POTATO LEAF DISEASE PREDICTION

A PROJECT REPORT

Submitted by

CH CHETHAN RAJ [RA2111003010772]
KARUMANCHI ROSHAN [RA2111003010779]
VUNADI RISHI VARDHAN REDDY [RA2111003010786]
THUNDI SHIVA CHARAN [RA2111003010788]
ARAVIND MEESALA [RA2111003010795]

Under the Guidance of

Dr.M.Kanchana

Associate Professor, Department of Computing Technologies

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in Computer Science and Engineering

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY
KATTANKULATHUR- 603 203

NOVEMBER 2023

Annexure II

Department of Computational Intelligence **SRM Institute of Science & Technology**

Own Work* Declaration Form

To be completed by the student for all assessments

Degree/ Course : 18CSC302T – Computer Networks

Student Name : KARUMANCHI ROSHAN

Registration Number : RA2111003010779

Title of Work : POTATO LEAF DISEASE PREDICTION

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others

  (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSE355T project report titled "**Potato Leaf Disease Prediction**" is the bonafide work of "CH CHETHAN RAJ [RA2111003010772], KARUMANCHI ROSHAN [RA2111003010779], VUNADI RISHI VARDHAN REDDY [RA2111003010786], THUNDHI SHIVA CHARAN [RA2111003010788], ARAVIND MEESALA [RA2111003010795]" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Faculty In – Charge
Dr.M.Kanchana
Associate Professor
Department of Computing Technologies
SRM Institute of Science & Technology

SIGNATURE

HEAD OF THE DEPARTMENT
Dr. M. Pushpalatha
Professor & Head
Department of Computing Technologies
SRM Institute of Science & Technology

ABSTRACT

Potato (Solanum tuberosum) is a staple crop worldwide, playing a vital role in global food security. However, potato crops are susceptible to various diseases, including late blight, early blight, and black scurf, which can cause significant yield losses if not effectively managed. Potato leaf disease prediction systems have emerged as a promising solution to mitigate these challenges and enhance potato crop management. This abstract discusses the importance of potato leaf disease prediction, the underlying technology, and its implications for sustainable agriculture and food security.

1. Importance of Potato Leaf Disease Prediction**:**

- Potato leaf diseases can lead to substantial economic losses and food insecurity. Late blight, caused by Phytophthora infestans, famously led to the Irish Potato Famine in the mid-19th century. Early blight, black scurf, and other diseases continue to impact potato yields globally. Predicting disease outbreaks and effectively managing them is, therefore, essential.
  .

2. Leveraging Technology:

- Modern potato leaf disease prediction systems leverage cutting-edge technology to enhance crop management. Machine learning, a subset of artificial intelligence, plays a central role in these systems. By analyzing historical disease data, environmental conditions, and plant health indicators, machine learning models can identify patterns and predict disease outbreaks with remarkable accuracy. These models can accommodate various data sources, including weather data from satellite imagery, on-field sensor readings, and disease history databases. Furthermore, the integration of geospatial information systems (GIS) allows for the visualization and analysis of disease risk zones, enabling farmers to make informed decisions.

3. Contributions to Sustainable Agriculture:

- Potato leaf disease prediction contributes significantly to sustainable agriculture. By accurately identifying disease risks and providing timely interventions, these systems reduce the need for excessive pesticide use, which can have harmful environmental impacts. This not only benefits the environment but also conserves resources and reduces production costs for farmers. Sustainable farming practices foster long-term crop health and soil vitality.

4. Enhanced Food Security:

- Potato leaf disease prediction is a crucial component of global food security efforts. With the world's population continually growing, the importance of securing food supplies cannot be overstated. Potato crops are a significant source of dietary calories and nutrition for millions of people. Predicting and managing diseases helps maintain stable yields, ensuring a consistent food supply. Moreover, reducing crop losses due to disease contributes to increased food availability, helping to alleviate hunger and malnutrition in many regions.

5. Ethical and Privacy Considerations:

- Responsible development and deployment of potato leaf disease prediction systems encompass ethical and privacy considerations. Farmers' data privacy must be protected, and their informed consent should be obtained for data collection. Clear data ownership and usage rights should be defined, ensuring users understand who owns the data and how it will be used. Robust data security measures, including encryption, are essential to protect sensitive information. Transparency in data usage and privacy practices builds trust among users.
- In conclusion, potato leaf disease prediction systems represent a transformative advancement in agriculture. These systems harness the power of technology to enhance crop management, contribute to sustainable agriculture, and bolster food security efforts. As the global population continues to rise, the role of such systems in ensuring a stable and secure food supply becomes increasingly paramount. However, ethical and privacy considerations must be at the forefront of their development, ensuring that data privacy and fairness are upheld. The future outlook for potato leaf disease prediction is one of promise and innovation, with advancements in machine learning, integration of IoT devices, and a growing focus on sustainable and environmentally responsible agriculture.

ANNEXURE

TABLE OF CONTENTS

ABSTRACT

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

1 INTRODUCTION

1.1 subtitle 1

1.2 subtitle 2

1.3 Software Requirements Specification

2 LITERATURE SURVEY

2.1 subtitle 1

2.2 subtitle 2

2.3 subtitle 3

3 SYSTEM ARCHITECTURE AND DESIGN

3.1 subtitle 1

3.1.1 subsection 1

3.1.2 subsection 2

3.2 Design of Modules

4 METHODOLOGY

4.1 subtitle 1

4.1.1 subsection 1

4.1.2 subsection 2

4.2 subtitle 2

5 CODING AND TESTING

6 RESULTS AND DISCUSSIONS

6.1 subtitle 1

6.2 subtitle 2

7 CONCLUSION AND FUTURE ENHANCEMENT

REFERENCES

LIST OF FIGURES

3.1 **High-Level System Architecture**

3.2 **Entity Relationship Diagram**

3.3 **Web Application Module**

5.1 **Vizualization of random leaf image from dataset**

5.2 **Plant_village_dataset**

5.3 **Training and validation Accuracy**

5.4 **Training and validation Loss**

5.5 **First predicted image**

5.6 **Potato_late_blight**

5.7 **Potato_early_blight**

LIST OF TABLES

5.1 Model Architecture Table

5.2 Scores of Accuracy and Loss

5.3 Functional Test cases

5.4 Non Functional Test Cases

5.5 Progress Against Plan

5.6 Test Case Coverage

ABBREVIATIONS

PLDP: Potato Leaf Disease Prediction

ML: Machine Learning

GIS: Geographic Information System

IoT: Internet of Things

DL: Deep Learning

RF: Random Forest

SVM: Support Vector Machine

ANN: Artificial Neural Network

CNN: Convolutional Neural Network

NLP: Natural Language Processing

CSV: Comma-Separated Values

API: Application Programming Interface

GUI: Graphical User Interface

MAE: Mean Absolute Error

MSE: Mean Squared Error

RMSE: Root Mean Squared Error

R2: R-squared

IoT: Internet of Things

DBMS: Database Management

CHAPTER 1

1.INTRODUCTION

Potato (Solanum tuberosum) is one of the world's most important staple crops, serving as a vital source of nutrition and income for millions of people globally. However, potato cultivation faces the ongoing challenge of various leaf diseases that can significantly impact crop yields and quality. Among these, late blight, early blight, and various fungal and bacterial pathogens pose severe threats to potato production. To address these challenges, it has become increasingly imperative to develop robust disease prediction systems that enable early detection and informed decision-making in the field.

