**Image Classification Model Using CNN**

**By**

**Mrigank Sharma**

**Submitted to Scifor Technologies**

****

****

**Under the guidance of**

**Urooj Khan**

**TABLE OF CONTENTS**

**Abstract 03**

**Introduction 04**

**Technology Used 05-06**

**Dataset Information 06**

**Methodology 06**

**Code Snippet 09-14**

**Results and Discussion 15**

**Conclusion 16**

**Abstract**

In this project, we developed and implemented a Convolutional Neural Network (CNN) for image classification, leveraging its powerful feature extraction capabilities to categorize images into distinct classes. The project aimed to explore the efficacy of CNNs in handling complex image datasets and to demonstrate their practical application in automated image recognition tasks. The dataset comprised thousands of images across various categories, each preprocessed and normalized to optimize model training.

The CNN architecture was designed with multiple convolutional layers, each followed by activation and pooling layers, to progressively capture and abstract visual patterns in the data. A series of dense layers culminated in a softmax output, providing probabilistic classifications. We employed techniques such as data augmentation to enhance the diversity of the training set and prevent overfitting, alongside regularization methods.

The model's performance was evaluated using metrics such as accuracy, precision, recall, and F1-score on a held-out validation set. Furthermore, TensorBoard visualizations were utilized to monitor training progress and tune hyperparameters. The results demonstrated a high classification accuracy, validating the CNN's ability to generalize across diverse image categories.

This project underscores the potential of CNNs in image classification tasks, highlighting their application in fields such as computer vision, healthcare, and automated systems. Future work may involve exploring deeper architectures, transfer learning, and further optimizing the model's performance through hyperparameter tuning and advanced regularization techniques.

**Introduction**

Emotion recognition through image classification is a significant application in the field of computer vision, with far-reaching implications in areas such as mental health assessment, human-computer interaction, and social robotics. Identifying facial expressions, particularly discerning between happy and sad emotions, provides valuable insights into human behavior and psychological states. This project leverages Convolutional Neural Networks (CNNs) to automatically classify images of people as either happy or sad, aiming to contribute to the development of intelligent systems capable of understanding and responding to human emotions.

CNNs are a specialized type of deep neural network designed to process visual data by mimicking the hierarchical pattern recognition found in the human visual cortex. They are particularly well-suited for tasks involving images due to their ability to learn and extract relevant features through multiple layers of convolutions and pooling operations. This makes CNNs an ideal choice for emotion detection, as they can effectively capture subtle differences in facial expressions that differentiate between happiness and sadness.

In this project, we build a CNN-based model to classify facial expressions into two categories: happy and sad. The dataset comprises a variety of images depicting diverse individuals displaying these emotions. To enhance the robustness and generalizability of the model, we apply preprocessing techniques such as normalization and data augmentation, ensuring the network can handle variations in lighting, pose, and other factors that commonly affect real-world images.

The CNN architecture is designed to efficiently process the input images and identify key facial features indicative of the targeted emotions. It includes multiple convolutional layers to detect patterns, followed by pooling layers to reduce dimensionality and computational load, and fully connected layers to make the final classification. The model is trained using a labeled dataset, and its performance is evaluated based on metrics such as accuracy, precision, and recall.

This introduction outlines the project's aim to develop a robust and accurate emotion recognition system using CNNs, focusing on distinguishing between happy and sad facial expressions. The results of this project are expected to contribute to the growing field of affective computing and provide a foundation for further advancements in emotion-aware technologies.

**TECHNOLOGY USED**

The Technology stack used in the project is summarized as follows:

1. 1. Programming Language: Python
2. 2. Deep learning framework: Tensor flow
3. 3. Libraries:

Tenserflow

Matplotlib

* Numpy
* OS

4. Data Handling and Manipulation:

* Loading, cleaning and processing the images downloaded via google
* Manipulating array and images using tensor flow

5. Model building:

* Utilization of Keras
* Sequential Model architecture with densely corrected layers.
* Activation function such as Linear function.
* Compilation of the model loss, optimization, and evaluating metrics

6. Training and Evaluation:

* Training the Model using Fit method

Evaluating the model using accuracy metrics

7.Model deployment:

# Saving and loading the function using Kera’s load\_model Function

In the end these Technologies are responsible for building, training and deployment the pipeline of the model of image classification using CNN.

## **DATASET INFORMATION**

The image classification dataset contains 178 Image file in the folder of data in which there are separate folders in which each folder contains images of both happy and sad people. This dataset is used to determine the emotion of a particular person in the images.

**METHODOLOGY**

The methodology used in the project involves several steps:

1**. Importing libraries:**

- The code begins by importing necessary libraries like TensorFlow , Os, Numbpy Matplotlib and Keras for building , deploying , loading processing, and visualizing the data

**2.Loading the data and processing:**

- First we load the necessary images that we kept in the data directory locally and then format and remove the unnecessary images which are not in the format that we require ,.

Normalization: Images are normalized to scale pixel values to a standard range, typically [0, 1], to facilitate faster convergence during training.

Resizing: All images are resized to a uniform dimension suitable for the CNN input layer (e.g., 64x64 pixels) to maintain consistency.

Data Augmentation: Techniques such as rotation, flipping, zooming, and shifting are applied to artificially expand the dataset and improve the model’s robustness to variations in input images.

3.**Model Architecture:**

**Methodology**

The methodology for this project involves several key stages, including data collection and preprocessing, model design and architecture, training and validation, and evaluation and testing. Each stage is critical to the development and success of the Convolutional Neural Network (CNN) for detecting happy or sad facial expressions in images.

**1. Data Collection and Preprocessing**

1. **Data Collection**:
   * **Dataset Selection**: A dataset comprising images of individuals displaying happy and sad facial expressions is selected. Publicly available datasets like FER2013 or custom datasets with labeled emotions are considered.
   * **Data Diversity**: Ensuring diversity in terms of age, gender, ethnicity, lighting conditions, and image quality to improve model generalization.
2. **Data Preprocessing**:
   * **Normalization**: Images are normalized to scale pixel values to a standard range, typically [0, 1], to facilitate faster convergence during training.
   * **Resizing**: All images are resized to a uniform dimension suitable for the CNN input layer (e.g., 64x64 pixels) to maintain consistency.
   * **Data Augmentation**: Techniques such as rotation, flipping, zooming, and shifting are applied to artificially expand the dataset and improve the model’s robustness to variations in input images.

