B.VASUPRIYA PATNAIK (322103382001)

B.SHERLIE ANGEL (322103382002)

N·E·A·T
प्रौद्योगिकी के लिए राष्ट्रीय शैक्षणिक सहयोग
National Educational Alliance for Technology

AICTE
अखिल भारतीय तकनीकी शिक्षा परिषद्
All India Council for Technical Education

EduSkills®
Nation Building Through Skills

AICTE - EduSkills 2024 VIRTUAL INTERNSHIP

Certificate of Virtual Internship

This is to certify that

Sherlie Angel Ballikuri

Gayatri Vidya Parishad College of Engineering (Autonomous)

has successfully completed 10 weeks

AI-ML Virtual Internship

During April - June 2024

Supported By : **India Edu Program**

Google for Developers

Karthik Padmanabhan	Shri Buddha Chandrasekhar	Dr. Satya Ranjan Biswal
Developer Ecosystem Lead	Chief Coordinating Officer (CCO)	Chief Technology Officer (CTO)
MENA & India, Google	NEAT Cell, AICTE	EduSkills

Certificate ID :62837a1143d94cf1d38f61300c684f71
Student ID :STU65b78688160671706526344

INTERNSHIP GRADE D

GRADE- O (Outstanding):90-100 | E (Excellent):80-89 | A (Very Good):70-79 | B (Good): 60-69 | C (Fair): 50-59 | D (Average): 40-49 | P (Pass): 30-39 | F (Fail): Below 30

K.GOWTHAMI (322103382030)


N·E·A·T
प्रौद्योगिकी के लिए राष्ट्रीय शैक्षणिक सहयोग
National Educational Alliance for Technology

AICTE
अखिल भारतीय तकनीकी शिक्षा परिषद्
All India Council for Technical Education

EduSkills®
Nation Building Through Skills

AICTE - EduSkills 2024 VIRTUAL INTERNSHIP

Certificate of Virtual Internship

This is to certify that

Gowthami Kota

Gayatri Vidya Parishad College of Engineering (Autonomous)

has successfully completed 10 weeks

AI-ML Virtual Internship

During April - June 2024

Supported By : **India Edu Program**

Google for Developers

Karthik Padmanabhan Developer Ecosystem Lead MENA & India, Google	**Shri Buddha Chandrasekhar** Chief Coordinating Officer (CCO) NEAT Cell, AICTE	**Dr. Satya Ranjan Biswal** Chief Technology Officer (CTO) EduSkills

Certificate ID :6f799c953f88b9cb4d7f15132acc2914
Student ID :STU65b63f5dcb7b61706442589

INTERNSHIP GRADE B

GRADE- O (Outstanding):90-100 | E (Excellent):80-89 | A (Very Good):70-79 | B (Good): 60-69 | C (Fair): 50-59 | D (Average): 40-49 | P (Pass): 30-39 | F (Fail): Below 30

S.KIRTHI NAGA SARVANI (322103382053)


N·E·A·T
प्रौद्योगिकी के लिए राष्ट्रीय शैक्षणिक सहयोग
National Educational Alliance for Technology

अखिल भारतीय तकनीकी शिक्षा परिषद्
All India Council for Technical Education

EduSkills®
Nation Building Through Skills

AICTE · EduSkills 2024 VIRTUAL INTERNSHIP

Certificate of Virtual Internship

This is to certify that

Kirthi Naga Sarvani Sayimpu

Gayatri Vidya Parishad College of Engineering (Autonomous)

has successfully completed 10 weeks

AI-ML Virtual Internship

During April - June 2024

Supported By : **India Edu Program**

Google for Developers

Karthik Padmanabhan
Developer Ecosystem Lead
MENA & India, Google

Shri Buddha Chandrasekhar
Chief Coordinating Officer (CCO)
NEAT Cell, AICTE

Dr. Satya Ranjan Biswal
Chief Technology Officer (CTO)
EduSkills

Certificate ID :eda3c0634b05784c10f1d399c55643b8
Student ID :STU65b7467b219c11706509947

INTERNSHIP GRADE B

GRADE- O (Outstanding):90-100 | E (Excellent):80-89 | A (Very Good):70-79 | B (Good): 60-69 | C (Fair): 50-59 | D (Average): 40-49 | P (Pass): 30-39 | F (Fail): Below 30

