ABSTRACT

Addressing the significant threat that plant diseases pose to global food security and the limitations of traditional, manual detection methods, this project develops an automated diagnostic system using a Convolutional Neural Network (CNN), implemented with the TensorFlow framework. By training on a comprehensive dataset of plant leaf images, the CNN model learns to automatically extract and analyze distinguishing visual features, such as spots, texture, and color anomalies, to accurately identify and classify a range of plant diseases. The primary objective is to create a robust and efficient tool that provides a rapid and reliable diagnosis from a simple leaf image. This technology is designed to serve as an accessible solution for early disease detection, enabling timely intervention to mitigate crop loss, optimize treatment, and enhance overall productivity with AI suggestions.

CASE STUDY

PROJECT NAME: Plant Disease Detection and AI-Based Suggestion System.

INTRODUCTION:

The health of a crop is central to agricultural success, yet diseases—often manifesting first on the leaves—pose a constant and significant threat. Current detection methods are manual, time-consuming, and often leading to costly misdiagnoses and delayed treatment. By the time a disease is identified from visible symptoms on the leaves, it may have already spread, causing substantial crop losses. This project aims to solve this problem by developing an AI-powered system that can instantly identify plant diseases from a simple photo of a leaf. By providing specific, actionable suggestions for treatment, our system will empower users to proactively manage crop health, reduce losses, and increase yields for a more sustainable future.

Objectives:

1. CNN-Based Disease Identification: To develop and train a robust Convolutional Neural Network (CNN) model capable of accurately identifying and classifying various plant diseases from leaf images.

2. Actionable Advisory System: To build an AI-powered advisory module that provides practical recommendations for treating and preventing identified diseases.

3. User-Friendly Interface: To design and implement an intuitive and user-friendly application (e.g., mobile or web) that allows for effortless image submission and instant results.

4. Data Collection and Knowledge Base Expansion: To establish a system for continuous data collection and feedback from users, thereby improving the model and advisory content over time.

5. Sustainable Agricultural Impact: To contribute to more sustainable and efficient farming by providing timely disease insights that help reduce crop loss and minimize the use of chemical treatments

IMPLEMENTATION

Data Preparation & Curation:

- This stage involves obtaining and organizing the dataset we'll use to train our plant disease prediction model.
- **Download the dataset from Kaggle:** We use the Kaggle to download the "plantvillage-dataset". This process confirms successful download and extraction.
- **Extract the dataset:** The downloaded file is a zip archive, which we extract to access the image files.
- **Explore the dataset structure:** We examine the extracted folders to understand how the data is organized. The dataset is divided into three main directories: 'color', 'segmented', and 'grayscale'. We'll use the 'color' images. We confirm the number of classes is 38 and explore examples of class directories.

