M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur – 639 113.

A Project Report

on

SMART FARMING USING PYTON

Submitted in partial fulfillment of requirements for the award of the course of

AMI1201 – DATA EXPLORATION AND VISUALIZATION

Under the guidance of

Ms. S. SUBHA SRI M.E.,

IBM Corporate Trainer/AIML

Submitted By

ABHI YAMINI P(927623BAM001)
DHANUSH M(927623BAM011)
HARISH M(927623BAM021)
ROHAN M(927623BAM301)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous
KARUR – 639 113

DECEMBER 2024

M.KUMARASAMY COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur – 639 113.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **"SMART FARMING USING PYTHON "** is Bonafide work of **ABHI YAMINI P(927623BAM001),DHANUSH M(927623BAM011),HARISH 927623BAM021), ROHAN M(927623BAM301)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature

Mrs. S. SUBHA SRI M.E.

SUPERVISOR,

Department of Artificial Intelligence,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

Signature

Dr. A.SELVI (Ph.D.,)

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To To become a renowned hub for AIML technologies to producing highly talented globally recognizable technocrats to meet industrial needs and societal expectation.

MISSION OF THE DEPARTMENT

M1: To impart advanced education in AI and Machine Learning, built upon a foundation in Computer Science and Engineering.

M2: To foster Experiential learning equips students with engineering skills to tackle real-world problems.

M3: To promote collaborative innovation in AI, machine learning, and related research and

development with industries.

M4: To provide an enjoyable environment for pursuing excellence while upholding strong personal and professional values and ethics.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Excel in technical abilities to build intelligent systems in the fields of AI & ML in order to find new opportunities.

PEO 2: Embrace new technology to solve real-world problems, whether alone or as a team, while prioritizing ethics and societal benefits.

PEO 3: Accept lifelong learning to expand future opportunities in research and product development.

PROGRAM OUTCOMES

Engineering students will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Expertise in tailoring ML algorithms and models to excel in designated applications and fields.

PSO2: Ability to conduct research, contributing to machine learning advancements and innovations that tackle emerging societal challenges

ABSTRACT

Smart Farming Using Python is an innovative project aimed at improving agricultural decision-making through data visualization and exploration. This web application is designed to help farmers predict crop yield by analyzing environmental parameters such as temperature, humidity, weather, and seasonal data .The user provides specific input values, including crop name and environmental conditions, while the system processes these inputs against a preloaded dataset. The application uses Python's Flask framework for backend processing and incorporates data analysis and visualization libraries such as Pandas and Matplotlib.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
Smart Farming Using Python is an innovative project aimed at improving agricultural decision-making through data visualization and exploration. This web application is designed to help farmers predict crop yield by analyzing environmental parameters such as temperature, humidity, weather, and seasonal data .The user provides specific input values, including crop name and environmental conditions, while the system processes these inputs against a preloaded dataset.	PO1(3) PO2(3) PO3(3) PO4(2) PO5(3) PO6(2) PO7(2) PO8(1) PO9(2) PO10(2) PO10(3) PO11(2) PO12(2)	PSO1(1) PSO2(2)

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

Vii

TABLE OF CONTENTS

1. Introduction

- **Project Objective**

The objective of this project is to develop a web-based application for farmers and agricultural analysts to visualize and explore crop yield predictions using various environmental and crop-specific factors. This application leverages data visualization techniques to provide insights into the relationship between temperature, humidity, weather conditions, seasonal changes, and crop yield. By doing so, it empowers users to make data-driven decisions to optimize farming practices and improve agricultural productivity. The project aims to promote sustainable agriculture by helping farmers reduce resource wastage and adapt to changing environmental conditions. Designed with scalability in mind, the application can be enhanced in the future with features like real-time weather data, IoT sensor integration, and machine learning models, making it a versatile solution for modern farming challenges.

