Potato Disease Classification Using Deep Learning

1. INTRODUCTION

1.1 Project Overview

Globally, the production of potatoes is a major agricultural activity that supports both economic stability and food security. On the other hand, a number of diseases can affect potato plants, which can lower crop quality and output. For efficient illness management, these diseases must be identified and classified as soon as possible. The goal of this research is to create a system that reliably classifies potato diseases using Convolutional Neural Networks (CNN) and deep learning techniques, enabling farmers to make well-informed decisions about disease prevention and treatment.

1.2 Purpose

Predicting potato leaf disease as soon as possible is crucial because it can have significant impacts on crop yield and quality. Early detection allows farmers to take prompt action to prevent the spread of the disease and reduce crop damage. Here are some reasons why predicting potato leaf disease early is essential:

1. Minimize crop loss: Potato leaf disease can significantly reduce crop yield and quality. By predicting the disease early, farmers can take timely measures to control its spread, thereby minimizing crop loss.
2. Reduce cost: Early detection of potato leaf disease can help farmers reduce costs associated with disease control measures, such as pesticides and other treatments. By identifying the disease early, farmers can target the specific area of the crop affected, which helps in reducing the overall cost of control measures.
3. Protect the environment: The excessive use of pesticides and other control measures can have negative impacts on the environment. Early detection of potato leaf disease can help farmers target only the affected areas and minimize the use of pesticides, reducing their environmental impact.
4. Improve crop quality: Potato leaf disease can affect the quality of the crop, making it less desirable to buyers. By predicting the disease early, farmers can take appropriate measures to prevent its spread, thereby improving the overall quality of the crop.

2. LITERATURE SURVEY

2.1 Existing problem

Plant disease classification challenges have seen the successful application of deep learning techniques, particularly Convolutional Neural Networks (CNNs), in numerous research. Transfer learning from pretrained models such as VGG19, ResNet50, and Inception has been used by researchers to effectively classify plant illnesses, especially potato infections. These models can identify a wide range of plant diseases with high accuracy, which helps with early identification and management. Large and diversified datasets, data augmentation techniques, and model interpretability are all highlighted in this literature as being vital factors to take into account while creating a potato disease classification system.

High-quality datasets are essential for training deep learning models that classify plant diseases, as previous studies have shown. Model development has benefited greatly from the availability of existing datasets, such as the Plant Village dataset and others. To increase the resilience and generalisation of models, researchers have used data preprocessing techniques such image scaling, normalisation, and data augmentation. In order to address frequent obstacles in plant disease classification, such as uneven class distribution and differences in lighting and background, some research has looked into domain-specific data augmentation techniques.

The literature emphasises the importance of developing accurate models, but it also emphasises the necessity of creating deployment interfaces that are easy to use. The combination of deep learning models with approachable applications to enable farmers to quickly identify agricultural illnesses has been covered in a number of publications. These apps must to work with web browsers, mobile devices, and Internet of Things gadgets, and they ought to give consumers dependable and timely results. In order to guarantee the system's practical applicability in actual agricultural settings, interpretability and user input integration are frequently explored in this domain's literature.

In summary, the body of current research offers a solid framework for creating a deep learning-based system for classifying potato diseases. It emphasises the crucial roles that high-quality datasets, data preparation, model architecture, and user-friendly deployment have in improving crop management in the agriculture industry and helping farmers.

2.2 References

1. https://www.cambridge.org/core/journals/advances-in-animal-biosciences/article/abs/potato-disease-classification-using-convolution-neural-networks/E9303F667377BD763C3054CB8488D36C
2. https://link.springer.com/chapter/10.1007/978-981-13-8406-6_37
3. https://ieeexplore.ieee.org/abstract/document/9231784
4. https://philpapers.org/rec/ELSPCU
5. https://ieeexplore.ieee.org/abstract/document/8905128

2.3 Problem Statement Definition

An important part of world agriculture is the production of potatoes, which are both a basic food and a source of income for many farmers. On the other hand, a number of diseases that can seriously affect crop quality and productivity can affect potato plants. Effective disease management depends on the quick and precise detection of these conditions. Conventional disease identification techniques frequently call for the manual inspection of agricultural specialists, which can take time and be unavailable to all farmers. Convolutional neural networks (CNNs) and deep learning techniques are being used to create an automated and precise system for classifying potato diseases in order to meet this challenge. By helping farmers make knowledgeable decisions about disease prevention and treatment, this technology will eventually increase crop productivity and food security

Important elements of the issue statement:

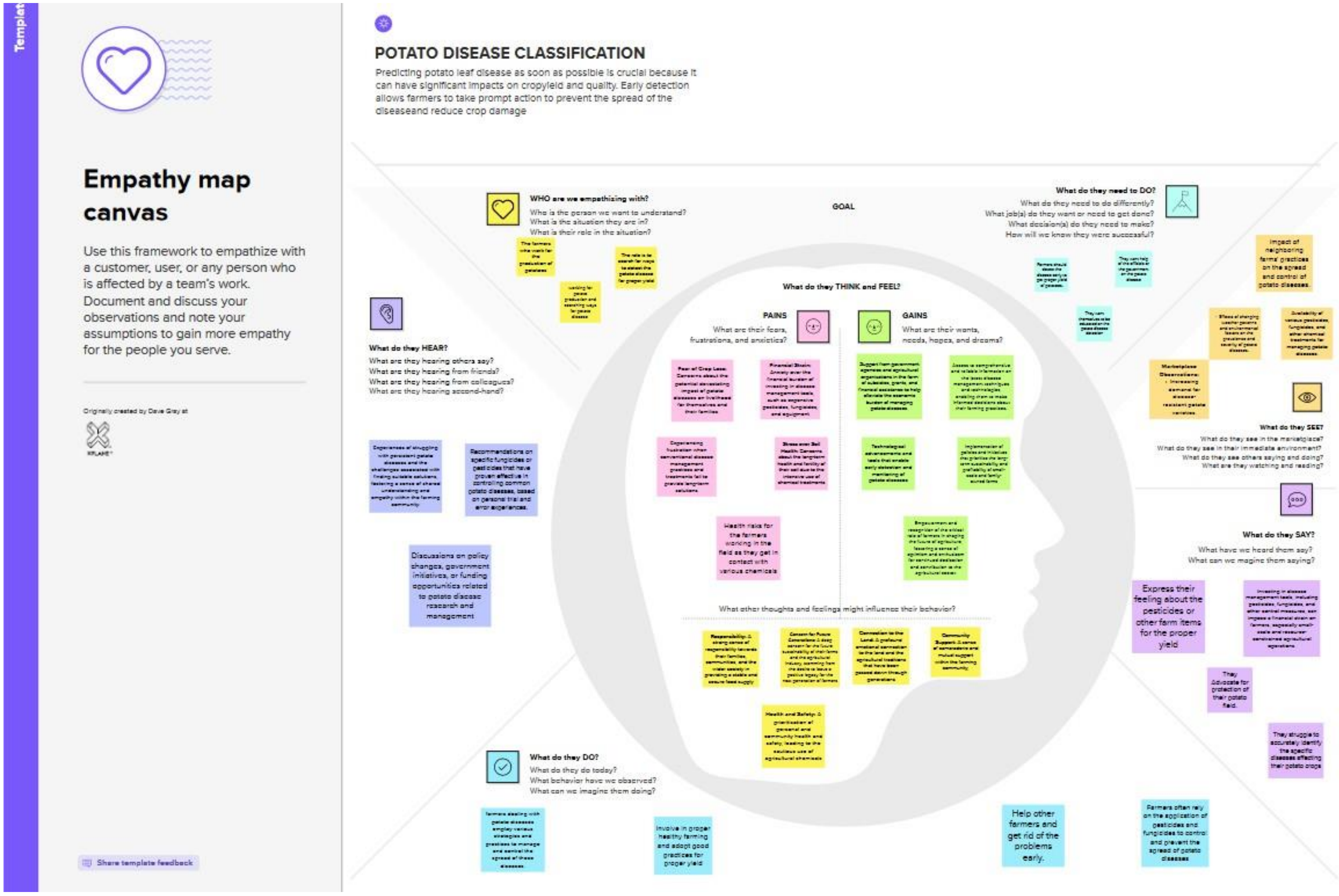
The main challenge is developing a trustworthy system for recognising and categorising the several diseases that impact potato plants, including potato scab, late blight, and early blight. The technology should be able to distinguish between plants that are healthy and those that are ill, and it should give precise details on the kind of disease that is present.

Timeliness and Precision: The system should deliver accurate and quick disease classification data so that farmers can begin managing and treating the illnesses that have been recognised right away. This takes care of the requirement for early disease identification, which can cut down on crop losses and excessive pesticide use.

User-Friendly Interface: To solve this problem effectively, a user-friendly interface or application should be developed, allowing farmers to easily upload images of their potato plants and receive disease classification results. The system should be intuitive and adaptable to the technological and educational background of the users.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



Empathy map canvas

Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.

Originally created by Dave Gray at XPLANE

POTATO DISEASE CLASSIFICATION

Predicting potato leaf disease as soon as possible is crucial because it can have significant impacts on cropyield and quality. Early detection allows farmers to take prompt action to prevent the spread of the diseaseand reduce crop damage

WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?

The farmers who work for the production of potatoes

working for potato production and searching ways for potato disease

The role is to search for ways to detect the potato disease for proper yield

GOAL

What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?

Experiences of struggling with persistent potato diseases and the challenges associated with finding suitable solutions, fostering a sense of shared understanding and empathy within the farming community

Recommendations on specific fungicides or pesticides that have proven effective in controlling common potato diseases, based on personal trial and error experiences.

Discussions on policy changes, government initiatives, or funding opportunities related to potato disease research and management

What do they THINK and FEEL?

PAINS
What are their fears, frustrations, and anxieties?

Fear of Crop Loss: Concerns about the potential devastating impact of potato diseases on livelihood for themselves and their families

Financial Strain: Anxiety over the financial burden of investing in disease management tools, such as expensive pesticides, fungicides, and equipment

Experiencing frustration when conventional disease management practices and treatments fail to provide longterm solutions

Stress over Soil Health: Concerns about the longterm health and fertility of their soil due to the intensive use of chemical treatments

Health risks for the farmers working in the field as they get in contact with various chemicals

GAINS
What are their wants, needs, hopes, and dreams?

Support from government agencies and agricultural organizations in the form of subsidies, grants, and financial assistance to help alleviate the economic burden of managing potato diseases

Access to comprehensive and reliable information on the latest disease management techniques and technologies enabling them to make informed decisions about their farming practices

Technological advancements and tools that enable early detection and monitoring of potato diseases

Implementation of policies and initiatives that prioritize the long-term sustainability and profitability of small-scale and family-owned farms

Empowerment and recognition of the critical role of farmers in shaping the future of agriculture, fostering a sense of optimism and enthusiasm for continued dedication and contribution to the agricultural sector

What other thoughts and feelings might influence their behavior?

Responsibility: A strong sense of responsibility towards their families, communities, and the wider society in providing a stable and secure food supply

Concern for Future Generations: A deep concern for the future sustainability of their farms and the agricultural industry, stemming from the desire to leave a positive legacy for the next generation of farmers

Connection to the Land: A profound emotional connection to the land and the agricultural traditions that have been passed down through generations

Community Support: A sense of camaraderie and mutual support within the farming community

Health and Safety: A prioritization of personal and community health and safety, leading to the cautious use of agricultural chemicals

What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?

Farmers should detect the disease early to get proper yield of potatoes.

They can take help of the officials or the government on the potato disease

They can themselves to be educated on the potato disease detection

Impact of neighboring farms' practices on the spread and control of potato diseases.

Effects of changing weather patterns and environmental factors on the prevalence and severity of potato diseases.

Availability of various pesticides, fungicides, and other chemical treatments for managing potato diseases.

Marketplace Observations:
- Increasing demand for disease-resistant potato varieties

What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?

What do they SAY?
What have we heard them say?
What can we magine them saying?

Express their feeling about the pesticides or other farm items for the proper yield

Investing in disease management tools, including pesticides, fungicides, and other control measures, can impose a financial strain on farmers, especially small-scale and resource-constrained agricultural operations

They Advocate for protection of their potato field.

They struggle to accurately identify the specific diseases affecting their potato crops

What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?

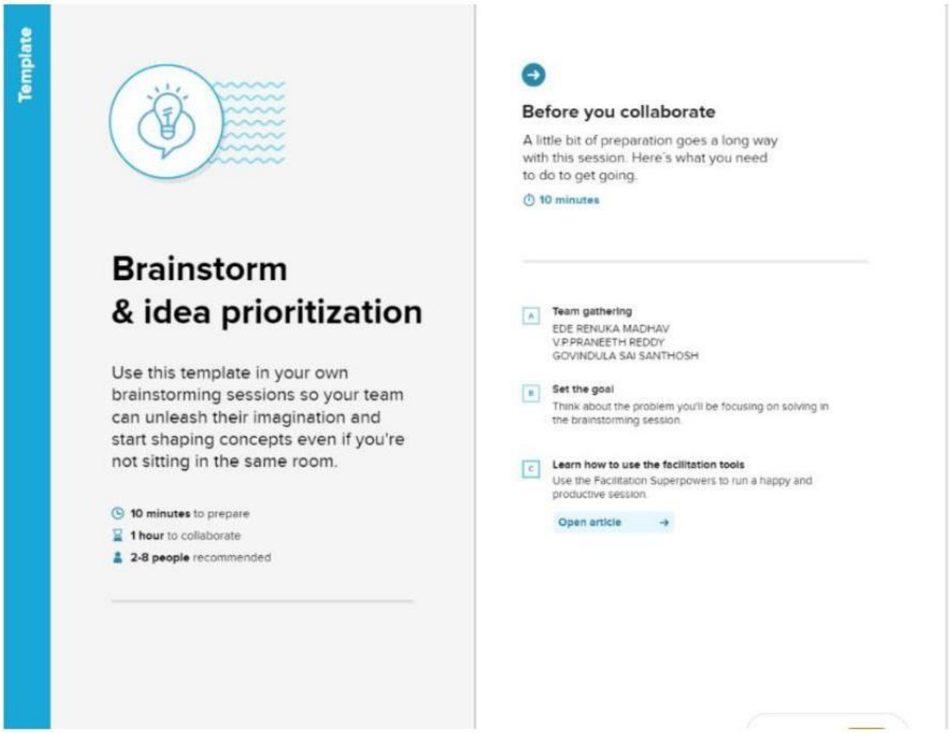
farmers dealing with potato diseases employ various strategies and practices to manage and control the spread of these diseases

Involve in proper healthy farming and adopt good practices for proper yield

Help other farmers and get rid of the problems early.