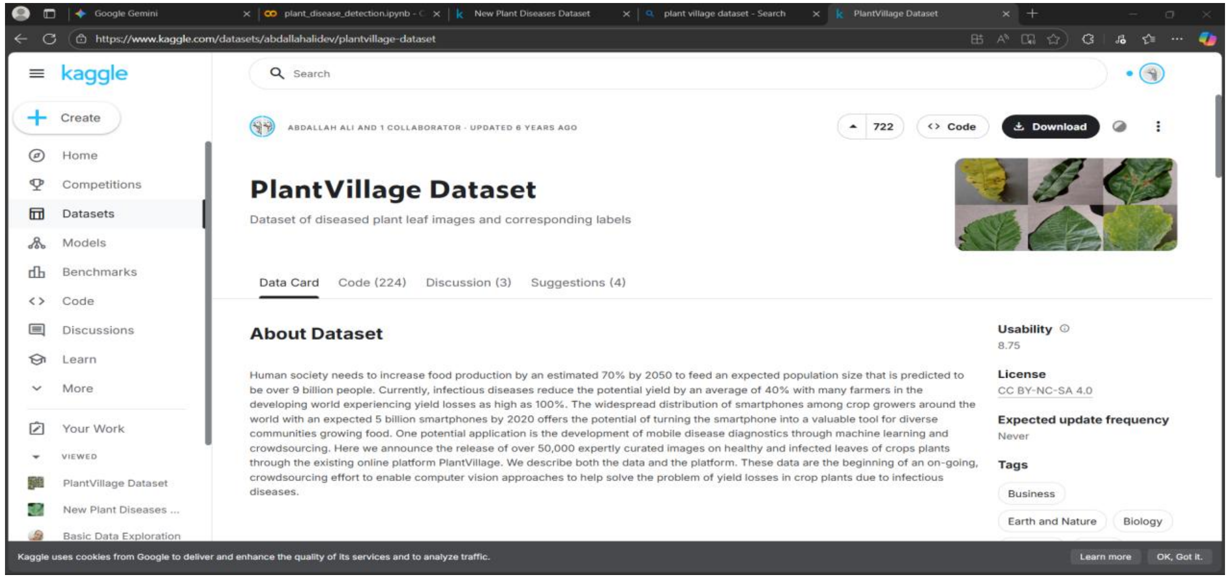


Image Preprocessing & Splitting

In this step, we prepare the images for input into our Convolutional Neural Network (CNN) and divide the dataset into training and validation sets.

Define Image Parameters: We set the target size for all images (224x224 pixels) and the batch size for training (32 images per batch).

Create Image Data Generators: We use ***ImageDataGenerator*** from ***Keras*** to handle the image data, rescaling pixel values to [0, 1] and splitting the data into 80% training and 20% validation sets.

Generate Training and Validation Data Batches: The ***flow_from_directory*** method reads images, resizes them, applies rescaling, and organizes them into batches. This confirms we found 43,456 images for training and 10,849 images for validation, and provides the ***train_gen*** and ***val_gen*** objects for training and evaluation.

Model Architecture Definition

Here, we design the structure of our Convolutional Neural Network (CNN) for image recognition.

Initialize Sequential Model: We create a Sequential model, a linear stack of layers.

Add Convolutional Layers: We add Conv2D layers (with 32, 64, and 128 filters) to detect image features.

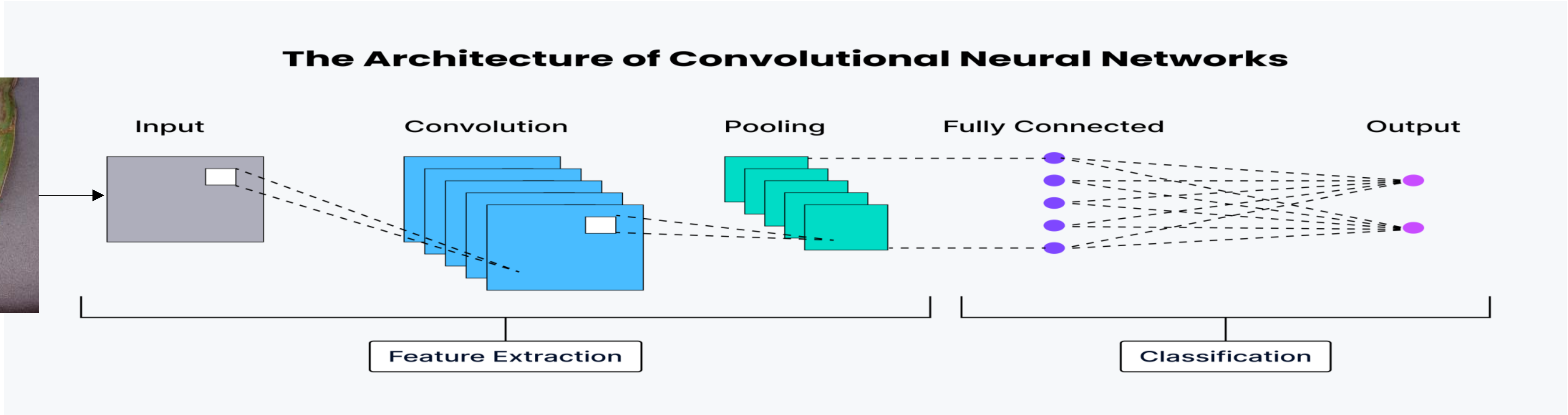
Add Pooling Layers: MaxPooling2D layers reduce spatial dimensions and retain important information.

Add Dropout Layers: Dropout layers are included to prevent overfitting.

Flatten Layer: The Flatten layer converts 2D feature maps to a 1D vector. (i.e. 1D Array)

Add Dense Layers: Dense layers learn to classify features.

Output Layer: The final Dense layer has 38 neurons with ***softmax*** activation for class probability distribution. The ***model.summary()*** provides a summary of the defined architecture, including layers, output shapes, and parameter counts.

