Machine Learning in Real-Time Crop Monitoring and Management

A PROJECT REPORT Early Stage Disease Detection

Submitted by

Abhay Vershwal (21BCS6347)

Vansh Yadav (21BCS6374)

Vaibhavi Joshi (21BCS8330)

Abhimanyu Saini (21BCS8255)

NAME OF THE CANDIDATE(S)
Abhay Vershwal
Vansh Yadav
Vaibhavi Joshi
Abhimanyu

in partial fulfillment for the award of the degree of

NAME OF THE DEGREE BE.CSE

IN

BRANCH OF STUDY

CSE-AIML

Chandigarh University

November , 2024

BONAFIDE CERTIFICATE

Certified that this project report **SMART AGRICULTURE** is the bonafide work of **Abhay Vershwal, Vansh Yadav, Vaibhavi Joshi and Abhimanyu Saini** who carried out the project work under my/our supervision.

SIGNATURE

(HEAD OF THE DEPARTMENT)
AIT-CSE (AIML)

SIGNATURE

(SUPERVISOR)

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS

List of Figures

List of Tables

INTRODUCTION

1. Identification of Client /Need / Relevant Contemporary issue

Agriculture is the backbone of one's nation because it not only provides food but also many enormous resources such as cotton, fibers and many others. Coming to India, India is the world's second largest producers(2nd) of wheat and rice which provides food security as well as exporting it enhances the economy, India has the world's Sixth largest economy(6th) and its 15%(approx) comes from the agriculture sector. When we know how important our agriculture sector is it becomes vital to protect it and provide measures so as to get the best out of this sector. As per reports every year 15-25% of the agricultural production is destroyed by pest, weed and other diseases, which bring us to our project "Early Stage Crop Disease Detection Using Machine Learning Algorithms"in Smart Agriculture.

According to a study by the Associated Chambers of Commerce and Industry of India, annual crop losses due to pests and diseases amount to Rs.50,000 crore ($500 billion), which is significant in a country where at least 200 million Indians go to bed hungry every night. The value of plant science is therefore huge.

Take pulse crops: Indians rely heavily on chick peas, pigeon peas, mung beans and lentils for their daily protein requirements, and these are an essential ingredient in many native dishes, but it is estimated that without crop protection products the pulse crop yield can fall by around 30%.

2. Identification of Problem

As we have seen devastating figures we should roll our technologies and science in the traditional farming area as per reports are horrifying which is hollowing our country at a high rate . For example-

Scientists, led by Sachin S. Gunthe, gathered data from media reports, official reports, online summaries, and local observers between 1998 and 2018. This led to the generation of a data-set of more than 4,000 records of fungi-related crop diseases in India.
They found that 69 disease-causing fungal attacks negatively affected 39 different crops over the past 20 years. During this period, the fungal species Puccinia striiformis affected wheat crop on 12 occasions. This is in addition to the impact of other disease-causing fungal species to wheat. Nine different fungal species, including Pyricularia oryzae , were found to affect rice crops.

This shows how dangerous one disease can be and there is a need to quickly stop these type of different disease in different types of plants. As it is creating a huge trouble for our hard working farmers

3. Identification of Tasks

Identifying the problem:

The first step is to identify the problem of early stage crop disease detection and define the scope of the solution. In our case we are identifying the images and doing a full process training on our model to identify it is being infected or not and if it is infected then which category do it belongs to .

Building the solution:

The second task is building the solution which involves developing a CNN-based model that can accurately detect the early stage of crop diseases. This task includes several sub-tasks such as collecting the relevant data, preprocessing the data, designing the CNN architecture, training the model, and optimizing the hyperparameters ,testing the result and implementation in the real world .

Testing the solution:

The third task is to evaluate the performance of the CNN model and determine its accuracy in identifying crop diseases. This task includes testing the model on a set of unseen data, evaluating the performance metrics such as accuracy, precision, recall, and F1-score, analyzing the results, and fine-tuning the model if necessary. And lasting plotting or visualization of solution via graph to make it more understanding for the user

In summary, identifying the problem involves researching and understanding the requirements and limitations of early stage crop disease detection, building the solution involves designing and training a CNN model, and testing the solution involves evaluating the performance of the model and fine-tuning it if necessary.

4. Timeline

Identifying the problem: The first step is to define the problem and identify the specific crop diseases that the solution will detect. This requires knowledge of the common crop diseases in the area and the symptoms they cause.This task involves researching the existing solutions, analyzing the requirements of the stakeholders, identifying the limitations of the current methods, and understanding the challenges associated with crop disease detection.

Gathering and labeling data: The second step is to gather a large dataset of labeled images of healthy crops and crops affected by the identified diseases. The images should be labeled with the appropriate disease category.

Pre-processing and augmenting data: The third step is to preprocess and augment the data to prepare it for training. This involves re-sizing, normalization, and augmentation of the images to increase the dataset size and diversity.

Developing the CNN model: The fourth step is to design the CNN architecture, which involves selecting the appropriate number and type of layers, activation functions, and optimization algorithms.

Training the model: The fifth step is to train the model using the preprocessed and augmented data. This involves running the images through the network, adjusting the weights, and backpropagating the error to improve the accuracy.

Evaluating the model: The sixth step is to evaluate the model's performance using a separate test dataset that the model has not seen before. This involves calculating metrics such as accuracy, precision, recall, and F1 score.

Fine-tuning the model: The seventh step is to fine-tune the model based on the evaluation results. This involves adjusting the hyperparameters such as learning rate, batch size, and number of epochs to improve the accuracy of the model.

Deployment: The final step is to deploy the solution in a real-world setting. This involves integrating the model into a larger system and developing a user interface to facilitate its use.

5. Organization of the Report

As we know the importance of agriculture, it is our honourable duty to take some steps for the improvement of this sector which will directly put an positive impact on our country's economy and development. So We are working on our model which will detect the disease in the early stage and tends to eleminate it so that it can't be spread and make no harm to our prosporous crop. We have a aknowledged data which will help us to tackle different kinds of diseases and the appropriate solution for every disease. It will help the golden hands of our country as they do a lot of hard work for their crop and if it gets infected it cause a lot of trouble to them so we tried to provide a method which will help our farmers and get the best of their harvest.

Our project works on the model that first we take a survey of the entire field using drone camera taking detail pictures for our data then testing the pictures for every minute detail, once get the data our model(using CNN,Open cv softwares) works on finding the exact disease and at exact positions on the feild. Once get the disease and infected area we tend to find the exact solution and then spray(pesticides and medicines) over the infected field using drone. It reduces the work for the farmer and also on our testing we got an accuracy of up-to 95%, which can be very efficient if implemented.

C

CHAPTER 2 LITERATURE REVIEW/BACKGROUND STUDY

1. Timeline of the reported problem

The use of image processing in agriculture is becoming increasingly important, as it can help detect defects and diseases in crops that may not be visible to the naked eye. This can help farmers prevent significant crop losses due to pests or other factors. Various techniques are used for disease recognition, including image acquisition, pre-processing, segmentation, feature extraction, and categorization. Some of the methods which are used are : Kmeans Clustering, GLCM & SVM, Otsu's Detection, CNN-ANN-KNN, Histogram Technique.

Firstly the images of healthy & unhealthy leaf are stored for experimentation & then they are sent for image pre-processing. Then K-means clustering is used for segmentation and feature extraction using GLCM. Then SVM is used for classification . Otsu's detection is used to convert the rgb image of leaf into HSV(Hue,S aturation,Value). The classification approach is then carried out by KNN,ANN & CNN. The KNN method classifies samples by finding the nearest distance between trained and testing subjects.ANN method is used as a classifier.

One of the earliest systems on early stage crop disease detection was developed by Singh et al. (2016) [2]. The system used image processing techniques to analyze images of plants infected with different types of diseases. The system achieved an accuracy of 85% in detecting different types of diseases.

In 2017, Khan et al. proposed a system that used machine learning algorithms to detect the presence of wheat rust disease. The system achieved an accuracy of 98.5% in detecting the disease. The authors also reported that the system could be used to detect other types of crop diseases with some modifications.

In 2018, Wei et al. [3] proposed a system that used deep learning algorithms to detect tomato diseases. The system used a convolutional neural network (CNN) to classify images of tomato leaves into different categories based on the type of disease present.

In 2019, Wang et al. [4] proposed a system that used hyperspectral imaging to detect apple diseases. The system used a support vector machine (SVM) algorithm to classify the hyperspectral images of apple leaves into healthy and diseased categories. The system achieved an accuracy of
94.1% in detecting apple diseases.

In 2020, Wang et al. [5] proposed a system that used machine learning algorithms to detect maize diseases. The system used a combination of image processing and machine learning techniques to analyze images of maize leaves and detect the presence of diseases. The system achieved an accuracy of 95.1% in detecting different types of maize diseases.

In 2021, Zhang et al. [6] proposed a system that used deep learning algorithms to detect wheat diseases. The system used a CNN to analyze images of wheat leaves and classify them into different categories based on the type of disease present. The system achieved an accuracy of 96.2% in detecting different types of wheat diseases.

2. Existing solutions

The model is based on the IP and ML approaches for detection of leaf disease in presented in this section. The proposed system: (DWT+PCA+LBP+GLCM+CNN). The DWT, PCA and GLCM are used to extract the informative regions/features of the samples. In the next stage as a part of machine learning approaches the SVM, KNN and CNN are used to classify the features and the performance of the model is recorded.

First we will be taking the dataset and resize it to (256px x 256px) to maintain the aspect ratio.

Then we feed the image into machine learning algorithm where the following work takes place :

1.Dataset

All the images of the crops/leafs are stored in one particular file. Images of healthy and unhealthy images are then fed into the machine learning.

2.Preprocessing

Kmean clustering is applied on the images to find the infected region. This algorithm is mainly used to get to the center of the image and make the clusters of that image and then measure the distance from the center of the image till the different cluster.

Then we use PCA to denoise the data and reduce any of the unwanted dataset in the image present.

3.Feature Extraction

In feature extraction we use the algorithms which are used to extract the leaf patterns of the leafs.

1.LBP

LBP is used to extract the texture characteristics of the surface of the leafs, which can later be plotted into a histogram.

2.GLCM

This is used to extract statistical measures of the image.The optimal features are selected which are obtained from wavelet decomposition is then carried out .GLCM uses in the distribution of higher order of gray values are defined with neighborhood cr iterion. The several properties are derived from the GLCM technique for extraction of the leaf features. The features obtained using GLCM ,KNN-Clustering,LBP are combined to form feature vector which are provided as an input sample to the classifiers to recognize classify the images.

3.Classification

The technique such as CNN and KNN are used for classifying the samples. The CNN is a type of ANN which is designed to process the data. The architecture of CNN incused input,output and hidden layers, which are multiple in its nature.

The confusion matrix for CNN having output class and target class.The progress of training samples of leaf features classified usin CNN to know the accuracy.

1. Evaluation of Leaf diease

The parameters such as Precession, Recall and F-measure for the proposed model is calculated and is given in the following equation :

$$\text{Precision Measure } (\%) = \frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}} \text{X100}$$

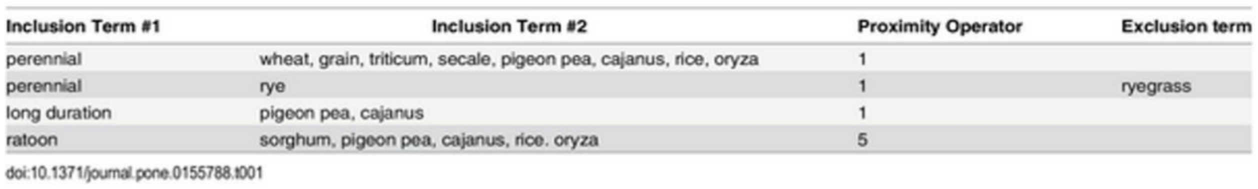
$$\text{Recall Measure } (\%) = \frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}} \text{X100}$$

$$\text{F} - \text{measure} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}} \text{X100}$$

3. Bibliometric analysis

We created an extensive annual library of plant literature using four databases: Web of Science, Scopus, ScienceDirect, and AGRICOLA. On June 3, 2015, we found 3026 documents, including published articles. Search is limited to journal articles, but there is no limitation for publication date. Call has two insertions associated with the corresponding operator Table 1. Initial inclusion is a modifier word or phrase that limits the results to the relevant journal, and the second word is perennial crop specific.

C



Inclusion Term #1	Inclusion Term #2	Proximity Operator	Exclusion term
perennial	wheat, grain, triticum, secale, pigeon pea, cajanus, rice, oryza	1	
perennial	rye	1	ryegrass
long duration	pigeon pea, cajanus	1	
ratoon	sorghum, pigeon pea, cajanus, rice. oryza	5	

doi:10.1371/journal.pone.0155788.t001

The proximity operator was set to "1" for the inclusion term "ratoon" and was increased to "5" for the inclusion term "rice" and "sorghum". The searches for rye were further refined to exclude results that reference perennial forage and weed perennial ryegrass. In choosing the search criteria, the authors also made certain inadvertent and intentional omissions. Searches were centered on American English to the neglect of foreign languages and other English dialects. Potentially useful terms like "perennial cereals," "rhizome," and "stolon" would have contributed additional material of relevance, as well as searches for emerging perennial crops like wild rice and kernza. However, as the authors were primarily interested in examining how historical literature potentially influenced current status, they chose to exclude emerging crops that did not have a sufficient development history, as well as extraneous terms that may have yielded interesting, relevant literature but were not directly related to the crops of interest.

We imported search results into Zotero and tagged bibliographic records with the name of the database and search that retrieved it. We then merged duplicate entries using Zotero's duplicate merge tool, and a manual merge was conducted on a small number of entries that had escaped detection. We also removed book chapters and author indices, stripped abstracts of copyright information and headings, and located missing abstracts using Digital Object Identifier and Google Scholar.

Our search criteria retrieved 2658 unique articles, but only 914 were deemed relevant. Articles specifically addressing the breeding or use of perennial crops for food production were retained, while articles that dealt only with their use for forage, silage, or bioenergy were eliminated.

Searches using the “ratoon” search term returned several papers about crops that were harvested, ratooned, and harvested for a second crop within the same 12 months. We chose to retain these articles because they included information highly applicable to the development of truly perennial crops and cropping systems.

Searches returned articles on perennial wild relatives of the crops of interest, but the relevance of these articles was difficult to discern. We retained articles on the traditional usage of wild relatives of domesticated crops, and articles on the genetics of wild relatives that dealt with sequencing the entire genome or investigating the genetic basis of specific plant traits that might make them useful perennials were retained. Articles on less relevant aspects of their genetics were excluded.

The search term "perennial grain" returned a handful of review papers that addressed the concept of perennial grain systems or utilized it in service of another argument. We exported the library from Zotero as a CSV file for analysis in R, and generated separate data frames for each crop. We then analyzed the metadata in each data frame to examine trends in publication, highlighting time periods of high publication activity and journals that publish such articles most frequently. Our R code is publicly available via GitHub.

4. Review Summary

PAPER	**Methodology**	**Future Work**	**ACCURACY VALUE**
1. Detection and classification technique of yellow vein mosaic virus disease in okra leaf image using naïve Bayesian classifier.	k- means clustering, Basic Morphological functions, Naïve Bayesian classifier , color co-occurrence method.	NIL	87%
2. Detecting Jute Plant Disease Using Image Processing and Machine Learning	Color co-occurrence methods, Multi SVM classifier.	NIL	86%
3. Cucumber disease detection using artificial	ANN, GICM(Gray level co-occurrence method)	Classification be accuracy c increased by usin	80.45%

neural network.		additional texture features.	

4. Detection and measurement of paddy leaf disease symptoms using image processing.	ANN, FUZZY classification, SVM, Kmeans algorithm, color co-occurrence method.	It evaluates the techniques in image processing, detecting diagnosing of crop leaf disease.	94.70%

5. Problem Definition

As the population increases, so do the challenges facing agriculture. Land constraints, Diseases and water scarcity play an important role in agriculture. Modern agriculture incorporates many technologies to solve these problems in real time.

* Field weed detection: Nearly 50% reduction in crop yield, mainly due to plant diseases and insect pests and reduced crop yield. Image processing techniques are mainly useful in disease analysis . Classifying plants under weeds helps detect weeds that provide greater growth.

Factors that cause plant diseases

Pathogens are the main cause of plant diseases. There is a department named after him called Plant Pathology, which primarily studies pathogens. There are two main factors that cause plant diseases and these are pathogens and environmental conditions.

Virus: It is a living organism with living cells inside that affects plants. Yellow streaks, yellow spots, leaf distortion and stunted growth are seen on virus infected areas of plant leaves and fruits.In cucumber, virus infections are mainly caused by cucumber mosaic viruses, which are contagious diseases that can be transmitted from plant to plant by insects or by contact. The best way to prevent viral illness is to treat the area affected by the virus.

Fungi: Fungi are also one of the main causes of reduced factory productivity. Ascomycetes and basidiomycetes are two main types of fungi that cause plant diseases. Fungicides are widely used to control fungal infections in plants.Magnaporthe grisea is commonly known as puffed rice. Sclerotinia is responsible for cotton rot. Oomycetes and phytomyxae are fungus-like organisms that harbor destructive pathogens in plants.

Bacteria: A bacterial infection occurs when a plant is infected by a microorganism. Bacteria are unicellular organisms.Burning, spotting and scabs can be seen in areas of the plant that are infected with the bacteria. Rust spots can spread quickly through plants. Tropical plants and vegetables are mainly affected by Fusarium wilt. Water uptake by plants is blocked by bacterial infection. Some bacterial plant pathogens are Burkholderia and Proteus.

6. Goals/Objectives

1) Therefore the goal of our study is to detect illnesses in crops from the start, and farmers will be able to detect diseases with the assistance of machine learning.
2) As a result, farmers should use pesticides to eliminate the illness as quickly as possible so that crop loss is minimised and crops stay safe. Now, with the use of drones, we will send high-resolution photographs of crops or utilize them in our software or app, where we will identify their disease or any sort of problem and notify them that it will occur soon. You can manage it by using this treatment or this method.
3) Will the farmer's profit from the technology in that they will learn about the illness in their crop at an early stage, and their crop will be protected from being completely destroyed by disease? Finally, they will not have to apply any sort of fertilizer or pesticide without knowing when and what type of these chemicals ought to be applied, and crops will be more natural, which is beneficial for human beings.
4) They will eventually not need to use unneeded pesticides since they will know when and what to use to save crops from illness.
5) And this will enhance agricultural productivity while also saving their land from unneeded pesticides, allowing them to plant a productive harvest in the future.

Chapter 3[1] DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features

1. We discovered the features in Gradation of Yellow Mosaic Virus Disease of Okra and Bitter Gourd based on Entropy-Based Binning and Naive Bayes Classifier after Identifying the Leaves where they are Applying Basic Techniques That Are Feature extraction and leaf identification made up the leaf identification stage. Disease Gradation Stage is divided into two phases, Training and Testing, with various steps in each phase. The initial stage in the training process is Leaf Specific Feature Extraction.

2. We learn about this characteristic in this work on Detecting Jute Plant Disease Using Image Processing and Machine Learning. They are using an Android application where the farmer takes photos and uploads them after which image processing is done using the following three techniques: enhancement, resizing, and noise removal. After that, hue-based segmentation is completed in the second step, and finally, the farmer extracts the feasture from the image where they obtain the mean variance after producing the result about the image.

3. Cucumber disease detection using artificial neural network. . In this paper, they set up an experiment where they used an ANN algorithm to detect disease. They started by using a microscope to find a detailed microscopic image, which automatically transferred to a computer where they used an ANN network to predict mathematical output. They are now using this network to provide output based on their data.

4. Detection and measurement of paddy leaf disease symptoms using image pro-cessing. The figures in this article are These come first. The process of picture acquisition begins with the gathering of photos, followed by image pre-processing that involves transforming RGB-based leaves into HSV models, segmenting leaves, and then extracting features of those leaves using SVM models. **Features list required in our solution**

1. Time effective
2. Accuracy should be high so the chances of error should be less.
3. User Friendly.
4. Cost Effective so every farmer should afford it and use it.

3.2. Design Constraints

1. *Cost:* The system should be cost-effective to build and operate, with a budget that is reasonable for farmers and other stakeholders. This could involve using affordable drones and cameras, and leveraging existing machine learning libraries and tools.

2. *Accuracy:* The system should have a high level of accuracy in detecting diseases at an early stage, with a low rate of false positives and false negatives. This could require using high-quality cameras, optimizing the feature extraction and machine learning algorithms, and validating the system against ground truth data.

3. *Speed:* The system should be able to detect diseases quickly, ideally in real-time or near-realtime, so that farmers can take action to prevent the spread of the disease. This could require using fast image processing algorithms and optimizing the system for low latency.

4. *Robustness:* The system should be able to operate in a range of environmental conditions, such as different lighting conditions, weather conditions, and crop types. This could require using adaptive machine learning algorithms that can handle variations in the data, and using durable hardware components that can withstand harsh environments.

5. *Ease of use:* The system should be easy to use and maintain, with a user-friendly interface and clear instructions for setup and operation. This could involve designing a web or mobile interface for farmers to access the results, and providing documentation and support for troubleshooting issues.

6. *Privacy and security:* The system should respect the privacy of farmers and protect the data from unauthorized access or misuse. This could involve using encryption to secure the data, implementing access controls and permissions, and following best practices for data handling and storage.

7. *Integration with existing systems:* The system should be able to integrate with existing farm management systems, such as irrigation and fertilization systems, to provide a holistic view of crop health and streamline operations. This could require using open standards and APIs for integration.