Farmers often rely on the application of pesticides and fungicides to control and prevent the spread of potato diseases

3.2 Ideation & Brainstorming



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **10 minutes** to prepare
- **1 hour** to collaborate
- **2-8 people** recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

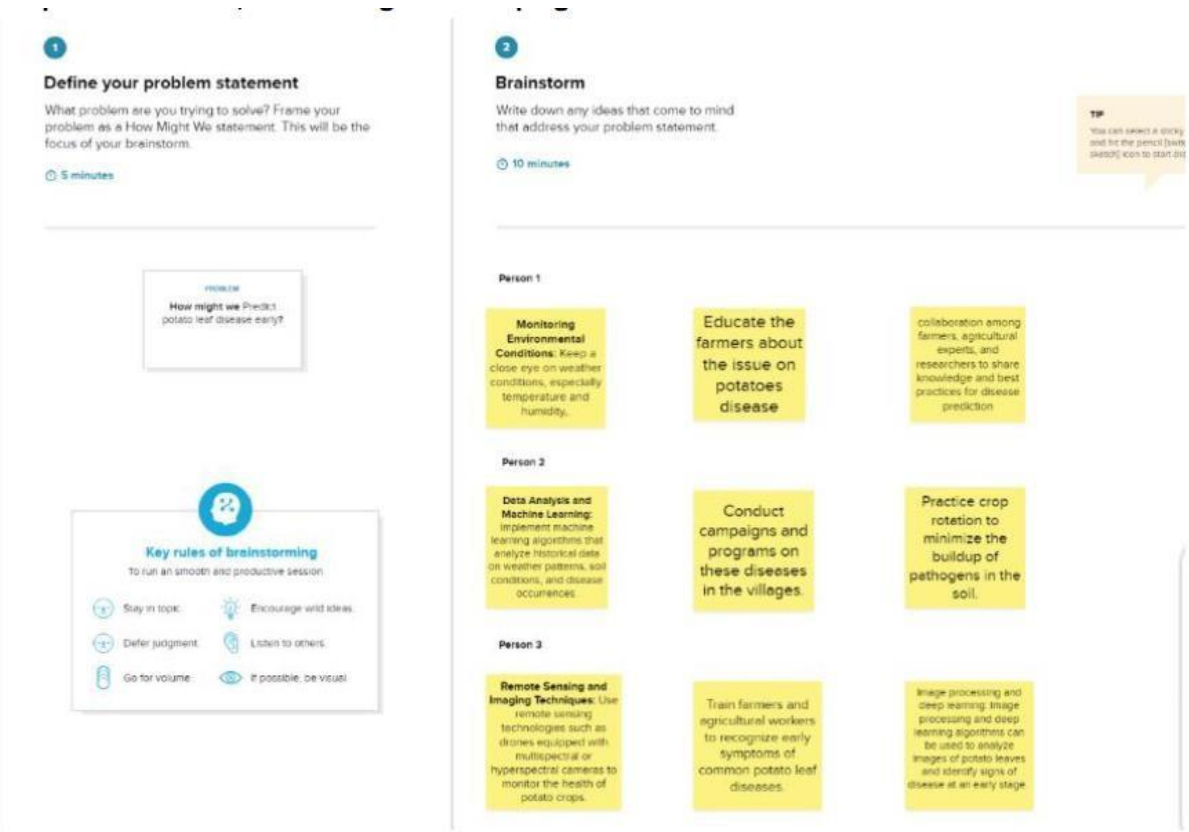
10 minutes

A **Team gathering**
EDE RENUKA MADHAV
V.P.PRANEETH REDDY
GOVINDULA SAI SANTHOSH

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →



1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we Predict potato leaf disease early?

Key rules of brainstorming
To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky and hit the pencil [swit sketch] icon to start dr

Person 1

Monitoring Environmental Conditions: Keep a close eye on weather conditions, especially temperature and humidity.

Educate the farmers about the issue on potatoes disease

collaboration among farmers, agricultural experts, and researchers to share knowledge and best practices for disease prediction

Person 2

Data Analysis and Machine Learning: implement machine learning algorithms that analyze historical data on weather patterns, soil conditions, and disease occurrences.

Conduct campaigns and programs on these diseases in the villages.

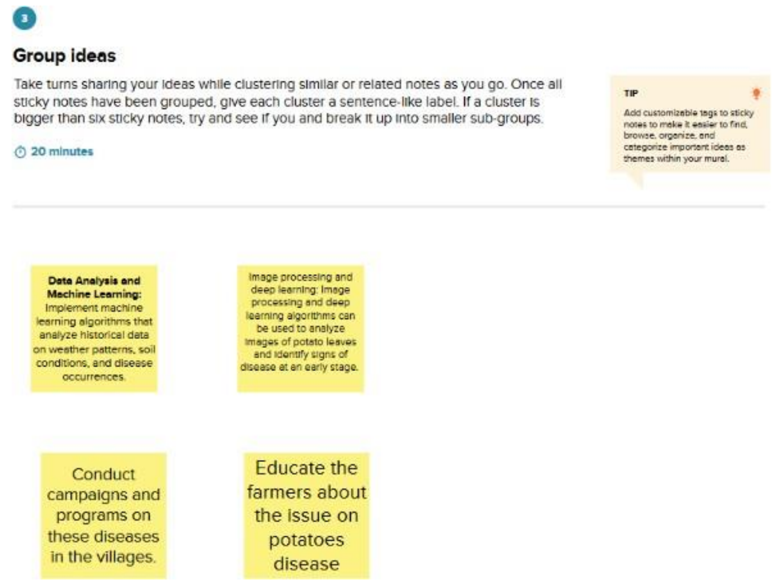
Practice crop rotation to minimize the buildup of pathogens in the soil.

Person 3

Remote Sensing and Imaging Techniques: Use remote sensing technologies such as drones equipped with multispectral or hyperspectral cameras to monitor the health of potato crops.

Train farmers and agricultural workers to recognize early symptoms of common potato leaf diseases.

Image processing and deep learning: Image processing and deep learning algorithms can be used to analyze images of potato leaves and identify signs of disease at an early stage.



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

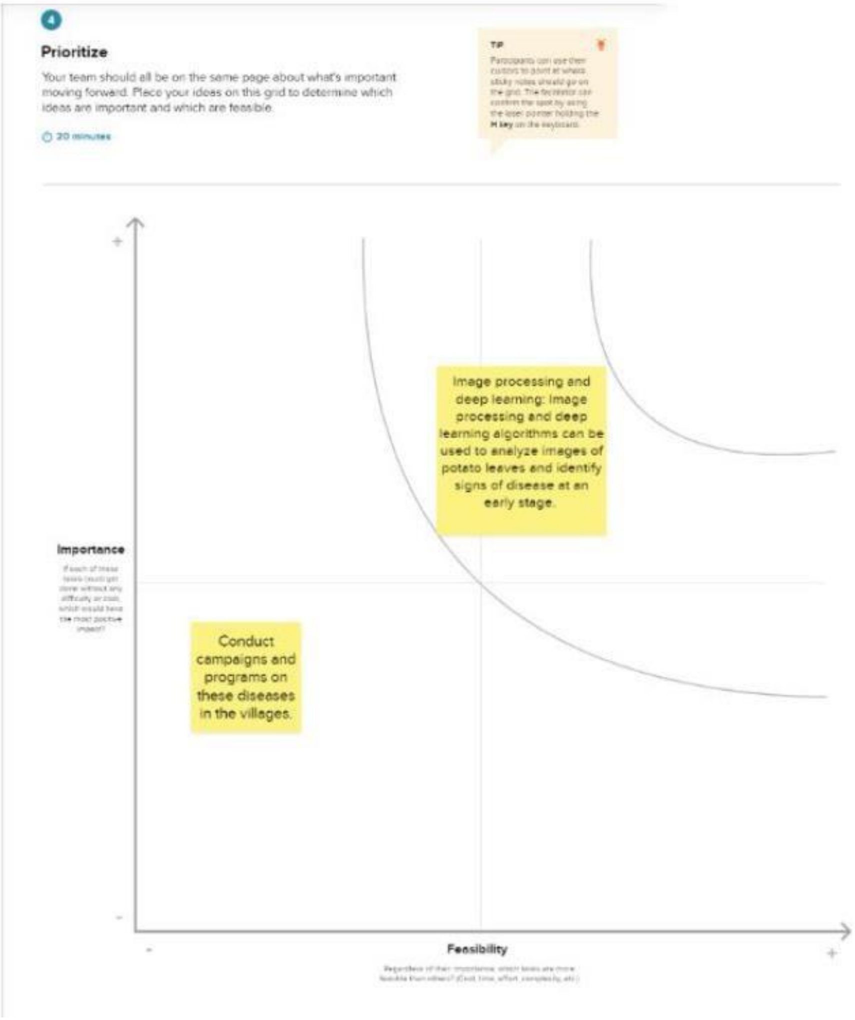
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Data Analysis and Machine Learning: Implement machine learning algorithms that analyze historical data on weather patterns, soil conditions, and disease occurrences.

Image processing and deep learning: Image processing and deep learning algorithms can be used to analyze images of potato leaves and identify signs of disease at an early stage.

Conduct campaigns and programs on these diseases in the villages.

Educate the farmers about the issue on potatoes disease



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Image processing and deep learning: Image processing and deep learning algorithms can be used to analyze images of potato leaves and identify signs of disease at an early stage.

Conduct campaigns and programs on these diseases in the villages.

+
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
-
-
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)
+

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Image Upload and Processing: In order to classify diseases, users (farmers) should be able to upload photos of potato plants to the system. To guarantee that these photographs are compatible with the deep learning model, it should preprocess them, resizing and normalising them as needed.

Disease Classification: The system's main job is to correctly identify potato illnesses from the supplied photos. It should be able to distinguish between distinct disease states and healthy plant states, giving each case a unique disease identifier.

User Interface: A user-friendly interface that is compatible with PCs, tablets, and smartphones must be provided by the system. It ought to offer a simple and intuitive user interface to suit consumers with varying degrees of technological expertise.

Model Interpretability: The system ought to provide justifications or illustrations of the model's decision-making procedure, giving users an understanding of the rationale behind the classification of a certain ailment. This feature improves user comprehension and trust.

Scalability and Integration: To enable wider acceptance and influence within the agricultural sector, make sure the system is scalable and compatible with other agricultural technology and information systems.

4.2 Non-Functional requirements

Accuracy and Precision: For farmers to receive trustworthy findings, the disease classification system needs to attain a high degree of accuracy. It should also minimise false positives and false negatives by precisely identifying diseases.

Response Time: In order to enable farmers to take rapid action, the system should provide disease categorization results as soon as possible, ideally in real-time or within a few seconds of image input.

Security and Privacy: Put security measures in place to guard user information and guarantee the privacy of photos that users post. Images and personal data should be treated carefully and in accordance with applicable data protection laws.

Robustness and Reliability: Strong and dependable, the system should be able to adapt to changes in disease stages, lighting, and image quality. In the event of an unplanned outage, failover procedures ought to be included.

Scalability and Performance: Make sure the system is capable of managing a sizable volume of user requests and expanding to accommodate a rising user base without experiencing performance issues. Optimization and load balancing ought to be implemented.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

Data Collection: Obtaining a wide range of potato leaf image datasets from image archives, field surveys, and other sources is the task of this phase. Following collection, the photos are kept in a raw data repository**.**

Data Pre-processing: Images of raw potato leaves are pre-processed in order to get them ready for model training. To improve dataset diversity, this can involve resizing photographs, normalising pixel values, and using data augmentation techniques.

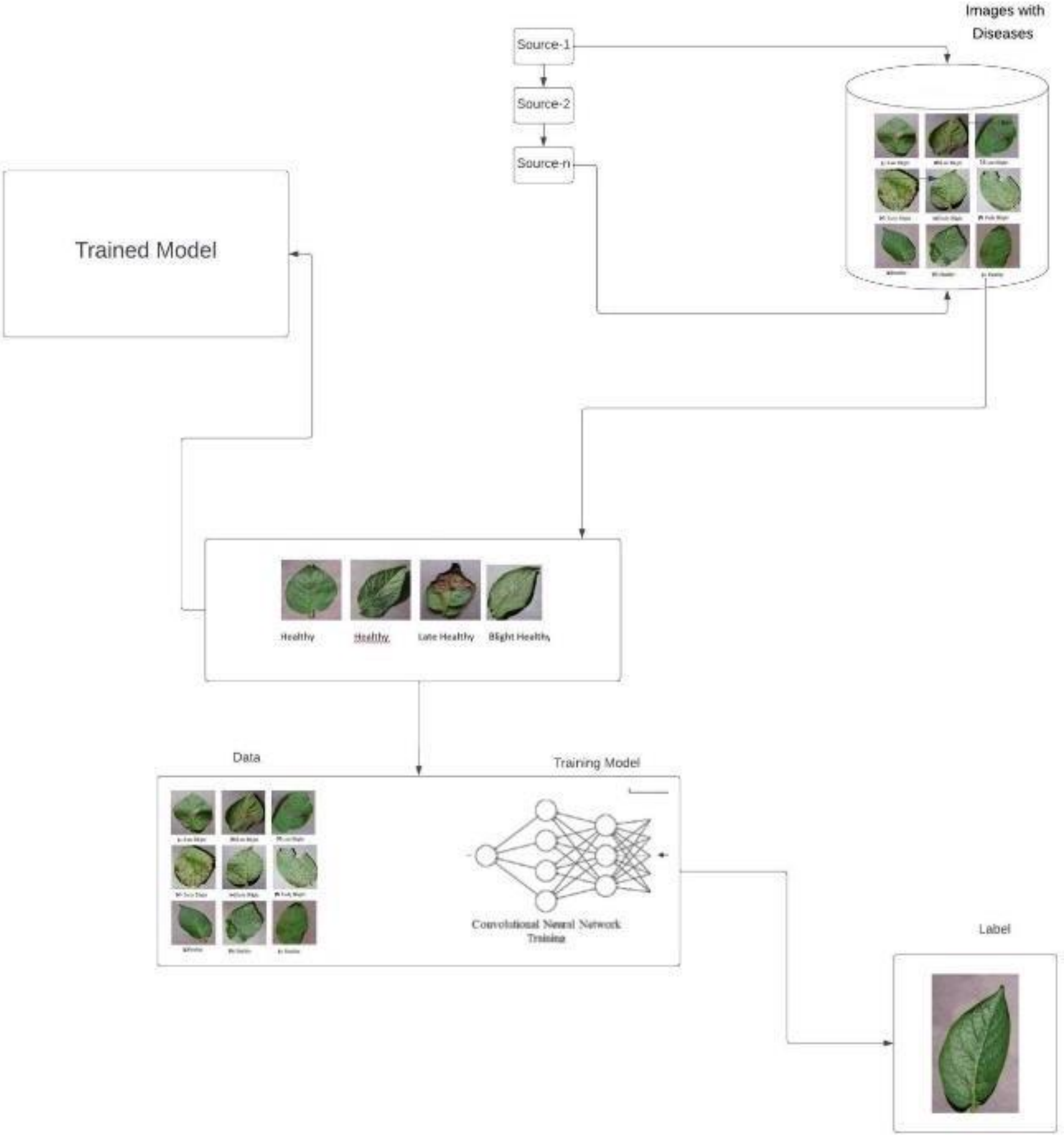
Model Training: This step involves using the pre-processed data to train a deep learning model so that it can identify different potato leaf diseases. For later use, the trained model is stored.