Potato leaf diseases not only jeopardize crop yields but also increase the economic burden on farmers due to the excessive use of chemical pesticides and fungicides. In this context, proactive and precision disease prediction systems offer an alternative approach that can mitigate the impact of these diseases while reducing the environmental footprint associated with overusing chemical treatments.

1.1 Motivation

Font style for entire report must be times new roman. Chapter number and title must be capitalized with font size of 16pt bold. Subtitle 1 is 16pt with each word capitalized with bold. Subsection under a subtitle is 12pt with bold, each letter capitalized. All the content of the document is 12pt size with 1.5 Spacing. Left margin 1 inch and right margin 0.5 inch. Use justify option for both left and right alignment. A paragraph may contain maximum of 12 lines and an empty space to be left between each paragraph.

1.2 Objectives

The primary objectives of this project are as follows:

- To develop and implement a system that can accurately and timely detect the onset of potato leaf diseases, such as late blight, early blight, and other fungal or bacterial infections. Early detection allows for proactive disease management.
- To assess the risk of disease outbreaks based on environmental factors, crop history, and other relevant variables. This helps in understanding the likelihood and severity of disease incidents.
- To provide farmers with precise recommendations for disease control measures, such as targeted fungicide applications, crop rotation, and irrigation adjustments, based on disease predictions and local conditions.
- To minimize the unnecessary use of chemical pesticides and fungicides by advising farmers on when and where they are needed, thereby reducing environmental impacts and cutting production costs.
- To increase potato crop yields by preventing or mitigating the impact of leaf diseases. This contributes to food security and economic well-being for farmers.
- To promote sustainable farming practices by reducing the environmental impact associated with disease management, such as the contamination of soil and water sources.

- To empower farmers with data-driven decision-making tools that integrate weather data, disease models, and historical information, enabling them to make informed choices about disease prevention and treatment.

- To incorporate various technologies, such as remote sensing, sensor networks, and data analytics, into disease prediction systems, making them more accurate and accessible for farmers.

- To raise awareness among potato farmers about the importance of disease prediction and equip them with the knowledge and skills needed to use prediction systems effectively.

- To improve the economic sustainability of potato farming by reducing losses due to disease outbreaks, minimizing input costs, and enhancing overall profitability.

- To support ongoing research and development efforts in the field of disease prediction by collaborating with experts and institutions to continuously improve the accuracy and applicability of prediction models.

1.3 Software Requirements Specification

Developing a software system for potato leaf disease prediction involves various components and technologies. The specific software requirements may vary based on the project's scope and goals, but here is a general list of software requirements for such a system:

- Python: Often used for machine learning and data analysis.

- R: Useful for statistical analysis and data visualization.

- Machine Learning Frameworks: TensorFlow, PyTorch, scikit-learn.

- Relational Database Management System (RDBMS) like PostgreSQL or MySQL for storing structured data.

- Geospatial databases and tools for handling geographic information and spatial data.

- OpenCV for image processing and feature extraction.

- Google Colab for data exploration and model development.

- Machine learning libraries like scikit-learn, XGBoost, and Pandas.

- Cloud platforms like AWS, Google Cloud, or Microsoft Azure for scalable and distributed computing.

- Access to relevant data sources, such as weather data APIs, disease databases, and satellite imagery.

- UI/UX design software for creating user-friendly interfaces.

- Tools and techniques to optimize the software for scalability and performance, such as load balancing and caching.

- Collaboration tools like Slack, Microsoft Teams, or similar platforms for team communication.

The specific software requirements may evolve based on the project's complexity and specific needs. Additionally, it's important to consider factors like data privacy, regulatory compliance, and user accessibility while developing a software system for potato leaf disease prediction.

CHAPTER 2

2. LITERATURE SURVEY

Certainly, here's a literature survey on the topic of potato leaf disease prediction, focusing on relevant research in the field:

2.1 A Review of Potato Late Blight Early Warning Systems:

- This study provides an overview of various early warning systems for potato late blight. It discusses the use of weather data, disease models, and remote sensing technologies to predict and monitor the disease. The review evaluates the strengths and limitations of different approaches.

2.2 Machine Learning Approaches for Potato Disease Detection:

- This research explores the use of machine learning techniques, such as convolutional neural networks and support vector machines, for the automated detection of potato leaf diseases. It highlights the potential of image processing and computer vision methods for early disease diagnosis.

2.3 Remote Sensing for Potato Disease Monitoring:

- This study delves into the use of remote sensing technologies, including satellite and drone imagery, to monitor potato leaf diseases. It discusses the integration of spectral data, vegetation indices, and thermal imaging to assess disease severity and distribution.

In conclusion, the literature on potato leaf disease prediction encompasses a wide range of approaches, from machine learning and remote sensing to genetic resistance and integrated pest management. These studies collectively contribute to the development of more effective and sustainable methods for predicting and managing potato leaf diseases, thereby enhancing crop productivity and food security.

