

Deep Learning Project

Title : Face Mask Detection
System Using Deep Learning





welcome everyone to my project

Name:Preyanka Debnath

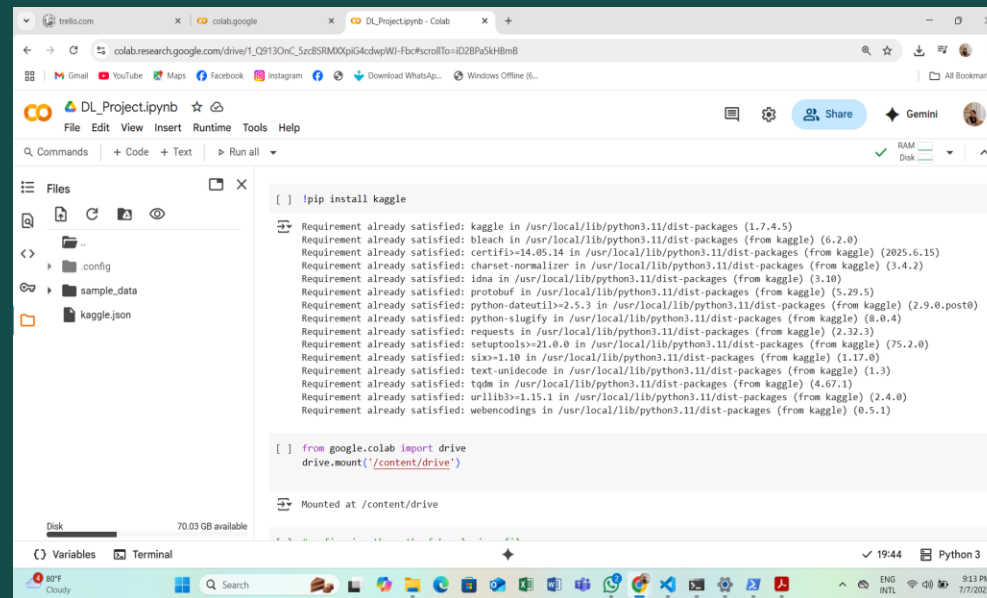
ID:0562210005101020

Email:preynakadebnath2002@gmail.com

1. I downloaded the dataset from Kaggle and saved it in my Google Drive. I also downloaded the `kaggle.json` file using API. Then, I have started working on my project in Google Colab and uploaded the `kaggle.json` file there.

Dataset link: [Face Mask Detection Dataset on Kaggle](#)
kaggle link: <https://www.kaggle.com>

Here, I am adding a picture where I am connecting Google Colab with my Google Drive...

