A MAJOR PROJECT REPORT ON

Smart Crop Prediction System

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In the department of

Computer Science and Engineering (Artificial Intelligence)

By

Karri Bhavan (21A81A4319)

Mallipudi Lakshmi Srija(21A81A4330)

Padam Harshitha (21A81A4338)

Cherukupalli Siva Teja (21A81A4311)

Maddipati Bala Roshan (21A81A4328)

Under the Esteemed Supervision of

Mr. R L PHANIKUMAR,M.Tech(Ph.D)

Department of Computer Science and Engineering (Artificial Intelligence)
SRI VASAVI ENGINEERING COLLEGE (Autonomous)
(Affiliated to JNTUK, Kakinada)
Pedatadepalli, Tadepalligudem-534101, A.P

2024-25.

SRIVASAVI ENGINEERINGCOLLEGE (Autonomous)

Department of Computer Science and Engineering (Artificial Intelligence)

Pedatadepalli, Tadepalligudem

Certificate

This is to certify that the Project Report entitled "**Smart Crop Prediction System**" submitted by **K. Bhavan Narayana(21A81A4319), M.Lakshmi Srija (21A81A4330), P. Harshita Krishna Sri (21A81A4338), Ch. Siva Teja (21A81A4311), M. Bala Roshan (21A81A4328)** for the award of the degree of Bachelor of Technology in the Department of CSE (Artificial Intelligence) and Artificial Intelligence and Machine Learning during the academic year 2024 – 2025.

Name of Project Guide

Mr. R L Phani Kumar
Asst.Professor

Head of the Department

Dr. Loshma Gunisetti
Professor & Head

External Examiner

DECLARATION

We hereby declare that the project report entitled "**Smart Crop Prediction**" submitted by us to Sri Vasavi Engineering College(Autonomous), Tadepalligudem, affiliated to JNTUK Kakinada in partial fulfilment of the requirement for the award of the degree of B.Tech in Computer Science and Engineering(Artificial Intelligence) is a record of Bonafide project work carried out by us under the guidance of **Mr. R L Phani Kumar, Asst.Professor**. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or University.

Project Associates

K.Bhavan (21A81A4319)

M.Lakshmi Srija (21A81A4330)

P.Harshita (21A81A4338)

Ch. Siva Teja (21A81A4311)

M. Bala Roshan (21A81A4328)

ACKNOWLEDGEMENT

First and foremost, we sincerely salute to our esteemed institute **Sri Vasavi Engineering College,** for giving us this golden opportunity to fulfill our warm dream to become an engineer.

Our sincere gratitude to our project guide **Mr. R L Phani Kumar, Asst.Professor**, Department of CSE (Artificial Intelligence)**,** for his timely cooperation and valuable suggestions while carrying out this project.We express our sincere thanks and heartful gratitude to **Dr. Loshma Gunisetti**, Professor & Head of the Department, Department of CSE (Artificial Intelligence) and Artificial Intelligence & Machine Learning, for permitting us to do our project.

We express our sincere thanks and heartful gratitude to **Dr. G.V.N.S.R. Ratnakara Rao**, Principal, for providing a favourable environment and supporting us during the development of this project.

Our special thanks to the management and all the teaching and non-teaching staff members, Department of CSE (Artificial Intelligence) and Artificial Intelligence & Machine Learning, for their support and cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

Project Associates

K.Bhavan (21A81A4319)

M.Lakshmi Srija (21A81A4330)

P.Harshita (21A81A4338)

Ch. Siva Teja (21A81A4311)

M. Bala Roshan (21A81A4328)

TABLE OF CONTENTS

LIST OF FIGURES

PROJECT OUTLINE

CHAPTER-1

INTRODUCTION

INTRODUCTION

1.1 Abstract

Agriculture plays a crucial role in food security and economic stability. However, unpredictable weather patterns and soil conditions make crop selection challenging for farmers. The Smart Crop Prediction system leverages machine learning to recommend the most suitable crops based on environmental factors such as temperature, rainfall, soil pH, and humidity. By integrating a trained ML model with a MERN stack application, this system provides data-driven insights to optimize agricultural productivity.

The system continuously learns from past data, improving prediction accuracy over time. Real-time weather updates and soil analysis enhance decision-making for farmers. A user- friendly interface ensures accessibility, even for those with minimal technical expertise. IoT-based sensors can be integrated to provide real-time soil and weather data. The system also offers market insights, helping farmers choose crops with higher profitability. Future enhancements include multilingual support, smart irrigation recommendations, and blockchain integration for data security, making the system a comprehensive solution for modern agriculture.

1.2 Introduction

The Smart Crop Prediction system has a wide scope in enhancing agricultural productivity through data-driven decision-making. By integrating machine learning, real-time weather data, and soil nutrient analysis, the system helps farmers choose the most suitable crops based on their location, climate, and soil conditions. The solution is scalable and adaptable, meaning it can be deployed across various regions with different soil types and weather conditions. Additionally, by storing prediction results in a MongoDB database, the system can be used for long-term agricultural analysis, helping policymakers and researchers track crop trends, climate impacts, and soil fertility changes over time.

To enhance prediction accuracy, we utilize a Random Forest classifier, which has been trained on a well-structured dataset containing crop yield patterns under different climatic and soil conditions. Additionally, the system fetches live weather data using an external API, allowing it to adapt to changing environmental conditions. The integration of MongoDB enables us to store past predictions, which can help in future analysis and improvements. This approach ensures that the system is data-driven, scalable, and user- friendly, making it accessible to farmers through a simple web interface.

Our project aims to bridge the gap between technology and agriculture by providing real- time, data-backed recommendations that help farmers make informed decisions. With further improvements, such as incorporating remote sensing data, satellite imagery, and deep learning models, the system can become even more robust. By leveraging AI-driven agriculture, this project contributes to sustainable farming, better resource management, and improved food security, ultimately empowering farmers with smart, efficient, and technology-driven decision-making tools.

1.3 Motivation

Agriculture is the backbone of many economies, yet farmers often struggle with unpredictable climate conditions, soil fertility issues, and inefficient crop selection. Traditional farming methods rely on experience and intuition, which may not always lead to optimal results, especially in the face of climate change and evolving soil conditions. The lack of access to scientific data leaves many farmers vulnerable to poor yields, financial losses, and food insecurity. This motivated us to develop a Smart Crop Prediction system that leverages machine learning and real-time weather data to provide accurate and data-driven crop recommendations. By integrating technology with agriculture, we aim to empower farmers to make smarter, more efficient decisions that lead to improved productivity and sustainability.

