Project Report

Deep Learning Model For Tea Leaf Disease Detection

Team ID- 592327
Team Members: Saatvik Sumanta
Abigna Kotamareddi
Abhigyan Ghoshal
Denesh L

1. **INTRODUCTION**

1.1 Project Overview

- Our team successfully created a tea leaf disease detection system using a custom Convolutional Neural Network (CNN) model, surpassing alternatives like VGG16 and ResNet50 with an 82% accuracy rate. The model is hosted as a user-friendly web application, powered by Flask for the backend and HTML, CSS, and JS for the frontend.
- The system not only identifies tea leaf diseases but also prioritizes simplicity and user understanding. We incorporated data augmentation during training to enhance the model's adaptability.

1.2 Purpose

- The purpose of this project is to empower tea farmers with an effective and accessible tool for early detection of diseases in tea leaves. Through the development of a specialized Convolutional Neural Network (CNN) model, hosted as a user-friendly web application, the system enables farmers to quickly and accurately identify potential issues. By prioritizing simplicity our goal is to enhance the overall health and yield of tea crops. This project aligns with the mission of leveraging technology to support sustainable agriculture practices and contribute to the well-being of tea farming communities.

2. **LITERATURE SURVEY**

2.1 Existing problem

- **Late Disease Detection:**
  Traditional methods of disease detection in tea plants may be time-consuming, leading to late identification of issues and subsequent delays in implementing necessary interventions. The deep learning model aims to provide a faster and more efficient way to identify diseases promptly.
- **Manual Inspection Challenges:**
  Relying solely on manual inspection by farmers could be prone to human error and may not cover large plantations comprehensively. The automated system helps overcome these challenges by providing consistent and objective assessments.
- **Limited Access to Expertise:**
  Small-scale farmers might lack access to agricultural experts who can accurately identify and diagnose diseases. The model serves as a readily available, user-friendly tool, democratizing access to disease detection capabilities.
- **Crop Yield and Quality Impact:**
  Diseases, if not detected and treated in a timely manner, can significantly impact the yield and quality of tea crops. The project aims to mitigate these negative effects by enabling early detection and intervention.
- **Data-Driven Decision Making:**
  Many farmers might not have access to data-driven tools that can assist in decision-making related to crop health. The project integrates machine learning to provide actionable insights, fostering informed decision-making.