**2. Model Design and Architecture**

1. **CNN Architecture Design**:
   * **Input Layer**: The input layer is designed to accept images of a predefined size and channel number (e.g., 64x64x3 for RGB images or 64x64x1 for grayscale images).
   * **Convolutional Layers**: Several convolutional layers are stacked to extract hierarchical features from the images. Each convolutional layer is followed by an activation function (ReLU) to introduce non-linearity.
   * **Pooling Layers**: Max-pooling layers are incorporated to reduce the spatial dimensions and computational complexity, helping the model focus on the most salient features.
   * **Fully Connected Layers**: One or more fully connected (dense) layers are included after the convolutional and pooling layers to integrate the features and perform classification.
   * **Output Layer**: The final layer uses a softmax activation function to output probabilities for each class (happy, sad).

**4. Model Compilation:**

There are bunch of Optimizer available but in this case we are using adam optimizer using compile method to evaluate metric.

**5. Training Data**

**-**The complied data is trained using fit() method which is stored in hist

-The process involves iterating through epochs (20 in this case), with a validation split of 20% to monitor the model's performance on unseen data.

**6.Model plotting:**

Using matplotlib we are grabbing our training loss, validating loss and we are thn plotting those.

7.**Evaluate Performance:**

In order to evaluate we are gonna import some key metrics Precision, Recall, BinaryAccuracy and then we gonna test them out we gonna loop them and then run them to calculate accuracy.