The rapid advancements in Artificial Intelligence (AI) and data science have provided new opportunities to revolutionize agriculture. By utilizing a trained ML model, weather APIs, and soil data analysis, our system offers precise crop predictions tailored to a farmer's location and soil conditions. The motivation behind this project is to reduce guesswork, enhance agricultural efficiency, and promote sustainable farming practices. Through automated crop prediction, farmers can make informed decisions, leading to higher yields, better resource management, and a positive impact on food security. This project ultimately seeks to bridge the gap between traditional farming and modern AI-driven solutions, making precision agriculture accessible to all.

1.4 Scope

The Smart Crop Prediction system has a wide scope in enhancing agricultural productivity through data-driven decision-making. By integrating machine learning, real-time weather data, and soil nutrient analysis, the system helps farmers choose the most suitable crops based on their location, climate, and soil conditions. The solution is scalable and adaptable, meaning it can be deployed across various regions with different soil types and weather conditions. Additionally, by storing prediction results in a MongoDB database, the system can be used for long-term agricultural analysis, helping policymakers and researchers track crop trends, climate impacts, and soil fertility changes over time.

Beyond immediate crop recommendations, the system can be expanded to include fertilizer suggestions, pest detection, and yield forecasting, making it a comprehensive precision agriculture tool. With further development, it can support multiple languages to assist farmers in different regions and even integrate with mobile applications for real- time accessibility. As AI and machine learning continue to advance, the system has the potential to evolve into a fully automated agricultural assistant, improving food security, reducing resource wastage, and increasing farm profitability on a global scale.

CHAPTER – 2

LITERATURE SURVEY

2. LITERATURE SURVEY

1. **Title:** Crop Prediction Using Machine Learning

   Author(s): Madhuri Shripathi Rao, Arushi Singh, N.V. Subba Reddy, Dinesh U Acharya

   Published Year:2022

   Description: This This study focuses on the application of machine learning models for predicting the most suitable crops based on environmental conditions such as soil type, temperature, and rainfall. The research compares three widely used classification models: K-Nearest Neighbors (KNN), Decision Tree, and Random Forest. The experimental results show that the Random Forest model achieves the highest accuracy of 99.32%, outperforming the other models in terms of predictive performance. The study highlights that Random Forest is more effective in capturing complex relationships between different agricultural factors, making it the best-suited algorithm for crop recommendation systems. This research provides valuable insights into selecting optimal machine learning models for precision agriculture.

2. **Title:** A Crop Prediction Model Using Machine Learning Algorithms

   Author(s): Ersin Elbasi, Chamseddine Zaki, Ahmet E. Topcu, Wiem Abdelbaki, Aymen I. Zreikat, Elda Cina, Ahmed Shdefat, Louai Saker.

   Published Year:2023

   Description: This paper presents an advanced crop prediction model that integrates 15 different machine learning algorithms to determine the most suitable crops based on environmental and climatic conditions.A key aspect of study is the incorporation of Internet of Things (IoT) sensor data, which provides real-time soil and weather updates to enhance model accuracy. Among tested models, Bayes Net achieves the highest classification accuracy of 99.59%, followed by Naïve Bayes with an accuracy of 99.46%. The findings suggest that machine learning, when combined with IoT sensor data, significantly improves crop selection, leading to optimized agricultural productivity. The study emphasizes the importance of real-time monitoring and data- driven decision-making in modern farming techniques.

3. **Title:** Design and Implementation of a Cloud-Based Smart Agriculture System for Crop Yield Prediction Using a Hybrid Deep Learning Algorithm

Author(s): Avdesh Kumar Sharma, Abhishek Singh Rathore.

Published Year:2024

Description: This study introduces a cloud-based smart agriculture system that leverages deep learning techniques to enhance crop yield prediction. The proposed system utilizes a hybrid model combining Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to analyze multi-sensor data, including temperature, humidity, soil pH, and rainfall patterns. The results indicate that the hybrid deep learning model achieves over 90% accuracy in crop yield prediction. Additionally, the study reports a Mean Absolute Error (MAE) of 2.17% and a Root Mean Square Error (RMSE) of 2.94%, demonstrating the model's effectiveness in reducing prediction errors. By implementing a cloud-based infrastructure, the system enables real-time analysis and decision-making for farmers, thereby improving agricultural efficiency and sustainability. This research highlights the potential of integrating deep learning with cloud computing for precision agriculture and smart farming solutions.

CHAPTER – 3

SYSTEM STUDY AND ANALYSIS

3. SYSTEM STUDY AND ANALYSIS

3.1.0 Problem Statement

The problem in modern agriculture is the lack of accurate and data-driven crop selection, leading to low yields, soil degradation, and inefficient resource utilization. Farmers often rely on traditional knowledge rather than scientific methods, making it difficult to adapt to changing climatic conditions and soil variations. Additionally, factors such as unpredictable weather, improper fertilizer use, and declining soil fertility further impact productivity. The Smart Crop Prediction system aims to solve this issue by leveraging machine learning, real- time weather data, and soil nutrient analysis to recommend the most suitable crop for a given location, ensuring higher yields, efficient resource management, and sustainable farming practices.

3.1.1 Existing System

The existing crop selection system primarily relies on traditional farming practices, historical knowledge, and generalized agricultural guidelines, which often fail to account for dynamic environmental factors such as soil nutrients, real-time weather conditions, and rainfall variations. Many farmers make decisions based on experience or outdated recommendations, leading to suboptimal crop choices, low yields, and financial losses. While some government and agricultural organizations provide basic advisory services, these are often not personalized, lack real-time data integration, and fail to leverage advanced technologies like machine learning for precise predictions. As a result, the current system is inefficient and does not maximize agricultural productivity.

3.1.2 Limitations of Existing System

1. Lack of real-time data integration for accurate crop prediction: Traditional systems use static datasets, leading to outdated and unreliable crop recommendations.
2. Dependence on historical knowledge and generalized agricultural guidelines: Farmers rely on past experiences, ignoring changing climate patterns and soil variations.
3. No personalized recommendations based on location-specific environmental factors: Existing systems fail to consider microclimate variations and specific soil conditions.
4. Inaccurate predictions leading to suboptimal crop selection and reduced yields:
5. Reliance on manual decision-making, increasing the risk of financial losses for farmers: Poor crop recommendations lead to wasted resources, financial losses, and low productivity,
6. Computational Complexity: Farmers depend on judgment and advice rather than data-driven insights.
7. Limited Integration with External Data Sources: Farmers lack insights into future crop prices, affecting profitability. Many systems lack user-friendly interfaces and regional language options, limiting adoption.