2.2 References

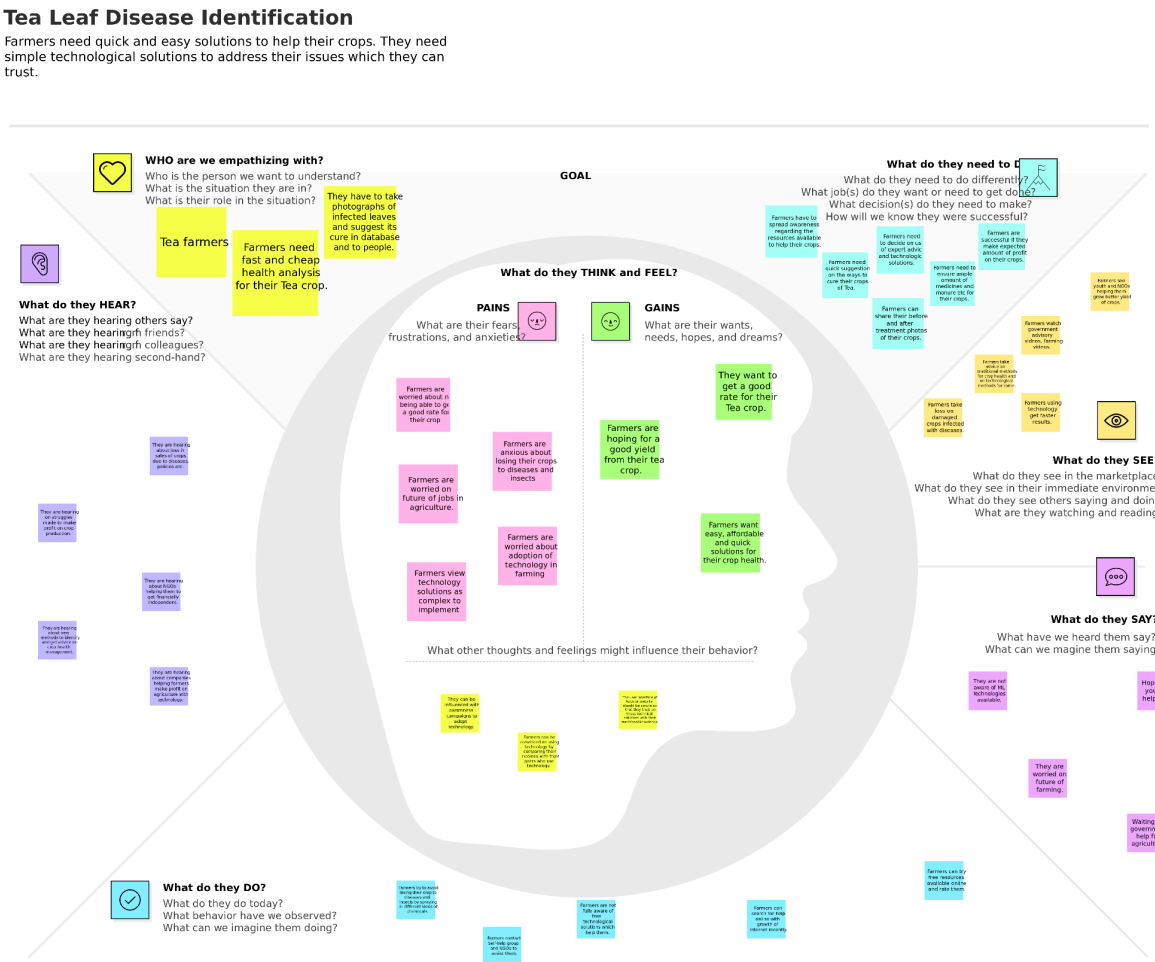
- https://www.nature.com/articles/s41598-023-33270-4
- https://www.sciencedirect.com/science/article/pii/S187705092300203X
- https://www.sciencedirect.com/science/article/abs/pii/S0045790621000471
- https://ieeexplore.ieee.org/document/9402225
- https://arxiv.org/pdf/2311.03240.pdf

2.3 Problem Statement Definition

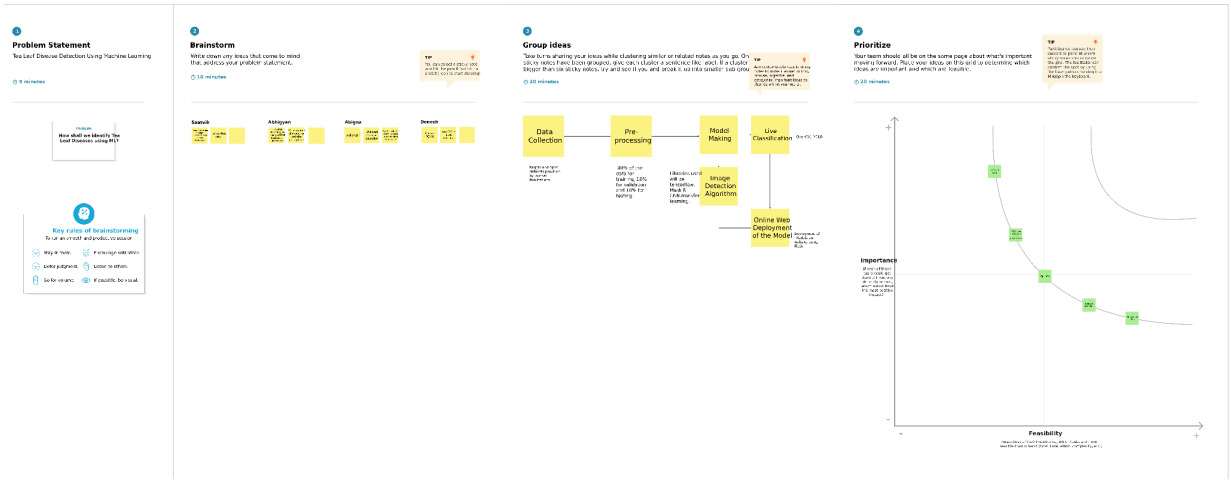
- Tea farmers face challenges in timely and accurate identification of diseases affecting tea leaves, leading to reduced crop yield and quality. Manual inspection methods are often prone to human error, and access to expertise for disease diagnosis is limited, particularly for small-scale farmers. Existing approaches lack the efficiency needed for large plantations, hindering proactive interventions. There is a critical need for a technology-driven solution that combines deep learning for robust disease detection, a user-friendly interface for accessibility, and a feedback mechanism for continuous improvement. This project aims to address these challenges by developing an automated tea leaf disease detection system, providing farmers with a reliable tool to enhance crop health and overall yield.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