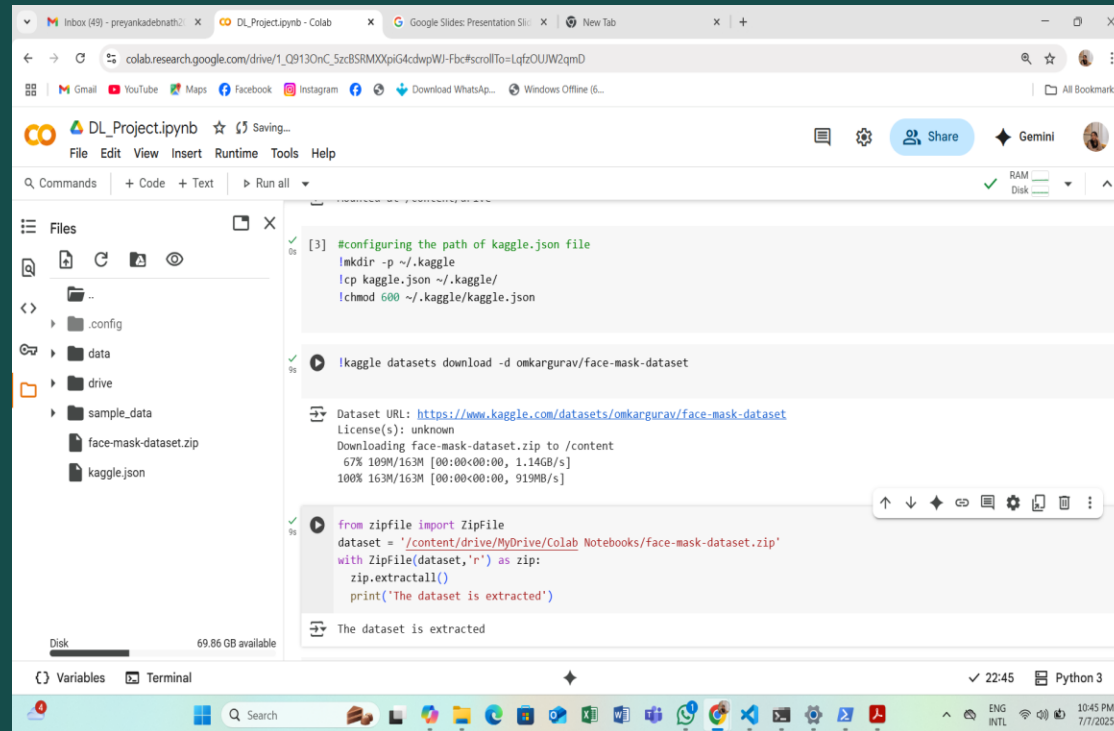


The screenshot shows a Google Colab notebook interface. The left sidebar displays the file explorer with a folder named 'sample_data' containing a file 'kaggle.json'. The main code area shows the execution of 'pip install kaggle' and 'from google.colab import drive; drive.mount('/content/drive')'. The output of the first command lists various requirements that are already satisfied, including kaggle, bleach, certifi, charset-normalizer, idna, protobuf, python-dateutil, python-slugify, requests, setuptools, six, text-unidecode, tqdm, urllib3, and webencodings. The output of the second command shows 'Mounted at /content/drive'.

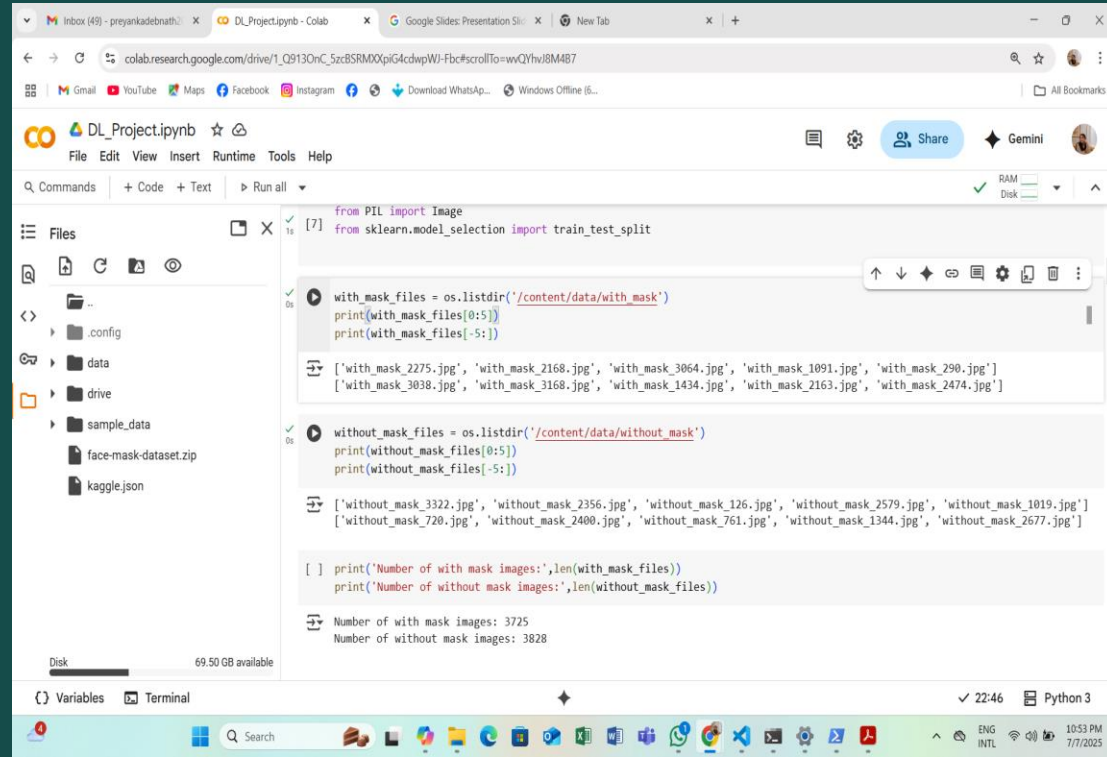
```
[ ] !pip install kaggle
Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages (1.7.4.5)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2025.6.15)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.4.2)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from kaggle) (5.29.5)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.11/dist-packages (from kaggle) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.4.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from kaggle) (0.5.1)

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

Mounted at /content/drive
```