3.3. Analysis of Features and finalization subject to constraints

Here, various features are being eliminated, such as the conversion of REB or HSv mode in image segmentation, the removal of noise from the image dataset, and the use of ANN or SVm models in feature extraction. Then, we are making some changes to the existing model that we are using, such as CNN analysis or image processing, and we are adding some new features to our model that we use across all algorithms to provide the best accuracy and reduce errors in our results.

3.4. Design Flow

1. Data Collection: First, you need to collect data from your drone, which includes both imagery and other environmental data such as temperature, humidity, and light intensity. The imagery should be captured from different angles, heights, and perspectives to get a comprehensive view of the crops.

2. Data Preprocessing: After collecting the data, the next step is to preprocess it. This includes filtering out noise, adjusting the images for lighting and shadows, and removing any distortion caused by the drone camera's lens.

3. Feature Extraction: Once you've preprocessed the data, the next step is to extract features that can be used to identify different types of crops. This could include things like color, texture, shape, and size.

4. Model Training:With the features extracted, the next step is to train a machine learning model using a labeled dataset. This involves splitting the data into training and testing sets, selecting an appropriate algorithm, and tuning hyperparameters to improve the model's accuracy.

5. Prediction: Once the model is trained, it can be used to predict the type of crops in new images captured by the drone. This involves feeding the images into the model and analyzing the output to identify the crops.

6 .Analysis and Visualization: After the prediction step, you can analyze and visualize the results to gain insights into the health of the crops, such as identifying areas of poor growth or early signs of disease. This information can be used to make informed decisions about how to manage the crops and improve their yield.

3.5. Design selection

Design Option 1:
- Cost: Moderate
- Accuracy: High
- Speed: High
- Robustness: High
- Ease of use: Moderate
- Privacy and security: High
- Integration with existing systems: Moderate

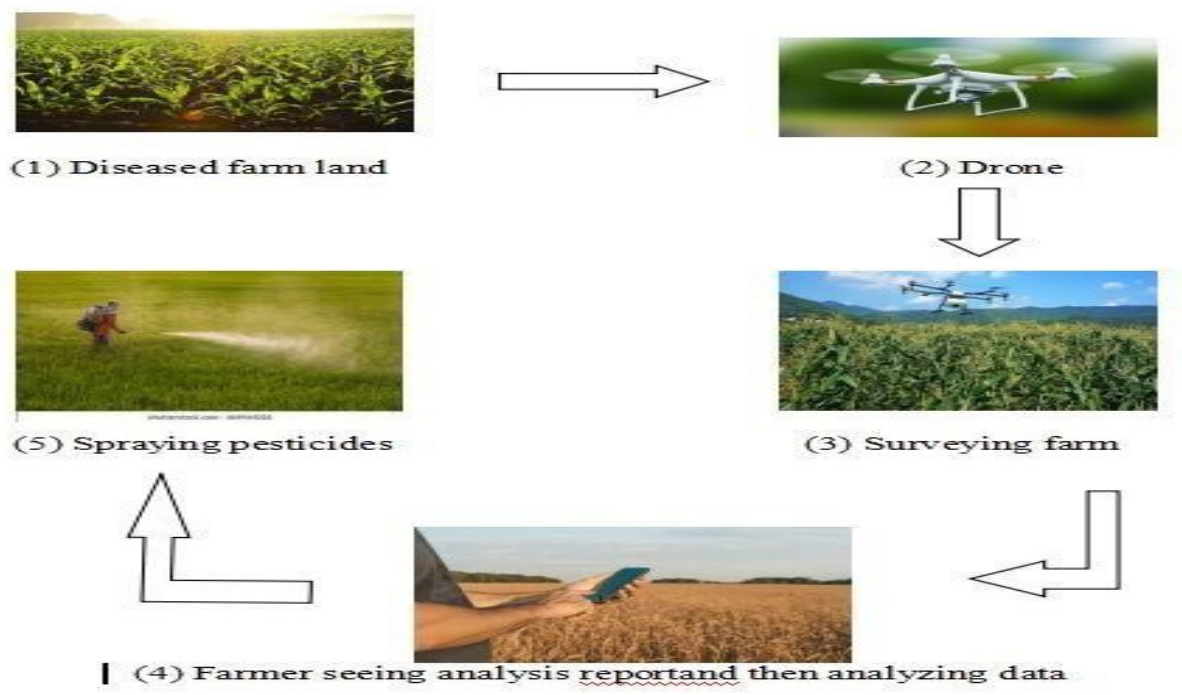
Design Option 2:
- Cost: Low
- Accuracy: Moderate
- Speed: Moderate
- Robustness: Low
- Ease of use: High
- Privacy and security: Moderate
- Integration with existing systems: High

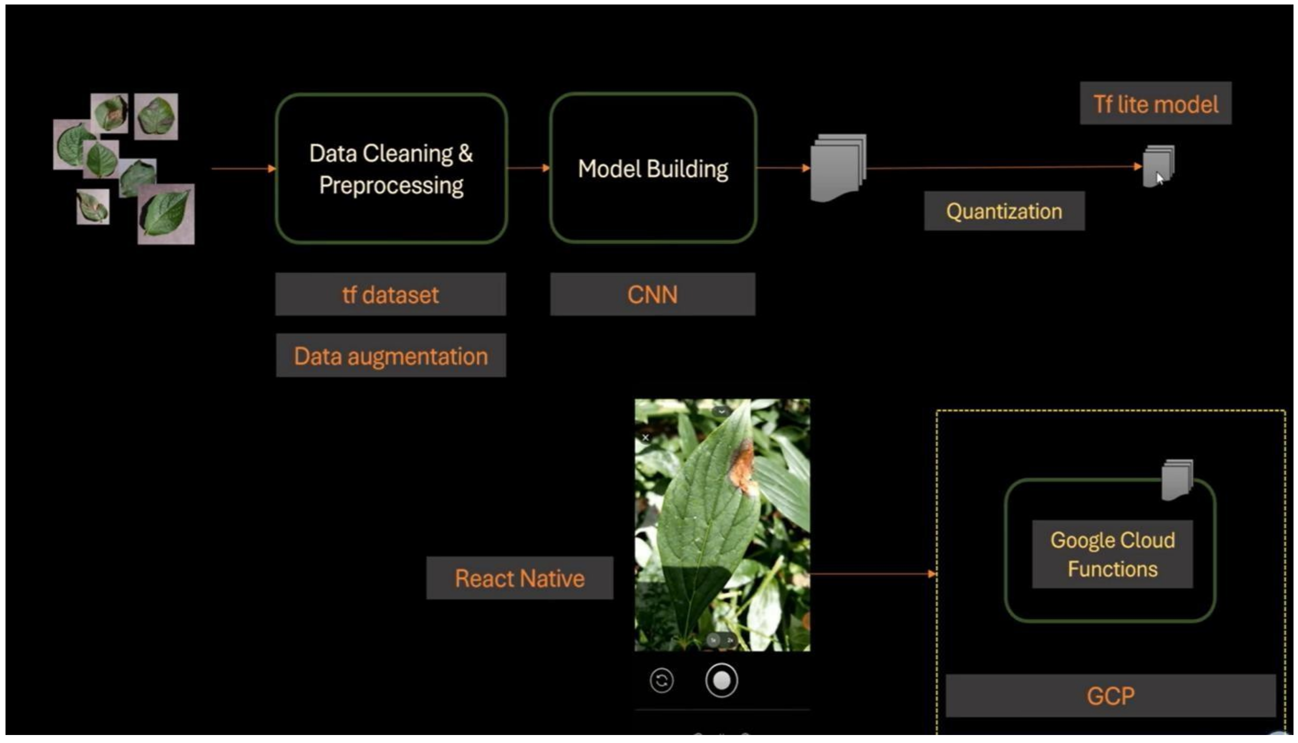
Comparison:

Design Option 1 has a higher accuracy and speed, and is more robust than Design Option 2. It also has good privacy and security features, but is slightly more expensive and may require more technical expertise to set up and use. Design Option 1 would be a good choice if accuracy and speed are the top priorities and if cost is less of a concern.

Design Option 2 has a lower cost and is easier to use than Design Option 1. It also has good integration with existing systems, which could be an important factor for farmers who want to streamline their operations. However, it has lower accuracy and speed, and is less robust, which could be a concern in some environments. Design Option 2 would be a good choice if cost and ease of use are the top priorities and if accuracy and speed are less.

3.6. Implementation plan/methodology


(1) Diseased farm land
(2) Drone
(3) Surveying farm
(4) Farmer seeing analysis reportand then analyzing data
(5) Spraying pesticides


Data Cleaning & Preprocessing
Model Building
Tf lite model
Quantization
tf dataset
CNN
Data augmentation
React Native
Google Cloud Functions
GCP

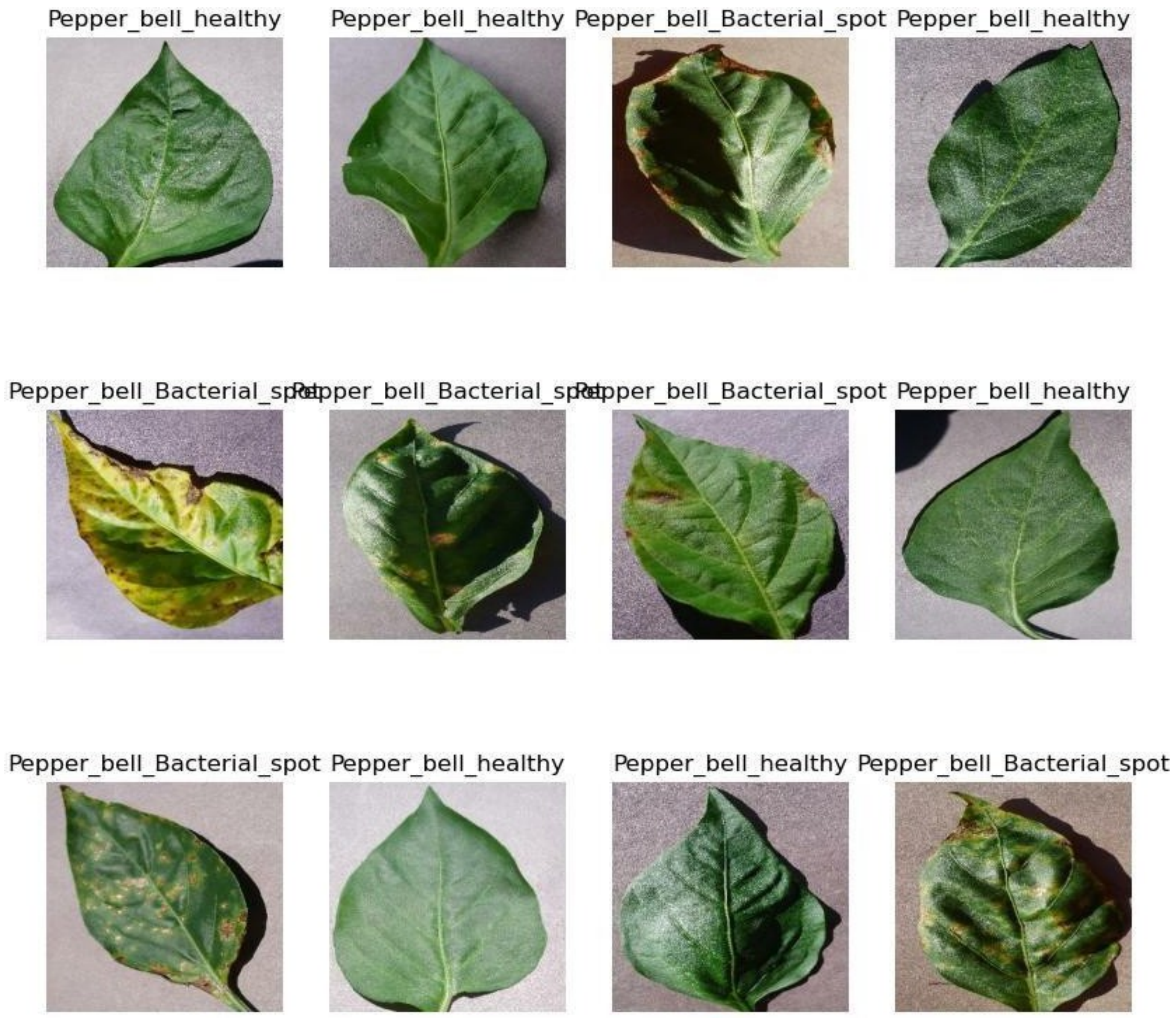
CHAPTER 4

RESULTS ANALYSIS AND VALIDATION

4.1. Implementation of solution

<u>Program code for Pepper Bell Plant:</u>

```
import matplotlib.pyplot as pltimport osimport tensorflow as tfimport numpy as npimport imghdrfrom tensorflow.keras import models, layersfrom IPython.display import HTML
In [2]:
BATCH_SIZE = 32IMAGE_SIZE = 256CHANNELS=3EPOCHS=25
In [3]:
trainingdataset=tf.keras.preprocessing.image_dataset_from_directory('Training
Dataset',seed=123,shuffle=True,image_size=(IMAGE_SIZE,IMAGE_SIZE),batch_size=BATCH_
SIZE)
Found 2475 files belonging to 2 classes.
In [4]: class_names =
trainingdataset.class_namesclass_names Out[4]:
['Pepper_bell_Bacterial_spot', 'Pepper_bell_healthy'] In
[5]:
plt.figure(figsize=(10, 10))for image_batch, labels_batch in trainingdataset.take(1):
for i in range(12):
    ax = plt.subplot(3, 4, i + 1)
plt.imshow(image_batch[i].numpy().astype("uint8"))
plt.title(class_names[labels_batch[i]])        plt.axis("off")
```


Pepper_bell_healthy
Pepper_bell_healthy
Pepper_bell_Bacterial_spot
Pepper_bell_healthy
Pepper_bell_Bacterial_spot
Pepper_bell_Bacterial_spot
Pepper_bell_Bacterial_spot
Pepper_bell_healthy
Pepper_bell_Bacterial_spot
Pepper_bell_healthy
Pepper_bell_healthy
Pepper_bell_Bacterial_spot

```
In [6]:
len(trainingdataset) Out[6]:
78
In [7]:
train_size = 0.8len(trainingdataset)*train_size Out[7]:
62.400000000000006
In [8]:
train_ds = trainingdataset.take(62)len(train_ds) Out[8]:
62
In [9]:
test_ds = trainingdataset.skip(64)len(test_ds) Out[9]:
14
```

In [10]:

val_size=0.1len(trainingdataset)*val_size Out[10]:

7.800000000000001

In [11]:

val_ds = trainingdataset.take(7)len(val_ds) Out[11]:

7 In

[12]:

len(train_ds) Out[12]:

62

In [13]:

len(test_ds) Out[13]:

14

In [14]: len(val_ds)

Out[14]:

7

In [15]:

```
def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True,
shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)      val_ds
= ds.skip(train_size).take(val_size)      test_ds
= ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds In
```

[16]:

```
train_ds, val_ds, test_ds = get_dataset_partitions_tf(trainingdataset) In
[17]:
len(train_ds) Out[17]:
62
In [18]: len(val_ds)
Out[18]:
7 In
[19]:
len(test_ds) Out[19]:
9
In [20]:
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)val_ds =
val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)test_ds =
test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE) In [21]:
resize_and_rescale =
tf.keras.Sequential([ layers.experimental.preprocessing.Resizing(IMAGE_SIZE,
IMAGE_SIZE), layers.experimental.preprocessing.Rescaling(1./255),]) In [22]:
data_augmentation =
tf.keras.Sequential([ layers.experimental.preprocessing.RandomFlip("horizontal
_and_vertical"), layers.experimental.preprocessing.RandomRotation(0.2),]) In
[23]:
train_ds = train_ds.map(
  lambda x, y: (data_augmentation(x, training=True),
y)).prefetch(buffer_size=tf.data.AUTOTUNE)
WARNING:tensorflow:From C:\Users\Asus\AppData\Roaming\Python\Python39\site-
packages\tensorflow\python\autograph\pyct\static_analysis\liveness.py:83: Analyzer.lamba_check
(from tensorflow.python.autograph.pyct.static_analysis.liveness) is deprecated and will be removed
after 2023-09-23. Instructions for updating:
Lambda fuctions will be no more assumed to be used in the statement where they are used, or at least
in the same block. https://github.com/tensorflow/tensorflow/issues/56089
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
registered converter for this op.
```

```
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
```

In [24]:

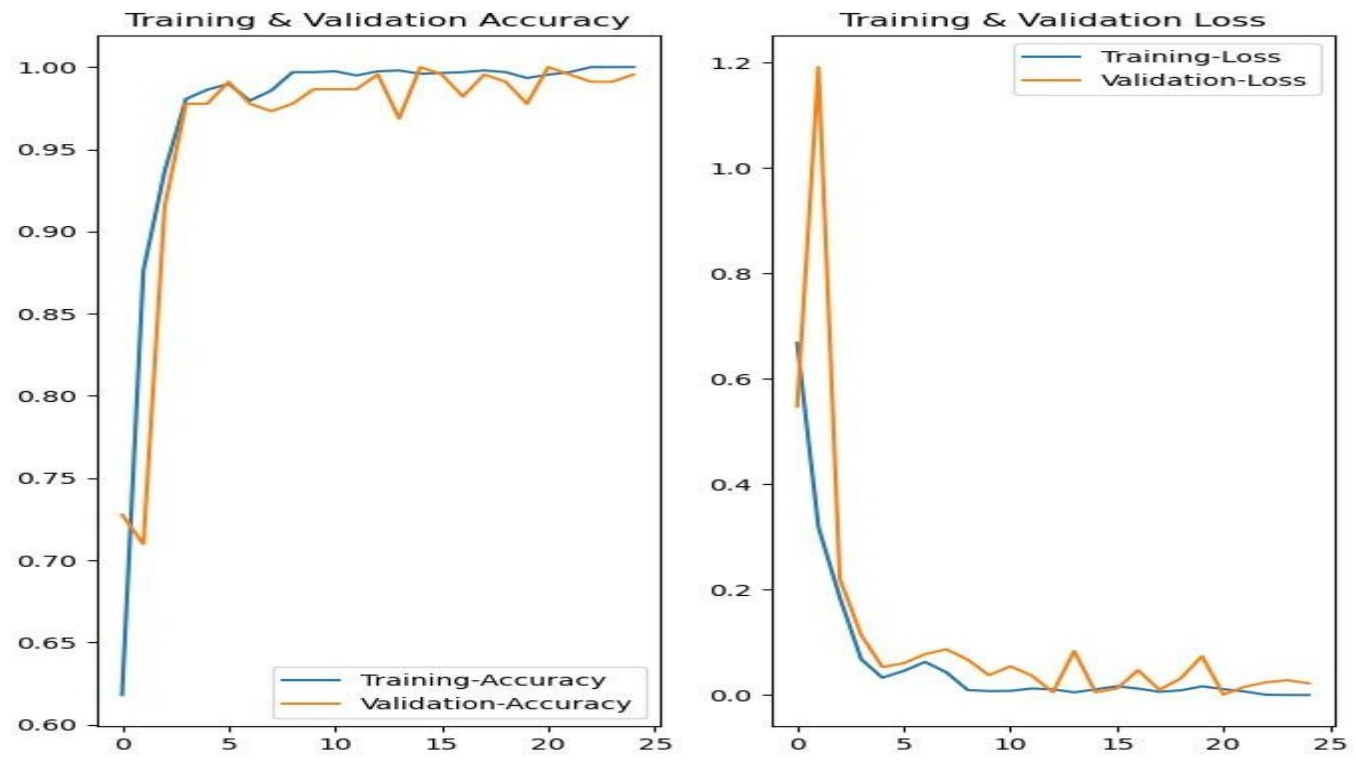
```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)n_classes = 3
model = models.Sequential([     resize_and_rescale,      layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),      layers.MaxPooling2D((2, 2)), layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),      layers.MaxPooling2D((2, 2)), layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),      layers.MaxPooling2D((2, 2)), layers.Conv2D(64, (3, 3), activation='relu'),      layers.MaxPooling2D((2, 2)), layers.Conv2D(64, (3, 3), activation='relu'),      layers.MaxPooling2D((2, 2)), layers.Conv2D(64, (3, 3), activation='relu'),      layers.MaxPooling2D((2, 2)), layers.Flatten(),      layers.Dense(64, activation='relu'),      layers.Dense(n_classes, activation='softmax'),]) model.build(input_shape=input_shape) In [25]:
model.summary() Model:
"sequential_2"
_________________________________________________________________

 Layer (type)                Output Shape              Param #
=================================================================

sequential (Sequential)     (32, 256, 256, 3)         0
```

```
conv2d (Conv2D)             (32, 254, 254, 32)896max_pooling2d (MaxPooling2D  (32, 127, 127, 32)      0
)

conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0          2D)

conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928

max_pooling2d_2 (MaxPooling  (32, 30, 30, 64)         0          2D)

conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928

max_pooling2d_3 (MaxPooling  (32, 14, 14, 64)         0          2D)

conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928

max_pooling2d_4 (MaxPooling  (32, 6, 6, 64)           0          2D)

conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928

max_pooling2d_5 (MaxPooling  (32, 2, 2, 64)           0         2D)


 flatten (Flatten)        (32, 256)                 0

  dense (Dense)            (32, 64)                  164
                                                     48

  dense_1 (Dense)          (32, 3)                   195


=================================================================

Total params: 183,747

Trainable params: 183,747
_________________________________________________________________
Non-trainable params: 0
```

```
In [70]:
model.compile( optimizer='
   adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
metrics=['accuracy']) In [71]:
history        =        model.fit(
train_ds,
   batch_size=BATCH_SIZE,
validation_data=val_ds,
verbose=1,     epochs=25,)
Epoch 1/25
62/62 [==============================] - 72s 1s/step - loss: 0.6686 - accuracy: 0.6179 -
val_loss: 0.5476 - val_accuracy: 0.7277
Epoch 2/25
62/62 [==============================] - 69s 1s/step - loss: 0.3185 - accuracy: 0.8762 -
val_loss: 1.1916 - val_accuracy: 0.7098
Epoch 3/25
62/62 [==============================] - 69s 1s/step - loss: 0.1853 - accuracy: 0.9368 -
val_loss: 0.2215 - val_accuracy: 0.9152
Epoch 4/25
62/62 [==============================] - 69s 1s/step - loss: 0.0683 - accuracy: 0.9806 -
val_loss: 0.1143 - val_accuracy: 0.9777
Epoch 5/25
62/62 [==============================] - 69s 1s/step - loss: 0.0332 - accuracy: 0.9862 -
val_loss: 0.0533 - val_accuracy: 0.9777
Epoch 6/25
62/62 [==============================] - 71s 1s/step - loss: 0.0460 - accuracy: 0.9898 -
val_loss: 0.0607 - val_accuracy: 0.9911
Epoch 7/25
62/62 [==============================] - 70s 1s/step - loss: 0.0631 - accuracy: 0.9796 -
val_loss: 0.0778 - val_accuracy: 0.9777
Epoch 8/25
62/62 [==============================]  - 68s 1s/step - loss: 0.0433 - accuracy: 0.9857 -

val_loss: 0.0871 - val_accuracy: 0.9732
```