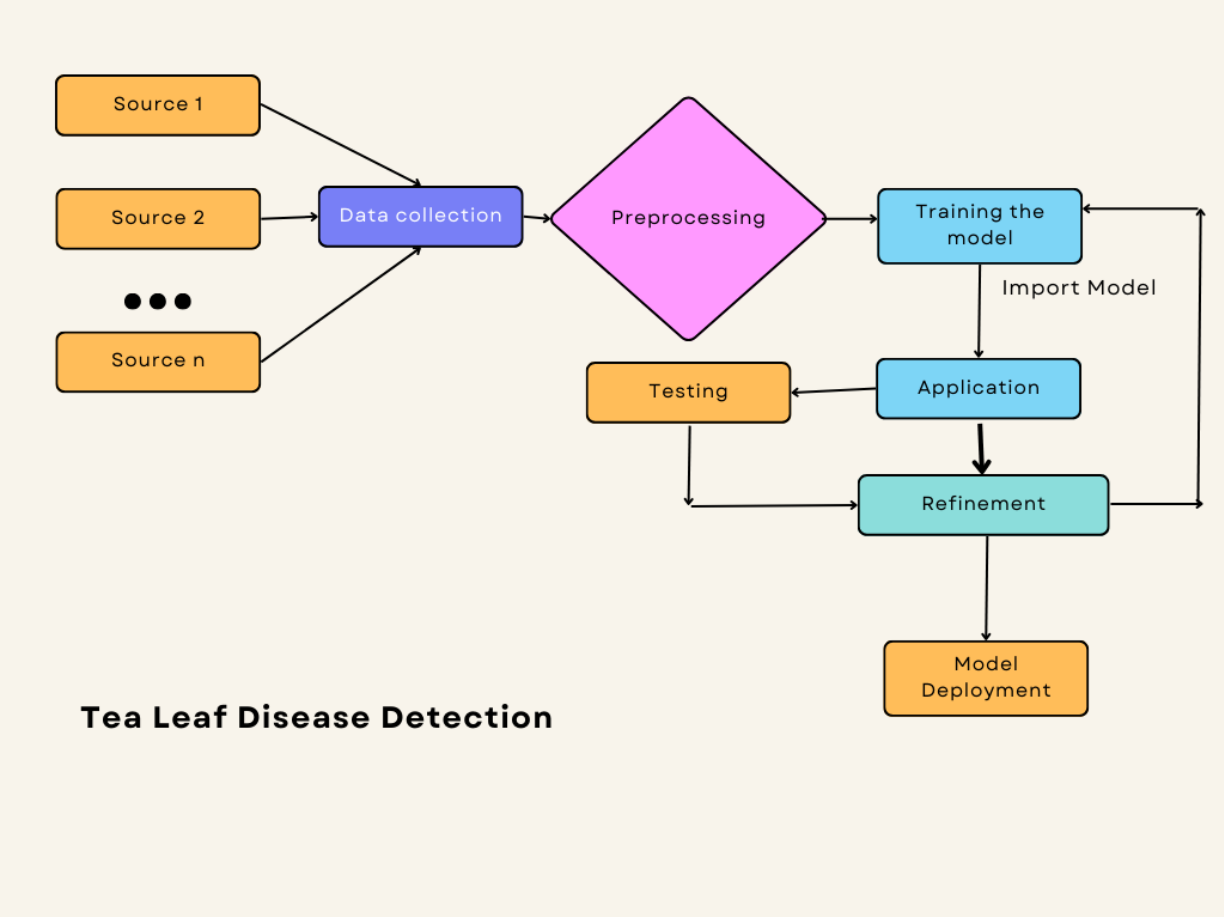
- Image Uploading Data Collection and Preprocessing
- Model Training
- Disease Identification
- User Interface
- Scalability