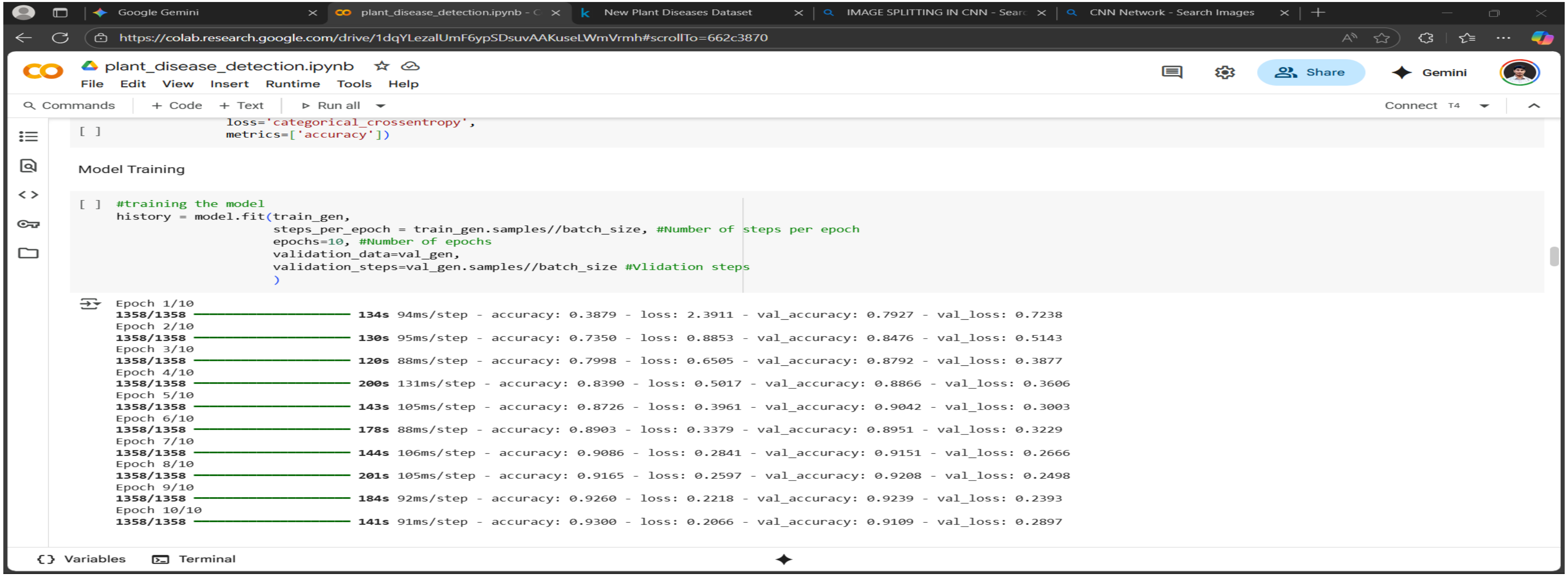


Model Training

In this phase, the CNN learns to identify plant diseases.

Compile the Model: We compile the model with the **adam optimizer**, categorical_crossentropy loss function, and accuracy metric.

Train the Model: We train the model using **model.fit()** with train_gen and val_gen, specifying steps_per_epoch, epochs, validation_data, and validation_steps. This process outputs training progress information for each epoch, including loss and accuracy, and provides a history object containing these metrics.



Model Evaluation

After training, we evaluate the model's performance on the validation dataset.