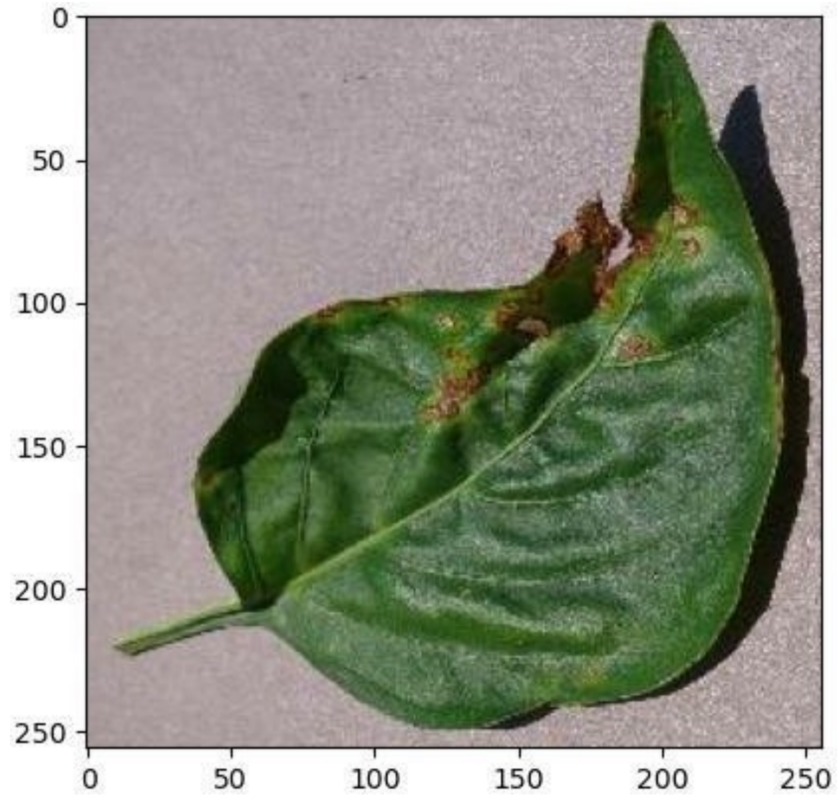
```
Epoch 9/25
62/62 [==============================] - 68s 1s/step - loss: 0.0098 - accuracy: 0.9969 -
val_loss: 0.0674 - val_accuracy: 0.9777
Epoch 10/25
62/62 [==============================] - 68s 1s/step - loss: 0.0078 - accuracy: 0.9969 -
val_loss: 0.0379 - val_accuracy: 0.9866
Epoch 11/25
62/62 [==============================] - 69s 1s/step - loss: 0.0082 - accuracy: 0.9975 -
val_loss: 0.0547 - val_accuracy: 0.9866
Epoch 12/25
62/62 [==============================] - 68s 1s/step - loss: 0.0127 - accuracy: 0.9949 -
val_loss: 0.0375 - val_accuracy: 0.9866
Epoch 13/25
62/62 [==============================] - 69s 1s/step - loss: 0.0111 - accuracy: 0.9975 -
val_loss: 0.0056 - val_accuracy: 0.9955
Epoch 14/25
62/62 [==============================] - 68s 1s/step - loss: 0.0054 - accuracy: 0.9980 -
val_loss: 0.0838 - val_accuracy: 0.9688
Epoch 15/25
62/62 [==============================] - 69s 1s/step - loss: 0.0112 - accuracy: 0.9959 -
val_loss: 0.0056 - val_accuracy: 1.0000
Epoch 16/25
62/62 [==============================] - 69s 1s/step - loss: 0.0169 - accuracy: 0.9964 -
val_loss: 0.0126 - val_accuracy: 0.9955
Epoch 17/25
62/62 [==============================] - 69s 1s/step - loss: 0.0126 - accuracy: 0.9969 -
val_loss: 0.0474 - val_accuracy: 0.9821
Epoch 18/25
62/62 [==============================] - 69s 1s/step - loss: 0.0065 - accuracy: 0.9980 -
val_loss: 0.0100 - val_accuracy: 0.9955
Epoch 19/25
62/62 [==============================]  - 70s 1s/step - loss: 0.0092 - accuracy: 0.9969 -
val_loss: 0.0321 - val_accuracy: 0.9911
Epoch 20/25
```

```
62/62 [==============================] - 69s 1s/step - loss: 0.0169 - accuracy: 0.9934 -
val_loss: 0.0742 - val_accuracy: 0.9777
Epoch 21/25
62/62 [==============================] - 69s 1s/step - loss: 0.0114 - accuracy: 0.9954 -
val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 22/25
62/62 [==============================] - 69s 1s/step - loss: 0.0071 - accuracy: 0.9969 -
val_loss: 0.0160 - val_accuracy: 0.9955
Epoch 23/25
62/62 [==============================] - 70s 1s/step - loss: 8.1576e-04 - accuracy:
1.0000 - val_loss: 0.0244 - val_accuracy: 0.9911
Epoch 24/25
62/62 [==============================] - 70s 1s/step - loss: 3.6768e-04 - accuracy:
1.0000 - val_loss: 0.0285 - val_accuracy: 0.9911
Epoch 25/25
62/62 [==============================] - 69s 1s/step - loss: 3.8279e-04 - accuracy: 1.0000
- val_loss: 0.0225 - val_accuracy: 0.9955 In [72]:
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy In
[73]:
pre = Precision()re = Recall()acc = BinaryAccuracy() In
[76]:
scores = model.evaluate(test_ds)
9/9 [==============================] - 2s 169ms/step - loss: 3.7601e-04 - accuracy:
1.0000 In [82]: acc = history.history['accuracy']val_acc = history.history['val_accuracy'] loss =
history.history['loss']val_loss = history.history['val_loss'] In [85]:
plt.figure(figsize=(8, 8))plt.subplot(1, 2, 1)plt.plot(range(EPOCHS), acc,
label='TrainingAccuracy')plt.plot(range(EPOCHS), val_acc, label='Validation-
Accuracy')plt.legend(loc='lower right')plt.title('Training & Validation Accuracy') plt.subplot(1, 2,
2)plt.plot(range(EPOCHS), loss, label='Training-Loss')plt.plot(range(EPOCHS), val_loss,
label='Validation-Loss')plt.legend(loc='upper right')plt.title('Training &
Validation Loss')plt.show()
```



In [43]:

```
import numpy as npfor images_batch, labels_batch in test_ds.take(1):

    first_image          =          images_batch[0].numpy().astype('uint8')
first_label = labels_batch[0].numpy()

    print("first image to predict")
plt.imshow(first_image)      print("actual
label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)      print("predicted
label:",class_names[np.argmax(batch_prediction[0])]) first image to
predict actual label: Pepper_bell_Bacterial_spot
1/1 [==============================]        -      0s      200ms/step
predicted label: Pepper_bell_Bacterial_spot
```



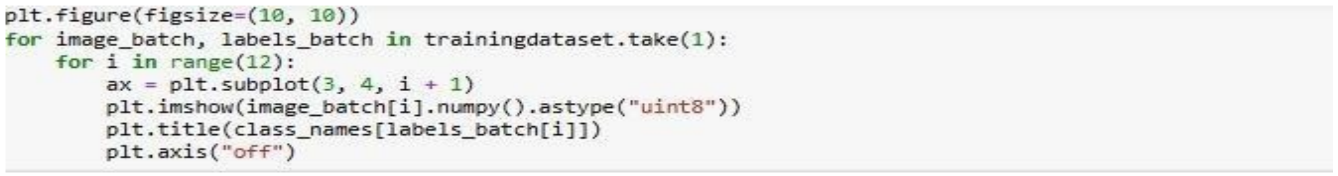
In

[ ]:

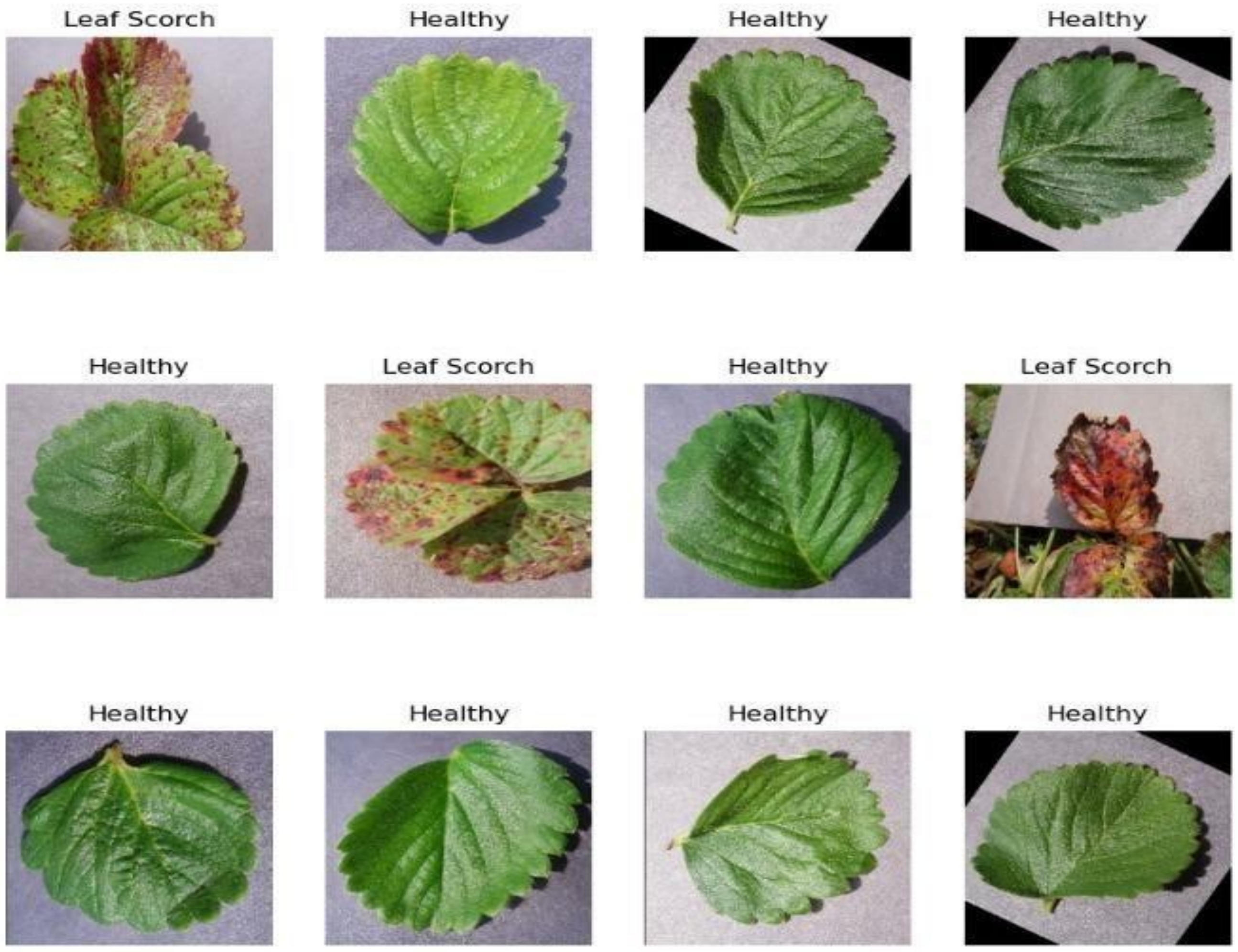
In [ ]:

<u>This code can work on different data sets such as:</u>

1. <u>Strawberry-</u>

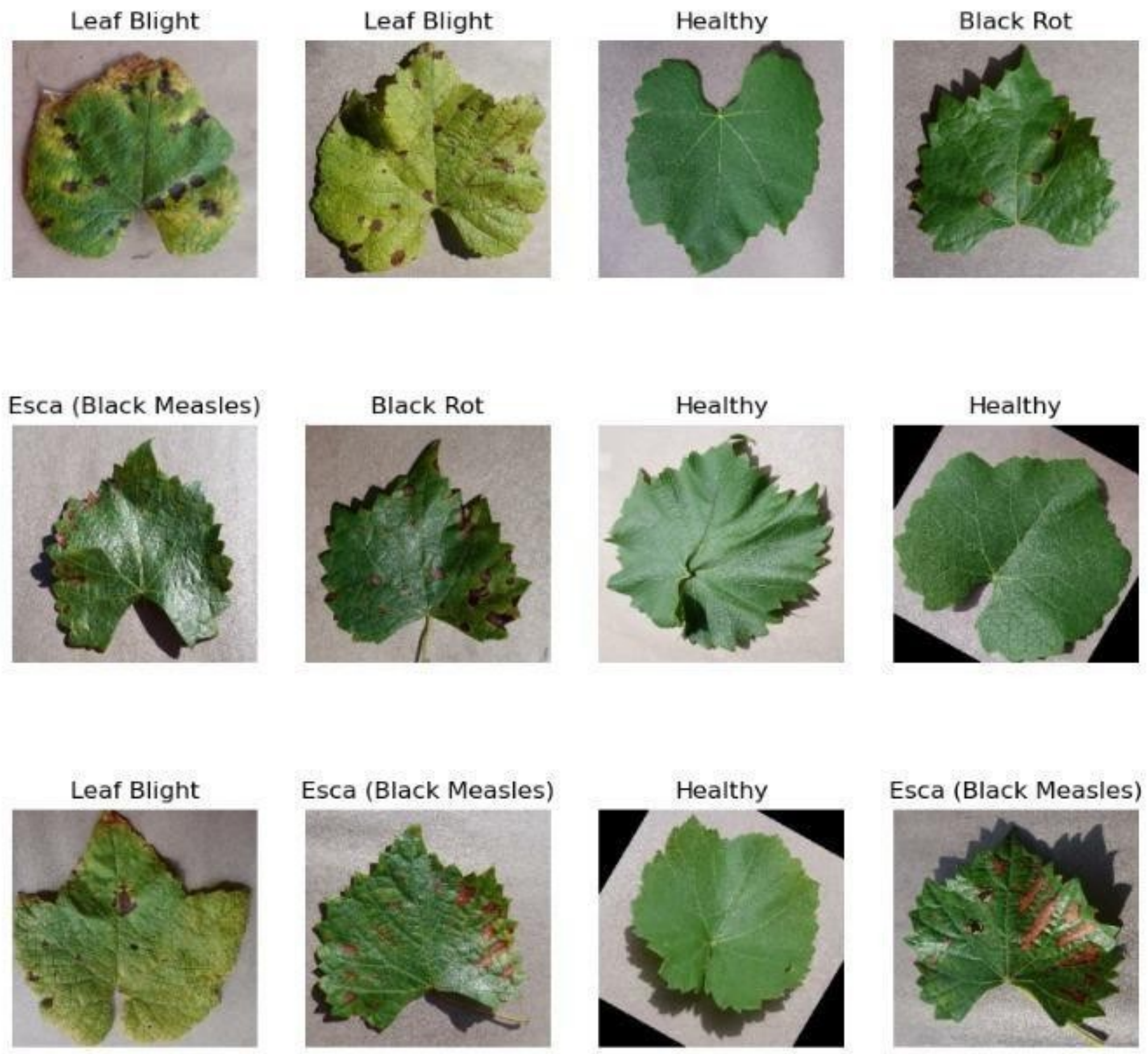


```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in trainingdataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```

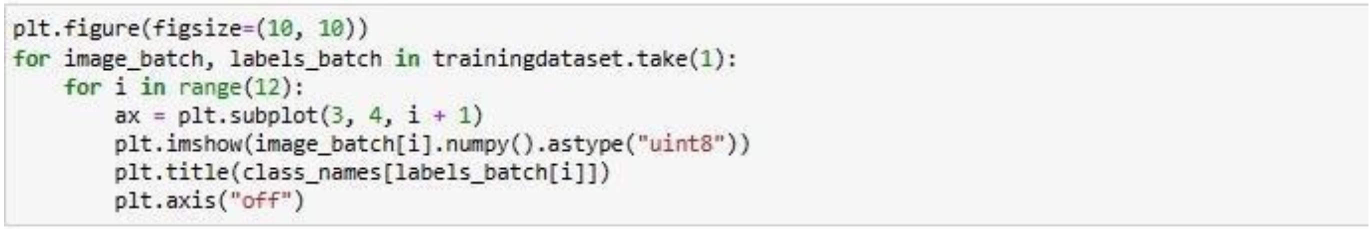


2. Grapes:

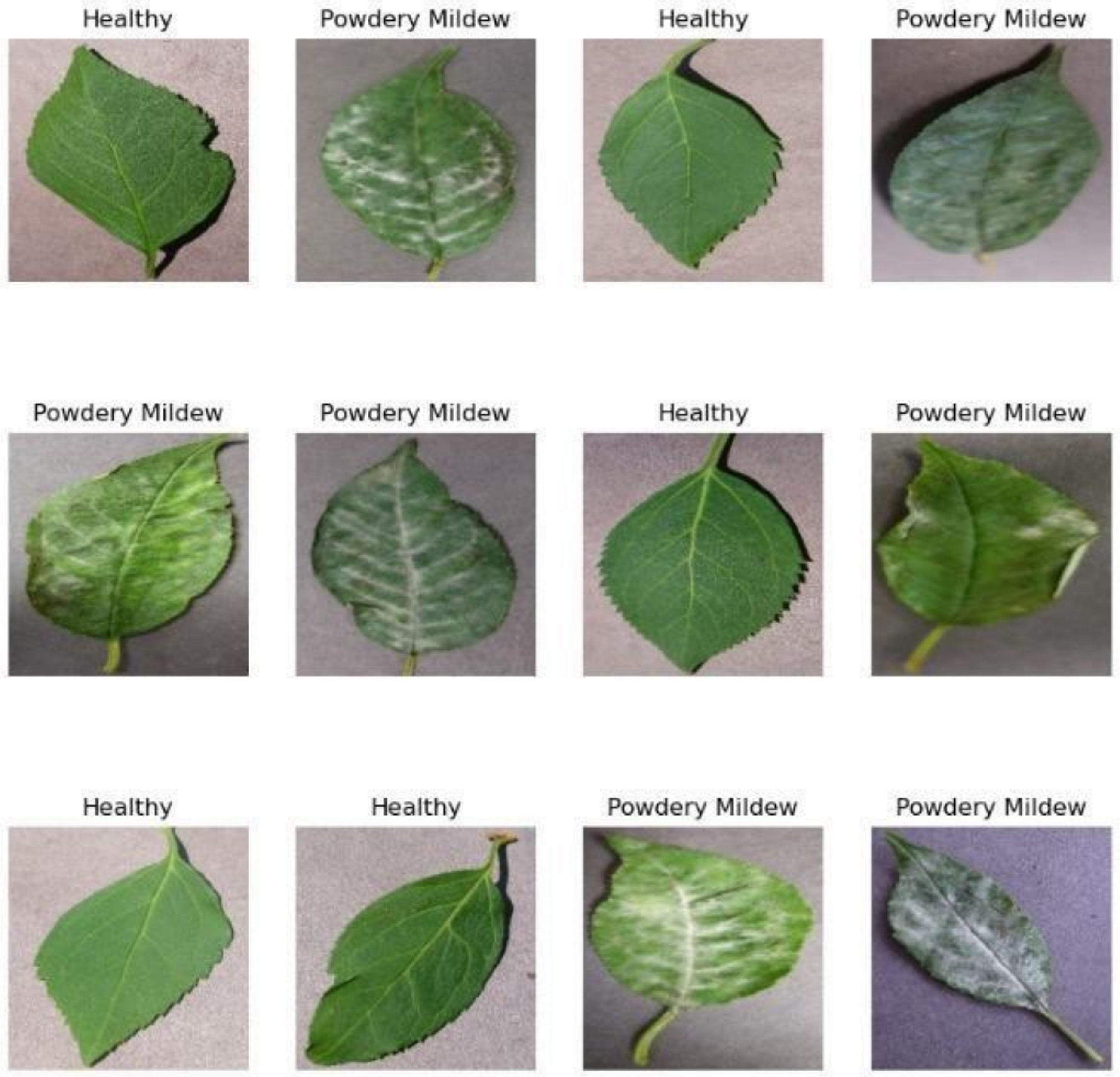
```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in trainingdataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