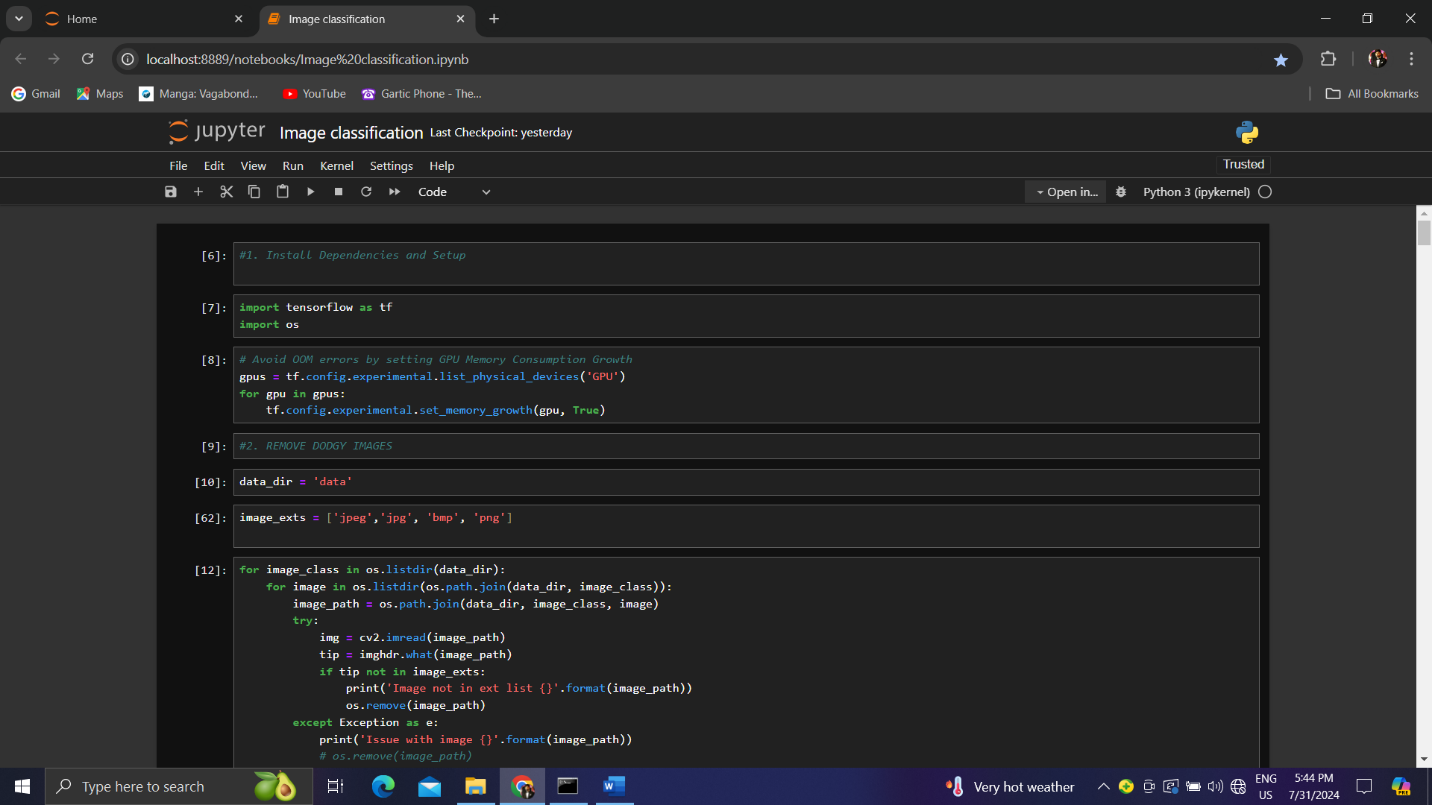
**8.Model Testing**

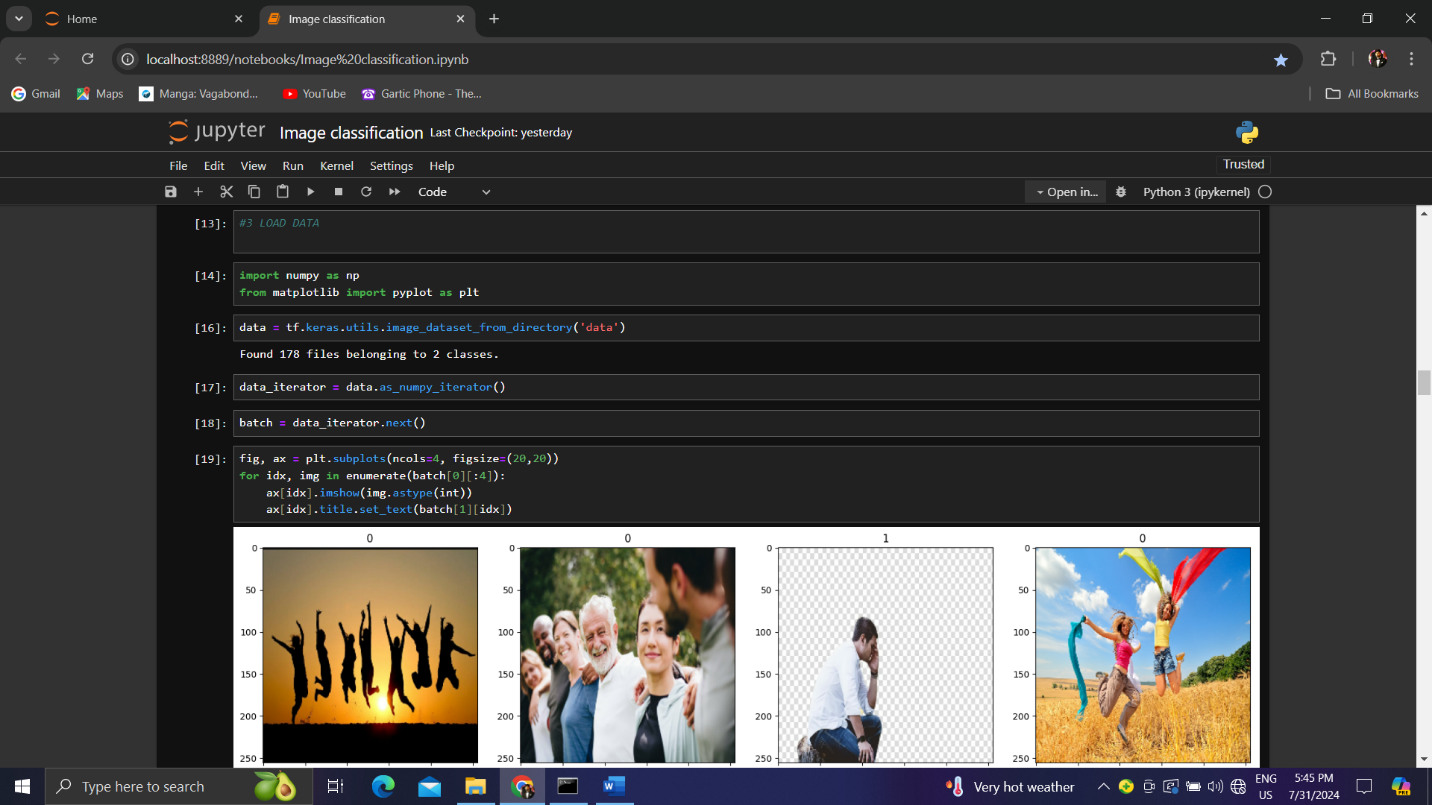
We are gonna grab a random image that we didn’t loop in the sample and our model never seen it eg . (I used a happy image) happy person and after applying image.cv2 and resizing it we gonna run it.