Model Evaluation: The performance of the trained model is assessed using a separate dataset not used in training, to gauge its accuracy, sensitivity, and specificity in disease classification.

Model Deployment: This step involves deploying the trained model to make it accessible for real-world disease classification applications, either on local devices or in the cloud.

User Interaction: Through an intuitive application or API, end users engage with the deployed model, allowing them to upload photographs of potato leaves for disease identification and obtain timely results.

Data Flow Diagram:



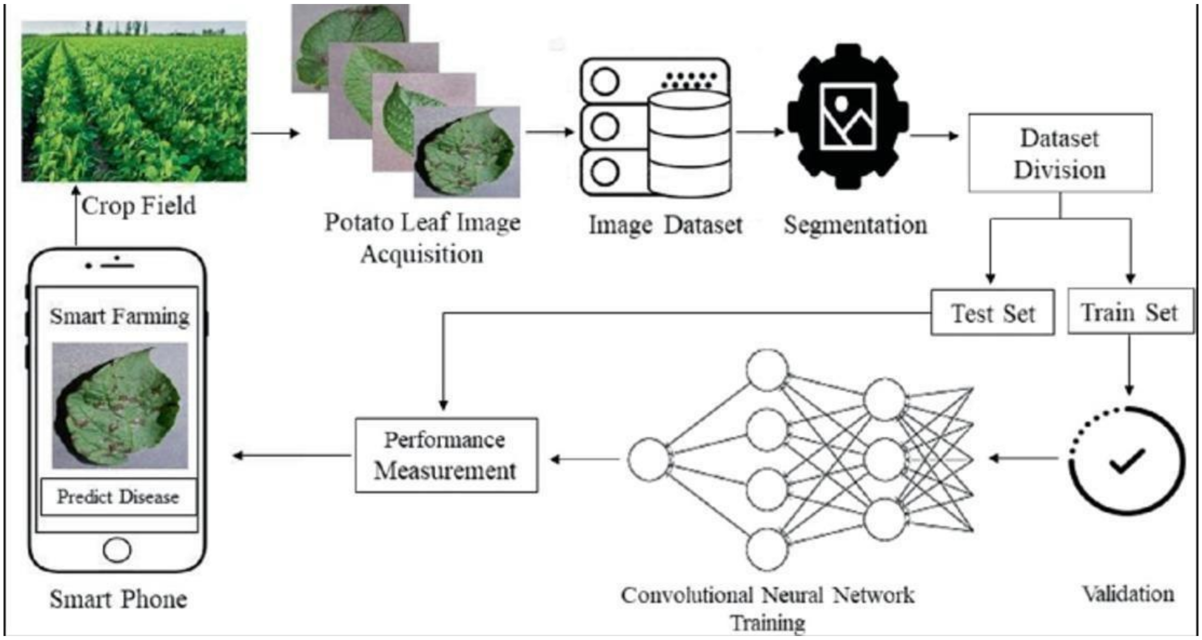
User Stories:

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
potatofarmer	Farming	USN-1	I want a potato disease classification system that can quickly identify the specific diseases affecting my crop, so that I can implement timely and targeted treatments to prevent further damage and ensure a successful harvest.	access the disease classification system viaa user-friendly interface, either through a web application or a mobile app.	High	Sprint-1
research scientist	Research On Plants	USN-2	I need a robust potato disease classification tool that can accurately differentiate between various types of diseases affecting potato crops, enablingme to conduct in-depth analyses and develop effective strategies for disease management and prevention.	potato disease classification system should be equipped withan extensive database that covers a wide rangeof potato diseases, including both common and rare occurrences, toprovide comprehensive information for research purposes.	High	Sprint-1
agronomist		USN-3	I require a user-friendly potato disease classification application that can be easily accessed and utilized in the field, allowingme to provide real-time guidance and support to farmers in diagnosing and addressing disease issues, thus improvingoverall crop health and yield.	The potato disease classification system should provide timely andaccurate information on various potato diseases, including their symptoms,causes, and appropriate management strategies, enabling agronomists to make informed decisions and provide effective guidance to farmers.	Medium	Sprint-2
government agricultural officer		USN-4	I am responsible for monitoring and controlling potato diseases at a regional level. I need an advanced potato disease classification system that can efficiently process large volumes of data, enabling me to make informed decisions and implement preventive measures to safeguard the agricultural industry and	It should have the capability to handle a large volume of data and provide real-time analysis, enabling government agricultural officers to make informed decisions and implement timely interventions for disease	Medium	Sprint-1

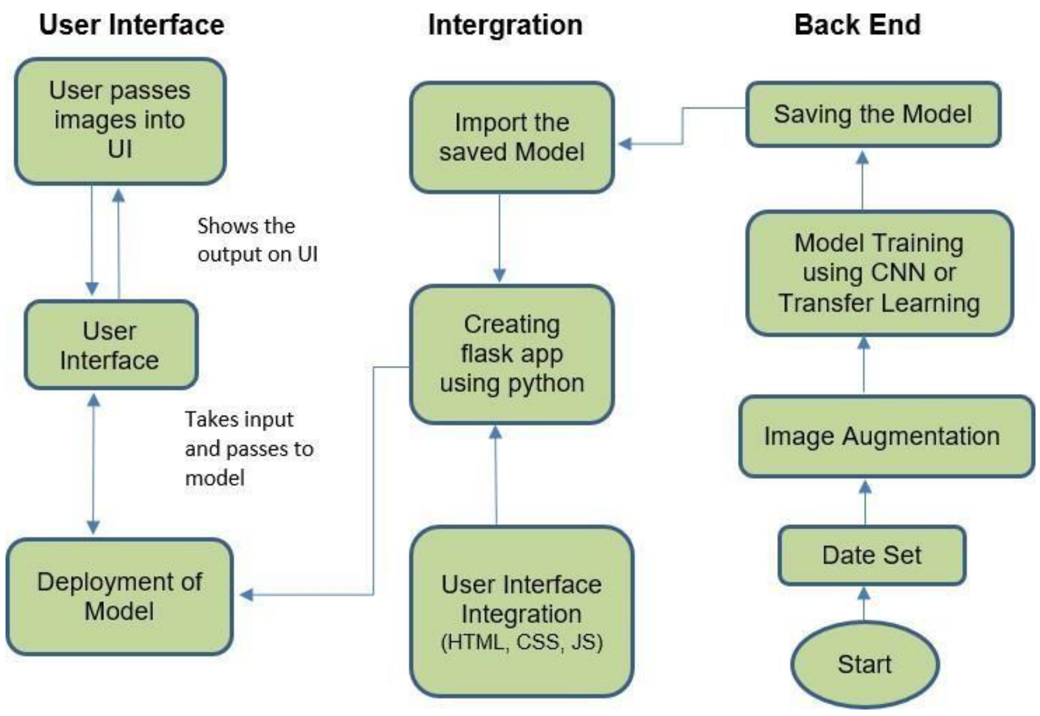
			ensure food security within the region.	control and prevention at a regional or national level.		
student studying plant pathology		USN-5	I am interested in learning about various potato diseases and their classifications. Iwould benefit from an interactive and educational potato disease classification platform that provides comprehensive information , helping me to understand thecomplexities of potato diseases and their impact on global agriculture.	The potato disease classification system should provide comprehensive and detailed information about various potato diseases, including their classifications, etiology, symptoms, and management strategies, enabling students to deepen their understanding of plant pathology.	High	Sprint-1

5.2 Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture


User Interface
Intergration
Back End
User passes images into UI
Shows the output on UI
User Interface
Takes input and passes to model
Deployment of Model
Import the saved Model
Creating flask app using python
User Interface Integration (HTML, CSS, JS)
Saving the Model
Model Training using CNN or Transfer Learning
Image Augmentation
Date Set
Start

6.2 Sprint Planning & Estimation

Use the below template to create product backlog and sprint schedule

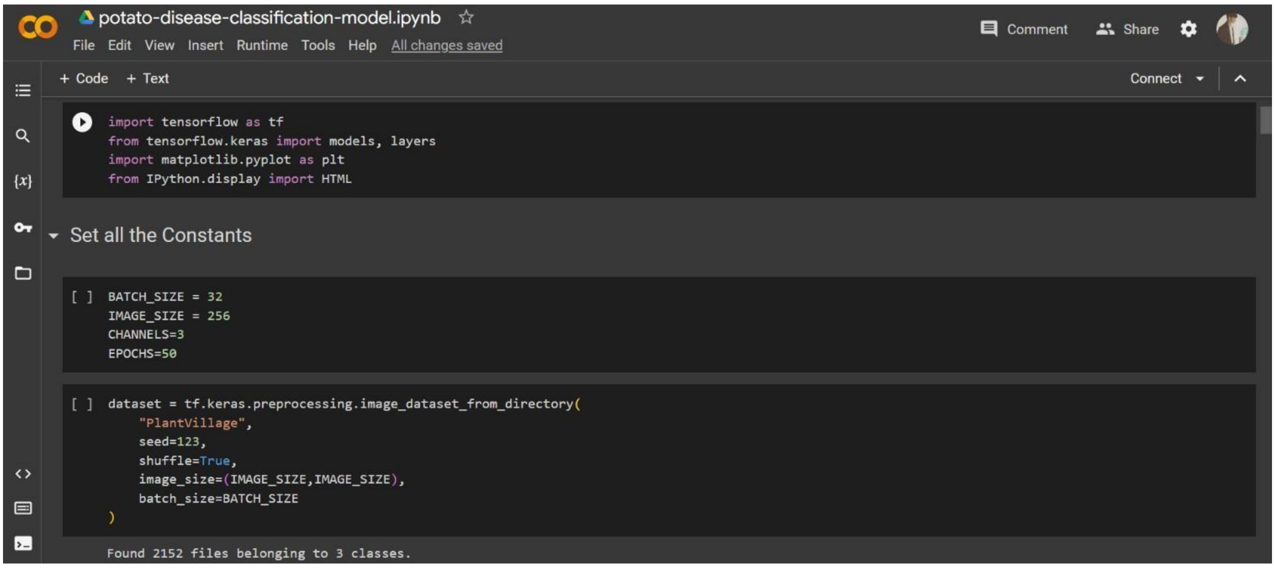
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Disease Classification	USN-1	As a user, I can upload images of potato plants affected by various diseases for classification.	3	High	SOMA UDAY KIRAN
Sprint-1	Disease Classification	USN-2	As a user, I can view the predicted disease class and its probability for the uploaded potato plant image.	2	High	BOLLI VAMSHI KRISHNA
Sprint-2	Disease Classification	USN-3	As a user, I can access historical data and analysis of previously classified potato diseases.	2	Medium	GOLLA BALA GANESH
Sprint-2	Disease Classification	USN-4	As a user, I can provide feedback on the accuracy of the disease classification for continuous model improvement.	1	Medium	
Sprint-3	Disease Classification	USN-5	As a user, I can receive recommendations for disease management strategies based on the classified potato disease.	3	Low	

6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022
Sprint-2	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022
Sprint-3	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022
Sprint-4	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022

7. CODING & SOLUTIONING

Activity 1.1: Importing the libraries



```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from IPython.display import HTML
```

Set all the Constants

```
[ ] BATCH_SIZE = 32
    IMAGE_SIZE = 256
    CHANNELS=3
    EPOCHS=50
```

```
[ ] dataset = tf.keras.preprocessing.image_dataset_from_directory(
        "PlantVillage",
        seed=123,
        shuffle=True,
        image_size=(IMAGE_SIZE,IMAGE_SIZE),
        batch_size=BATCH_SIZE
    )

    Found 2152 files belonging to 3 classes.
```

Activity 2: Data Preparation

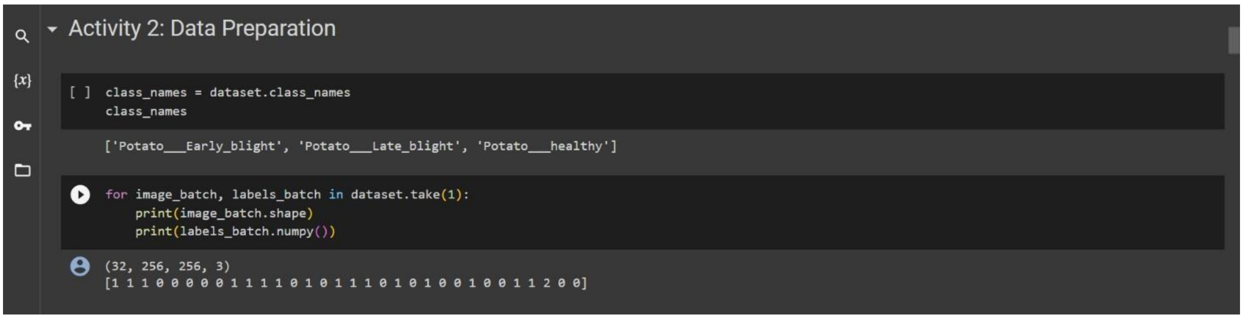
As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the deep learning model. We have to extract the class names of the data and let's try to visualise the data with labels.

- Extracting class names.
- Visualising the data

Activity 2.1: Extracting class names

- Let’s find the class names of our dataset first. To find the class names of the dataset, we can use .class_names.