3. Cherry:

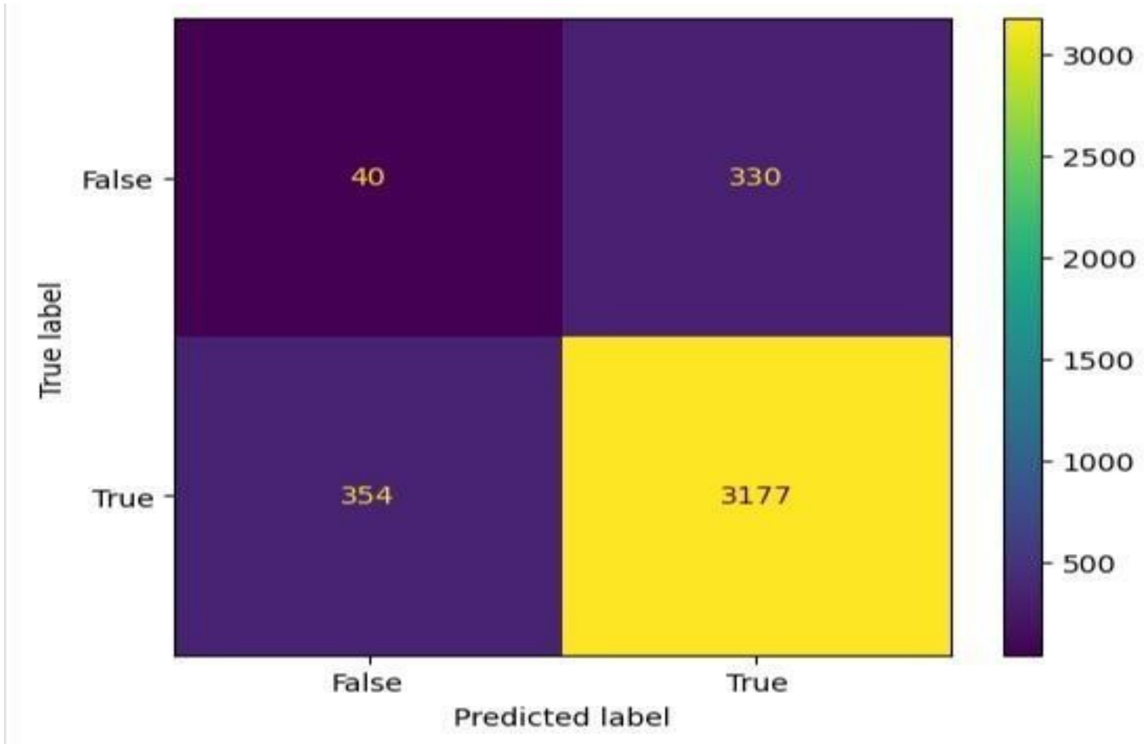


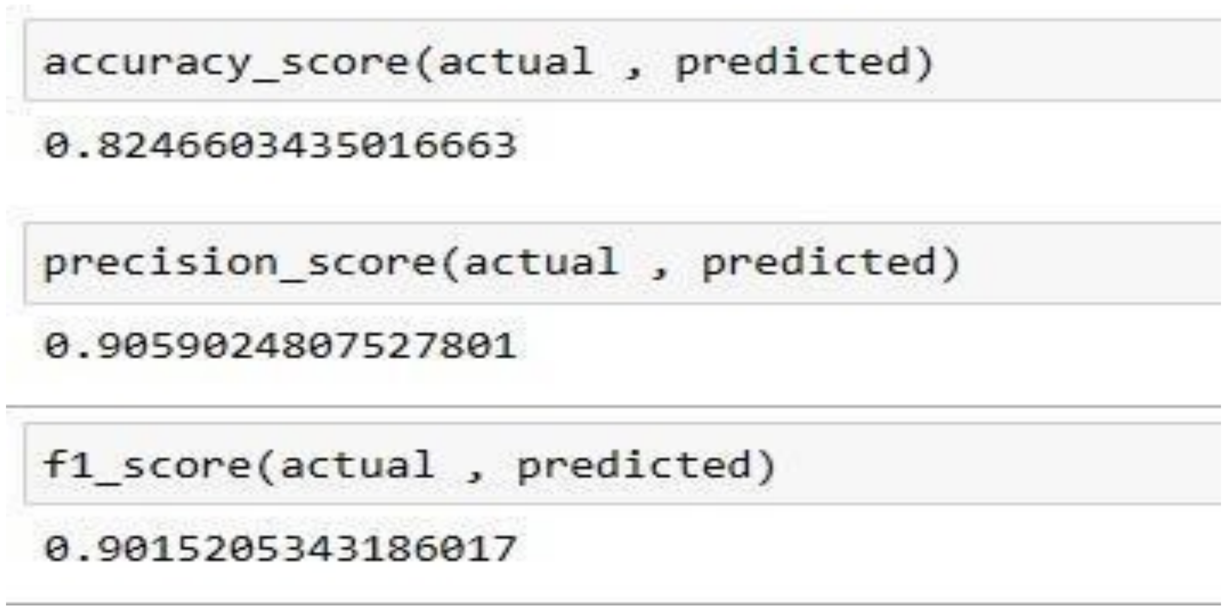
```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in trainingdataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



Obtained Confusion Matrices and Accuracy:

1. Pepper Bell-



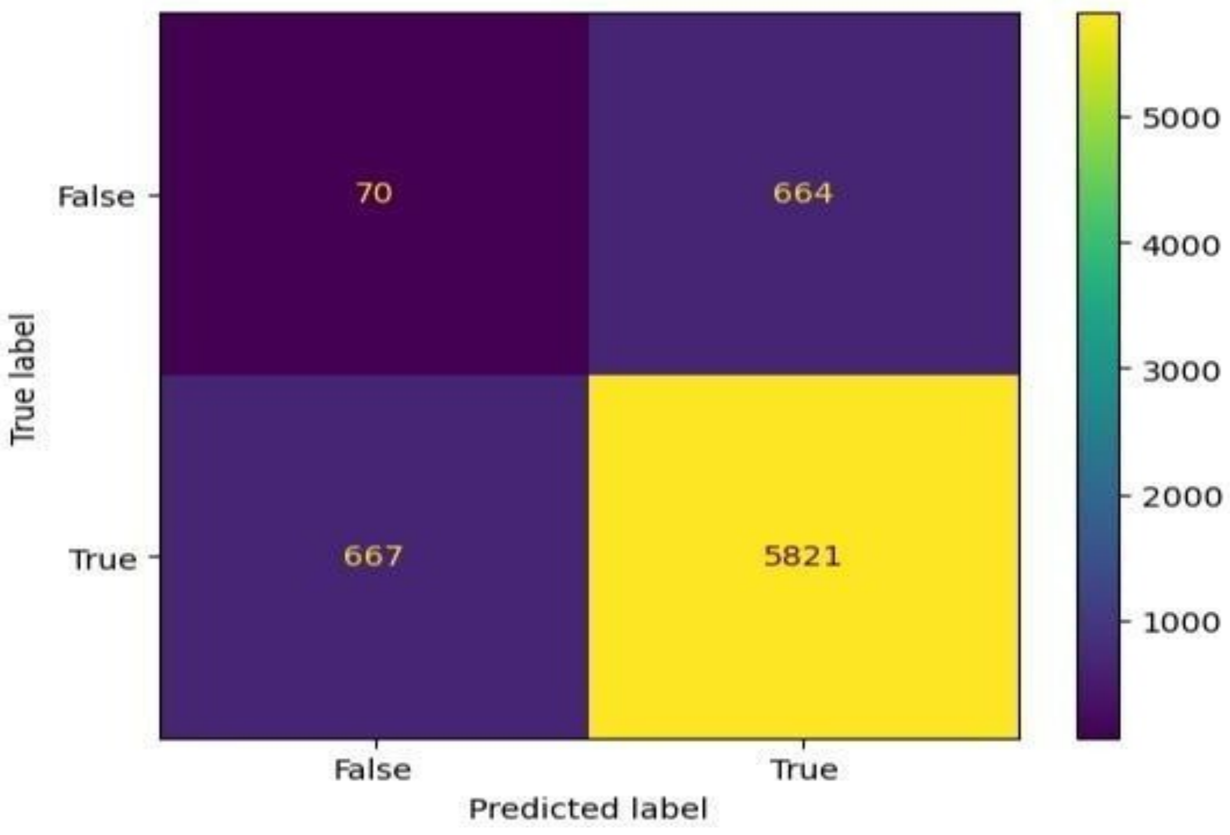


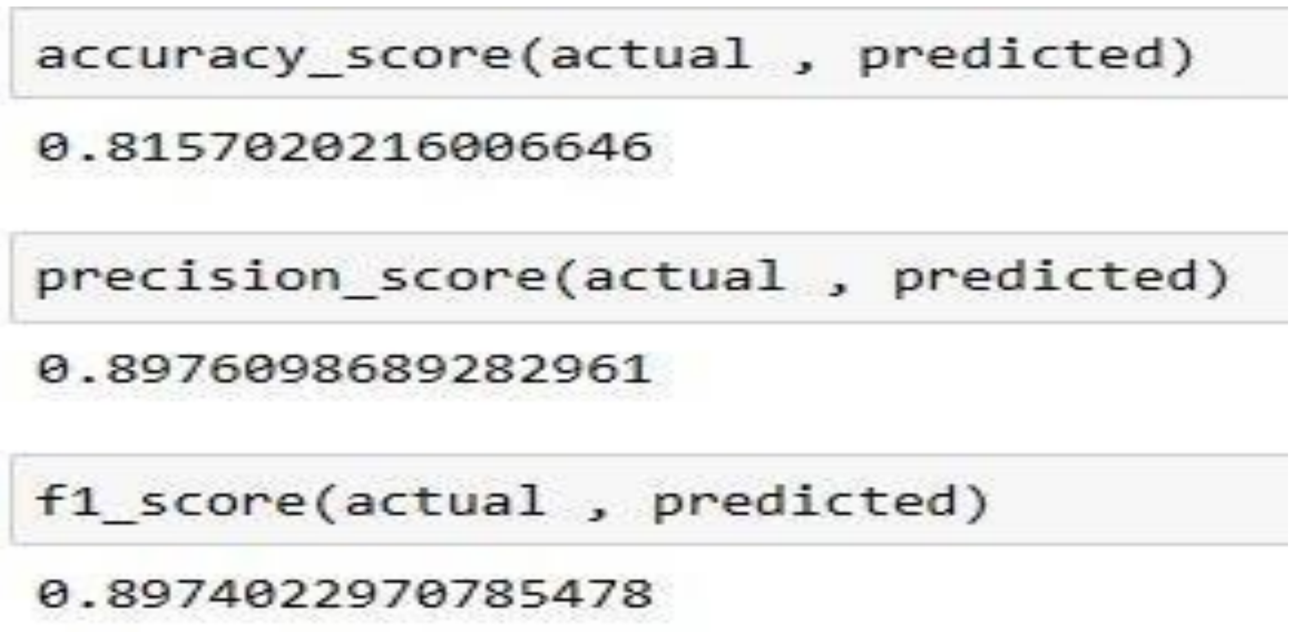
```
accuracy_score(actual , predicted)
0.8246603435016663
```

```
precision_score(actual , predicted)
0.9059024807527801
```

```
f1_score(actual , predicted)
0.9015205343186017
```

2. Strawberry:



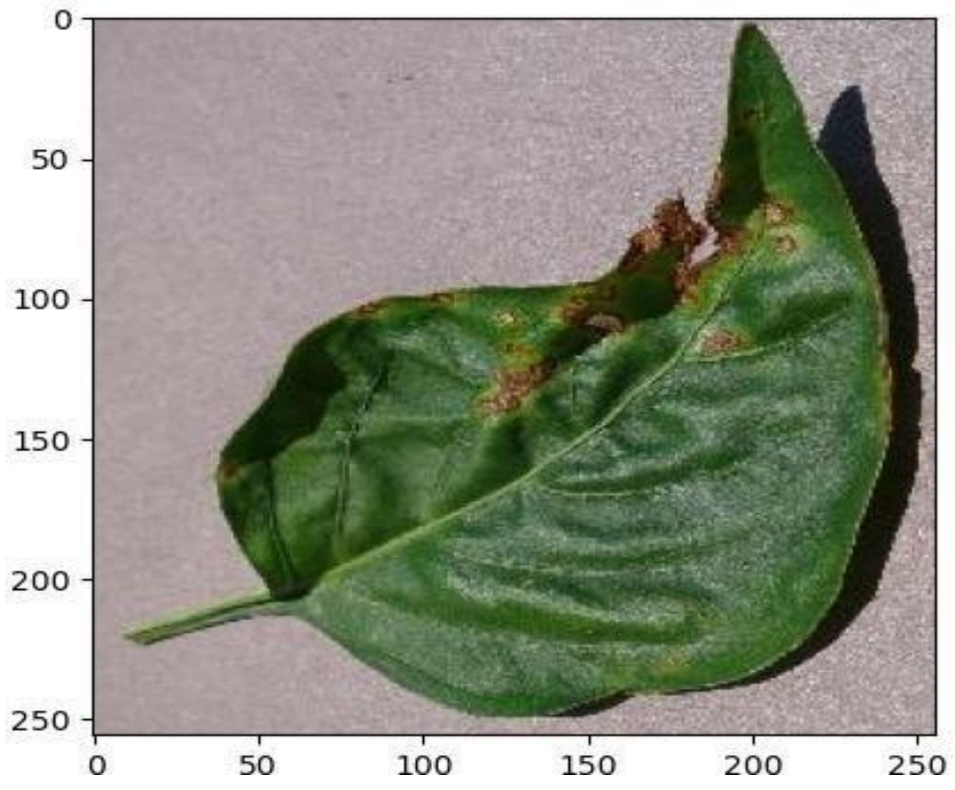


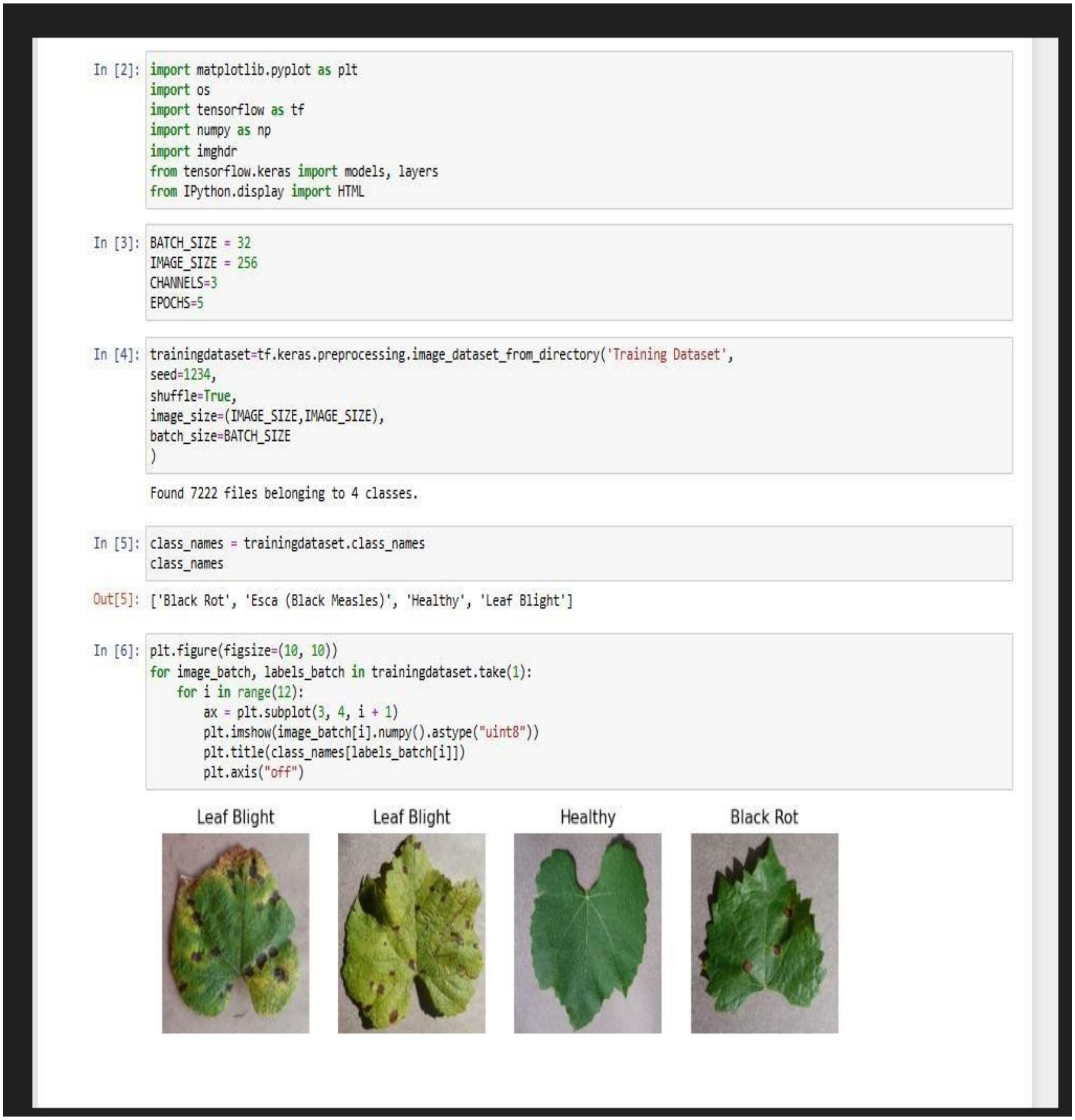
```
accuracy_score(actual , predicted)
0.8157020216006646

precision_score(actual , predicted)
0.8976098689282961

f1_score(actual , predicted)
0.8974022970785478
```

Accurately defining the disease spot:

```python
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
```

```
first image to predict
ctual label: Pepper_bell_Bacterial_spot
1/1 [==============================] - 0s 200ms/step
redicted label: Pepper_bell_Bacterial_spot
```





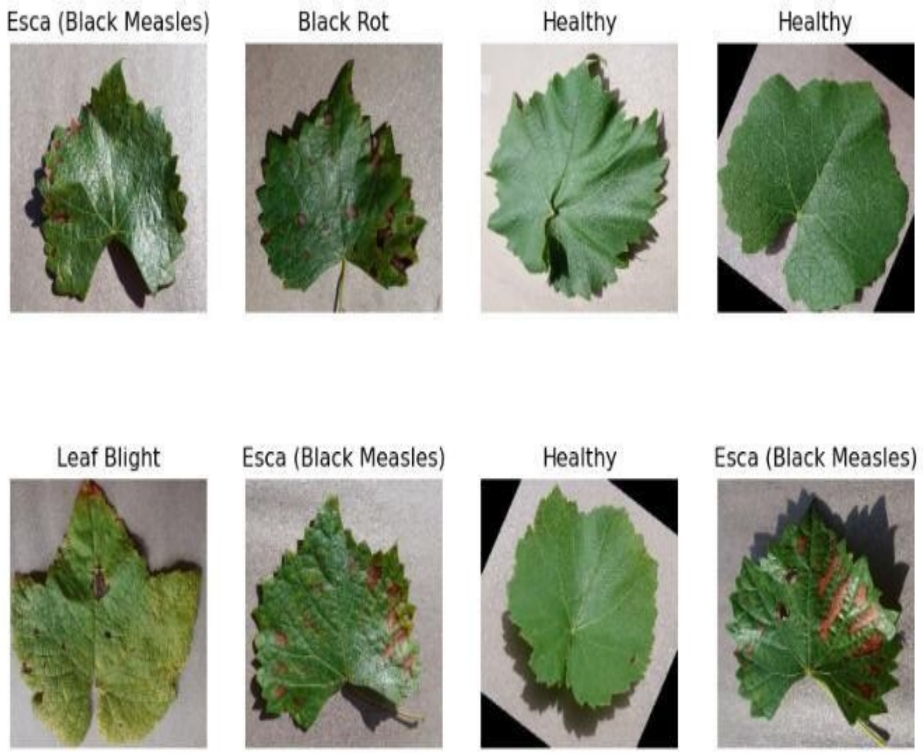
```
In [2]: import matplotlib.pyplot as plt
        import os
        import tensorflow as tf
        import numpy as np
        import imghdr
        from tensorflow.keras import models, layers
        from IPython.display import HTML
```