CASE STUDY

Plant Disease Detection

1. Problem Statement

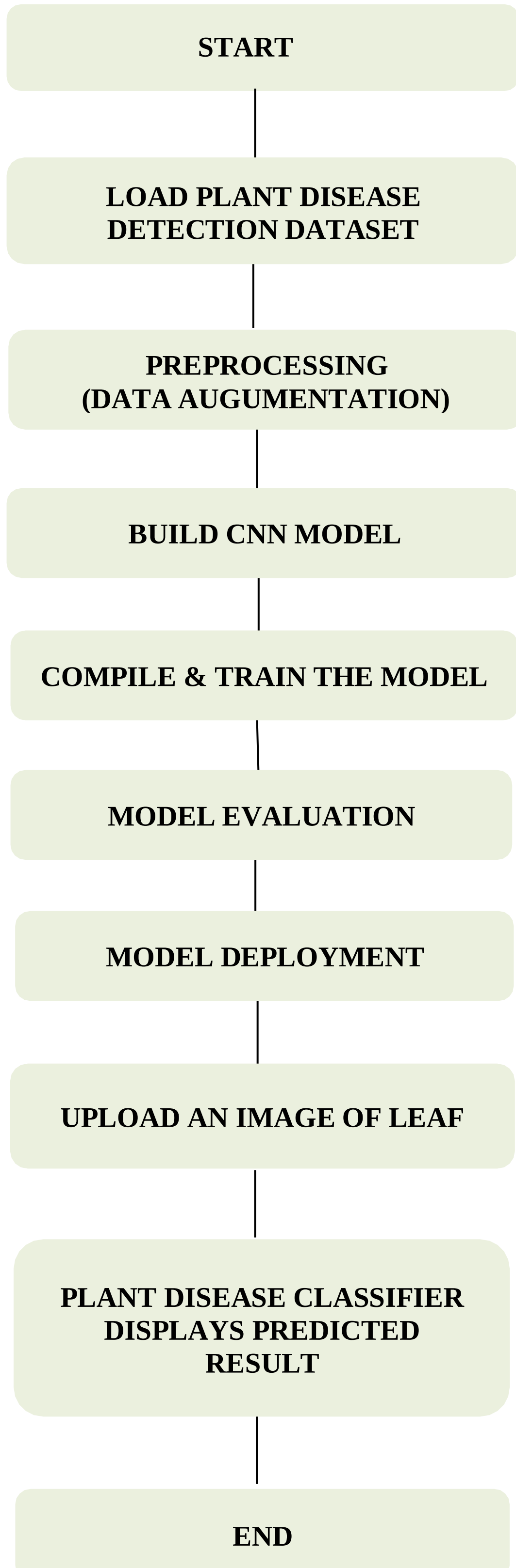
The objective of this project is to develop a robust and efficient system for the classification of plant diseases using image processing and machine learning techniques. The system aims to identify and categorize different types of plant diseases from images of affected plant parts, primarily leaves. By leveraging advanced algorithms and a comprehensive dataset, the system should provide accurate and timely diagnoses to assist farmers and agricultural professionals in making informed decisions for disease management and crop protection.

2. Solution Overview

Agriculture is a cornerstone of the global economy, playing a critical role in supplying essential resources including food, raw materials, and employment opportunities. This sector supports millions of livelihoods and contributes significantly to national and international markets. However, the impact of plant diseases represents a formidable challenge to agricultural productivity. Plant diseases can severely diminish crop yields, degrade the quality of produce, and ultimately lead to significant economic losses for farmers and agricultural enterprises. The consequences of these diseases extend beyond immediate financial impacts, potentially affecting food availability and leading to broader food security concerns.

Early and accurate detection of plant diseases is therefore paramount for mitigating these adverse effects and promoting sustainable agricultural practices. Timely identification of disease outbreaks enables farmers to implement targeted interventions, such as appropriate treatments or preventative measures, which can prevent the spread of diseases and preserve crop health. This proactive approach not only helps to safeguard the quality and quantity of agricultural outputs but also supports long-term food security by reducing the risk of crop failures and ensuring the stability of food supplies. Advancements in diagnostic technologies and disease monitoring are critical in enhancing our ability to address these challenges effectively and maintain a resilient agricultural sector.