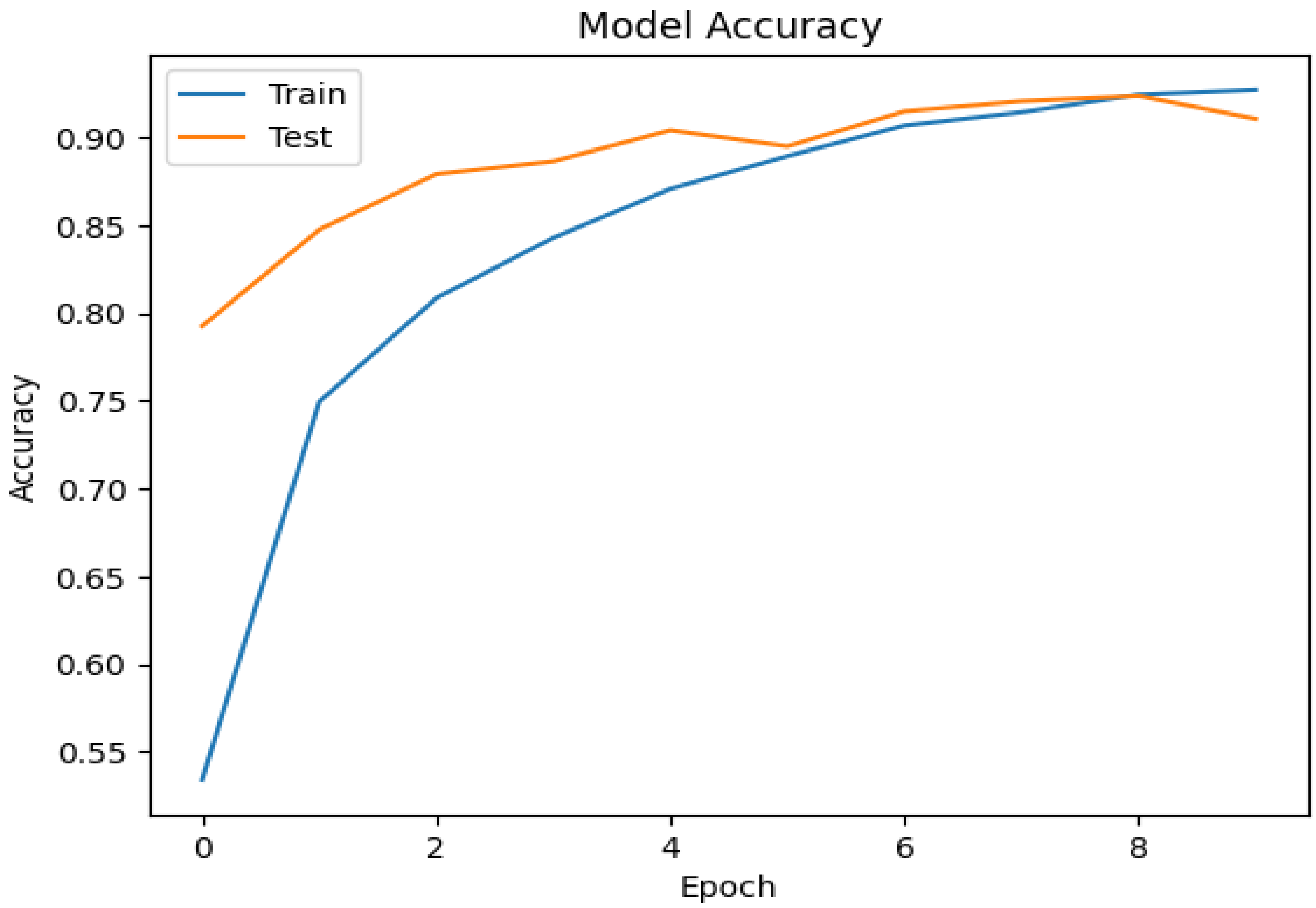
Evaluate the Model: We use ***model.evaluate()*** with val_gen to calculate the loss and accuracy on the validation set, providing the validation loss and accuracy scores (we got 91% validation accuracy here) .

Visualize Performance: We plot the training and validation accuracy and loss over epochs using the history object, visualizing the model's learning progress.