```
In [3]: BATCH_SIZE = 32
        IMAGE_SIZE = 256
        CHANNELS=3
        EPOCHS=5
```

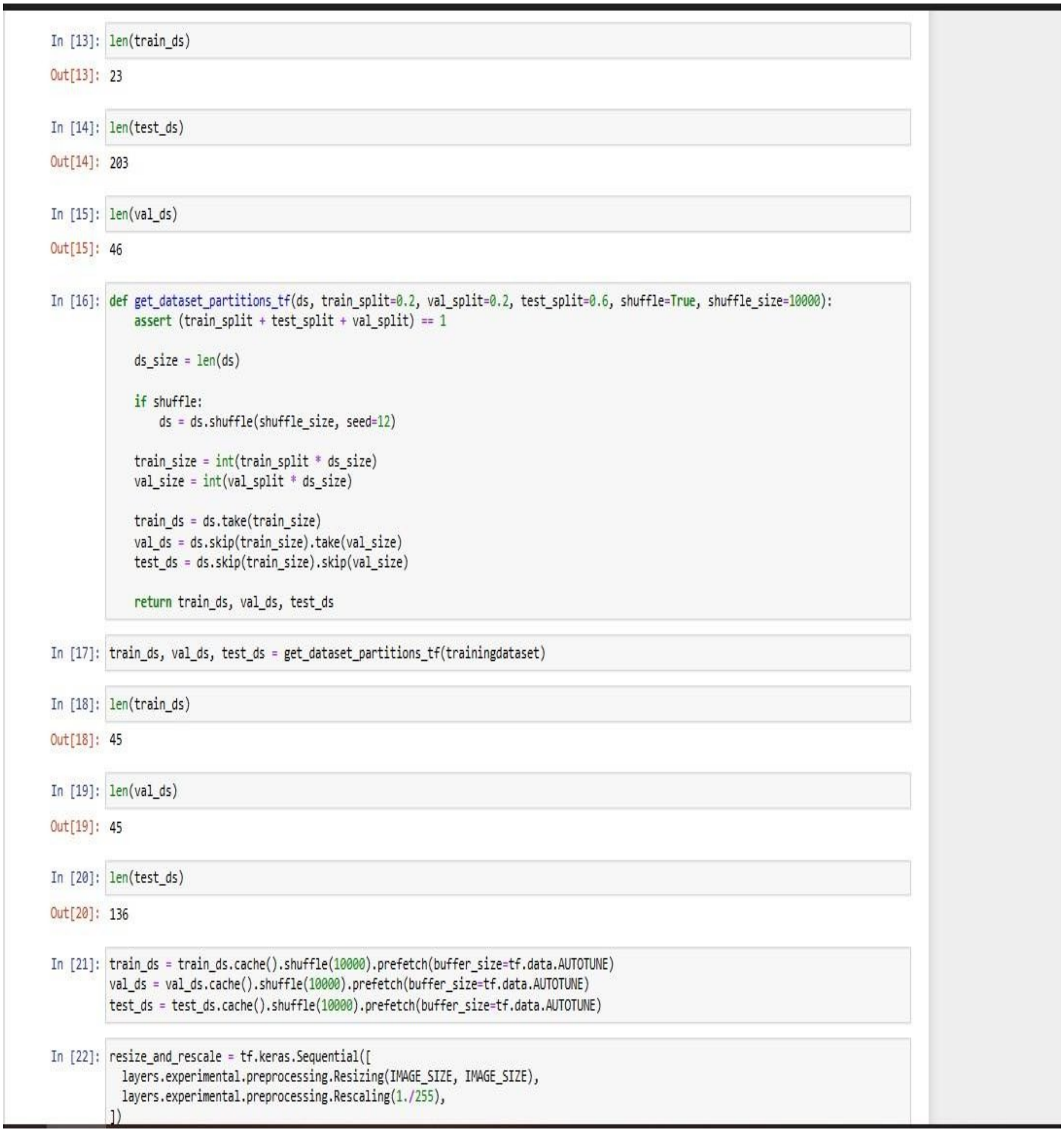
```
In [4]: trainingdataset=tf.keras.preprocessing.image_dataset_from_directory('Training Dataset',
        seed=1234,
        shuffle=True,
        image_size=(IMAGE_SIZE,IMAGE_SIZE),
        batch_size=BATCH_SIZE
        )

        Found 7222 files belonging to 4 classes.
```

```
In [5]: class_names = trainingdataset.class_names
        class_names

Out[5]: ['Black Rot', 'Esca (Black Measles)', 'Healthy', 'Leaf Blight']
```

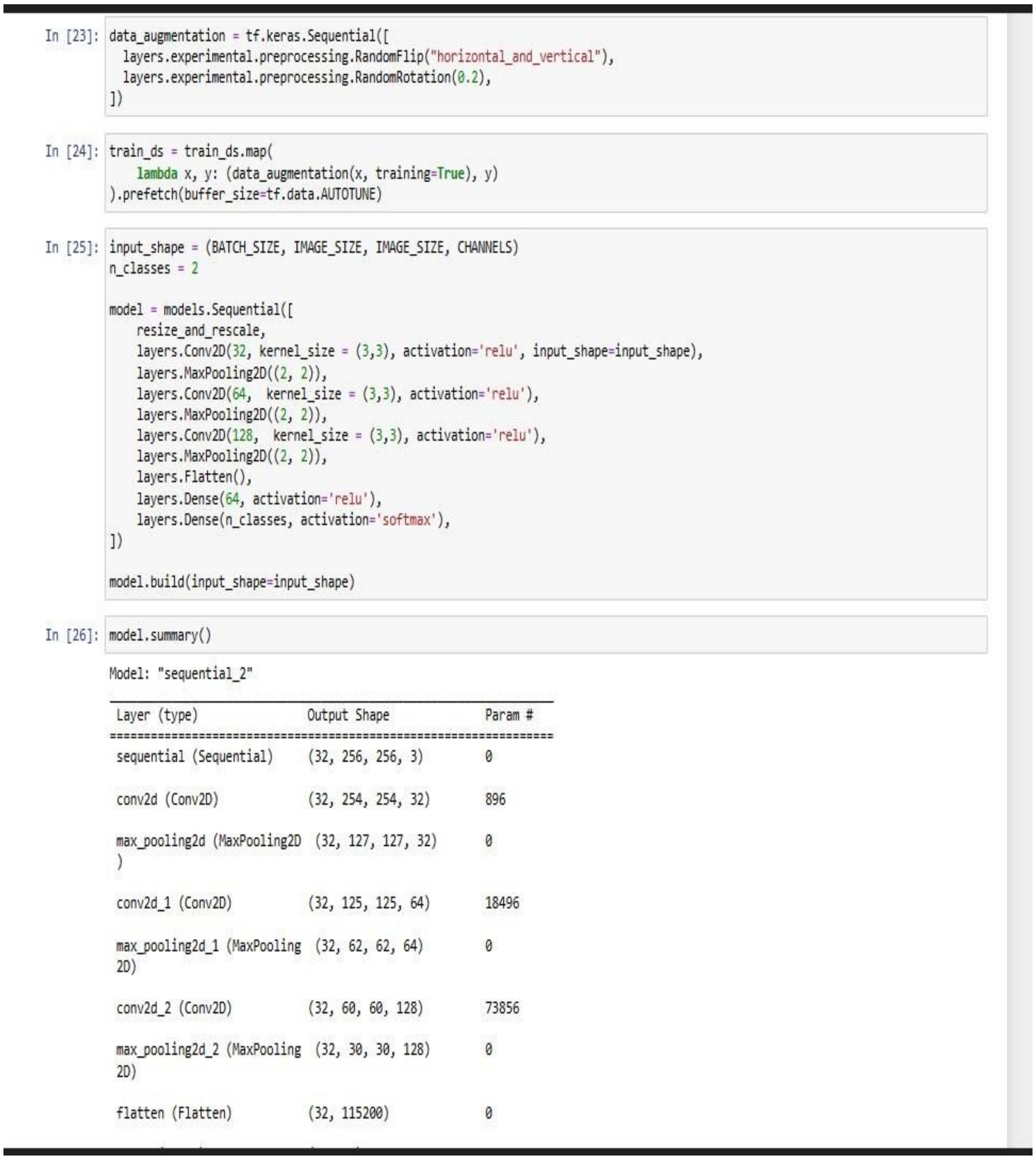
```
In [6]: plt.figure(figsize=(10, 10))
        for image_batch, labels_batch in trainingdataset.take(1):
            for i in range(12):
                ax = plt.subplot(3, 4, i + 1)
                plt.imshow(image_batch[i].numpy().astype("uint8"))
                plt.title(class_names[labels_batch[i]])
                plt.axis("off")
```



```
In [7]: len(trainingdataset)

Out[7]: 226

In [8]: train_size = 0.2
        len(trainingdataset)*train_size

Out[8]: 45.2

In [9]: train_ds = trainingdataset.take(23)
        len(train_ds)

Out[9]: 23

In [10]: test_ds = trainingdataset.skip(23)
         len(test_ds)

Out[10]: 203

In [11]: val_size=0.2
         len(trainingdataset)*val_size

Out[11]: 45.2
```



```
In [13]: len(train_ds)

Out[13]: 23

In [14]: len(test_ds)

Out[14]: 203

In [15]: len(val_ds)

Out[15]: 46

In [16]: def get_dataset_partitions_tf(ds, train_split=0.2, val_split=0.2, test_split=0.6, shuffle=True, shuffle_size=10000):
             assert (train_split + test_split + val_split) == 1

             ds_size = len(ds)

             if shuffle:
                 ds = ds.shuffle(shuffle_size, seed=12)

             train_size = int(train_split * ds_size)
             val_size = int(val_split * ds_size)

             train_ds = ds.take(train_size)
             val_ds = ds.skip(train_size).take(val_size)
             test_ds = ds.skip(train_size).skip(val_size)

             return train_ds, val_ds, test_ds

In [17]: train_ds, val_ds, test_ds = get_dataset_partitions_tf(trainingdataset)

In [18]: len(train_ds)

Out[18]: 45

In [19]: len(val_ds)

Out[19]: 45

In [20]: len(test_ds)

Out[20]: 136

In [21]: train_ds = train_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         val_ds = val_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         test_ds = test_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)

In [22]: resize_and_rescale = tf.keras.Sequential([
           layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
           layers.experimental.preprocessing.Rescaling(1./255),
         ])
```

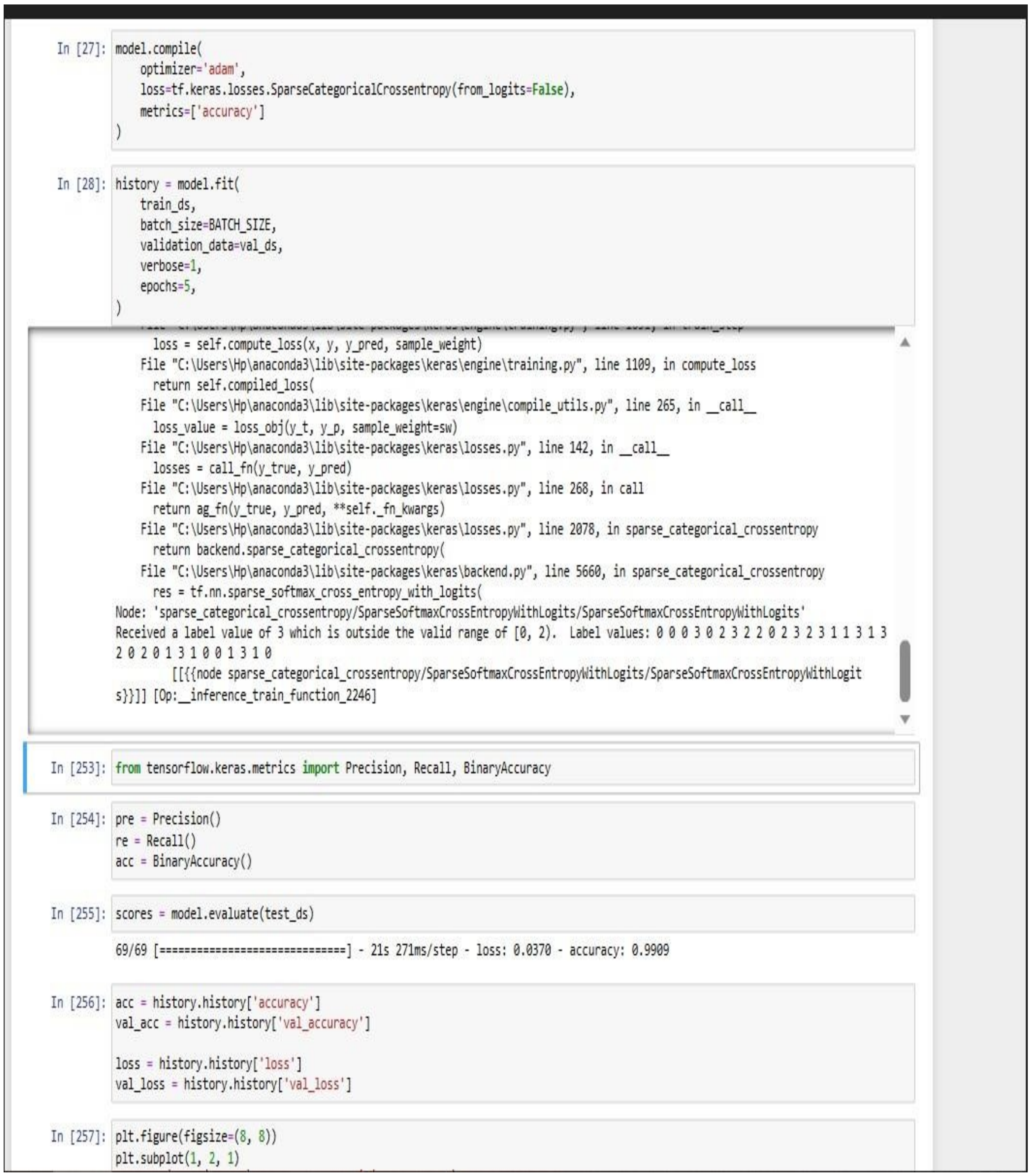


```
In [23]: data_augmentation = tf.keras.Sequential([
           layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
           layers.experimental.preprocessing.RandomRotation(0.2),
         ])
```

```
In [24]: train_ds = train_ds.map(
             lambda x, y: (data_augmentation(x, training=True), y)
         ).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
In [25]: input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
         n_classes = 2

         model = models.Sequential([
             resize_and_rescale,
             layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(128,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Flatten(),
             layers.Dense(64, activation='relu'),
             layers.Dense(n_classes, activation='softmax'),
         ])

         model.build(input_shape=input_shape)
```

```
In [26]: model.summary()

         Model: "sequential_2"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          sequential (Sequential)     (32, 256, 256, 3)         0

          conv2d (Conv2D)             (32, 254, 254, 32)        896

          max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
          )

          conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

          max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
          2D)

          conv2d_2 (Conv2D)           (32, 60, 60, 128)         73856

          max_pooling2d_2 (MaxPooling  (32, 30, 30, 128)        0
          2D)

          flatten (Flatten)           (32, 115200)              0
```



```
In [27]: model.compile(
             optimizer='adam',
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
             metrics=['accuracy']
         )
```

```
In [28]: history = model.fit(
             train_ds,
             batch_size=BATCH_SIZE,
             validation_data=val_ds,
             verbose=1,
             epochs=5,
         )
```

```
      loss = self.compute_loss(x, y, y_pred, sample_weight)
    File "C:\Users\Hp\anaconda3\lib\site-packages\keras\engine\training.py", line 1109, in compute_loss
      return self.compiled_loss(
    File "C:\Users\Hp\anaconda3\lib\site-packages\keras\engine\compile_utils.py", line 265, in __call__
      loss_value = loss_obj(y_t, y_p, sample_weight=sw)
    File "C:\Users\Hp\anaconda3\lib\site-packages\keras\losses.py", line 142, in __call__
      losses = call_fn(y_true, y_pred)
    File "C:\Users\Hp\anaconda3\lib\site-packages\keras\losses.py", line 268, in call
      return ag_fn(y_true, y_pred, **self._fn_kwargs)
    File "C:\Users\Hp\anaconda3\lib\site-packages\keras\losses.py", line 2078, in sparse_categorical_crossentropy
      return backend.sparse_categorical_crossentropy(
    File "C:\Users\Hp\anaconda3\lib\site-packages\keras\backend.py", line 5660, in sparse_categorical_crossentropy
      res = tf.nn.sparse_softmax_cross_entropy_with_logits(
Node: 'sparse_categorical_crossentropy/SparseSoftmaxCrossEntropyWithLogits/SparseSoftmaxCrossEntropyWithLogits'
Received a label value of 3 which is outside the valid range of [0, 2).  Label values: 0 0 0 3 0 2 3 2 2 0 2 3 2 3 1 1 3 1 3
2 0 2 0 1 3 1 0 0 1 3 1 0
         [[{{node sparse_categorical_crossentropy/SparseSoftmaxCrossEntropyWithLogits/SparseSoftmaxCrossEntropyWithLogit
s}}]] [Op:__inference_train_function_2246]
```

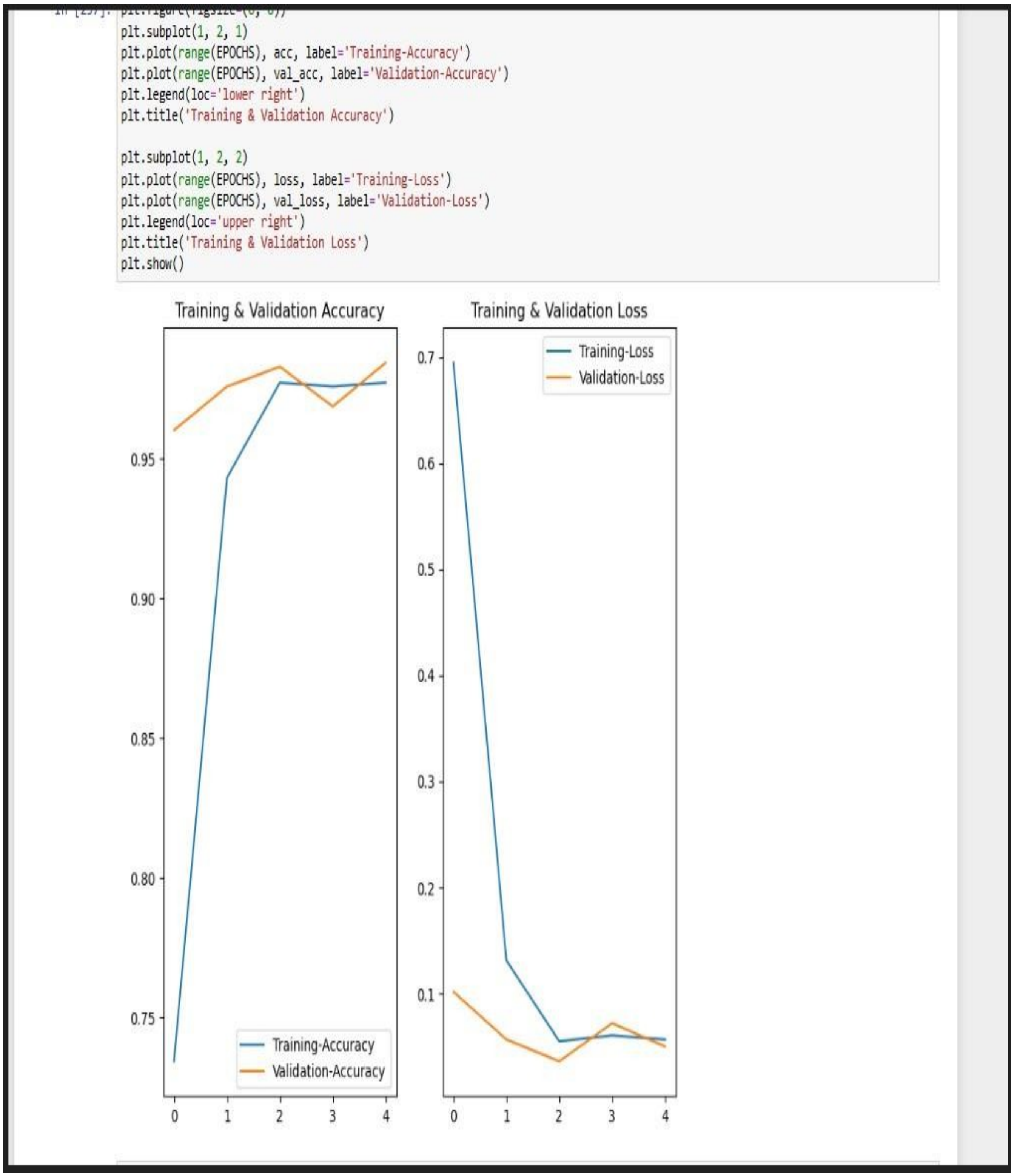
```
In [253]: from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

```
In [254]: pre = Precision()
          re = Recall()
          acc = BinaryAccuracy()
```