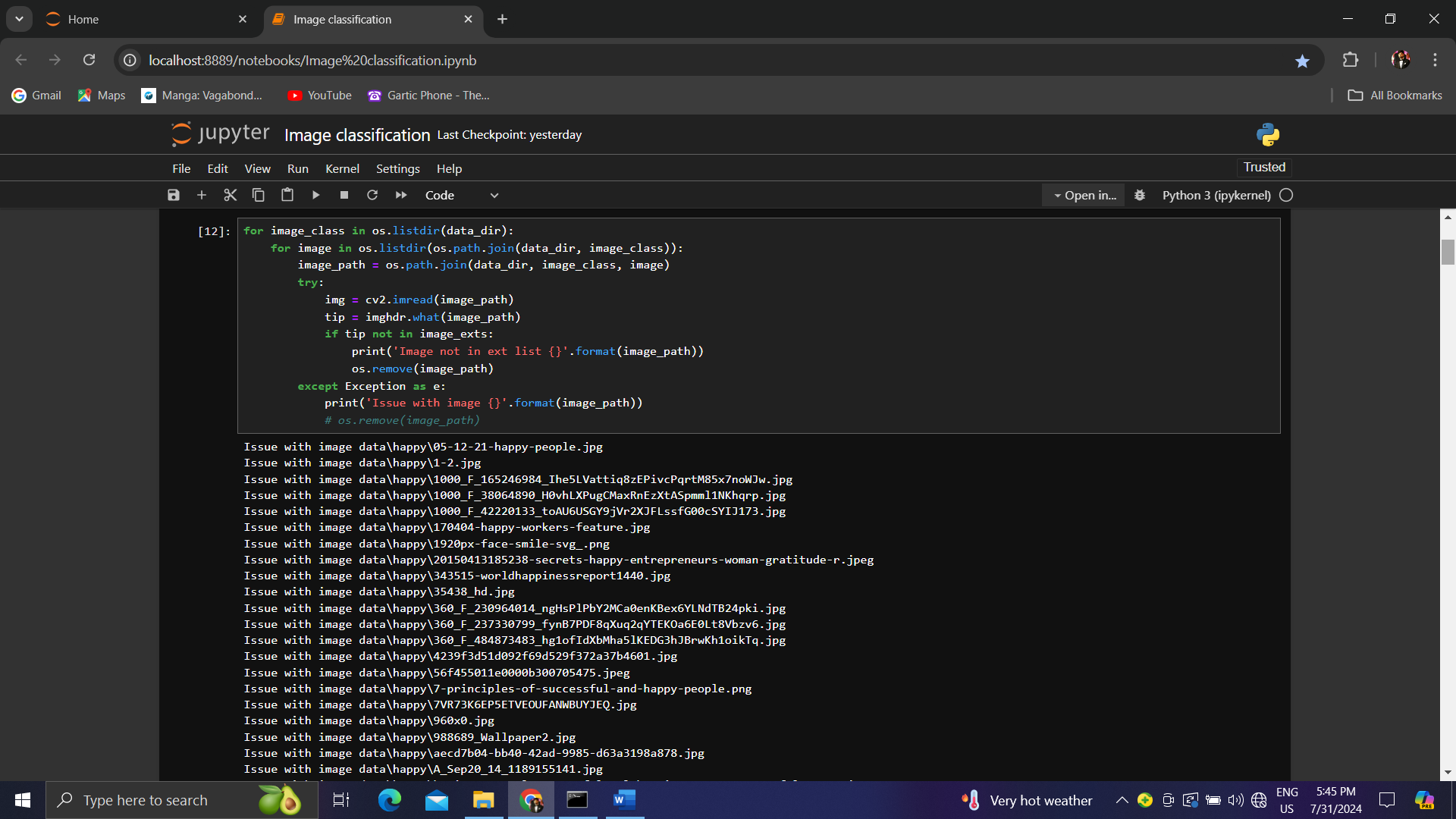
**9.Saving the model**

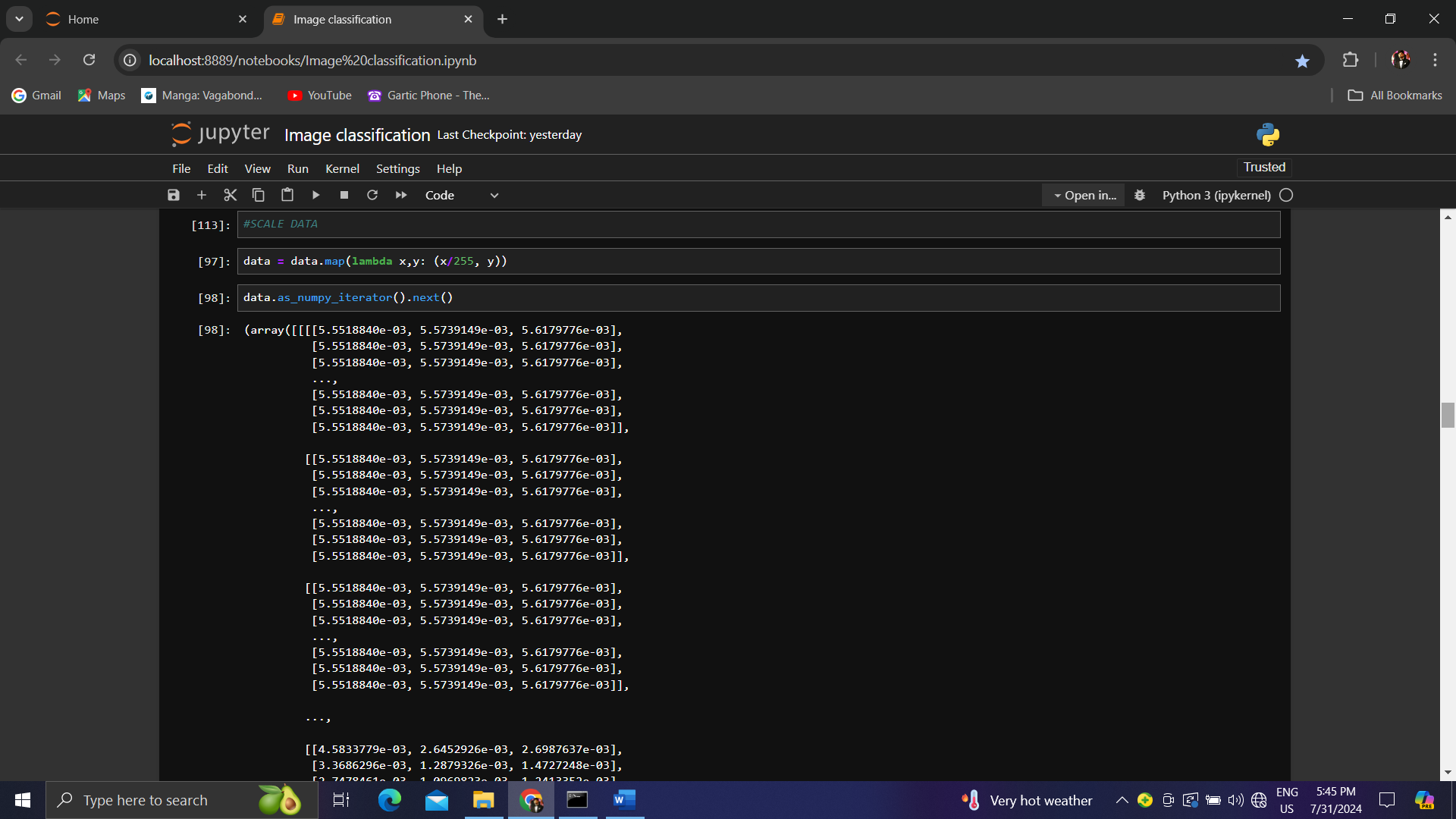
We gonna use tensorflow for this and just load\_model to save the model .