4.2 Non-Functional requirements

- Performance
- Reliability
- Usability
- Response time

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



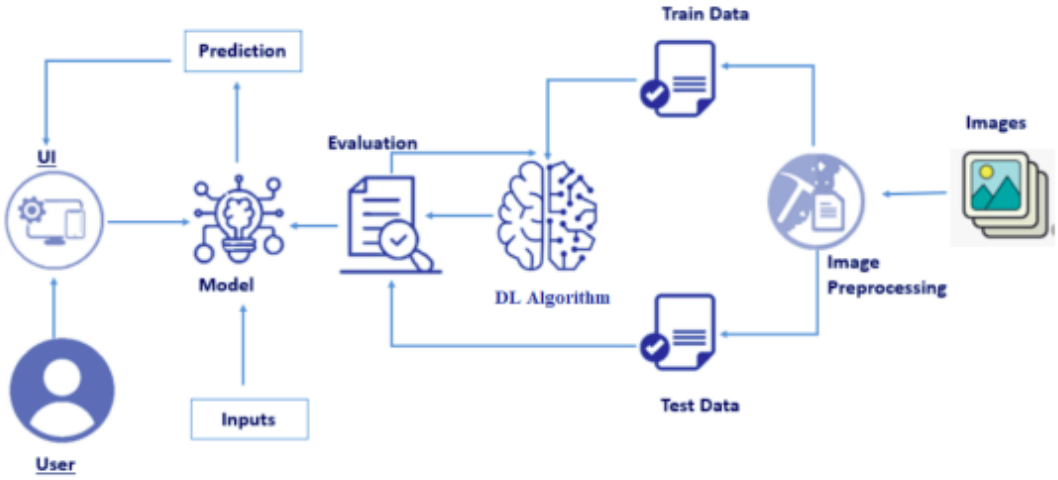
User Type	Functional Requirements	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Tea Farming companies	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to check for tea leaf diseases	successfully configured with all necessary tools and frameworks	High	sprint 1
Municipalities and Local Governments	development environment	USN-2	Gather a diverse dataset of images containing different	Gathered a diverse dataset of images depicting various types	High	sprint 1

			types of tea leaf diseases for training the deep learning model.	of tea leaf diseases		
Households and Individuals	Data collection	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets.	preprocessed the dataset	High	sprint 2
Researchers and Academics	data preprocessing	USN–4	Explore and evaluate different deep learning architectures (e.g., CNNs) to select the most suitable model for tea leaf disease detection	we could explore various DL models	High	sprint 2
Non-Governmental Organizations (NGOs)	model development	USN-5	train the selected deep learning model using the preprocessed dataset and monitor its performanc	we could do validation	High	sprint 3

			e on the validation set.			
Educational Institutions	Training	USN-6	implement data augmentation techniques (e.g., rotation, flipping) to improve the model's robustness and accuracy.	we could do testing	Medium	sprint 3
	model deployment & Integration	USN-7	deploy the trained deep learning model as an API or web service to make it accessible for garbage classification. integrate the model's API into a user-friendly web interface for users to upload images and receive tea leaf images and give results on the kind of tea leaf disease if it exists.	we could check the scalability	Medium	sprint 4
	Testing & quality	USN-8	conduct thorough	we could create web	Medium	sprint 5