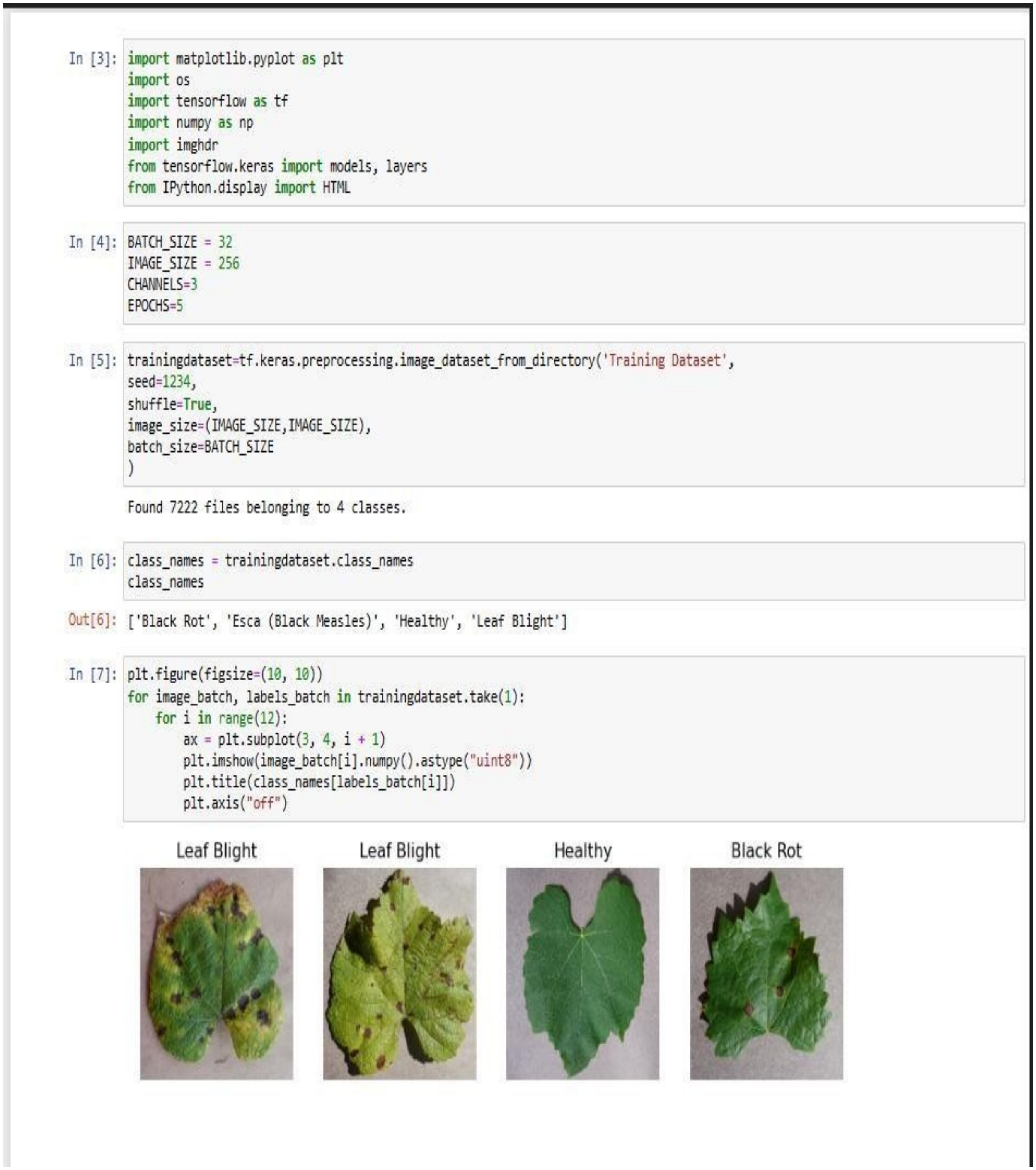
```
In [255]: scores = model.evaluate(test_ds)

          69/69 [==============================] - 21s 271ms/step - loss: 0.0370 - accuracy: 0.9909
```

```
In [256]: acc = history.history['accuracy']
          val_acc = history.history['val_accuracy']

          loss = history.history['loss']
          val_loss = history.history['val_loss']
```

```
In [257]: plt.figure(figsize=(8, 8))
          plt.subplot(1, 2, 1)
```



```
In [257]: plt.figure(figsize=(8, 8))
          plt.subplot(1, 2, 1)
          plt.plot(range(EPOCHS), acc, label='Training-Accuracy')
          plt.plot(range(EPOCHS), val_acc, label='Validation-Accuracy')
          plt.legend(loc='lower right')
          plt.title('Training & Validation Accuracy')

          plt.subplot(1, 2, 2)
          plt.plot(range(EPOCHS), loss, label='Training-Loss')
          plt.plot(range(EPOCHS), val_loss, label='Validation-Loss')
          plt.legend(loc='upper right')
          plt.title('Training & Validation Loss')
          plt.show()
```



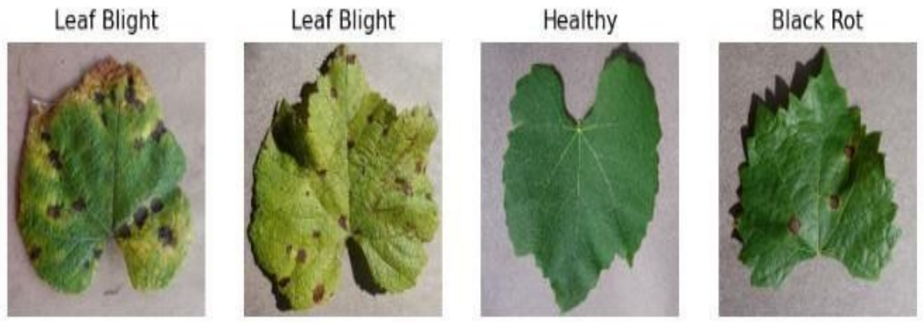
```
In [3]: import matplotlib.pyplot as plt
        import os
        import tensorflow as tf
        import numpy as np
        import imghdr
        from tensorflow.keras import models, layers
        from IPython.display import HTML
```

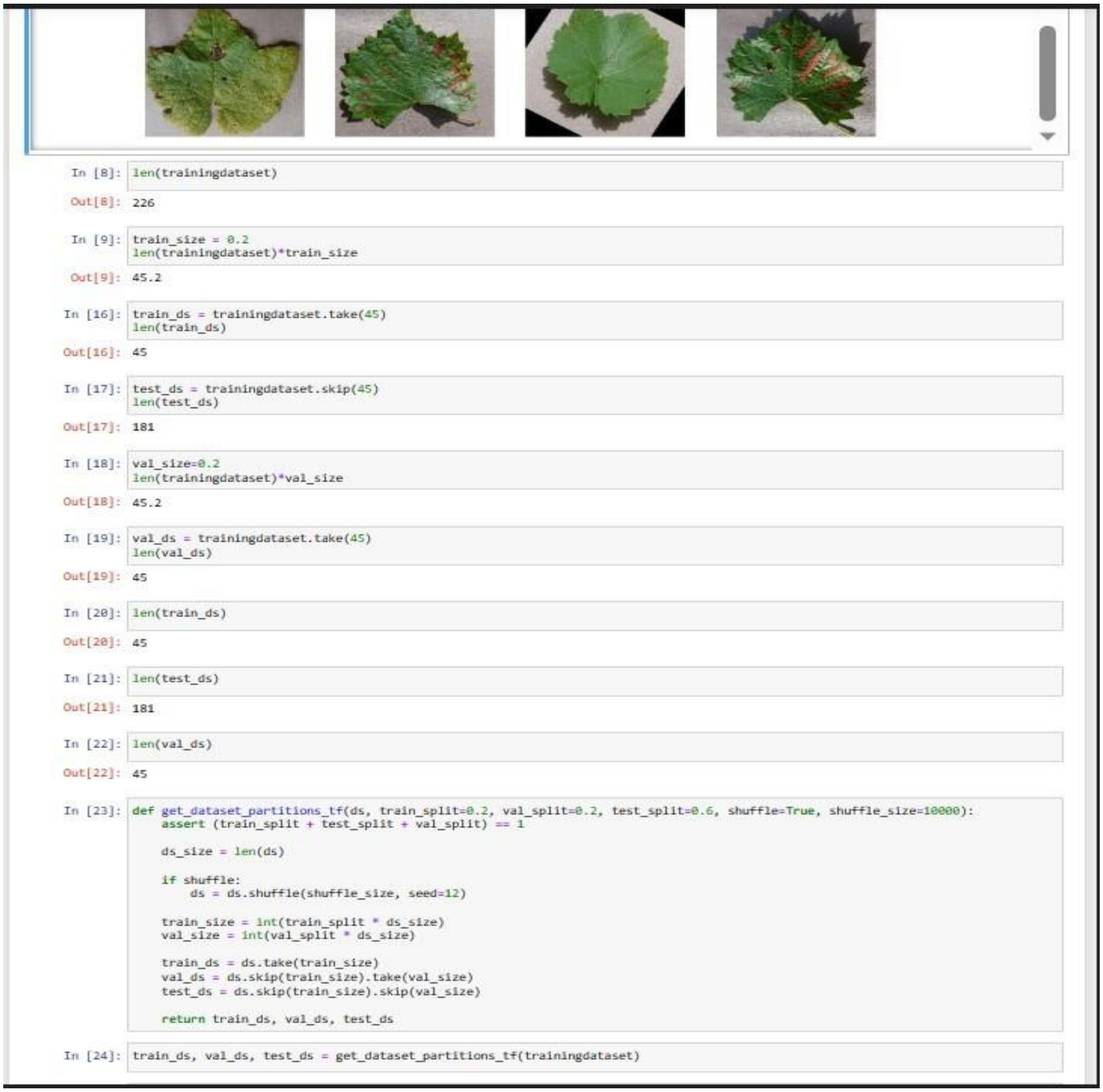
```
In [4]: BATCH_SIZE = 32
        IMAGE_SIZE = 256
        CHANNELS=3
        EPOCHS=5
```

```
In [5]: trainingdataset=tf.keras.preprocessing.image_dataset_from_directory('Training Dataset',
        seed=1234,
        shuffle=True,
        image_size=(IMAGE_SIZE,IMAGE_SIZE),
        batch_size=BATCH_SIZE
        )

        Found 7222 files belonging to 4 classes.
```

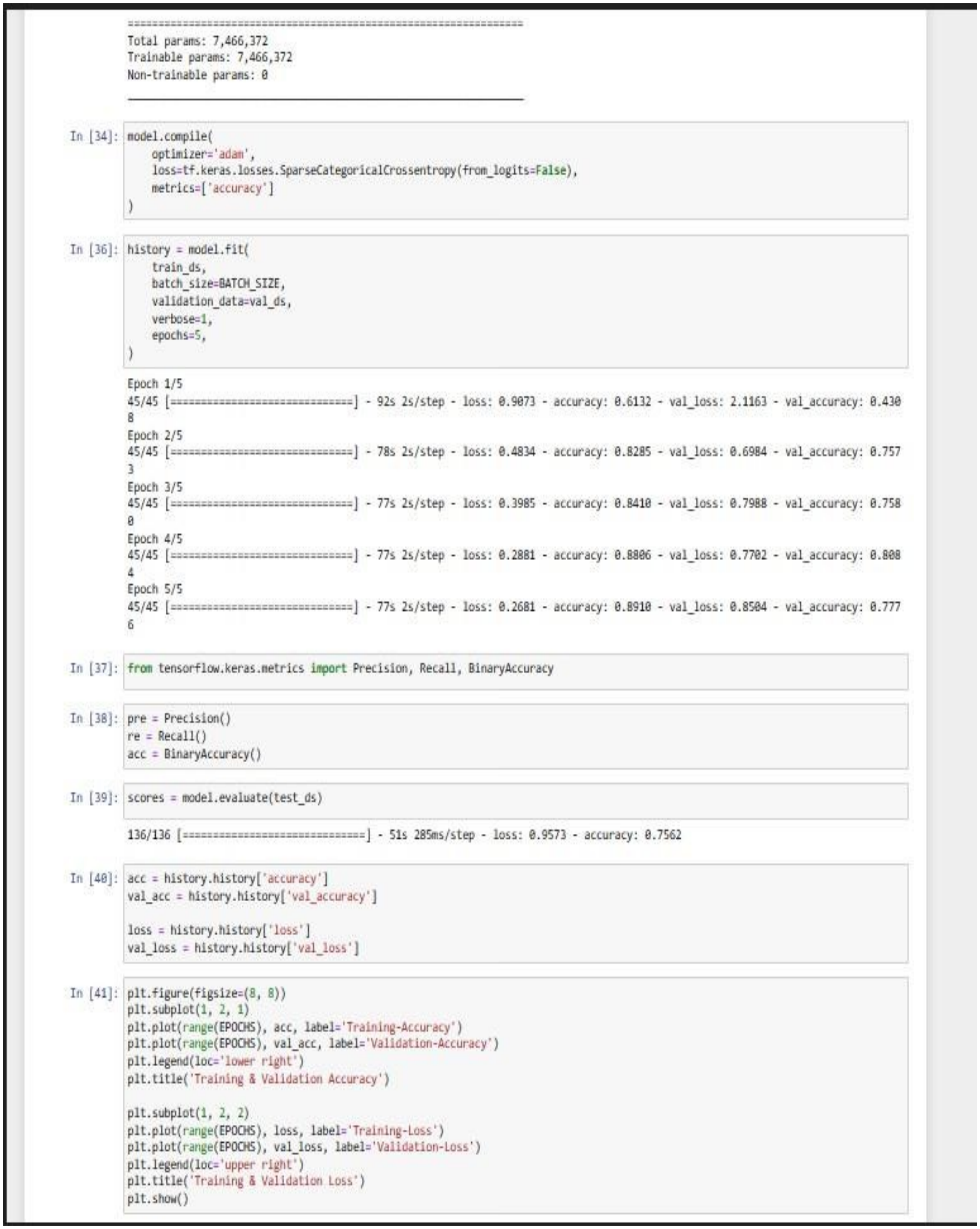
```
In [6]: class_names = trainingdataset.class_names
        class_names

Out[6]: ['Black Rot', 'Esca (Black Measles)', 'Healthy', 'Leaf Blight']
```

```
In [7]: plt.figure(figsize=(10, 10))
        for image_batch, labels_batch in trainingdataset.take(1):
            for i in range(12):
                ax = plt.subplot(3, 4, i + 1)
                plt.imshow(image_batch[i].numpy().astype("uint8"))
                plt.title(class_names[labels_batch[i]])
                plt.axis("off")
```





```
In [8]: len(trainingdataset)

Out[8]: 226

In [9]: train_size = 0.2
        len(trainingdataset)*train_size

Out[9]: 45.2

In [16]: train_ds = trainingdataset.take(45)
         len(train_ds)

Out[16]: 45

In [17]: test_ds = trainingdataset.skip(45)
         len(test_ds)

Out[17]: 181

In [18]: val_size=0.2
         len(trainingdataset)*val_size

Out[18]: 45.2

In [19]: val_ds = trainingdataset.take(45)
         len(val_ds)

Out[19]: 45

In [20]: len(train_ds)

Out[20]: 45

In [21]: len(test_ds)

Out[21]: 181

In [22]: len(val_ds)

Out[22]: 45

In [23]: def get_dataset_partitions_tf(ds, train_split=0.2, val_split=0.2, test_split=0.6, shuffle=True, shuffle_size=10000):
             assert (train_split + test_split + val_split) == 1

             ds_size = len(ds)

             if shuffle:
                 ds = ds.shuffle(shuffle_size, seed=12)

             train_size = int(train_split * ds_size)
             val_size = int(val_split * ds_size)

             train_ds = ds.take(train_size)
             val_ds = ds.skip(train_size).take(val_size)
             test_ds = ds.skip(train_size).skip(val_size)

             return train_ds, val_ds, test_ds

In [24]: train_ds, val_ds, test_ds = get_dataset_partitions_tf(trainingdataset)
In [25]: len(train_ds)

Out[25]: 45

In [26]: len(val_ds)

Out[26]: 45

In [27]: len(test_ds)

Out[27]: 136

In [28]: train_ds = train_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         val_ds = val_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         test_ds = test_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)

In [29]: resize_and_rescale = tf.keras.Sequential([
           layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
           layers.experimental.preprocessing.Rescaling(1./255),
         ])

In [30]: data_augmentation = tf.keras.Sequential([
           layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
           layers.experimental.preprocessing.RandomRotation(0.2),
         ])

In [31]: train_ds = train_ds.map(
             lambda x, y: (data_augmentation(x, training=True), y)
         ).prefetch(buffer_size=tf.data.AUTOTUNE)

In [32]: input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
         n_classes = 4

         model = models.Sequential([
             resize_and_rescale,
             layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(128,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Flatten(),
             layers.Dense(64, activation='relu'),
             layers.Dense(n_classes, activation='softmax'),
         ])

         model.build(input_shape=input_shape)

In [33]: model.summary()

         Model: "sequential_2"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          sequential (Sequential)     (32, 256, 256, 3)         0

          conv2d (Conv2D)             (32, 254, 254, 32)        896

          max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
          )

          conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

          max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
          2D)

          conv2d_2 (Conv2D)           (32, 60, 60, 128)         73856

          max_pooling2d_2 (MaxPooling  (32, 30, 30, 128)        0
          2D)
```



```
=================================================================
Total params: 7,466,372
Trainable params: 7,466,372
Non-trainable params: 0
_________________________________________________________________
```

In [34]:
```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

In [36]:
```
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=5,
)
```

```
Epoch 1/5
45/45 [==============================] - 92s 2s/step - loss: 0.9073 - accuracy: 0.6132 - val_loss: 2.1163 - val_accuracy: 0.430
8
Epoch 2/5
45/45 [==============================] - 78s 2s/step - loss: 0.4834 - accuracy: 0.8285 - val_loss: 0.6984 - val_accuracy: 0.757
3
Epoch 3/5
45/45 [==============================] - 77s 2s/step - loss: 0.3985 - accuracy: 0.8410 - val_loss: 0.7988 - val_accuracy: 0.758
0
Epoch 4/5
45/45 [==============================] - 77s 2s/step - loss: 0.2881 - accuracy: 0.8806 - val_loss: 0.7702 - val_accuracy: 0.808
4
Epoch 5/5
45/45 [==============================] - 77s 2s/step - loss: 0.2681 - accuracy: 0.8910 - val_loss: 0.8504 - val_accuracy: 0.777
6
```

In [37]:
```
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

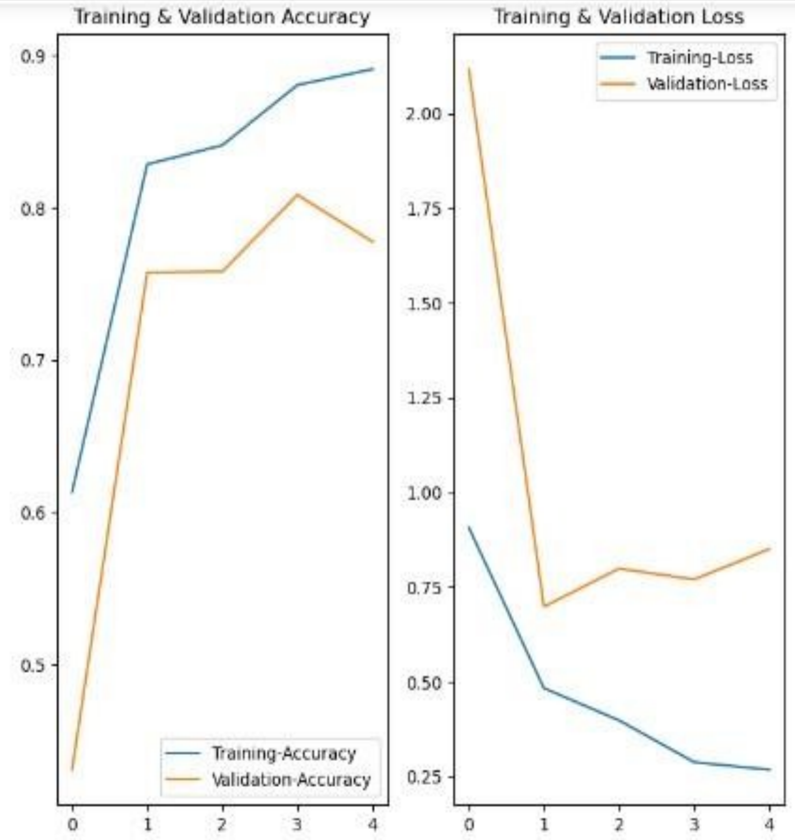
In [38]:
```
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
```

In [39]:
```
scores = model.evaluate(test_ds)
```

```
136/136 [==============================] - 51s 285ms/step - loss: 0.9573 - accuracy: 0.7562
```

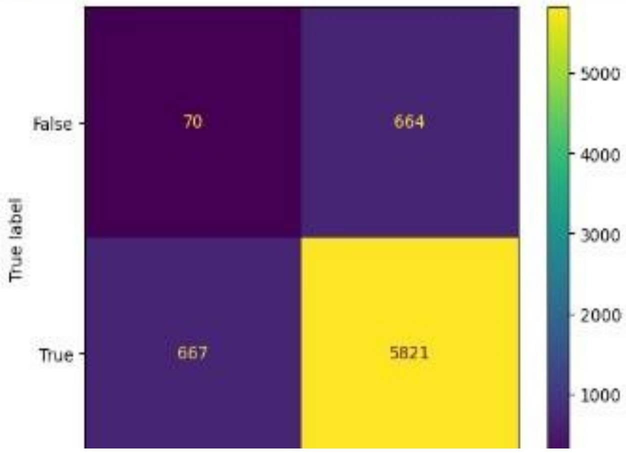
In [40]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [41]:
```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label='Training-Accuracy')
plt.plot(range(EPOCHS), val_acc, label='Validation-Accuracy')
plt.legend(loc='lower right')
plt.title('Training & Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(EPOCHS), loss, label='Training-Loss')
plt.plot(range(EPOCHS), val_loss, label='Validation-Loss')
plt.legend(loc='upper right')
plt.title('Training & Validation Loss')
plt.show()
```



```
In [42]: from sklearn import metrics
         import numpy
         import matplotlib.pyplot as plt
         from sklearn.metrics import accuracy_score , precision_score , f1_score
```

```
In [47]: actual = numpy.random.binomial(1,.9,size = 7222)
         predicted = numpy.random.binomial(1,.9,size = 7222)
         confusion_matrix = metrics.confusion_matrix(actual, predicted)

         cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
         Accuracy1 = metrics.accuracy_score(actual, predicted)

         cm_display.plot()
         plt.show()
```



In [1]:
```
import matplotlib.pyplot as plt
import os
import tensorflow as tf
import numpy as np
import imghdr
from tensorflow.keras import models, layers
from IPython.display import HTML
```

