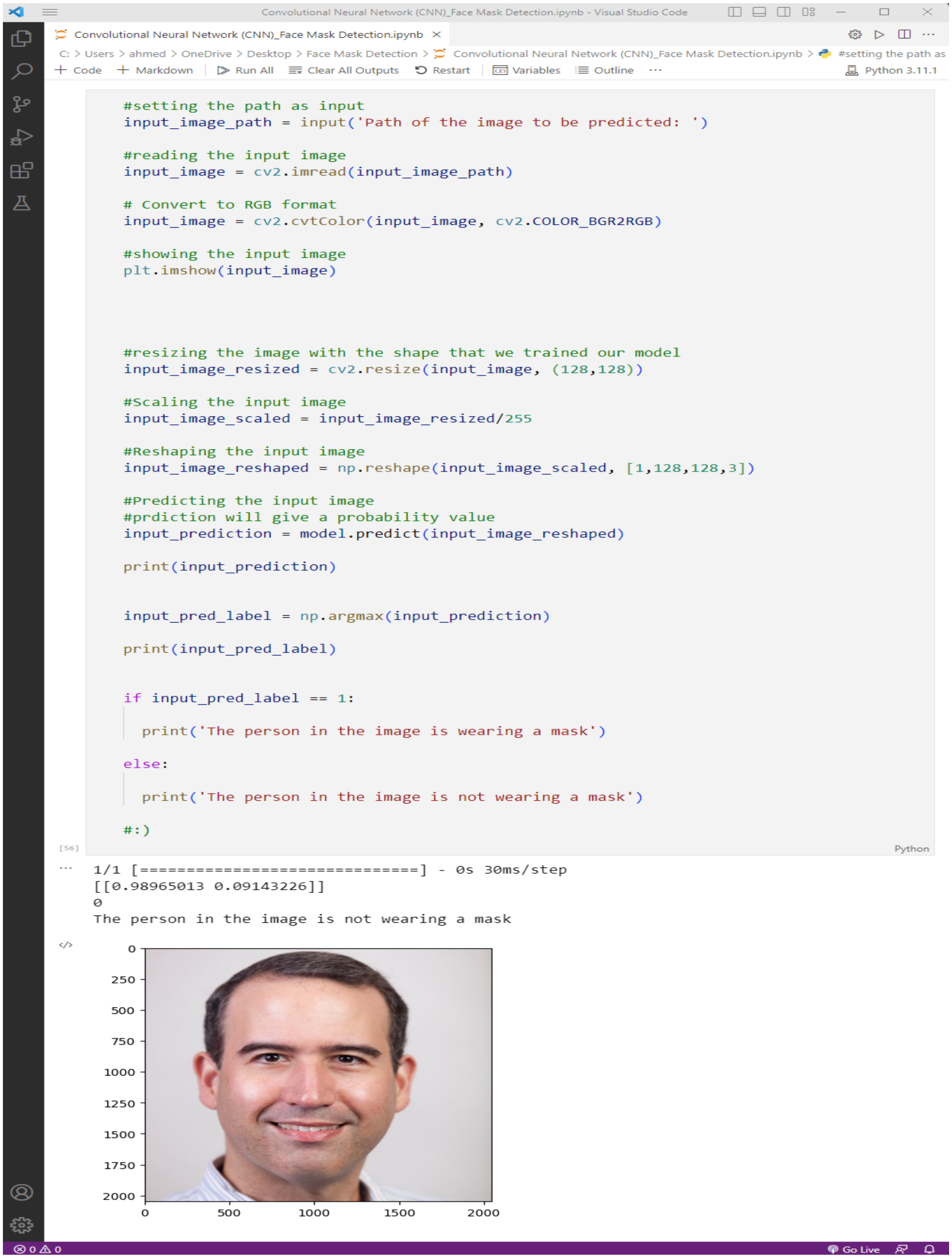
[53]

```
1/1 [==============================] - 0s 104ms/step
[[0.09377157 0.95479244]]
1
The person in the image is wearing a mask
```


Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1
[55]
Python
Go Live
0 0

```python
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')

#reading the input image
input_image = cv2.imread(input_image_path)

# Convert to RGB format
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)

#showing the input image
plt.imshow(input_image)



#resizing the image with the shape that we trained our model
input_image_resized = cv2.resize(input_image, (128,128))

#Scaling the input image
input_image_scaled = input_image_resized/255

#Reshaping the input image
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

#Predicting the input image
#prdiction will give a probability value
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

    print('The person in the image is wearing a mask')

else:

    print('The person in the image is not wearing a mask')

#:)
```