Activity 2: Data Preparation

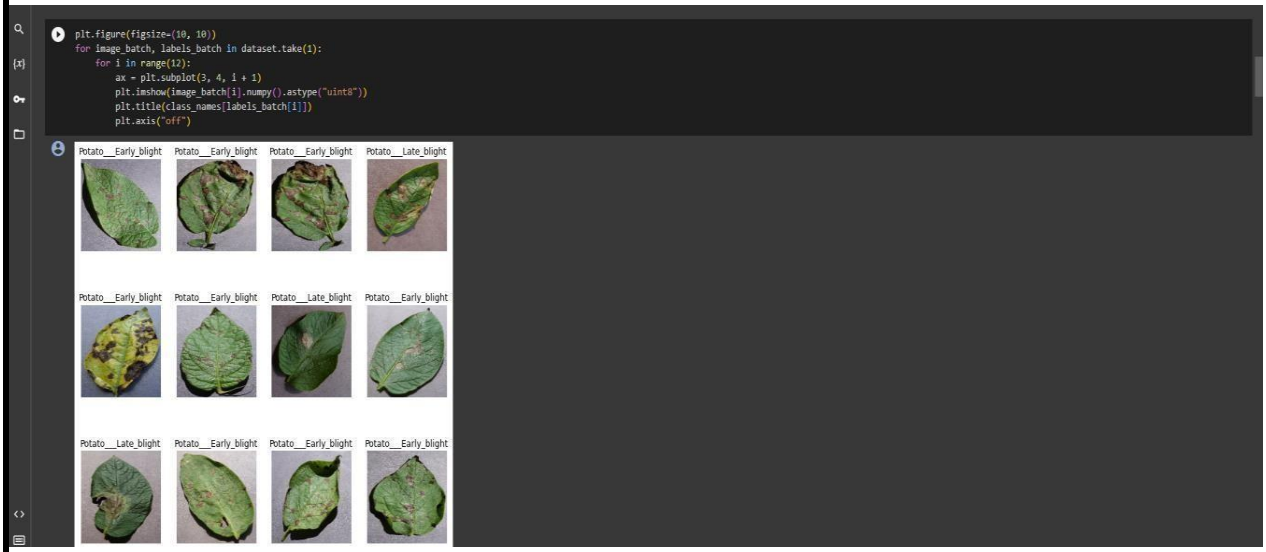
```
[ ] class_names = dataset.class_names
    class_names
```

```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

```
for image_batch, labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
```

```
(32, 256, 256, 3)
[1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 0 1 0 1 0 0 1 0 0 1 1 2 0 0]
```

Activity 2.2: Visualising the data



```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.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")
```

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features. Which are not suitable for our dataset.

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

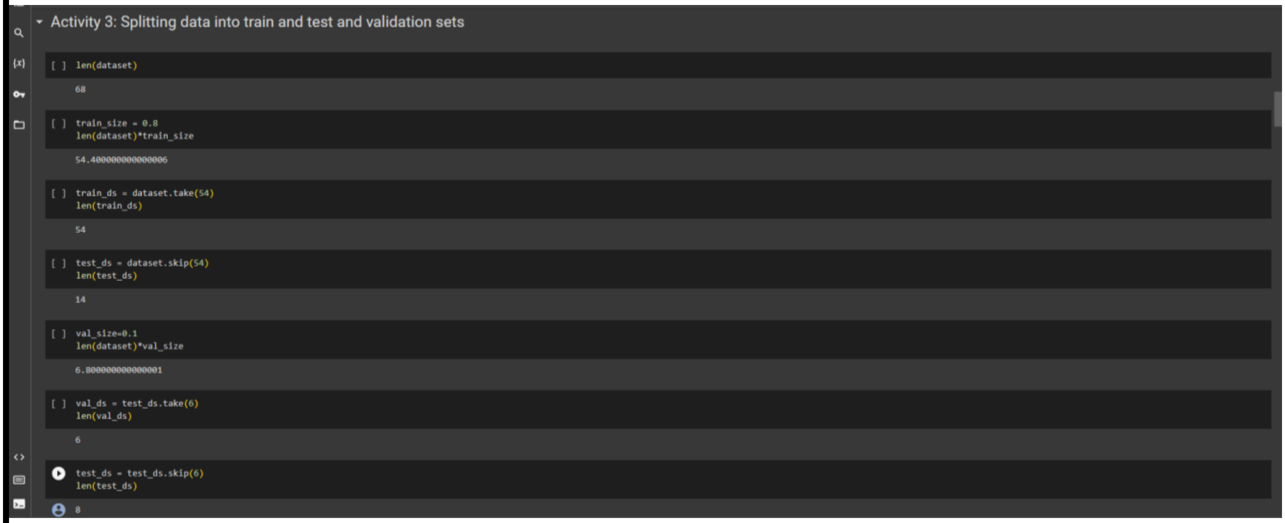
This is not needed for our current dataset as it is an image

dataset. Visual analysis is mostly used for numerical data.

Activity 3: Splitting data into train and test and validation sets

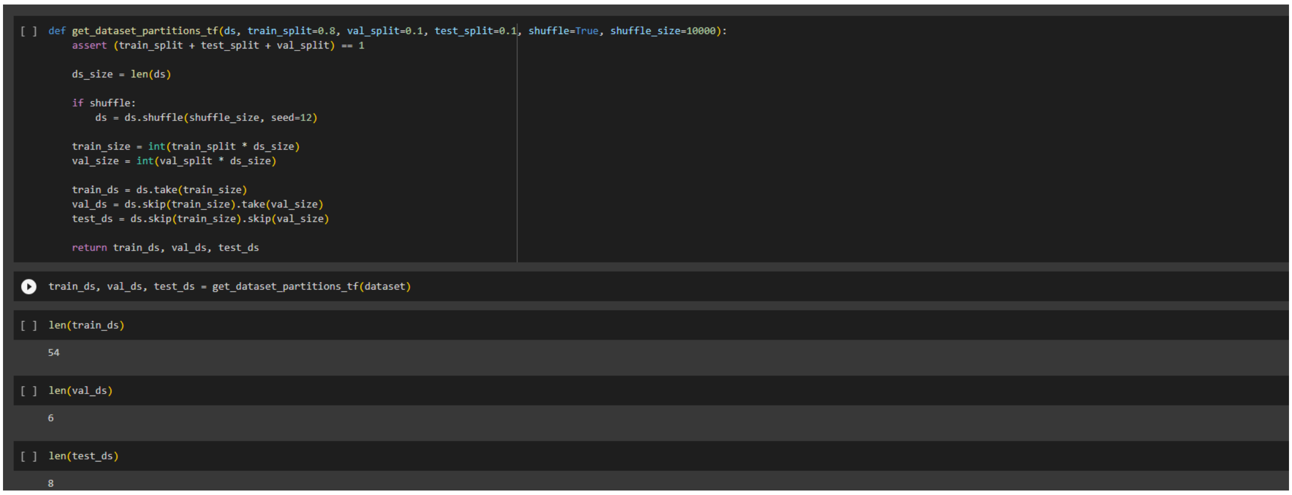
Now let’s split the Dataset into train, test and validation sets. First split the dataset into train and test sets.

The spilt will be in 8:1:1 ratio : train : test : validation respectively.



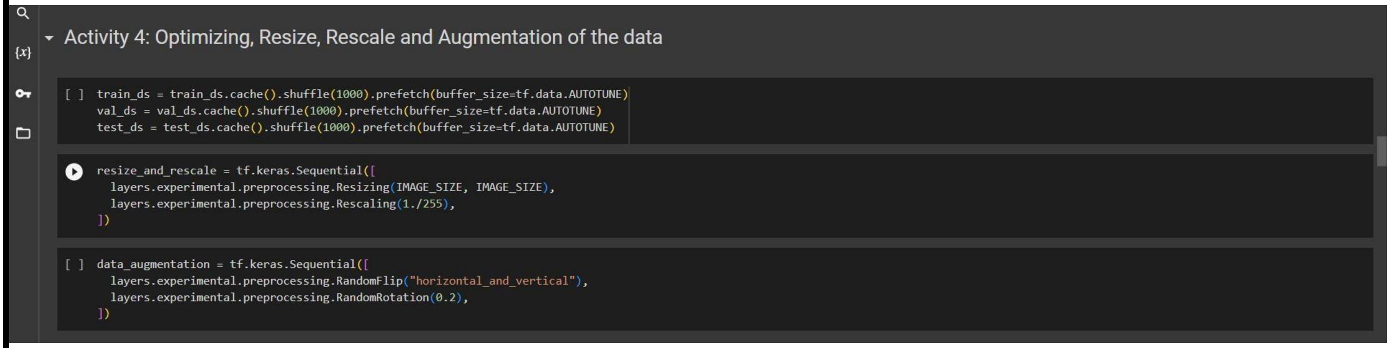
Activity 3: Splitting data into train and test and validation sets

```
[ ] len(dataset)

    68

[ ] train_size = 0.8
    len(dataset)*train_size

    54.400000000000006

[ ] train_ds = dataset.take(54)
    len(train_ds)

    54

[ ] test_ds = dataset.skip(54)
    len(test_ds)

    14

[ ] val_size=0.1
    len(dataset)*val_size

    6.800000000000001

[ ] val_ds = test_ds.take(6)
    len(val_ds)

    6

    test_ds = test_ds.skip(6)
    len(test_ds)

    8
```



```
[ ] 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

    train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)

[ ] len(train_ds)

    54

[ ] len(val_ds)

    6

[ ] len(test_ds)

    8
```

Activity 4: Optimizing, Resize, Rescale and Augmentation of the data



Activity 4: Optimizing, Resize, Rescale and Augmentation of the data

```
[ ] 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)
```

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

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

Prefetching and Caching saves a lot of time in accessing the data from disks. Refer this link for better understanding.

Resize and Rescale is used to maintain the uniformity among the data.

Augmentation helps you to increase the amount of data and helps your model to learn better.

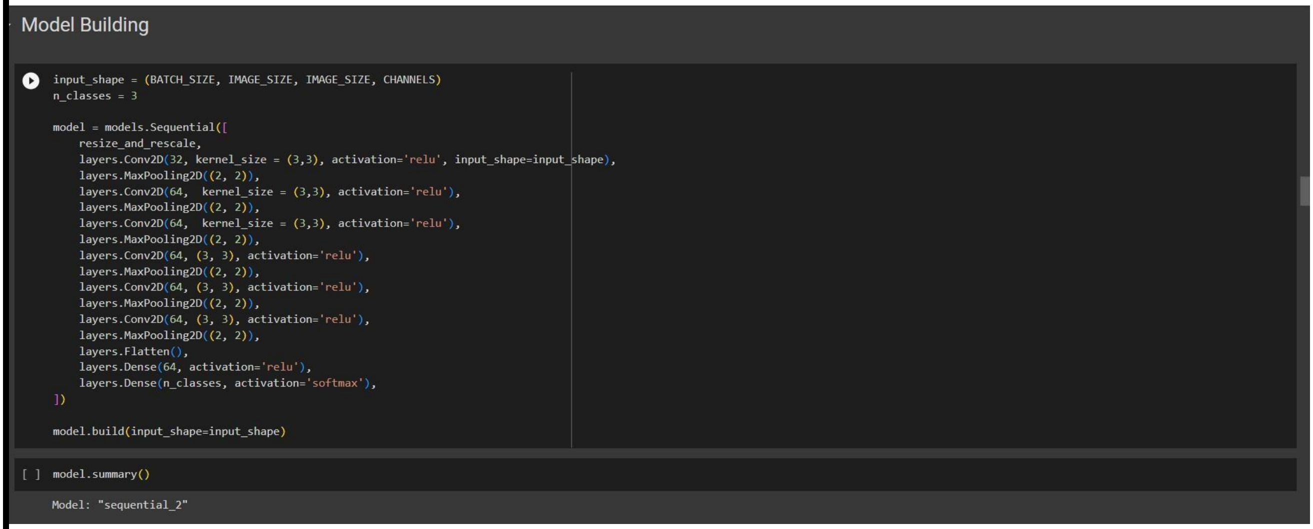
Milestone 4: Model Building

Activity 1: Building a model

Now our data is ready and it’s time to build the model.

Here we are going to build our model layer by layer using .Sequential() from

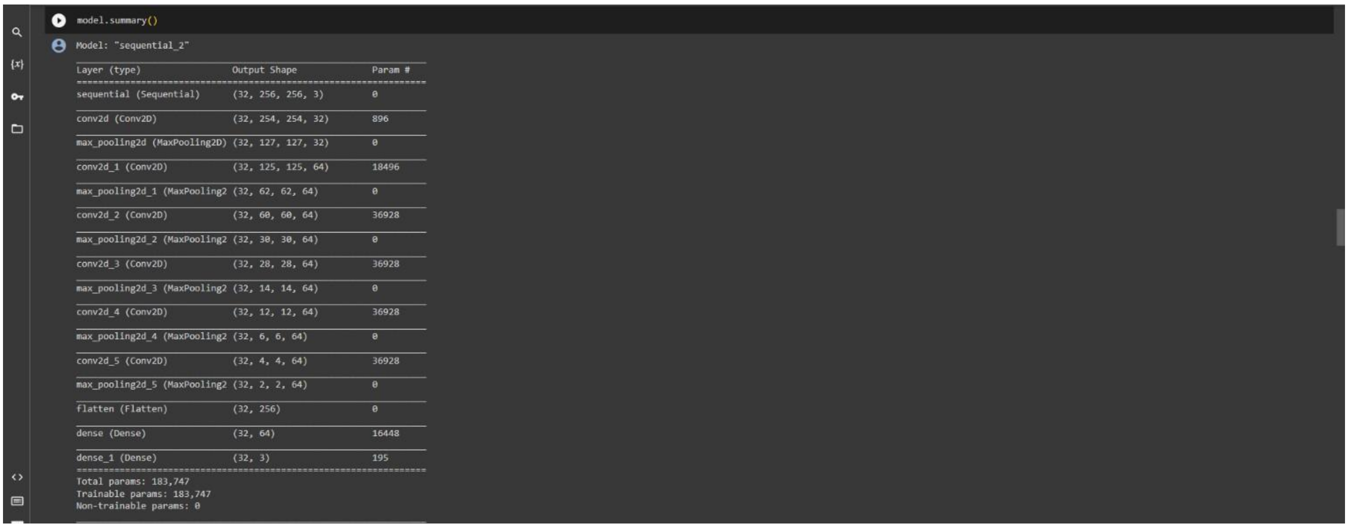
keras. Let’s go !



Model Building

```
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)
```

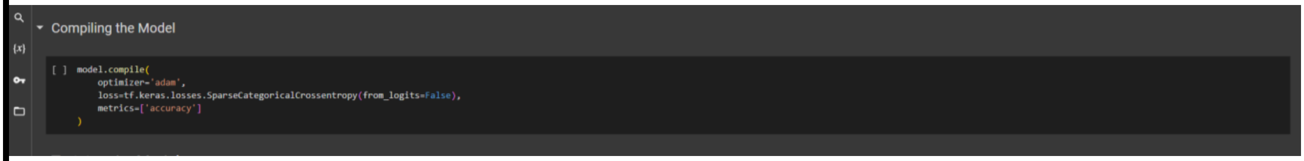
```
[ ] model.summary()