- **Technologies Used**

Frontend (User Interface):

HTML5

CSS3

Backend (Server-Side Logic):

Python:

Flask Framework

Data Analysis and Visualization:

Pandas

Matplotlib

Dataset Handling:

CSV Format

Integrated Development Environment (IDE):

Visual Studio Code (VS Code)

2. Project Requirements

○ Input Data Description

The input data for the Smart Farming Web Application includes Temperature (°C)**,** Humidity (%)**,** Weather (e.g., Sunny, Rainy), Season (e.g., Summer, Monsoon), and Crop Name (e.g., Rice, Wheat). Users provide these details, and the application filters the dataset to predict crop yield based on similar conditions. Additionally, users can select a visualization type Line Plot**,** Bar Chart, or Scatter Plot to analyze and understand the impact of environmental factors on crop yield effectively.

○ Expected Output

The expected output of the Smart Farming Web Application is a dynamic visualization (Line Plot, Bar Chart, or Scatter Plot) that displays the predicted crop yield based on the user's input parameters (temperature, humidity, weather, season, and crop name). If no exact matches are found, a general visualization of crop yield trends is provided using the dataset. The graph

includes labeled axes and a title summarizing the conditions, helping users analyze the relationship between environmental factors and crop yield effectively.

3. Design and Architecture

○ Frontend Overview

The frontend of the Smart Farming Web Application is a user-friendly interface built with HTML5**,** CSS3**,** and optionally Bootstrap**,** ensuring responsiveness across devices. It includes input fields for parameters like temperature, humidity, weather, season, crop name, and a dropdown for selecting visualization types (Line Plot, Bar Chart, Scatter Plot). Designed for simplicity, it enables seamless interaction, dynamically displaying the output graph for easy analysis of crop yield trends.

○ Backend Overview

The backend of the Smart Farming Web Application is powered by Flask**,** a lightweight Python web framework. It handles user inputs, processes data using Pandas for filtering and analysis, and generates visualizations with Matplotlib based on the input parameters. The backend routes requests between the frontend and the dataset, ensuring dynamic responses and generating graphs as per the selected visualization type. It is designed to be modular, scalable, and efficient, providing seamless integration between the user interface and data processing.

- ### Interaction Flow

The interaction flow begins with the user entering input values (temperature, humidity, weather, season, crop name, and visualization type) in the web form. The data is sent to the backend, where it is processed using Pandas and the relevant dataset is filtered. Based on the selected visualization type, the backend generates a graph using Matplotlib**,** which is then displayed on the frontend for the user to analyze. If no exact data matches, a fallback visualization is shown, allowing users to gain insights into crop yield trends.

4. Implementation

- ### Data Processing Workflow

The data processing workflow begins with the user entering parameters (temperature, humidity, weather, season, crop name, and visualization type). The backend filters the dataset using Pandas based on these inputs, analyzes the relevant data, and generates a graph (Line Plot, Bar Chart, or Scatter Plot) with Matplotlib**.** The graph is then encoded and sent back to the frontend, where it is displayed to the user, visualizing the relationship between environmental factors and crop yield.

- ### Chart Generation Techniques

Chart generation in the Smart Farming Web Application uses Matplotlib to create high-quality visualizations. Line plots show trends over time, bar charts compare crop yields across categories, and scatter plots highlight relationships between variables like temperature and yield. Custom styling options in

Matplotlib enhance the charts with titles, labels, colors , and legends for better clarity and presentation. These techniques ensure clear, insightful visualizations to help users analyze crop yield trends based on environmental factors.