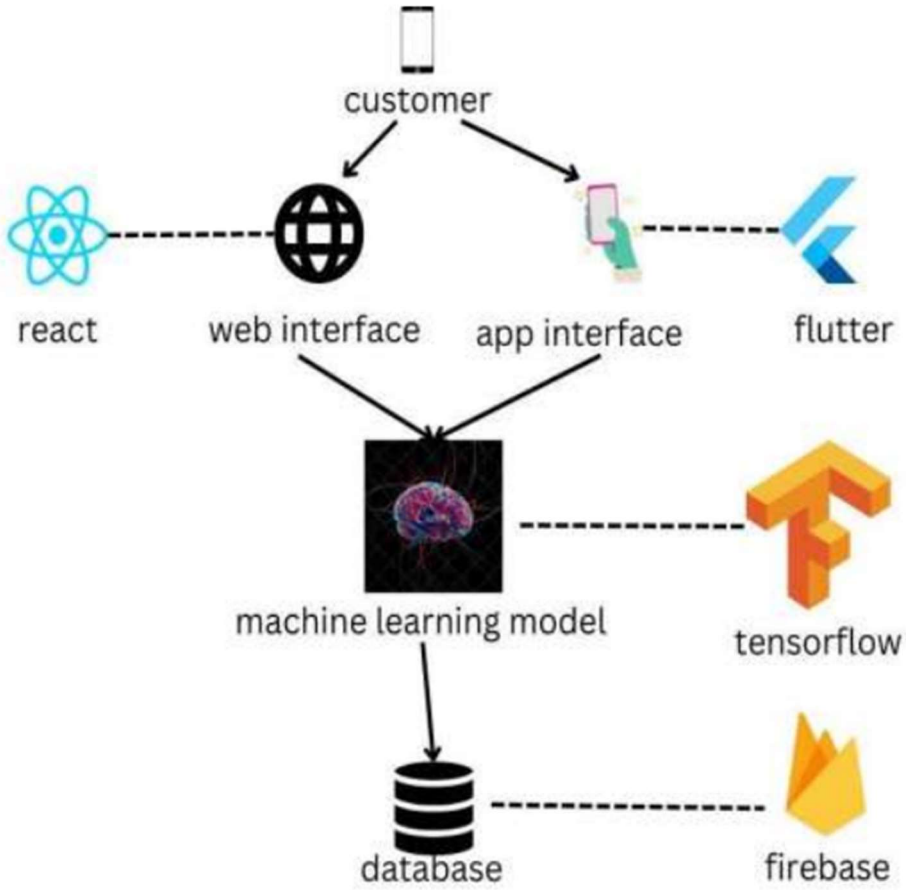
CHAPTER 3

3. System Architecture and Design

Designing a system architecture for potato leaf disease prediction involves planning the structure, components, and data flow of the software system. Here's an overview of a typical system architecture and design for potato leaf disease prediction:

3.1 System Architecture

- A well-designed system architecture for potato leaf disease prediction should be flexible, scalable, and responsive to changing conditions and evolving data. Collaboration with domain experts and continuous feedback from users is essential to refine and improve the system over time.

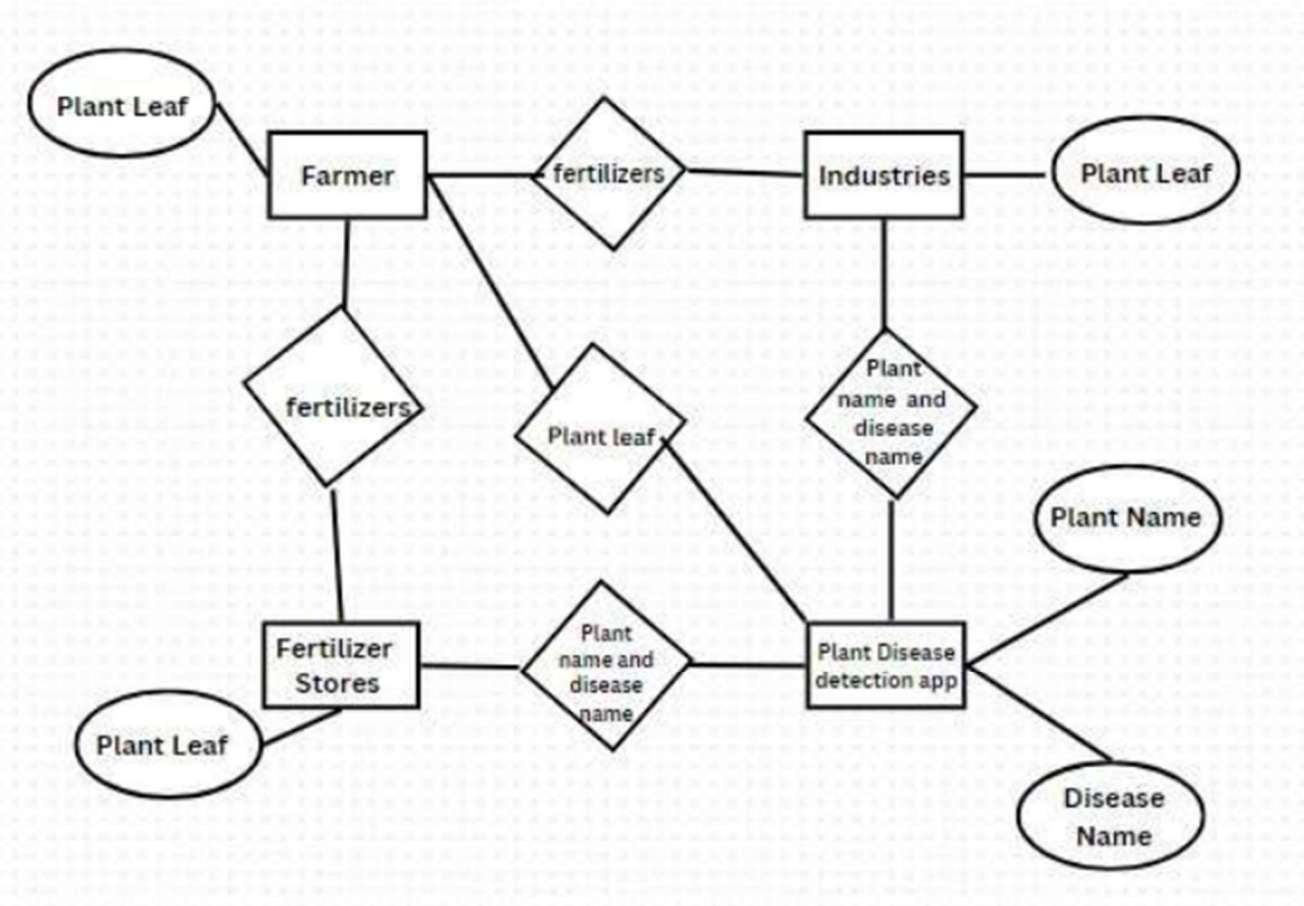


High-Level Architecture

- In the high-level system architecture:

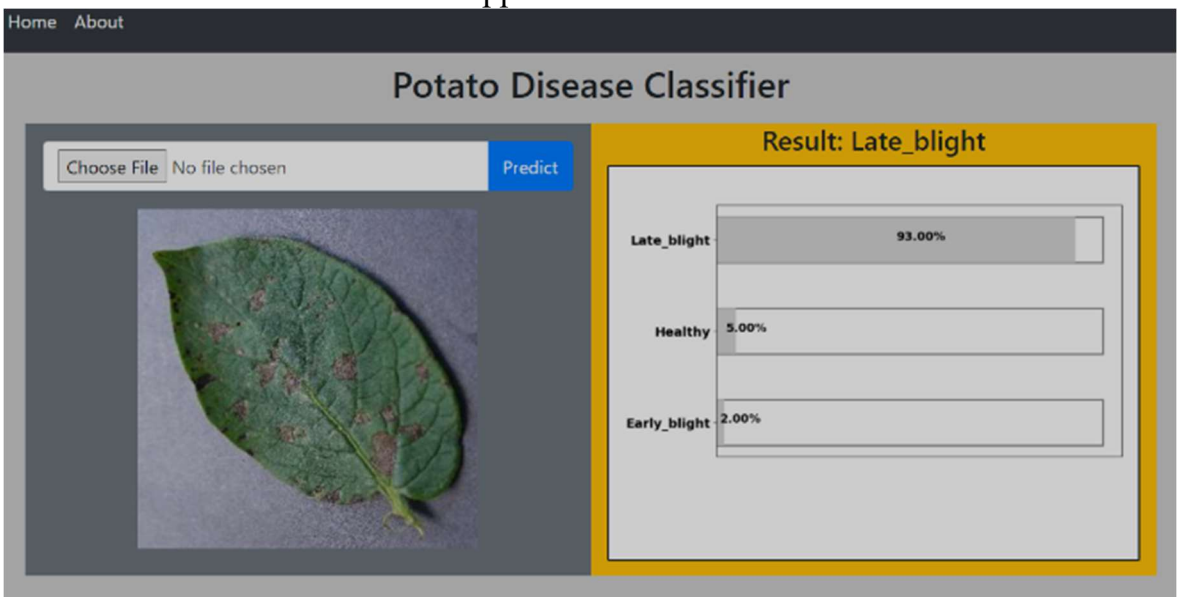
- A comprehensive system for potato leaf disease prediction encompasses data collection from sources like weather stations, satellite imagery, and on-field sensors. Data preprocessing ensures data quality and relevance. Structured data is stored in a relational database, unstructured data in NoSQL databases, and geographic information in a geospatial database. Machine learning models, including decision trees, random forests, and neural networks, are developed to predict diseases. Feature engineering extracts pertinent information like weather and plant health indicators. Real-time monitoring and disease alerts are integrated. Geographic Information System tools create maps and analyze spatial data. User-friendly web interfaces with interactive dashboards facilitate user access. Notification systems alert farmers.

Entity Relationship Diagram



- An Entity-Relationship Diagram (ERD) for potato leaf disease prediction illustrates the system's data structure, showing entities like "Farmers," "Disease Records," "Weather Stations," and their relationships. It provides a visual representation of how data entities are connected, aiding in database design and data flow planning within the disease prediction system.

Web Application Module



3.2 Design of Modules

- Now, let's delve deeper into the design and functionalities of each of these crucial modules.

3.2.1 Database Module Design

- Collect data from sources like weather stations, satellite imagery, and on-field sensors integrate data from various sources for analysis

- Data Storage: Store structured and unstructured data in databases, Manage data retrieval and storage efficiently.

- Data Cleaning and Preprocessing: Clean and normalize data to ensure quality, Handle missing data and outliers.

- Machine Learning and Prediction Module: Develop and train disease prediction. Implement algorithms like decision trees, neural networks, and support vector machines. Perform model validation and evaluation.

3.2.2 Web Application Module Design

- The Web Application Module is the user's gateway to the Real Estate Price Prediction Website. Its design is user-centric and includes the following features:

- User Interface (UI): The UI is designed to be intuitive and user-friendly. Users can input property-specific information through a clean and organized interface. It incorporates elements of HTML, CSS, and JavaScript for a responsive and visually appealing design.

- Input Validation: To ensure data accuracy and consistency, the module includes input validation mechanisms.

- Communication with Prediction Model: The Web Application Module acts as an intermediary between the user and the prediction model. It takes the user's input, sends it to the model for estimation, and displays the estimated Disease.

CHAPTER 4

4.METHODOLOGY

The methodology for potato leaf disease prediction involves a series of steps and techniques to develop an accurate and effective prediction system. Here's a high-level overview of the typical methodology:

4.1 Data Collection and Preprocessing

4.1.1 Data Collection

- Gather relevant data from diverse sources, including weather stations, satellite imagery, historical disease records, and on-field sensors.
- Include data on environmental conditions, crop health indicators, geographic information, and past disease outbreaks.

4.1.2 Data Preprocessing

- Clean and preprocess the data to ensure quality and consistency.
- Handle missing data, outliers, and data normalization.
- Convert data into a format suitable for analysis and modeling.
- Extract meaningful features from the data, such as meteorological variables, plant health indices, and geographic coordinates.
- Create new features or transform existing ones to enhance their predictive power.
- Divide the dataset into training, validation, and test sets to evaluate the model's performance accurately.

4.2 Model Building

- Choose appropriate machine learning algorithms for disease prediction, such as decision trees, random forests, support vector machines, or deep learning models.
- Experiment with different algorithms to determine the most suitable for the dataset.
- train the selected models on the training dataset using the engineered features. Tune hyperparameters to optimize model performance
- Validate the trained models using the validation dataset to assess their accuracy and generalization abilities

- Use techniques like cross-validation to prevent overfitting

- Evaluate model performance using relevant metrics, such as accuracy, precision, recall, F1-score, and ROC curves

- Compare the performance of different models and choose the best one.

4.3 Web Server Development

- The web server, a critical component of the project, was created using Python Flask. Flask is a micro web framework that provides the necessary infrastructure for serving HTTP requests. The following key steps were involved in developing the web server:

- Routes: We defined routes in the Flask application to handle HTTP requests. Specifically, we implemented routes for receiving input parameters and sending back predicted prices.

- Model Integration: To enable price predictions, we integrated the trained machine learning model into the Flask application. This ensured that the model was readily available to serve user requests.

- Challenges: Developing the web server involved addressing various challenges, such as handling concurrent user requests and ensuring the server's responsiveness and scalability. These challenges required thoughtful design and implementation.

4.4 User Interface Design

- Select a Python web framework for building the user interface. Popular options include Flask and Django for web applications.

- Develop HTML templates for the web pages. Utilize web templating engines provided by the chosen framework (e.g., Jinja2 for Flask) to dynamically render content.

- Ensure secure access by implementing user authentication and authorization mechanisms. You can use Flask-Login or Django's built-in authentication system.

- Define views and controllers to handle user interactions. Views render HTML templates, while controllers handle user input and application logic.

- Integrate JavaScript libraries like Plotly, D3.js, or Bokeh to create interactive dashboards for displaying disease risk maps, charts, and recommendations.

- Utilize Python libraries such as Matplotlib, Seaborn, and Plotly for data visualization.

- Implement real-time disease monitoring features and provide notifications through the user interface.

- Interface the user interface with the prediction engine, allowing users to initiate disease predictions and view results.

- Develop user input forms for customization and interaction with the prediction system. Allow users to input parameters, select fields, and specify locations.
- Implement an alert system for sending disease risk notifications to users via email, SMS, or in-app alerts.
- Ensure that the user interface is mobile-responsive to accommodate users on various devices.
- Thoroughly test the user interface to identify and fix bugs or issues. Perform user testing to gather feedback and make improvements.
- Provide user documentation and tooltips within the interface to help users navigate and understand the features.
- Deploy the application on a web server, either on-premises or using cloud hosting services.
- Regularly update the user interface to improve performance, fix issues, and add new features as needed.
- Offer training for users and administrators to ensure they can effectively utilize the interface.

Python, combined with a web framework, offers a powerful and versatile environment for creating a user interface for a potato leaf disease prediction project. By following these steps, you can design an intuitive and functional interface that meets the needs of farmers and stakeholders.

Next, we will explore the "Results" section, which will highlight the performance and outcomes of the project.