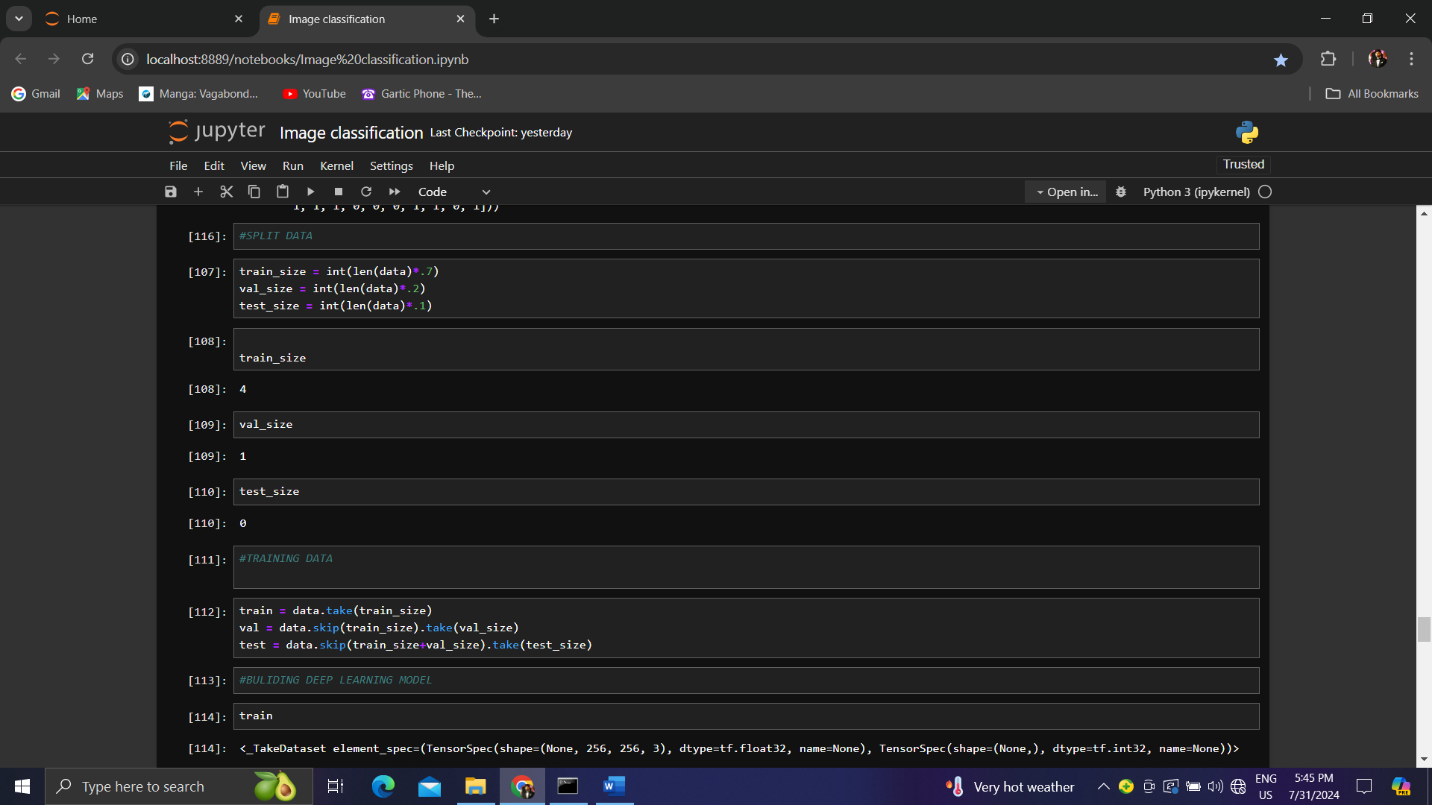
**CODE SNIPETS**

****

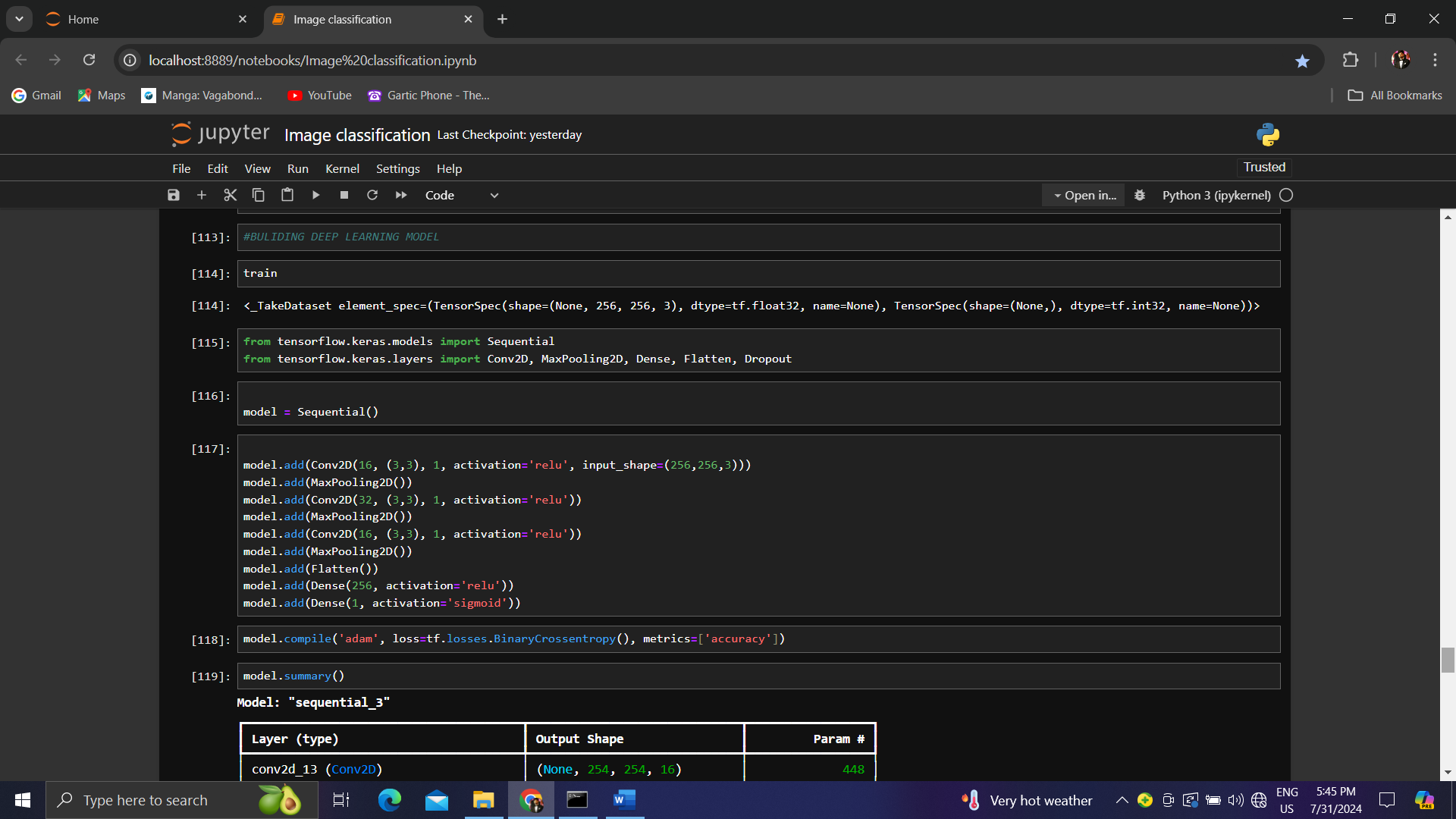
****

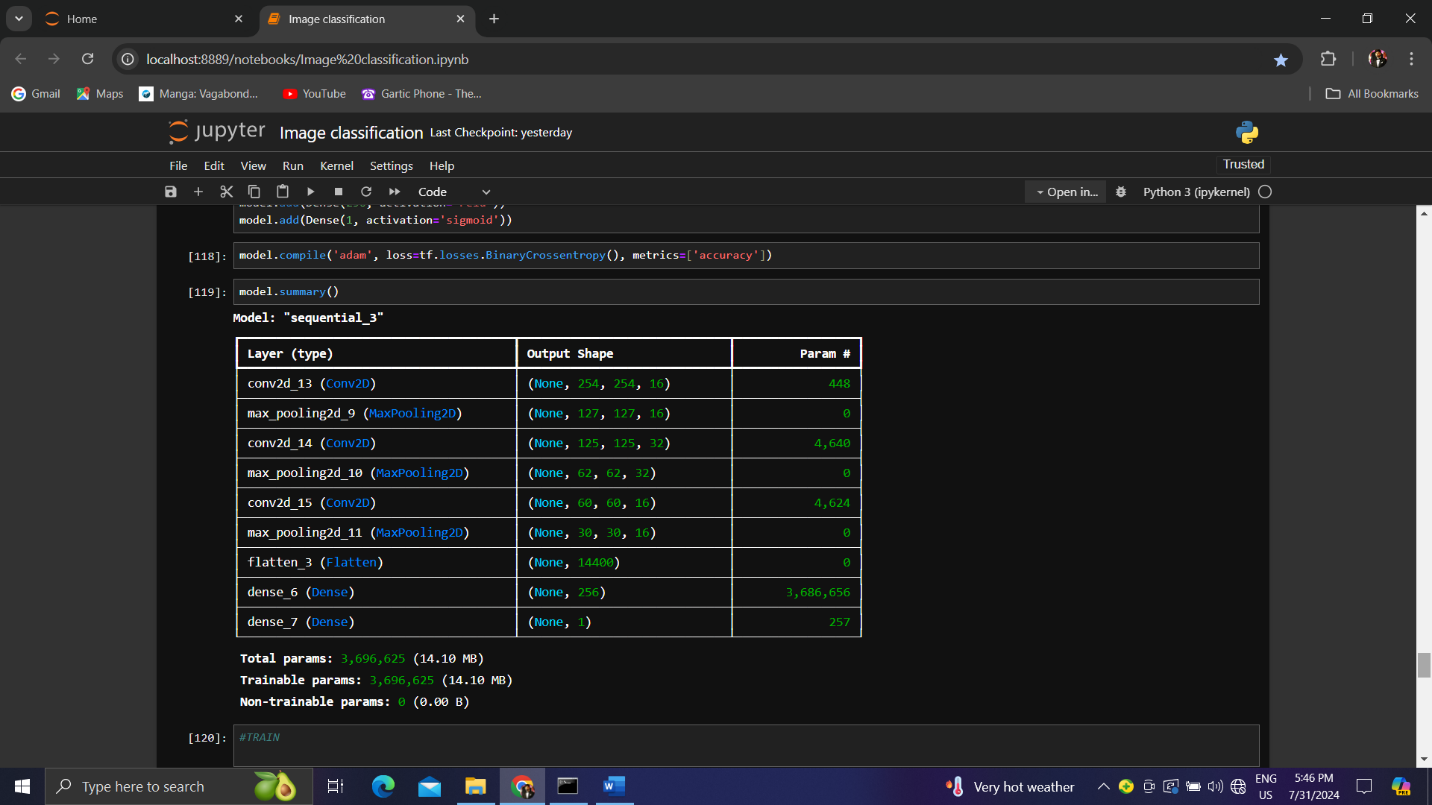