    Model: "sequential_2"
```



```
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 (MaxPooling2 (32, 62, 62, 64)         0
_________________________________________________________________
conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (32, 30, 30, 64)         0
_________________________________________________________________
conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (32, 14, 14, 64)         0
_________________________________________________________________
conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (32, 6, 6, 64)           0
_________________________________________________________________
conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (32, 2, 2, 64)           0
_________________________________________________________________
flatten (Flatten)           (32, 256)                 0
_________________________________________________________________
dense (Dense)               (32, 64)                  16448
_________________________________________________________________
dense_1 (Dense)             (32, 3)                   195
=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

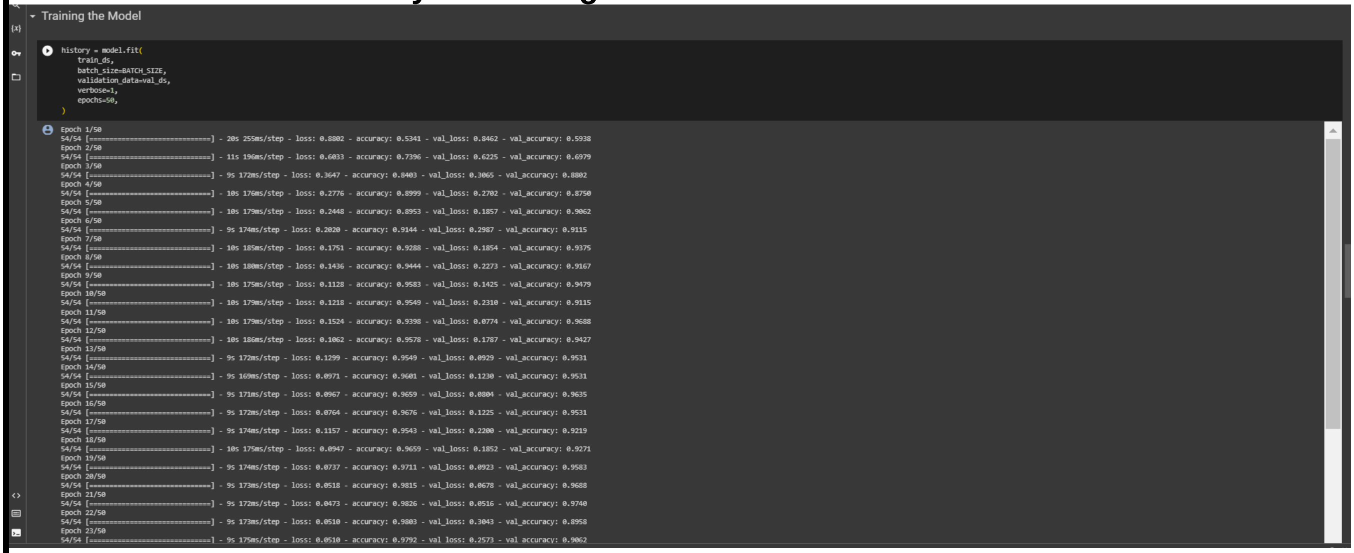
Activity 2: Compiling the model



Compiling the Model

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

Activity 3: Training the model



Training the Model

```
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)
```

```
Epoch 1/50
54/54 [==============================] - 20s 255ms/step - loss: 0.8802 - accuracy: 0.5341 - val_loss: 0.8462 - val_accuracy: 0.5938
Epoch 2/50
54/54 [==============================] - 11s 196ms/step - loss: 0.6033 - accuracy: 0.7396 - val_loss: 0.6225 - val_accuracy: 0.6979
Epoch 3/50
54/54 [==============================] - 9s 172ms/step - loss: 0.3647 - accuracy: 0.8403 - val_loss: 0.3065 - val_accuracy: 0.8802
Epoch 4/50
54/54 [==============================] - 10s 176ms/step - loss: 0.2776 - accuracy: 0.8999 - val_loss: 0.2702 - val_accuracy: 0.8750
Epoch 5/50
54/54 [==============================] - 10s 179ms/step - loss: 0.2448 - accuracy: 0.8953 - val_loss: 0.1857 - val_accuracy: 0.9062
Epoch 6/50
54/54 [==============================] - 9s 174ms/step - loss: 0.2020 - accuracy: 0.9144 - val_loss: 0.2987 - val_accuracy: 0.9115
Epoch 7/50
54/54 [==============================] - 10s 185ms/step - loss: 0.1751 - accuracy: 0.9288 - val_loss: 0.1854 - val_accuracy: 0.9375
Epoch 8/50
54/54 [==============================] - 10s 180ms/step - loss: 0.1436 - accuracy: 0.9444 - val_loss: 0.2273 - val_accuracy: 0.9167
Epoch 9/50
54/54 [==============================] - 10s 175ms/step - loss: 0.1128 - accuracy: 0.9583 - val_loss: 0.1425 - val_accuracy: 0.9479
Epoch 10/50
54/54 [==============================] - 10s 179ms/step - loss: 0.1218 - accuracy: 0.9549 - val_loss: 0.2310 - val_accuracy: 0.9115
Epoch 11/50
54/54 [==============================] - 10s 179ms/step - loss: 0.1524 - accuracy: 0.9398 - val_loss: 0.0774 - val_accuracy: 0.9688
Epoch 12/50
54/54 [==============================] - 10s 186ms/step - loss: 0.1062 - accuracy: 0.9578 - val_loss: 0.1787 - val_accuracy: 0.9427
Epoch 13/50
54/54 [==============================] - 9s 172ms/step - loss: 0.1299 - accuracy: 0.9549 - val_loss: 0.0929 - val_accuracy: 0.9531
Epoch 14/50
54/54 [==============================] - 9s 169ms/step - loss: 0.0971 - accuracy: 0.9601 - val_loss: 0.1230 - val_accuracy: 0.9531
Epoch 15/50
54/54 [==============================] - 9s 171ms/step - loss: 0.0967 - accuracy: 0.9659 - val_loss: 0.0804 - val_accuracy: 0.9635
Epoch 16/50
54/54 [==============================] - 9s 172ms/step - loss: 0.0764 - accuracy: 0.9676 - val_loss: 0.1225 - val_accuracy: 0.9531
Epoch 17/50
54/54 [==============================] - 9s 174ms/step - loss: 0.1157 - accuracy: 0.9543 - val_loss: 0.2200 - val_accuracy: 0.9219
Epoch 18/50
54/54 [==============================] - 10s 175ms/step - loss: 0.0947 - accuracy: 0.9659 - val_loss: 0.1852 - val_accuracy: 0.9271
Epoch 19/50
54/54 [==============================] - 9s 174ms/step - loss: 0.0737 - accuracy: 0.9711 - val_loss: 0.0923 - val_accuracy: 0.9583
Epoch 20/50
54/54 [==============================] - 9s 173ms/step - loss: 0.0518 - accuracy: 0.9815 - val_loss: 0.0678 - val_accuracy: 0.9688
Epoch 21/50
54/54 [==============================] - 9s 172ms/step - loss: 0.0473 - accuracy: 0.9826 - val_loss: 0.0516 - val_accuracy: 0.9740
Epoch 22/50
54/54 [==============================] - 9s 173ms/step - loss: 0.0510 - accuracy: 0.9803 - val_loss: 0.3043 - val_accuracy: 0.8958
Epoch 23/50
54/54 [==============================] - 9s 175ms/step - loss: 0.0510 - accuracy: 0.9792 - val_loss: 0.2573 - val_accuracy: 0.9062
```

The verbose option specifies that you want to display detailed processing information on your screen

If your laptop is functioning slow for 100 epochs, then you can try Google Colab Notebooks.

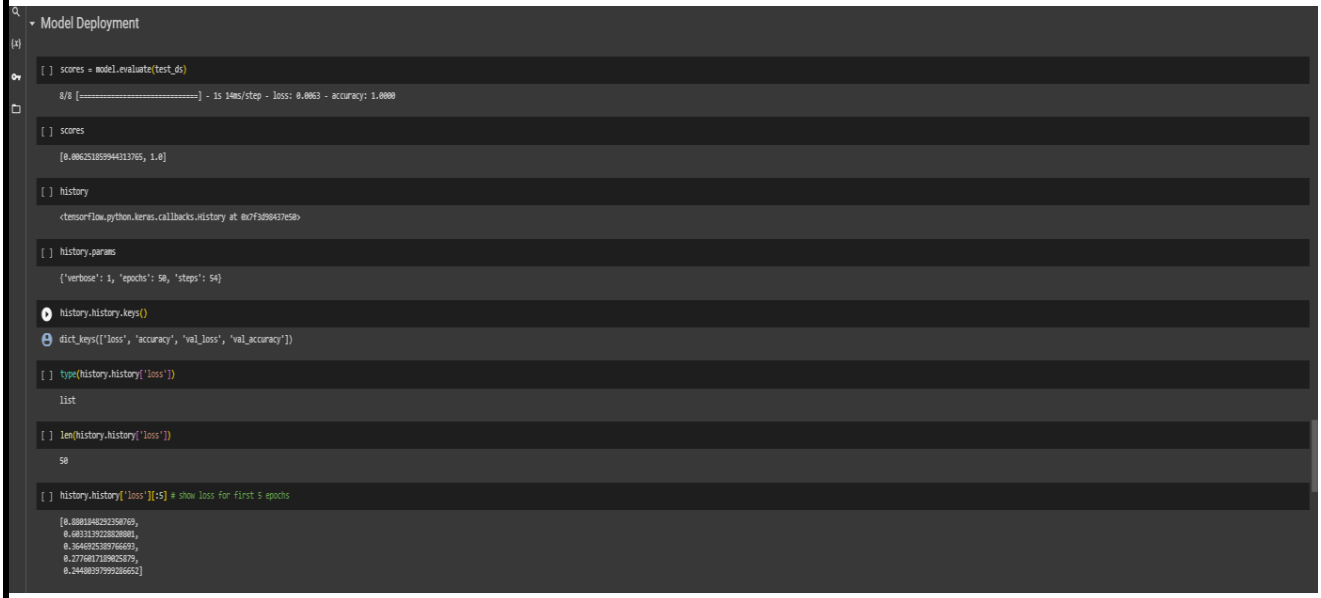
You just have to follow the below steps:

- Go to GoogleColab and create a new notebook.
- Click on Files and mount GoogleDrive.
- Upload your dataset onto GoogleDrive.
- After uploading the dataset go to runtime and click on “Change runtime type”.
- Select “gpu” and choose “standard” and click on save.
- Booyah! Now you can run the whole code.

Milestone 5: Model Deployment

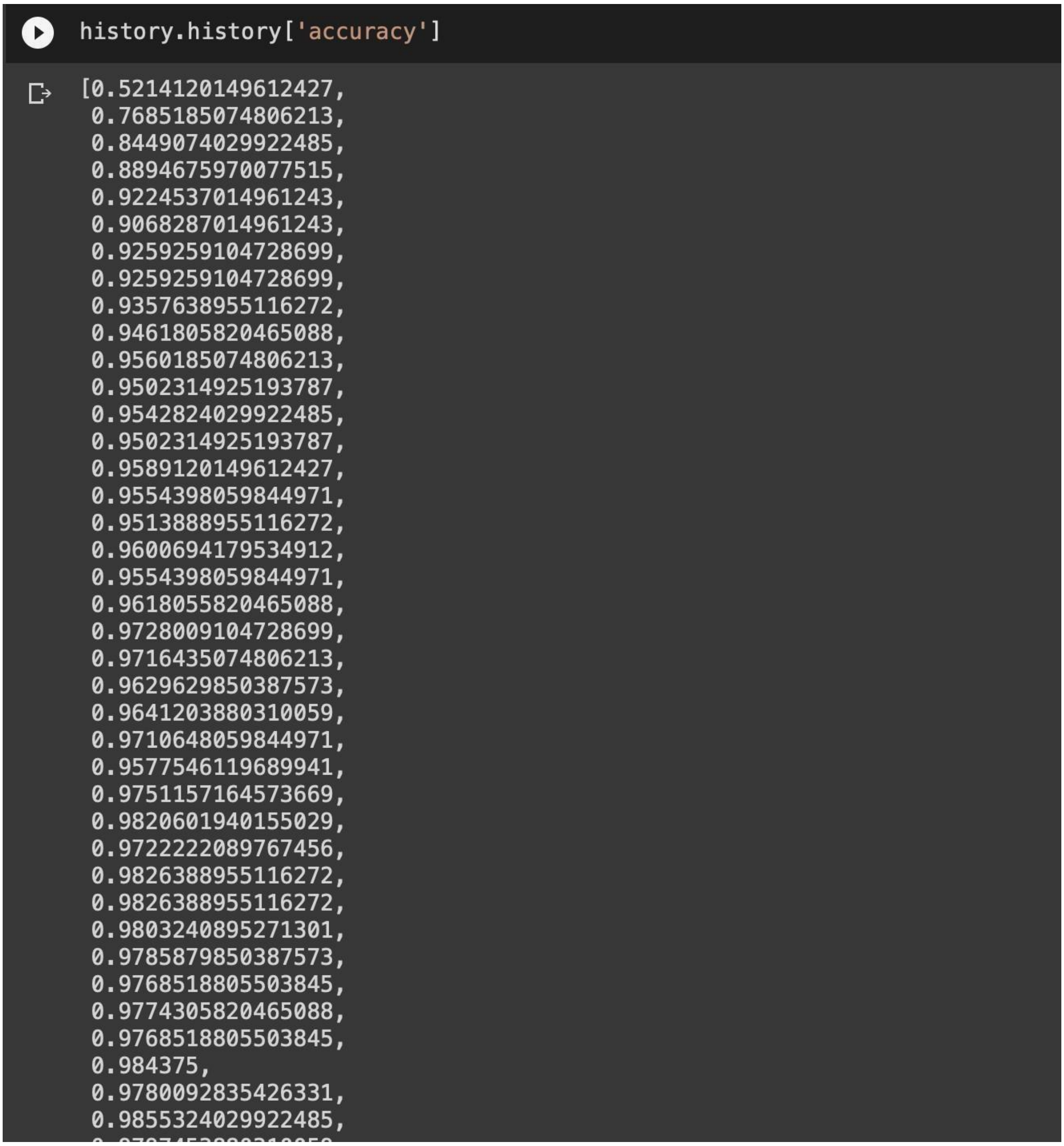
Activity 1: Model Evaluation

As we have the record of accuracy stored in the history variable. We can try visualising the performance of our model.