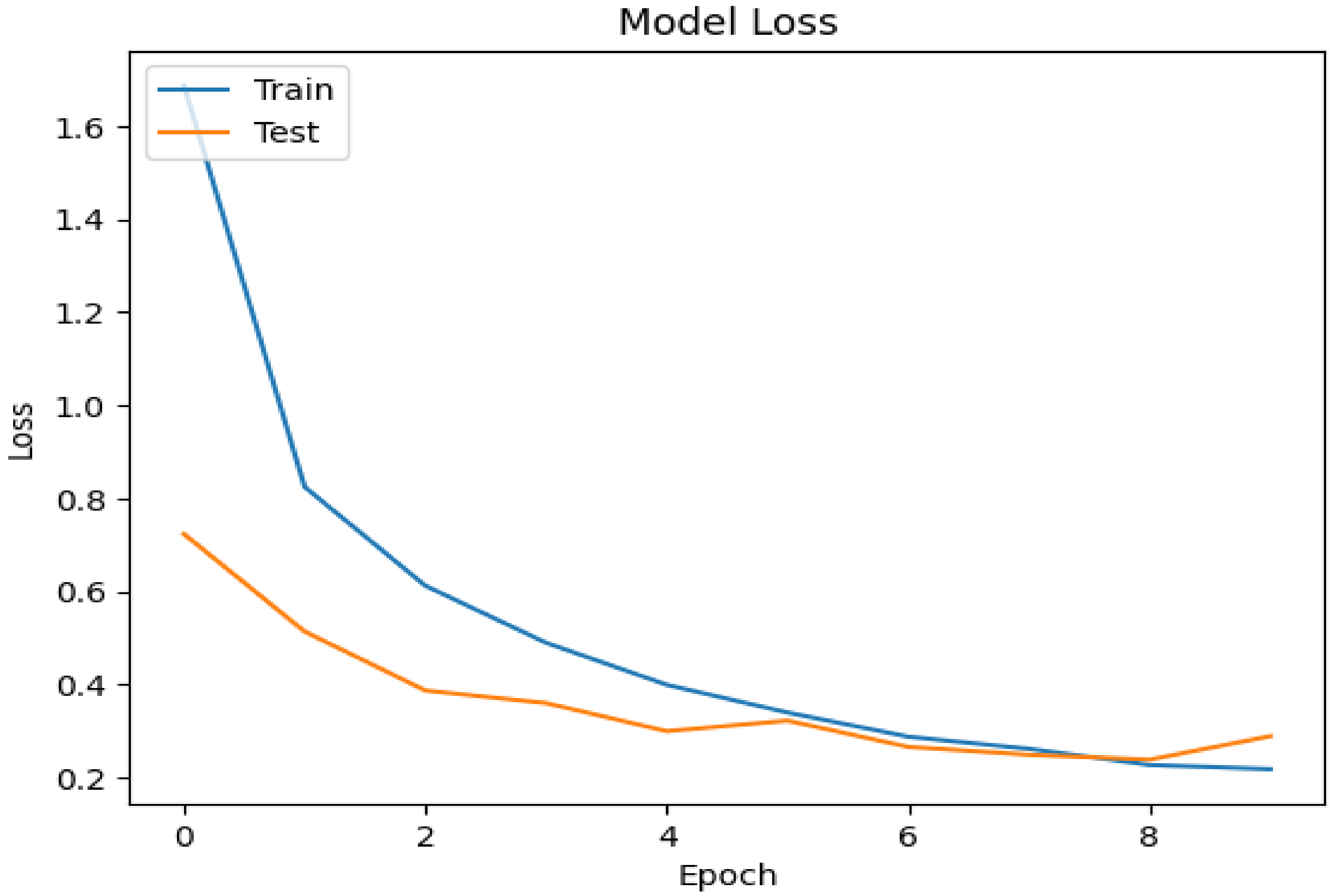
```
val_loss, val_acc = model.evaluate(val_gen, steps=val_gen.samples//batch_size)
print("Validation Loss:", val_loss)
print("Validation Accuracy:", val_acc)
```

```
339/339 ━━━━━━━━━━━━━━━━━━━━ 23s 68ms/step - accuracy: 0.9146 - loss: 0.2819
Validation Loss: 0.2897346615791321
Validation Accuracy: 0.9108591675758362
```

1. Plot representing the Accuracy of Model at each Epoch



2. Plot representing the Loss of Model at each Epoch



Predictive System & Saving

In this final stage, we create a system for making predictions and save the model.

Create Prediction Function: We define predict_image_class to load, preprocess, and predict the class of a new image using the trained model and class indices mapping.