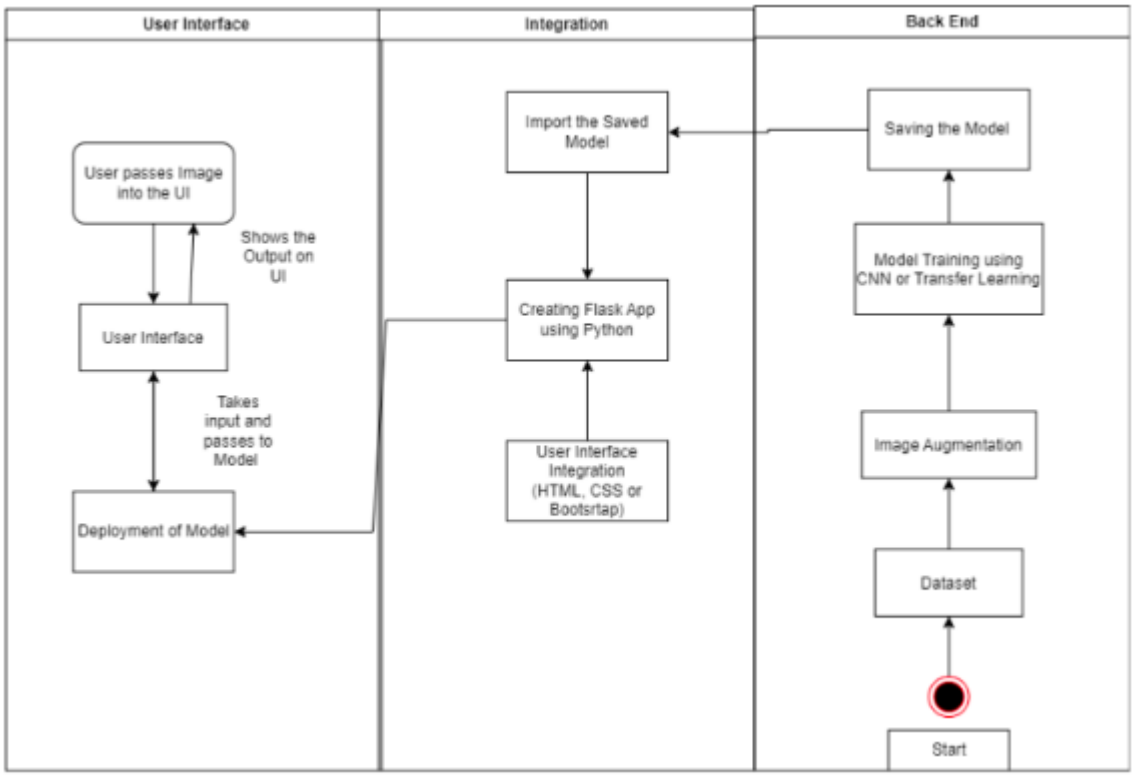
	assurance		testing of the model and web interface to identify and report any issues or bugs. fine-tune the model hyperpara meters and optimize its performanc e based on user feedback and testing results.	application		

5.2 Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

- Sprint 1 - Project setup & Infrastructure, development environment
- Sprint 2 - Data collection & data preprocessing
- Sprint 3 - Training
- Sprint 4 - Model deployment & Integration
- Sprint 5 - Testing & quality assurance

6.3 Sprint Delivery Schedule

Phase	Phase Name	Files to be Submitted	Deadline
Phase 1	Assignment & Quiz	Assignments	NA
Phase 2	Ideation Phase	Empathise & Discover	18.10.2023
		Brainstorm & Prioritize Ideas	
Phase 3	Project Design Phase	Proposed solution	23.10.2023
		Solution Architecture	
		Determine the Requirements (Data Flow Diagram & User Stories)	
Phase 4	Project Planning Phase	Technical Architecture	27.10.2023
		Project Planning Details	
Phase 5	Project Development Phase	Project Files	06.11.2023
Phase 6	Performance & Final Submission Phase	Solution Performance	09.11.2023
		Project Documentation	
		Project Demonstration	

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Image Detection Model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import PIL

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from sklearn.metrics import classification_report
import pathlib
dataset_path = "/content/tea sickness dataset"
dataset_dir = pathlib.Path(dataset_path)
# list of sub directory(class)

class_names = []
for root, dirs, files in os.walk(dataset_path):
   if len(root) > len(dataset_path):
       x_class = os.path.relpath(root, dataset_path)
       class_names.append(x_class)

print(class_names)
# print total number of images in the dataset
for class_i in class_names:
   image_count = len(list(dataset_dir.glob(f'{class_i}/*.jpg')))
   print(f"Images in class {class_i}:",image_count)
# Parameter setting
train_batch = 128
val_batch = 128
```

```
img_height = 224
img_width = 224
IMG_SIZE = (img_height, img_width)
val_split = 0.2