3.1.3 Proposed System

1. Data Integration:

- Retrieve and Uses APIs to fetch live weather data, ensuring up-to-date environmental conditions are factored into predictions. This helps farmers make timely and informed decisions based on current weather trends.

2. Machine Learning-based Predictions:

- Analyzes both historical and real-time data using ML algorithms to recommend the best-suited crops. This enhances prediction accuracy and adapts to changing agricultural conditions.

3. User-Friendly Interface :

- A React-based frontend provides a simple, interactive platform for farmers to input data and view recommendations. The interface is designed for ease of use, even for users with minimal technical knowledge.Model Selection and Training:

4. Automated Decision Support:

- The system automates crop selection decisions, reducing reliance on manual expertise. By minimizing guesswork, it helps farmers avoid financial losses and optimize their resources efficiently.

5. Deployment and Future Enhancements:

- Deploythe model as a web-based or mobile application for user accessibility.
- Using machine learning to predict crop prices, helping farmers make informed planting decisions based on expected profitability.

3.1.4 Advantages of Proposed System

1. Improved Prediction Accuracy:

- Uses machine learning algorithms to provide precise crop recommendation and statistical models.

2. Personalized Predictions:

- Considers real-time weather data and soil conditions for location-specific suggestions.

3. Higher Yield & Profitability:

- Automates decision-making, reducing reliance on traditional experience-based farming.

4. Efficient Resource Utilization:

- Optimizes water, fertilizers, and other agricultural inputs by selecting crops that best fit current conditions.

5. Scalability & Accessibility:

- Cloud-based deployment allows farmers to access recommendations from anywhere visually.

6. User-Friendly Interface::

- Simple and interactive design for ease of use by farmers and agricultural experts.

7. **Market-Driven Crop Selection:**

- Integrates market price predictions to help farmers choose crops with higher demand and profitability.

8. **Climate Resilience and Sustainability:**

- Recommends climate-resilient crops and promotes sustainable farming practices by adapting to changing environmental conditions.

3.1.5 Functional Requirements

- User can view current stock data which is plotted in a graph for different tickers.
- User can also predict stock for their choice of ticker for no. of days they want to predict the stock price.

3.1.6 Non-Functional Requirements

- Easy to use
- Available
- Scalable
- Platform Independent

3.1.7 User Interface Requirements

3.1.7.1 System Requirements

3.7.1.1.1 Hardware Requirements

Processor Support: i5 onwards

RAM: Minimum 8GB

Hard Disk: Minimum 4GB

3.7.1.1.2 Software Requirements

Platform: Visual studio code

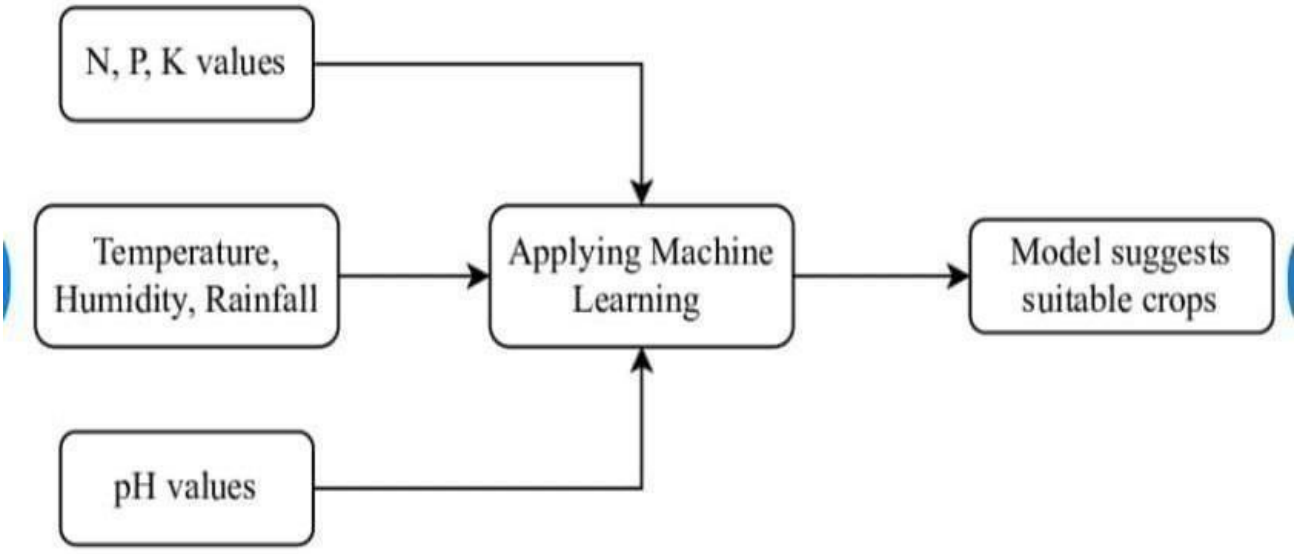
Language: Python version – 3.7 above Framework: Flask (LTS Support)

CHAPTER – 4

SYSTEM DESIGN

4. SYSTEM DESIGN

SYSTEM ARCHITECTURE



4.1 Collection

The process starts with gathering agricultural data from reliable sources, including government records, weather APIs, and soil health reports. This data includes soil nutrients (N, P, K, pH), weather conditions (temperature, rainfall, humidity), and past crop yield records, serving as the foundation for analysis and prediction.

4.2 Data Preprocessing

Raw agricultural data may contain missing values, inconsistencies, or outliers that could affect model performance. In this phase, data is cleaned, normalized, and transformed to ensure accuracy. Handling missing values and outliers improves data reliability, leading to better crop predictions.

4.3 Feature Selection

Feature selection involves identifying the most critical factors influencing crop growth and yield. By removing redundant or irrelevant attributes, this step enhances model efficiency and prevents overfitting. Relevant features include soil nutrients, weather parameters, and historical crop yields.