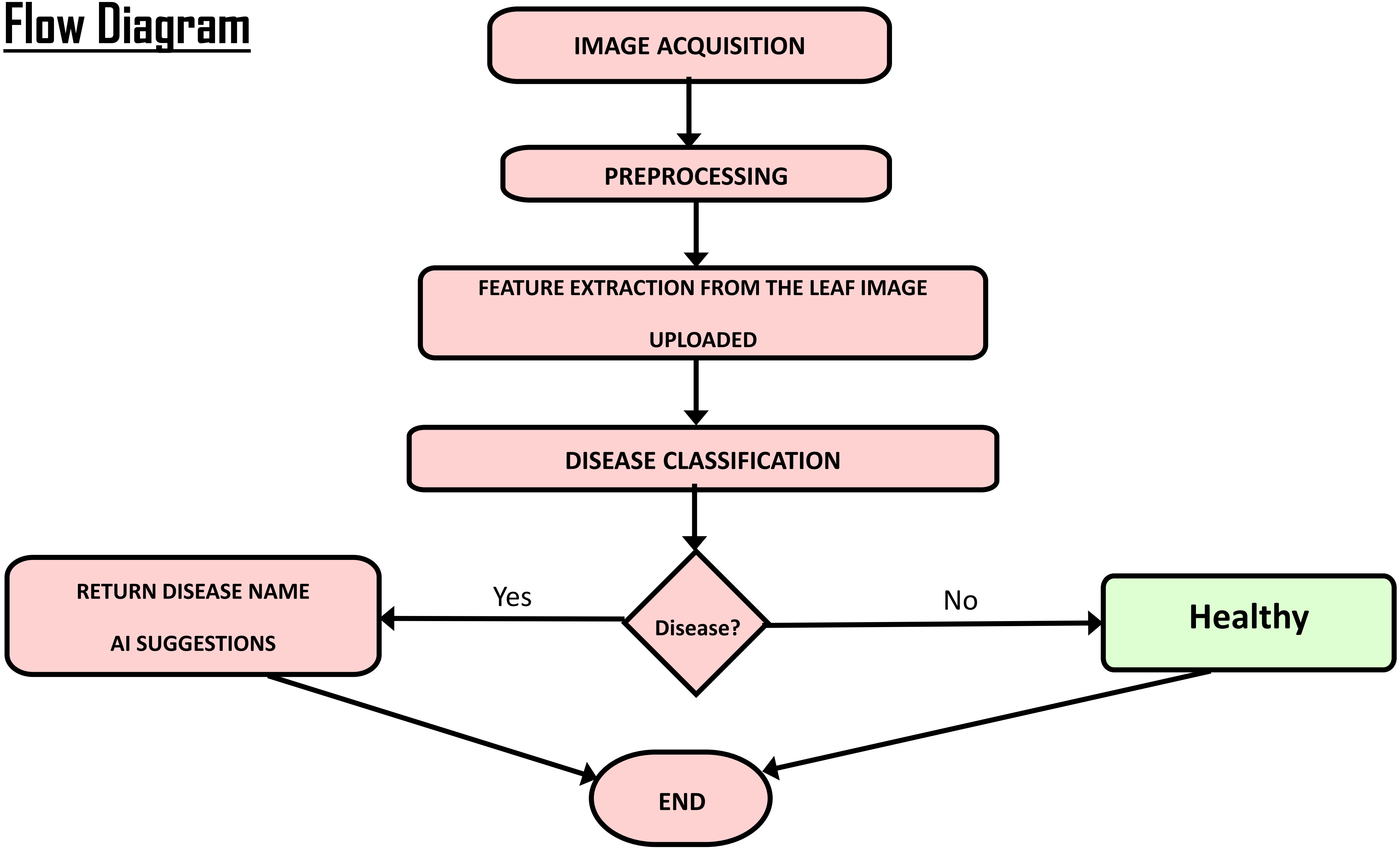
Create Class Indices Mapping: We create a class_indices dictionary from train_gen.class_indices that maps numerical indices to class names.

Save Class Indices: The class_indices dictionary is saved as class_indices.json for later use.