Model Deployment

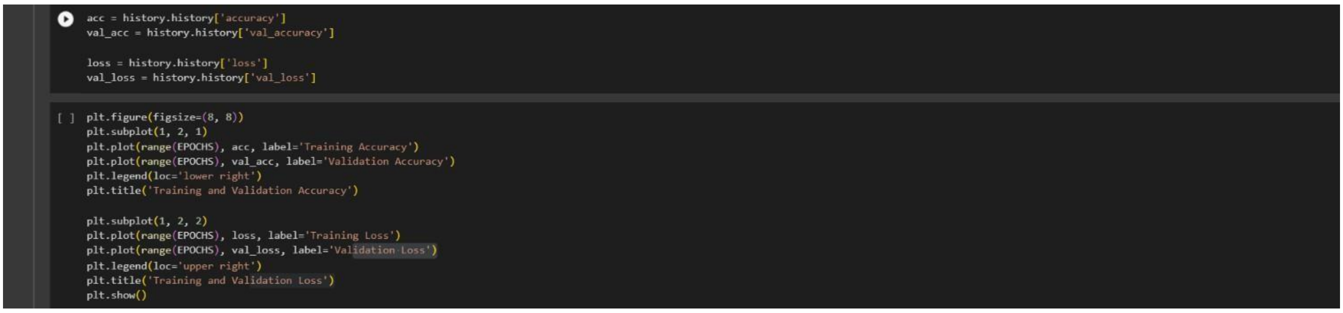
```
scores = model.evaluate(test_ds)
8/8 [==============================] - 1s 14ms/step - loss: 0.0063 - accuracy: 1.0000

scores
[0.006251859944313765, 1.0]

history
<tensorflow.python.keras.callbacks.History at 0x7f3d98437e50>

history.params
{'verbose': 1, 'epochs': 50, 'steps': 54}

history.history.keys()
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

type(history.history['loss'])
list

len(history.history['loss'])
50

history.history['loss'][:5] # show loss for first 5 epochs
[0.8801848292350769,
 0.6033139228820801,
 0.3646925389766693,
 0.2776017189025879,
 0.24480397999286652]
```



```
history.history['accuracy']
```

```
[0.5214120149612427,
 0.7685185074806213,
 0.8449074029922485,
 0.8894675970077515,
 0.9224537014961243,
 0.9068287014961243,
 0.9259259104728699,
 0.9259259104728699,
 0.9357638955116272,
 0.9461805820465088,
 0.9560185074806213,
 0.9502314925193787,
 0.9542824029922485,
 0.9502314925193787,
 0.9589120149612427,
 0.9554398059844971,
 0.9513888955116272,
 0.9600694179534912,
 0.9554398059844971,
 0.9618055820465088,
 0.9728009104728699,
 0.9716435074806213,
 0.9629629850387573,
 0.9641203880310059,
 0.9710648059844971,
 0.9577546119689941,
 0.9751157164573669,
 0.9820601940155029,
 0.9722222089767456,
 0.9826388955116272,
 0.9826388955116272,
 0.9803240895271301,
 0.9785879850387573,
 0.9768518805503845,
 0.9774305820465088,
 0.9768518805503845,
 0.984375,
 0.9780092835426331,
 0.9855324029922485,
```

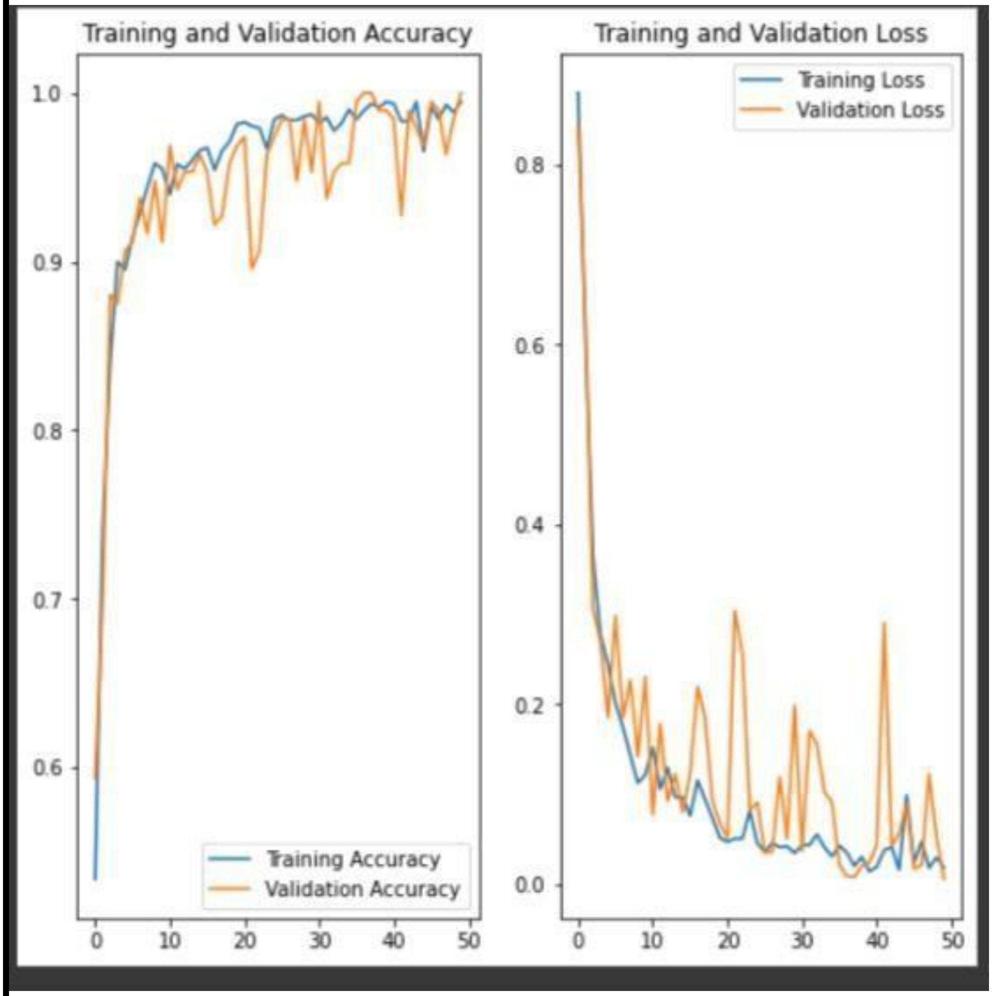
history.history[‘accuracy’] contains the track of accuracies achieved while you were training your model.



```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

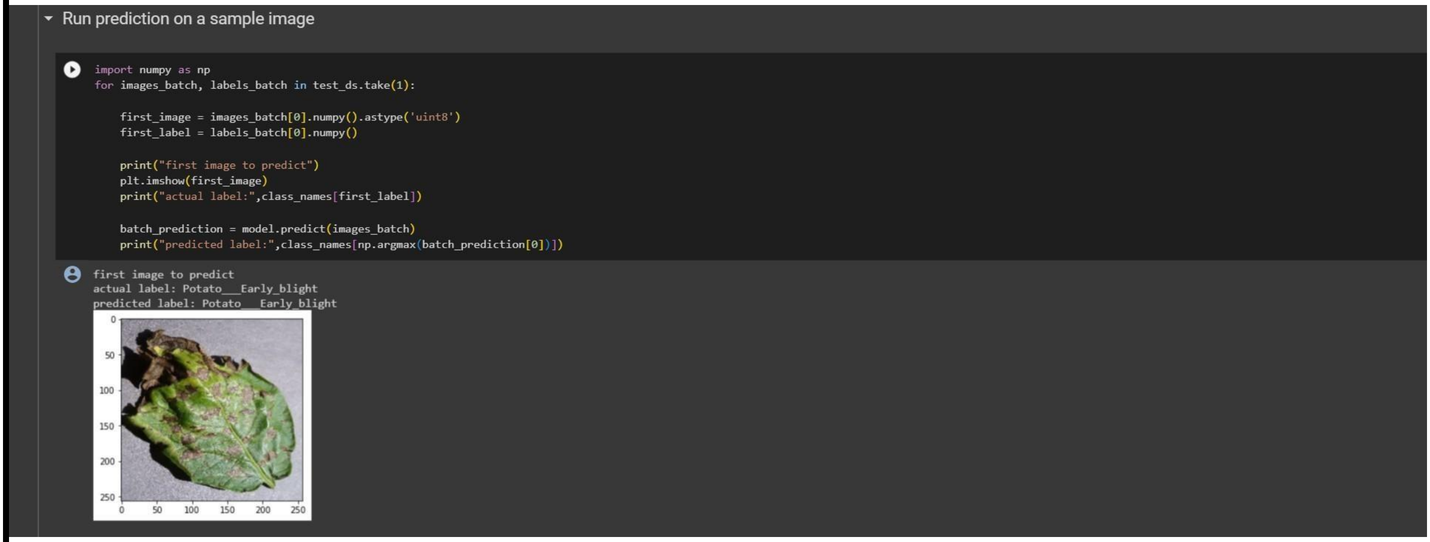
```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label='Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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 and Validation Loss')
plt.show()
```

The above code plots two graphs, one graph b/w Training and Validation Accuracy and the other graphs b/w Training and Validation Loss.



Activity 2: Model Evaluation with Visualisation

First we’ll select one image from the test dataset and try to predict using the trained model. Then we’ll try to print the actual class and predicted class of that particular image.



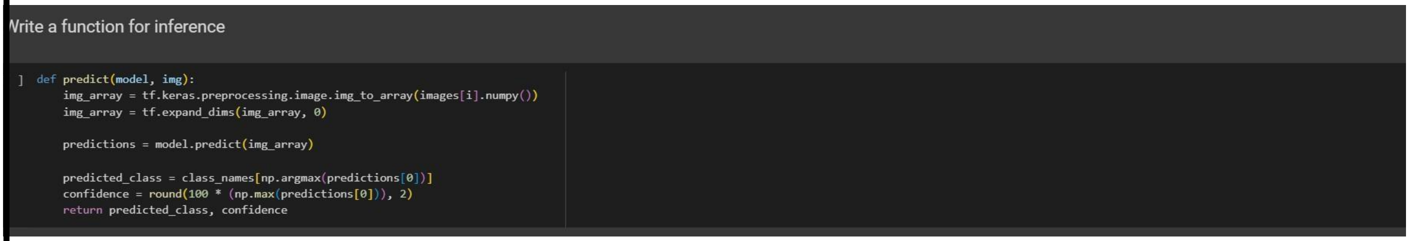
Run prediction on a sample image

```
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
actual label: Potato___Early_blight
predicted label: Potato___Early_blight
```

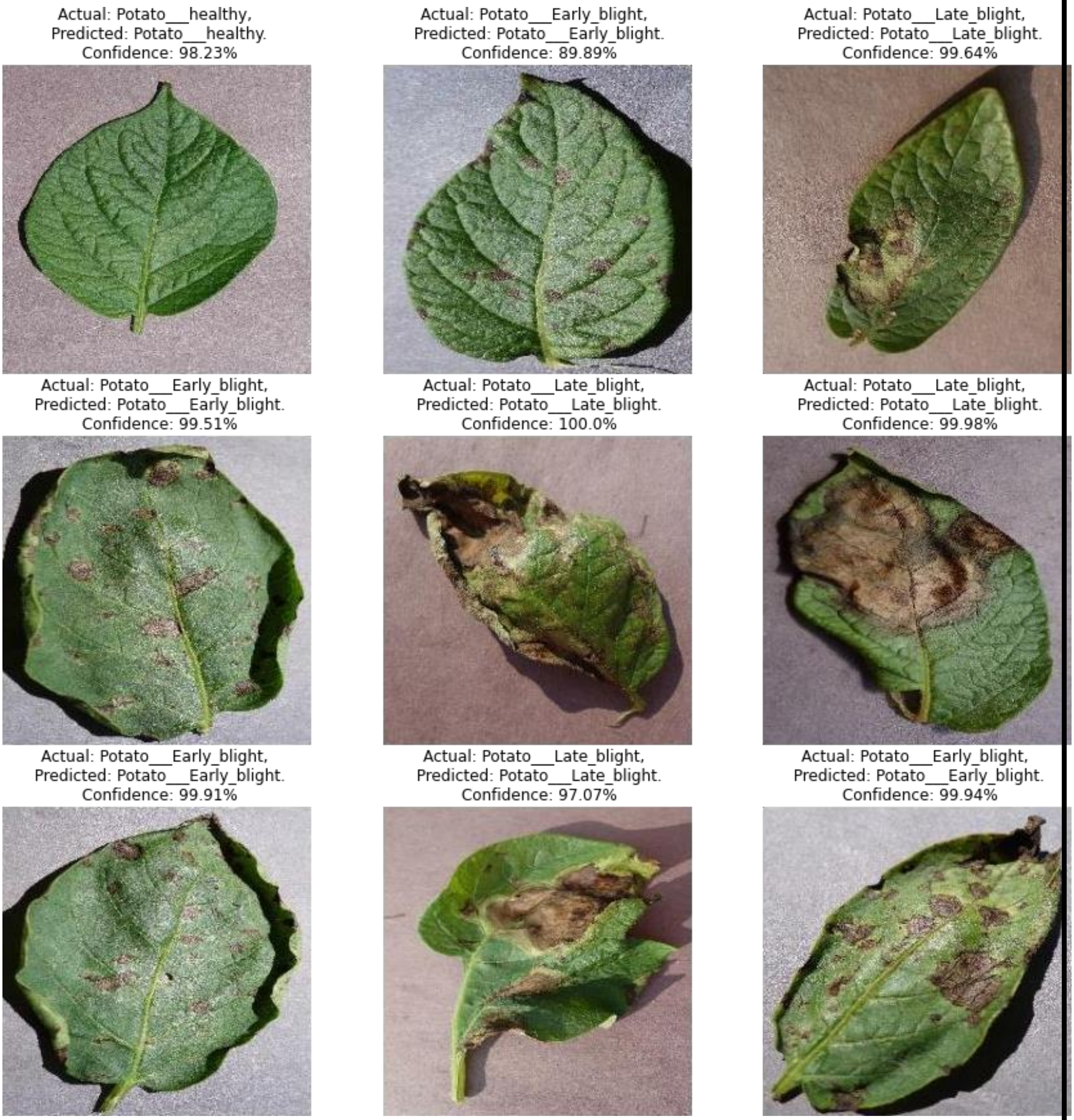
Each time you run this code, you’ll get a different image to predict.

Now let us write a function for prediction, which returns the predicted class and the confidence of its prediction.



/rite a function for inference