Key Code Snippets

Python:

```
from flask import Flask, render_template, request
import pandas as pd
import matplotlib.pyplot as plt
import io
import base64

app = Flask(_name_)

# Load the dataset (replace with your actual file path)
data = pd.read_csv('data/crop_data.csv')

# Ensure correct data types (if needed)
data['Temperature'] = pd.to_numeric(data['Temperature'], errors='coerce')
data['Humidity'] = pd.to_numeric(data['Humidity'], errors='coerce')
data['Yield'] = pd.to_numeric(data['Yield'], errors='coerce')

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

```
@app.route('/predict', methods=['POST'])
def predict():
    # Get user input from the form
    temp = float(request.form['temperature'])
    humidity = float(request.form['humidity'])
    weather = request.form['weather']
    season = request.form['season']
    crop = request.form['crop_name']
    visualization_type = request.form['visualization']

    # Call the function to predict crop yield and visualize
    plot_url = visualize_crop_yield(temp, humidity, weather, season, crop, visualization_type)

    return render_template('result.html', plot_url=plot_url)

def visualize_crop_yield(temp, humidity, weather, season, crop, visualization_type):
    # Filter dataset based on input values
    filtered_data = data[
        (data['Temperature'].between(temp-5, temp+5)) &  # Temperature within +-5°C
        (data['Humidity'].between(humidity-10, humidity+10)) &  # Humidity within +-10%
        (data['Weather'].str.contains(weather, case=False)) &
        (data['Season'].str.contains(season, case=False)) &
        (data['Crop'].str.contains(crop, case=False))
```

```
    ]

    # If no data matches, generate a plot with default data or suggest the user adjust inputs
    if filtered_data.empty:
        filtered_data = data  # Use all data for a general visualization
        message = "No exact matches found. Displaying general crop yield data."
    else:
        message = "Displaying data based on your inputs."

    # Create the plot based on the selected type
    plt.figure()

    if visualization_type == 'line':
        plt.plot(filtered_data['Temperature'], filtered_data['Yield'], marker='o', label='Temperature vs Yield')
        plt.title(f'Yield Prediction for {crop} during {season}')
        plt.xlabel('Temperature (°C)')
        plt.ylabel('Yield (kg/hectare)')

    elif visualization_type == 'bar':
        plt.bar(filtered_data['Temperature'], filtered_data['Yield'], label='Temperature vs Yield')
        plt.title(f'Yield Prediction for {crop} during {season}')
        plt.xlabel('Temperature (°C)')
        plt.ylabel('Yield (kg/hectare)')
```

```
    elif visualization_type == 'scatter':
        plt.scatter(filtered_data['Temperature'], filtered_data['Yield'],
label='Temperature vs Yield')
        plt.title(f'Yield Prediction for {crop} during {season}')
        plt.xlabel('Temperature (°C)')
        plt.ylabel('Yield (kg/hectare)')

    # Save plot to a string buffer
    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)
    image = base64.b64encode(buffer.getvalue()).decode('utf-8')

    return f'{image}'

if _name_ == '_main_':
    app.run(debug=True)
import pandas as pd

def load_dataset():
    return pd.read_csv('D://smart_farming_app//data//crop_data.csv')

def predict_yield(temp, humidity, weather, season, crop):
    data = load_dataset()
    # Filter the data based on input and calculate crop yield
    filtered_data = data[(data['Temperature'] == temp) &
                         (data['Humidity'] == humidity) &
                         (data['Weather'] == weather) &
```