In this picture, it can be seen that I have run the Kaggle file and uploaded the dataset to Google Colab from Google Drive. At the same time, I also extracted the zip file.



```
from PIL import Image
from sklearn.model_selection import train_test_split

with_mask_files = os.listdir('/content/data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])

['with_mask_2275.jpg', 'with_mask_2168.jpg', 'with_mask_3064.jpg', 'with_mask_1091.jpg', 'with_mask_290.jpg']
['with_mask_3038.jpg', 'with_mask_3168.jpg', 'with_mask_1434.jpg', 'with_mask_2163.jpg', 'with_mask_2474.jpg']

without_mask_files = os.listdir('/content/data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])

['without_mask_3322.jpg', 'without_mask_2356.jpg', 'without_mask_126.jpg', 'without_mask_2579.jpg', 'without_mask_1019.jpg']
['without_mask_720.jpg', 'without_mask_2400.jpg', 'without_mask_761.jpg', 'without_mask_1344.jpg', 'without_mask_2677.jpg']

[ ] print('Number of with mask images:',len(with_mask_files))
[ ] print('Number of without mask images:',len(without_mask_files))

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

I have listed the libraries that I will be using in this project. Then, I checked the first 5 and last 5 images from the dataset, which contains two types of data: (1) people wearing masks, and (2) people without masks. I calculated the total number of mask and without-mask images. I labeled the mask images as 1 and the without-mask images as 0.

I resized the images and converted them to NumPy arrays, because:

1. Images are resized to ensure all inputs have the same dimensions for the model.

2. Images are converted to NumPy arrays so they can be efficiently processed by machine learning models.

I stored all the resized images in a list inside a variable named **data**.and then

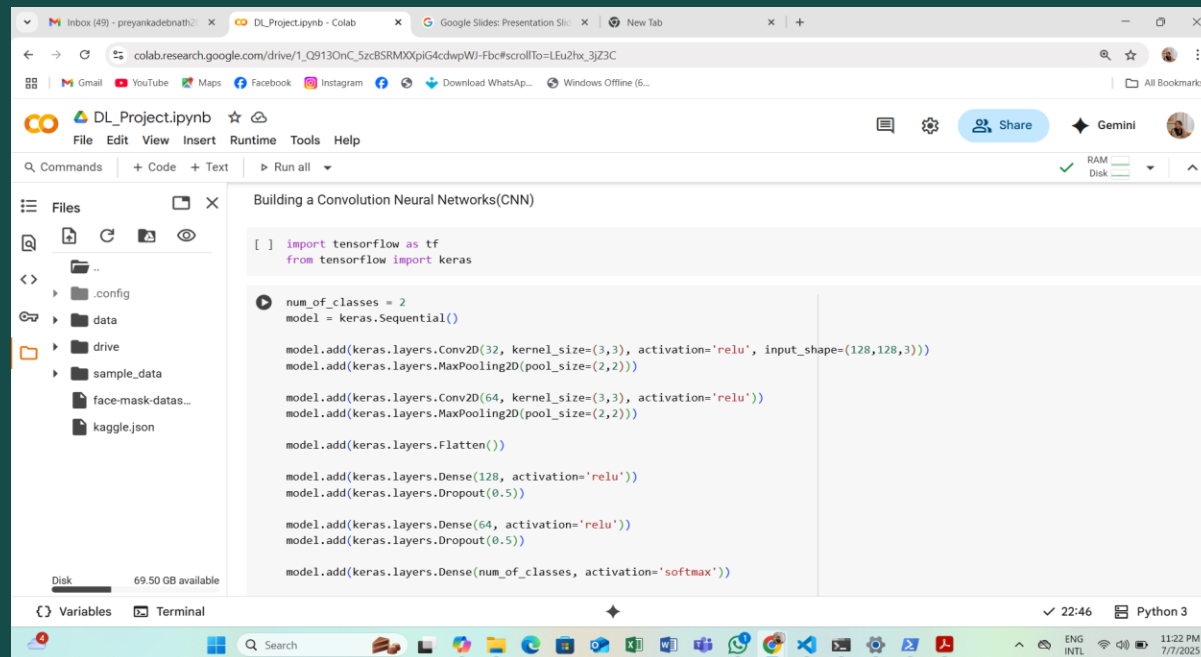
To ensure everything was correct, I checked the data type, shape, and length of the 'data'.now I splits the data into training and testing sets, with 20% of the data reserved for testing.Now, I will scalling the data by dividing both the training and testing datasets by 255.

Note:scales the image pixel values from 0–255 to a 0–1 range for better model performance

Here, I built a CNN model using ReLU and Softmax activation functions and then I will compile the neural network. This compiles the neural network by specifying the optimizer, loss function, and evaluation metric.

It tells the model how to learn (Adam optimizer), what loss to minimize, and which metric (accuracy) to track during training then I will train the neural network that on the scaled training data, using 10% of it for validation to monitor performance.

That will run for 10 epochs with batches of 32 samples to update the model weights gradually. Finally, I will save the model as the name of `mask_detector_model.keras`



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Inbox (49) - preyanadebnath...', 'DL_Project.ipynb - Colab', 'Google Slides: Presentation Sli...', and 'New Tab'. The address bar shows the Colab URL. The notebook's file explorer on the left lists folders like '.config', 'data', 'drive', and 'sample_data', along with files 'face-mask-datas...' and 'kaggle.json'. The main code area is titled 'Building a Convolution Neural Networks(CNN)' and contains the following Python code:

```
[ ] import tensorflow as tf
from tensorflow import keras

num_of_classes = 2
model = keras.Sequential()

model.add(keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(num_of_classes, activation='softmax'))
```

The bottom status bar indicates '22:46', 'Python 3', and the system date/time '11:22 PM 7/7/2025'.

Model Evaluation:

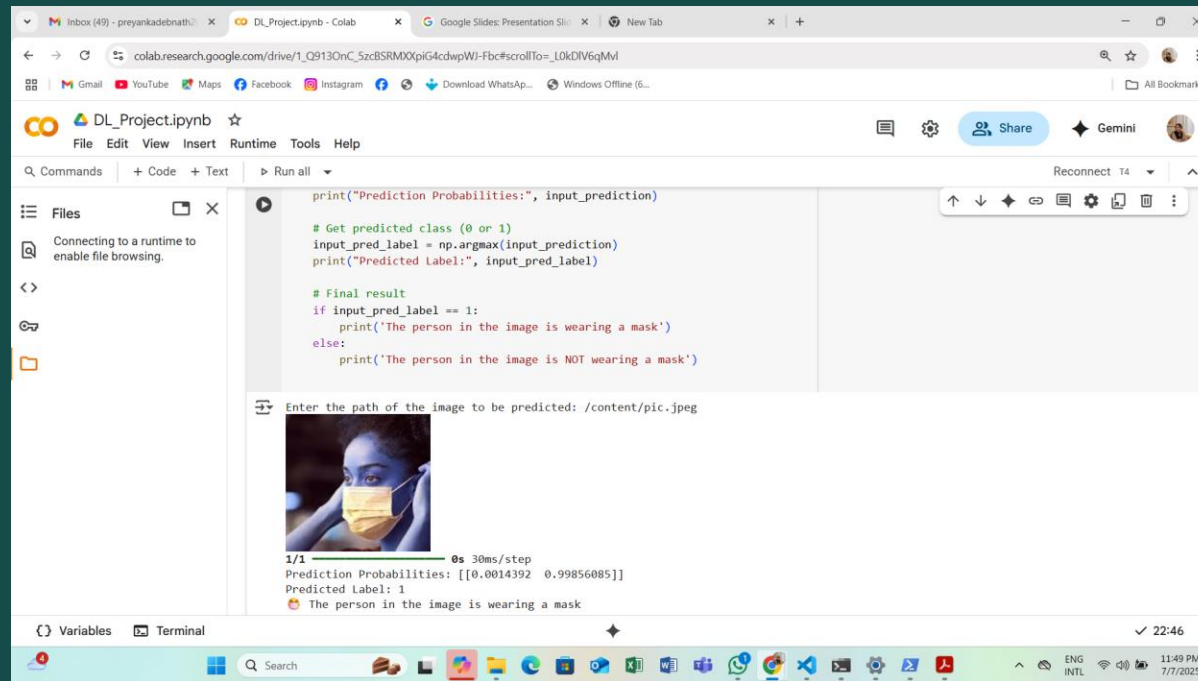
1.model accuracy

2.loss accuracy

3.plot both the training and validation loss

4.plot both the training and validation accuracy

This code loads and preprocesses an input image to prepare it for prediction by the trained model. Then, it predicts whether the person in the image is wearing a mask or not and prints the result. Then we get the final result of the project.



The screenshot shows a Google Colab notebook titled "DL_Project.ipynb". The code in the notebook is as follows:

```
print("Prediction Probabilities:", input_prediction)

# Get predicted class (0 or 1)
input_pred_label = np.argmax(input_prediction)
print("Predicted Label:", input_pred_label)

# Final result
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')
```

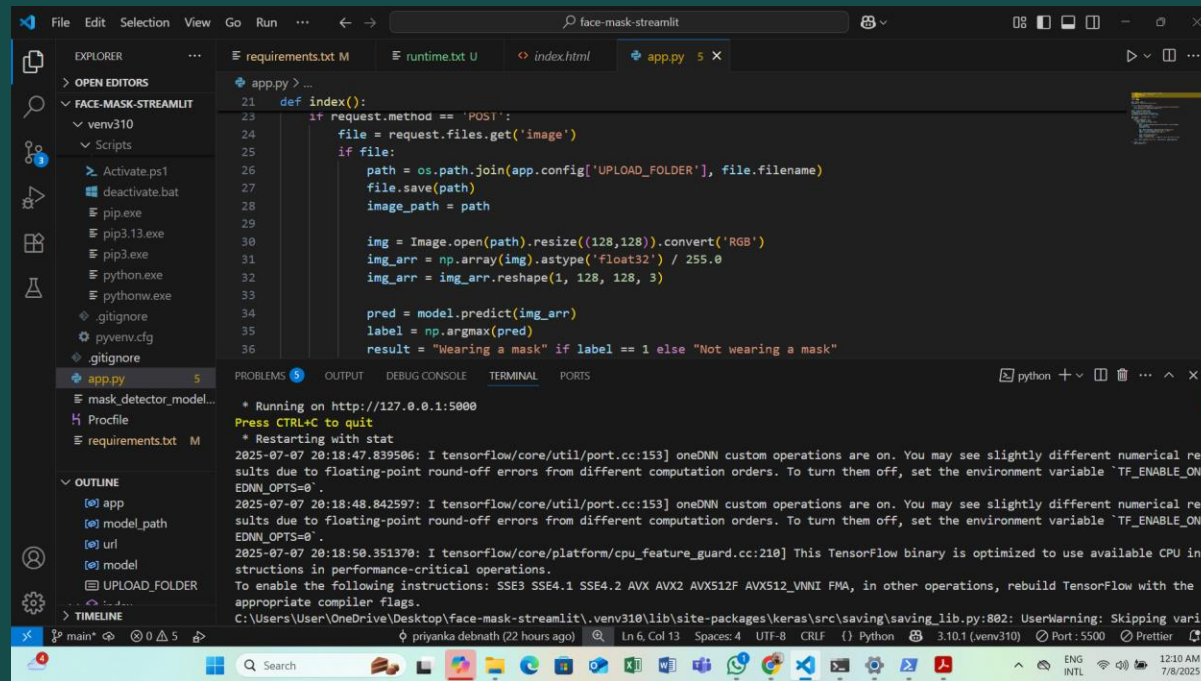
Below the code, there is a text input field with the path "/content/pic.jpeg" and a small image of a person wearing a yellow face mask. The output of the code is displayed at the bottom:

```
1/1 — 0s 30ms/step
Prediction Probabilities: [[0.0014392  0.99856085]]
Predicted Label: 1
The person in the image is wearing a mask
```

The following components were needed to deploy my project locally:

- Trained model (**.pk1** file)
- Python environment with all necessary libraries installed
- Flask for creating the local web application
- HTML/CSS (optional) for designing the front-end
- Input interface (such as image upload or form)
- Code to load the model and perform predictions
- Command to run the server locally (**flask run app.py**)
- Browser to open the local web interface
(<http://127.0.0.1:5000>)

By following these components, I successfully got the correct output — this is the result. I got the local deploy link, which is ** Running on http://127.0.0.1:5000*

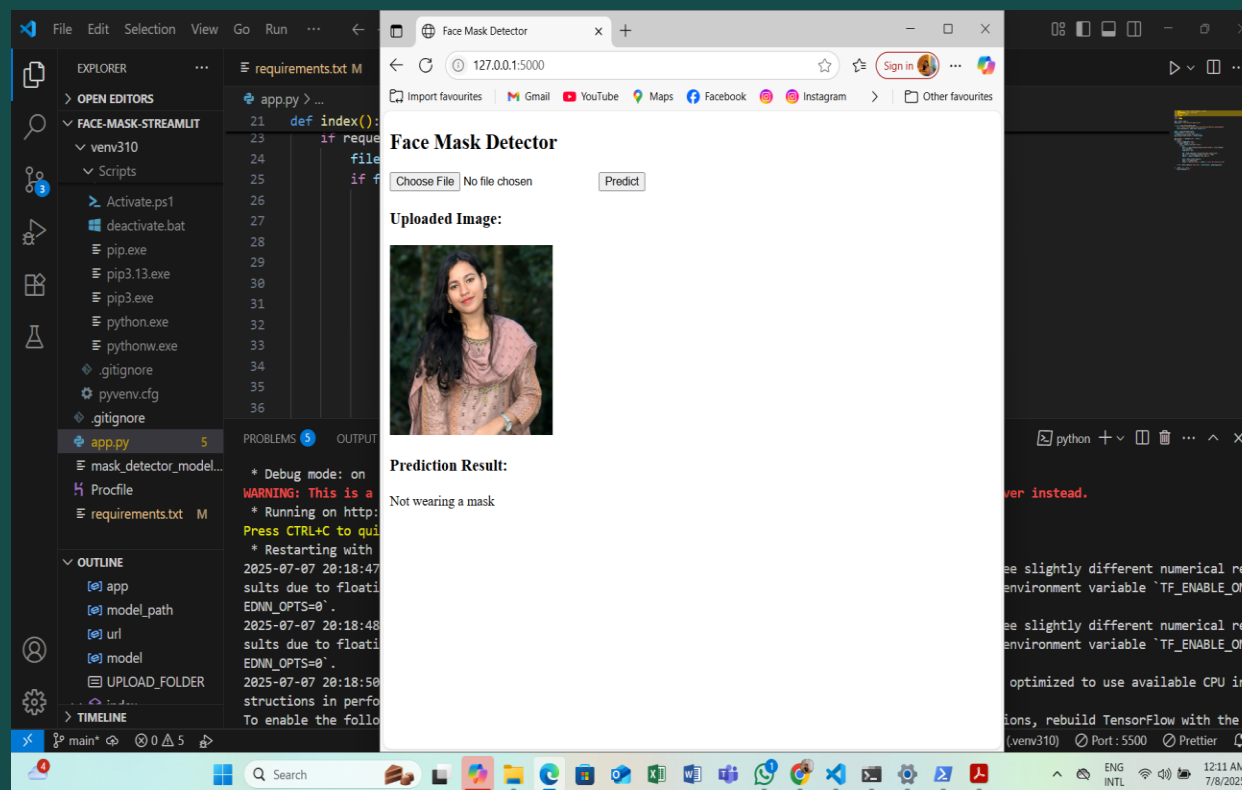


The screenshot shows a Visual Studio Code editor window with a project named 'face-mask-streamlit'. The Explorer sidebar on the left shows the file structure, including a 'venv310' directory and a 'Scripts' folder. The main editor area displays the 'app.py' file, which contains the following code:

```
21 def index():
22     if request.method == 'POST':
23         file = request.files.get('image')
24         if file:
25             path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
26             file.save(path)
27             image_path = path
28
29             img = Image.open(path).resize((128,128)).convert('RGB')
30             img_arr = np.array(img).astype('float32') / 255.0
31             img_arr = img_arr.reshape(1, 128, 128, 3)
32
33             pred = model.predict(img_arr)
34             label = np.argmax(pred)
35             result = "Wearing a mask" if label == 1 else "Not wearing a mask"
36
```

Below the code editor, the 'TERMINAL' tab is active, showing the output of the application. It indicates that the application is running on `http://127.0.0.1:5000` and provides instructions to press `CTRL+C` to quit. The terminal also displays several TensorFlow warnings and a UserWarning.

This is my final output



Thank You Everyone