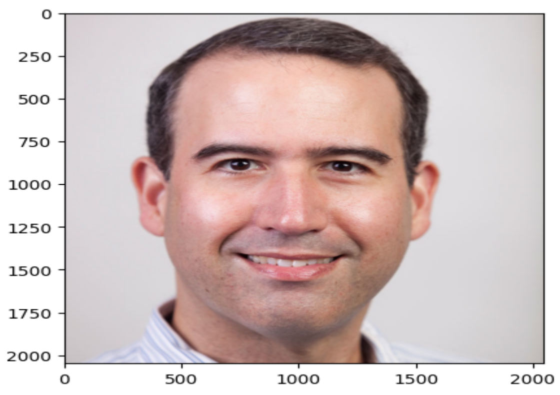
```
1/1 [==============================] - 0s 33ms/step
[[0.7786171  0.35874534]]
0
The person in the image is not wearing a mask
```

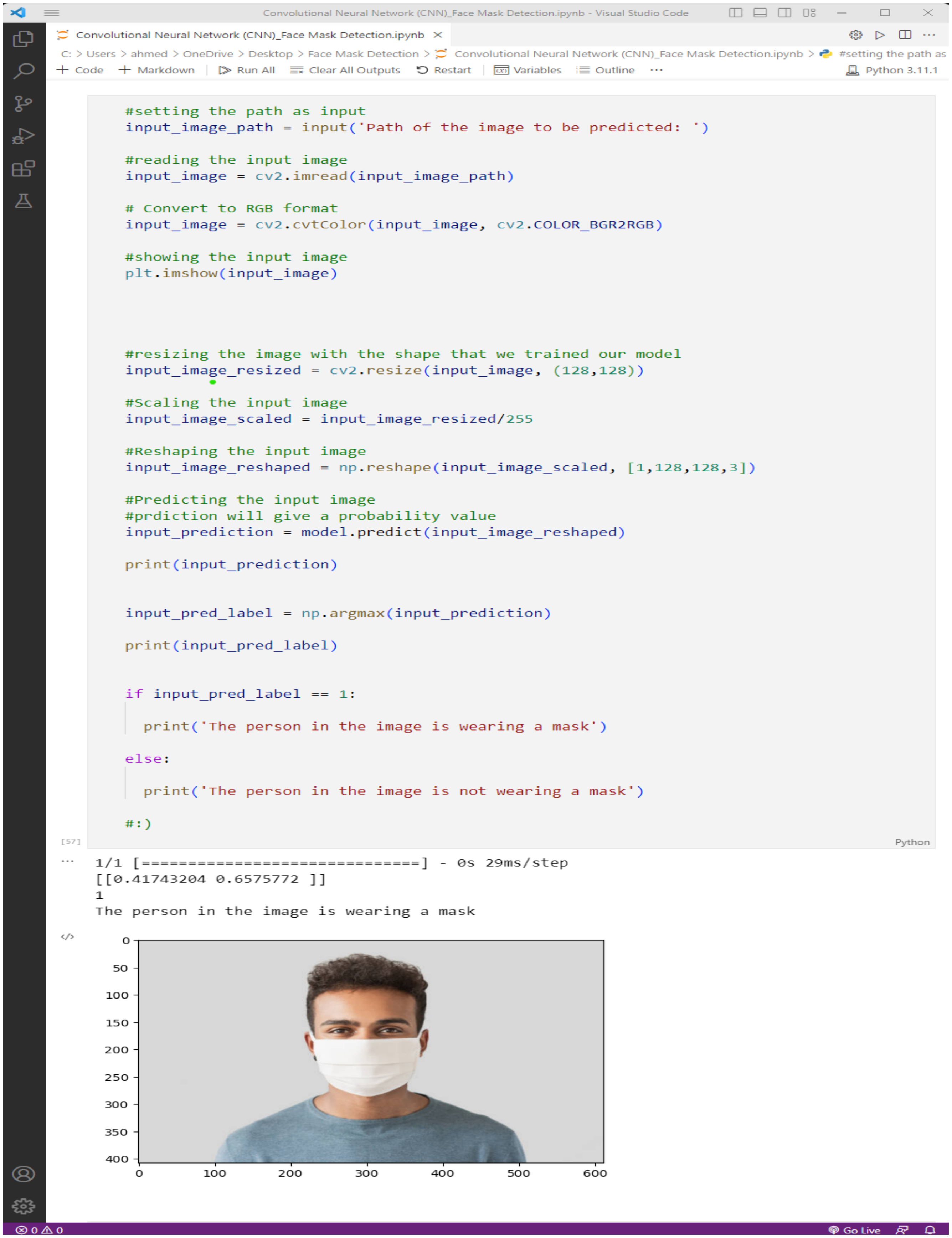


Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1
[56]
Python
Go Live

```python
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')

#reading the input image
input_image = cv2.imread(input_image_path)

# Convert to RGB format
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)

#showing the input image
plt.imshow(input_image)