****

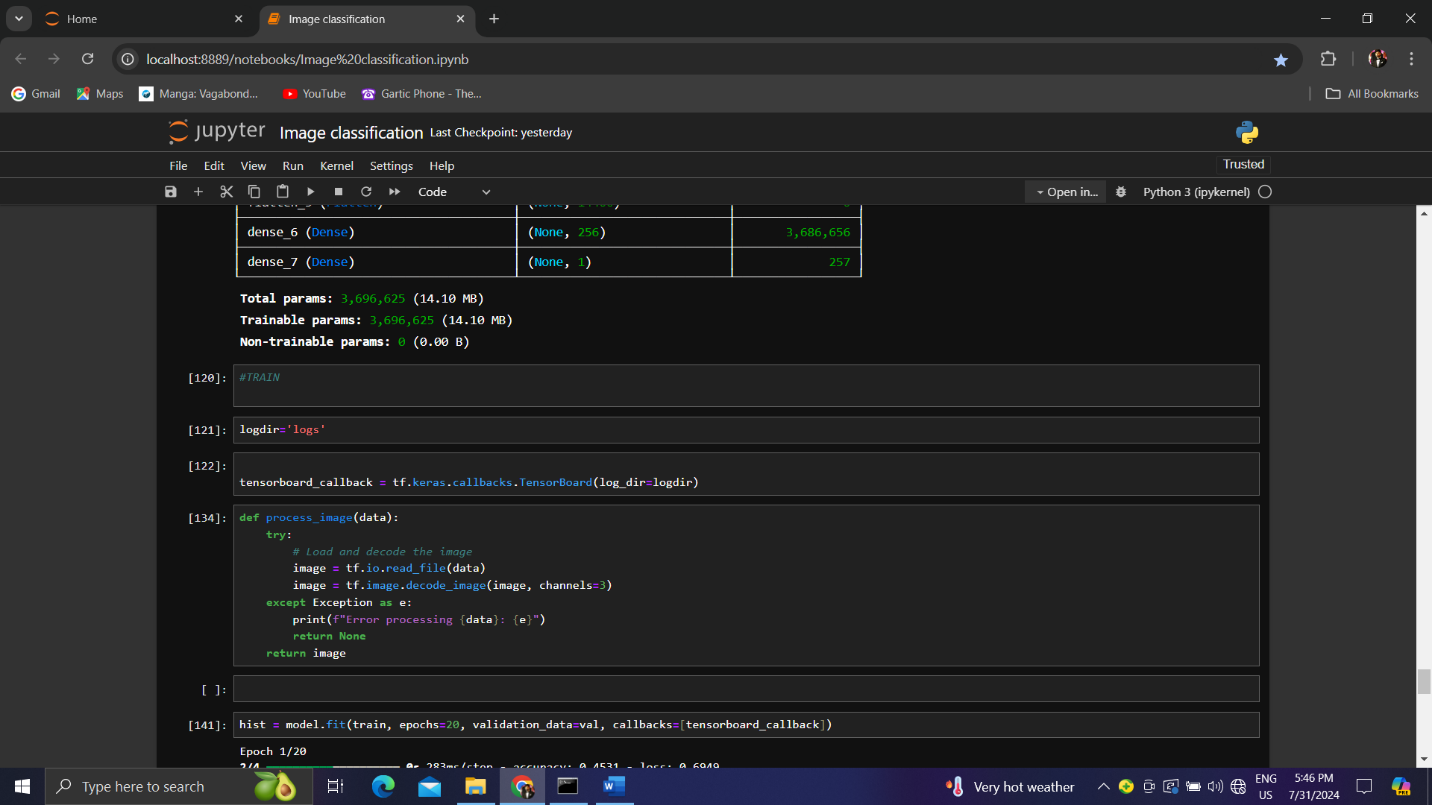


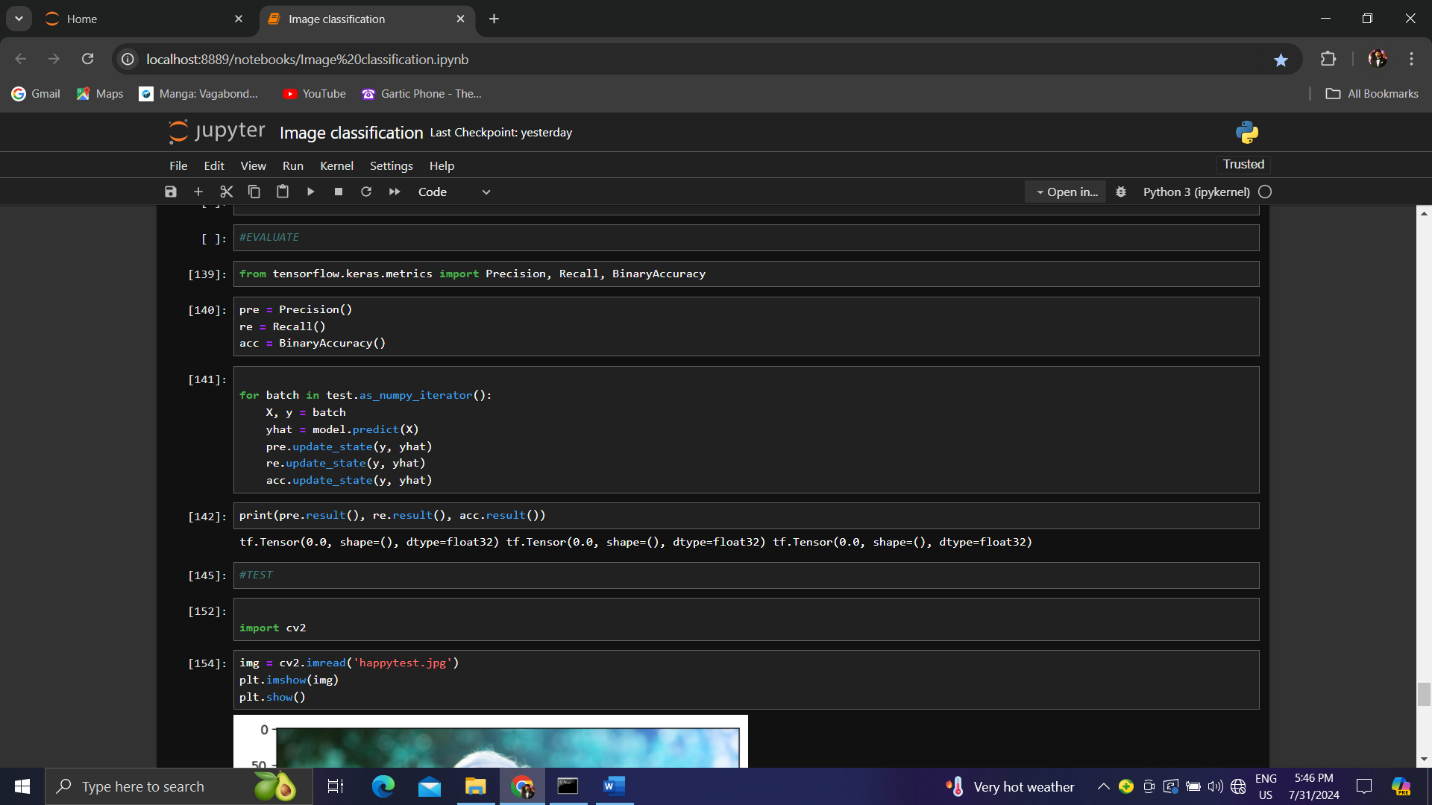


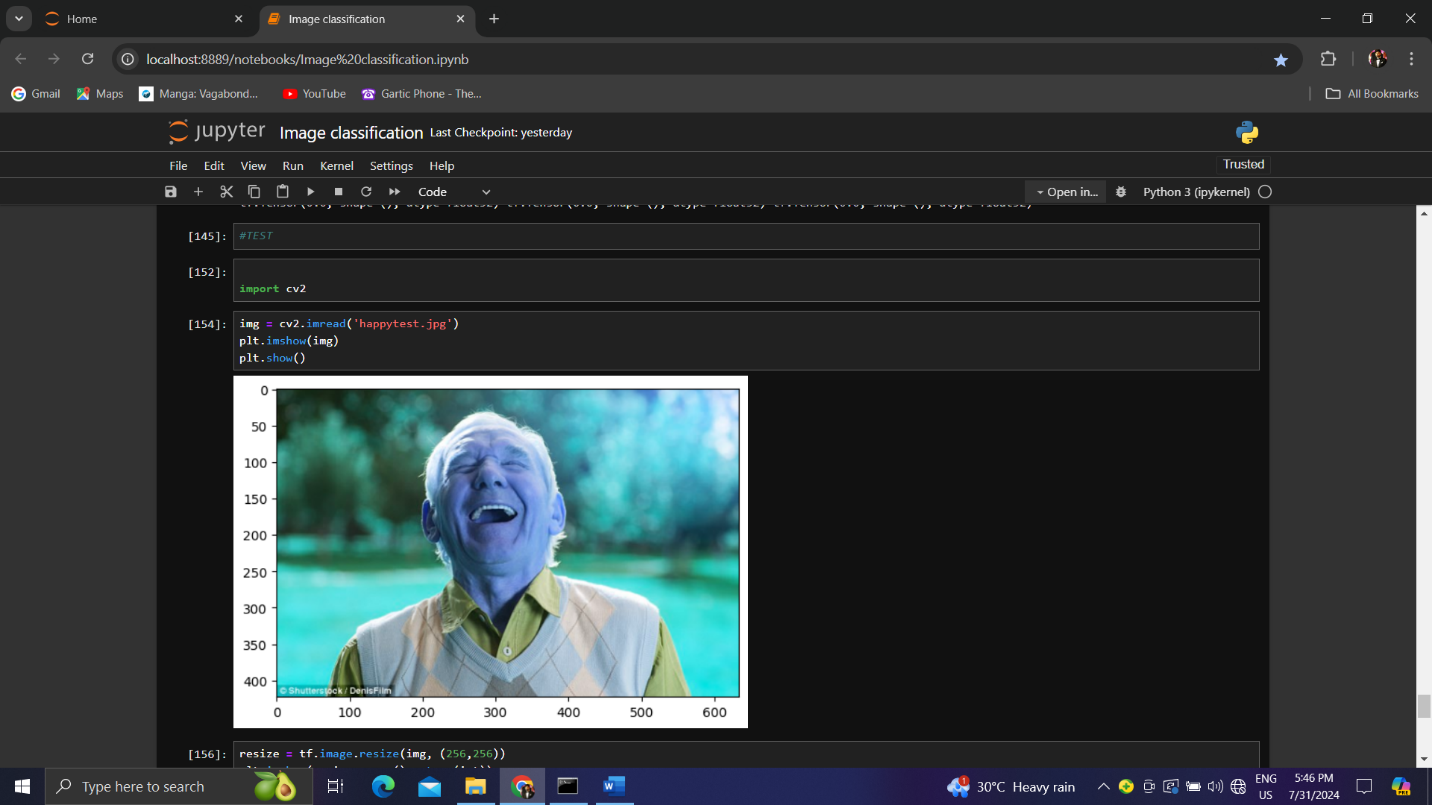