In [2]:
```
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=5
```

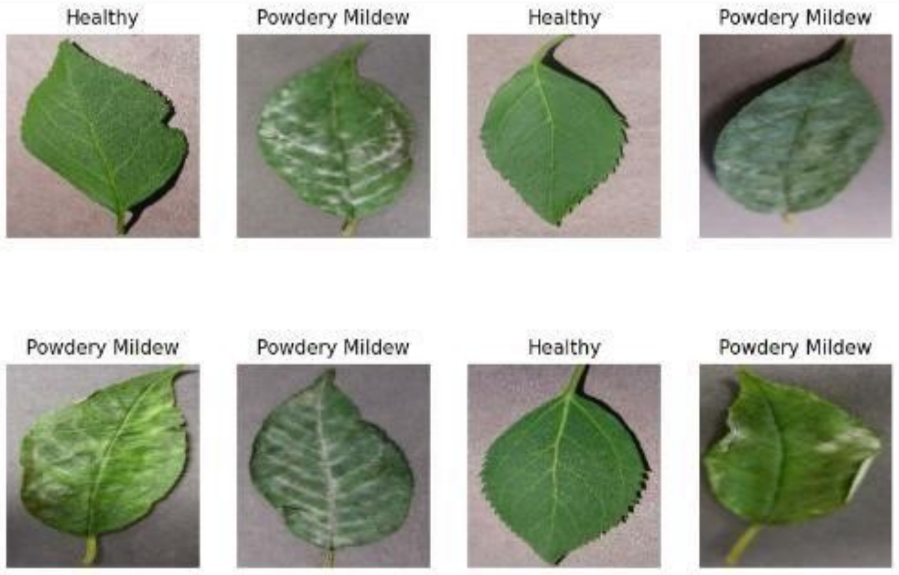
In [4]:
```
trainingdataset=tf.keras.preprocessing.image_dataset_from_directory('Training Dataset',
seed=1234,
shuffle=True,
image_size=(IMAGE_SIZE,IMAGE_SIZE),
batch_size=BATCH_SIZE
)
```

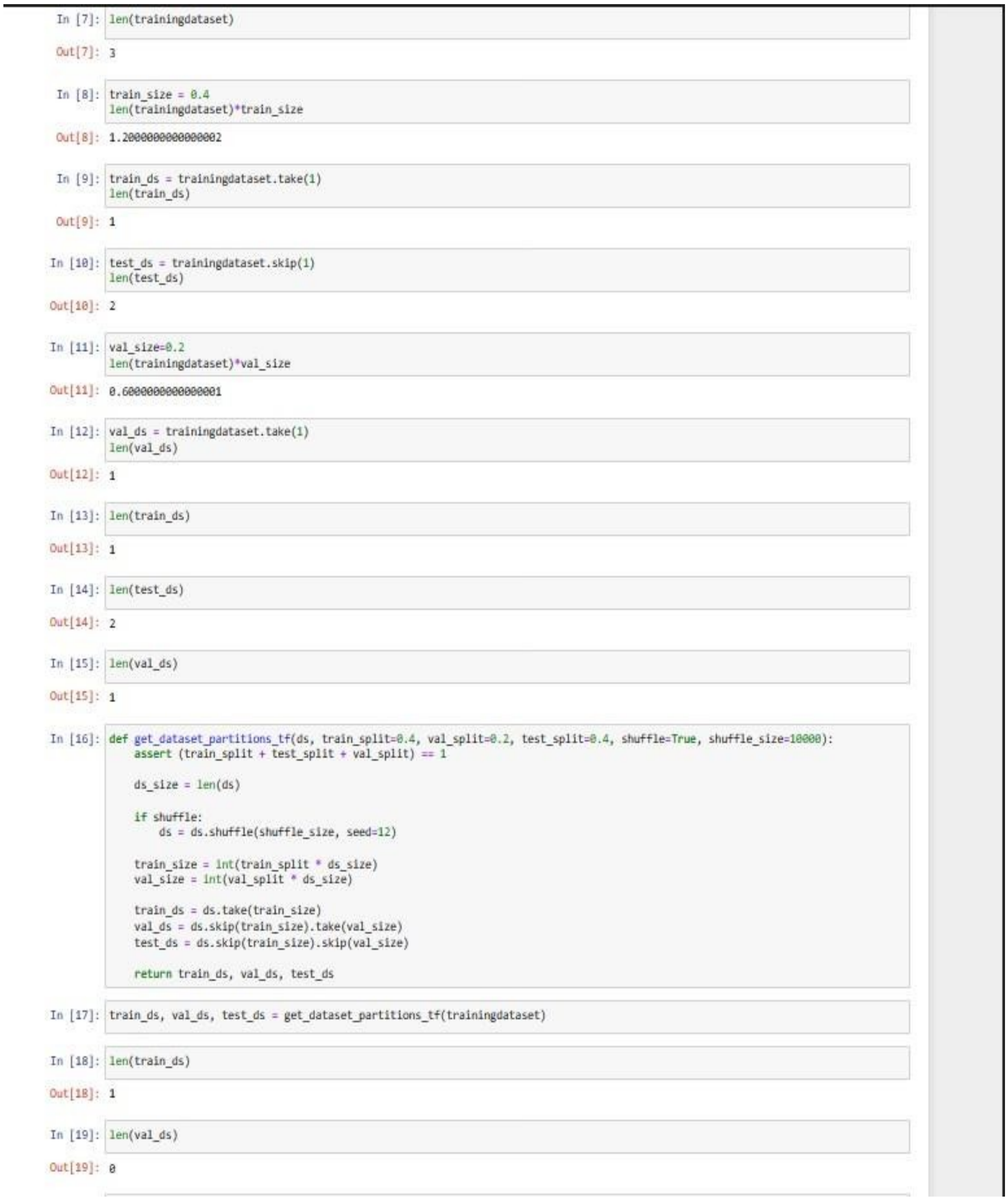
```
Found 89 files belonging to 2 classes.
```

In [5]:
```
class_names = trainingdataset.class_names
class_names
```

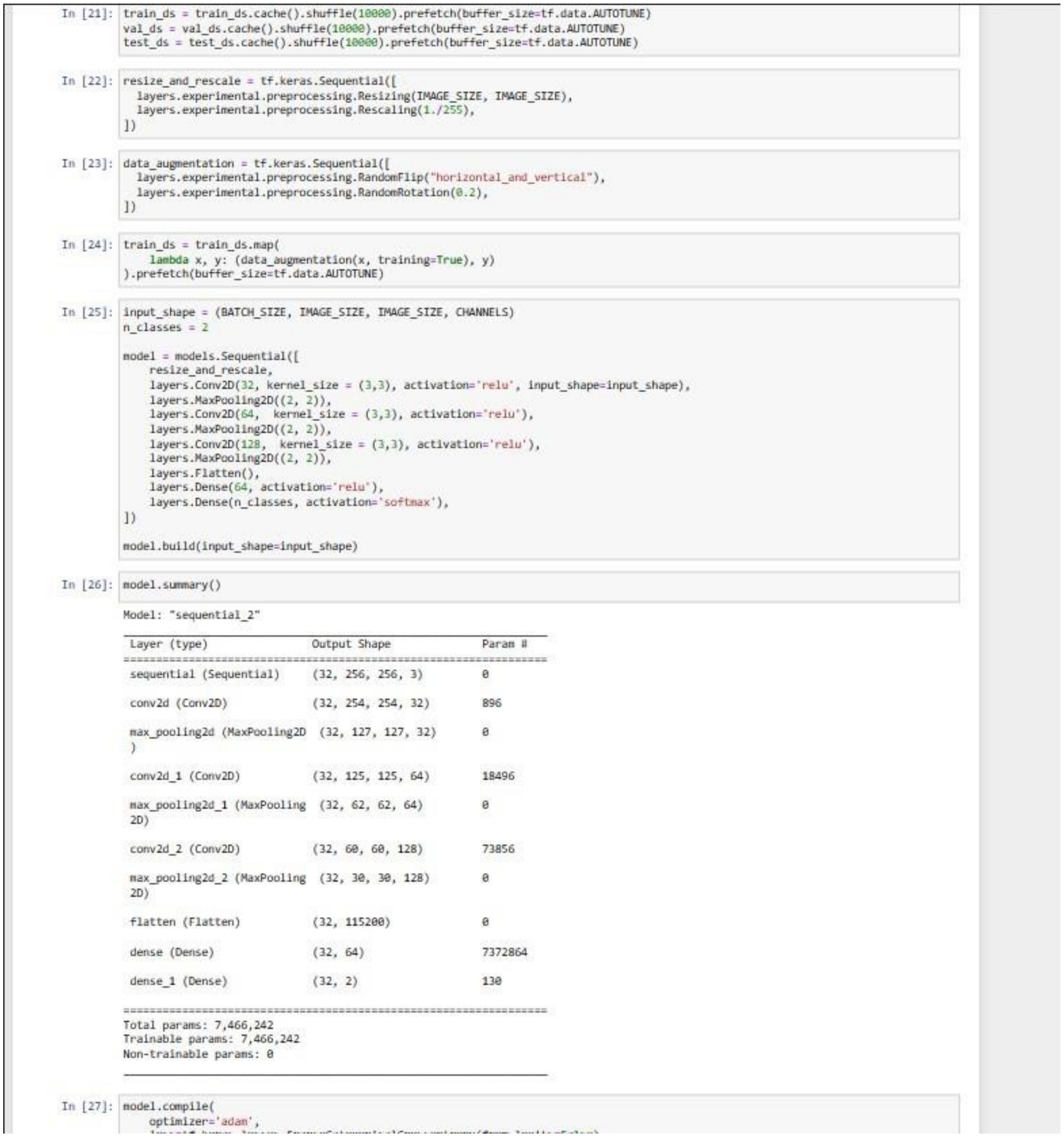
Out[5]:
```
['Healthy', 'Powdery Mildew']
```

In [6]:
```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in trainingdataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```





```
In [7]:  len(trainingdataset)

Out[7]:  3

In [8]:  train_size = 0.4
         len(trainingdataset)*train_size

Out[8]:  1.2000000000000002

In [9]:  train_ds = trainingdataset.take(1)
         len(train_ds)

Out[9]:  1

In [10]: test_ds = trainingdataset.skip(1)
         len(test_ds)

Out[10]: 2

In [11]: val_size=0.2
         len(trainingdataset)*val_size

Out[11]: 0.6000000000000001

In [12]: val_ds = trainingdataset.take(1)
         len(val_ds)

Out[12]: 1

In [13]: len(train_ds)

Out[13]: 1

In [14]: len(test_ds)

Out[14]: 2

In [15]: len(val_ds)

Out[15]: 1

In [16]: def get_dataset_partitions_tf(ds, train_split=0.4, val_split=0.2, test_split=0.4, shuffle=True, shuffle_size=10000):
             assert (train_split + test_split + val_split) == 1

             ds_size = len(ds)

             if shuffle:
                 ds = ds.shuffle(shuffle_size, seed=12)

             train_size = int(train_split * ds_size)
             val_size = int(val_split * ds_size)

             train_ds = ds.take(train_size)
             val_ds = ds.skip(train_size).take(val_size)
             test_ds = ds.skip(train_size).skip(val_size)

             return train_ds, val_ds, test_ds

In [17]: train_ds, val_ds, test_ds = get_dataset_partitions_tf(trainingdataset)

In [18]: len(train_ds)

Out[18]: 1

In [19]: len(val_ds)

Out[19]: 0
```



```
In [21]: train_ds = train_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         val_ds = val_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         test_ds = test_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
In [22]: resize_and_rescale = tf.keras.Sequential([
           layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
           layers.experimental.preprocessing.Rescaling(1./255),
         ])
```

```
In [23]: data_augmentation = tf.keras.Sequential([
           layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
           layers.experimental.preprocessing.RandomRotation(0.2),
         ])
```

```
In [24]: train_ds = train_ds.map(
             lambda x, y: (data_augmentation(x, training=True), y)
         ).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
In [25]: input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
         n_classes = 2

         model = models.Sequential([
             resize_and_rescale,
             layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(128,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Flatten(),
             layers.Dense(64, activation='relu'),
             layers.Dense(n_classes, activation='softmax'),
         ])

         model.build(input_shape=input_shape)
```

```
In [26]: model.summary()

         Model: "sequential_2"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          sequential (Sequential)     (32, 256, 256, 3)         0

          conv2d (Conv2D)             (32, 254, 254, 32)        896

          max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
          )

          conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

          max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
          2D)

          conv2d_2 (Conv2D)           (32, 60, 60, 128)         73856

          max_pooling2d_2 (MaxPooling  (32, 30, 30, 128)        0
          2D)

          flatten (Flatten)           (32, 115200)              0

          dense (Dense)               (32, 64)                  7372864

          dense_1 (Dense)             (32, 2)                   130

         =================================================================
         Total params: 7,466,242
         Trainable params: 7,466,242
         Non-trainable params: 0
         _________________________________________________________________
```

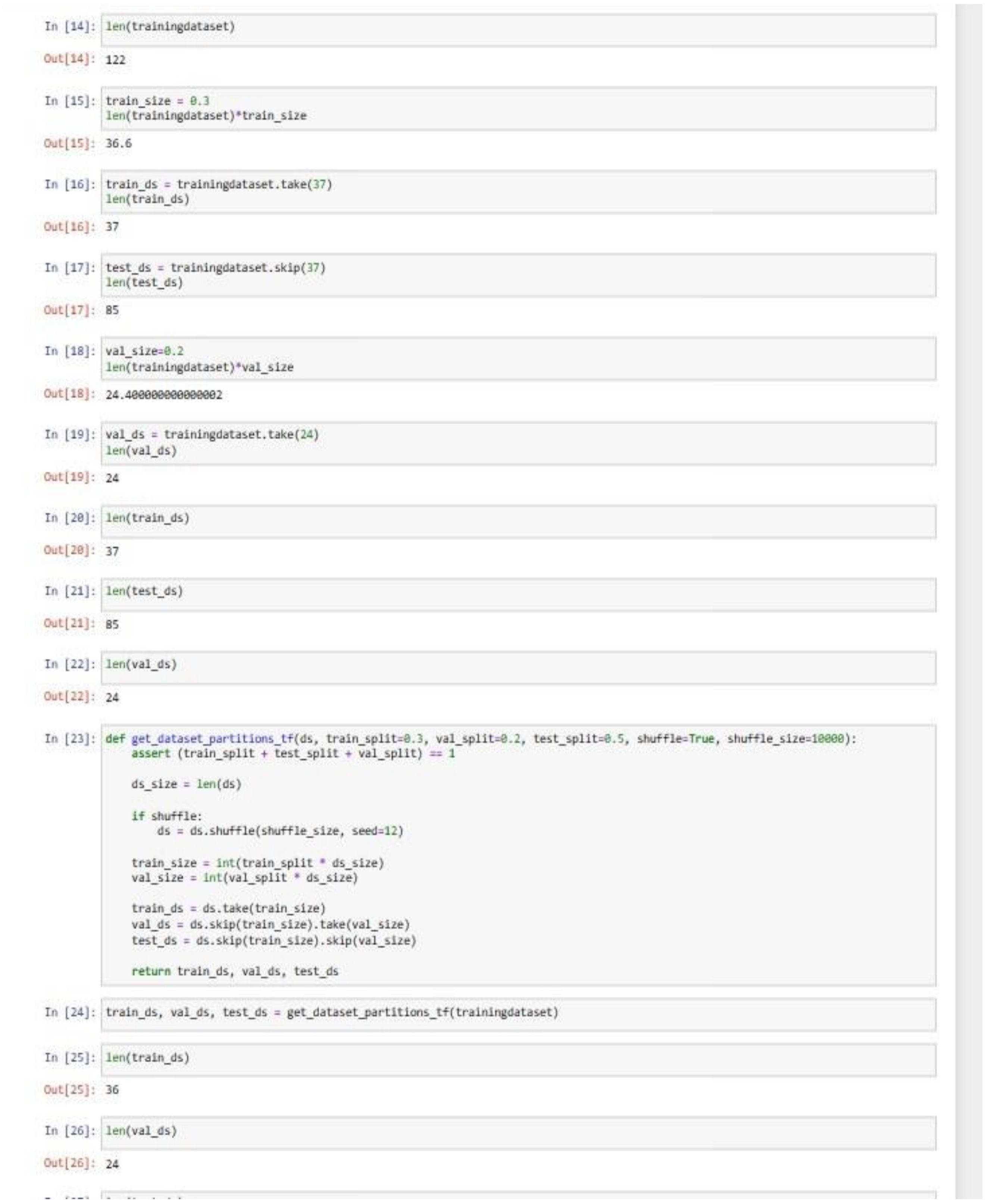
```
In [27]: model.compile(
             optimizer='adam',
In [5]: import matplotlib.pyplot as plt
        import os
        import tensorflow as tf
        import numpy as np
        import imghdr
        from tensorflow.keras import models, layers
        from IPython.display import HTML
```

```
In [6]: BATCH_SIZE = 32
        IMAGE_SIZE = 256
        CHANNELS=3
        EPOCHS=5
```

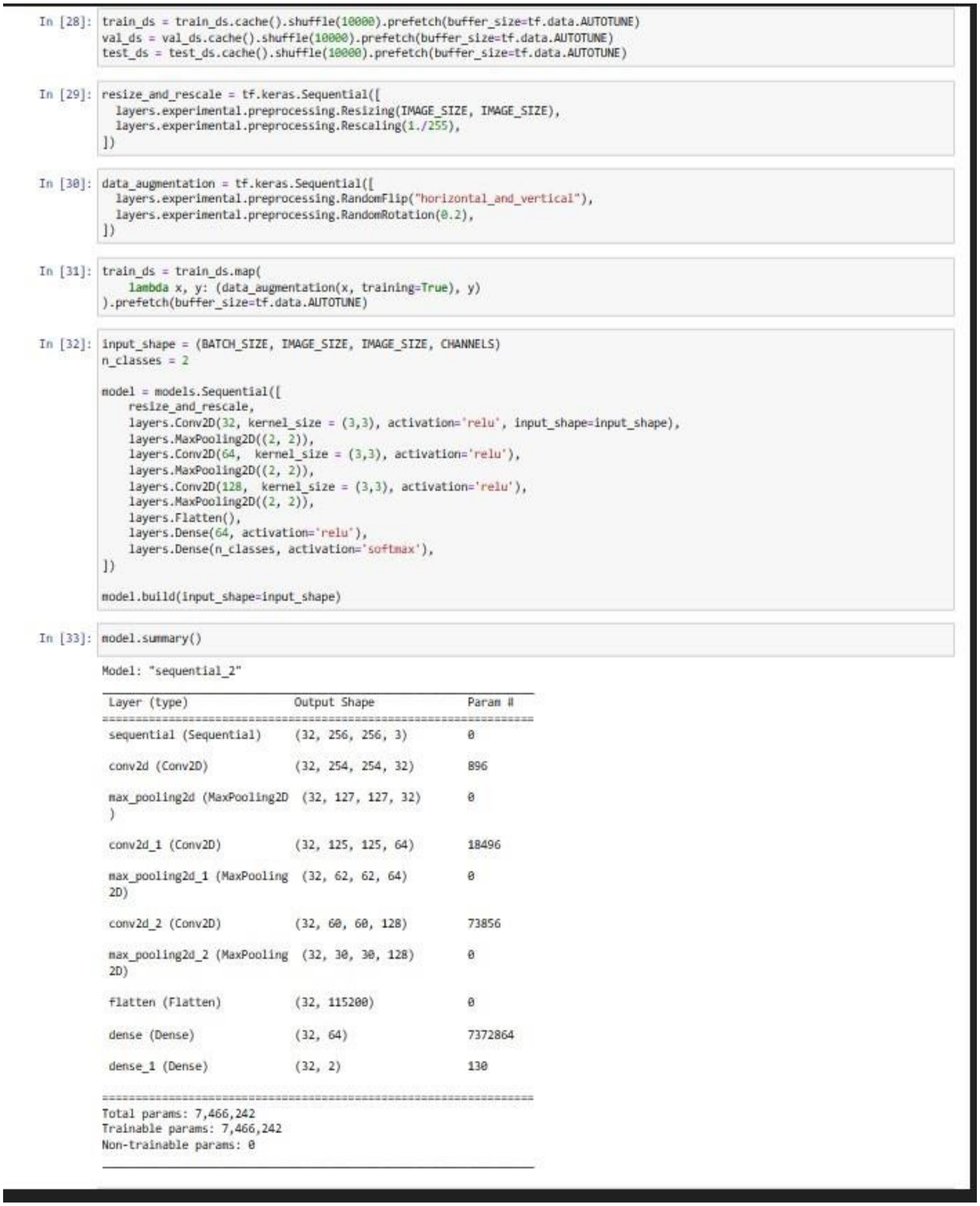
```
In [11]: trainingdataset=tf.keras.preprocessing.image_dataset_from_directory('Training Dataset',
         seed=1234,
         shuffle=True,
         image_size=(IMAGE_SIZE,IMAGE_SIZE),
         batch_size=BATCH_SIZE
         )