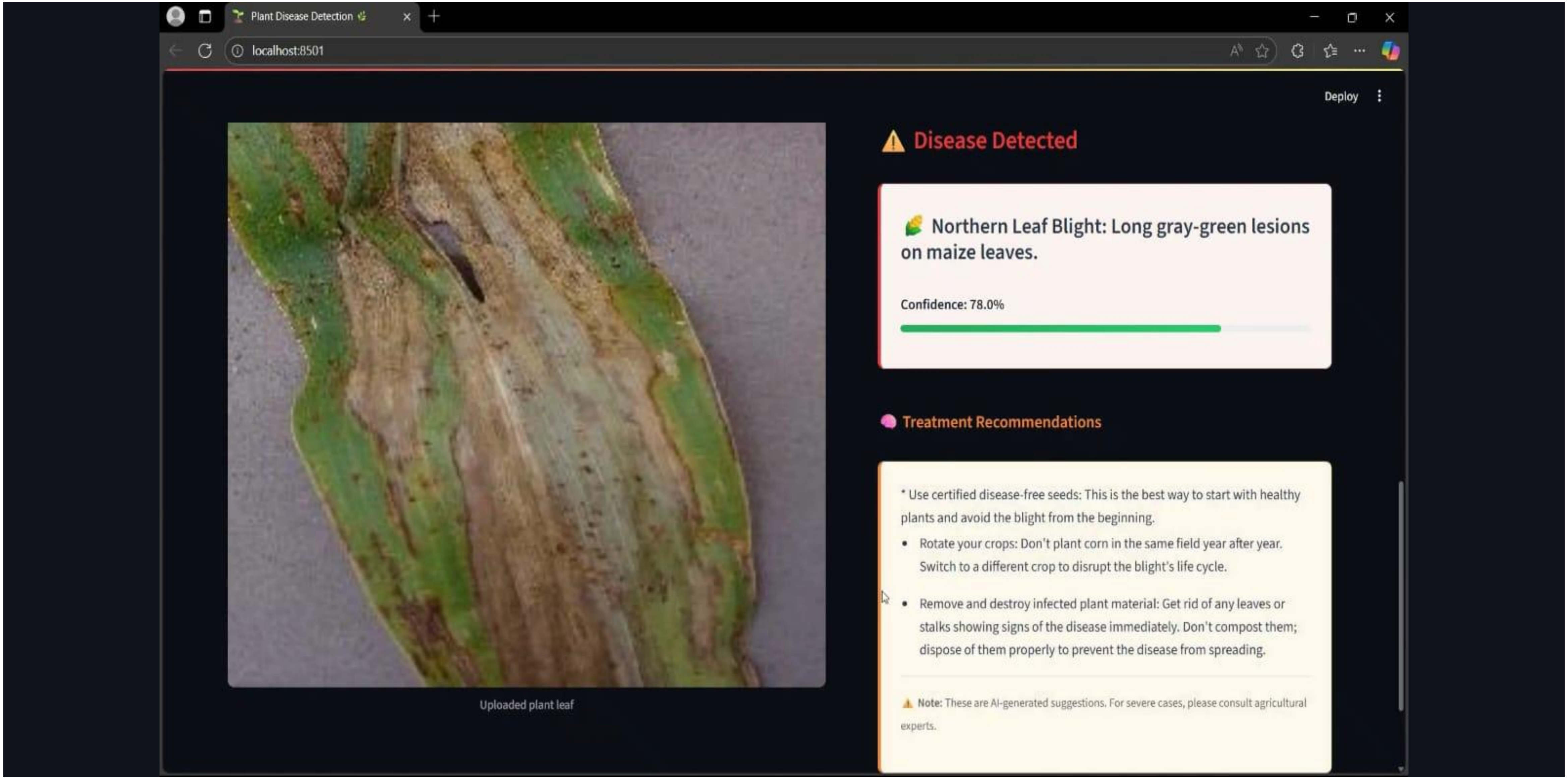
Save the Trained Model: The trained model is saved to a file (plant_disease_prediction_model.h5), allowing it to be loaded for future predictions.

We'll use the file i.e. **.h5** file in our fronted so that user interact with our site and upload the images then the model predict whether the disease was there or not if the disease is detected then along with prediction an AI suggestion message automatically generated. (we used Gemini 1.5 API (text-text) so it'll generate suggestions accordingly) . And we used Streamlit a Python Library to implement frontend and logical operations.