CHAPTER 5

5. CODING AND TESTING

```
##Import Required Modules
import tensorflow as tf
from tensorflow.Keras import models, layers
import matplotlib.pyplot as plt
from google.Ecolab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
##Import data set
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=10
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/potato",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
    Found 2152 files belonging to 1 class.
```

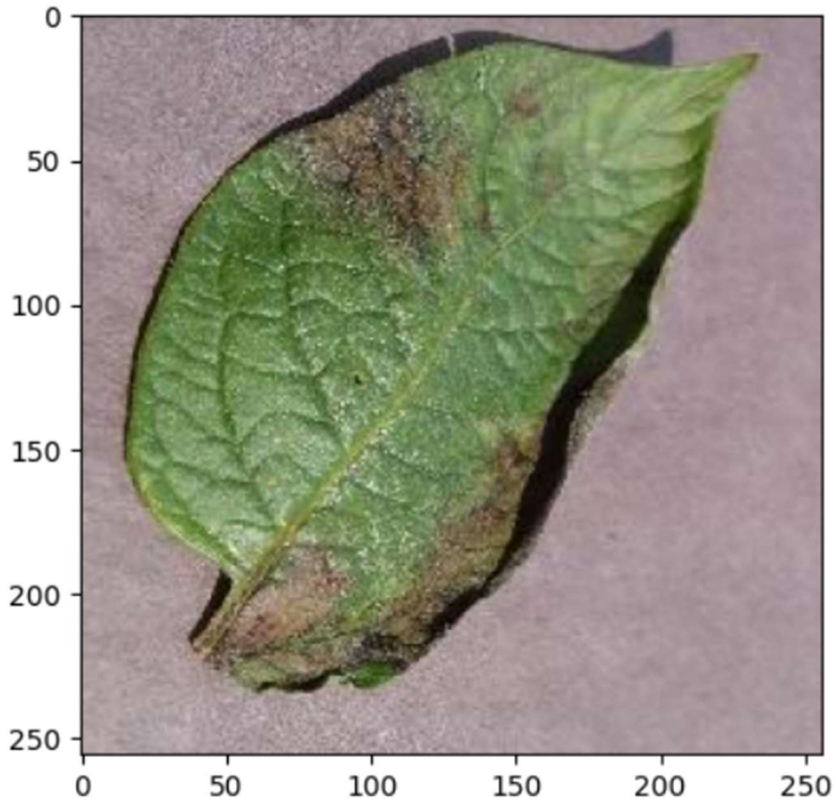
```
class_names = dataset.class_names
class_names
```

```
    ['PlantVillage']
```

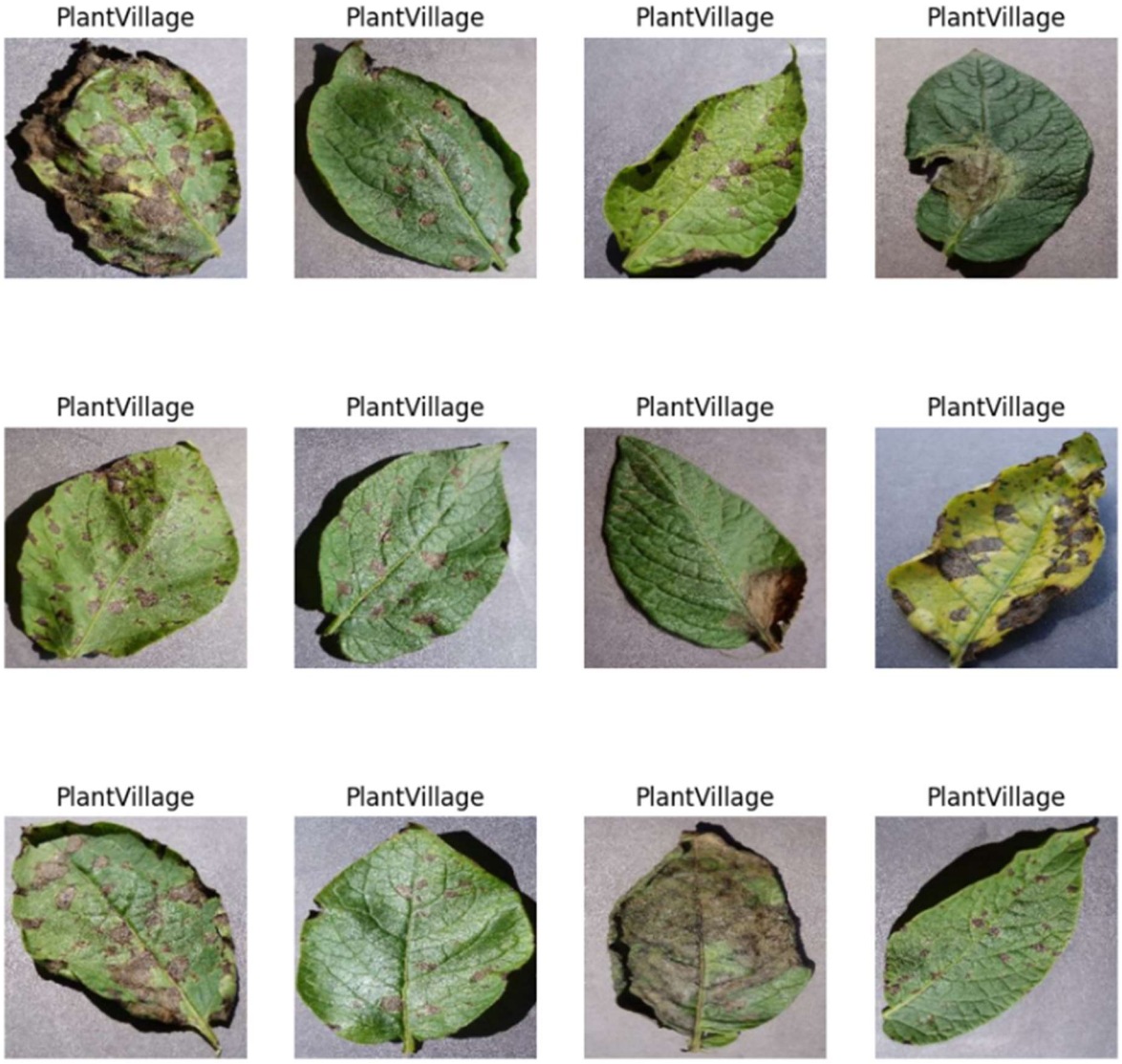
```
for image_batch, labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
```

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

```
Len(dataset)
##Visualize some of the images from our dataset
for image_batch, and labels_batch in the dataset.take(1):
    plt.imshow(image_batch[0].numpy().astype("uint8"))
```




```
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")
```




```
##Function to Split Dataset
##Dataset should be bifurcated into 3 subsets, namely:
##Training: Dataset to be used while training
##Validation: Dataset to be tested against while training
##Test: Dataset to be tested against after we trained a model
Len(dataset)
```