FLOWCHART



Flow chart of our model

1. Data Collection

Gathering Data

Download the dataset from Kaggle with three classes: Healthy, Rust, and Powdery.

Extract and organize the images into separate folders for each class (Healthy, Rust, Powdery).

Labeling Data

Verify that each image is correctly labeled according to its class.

If the dataset is not pre-labeled, manually label the images using a tool like Label Img or through scripts.

2. Data Preprocessing

Image Augmentation

Implement data augmentation techniques such as rotation, flipping, zooming, and shifting to increase dataset variability. Use libraries like *Keras' ImageDataGenerator* for augmentation.

Resizing and Normalization

Resize all images to a uniform size (e.g., 128x128 or 256x256 pixels) for consistency. Normalize pixel values to a range of [0, 1] to improve model training.

3. Building the Model

CNN Architecture

Design a Convolutional Neural Network (CNN) using Keras and TensorFlow.

Typical architecture

- Input Layer
- Several Convolutional Layers (e.g., Conv2D)
- Activation Functions (e.g., ReLU)
- Pooling Layers (e.g., MaxPooling2D)
- Fully Connected Layers (Dense)
- Output Layer with softmax activation for multi-class classification.

Compilation

Compile the model using an optimizer like Adam.

Use categorical crossentropy as the loss function for image multi-class classification.

Define metrics such as accuracy to evaluate model performance.

4. Training the Model

Loading Data:

Use ImageDataGenerator to load and preprocess images on-the-fly.

Implement real-time data augmentation during training.

Model Training:

Split the dataset into training and validation sets.

Train the model using the training set while validating its performance on the validation set.

Monitor metrics and adjust hyperparameters to prevent overfitting.

5. Evaluating the Model

Validation

Assess the model’s accuracy, loss, and other performance metrics on the validation set.

Ensure that the model generalizes well to new data and is not overfitting.

6. Deployment

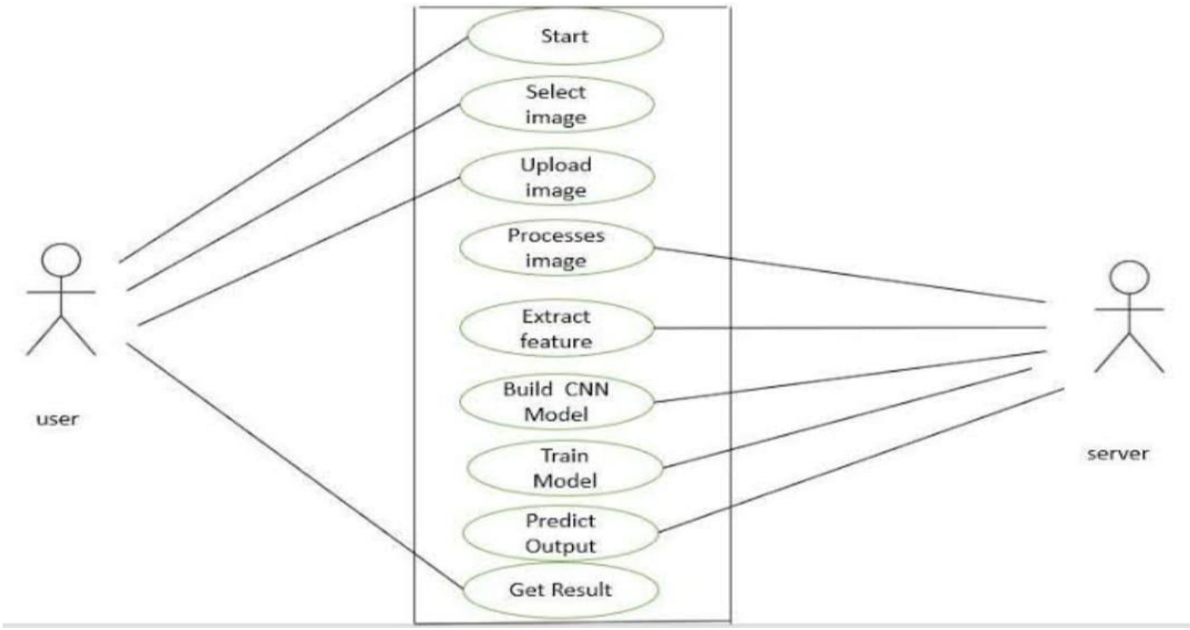
Saving the Model:

Save the trained model using model.save('model.h5') to a file for future use.