         Found 3901 files belonging to 2 classes.
```

```
In [12]: class_names = trainingdataset.class_names
         class_names

Out[12]: ['Bacterial Spot', 'Healthy']
```

```
In [13]: plt.figure(figsize=(10, 10))
         for image_batch, labels_batch in trainingdataset.take(1):
             for i in range(12):
                 ax = plt.subplot(3, 4, i + 1)
                 plt.imshow(image_batch[i].numpy().astype("uint8"))
                 plt.title(class_names[labels_batch[i]])
                 plt.axis("off")
```

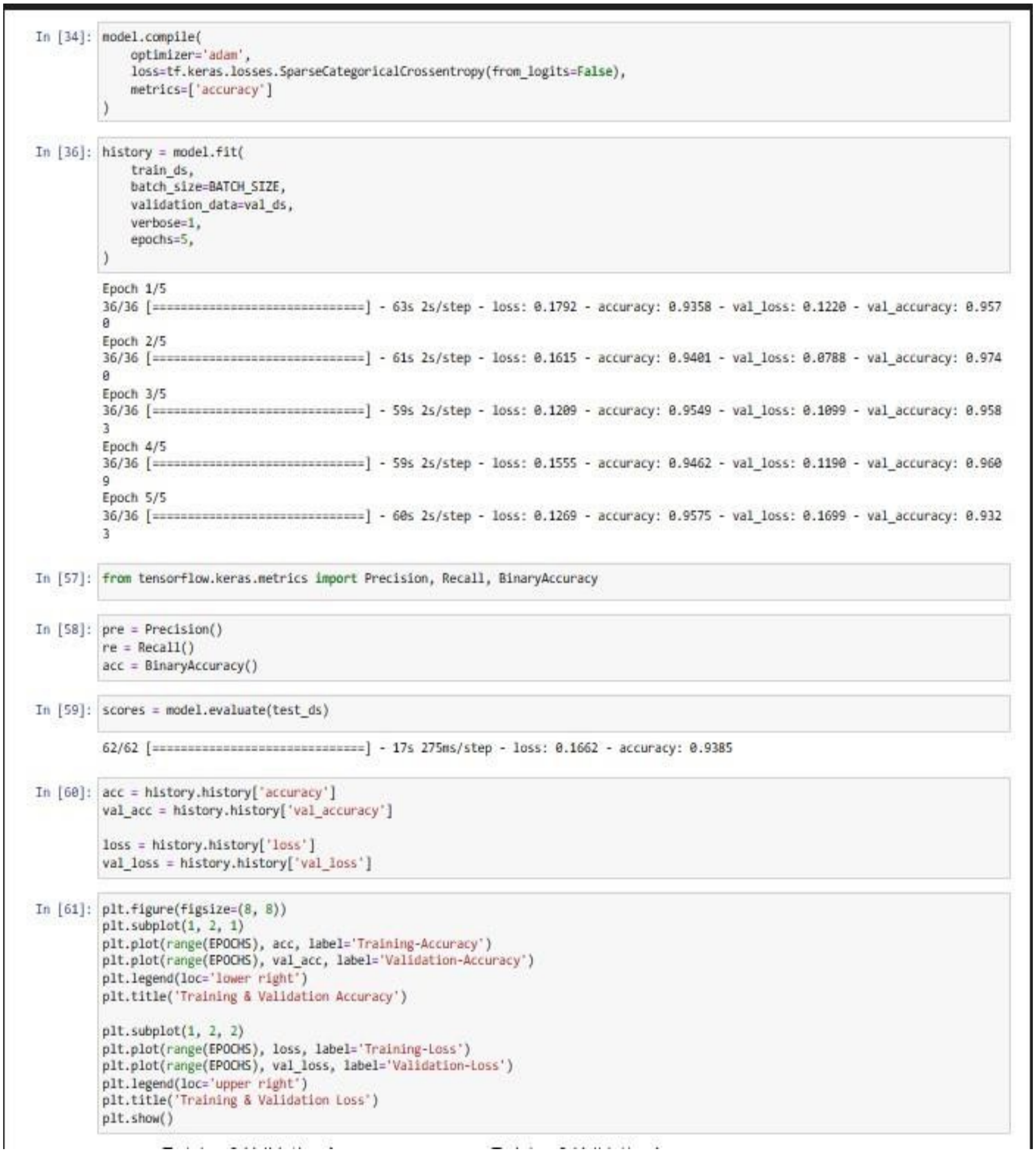


```
In [14]: len(trainingdataset)

Out[14]: 122

In [15]: train_size = 0.3
         len(trainingdataset)*train_size

Out[15]: 36.6

In [16]: train_ds = trainingdataset.take(37)
         len(train_ds)

Out[16]: 37

In [17]: test_ds = trainingdataset.skip(37)
         len(test_ds)

Out[17]: 85

In [18]: val_size=0.2
         len(trainingdataset)*val_size

Out[18]: 24.400000000000002

In [19]: val_ds = trainingdataset.take(24)
         len(val_ds)

Out[19]: 24

In [20]: len(train_ds)

Out[20]: 37

In [21]: len(test_ds)

Out[21]: 85

In [22]: len(val_ds)

Out[22]: 24

In [23]: def get_dataset_partitions_tf(ds, train_split=0.3, val_split=0.2, test_split=0.5, shuffle=True, shuffle_size=10000):
             assert (train_split + test_split + val_split) == 1

             ds_size = len(ds)

             if shuffle:
                 ds = ds.shuffle(shuffle_size, seed=12)

             train_size = int(train_split * ds_size)
             val_size = int(val_split * ds_size)

             train_ds = ds.take(train_size)
             val_ds = ds.skip(train_size).take(val_size)
             test_ds = ds.skip(train_size).skip(val_size)

             return train_ds, val_ds, test_ds

In [24]: train_ds, val_ds, test_ds = get_dataset_partitions_tf(trainingdataset)

In [25]: len(train_ds)

Out[25]: 36

In [26]: len(val_ds)

Out[26]: 24
```



```
In [28]: train_ds = train_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         val_ds = val_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)
         test_ds = test_ds.cache().shuffle(10000).prefetch(buffer_size=tf.data.AUTOTUNE)

In [29]: resize_and_rescale = tf.keras.Sequential([
           layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
           layers.experimental.preprocessing.Rescaling(1./255),
         ])

In [30]: data_augmentation = tf.keras.Sequential([
           layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
           layers.experimental.preprocessing.RandomRotation(0.2),
         ])

In [31]: train_ds = train_ds.map(
             lambda x, y: (data_augmentation(x, training=True), y)
         ).prefetch(buffer_size=tf.data.AUTOTUNE)

In [32]: input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
         n_classes = 2

         model = models.Sequential([
             resize_and_rescale,
             layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Conv2D(128,  kernel_size = (3,3), activation='relu'),
             layers.MaxPooling2D((2, 2)),
             layers.Flatten(),
             layers.Dense(64, activation='relu'),
             layers.Dense(n_classes, activation='softmax'),
         ])

         model.build(input_shape=input_shape)

In [33]: model.summary()

         Model: "sequential_2"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          sequential (Sequential)     (32, 256, 256, 3)         0

          conv2d (Conv2D)             (32, 254, 254, 32)        896

          max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
          )

          conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

          max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
          2D)

          conv2d_2 (Conv2D)           (32, 60, 60, 128)         73856

          max_pooling2d_2 (MaxPooling  (32, 30, 30, 128)        0
          2D)

          flatten (Flatten)           (32, 115200)              0

          dense (Dense)               (32, 64)                  7372864

          dense_1 (Dense)             (32, 2)                   130

         =================================================================
         Total params: 7,466,242
         Trainable params: 7,466,242
         Non-trainable params: 0
         _________________________________________________________________
```

\



```
In [34]: model.compile(
             optimizer='adam',
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
             metrics=['accuracy']
         )
```

```
In [36]: history = model.fit(
             train_ds,
             batch_size=BATCH_SIZE,
             validation_data=val_ds,
             verbose=1,
             epochs=5,
         )
```

```
Epoch 1/5
36/36 [==============================] - 63s 2s/step - loss: 0.1792 - accuracy: 0.9358 - val_loss: 0.1220 - val_accuracy: 0.957
0
Epoch 2/5
36/36 [==============================] - 61s 2s/step - loss: 0.1615 - accuracy: 0.9401 - val_loss: 0.0788 - val_accuracy: 0.974
0
Epoch 3/5
36/36 [==============================] - 59s 2s/step - loss: 0.1209 - accuracy: 0.9549 - val_loss: 0.1099 - val_accuracy: 0.958
3
Epoch 4/5
36/36 [==============================] - 59s 2s/step - loss: 0.1555 - accuracy: 0.9462 - val_loss: 0.1190 - val_accuracy: 0.960
9
Epoch 5/5
36/36 [==============================] - 60s 2s/step - loss: 0.1269 - accuracy: 0.9575 - val_loss: 0.1699 - val_accuracy: 0.932
3
```

```
In [57]: from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

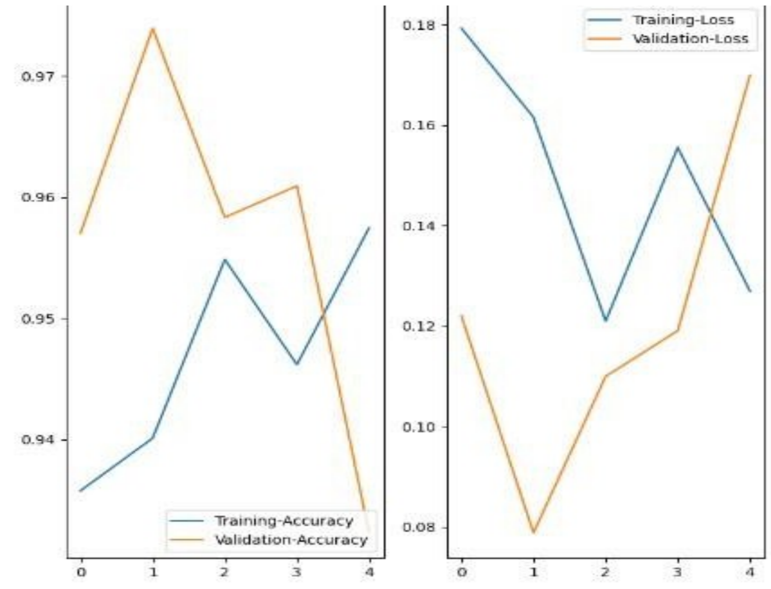
```
In [58]: pre = Precision()
         re = Recall()
         acc = BinaryAccuracy()
```

```
In [59]: scores = model.evaluate(test_ds)
```

```
62/62 [==============================] - 17s 275ms/step - loss: 0.1662 - accuracy: 0.9385
```

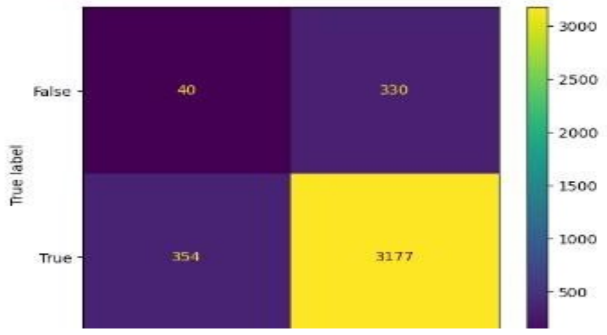
```
In [60]: acc = history.history['accuracy']
         val_acc = history.history['val_accuracy']

         loss = history.history['loss']
         val_loss = history.history['val_loss']
```

```
In [61]: plt.figure(figsize=(8, 8))
         plt.subplot(1, 2, 1)
         plt.plot(range(EPOCHS), acc, label='Training-Accuracy')
         plt.plot(range(EPOCHS), val_acc, label='Validation-Accuracy')
         plt.legend(loc='lower right')
         plt.title('Training & Validation Accuracy')

         plt.subplot(1, 2, 2)
         plt.plot(range(EPOCHS), loss, label='Training-Loss')
         plt.plot(range(EPOCHS), val_loss, label='Validation-Loss')
         plt.legend(loc='upper right')
         plt.title('Training & Validation Loss')
         plt.show()
```



```
In [62]: from sklearn import metrics
         import numpy
         import matplotlib.pyplot as plt
         from sklearn.metrics import accuracy_score , precision_score , f1_score
```

```
In [63]: actual = numpy.random.binomial(1,.9,size = 3901)
         predicted = numpy.random.binomial(1,.9,size = 3901)
         confusion_matrix = metrics.confusion_matrix(actual, predicted)

         cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
         Accuracy1 = metrics.accuracy_score(actual, predicted)

         cm_display.plot()
         plt.show()
```



```
In [64]: accuracy_score(actual , predicted)

Out[64]: 0.8246603435016663

In [65]: precision_score(actual , predicted)

Out[65]: 0.9059024807527801

In [46]: f1_score(actual , predicted)

Out[46]: 0.9015205343186017

In [ ]:
```

CHAPTER 5

CONCLUSION AND FUTURE WORK

1. Conclusion

We have taken a decent amount of high quality images(via datasets) of different crops tested the images(via open-cv) and had trained our model through several cases to deeply examine the leafs layer by layer(via CNN- convolutional neural network with other techniques) we got the technique to find the diseases and after finding the disease accurately (85-95)% , we use visualization technologies(matplotlib) so that the user can understand the project easily and show their interests in implementing. We also provide confusion matrix with accuracy precision and f -value factor to show the probability of success rate in the future.

After the implementation of our project the farmer can get following benefits-

1) **Early detection:** With the help of CNN, farmers can detect diseases in crops at an early stage. This can help in preventing the spread of diseases to other crops and also limit the damage caused to the affected crops.

2) **Improved crop yield:** Early detection of crop diseases can help farmers take timely and appropriate measures to treat the affected crops. This can result in a higher crop yield and better quality produce.

3) **Reduced cost:** Early detection and treatment of crop diseases can help reduce the cost of crop production. This is because farmers can avoid using excessive amounts of pesticides or other chemicals to treat crops, which can be expensive.

Taking everything above in account we can conclude that this paper gives an overall review for the technique and presents brief summary of different methods useful for early detection of plant diseases.Our model can provide great value to one's crop as we have tried to combine the mordern technology with traditional farming technique which come up with a great accuracy and efficiency that can directly increase the overall output from the land. We have tried to fulfill the need of efficient method within cost effective way, which gives the farmers a reliable solution for the betterment intheir agricultural feild. This project will help the farmers in tackling one of the main problem which can easily ruin their hard work and had come with a less complex technique with having a great efficiency(85-95% approx)that will create a great impact and led to the betterment of farmer as well as the world. In future, we aim to work down to develop a more efficient, and automated system for early automatic tracing and solving which can be extended to identify all possible diseases.

2. Future work

In future we have though of making our project more futuristic with fully automated drone system , In which we will select a land of field and then drones will do our full going work. As they will we collecting the database with the camera and sensors attached to them and then sending the details to our system then the system will respond and detect the exact disease in the exact particular area now drones will we provided with the solution and they will we spraying it all over the infected area.

This is a fully automated technique with no human touch and do an very efficient work which will take a lot of time . We had this vision and we are working on it to get the best results in future. With that there are some more points that can we upheld in the futuristic approaches such as-

- **Improved accuracy**: With more research and development, the accuracy of crop disease detection using CNNs is likely to improve. This will help farmers detect diseases early and take action to prevent crop loss.

- **Increased efficiency:** As the technology becomes more advanced, the time it takes to detect diseases will decrease, allowing for faster and more efficient disease management.

- **Enhanced accessibility:** The use of CNNs for crop disease detection can be made more accessible to farmers through the development of mobile applications that can be used with smartphones or tablets.

- **Better disease management:** By using CNNs for early detection, farmers can take steps to prevent the spread of disease, such as isolating infected crops and using targeted treatments.

- **Improved yields:** Early detection and management of crop diseases can help farmers achieve higher yields and reduce the risk of crop loss

5.3 References

1. Mohanty SP, Hughes DP and Salathé M (2016) Using Deep Learning for Image-Based Plant Disease Detection. Front. Plant Sci. 7:1419. doi: 10.3389/fpls.2016.01419
2. Wei Zhong, Ning Yu, Chunyu Ai. Applying Big Data Based Deep Learning System to Intrusion Detection. Big Data Mining and Anyalytics 2020, 3(3): 181-195.
3. TY - JOUR, Zhang, Baohua, Fan, Shuxiang, Li, Jiangbo, Huang, Wenqian, Zhao, Chunjiang, Qian, Man, Zheng, Ling PY - 2015/01/26SP "Detection of Early Rottenness on Apples by Using Hyperspectral Imaging Combined with Spectral Analysis and Image Processing", VL - 8, DO - 10.1007, /s12161-015-0097-7, Food Analytical Methods
4. T. Ren, Y. Zhang and C. Wang, "Identification of Corn Leaf Disease Based on Image Processing," 2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2019, pp. 165-168, doi: 10.1109/ ICISCAE48440.2019.221610.
5. Laixiang Xu, Bingxu Cao, Fengjie Zhao, Shiyuan Ning, Peng Xu, Wenbo Zhang, Xiangguan Hou, "Wheat leaf disease identification based on deep learning algorithms", Physiological and Molecular Plant Pathology, Volume 123, 2023,101940,ISSN 08855765,https://doi.org/10.1016/j.pmpp.2022.101940.
6. T. Ren, Y. Zhang and C. Wang, "Identification of Corn Leaf Disease Based on Image Processing," 2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2019, pp. 165-168, doi: 10.1109/ICISCAE48440.2019.221610.
7. BIG DATA MINING AND ANALYTICS ISSN 2096-0654 03/06 pp181–195 Volume 3, Number 3, September 2020 DOI: 10.26599/BDMA.2020.9020003
8. S. D. Khirade and A. B. Patil, "Plant Disease Detection Using Image Processing," 2015 International Conference on Computing Communication Control and Automation, Pune, India, 2015, pp. 768-771, doi: 10.1109/ICCUBEA.2015.153.

9. S. Ramesh et al., "Plant Disease Detection Using Machine Learning," 2018 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C), Bangalore, India, 2018, pp. 41-45, doi: 10.1109/ICDI3C.2018.00017.
10. L. Li, S. Zhang and B. Wang, "Plant Disease Detection and Classification by Deep Learning—A Review," in IEEE Access, vol. 9, pp. 56683-56698, 2021, doi: 10.1109/ ACCESS.2021.3069646.
11. E. Hirani, V. Magotra, J. Jain and P. Bide, "Plant Disease Detection Using Deep Learning," 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India, 2021, pp. 1-4, doi: 10.1109/I2CT51068.2021.9417910.
12. D. Gosai, B. Kaka, D. Garg, R. Patel and A. Ganatra, "Plant Disease Detection and Classification Using Machine Learning Algorithm," 2022 International Conference for Advancement in Technology (ICONAT), Goa, India, 2022, pp. 1-6, doi: 10.1109/ ICONAT53423.2022.9726036.
13. A.Lakshmanarao, M. R. Babu and T. S. R. Kiran, "Plant Disease Prediction and classification using Deep Learning ConvNets," 2021 International Conference on Artificial Intelligence and Machine Vision (AIMV), Gandhinagar, India, 2021, pp. 1-6, doi: 10.1109/ AIMV53313.2021.9670918.
14. S. Kumar, K. Prasad, A. Srilekha, T. Suman, B. P. Rao and J. N. Vamshi Krishna, "Leaf Disease Detection and Classification based on Machine Learning," 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India, 2020, pp. 361-365, doi: 10.1109/ICSTCEE49637.2020.9277379.

USER MANUAL

This user manual will guide you through the process of conducting early stage crop disease detection using an advanced computer vision model. Follow the steps below to achieve accurate and efficient detection of diseases in crops.

Step 1: Obtain the Dataset

To train an effective crop disease detection model, you need a dataset that consists of images of healthy crops and crops affected by various diseases. Ensure that the dataset is diverse, representative of different crops, and contains labeled annotations indicating the presence of diseases.

Step 2: Preprocess and Augment the Dataset

Preprocess the dataset to ensure consistent image sizes, correct color channels, and appropriate normalization. Additionally, consider augmenting the dataset by applying transformations such as rotations, flips, and zooms to increase its size and variability. This step helps the model learn robust features and generalize better.

Step 3: Train the Crop Disease Detection Model

Select a suitable computer vision model, such as a convolutional neural network (CNN), and train it on the preprocessed and augmented dataset. Divide the dataset into training and validation sets to evaluate the model's performance. During training, optimize the model's parameters using appropriate loss functions and optimization algorithms. Iterate through multiple training epochs until the model achieves satisfactory performance.

Step 4: Evaluate the Model

After training, evaluate the performance of the crop disease detection model using the validation set. Calculate metrics such as accuracy, precision, recall, and F1-score to assess the model's ability to detect diseases accurately. Adjust the model's hyperparameters, architecture, or dataset if necessary to improve its performance.

Step 5: Prepare Test Images

Collect a set of test images that represent real-world scenarios. These images should include crops affected by various diseases as well as healthy crops. Ensure that the test images are distinct from the training and validation sets to evaluate the model's generalization capability effectively.

Step 6: Perform Crop Disease Detection

Use the trained crop disease detection model to predict diseases in the test images. Apply the following steps:

Preprocess the test image: Resize the image to match the input size expected by the model. Perform any necessary normalization or color channel adjustments.

Run the inference: Pass the preprocessed image through the trained model to obtain predictions. The model will identify the presence of diseases and provide corresponding probabilities or confidence scores.

Post-process the results: Set an appropriate threshold to determine if a crop is diseased or healthy based on the model's output probabilities. Apply non-maximum suppression if multiple disease detections overlap.

Visualize the results: Overlay the detected disease regions or bounding boxes on the original test image. Optionally, display the predicted disease labels and confidence scores.

Interpret and analyze the results: Assess the accuracy and reliability of the crop disease detection by comparing the model's predictions with ground truth information. Analyze any false positives or false negatives to understand the model's limitations and areas for improvement.