Flow Diagram

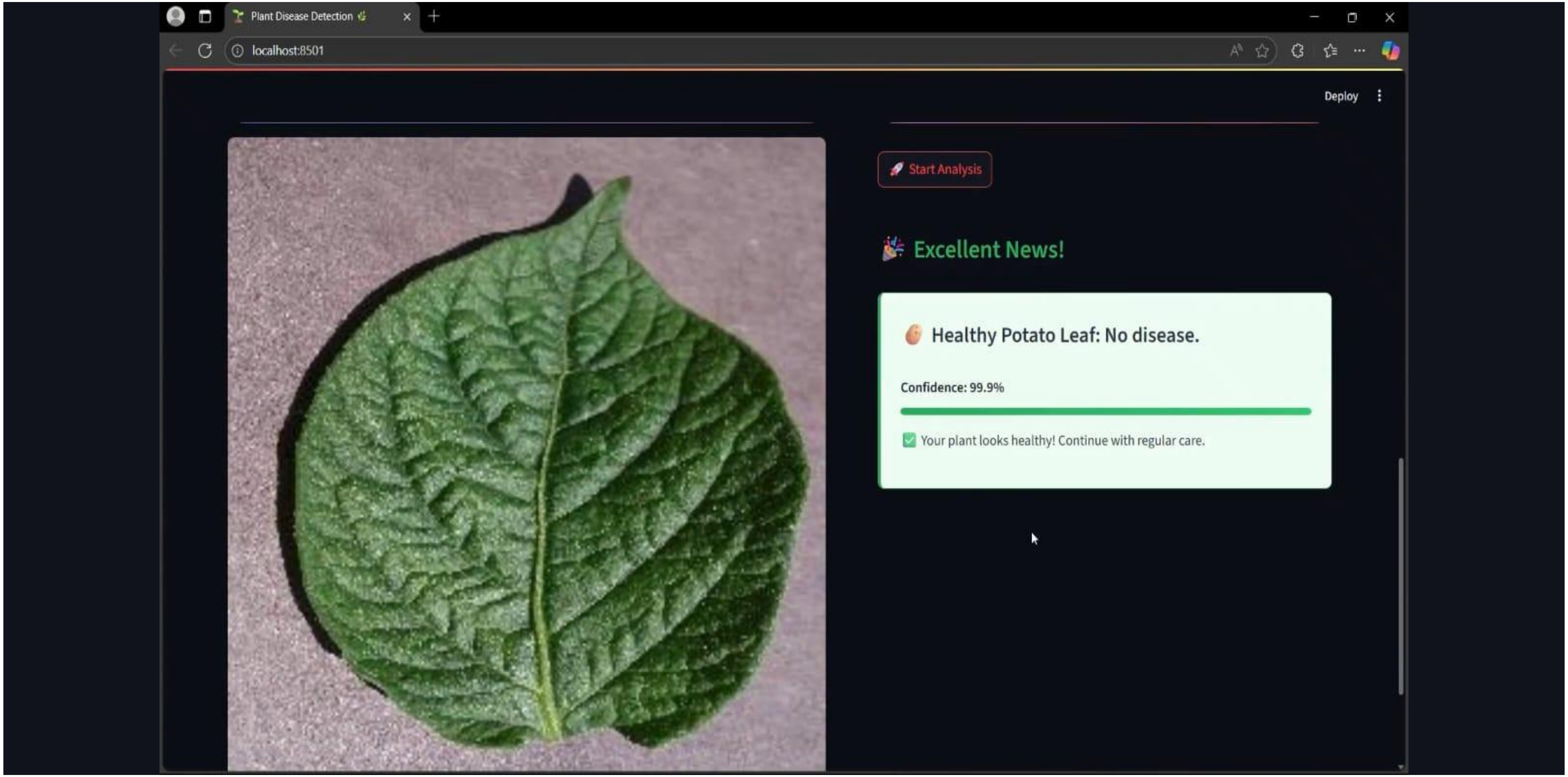


Outputs:



Plant Disease Detection
localhost:8501
Deploy
Disease Detected
Northern Leaf Blight: Long gray-green lesions on maize leaves.
Confidence: 78.0%
Treatment Recommendations
* Use certified disease-free seeds: This is the best way to start with healthy plants and avoid the blight from the beginning.
• Rotate your crops: Don't plant corn in the same field year after year. Switch to a different crop to disrupt the blight's life cycle.
• Remove and destroy infected plant material: Get rid of any leaves or stalks showing signs of the disease immediately. Don't compost them; dispose of them properly to prevent the disease from spreading.
Note: These are AI-generated suggestions. For severe cases, please consult agricultural experts.
Uploaded plant leaf

Outputs:



Plant Disease Detection
localhost:8501
Deploy
Start Analysis
Excellent News!
Healthy Potato Leaf: No disease.
Confidence: 99.9%
Your plant looks healthy! Continue with regular care.

References:

- Real-Time Plant Leaf Disease Detection using CNN and Solutions to Cure with Android App https://ieeexplore.ieee.org/document/10425034
- Using Deep Learning for Image-Based Plant Disease Detection https://arxiv.org/abs/1604.03169
- PlantXViT: Explainable Vision Transformer Enabled CNN for Plant Disease Identification https://arxiv.org/abs/2207.07919
- LeafGAN: An Effective Data Augmentation Method for Practical Plant Disease Diagnosis https://arxiv.org/abs/2002.10100
- A hybrid framework using CNNs and Vision Transformer (ViT) https://link.springer.com/article/10.1007/s40747-024-01764-x
- Soybean Disease Detection via Interpretable Hybrid CNN-GNN (CNN) https://arxiv.org/abs/2503.01284

THANK YOU

~ SAI BHASKAR NANDURI