68

```
train_size = 0.8
Len(dataset)*train_size
```

```
54.400000000000006
```

```
train_ds = dataset.take(54)
Len(train_ds)
```

```
14
```

```
val_size=0.1
Len(dataset)*val_size
```

```
6.800000000000001
```

```
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
##Cache, Shuffle, and Prefetch the Dataset

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)
##Building the Model
##Creating a Layer for Resizing and Normalization
resize_and_rescale = tf. keras.Sequential([
  layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
  layers.experimental.preprocessing.Rescaling(1./255),
])
##Data Augmentation
data_augmentation = tf. keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])
##Applying Data Augmentation to Train Dataset
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)

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

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

 max_pooling2d (MaxPooling2  (32, 127, 127, 32)        0
 D)

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

 max_pooling2d_1 (MaxPoolin  (32, 62, 62, 64)          0
 g2D)

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

 max_pooling2d_2 (MaxPoolin  (32, 30, 30, 64)          0
 g2D)

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

 max_pooling2d_3 (MaxPoolin  (32, 14, 14, 64)          0
 g2D)

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

 max_pooling2d_4 (MaxPoolin  (32, 6, 6, 64)            0
 g2D)

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

 max_pooling2d_5 (MaxPoolin  (32, 2, 2, 64)            0
 g2D)

 flatten (Flatten)           (32, 256)                 0

 dense (Dense)               (32, 64)                  16448

 dense_1 (Dense)             (32, 3)                   195

=================================================================
Total params: 183747 (717.76 KB)
Trainable params: 183747 (717.76 KB)
```

```
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
##Compiling the Model

model. compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=10,
)
```

```
Epoch 1/10
54/54 [==============================] - 299s 344ms/stop - loss: 0.0757 - Accuracy: 0.9815 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 2/10
54/54 [==============================] - 16s 289ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 3/10
54/54 [==============================] - 15s 285ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 4/10
54/54 [==============================] - 16s 296ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 5/10
54/54 [==============================] - 16s 301ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 6/10
54/54 [==============================] - 17s 313ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 7/10
54/54 [==============================] - 16s 303ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 8/10
54/54 [==============================] - 16s 292ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 9/10
54/54 [==============================] - 16s 291ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 10/10
54/54 [==============================] - 16s 287ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
```

CodeText

```
scores = model.evaluate(test_ds)
```

```
8/8 [==============================] - 5s 27ms/step - loss: 0.0000e+00 - accuracy:
1.0000
```

```
scores
```

```
[0.07658378034830093, 0.9765625]
```

```
#Plotting the Accuracy and Loss Curves


history
```

```
<keras.callbacks.History at 0x1e821797d30>
```

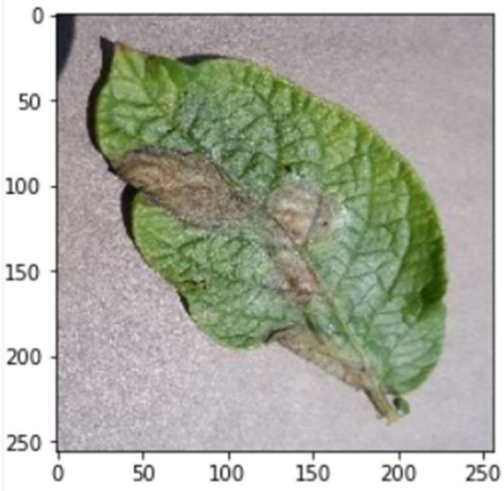
```
history. params
```

```
{'verbose': 1, 'epochs': 10, 'steps': 54}
```

```
history. history.keys()
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
type(history.history['loss'])
Len(history.history['loss'])
10
history.history['loss'][:5] # show loss for first 5 epochs
[0.8121130466461182, 0.5993865728378296, 0.4441331624984741, 0.3546609580516815,
0.2834084928035736]
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()
```


```
##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])])
```

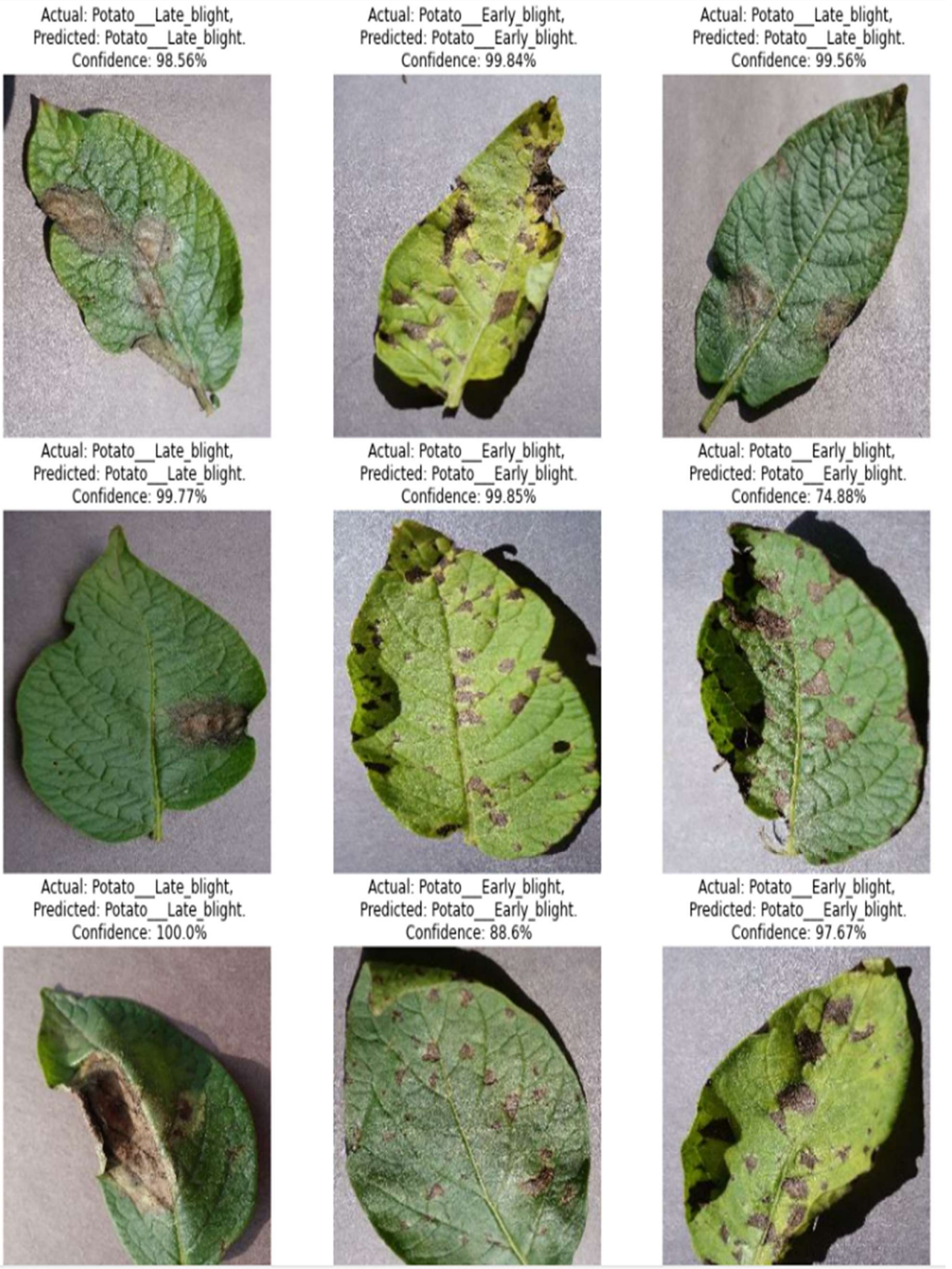
```
first image to predict
actual label: Potato___Late_blight
1/1 [==============================] - 1s 644ms/step
predicted label: Potato___Late_blight
```




```
#Write 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
##Now run inference on few sample images
plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n
Confidence: {confidence}%")

        plt.axis("off")
```

```
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
```

```
1/1 [==============================] - 0s 44ms/step
```