4.4 Training and Testing Data Split

The dataset is divided into training and testing sets, where the training data helps the model learn patterns and relationships. The testing data evaluates the model's performance on unseen inputs, ensuring accurate and reliable predictions for real-world application.

4.5 Testing the model

After training, the model is tested on unseen data to measure its accuracy in recommending suitable crops. The test phase helps analyze the model's ability to generalize predictions based on environmental factors, ensuring reliable suggestions for farmers.

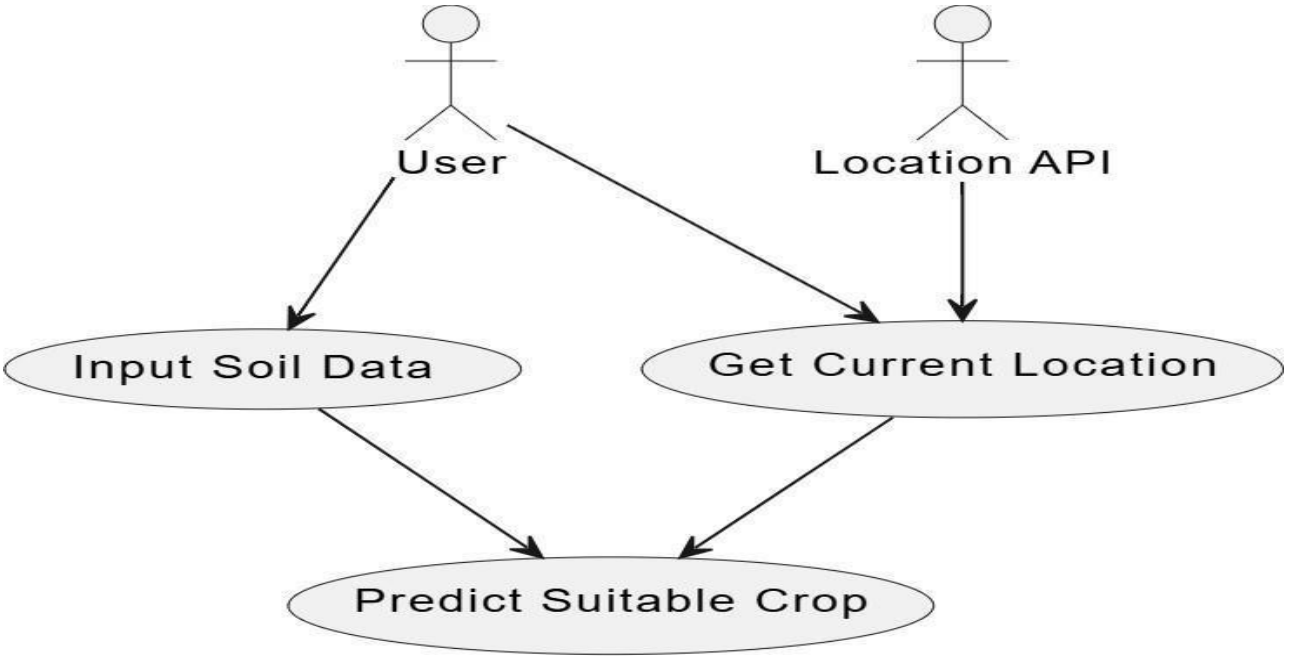
4.6 Evaluate Prediction Accuracy

The final step is evaluating the model's accuracy using performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared (R^2). This step helps determine how well the model performs in real-world scenarios and whether further improvements are needed.

UML DIAGRAMS:

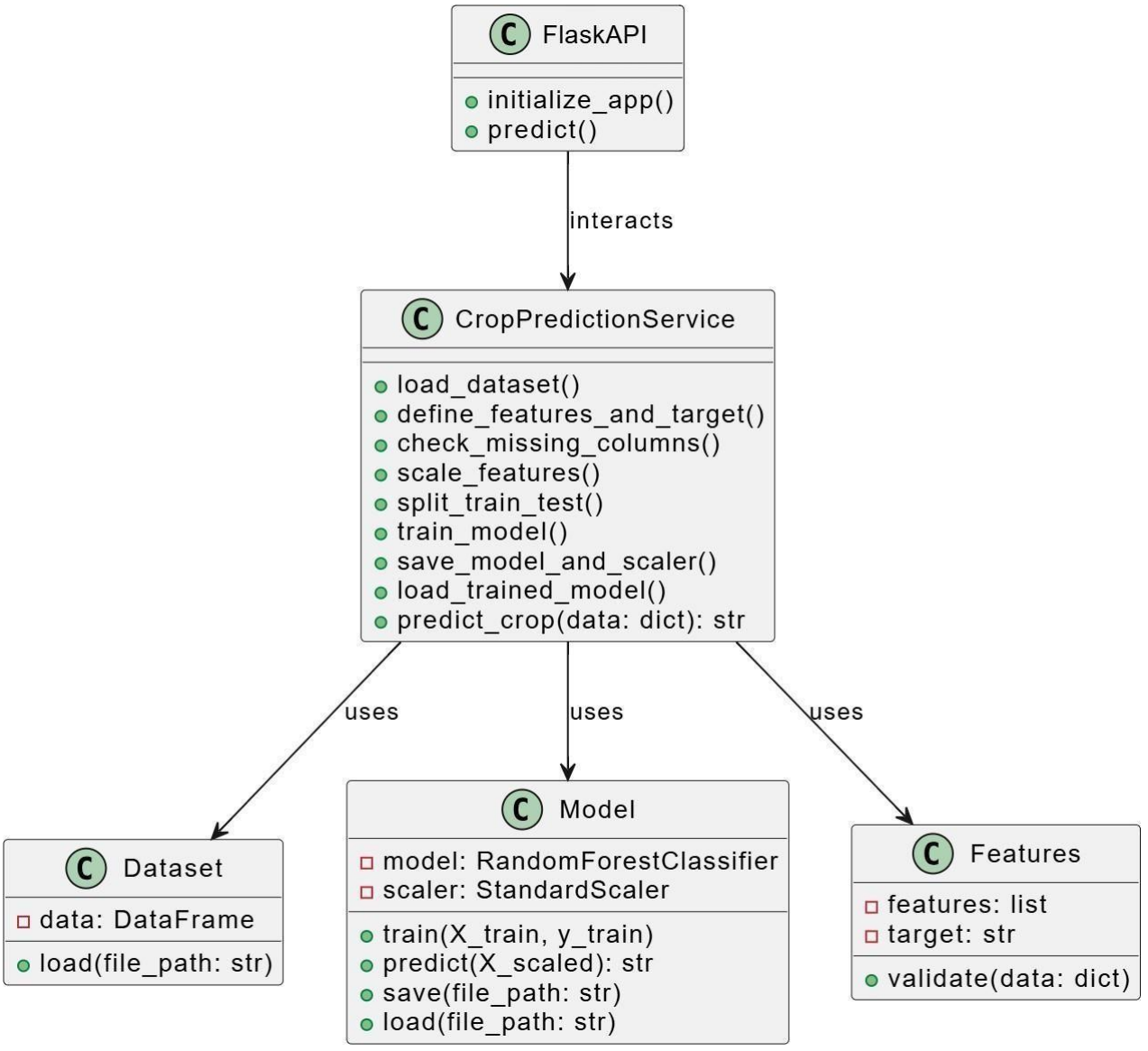
Use case Diagram:

The use case diagram illustrates the **Smart Crop Prediction System**, where a **User** inputs soil parameters (**Nitrogen, Phosphorus, Potassium, pH**) while the **Location API** fetches the current location to determine environmental factors. These inputs are processed by a **Machine Learning model (Random Forest Classifier)** to predict the most **suitable crop** for the given soil and climatic conditions, providing a data-driven recommendation to the user.



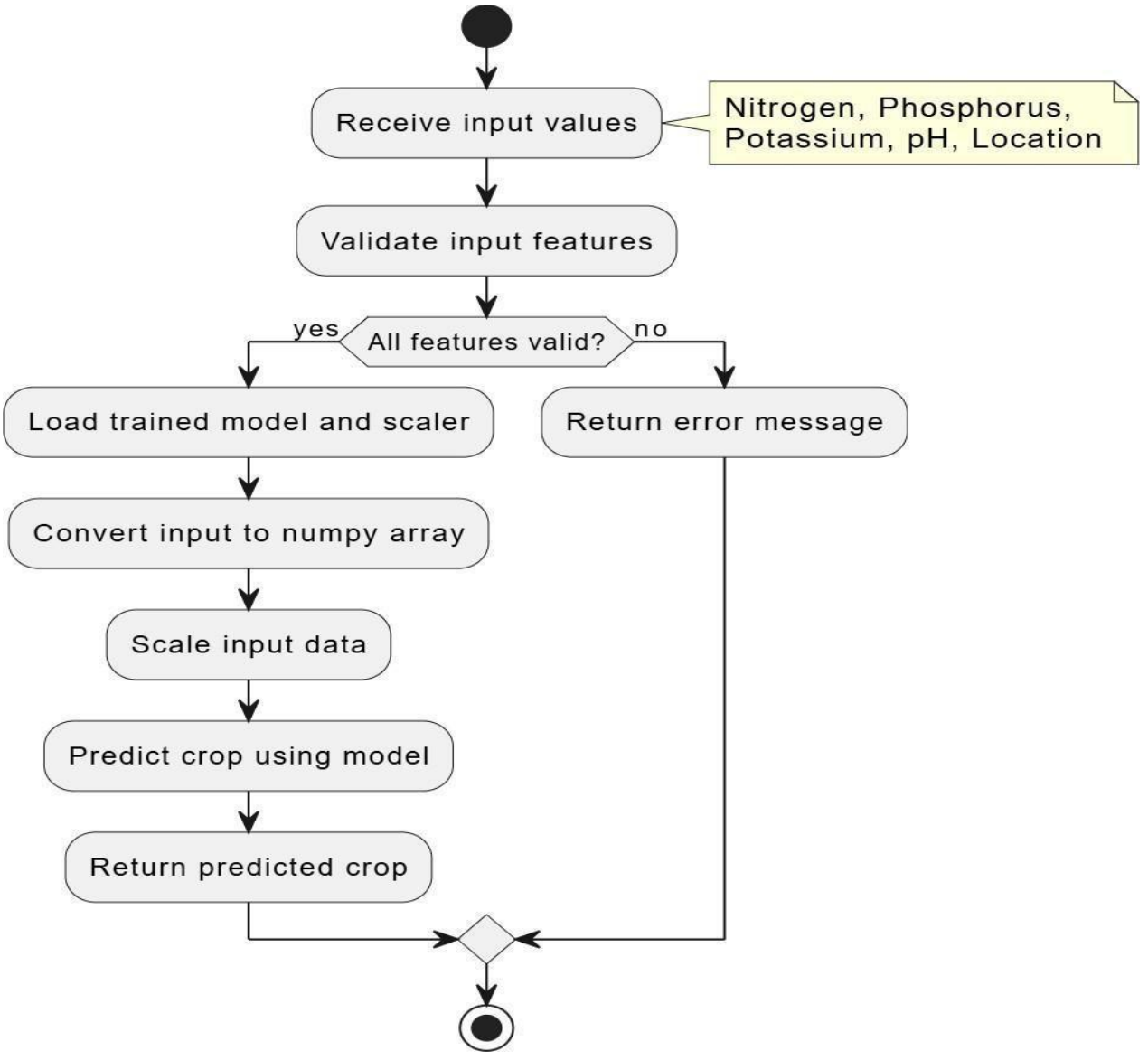
Class Diagram:

The **class diagram** showcases the **Crop Prediction System**, where **FlaskAPI** manages the web interface and interacts with **CropPredictionService**, the core logic handling data loading, preprocessing, model training, and predictions. This service utilizes **Dataset** (to load and store crop data), **Model** (which holds the **RandomForestClassifier** and **StandardScaler** for training and predictions), and **Features** (defining input variables and validating user data), ensuring a modular and efficient machine learning workflow.