# Load the training dataset
train_ds = tf.keras.utils.image_dataset_from_directory(
   dataset_path,
   validation_split=val_split,
   subset="training",
   seed=123,
   image_size=(img_height, img_width),
   batch_size=train_batch
)
# Load data for Validation
val_ds = tf.keras.utils.image_dataset_from_directory(dataset_dir,

validation_split=val_split,

subset="validation",
                                                    seed=123,

image_size=(img_height, img_width),

batch_size=val_batch
                                                   )
class_names = train_ds.class_names
print(class_names)

num_classes=len(class_names)
# Review dataset sample
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
```

```
   for i in range(9):
       ax = plt.subplot(3, 3, i + 1)
       plt.imshow(images[i].numpy().astype("uint8"))
       plt.title(class_names[labels[i]])
       plt.axis("off")
AUTOTUNE = tf.data.AUTOTUNE


train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
#Data augmentation layers to increase data variation for training
data_augmentation = keras.Sequential([

layers.RandomFlip("horizontal",input_shape=(img_height,img_width,3
)),
   layers.RandomFlip("vertical"),
   layers.RandomRotation(0.2),
   layers.RandomZoom(0.2),
])
# Model architecture
cnn_model = Sequential([
   data_augmentation,
   layers.Rescaling(1./255),
   layers.Conv2D(16, 3, padding='same', activation='relu'),
   layers.MaxPooling2D(),
   layers.Conv2D(16, 3, padding='same', activation='relu'),
   layers.MaxPooling2D(),
   layers.Conv2D(32, 3, padding='same', activation='relu'),
   layers.MaxPooling2D(),
   layers.Flatten(),
   layers.Dense(32, activation='relu'),
   layers.Dense(32, activation='relu'),
```

```
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(32, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(num_classes)   #num_classes=len(class_names) i.e 8
])
# compile model
base_learning_rate = 0.0005
cnn_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate
=base_learning_rate),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=Tru
e),
              metrics=['accuracy'])
# display model
cnn_model.summary()
epochs=300
history = cnn_model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    verbose=0
)
# Check training result
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

epochs_range = range(epochs)
```

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
# Retrieve a batch of images from the test set
image_batch, label_batch = val_ds.as_numpy_iterator().next()
predictions = cnn_model.predict_on_batch(image_batch)

class_predictions = []
for i in predictions:
   class_prediction = np.argmax(i)
   class_predictions.append(class_prediction)

class_predictions = np.array(class_predictions)
print('Predictions:\n', class_predictions)
print('Labels:\n', label_batch)
print()
print(classification_report(label_batch, class_predictions))

plt.figure(figsize=(10, 21))
for i in range(18):
   ax = plt.subplot(6, 3, i + 1)
```

```
   plt.imshow(image_batch[i].astype("uint8"))
   plt.title("Prediction:
"+class_names[class_predictions[i]]+"\nLabel:
"+class_names[label_batch[i]])
   plt.axis("off")
```

7.2 Web App

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Falling Leaves Animation</title>
   <link rel="stylesheet" type="text/css" href="{{
url_for('static', filename='style.css') }}">
   <style>
       body {
           font-family: Arial, sans-serif;
           margin: 0;
           padding: 0;
       }
  table {
       margin: 0 auto;
     }

       #grey-rectangle {
           width: 1000px;
           height: 600px;
           background-color: rgba(204, 204, 204, 0.5);
           position: absolute;
           top: 50%;
           left: 50%;
           transform: translate(-50%, -50%);
           border-radius: 10px;
           padding: 20px;
       }
```

```
        #writing-area {
            background-color: rgba(255, 255, 255, 0.7);
            padding: 10px;
            border-radius: 10px;
            text-align: center;
        }

        #upload-container {
            width: 10px;
            height: 50px;
            background-color: rgba(204, 204, 204, 0);
            position: absolute;
            top: 85%;
            left: 5%;
            transform: translate(-50%, -50%);
            border-radius: 10px;
            display: flex;
            align-items: center;
        }

        #upload-box {
            margin-right: 10px;
        }

        input[type="file"] {
            margin-right: 10px;
        }

        .prediction {
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div id="grey-rectangle">
        <div id="writing-area">
            <!-- Add your content here --><h3><u>Convolutional
Neural Network Based Tea Leaf Disease Detection</u></h3>
            <div><br>This Web App helps the user identify the
```