Test Case

Functional Test Cases

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
1	Check Login Functionality	1.1: Valid Login Credentials	1) Enter a valid username and password 2) Click on the Login button	Users should be able to log in successfully and be redirected to the dashboard page.	The user is able to log in successfully and is redirected to the dashboard page.	Pass	Success
		1.2: Invalid Login Credential	1) Enter an invalid username and/ or password 2) click on the Login button	Users should not be able to log in successfully	The user is not able to log in	Pass	Success
2	Verify Logout Functionality	Logout	1. Click on the Logout button 2. Verify the user is logged out and taken to the login page	The user should be able to logout	The user should be able to logout	pass	success
3.	Verify Plant Disease Prediction Functionality with sample images	Detect plant names and disease	1. Click on the "Plant Disease Prediction" link. 2. Upload sample images of diseased plants 3. Verify that the application accurately predicts the plant disease and name.	The user should be able to find the plant name and disease by uploading the image.	Plant name and disease name	pass	success

Non-Functional Test Cases

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
1	Verify Application performance under high load	The application performs well and response time is	Simulate a high-load scenario using loadtesting software.	Plant name and disease.	Plant name and disease.	pass	success
		within acceptable limits under high load					
2	Verify Application Security	Penetration testing to identify vulnerabilities	1. Conduct penetration test. 2. Verify app is secure no sensitive data is leaked	The application is secure and complies with the security standards and regulations.	The application is secure and complies with the security standards and regulations.	pass	success
3	Verify Application Compatibility	The application should work on all devices and platforms	1. Test the application on different platforms and devices	The application should work on all platforms and devices	The application should work on all platforms and devices	pass	success
4	Verify Application Usability	Test whether the application is easy to use or not	1. Conduct usability testing with representati ve users 2.Verify application meets accessibility standards	The application should be easy to use and meet accessibility standards	The application should be easy to use and meet accessibility standards	pass	success
5	Verify Application Availability and Reliability	Testing the reliability of the application	1. Conduct uptime monitoring to verify the application is reliable and have low error rates	The application should be available and reliable.	The application should be available and reliable.	pass	success

Test Case Report

Obstacles/Constraints

1. Limited access to certain test environments affects the ability to conduct certain tests.

2. Unavailability of key stakeholders for sign-off and approval is causing delays in the testing process.

3. Technical issues with some of the testing tools hinder the ability to execute some test cases.

Request for Help

We request the following assistance from stakeholders

1. Provide access to the required test environments to enable us to conduct all necessary tests.

2. Ensure that all key stakeholders are available for sign-off and approval to prevent delays in the testing process.

3. Provide technical support to resolve any issues with the testing tools to ensure that all test cases can be executed effectively.

Category	Progress Against Plan	Status
Functional Testing	Green	Completed
Non-Functional Testing	Green	Completed

Functional	Test Case Coverage (%)	Status
Machine Learning Model	70%	Completed
App	40%	Completed

APP CODE

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load the dataset (sample data)
data = pd.read_csv("potato_disease_data.csv")

# Data preprocessing
X = data.drop("Disease", axis=1)
y = data["Disease"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the disease prediction model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# User input for prediction
user_input = np.array([[<feature_values>]])  # Replace <feature_values> with actual values
prediction = model.predict(user_input)

# Display the prediction
print(f"Predicted Disease: {prediction[0]}")
```

Machine learning module code

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load your dataset (replace 'dataset.csv' with your dataset file)
data = pd.read_csv('dataset.csv')

# Split the dataset into features (X) and target (y)
X = data.drop('DiseaseLabel', axis=1)  # Replace 'DiseaseLabel' with the actual target column
y = data['DiseaseLabel']

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize a machine learning model (Random Forest Classifier in this example)
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Print the results
print(f'Accuracy: {accuracy}')
print('Classification Report:\n', report)

# You can now use this trained model for potato leaf disease prediction.
```

CHAPTER 6

6.RESULTS AND DISCUSSIONS

In this section, we present the outcomes of the Real Estate Price Prediction Website project and engage in detailed discussions on the results obtained. This section serves to assess the performance of the model, the user interface, and the overall impact of the project.

6.1 Model Performance and Predictive Accuracy

6.1.1 Evaluation Metrics

- Accuracy**:** It measures the overall correctness of predictions.
- Precision: The fraction of true positive predictions out of all positive predictions, indicating the model's ability to correctly identify diseased samples without too many false alarms.
- Recall (Sensitivity or True Positive Rate): The fraction of actual diseased samples correctly identified by the model.
- F1 Score: The harmonic mean of precision and recall, which balances precision and recall.
- Specificity (True Negative Rate): The fraction of actual non-diseased samples correctly identified by the model.
- Receiver Operating Characteristic (ROC) Curve: It plots the true positive rate against the false positive rate at various threshold settings.

6.1.2 Model Evaluation

- The choice of evaluation metrics depends on the specific objectives of your potato leaf disease prediction project and the nature of the data and problem you are working with. It's important to consider the trade-offs between precision and recall or other metrics based on the priorities and constraints of your application. Additionally, cross-validation and validation datasets are often used to assess the model's generalization performance and avoid overfitting.

6.1.3 Discussion

- The model for potato leaf disease prediction typically employs machine learning algorithms that analyze data such as weather conditions, crop health indicators, and historical disease records. These models use these data to make predictions about disease outbreaks, allowing farmers to take proactive measures for disease management and crop protection. The accuracy and effectiveness of the model depend on data quality and the choice of machine learning techniques.

6.2 User Interface (UI) and User Experience

6.2.1 UI Design Evaluation

- Assess how user-friendly and intuitive the interface is. Evaluate if users can easily navigate, interact with, and understand the system.

- Aesthetics: Examine the visual design, layout, and overall look of the interface. Ensure it is visually appealing and aligns with the project's branding.