FlaskAPI
initialize_app()
predict()
interacts
CropPredictionService
load_dataset()
define_features_and_target()
check_missing_columns()
scale_features()
split_train_test()
train_model()
save_model_and_scaler()
load_trained_model()
predict_crop(data: dict): str
uses
uses
uses
Dataset
data: DataFrame
load(file_path: str)
Model
model: RandomForestClassifier
scaler: StandardScaler
train(X_train, y_train)
predict(X_scaled): str
save(file_path: str)
load(file_path: str)
Features
features: list
target: str
validate(data: dict)

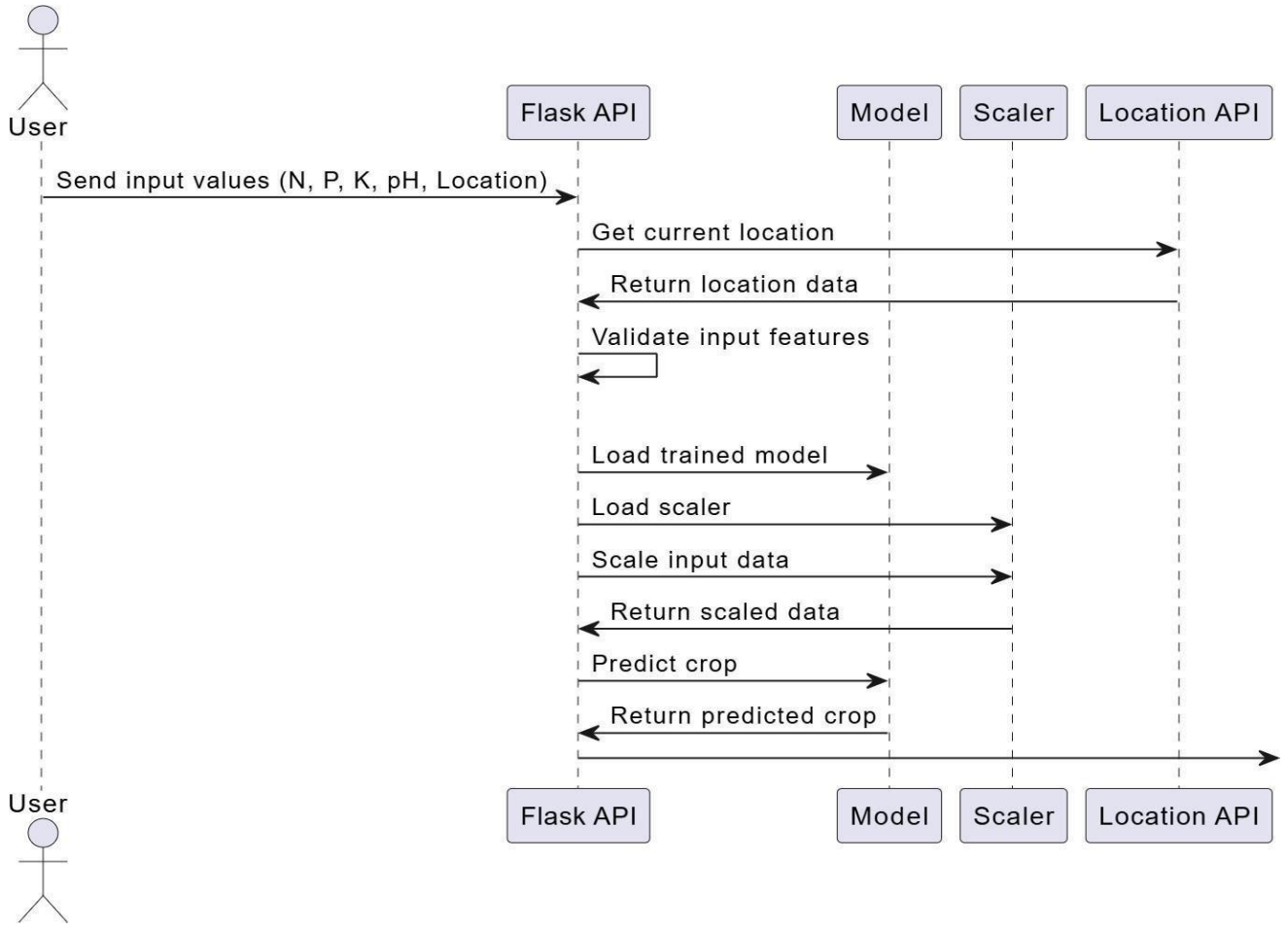
Activity Diagram:

This **flowchart** represents the **crop prediction process**, starting with receiving input values such as **Nitrogen, Phosphorus, Potassium, pH, and Location**. The system validates these input features; if invalid, it returns an error message. If valid, it loads a **trained model and scaler**, converts the input into a **NumPy array**, scales the data, and uses the model to predict the suitable crop. Finally, the system returns the predicted crop as outpu



Sequence Diagram:

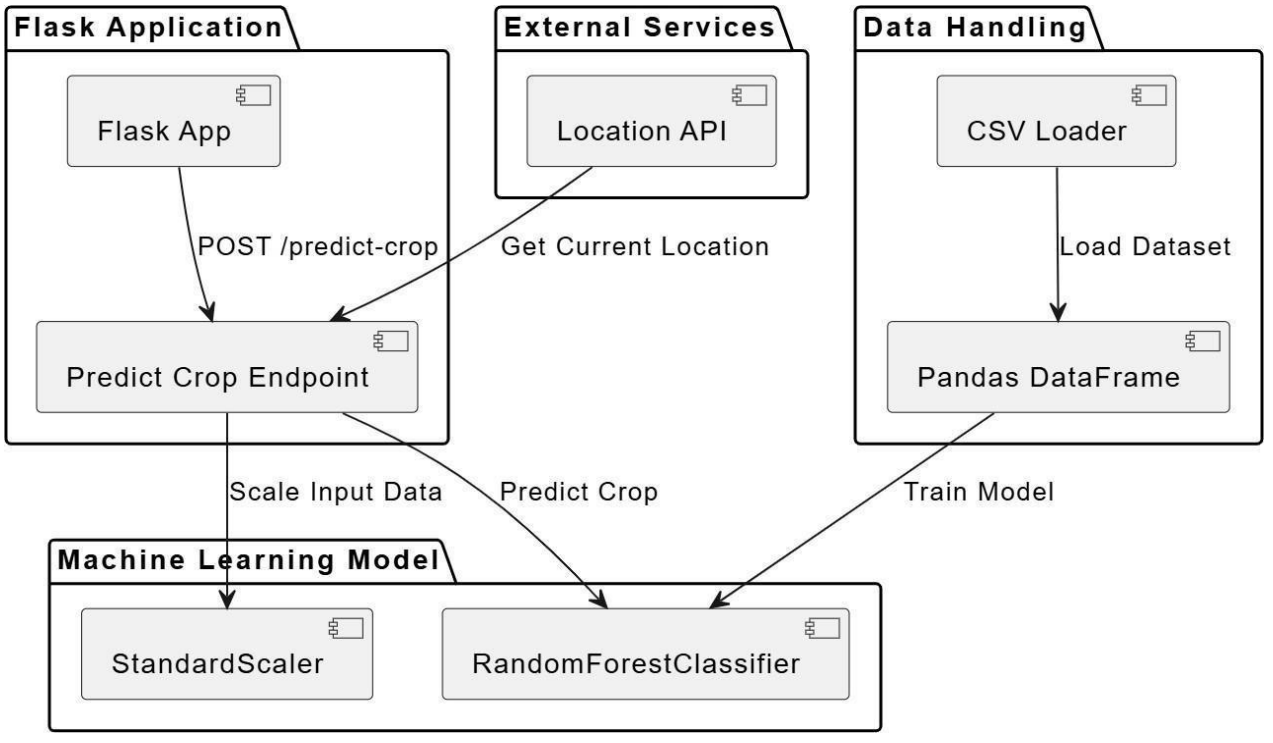
This sequence diagram represents a crop prediction system using a Flask API. The user inputs values such as nitrogen (N), phosphorus (P), potassium (K), pH level, and location. The Flask API retrieves the user's location using a Location API and validates the input features. It then loads a trained machine learning model and a scaler. The input data is scaled using the scaler, and the scaled data is fed into the model for crop prediction. The predicted crop is then returned to the user via the Flask API. This interaction ensures an automated recommendation system based on soil