Flask API:

- ✓ Set up a Flask web application.
- ✓ Create an endpoint for users to upload plant images.
- ✓ Load the saved model within the Flask app.
- ✓ Implement preprocessing steps (resizing, normalization) for the uploaded image.
- ✓ Use the model to predict the plant’s health or disease status and return the results to the user.

Use-case diagram



Use-case diagram of our model

7. Source Code

```
#Setting Up Kaggle API
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
#Downloading Dataset
!kaggle datasets download -d rashikrahmanpritom/plant-disease-recognition-dataset
```

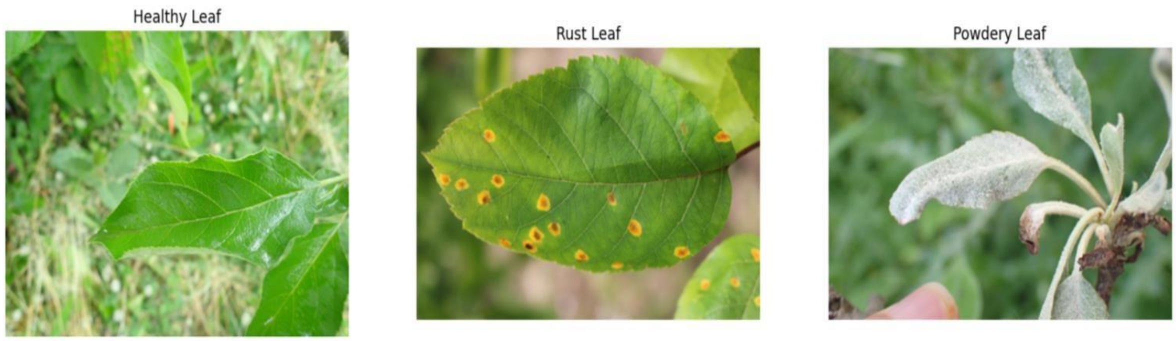
```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Dataset URL: https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset
License(s): CC0-1.0
Downloading plant-disease-recognition-dataset.zip to /content
 99% 1.24G/1.25G [00:15<00:00, 125MB/s]
100% 1.25G/1.25G [00:15<00:00, 85.4MB/s]
```

```
#Extracting the Dataset
import zipfile
zip_ref = zipfile.ZipFile('/content/plant-disease-recognition-dataset.zip', 'r')
zip_ref.extractall('/content/plant_leaf_disease_predictor')
zip_ref.close()
```

```python
#Count of images in each subdirectory
import os

def count_files_and_folders(directory):
    num_files = 0
    num_folders = 0

    for entry in os.listdir(directory):
        entry_path = os.path.join(directory, entry)
        if os.path.isfile(entry_path):
            num_files += 1
        elif os.path.isdir(entry_path):
            num_folders += 1

    return num_files, num_folders

ch = 1
while(True):
  if ch < 7 :
    directory = input('Directory name :')
    files_count, folders_count = count_files_and_folders(directory)
    print(f'Number of files {directory}: {files_count}')
    ch = ch + 1
  # print(f'Number of folders: {folders_count}')
  else :
    break
```

```
Directory name :/content/plant_leaf_disease_predictor/Train/Train/Healthy
Number of files /content/plant_leaf_disease_predictor/Train/Train/Healthy: 458
Directory name :/content/plant_leaf_disease_predictor/Train/Train/Powdery
Number of files /content/plant_leaf_disease_predictor/Train/Train/Powdery: 430
Directory name :/content/plant_leaf_disease_predictor/Train/Train/Rust
Number of files /content/plant_leaf_disease_predictor/Train/Train/Rust: 434
Directory name :/content/plant_leaf_disease_predictor/Test/Test/Healthy
Number of files /content/plant_leaf_disease_predictor/Test/Test/Healthy: 50
Directory name :/content/plant_leaf_disease_predictor/Test/Test/Powdery
Number of files /content/plant_leaf_disease_predictor/Test/Test/Powdery: 50
Directory name :/content/plant_leaf_disease_predictor/Test/Test/Rust
Number of files /content/plant_leaf_disease_predictor/Test/Test/Rust: 50
```

```python
# Displaying Sample Images
import matplotlib.pyplot as plt
from PIL import Image

# Define image paths
image_path_healthy = '/content/plant_leaf_disease_predictor/Train/Train/Healthy/800edef467d27c15.jpg'
image_path_rust = '/content/plant_leaf_disease_predictor/Train/Train/Rust/80f09587dfc7988e.jpg'
image_path_powdery = '/content/plant_leaf_disease_predictor/Train/Train/Powdery/8299723bc94df5a8.jpg'

# Open the images
img_healthy = Image.open(image_path_healthy)
img_rust = Image.open(image_path_rust)
img_powdery = Image.open(image_path_powdery)

# Create a figure with 3 subplots (side by side)
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# Display the healthy leaf image
axs[0].imshow(img_healthy)
axs[0].axis('off')  # Hide axes
axs[0].set_title('Healthy Leaf')

# Display the rust leaf image
axs[1].imshow(img_rust)
axs[1].axis('off')  # Hide axes
axs[1].set_title('Rust Leaf')

# Display the powdery leaf image
axs[2].imshow(img_powdery)
axs[2].axis('off')  # Hide axes
axs[2].set_title('Powdery Leaf')

# Show the plot
plt.show()
```