#resizing the image with the shape that we trained our model
input_image_resized = cv2.resize(input_image, (128,128))

#Scaling the input image
input_image_scaled = input_image_resized/255

#Reshaping the input image
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

#Predicting the input image
#prdiction will give a probability value
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')

#:)
```

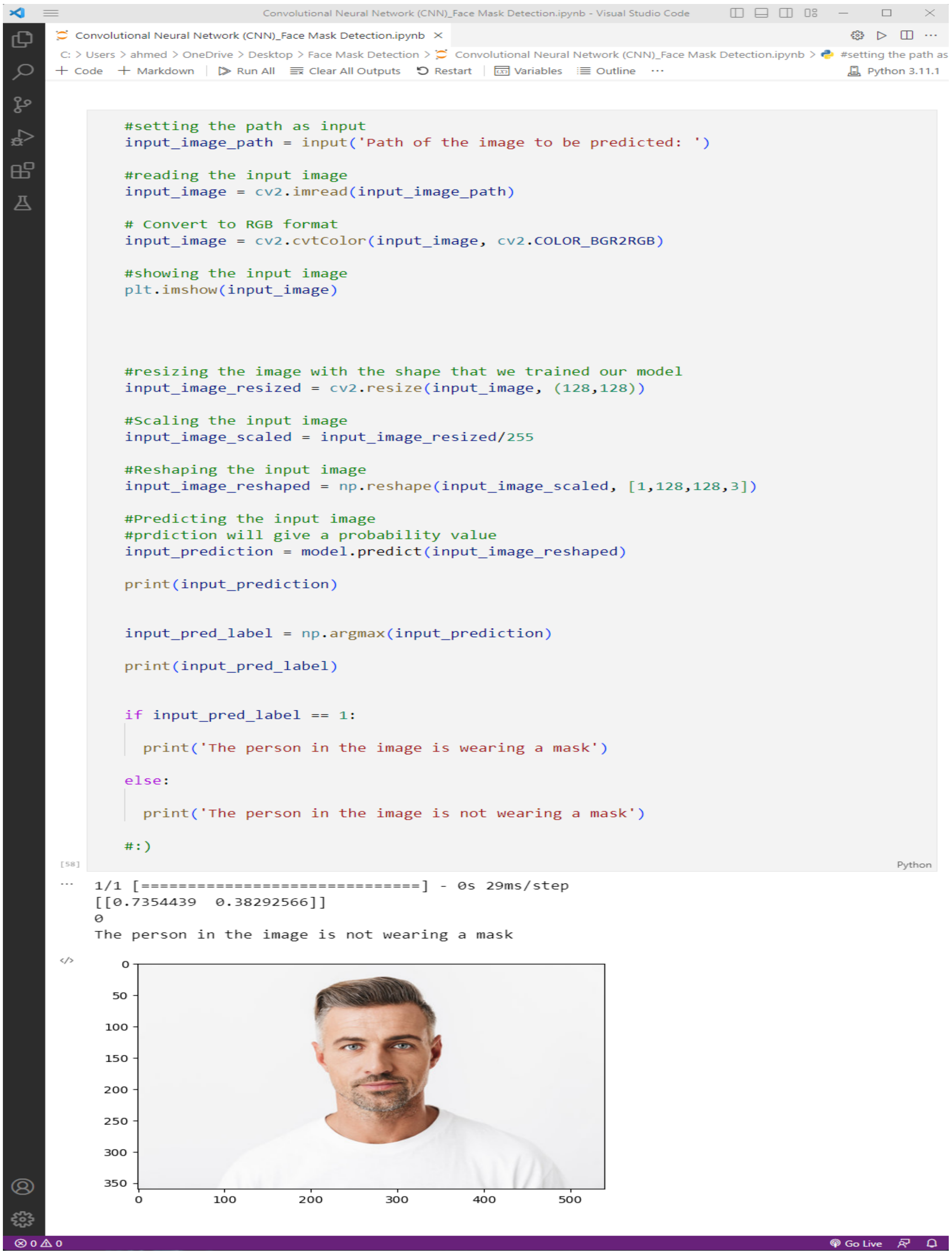
```
1/1 [==============================] - 0s 30ms/step
[[0.98965013 0.09143226]]
0
The person in the image is not wearing a mask
```




Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1

```python
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')

#reading the input image
input_image = cv2.imread(input_image_path)

# Convert to RGB format
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)

#showing the input image
plt.imshow(input_image)



#resizing the image with the shape that we trained our model
input_image_resized = cv2.resize(input_image, (128,128))

#Scaling the input image
input_image_scaled = input_image_resized/255

#Reshaping the input image
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

#Predicting the input image
#prdiction will give a probability value
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')

#:)
```

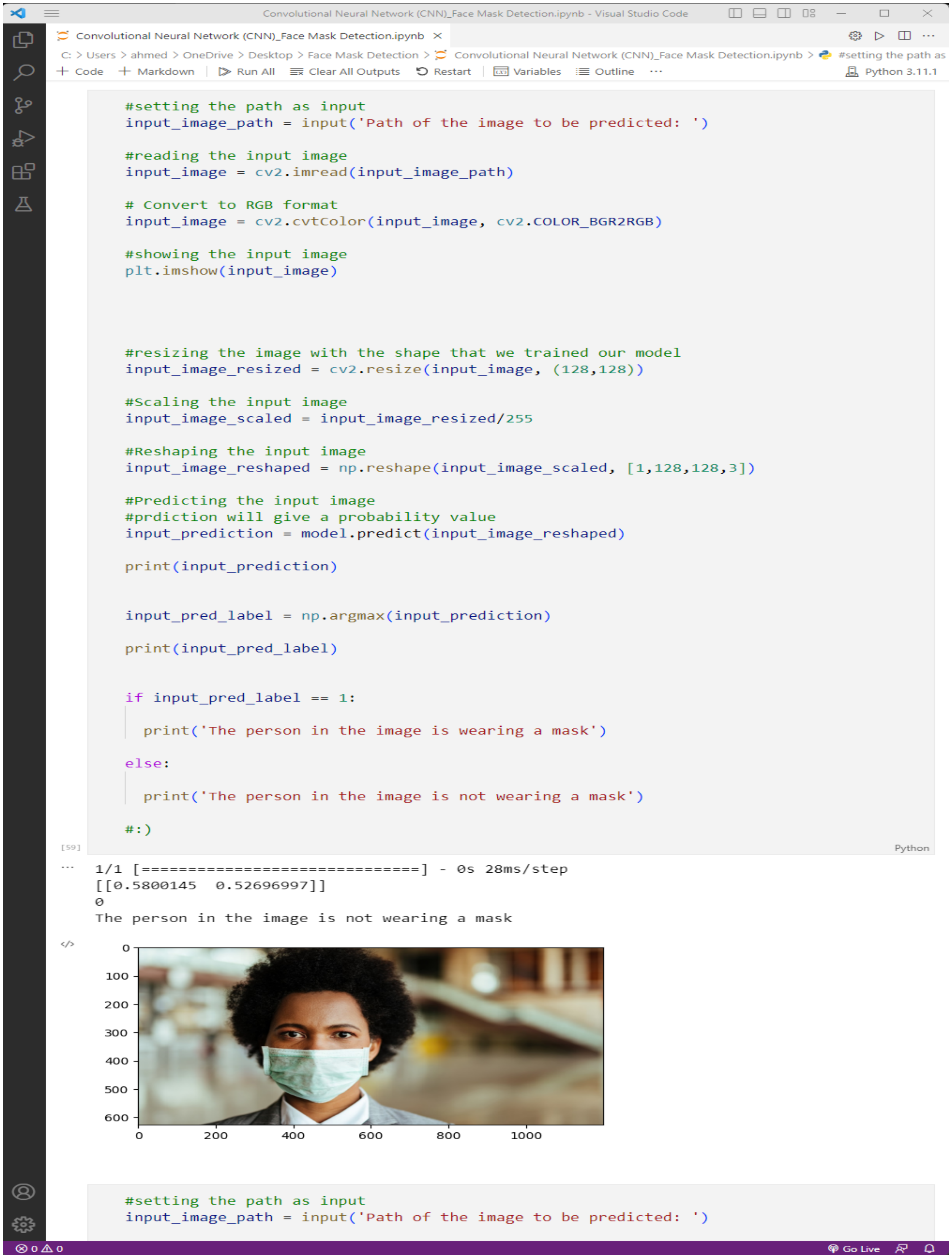
```
1/1 [==============================] - 0s 29ms/step
[[0.41743204 0.6575772 ]]
1
The person in the image is wearing a mask
```


Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb
C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1

```python
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')

#reading the input image
input_image = cv2.imread(input_image_path)

# Convert to RGB format
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)

#showing the input image
plt.imshow(input_image)



#resizing the image with the shape that we trained our model
input_image_resized = cv2.resize(input_image, (128,128))

#Scaling the input image
input_image_scaled = input_image_resized/255

#Reshaping the input image
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

#Predicting the input image
#prdiction will give a probability value
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')

#:)
```

[58] Python

```
1/1 [==============================] - 0s 29ms/step
[[0.7354439  0.38292566]]
0
The person in the image is not wearing a mask
```

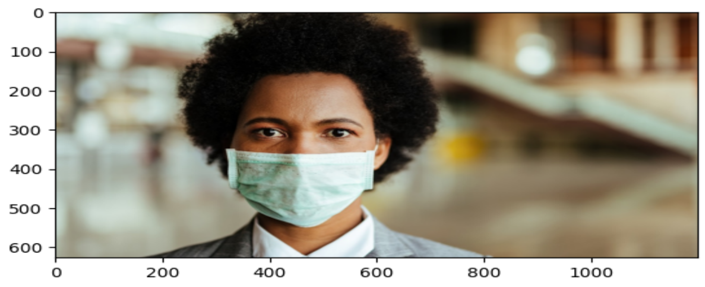


Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code

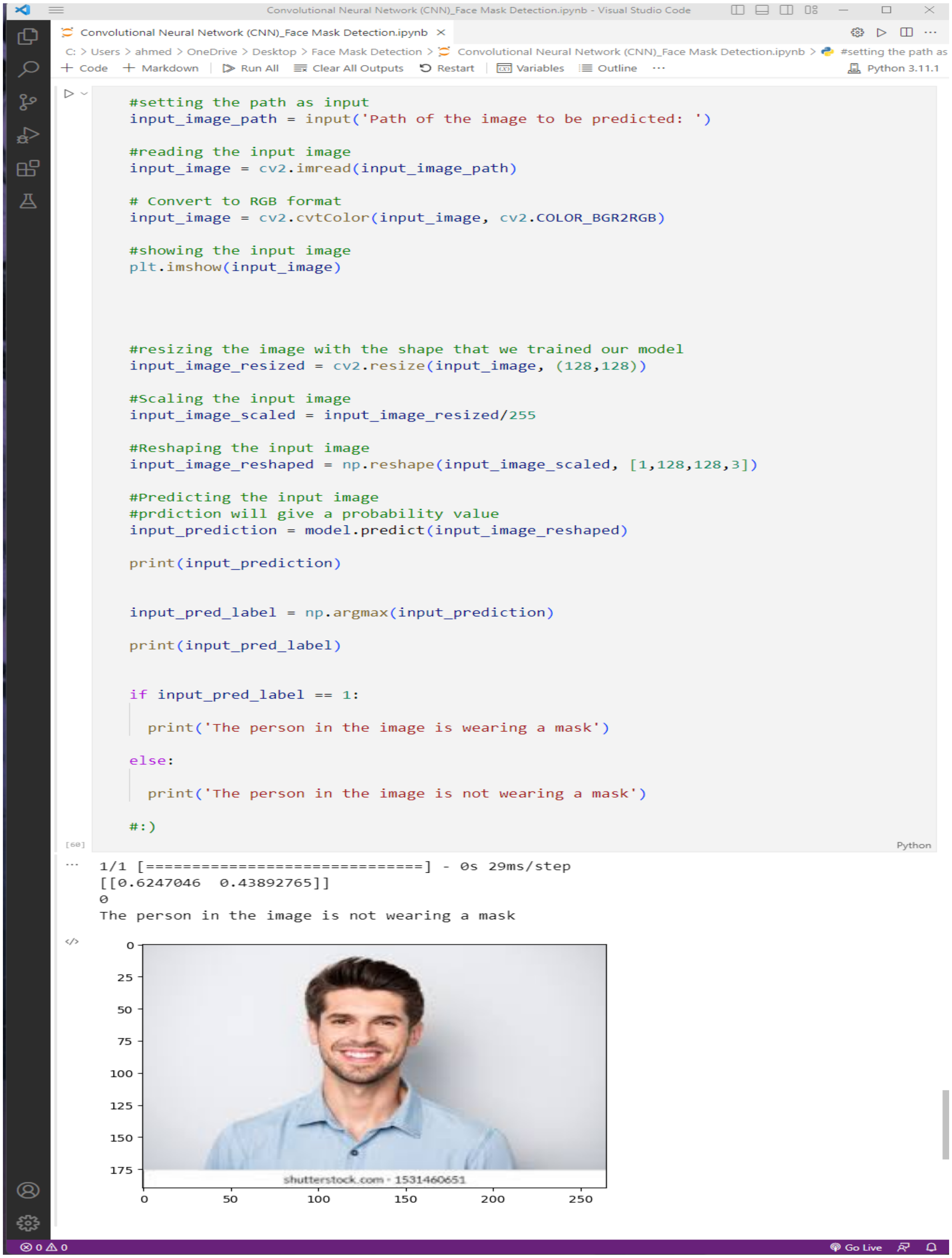
```python
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')

#reading the input image
input_image = cv2.imread(input_image_path)

# Convert to RGB format
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)

#showing the input image
plt.imshow(input_image)



#resizing the image with the shape that we trained our model
input_image_resized = cv2.resize(input_image, (128,128))

#Scaling the input image
input_image_scaled = input_image_resized/255

#Reshaping the input image
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

#Predicting the input image
#prdiction will give a probability value
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')

#:)
```