nutrients and environmental factors.

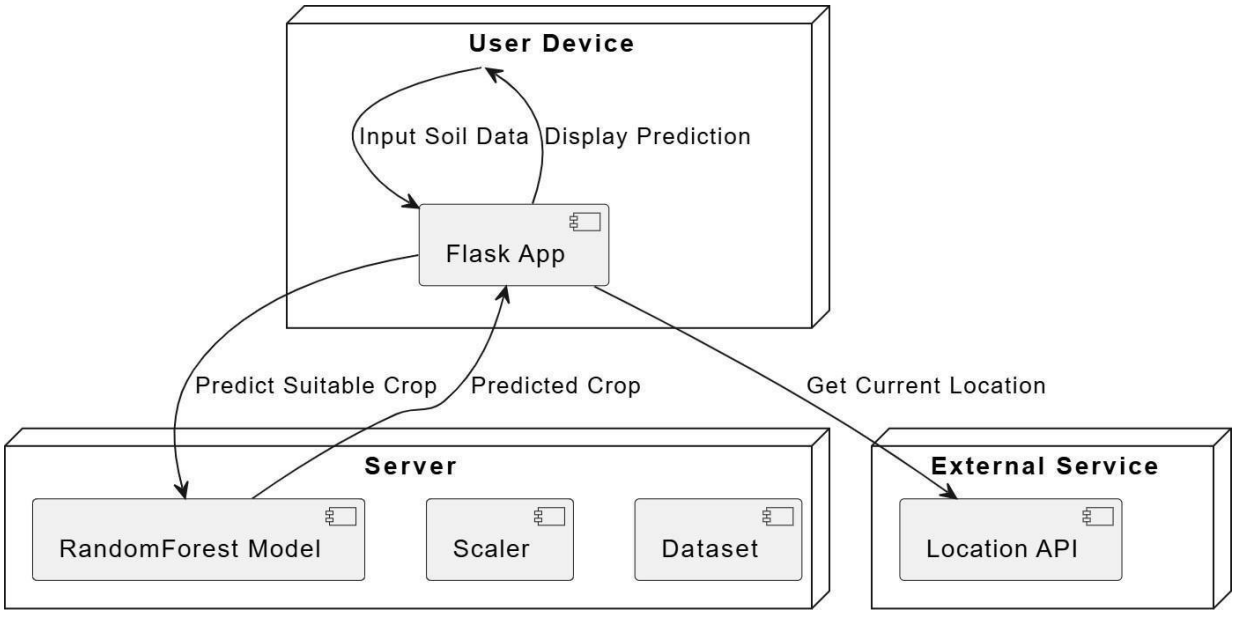
Component Diagram:

This diagram represents the architecture of a crop prediction system built using Flask, machine learning, and external services. The **Flask Application** consists of a Flask app that provides a **Predict Crop Endpoint**, which receives input data via a POST request. It interacts with **External Services** like the **Location API** to get the current location. The **Data Handling** module includes a **CSV Loader** that loads datasets into a **Pandas DataFrame** for processing and training. The **Machine Learning Model** consists of a **StandardScaler** for scaling input data and a **RandomForestClassifier** for making crop predictions. The system ensures that input data is properly processed, scaled, and used for accurate predictions based on soil and environmental conditions.



Deployment Diagram:

This diagram illustrates the architecture of a crop prediction system using a Flask-based web application. The **User Device** interacts with the **Flask App** by providing soil data input and receiving crop predictions. The **Flask App** communicates with an **External Service** (Location API) to fetch the current location. It also connects to a **Server**, which consists of a **RandomForest Model** for crop prediction, a **Scaler** for normalizing input data, and a **Dataset** for training and reference. The Flask app processes user inputs, scales the data, sends it to the machine learning model for prediction, and returns the most suitable crop recommendation to the user.



CHAPTER – 5

TECHNOLOGIES

5. Technologies

5.1 FrontEnd-ReactJS

Technology: React.js (JavaScript Library)

Purpose: Provides an interactive and dynamic user

Features:

- Responsive design
- Interactive dashboards
- User-friendly forms for inputting location and weather data
- Data visualization for predictions

5.2 Backend – Node.js & Express.js

Technology: Node.js (JavaScript Runtime) with Express.js (Framework)

Purpose: Handles API requests, processes data, and communicates with the ML model.

Features:

- RESTful API for handling crop prediction requests
- Secure authentication and authorization
- Middleware for request validation

5.3 Database – MongoDB

Technology: MongoDB (NoSQL Database)

Purpose: Stores user data, past predictions, and crop information.

Features:

- Flexible schema for storing different crop datasets
- Fast retrieval and storage of prediction results

5.4 Machine Learning – Python

Technology: Python-based ML model (Scikit-Learn, TensorFlow, Pandas, NumPy)

Purpose: Predicts the best crop based on weather conditions and soil parameters.

Features:

- Uses trained .pkl model for fast predictions
- Processes weather and soil data to recommend the best crop
- Can improve accuracy with more training data

5.5 Weather Data API (Optional)

Technology: OpenWeatherMap / WeatherStack API

Purpose: Fetches real-time weather data for accurate predictions.