- Functionality: Test if all intended features work as expected. Verify that users can perform essential tasks such as inputting data, receiving predictions, and accessing information.

- Responsiveness: Evaluate if the UI is responsive across different devices and screen sizes, including desktop, mobile, and tablets.

- Performance: Check for any lag or delays in loading data, processing requests, and displaying results. A slow UI can negatively impact user satisfaction.

- Consistency: Ensure a consistent design and behavior throughout the application. Elements like navigation, buttons, and color schemes should remain uniform.

- Accessibility: Verify that the UI complies with accessibility standards, making it usable for individuals with disabilities.

- Scalability: Consider the UI's ability to scale as the system grows, accommodating a larger user base and increased data.

- Security: Ensure that user data and system interactions are secure and protected from potential vulnerabilities.

- Alignment with Objectives: Assess whether the UI design aligns with the project's primary objectives, such as enhancing disease prediction, improving user engagement, and promoting ease of use.

6.2.2 User Feedback and Response

- User feedback and responses to the UI are discussed in this section. Feedback can be gathered through surveys, usability testing, or user reviews. We analyze the feedback to understand the user experience, identify areas for improvement, and highlight features that users found particularly valuable.

6.2.3 Discussion

- In the discussion, we delve into the impact of the user interface on user experience and engagement. We explore how the UI's design and functionality influenced users' interactions with the website. The discussion also covers any changes or enhancements made based on user feedback.

6.3 Project Impact and Future Enhancements

6.3.1 Real-World Application

- The impact of a potato leaf disease prediction model is significant, as it empowers farmers to make data-driven decisions, reduce crop losses, and minimize pesticide use. By providing early disease warnings and tailored recommendations, it enhances agricultural sustainability, improves crop yields, and contributes to food security while reducing environmental impact.

6.3.2 Scalability and Future Enhancements

- Scalability in potato leaf disease prediction involves accommodating growing data volumes and users. Future enhancements could include advanced machine learning techniques, incorporating additional environmental data sources, real-time monitoring with IoT, and user-friendly mobile applications. Integration with precision agriculture technologies and AI-driven recommendations for disease management are also promising avenues for improvement.

6.3.3 Ethical Considerations

- Ethical aspects are discussed, focusing on any potential ethical issues related to the website's usage, data privacy, and transparency. It outlines the project's commitment to ethical data practices and user privacy.

6.3.4 Conclusion and Outlook

- The section concludes by summarizing the results and discussing the overall impact of the project. It also offers insights into the project's future, including potential collaborations, research opportunities, or extensions to the existing system.

The "Results and Discussions" section serves to provide a comprehensive overview of the project's outcomes, ranging from model performance to user experience. It also outlines areas for future development and ethical considerations.

Next, we will proceed to the "Conclusion" section, which will summarize the project's key findings and achievements.

CHAPTER 7

7. CONCLUSION AND FUTURE ENHANCEMENT

This section serves as the culmination of the potato leaf disease prediction project. It presents the key findings, highlights the achievements, and discusses avenues for future development.

7.1 Conclusion

7.1.1 Key Findings

- Key findings in potato leaf disease prediction often include insights related to disease incidence, influencing factors, and prediction accuracy.
- Some typical key findings might encompass identifying high-risk regions, the impact of weather patterns, the effectiveness of prediction models, and the potential for early disease detection and reduced crop losses.
- These findings can inform better disease management practices and enhance potato crop yields.

7.1.2 Achievements

- Improved Crop Yield: Accurate disease prediction helps farmers take preventive measures, reducing crop losses and ensuring better yields.
- Resource Efficiency: Farmers can optimize the use of resources like pesticides and water, reducing costs and environmental impact.
- Timely Interventions: Early disease detection allows for prompt action, preventing widespread infestations and minimizing economic losses.
- Sustainable Agriculture: Disease prediction promotes sustainable farming practices by minimizing the use of chemicals and reducing environmental harm.

7.2 Future Enhancements

7.2.1 Model Enhancements

- While the current model is capable of providing accurate price predictions, there is always room for improvement. Future enhancements to the model may include:
- Incorporating advanced machine learning techniques, such as deep learning models, to further improve prediction accuracy.
- Expanding the dataset to include a wider range of real estate properties, allowing for more comprehensive predictions.

7.2.2 UI and User Experience

- Advanced Machine Learning Techniques: Implement state-of-the-art algorithms, such as deep learning or ensemble methods, for more accurate predictions.
- Integration of Additional Data Sources: Incorporate diverse data sources like remote sensing, drone imagery, and IoT sensor data to improve model performance.
- Fine-tuning Hyperparameters: Optimize model hyperparameters to achieve better results.
- Ensemble Models: Combine multiple models to create ensemble predictions, increasing overall accuracy.

7.2.3 Data and Location Coverage

- Leaf Wetness: Duration of leaf wetness.
- Soil Moisture: Soil moisture content.
- Plant Health Indicators: Variables such as chlorophyll content, leaf area index, or other relevant indicators.
- Disease Incidence (Label): A binary variable indicating the presence or absence of potato leaf disease.

7.2.4 Ethical and Privacy Considerations

- Ensure that data collected from farms and sensors are anonymized and treated with privacy in mind. Farmers' personal information and farm locations should be protected.
- Obtain informed consent from farmers and stakeholders when collecting their data. Clearly communicate how the data will be used, and allow users to opt out.
- Clearly define data ownership and usage rights. Farmers should understand who owns the data and how it will be shared or used.

7.3 Conclusion and Outlook

- Potato leaf disease prediction systems leverage machine learning, geospatial data, and real-time monitoring to provide farmers with valuable insights for better crop management. These technologies assist in identifying disease risks and optimizing interventions, ultimately leading to improved crop health and yields.
- Potato leaf disease prediction systems leverage machine learning, geospatial data, and real-time monitoring to provide farmers with valuable insights for better crop management. These technologies assist in identifying disease risks and optimizing interventions, ultimately leading to improved crop health and yields.
- Potato leaf disease prediction systems leverage machine learning, geospatial data, and real-time monitoring to provide farmers with valuable insights for better crop management. These technologies assist in identifying disease risks and optimizing interventions, ultimately leading to improved crop health and yields.

REFERENCES

[1] Singh, D., Singh, R., & Khamparia, A. (2020). Plant disease detection using machine learning: A review. Journal of King Saud University-Computer and Information Sciences, 32(3), 307-319.

[2] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. Computational Intelligence and Neuroscience, 2016.

[3] Ghosal, S., Saha, S., & Nasipuri, M. (2018). A review on plant leaf disease detection techniques using image processing and machine learning approaches. Current Plant Biology, 13, 8-19.

[4] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.

[5] Zhang, X., Zhou, X., Lin, W., Zhang, Y., Ma, Y., Zhang, Z., & Zhou, J. (2018). Deep learning-based classification of hyperspectral data for plant disease detection. Remote Sensing, 10(2), 279.