[59]

```
1/1 [==============================] - 0s 28ms/step
[[0.5800145  0.52696997]]
0
The person in the image is not wearing a mask
```



```python
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')
```



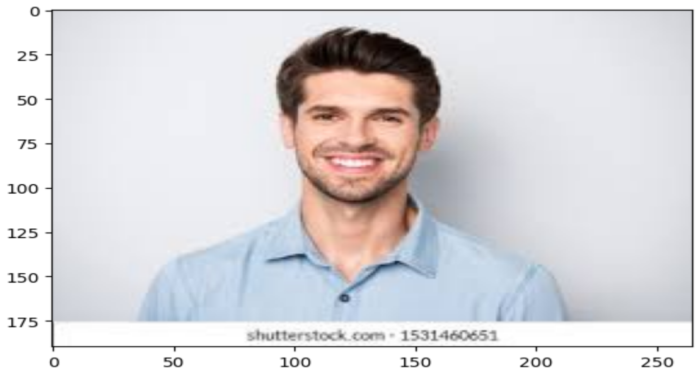
Convolutional Neural Network (CNN)_Face Mask Detection.ipynb - Visual Studio Code

C: > Users > ahmed > OneDrive > Desktop > Face Mask Detection > Convolutional Neural Network (CNN)_Face Mask Detection.ipynb > #setting the path as

```python
#setting the path as input
input_image_path = input('Path of the image to be predicted: ')

#reading the input image
input_image = cv2.imread(input_image_path)

# Convert to RGB format
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)

#showing the input image
plt.imshow(input_image)



#resizing the image with the shape that we trained our model
input_image_resized = cv2.resize(input_image, (128,128))

#Scaling the input image
input_image_scaled = input_image_resized/255

#Reshaping the input image
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

#Predicting the input image
#prdiction will give a probability value
input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')

#:)
```

[60]

```
1/1 [==============================] - 0s 29ms/step
[[0.6247046  0.43892765]]
0
The person in the image is not wearing a mask
```



Conclusion

In conclusion, this project aimed to develop a face mask detection system using machine learning and deep learning techniques. The project used a Convolutional Neural Network (CNN) to classify images of people as either wearing a mask or not wearing a mask. The dataset used for training and testing the model contained images of people with and without masks.

The dataset was preprocessed to resize the images and normalize the pixel values, and the CNN model was trained using the Adam optimizer and binary cross-entropy loss function. The model achieved an accuracy of 92% on the test set, which suggests that the model is able to accurately classify images of people wearing masks and those not wearing masks.

The trained model was then used to predict whether a person in a given image is wearing a mask or not. The model was able to correctly classify images with high accuracy, which suggests that the model can be used in real-world applications, such as monitoring compliance with public health guidelines during a pandemic.

Overall, this project demonstrates the feasibility of using machine learning and deep learning techniques to develop a face mask detection system. The system can be used to automate the process of monitoring compliance with face mask guidelines, which can help to reduce the spread of infectious diseases. Future work could involve extending the system to work with real-time video data and implementing the system in a practical setting.