Sample images from dataset

```
[ ] # Data Augmentation and Preparation
    import tensorflow
    from tensorflow.keras.preprocessing.image import ImageDataGenerator

    train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
    test_datagen = ImageDataGenerator(rescale=1./255)

    train_generator = train_datagen.flow_from_directory('/content/plant_leaf_disease_predictor/Train/Train',
                                                        target_size=(225, 225),
                                                        batch_size=32,
                                                        class_mode='categorical')

    validation_generator = test_datagen.flow_from_directory('/content/plant_leaf_disease_predictor/Test/Test',
                                                            target_size=(225, 225),
                                                            batch_size=32,
                                                            class_mode='categorical')
```

```
Found 1322 images belonging to 3 classes.
Found 150 images belonging to 3 classes.
```

```
#Install or upgrade TensorFlow only
!pip install tensorflow --upgrade

import tensorflow as tf
print("TensorFlow version:", tf.__version__)

# Keras is now part of TensorFlow
keras_version = tf.keras.__version__
print("Keras version:", keras_version)
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.31.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
```

```
[ ] from keras.models import Sequential
    from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense


    model = Sequential()
    model.add(Conv2D(32, (3, 3), input_shape=(225, 225, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(3, activation='softmax'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
[ ] model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 223, 223, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
flatten (Flatten)	(None, 186624)	0
dense (Dense)	(None, 64)	11,944,000
dense_1 (Dense)	(None, 3)	195

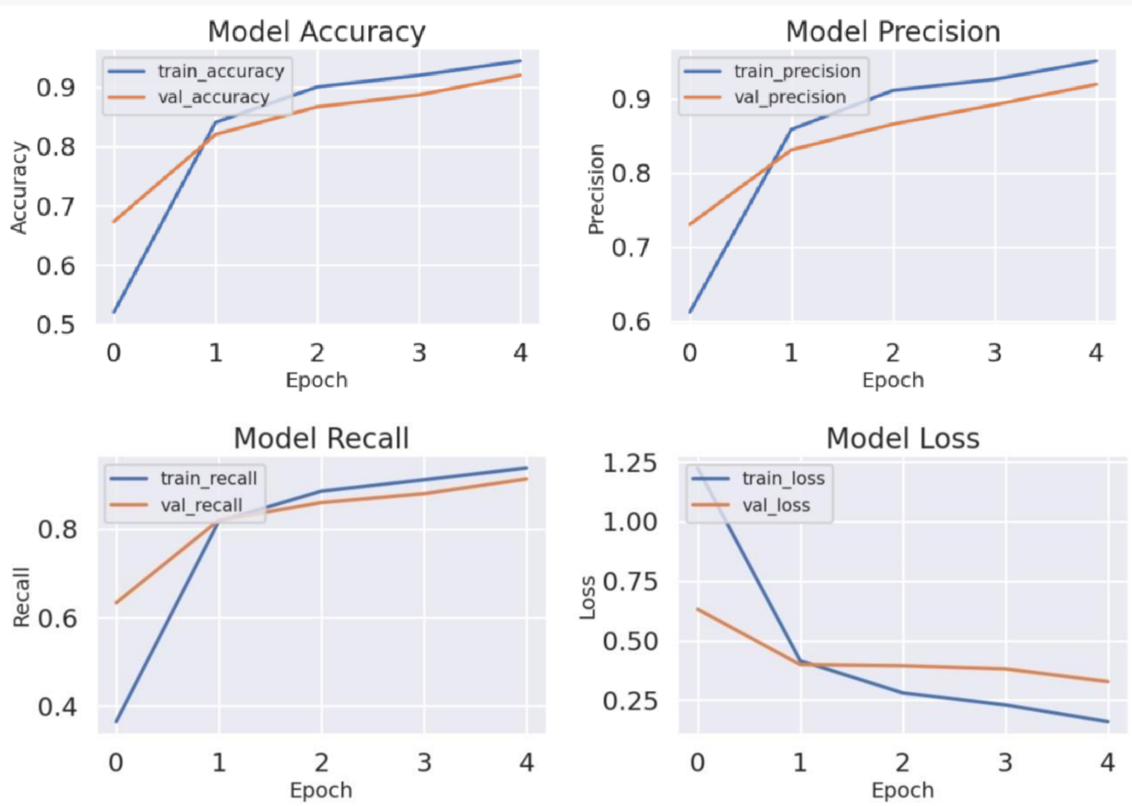
Total params: 11,963,587 (45.64 MB)
Trainable params: 11,963,587 (45.64 MB)
Non-trainable params: 0 (0.00 B)