```
condition of their Tea Plants.
            <br>The following 8 Leaf Health conditions can be
classified using the CNN Model with an accuracy of <b>82%</b>.
            <br><br>
            <table>
   <tr>
       <td style="text-align: center;">
           <img src="{{ url_for('static', filename='Healthy.jpg')
}}" alt="Healthy Leaf" width="100" height="100"><br>
           Healthy Leaf
       </td>
       <td style="text-align: center;">
           <img src="{{ url_for('static', filename='Algal.jpg')
}}" alt="Algal Leaf Disease" width="100" height="100"><br>
           Algal Leaf Disease
       </td>
       <td style="text-align: center;">
           <img src="{{ url_for('static',
filename='Anthracnose.jpg') }}" alt="Anthracnose" width="100"
height="100"><br>
           Anthracnose
       </td>
       <td style="text-align: center;">
           <img src="{{ url_for('static', filename='Bird Eye
Spot.jpg') }}" alt="Bird Eye Spot" width="100" height="100"><br>
           Bird Eye Spot
       </td>
   </tr>
   <tr>
       <td style="text-align: center;">
           <img src="{{ url_for('static', filename='Gray
Blight.jpg') }}" alt="Gray Blight" width="100" height="100"><br>
           Gray Blight
       </td>
       <td rowspan="2" style="text-align: center;">
           <img src="{{ url_for('static', filename='Red Leaf
Spot.jpg') }}" alt="Red Leaf Spot" width="100" height="100"><br>
           Red Leaf Spot
       </td>
       <td>
```

```
            <img src="{{ url_for('static', filename='White
Spot.jpg') }}" alt="White Spot" width="100" height="100"><br>
            White Spot
        </td>
        <td style="text-align: center;">
            <img src="{{ url_for('static', filename='Brown
Blight.jpg') }}" alt="Brown Blight" width="100" height="100"><br>
            Brown Blight
        </td>
    </tr>

</table>
</div>
        </div>
        <div id="upload-container">
            <div id="upload-box">
                <form action="/predict" method="post"
enctype="multipart/form-data">
                    <input type="file" name="image" id="fileInput"
accept=".jpg, .jpeg, .png">
                    <input type="submit" value="Predict">
                </form>
            </div>
            <div class="prediction">Predicted class: {{ prediction
}}</div>
        </div>
    </div>

    <div id="canvas-container">
        <canvas id="canvas"></canvas>
    </div>

    <script src="{{ url_for('static', filename='script.js')
}}"></script>
</body>
</html>
```

Flask App

```
from flask import Flask, render_template, request
import pickle
```

```
import numpy as np
import pandas as pd
import PIL


app = Flask(__name__, static_folder='static',
static_url_path='/static')

model =
pickle.load(open('CNN_Tea_Leaf_Disease_Detection.pkl','rb'))

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

@app.route('/predict', methods=['POST'])
def upload():
   uploaded_file = request.files['image']
   if uploaded_file.filename != '':
       uploaded_file.save(uploaded_file.filename)  # Save the
uploaded file
       #return "File uploaded successfully!"
       return predict()
   return "No file uploaded."

def predict():
   # Get the image from the request
   image = request.files["image"]
   image = PIL.Image.open(image).resize((224, 224))
   x = np.array(image)
   x = np.expand_dims(x, axis=0)
   pred = np.argmax(model.predict(x))

   op = ['Anthracnose', 'Algal Leaf', 'Bird Eye Spot', 'Brown
Blight', 'Gray Blight', 'Healthy', 'Red Leaf Spot', 'White Spot']