```
                (data['Season'] == season) &
                (data['Crop'] == crop)]
    return filtered_data['Yield'].mean()  # Example: return average yield
```

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Smart Farming App</title>
    <style>
        /* Add your styling here */
        body {
            font-family: 'Arial', sans-serif;
            background-color: #e0f7fa;
            color: #333;
            margin: 0;
            padding: 0;
        }

        h1 {
            text-align: center;
            color: #00796b;
            margin-top: 40px;
        }
```

```
.form-container {
    width: 50%;
    margin: 40px auto;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.form-container label {
    font-size: 16px;
    margin-bottom: 10px;
    display: block;
    color: #00796b;
}

.form-container input,
.form-container select {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    font-size: 16px;
    border: 1px solid #00796b;
    border-radius: 4px;
    box-sizing: border-box;
}

.form-container button {
```

```
            width: 100%;
            padding: 12px;
            background-color: #00796b;
            color: white;
            border: none;
            border-radius: 4px;
            font-size: 16px;
            cursor: pointer;
        }

        .form-container button:hover {
            background-color: #004d40;
        }

        .form-container p {
            text-align: center;
            font-size: 16px;
            color: #555;
        }

        .form-container a {
            text-decoration: none;
            color: #00796b;
            font-size: 16px;
        }
    </style>
</head>
<body>
```

```
    <h1>Smart Farming - Crop Yield Prediction</h1>
    <div class="form-container">
        <form action="/predict" method="POST">
            <label for="temperature">Temperature (°C):</label>
            <input type="text" id="temperature" name="temperature" required><br>

            <label for="humidity">Humidity (%):</label>
            <input type="text" id="humidity" name="humidity" required><br>

            <label for="weather">Weather (e.g., Sunny, Rainy):</label>
            <input type="text" id="weather" name="weather" required><br>

            <label for="season">Season (e.g., Summer, Monsoon):</label>
            <input type="text" id="season" name="season" required><br>

            <label for="crop_name">Crop Name (e.g., Rice, Corn):</label>
            <input type="text" id="crop_name" name="crop_name" required><br>

            <label for="visualization">Choose Visualization Type:</label>
            <select id="visualization" name="visualization" required>
                <option value="line">Line Plot</option>
                <option value="bar">Bar Chart</option>
                <option value="scatter">Scatter Plot</option>
            </select><br>
```

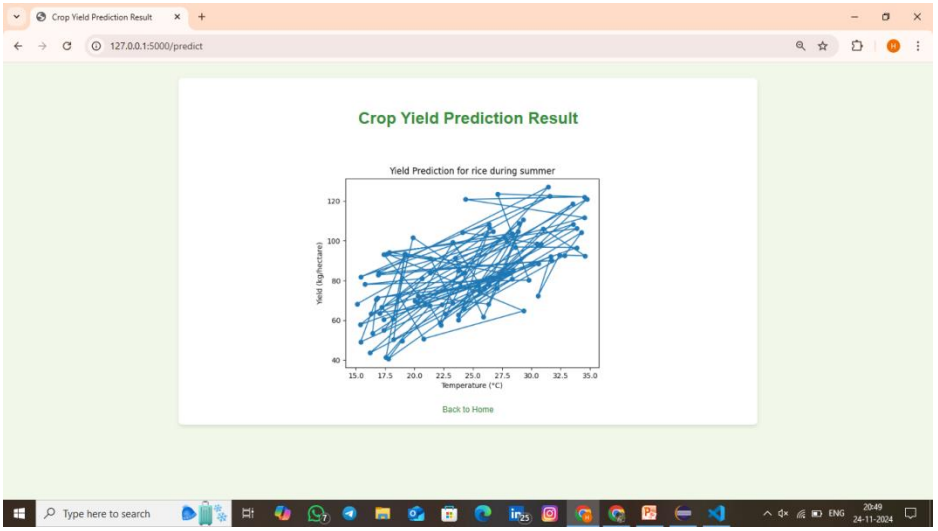
```
            <button type="submit">Predict Yield</button>
        </form>
    </div>
</body>
</html>
```