```
] def predict(model, img):
      img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
      img_array = tf.expand_dims(img_array, 0)

      predictions = model.predict(img_array)

      predicted_class = class_names[np.argmax(predictions[0])]
      confidence = round(100 * (np.max(predictions[0])), 2)
      return predicted_class, confidence
```



Actual: Potato___healthy,
Predicted: Potato___healthy.
Confidence: 98.23%

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight.
Confidence: 89.89%

Actual: Potato___Late_blight,
Predicted: Potato___Late_blight.
Confidence: 99.64%

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight.
Confidence: 99.51%

Actual: Potato___Late_blight,
Predicted: Potato___Late_blight.
Confidence: 100.0%

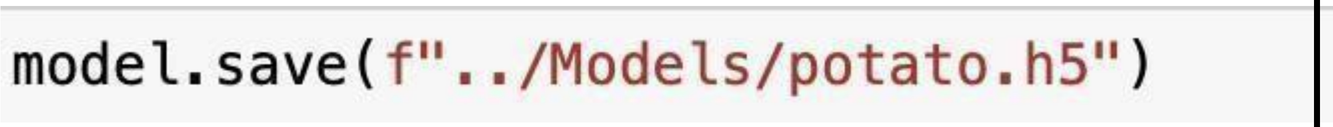
Actual: Potato___Late_blight,
Predicted: Potato___Late_blight.
Confidence: 99.98%

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight.
Confidence: 99.91%

Actual: Potato___Late_blight,
Predicted: Potato___Late_blight.
Confidence: 97.07%

Actual: Potato___Early_blight,
Predicted: Potato___Early_blight.
Confidence: 99.94%

Activity 3: Save the model



```
model.save(f"../Models/potato.h5")
```

This will allow us to save the .h5 format of the model.

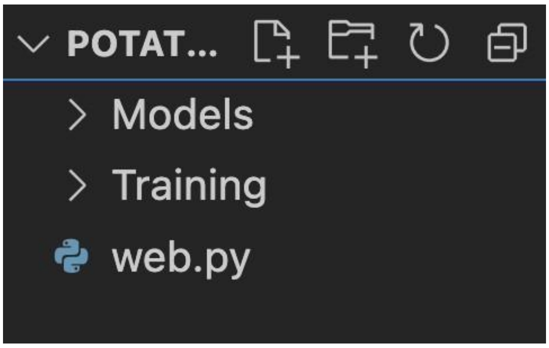
Activity 4: Integrate with Web Framework

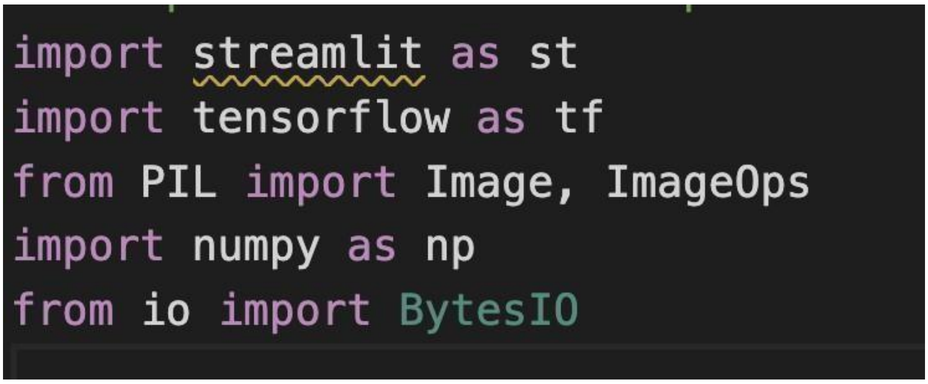
In this section, we will be building a web application that is integrated to the model we built.

We will be using the streamlit package for our website development.

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.

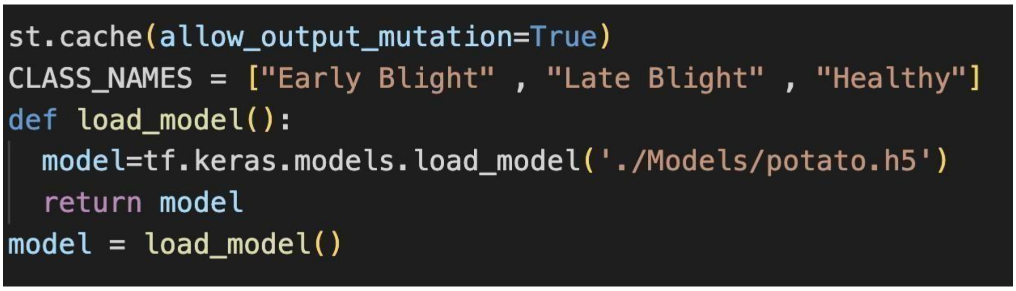
Activity 4.1: Create a web.py file and import necessary packages:





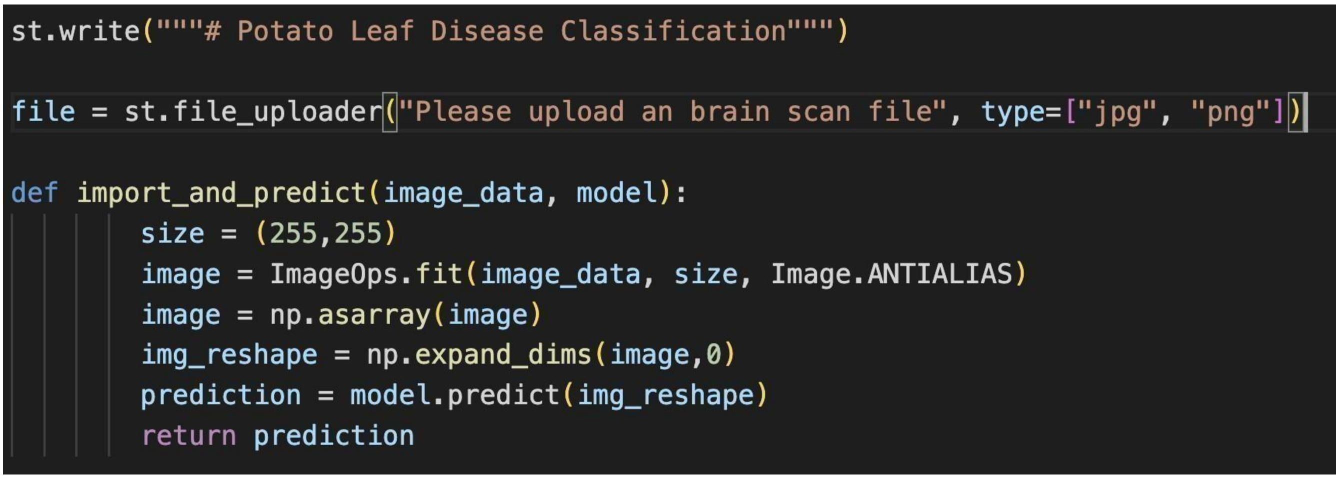
```python
import streamlit as st
import tensorflow as tf
from PIL import Image, ImageOps
import numpy as np
from io import BytesIO
```

Activity 4.2: Defining class names and loading the model:

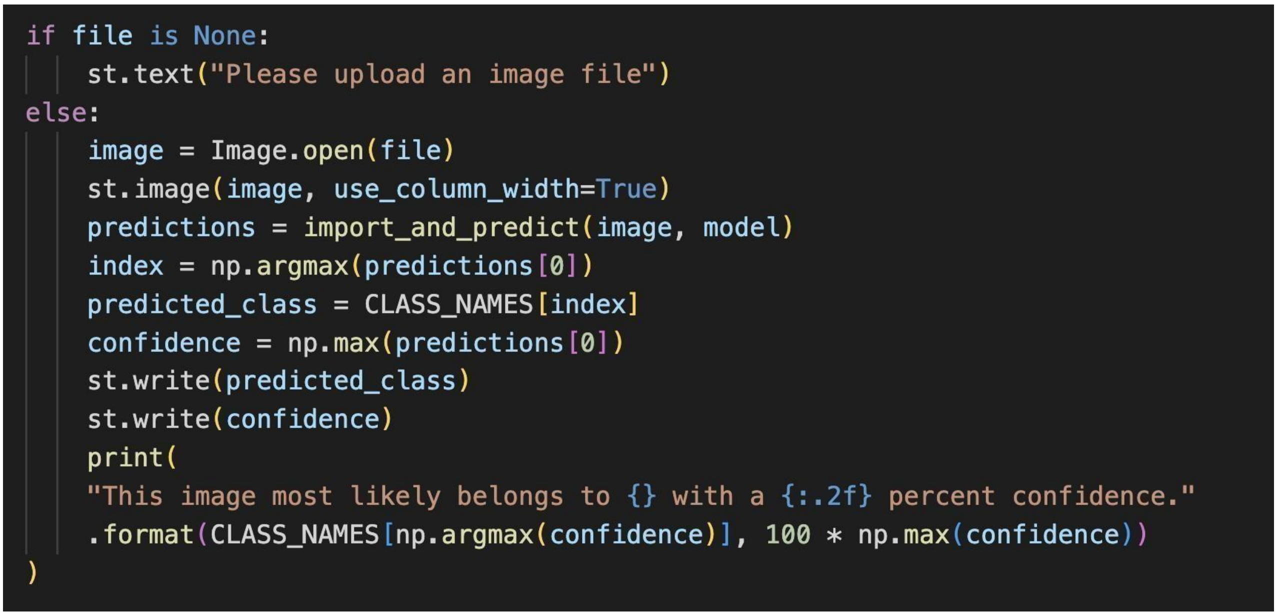


```python
st.cache(allow_output_mutation=True)
CLASS_NAMES = ["Early Blight" , "Late Blight" , "Healthy"]
def load_model():
    model=tf.keras.models.load_model('./Models/potato.h5')
    return model
model = load_model()
```

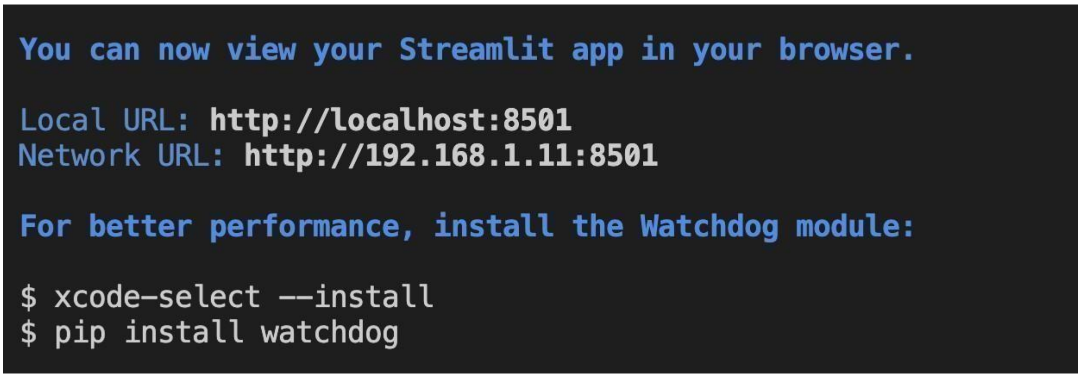
Activity 4.3: Accepting the input from user and prediction



```
st.write("""# Potato Leaf Disease Classification""")

file = st.file_uploader("Please upload an brain scan file", type=["jpg", "png"])

def import_and_predict(image_data, model):
        size = (255,255)
        image = ImageOps.fit(image_data, size, Image.ANTIALIAS)
        image = np.asarray(image)
        img_reshape = np.expand_dims(image,0)
        prediction = model.predict(img_reshape)
        return prediction
```



```
if file is None:
    st.text("Please upload an image file")
else:
    image = Image.open(file)
    st.image(image, use_column_width=True)
    predictions = import_and_predict(image, model)
    index = np.argmax(predictions[0])
    predicted_class = CLASS_NAMES[index]
    confidence = np.max(predictions[0])
    st.write(predicted_class)
    st.write(confidence)
    print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(CLASS_NAMES[np.argmax(confidence)], 100 * np.max(confidence))
)
```

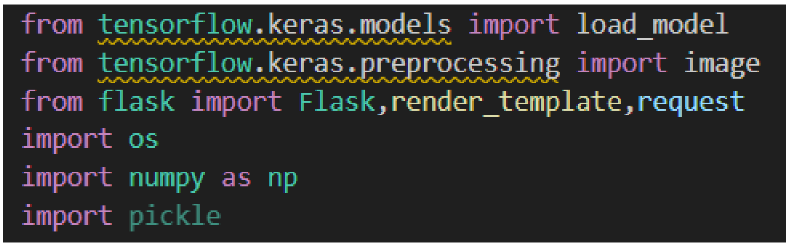


```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.11:8501

For better performance, install the Watchdog module:

$ xcode-select --install
$ pip install watchdog
```

Milestone 5: Building Flask application

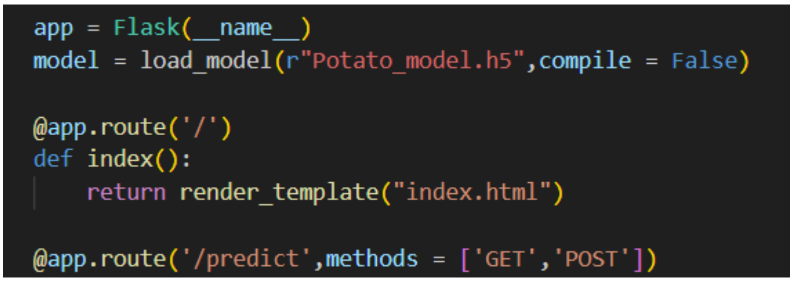
After the model is built, we will be integrating it to a web application so that normal users can also use it. The new users need to initially register in the portal. After registration users can login to browse the images to detect the condition of potatoes. Activity 1: Build a python application

Step 1: Load the required packages.



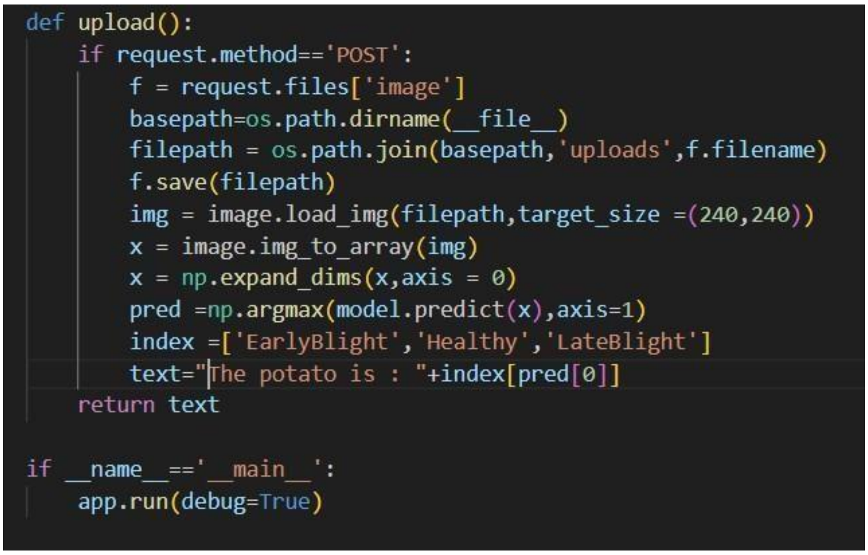
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask,render_template,request
import os
import numpy as np
import pickle
```