   # Return the prediction
   return render_template("index.html", prediction=op[pred])
```

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

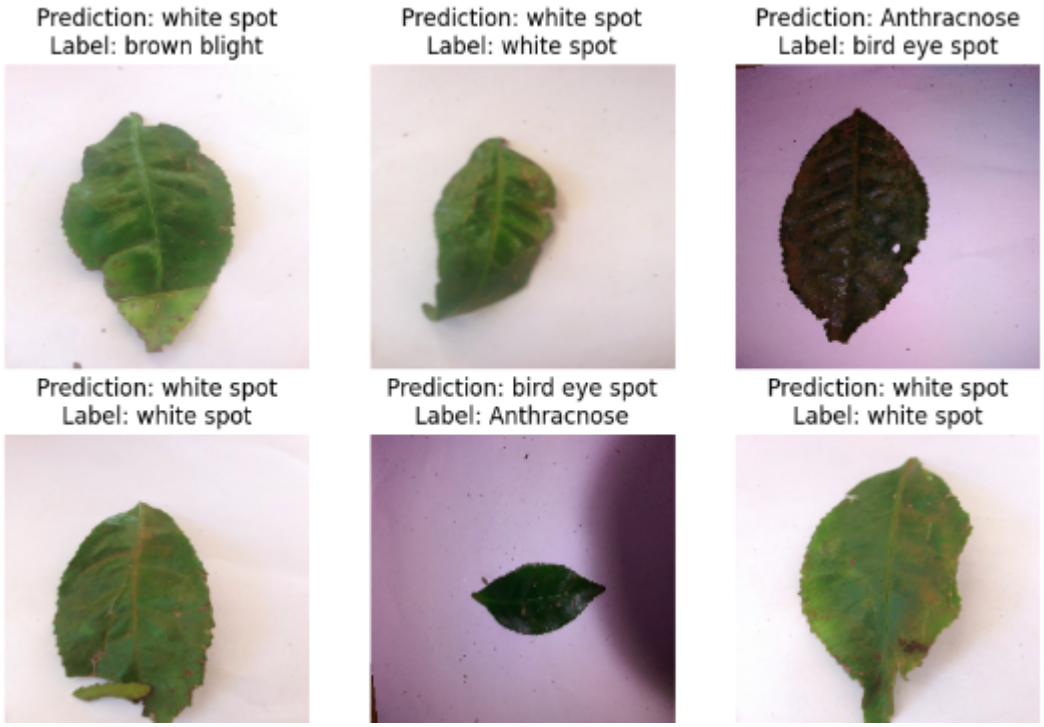
8. PERFORMANCE TESTING

8.1 Performace Metrics

	precision	recall	f1-score	support
0	0.86	0.67	0.75	18
1	0.76	1.00	0.87	13
2	0.60	0.82	0.69	11
3	0.81	0.74	0.77	23
4	0.90	0.90	0.90	20
5	1.00	1.00	1.00	7
6	1.00	1.00	1.00	15
7	0.74	0.67	0.70	21
accuracy			0.82	128
macro avg	0.83	0.85	0.84	128
weighted avg	0.83	0.82	0.82	128

9. RESULTS

9.1 Output Screenshots



10. ADVANTAGES & DISADVANTAGES

Advantages:

- ➢ Early Disease Detection
- ➢ Increased Crop Yield
- ➢ Accessibility
- ➢ Cost Efficient
- ➢ Continuous Improvement
- ➢ Scalability
- ➢ Data Driven Decision Making

Disadvantages:

- ➢ Dependency on Data Quality
- ➢ Limited Scope
- ➢ Internet Dependency
- ➢ Resistance to change/ Adoption of technology over traditional methods

11. CONCLUSION

In conclusion, the tea leaf disease detection project represents a significant leap toward precision agriculture. By leveraging a custom Convolutional Neural Network (CNN) model deployed as a user-friendly web application, the system addresses crucial challenges in the timely identification of diseases, potentially improving crop yield and quality. The incorporation of interpretability features and a feedback loop enhances user trust and contributes to continuous model improvement. While offering promising advantages, challenges such as data quality dependency, security concerns, and the need for ongoing maintenance must be carefully managed. Overall, this project not only demonstrates the potential of technology in agriculture but also highlights the importance of ethical considerations and user engagement for the successful implementation of innovative solutions in real-world farming contexts.

12. FUTURE SCOPE

- Expansion to Other Crops
- Integration of IoT Devices
- Machine Learning Model Enhancements
- Mobile Application Development
- Collaboration with Agricultural Experts
- Localization for Regional Agriculture
- Partnerships with Agricultural Agencies
- Education and Training Programs

13. APPENDIX

Source Code: https://github.com/smartinternz02/SI-GuidedProject-600840-1697472978/tree/main

GitHub & Project Demo Link: https://youtu.be/xevo5geBMxs