5. Data Visualizations

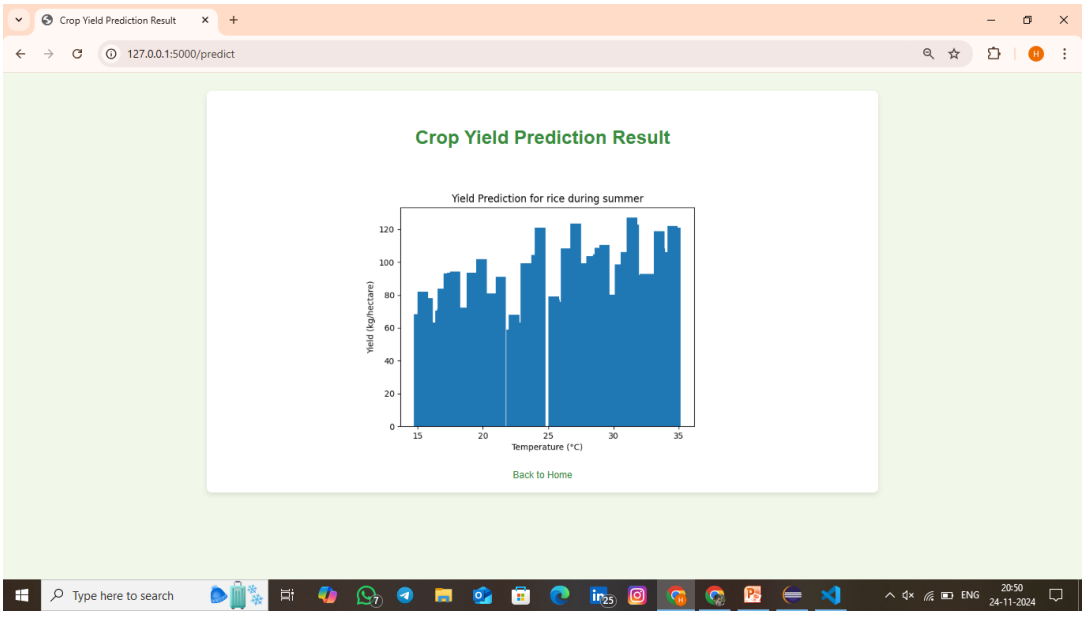
○ Types of Charts Implemented

The Smart Farming Web Application implements three types of charts: Line Plot to visualize trends over continuous variables like temperature or humidity, Bar Chart for comparing crop yields across categories such as weather or season, and Scatter Plot to show relationships between two variables like temperature and yield. These charts help users analyze the impact of environmental factors on crop yield effectively.

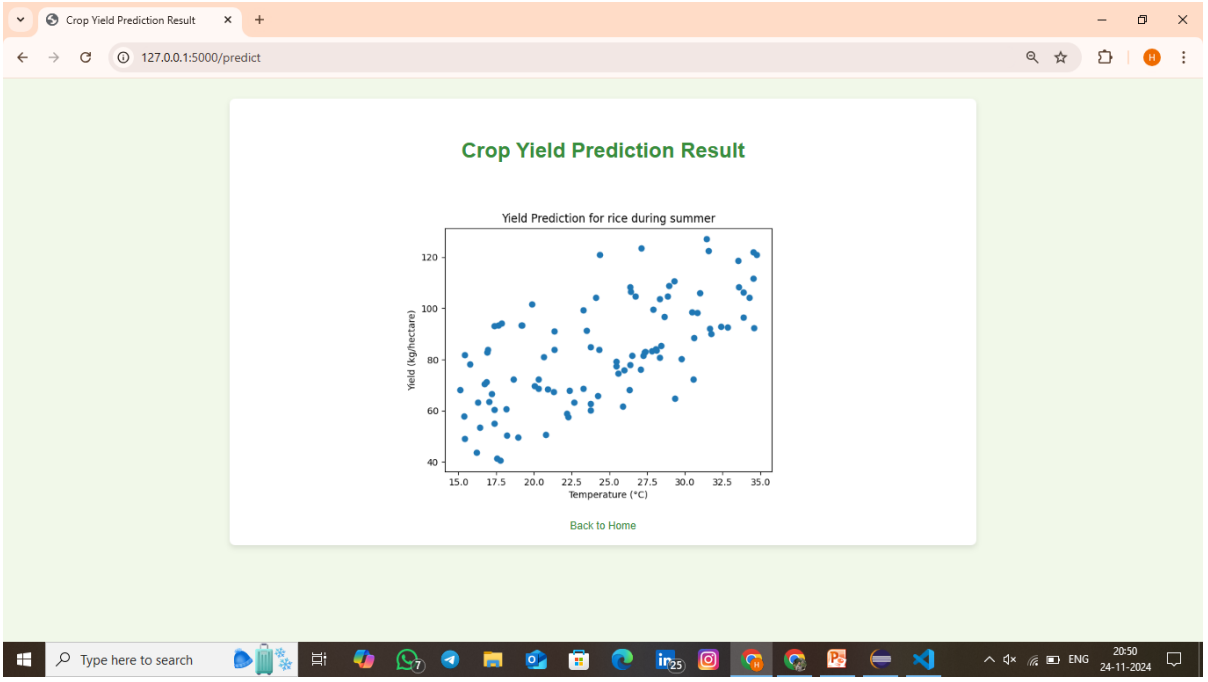
LINE CHARTS:



BAR CHARTS:



SCATTER PLOT:



- **Example Visualizations and Descriptions**

Example visualizations include a Line Plot showing how crop yield changes with temperature, a Bar Chart comparing crop yields under different weather conditions, and a Scatter Plot displaying the relationship between humidity and

yield. These charts help users analyze the impact of environmental factors on crop yield, providing valuable insights for better farming decisions.

6. Challenges and Solutions

- **Issues Encountered**

Several issues were encountered during the development of the Smart Farming Web Application. One challenge was data mismatch, where user inputs did not always align with the dataset, making it difficult to generate accurate visualizations. Data quality issues, such as missing values or outliers, required careful handling to ensure accurate predictions. Chart rendering also posed challenges, especially with large datasets or complex visualizations, necessitating optimization for faster load times. User input validation was essential to prevent errors and ensure smooth functionality. Additionally, cross-browser compatibility required testing and adjustments to ensure consistent display across devices. These issues were addressed by refining data processing, enhancing input validation, and optimizing chart rendering for a better user experience.

- **Resolution Strategies**

To resolve issues, fuzzy matching and range-based filtering were implemented to handle data mismatches. Data quality was improved by cleaning missing values and handling outliers. Chart rendering was optimized by reducing dataset size, ensuring faster load times. Input validation was enhanced with stricter checks to prevent errors, while cross-browser compatibility was ensured through extensive testing and responsive design adjustments. These strategies helped improve the application's accuracy, performance, and user experience.

7. Conclusion

Summary of Outcomes

The Smart Farming Web Application successfully provides users with a dynamic and interactive platform to visualize crop yield predictions based on environmental factors. By implementing effective data processing and visualization techniques, the app generates insightful graphs (Line Plot, Bar Chart, Scatter Plot) based on user inputs such as temperature, humidity, weather, season, and crop type. Despite challenges with data mismatches, quality issues, and rendering performance, resolution strategies like data cleaning, optimization, and input validation ensured a smooth user experience. Ultimately, the application helps users make informed decisions, improving agricultural practices by understanding how various factors impact crop yield.

Future Enhancements

Future enhancements for the Smart Farming Web Application include integrating machine learning for more accurate crop yield predictions, real-time weather data for up-to-date insights, and advanced filtering options for more interactive analysis. Additionally, a mobile app version and user authentication for personalized profiles will improve accessibility and user

experience. These improvements aim to make the application more accurate, user-friendly, and valuable for modern farming practices.

8. References

1. **Flask Documentation**
   Flask web framework documentation. Retrieved from: https://flask.palletsprojects.com/
2. **Pandas Documentation**
   Pandas data analysis library documentation. Retrieved from: https://pandas.pydata.org/
3. **Matplotlib Documentation**
   Matplotlib library for generating plots and visualizations. Retrieved from: https://matplotlib.org/
4. **Bootstrap Documentation**
   Responsive design framework documentation. Retrieved from: https://getbootstrap.com/
5. **Machine Learning for Agriculture**
   "A Survey on Machine Learning Applications in Agriculture." Retrieved from: https://www.researchgate.net/
6. **Real-time Weather APIs**
   "Weather API Documentation." Retrieved from: https://openweathermap.org/api