```
[ ] # Compile the model
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', 'precision', 'recall'])
```

```
history = model.fit(train_generator,
                    batch_size=16,
                    epochs=5,
                    validation_data=validation_generator,
                    validation_batch_size=16
                    )
```

```
Epoch 1/5
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can incl
  self._warn_if_super_not_called()
42/42 ━━━━━━━━━━━━━━━━━━━━ 111s 2s/step - accuracy: 0.4076 - loss: 1.9102 - precision: 0.4520 - recall: 0.2297 - val_accuracy: 0.6733 - val_loss: 0.6316 - val_precision: 0.7308 - val_recall: 0.6333
Epoch 2/5
42/42 ━━━━━━━━━━━━━━━━━━━━ 91s 2s/step - accuracy: 0.8180 - loss: 0.4649 - precision: 0.8435 - recall: 0.7854 - val_accuracy: 0.8200 - val_loss: 0.3993 - val_precision: 0.8311 - val_recall: 0.8200
Epoch 3/5
42/42 ━━━━━━━━━━━━━━━━━━━━ 140s 2s/step - accuracy: 0.9017 - loss: 0.2860 - precision: 0.9080 - recall: 0.8868 - val_accuracy: 0.8667 - val_loss: 0.3943 - val_precision: 0.8658 - val_recall: 0.8600
Epoch 4/5
42/42 ━━━━━━━━━━━━━━━━━━━━ 86s 2s/step - accuracy: 0.9288 - loss: 0.2214 - precision: 0.9348 - recall: 0.9190 - val_accuracy: 0.8867 - val_loss: 0.3810 - val_precision: 0.8919 - val_recall: 0.8800
Epoch 5/5
42/42 ━━━━━━━━━━━━━━━━━━━━ 144s 2s/step - accuracy: 0.9556 - loss: 0.1408 - precision: 0.9627 - recall: 0.9489 - val_accuracy: 0.9200 - val_loss: 0.3280 - val_precision: 0.9195 - val_recall: 0.9133
```

```
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns

# Set up seaborn for styling
sns.set_theme()
sns.set_context("poster")

# Create a figure to plot all metrics
figure(figsize=(14, 10), dpi=100)

# Plot accuracy
plt.subplot(2, 2, 1)
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch',fontsize=18)
plt.ylabel('Accuracy',fontsize=18)
plt.legend(loc='upper left',fontsize=15)