Step 2: Initialize the flask app, load the model and configure the html pages Instance of Flask is created and the model is loaded using load_model from keras.



```
app = Flask(__name__)
model = load_model(r"Potato_model.h5",compile = False)

@app.route('/')
def index():
    return render_template("index.html")

@app.route('/predict',methods = ['GET','POST'])
```

Step 3: Pre-process the frame and run



```
def upload():
    if request.method=='POST':
        f = request.files['image']
        basepath=os.path.dirname(__file__)
        filepath = os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)
        img = image.load_img(filepath,target_size =(240,240))
        x = image.img_to_array(img)
        x = np.expand_dims(x,axis = 0)
        pred =np.argmax(model.predict(x),axis=1)
        index =['EarlyBlight','Healthy','LateBlight']
        text="The potato is : "+index[pred[0]]
    return text

if __name__=='__main__':
    app.run(debug=True)
```

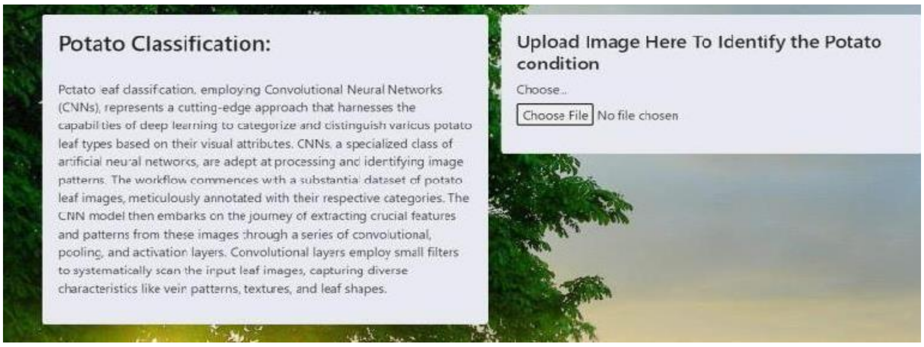
Run the flask application using the run method. By default, the flask runs on port 5000. If the port is to be changed, an argument can be passed and the port can be modified.

Activity 2: Build the HTML page and execute

Build the UI where a home page will have details about the application and prediction page where a user is allowed to browse an image and get the predictions of images. Step 1: Run the application

In the anaconda prompt, navigate to the folder in which the flask app is present. When the python file is executed, the localhost is activated on port 5000 and can be accessed through it.

Step 2: Open the browser and navigate to localhost:5000 to check your application The home page looks like this:



Potato Classification:

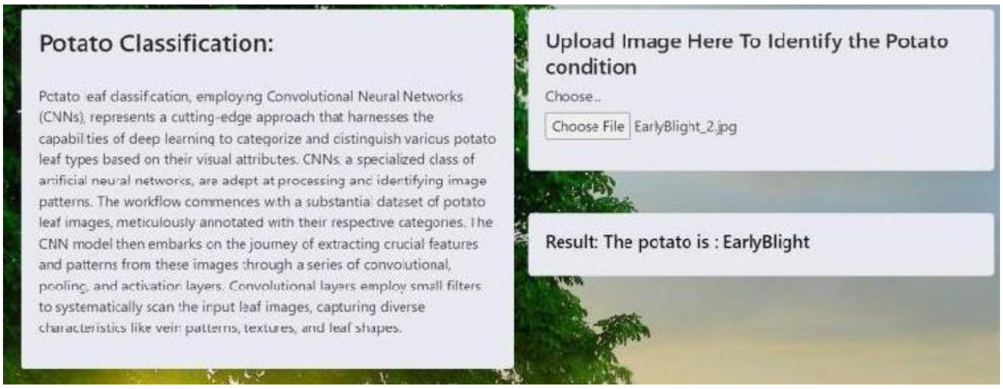
Potato leaf classification, employing Convolutional Neural Networks (CNNs), represents a cutting-edge approach that harnesses the capabilities of deep learning to categorize and distinguish various potato leaf types based on their visual attributes. CNNs, a specialized class of artificial neural networks, are adept at processing and identifying image patterns. The workflow commences with a substantial dataset of potato leaf images, meticulously annotated with their respective categories. The CNN model then embarks on the journey of extracting crucial features and patterns from these images through a series of convolutional, pooling, and activation layers. Convolutional layers employ small filters to systematically scan the input leaf images, capturing diverse characteristics like vein patterns, textures, and leaf shapes.

Upload Image Here To Identify the Potato condition

Choose..

Choose File No file chosen

Click on choose the image to upload and select an image to be classified.<u>Input-1:</u>



Potato Classification:

Potato leaf classification, employing Convolutional Neural Networks (CNNs), represents a cutting-edge approach that harnesses the capabilities of deep learning to categorize and distinguish various potato leaf types based on their visual attributes. CNNs, a specialized class of artificial neural networks, are adept at processing and identifying image patterns. The workflow commences with a substantial dataset of potato leaf images, meticulously annotated with their respective categories. The CNN model then embarks on the journey of extracting crucial features and patterns from these images through a series of convolutional, pooling, and activation layers. Convolutional layers employ small filters to systematically scan the input leaf images, capturing diverse characteristics like vein patterns, textures, and leaf shapes.

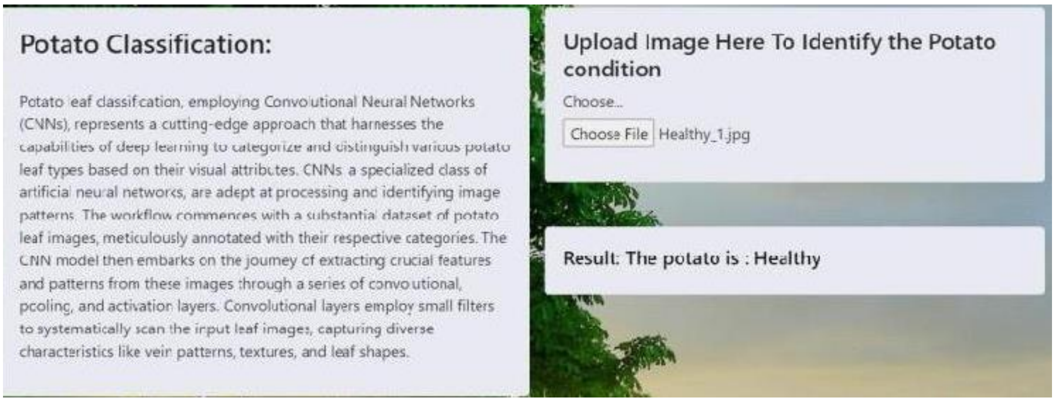
Upload Image Here To Identify the Potato condition

Choose..

Choose File EarlyBlight_2.jpg

Result: The potato is : EarlyBlight

<u>Input-2:</u>



Potato Classification:

Potato leaf classification, employing Convolutional Neural Networks (CNNs), represents a cutting-edge approach that harnesses the capabilities of deep learning to categorize and distinguish various potato leaf types based on their visual attributes. CNNs, a specialized class of artificial neural networks, are adept at processing and identifying image patterns. The workflow commences with a substantial dataset of potato leaf images, meticulously annotated with their respective categories. The CNN model then embarks on the journey of extracting crucial features and patterns from these images through a series of convolutional, pooling, and activation layers. Convolutional layers employ small filters to systematically scan the input leaf images, capturing diverse characteristics like vein patterns, textures, and leaf shapes.

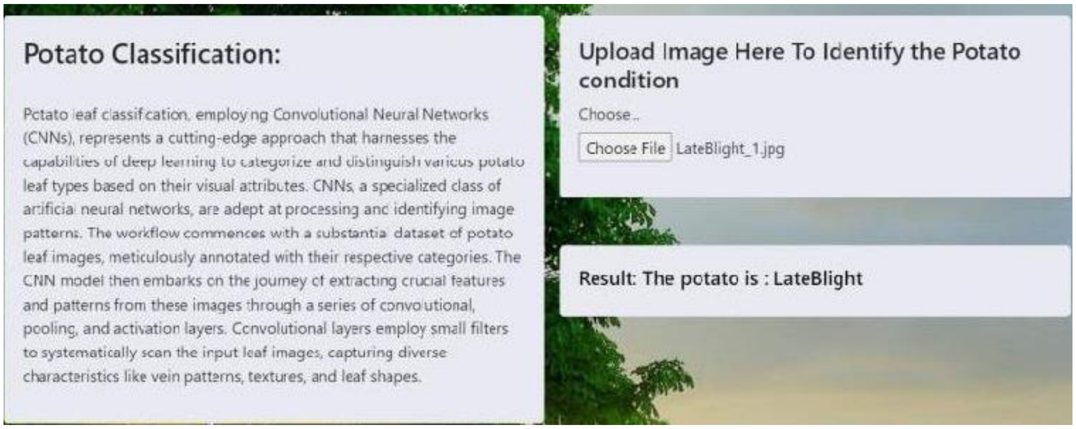
Upload Image Here To Identify the Potato condition

Choose..

Choose File Healthy_1.jpg

Result: The potato is : Healthy

Input-3:



Potato Classification:

Potato leaf classification, employing Convolutional Neural Networks (CNNs), represents a cutting-edge approach that harnesses the capabilities of deep learning to categorize and distinguish various potato leaf types based on their visual attributes. CNNs, a specialized class of artificial neural networks, are adept at processing and identifying image patterns. The workflow commences with a substantial dataset of potato leaf images, meticulously annotated with their respective categories. The CNN model then embarks on the journey of extracting crucial features and patterns from these images through a series of convolutional, pooling, and activation layers. Convolutional layers employ small filters to systematically scan the input leaf images, capturing diverse characteristics like vein patterns, textures, and leaf shapes.

Upload Image Here To Identify the Potato condition

Choose..

Choose File LateBlight_1.jpg

Result: The potato is : LateBlight

8. ADVANTAGES & DISADVANTAGES

Advantages:

Early Disease Detection: The technology makes it easier to identify potato illnesses early, enabling farmers to respond quickly to minimise crop damage. Higher crop yields and fewer financial losses may result from this.

Accurate Classification: When taught and maintained appropriately, deep learning models can yield remarkably accurate illness classification outcomes. Making educated judgements about illness prevention and treatment depends on this accuracy.

User-Friendly: The system's user interface and feedback mechanisms enable a broad spectrum of users, even those with low technical competence, to utilise it. It gives farmers the ability to control their crops with cutting-edge technologies.

Continuous Improvement: The accuracy and robustness of the model can be continuously enhanced by utilising user feedback. As more data and feedback are collected, the system becomes increasingly reliable and effective.

Scalability: The system can be scaled to accommodate a growing number of users and adapt to the specific needs of different agricultural regions and crops.

Disadvantages:

Data Requirements: Obtaining big and diverse datasets for training is a challenge for deep learning models, particularly for rare or regionally unique diseases. Such datasets can be expensive and time-consuming to obtain and annotate.

Model Complexity: Deep learning models, particularly CNNs, can be computationally intensive and may require high-performance hardware for training and inference. This could be a barrier for small-scale farmers with limited resources.

Interpretability: Even while deep learning models are capable of great accuracy, they are frequently regarded as "black boxes," which makes it challenging to provide an explanation for a particular categorization conclusion. The difficulty of interpreting a model might affect user acceptability and trust**.**

Maintenance and Updates: Continuous model maintenance and updates are required to keep the system effective. This includes retraining the model with new data and addressing evolving disease patterns.

Privacy and Security: Handling user-generated data, including images, requires robust security measures to protect user privacy. Information misuse or data breaches may have moral and legal ramifications.

Dependency on Technology: The technology infrastructure—which includes internet connection and suitable devices—is necessary for the system to function. The effectiveness of the system can be compromised in areas with inadequate connectivity or restricted device access.

False Positives and Negatives: Like all machine learning models, false positives and false negatives can occur. Misclassifications can lead to unnecessary pesticide use or failure to address actual diseases, potentially causing harm or economic losses.

9. CONCLUSION

An encouraging approach for agriculture is the creation of a deep learning system, specifically Convolutional Neural Networks (CNNs), for the categorization of potato diseases. The prompt and precise identification of illnesses in potato plants by this technology allows for early detection, which can result in higher crop yields, lower financial losses, and improved food security. A wide range of users, including non-technical people, can utilise this technology because of its user-friendly interface and feedback systems. To guarantee the system's efficacy, however, issues with data quality, model complexity, interpretability, and continuing maintenance must be properly handled. As the project moves forward, accuracy and relevance must be maintained by regular model changes and the incorporation of user feedback. To sum up, this project has the potential to greatly help farmers and the agricultural sector as a whole when implemented responsibly and thoughtfully.

10. FUTURE SCOPE

Expanded Crop Coverage: The underlying deep learning architecture can be extended to classify diseases in numerous other crops, while the current focus is on potato diseases. Numerous agricultural techniques can profit from its scalability, which also helps with crop health management.

AI-Driven Precision Agriculture: By integrating this system with more comprehensive precision agriculture techniques, resource allocation may be optimised, pesticide use can be decreased, and sustainable farming can be encouraged. The deep learning model is able to adjust to particular crop varieties and local conditions.

IoT Integration: The capabilities of the system can be improved by integration with Internet of Things (IoT) devices, such as automated irrigation systems and sensors. By gathering data in real time, these gadgets can enhance illness detection and prevention even further.

AI-Enhanced Recommendations: In addition to categorising diseases, the system has the capacity to develop recommendations for disease management tactics, such as the choice of suitable pesticides, organic treatments, and preventive measures.

Mobile Apps and Accessibility: The creation of specialised mobile applications can improve accessibility and make it possible for farmers in isolated locations to use the technology for managing and diagnosing diseases.

11. APPENDIX

The drive link for the ipynb file with the potato classification model using Resnet_50

https://drive.google.com/file/d/16ewXg9IGlHdGm2UILM35V22_WisV6nnk/view?usp=sharing

The github link for the ipynb file

https://github.com/smartinternz02/SI-GuidedProject-600930-1697632720/blob/main/DEVELOPMENT%20PHASE/potato_disease_classification_model.ipynb

The github link for the flask source code
https://github.com/smartinternz02/SI-GuidedProject-600930-1697632720/blob/main/DEVELOPMENT%20PHASE/Flask/app.py