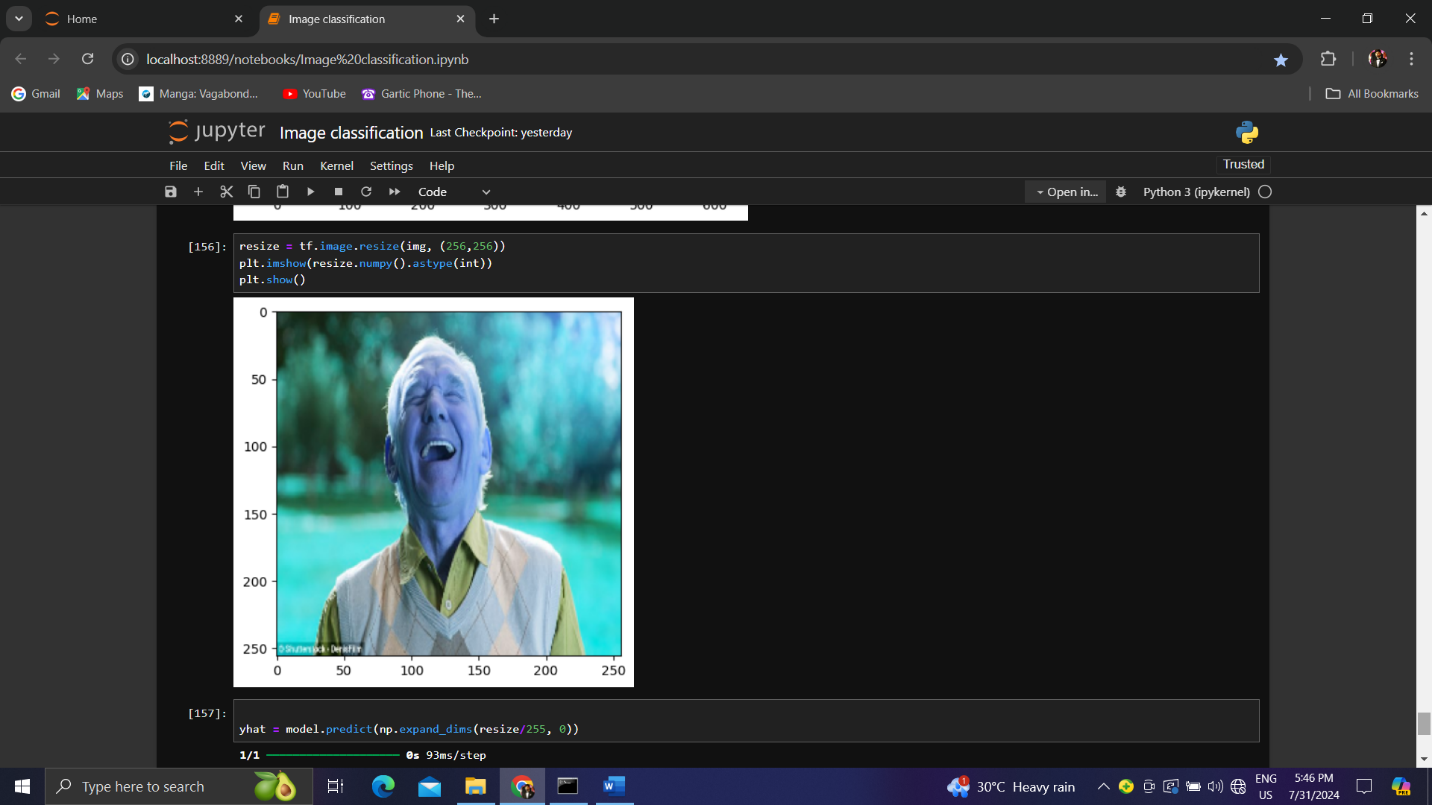


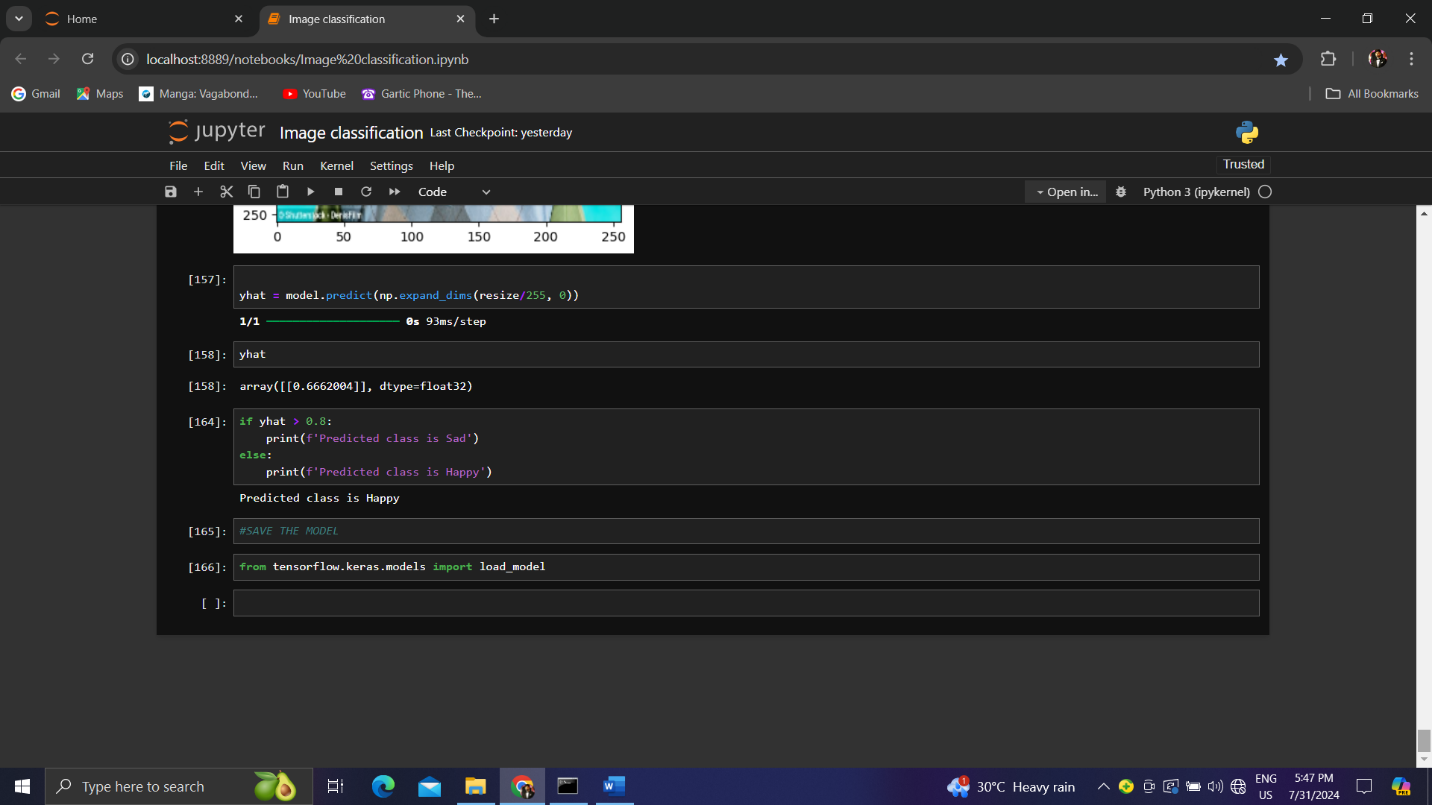






###### RESULT AND TESTING





##### CONCLUSION

In conclusion, this project seamlessly demonstrate the implementation of image classification using cnn and determines the emotion of a Person precisely