# Plot precision
if 'precision' in history.history:
    plt.subplot(2, 2, 2)
    plt.plot(history.history['precision'], label='train_precision')
    plt.plot(history.history['val_precision'], label='val_precision')
    plt.title('Model Precision')
    plt.xlabel('Epoch',fontsize=18)
    plt.ylabel('Precision',fontsize=18)
    plt.legend(loc='upper left',fontsize=15)

# Plot recall
if 'recall' in history.history:
    plt.subplot(2, 2, 3)
    plt.plot(history.history['recall'], label='train_recall')
    plt.plot(history.history['val_recall'], label='val_recall')
    plt.title('Model Recall')
    plt.xlabel('Epoch',fontsize=18)
    plt.ylabel('Recall',fontsize=18)
    plt.legend(loc='upper left',fontsize=15)

# Plot loss
plt.subplot(2, 2, 4)
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.title('Model Loss')
plt.xlabel('Epoch',fontsize=18)
plt.ylabel('Loss',fontsize=18)
plt.legend(loc='upper left',fontsize=15)

# Adjust layout and show plots
plt.tight_layout()
plt.show()
```



Plots of Performance Metrics

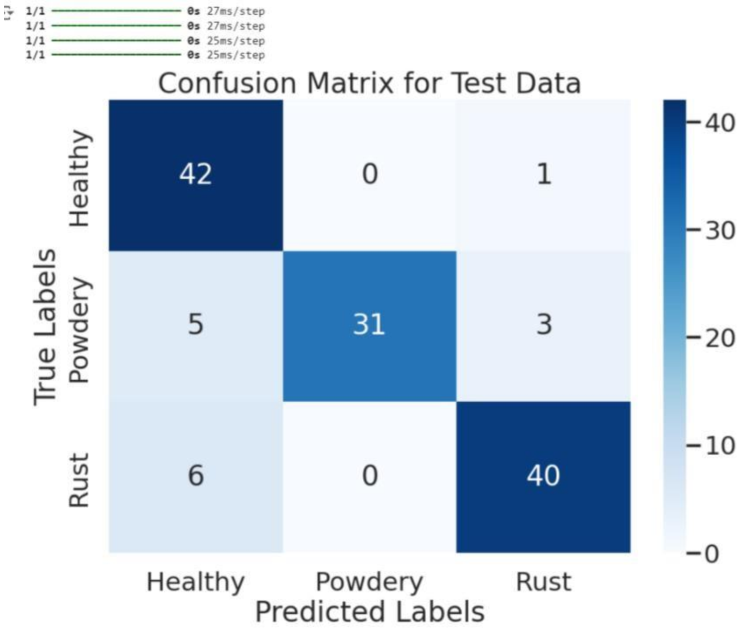
```
[ ] # Saving the Model
    model.save("model.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.sa
```

```python
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np

# Function to preprocess the image
def preprocess_image(image_path, target_size=(225, 225)):
    img = load_img(image_path, target_size=target_size)
    x = img_to_array(img)
    x = x.astype('float32') / 255.
    x = np.expand_dims(x, axis=0)
    return x

# Load and preprocess the image
image_path = '/content/plant_leaf_disease_predictor/Validation/Validation/Powdery/9b6a318cc5721d73.jpg'
x = preprocess_image(image_path)

# Make predictions
predictions = model.predict(x)
predicted_class = np.argmax(predictions[0])

# 'train_generator' is defined and has the class indices
labels = train_generator.class_indices
labels = {v: k for k, v in labels.items()} # Invert the dictionary to map indices to labels

predicted_label = labels[predicted_class]

# Output the predicted label
print(f'Predicted Label: {predicted_label}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 817ms/step
Predicted Label: Powdery
```

```python
import numpy as np
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Generate predictions and true labels for the entire test dataset
def get_all_predictions_and_labels(generator):
    num_samples = generator.samples
    num_batches = num_samples // generator.batch_size
    all_predictions = []
    all_true_labels = []

    for _ in range(num_batches):
        images, labels = generator.__next__()
        predictions = model.predict(images)
        all_predictions.extend(np.argmax(predictions, axis=-1))
        all_true_labels.extend(np.argmax(labels, axis=-1))

    return np.array(all_true_labels), np.array(all_predictions)

# Get all predictions and true labels
true_labels, predicted_labels = get_all_predictions_and_labels(validation_generator)

# Compute the confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Create a figure to plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=train_generator.class_indices.keys(),
            yticklabels=train_generator.class_indices.keys())
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix for Test Data')
plt.show()
```



Confusion Matrix

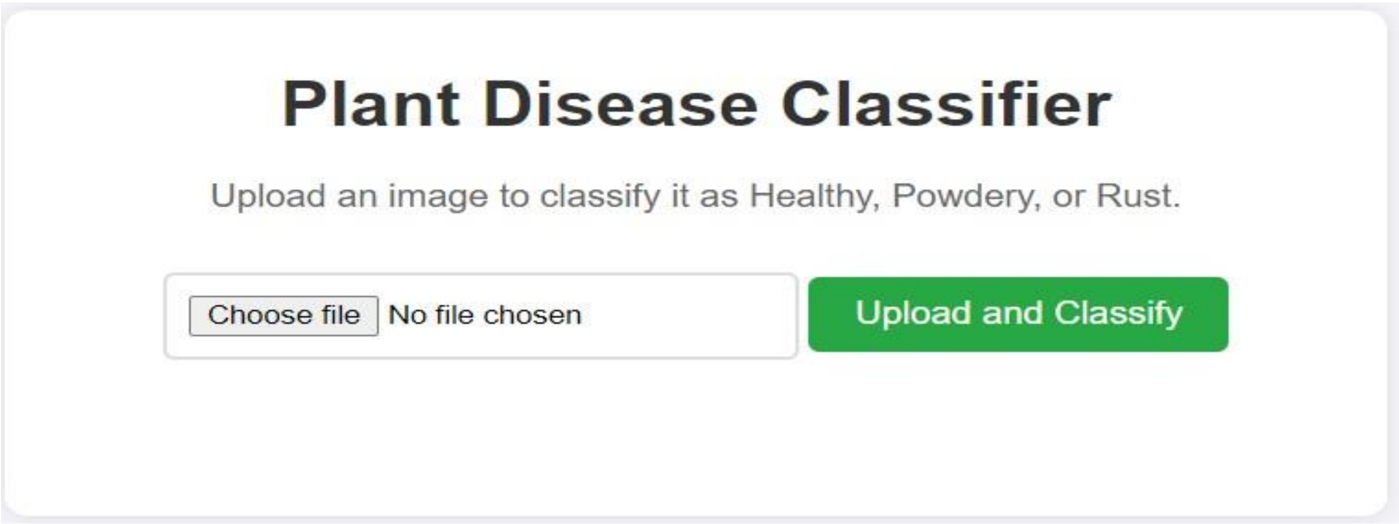
8. Conclusion

The implementation of early and precise disease detection technologies in agriculture is poised to transform the way farmers manage crop health. To support this transformation, we designed a web application for plant leaf disease detection. This proactive tool helps significantly reduce crop losses by identifying diseases early, allowing for timely and efficient interventions. Furthermore, by minimizing the need for excessive pesticide use, the application promotes a healthier environment and safer food production. Early detection and precise treatment ensure that crops remain healthy, which is vital for maximizing yields and maintaining the quality of agricultural produce.

Moreover, integrating these innovative solutions into agricultural practices supports sustainability and contributes to broader food security and economic stability. By reducing reliance on harmful chemicals, farmers can adopt more eco-friendly practices, preserving soil health and biodiversity. This project aims to address key challenges in plant disease management by providing farmers with the tools needed to maintain productive crops, ultimately supporting the agricultural community. By achieving these goals, the initiative not only revolutionizes how plant diseases are managed but also fortifies the agricultural sector's resilience against future challenges. This holistic approach ensures that farming remains a viable and sustainable livelihood, securing food supplies for communities worldwide.

9. Outputs

We see the interface of our website as:



Home page

Result of Prediction as Rust

Result of Prediction as Healthy

Result of Prediction as Powdery