Features:

- Provides temperature, humidity, rainfall, and other climate details
- Helps in location-based predictions

5.6 Hosting & Deployment

Frontend: Vercel / Netlify (for React.js)

Backend: Render / AWS / DigitalOcean (for Node.js)

Database: MongoDB Atlas (Cloud Database)

CHAPTER – 6

IMPLEMENTATION

6.1 Set Up Development Environment

1. The Install required libraries like Flask, scikit-learn, joblib, etc., using pip.
2. Organize the project folder with files for the dataset, model, and API script.
3. Set up a virtual environment (optional) to isolate dependencies.
4. Preparethe dataset, clean data, and ensure required columns are present.
5. Trainthe model using scikit-learn, scale features, and save the trained model and scaler.
6. Create a Flask API, define endpoints, handle input validation, and add prediction functionality.
7. Test the API locally using tools like curl or Postman.
8. Document dependencies in requirements.txt for easy setup.

6.2 Process the historical data of crops Mar

Preprocessing historical crop data ensures it is ready for analysis or machine learning. It involves data collection from reliable sources, cleaning missing values, and selecting relevant features like soil nutrients and weather conditions. Continuous variables are normalized, and categorical data is encoded for model compatibility. Outliers are handled statistically, and data is aggregated by region, season, or crop type. Feature engineering creates new attributes like crop yield ratios to enhance predictions. Finally, the dataset is split into training, validation, and testing sets for proper model evaluation.

6.3 Data into Training and Testing Sets

In the above model, splitting and scaling data are essential steps for effective training. The dataset is first divided into training and testing sets using the train_test_split function from scikit-learn, ensuring 80% of the data is used for training and 20% for testing. Feature scaling is then applied using StandardScaler, which standardizes the features by removing the mean and scaling them to unit variance. This ensures all features are treated equally by the model, improving performance and accuracy, particularly for non-linear relationships.

6.4 Build the Random forest model for non linear data

Building a Random Forest model for non-linear data involves utilizing the ensemble learning approach. The model trains multiple decision trees that collectively capture complex patterns and interactions within the features, such as "N", "P", "K", "Temperature", and more. Using scaled data ensures balanced processing of variables, and splitting the dataset into training and testing sets validates performance. The Random Forest algorithm's ability to handle non- linear relationships makes it ideal for crop prediction, capturing intricate dependencies effectively. Hyperparameters like n_estimators and max_depth are tuned for accuracy, ensuring a robust and reliable model.

6.5 Training and evaluating the Model

Training the model involves feeding the scaled training data into the Random Forest algorithm, where multiple decision trees learn from patterns and relationships. The trained model is then evaluated using testing data to measure performance through metrics like accuracy, precision, recall, or F1-score, ensuring it can predict reliably and handle unseen non-linear data effectively.

6.6 Saving and loading the Model

To save the model, the joblib.dump() function stores both the trained Random Forest model and the scaler as .pkl files, ensuring they can be reused without retraining. Later, the model and scaler are loaded using joblib.load(), allowing seamless integration into the Flask API for predictions. This ensures efficiency and consistency in deployment.

6.7 Develop a Flask Web Application for Predictions

Load the trained model and scaler using joblib.load() to ensure consistency. Initialize the Flask app and define a POST route /predict-crop to receive JSON input. Validate input, convert it into a NumPy array, and scale it using loaded scaler. Pass the scaled input to the trained Random Forest model for prediction and return predicted crop as a

JSON response. Run the app on host="0.0.0.0", port=5003, with debug=True for local testing

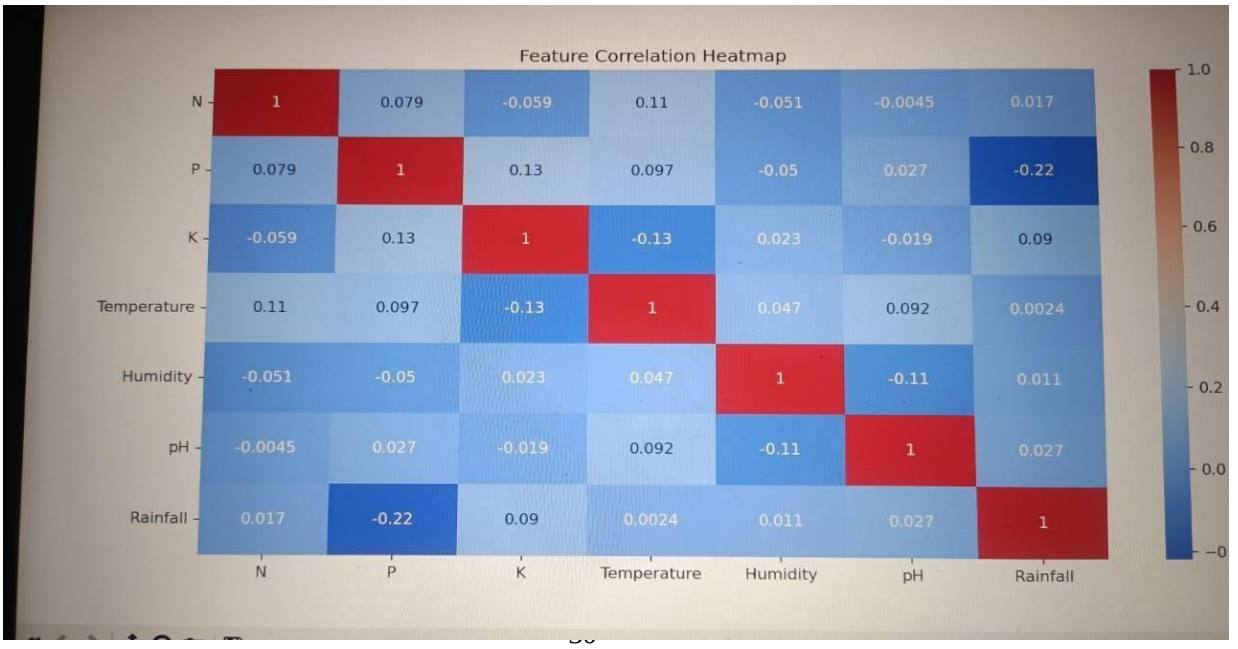
6.8 Create a Frontend for User Interaction

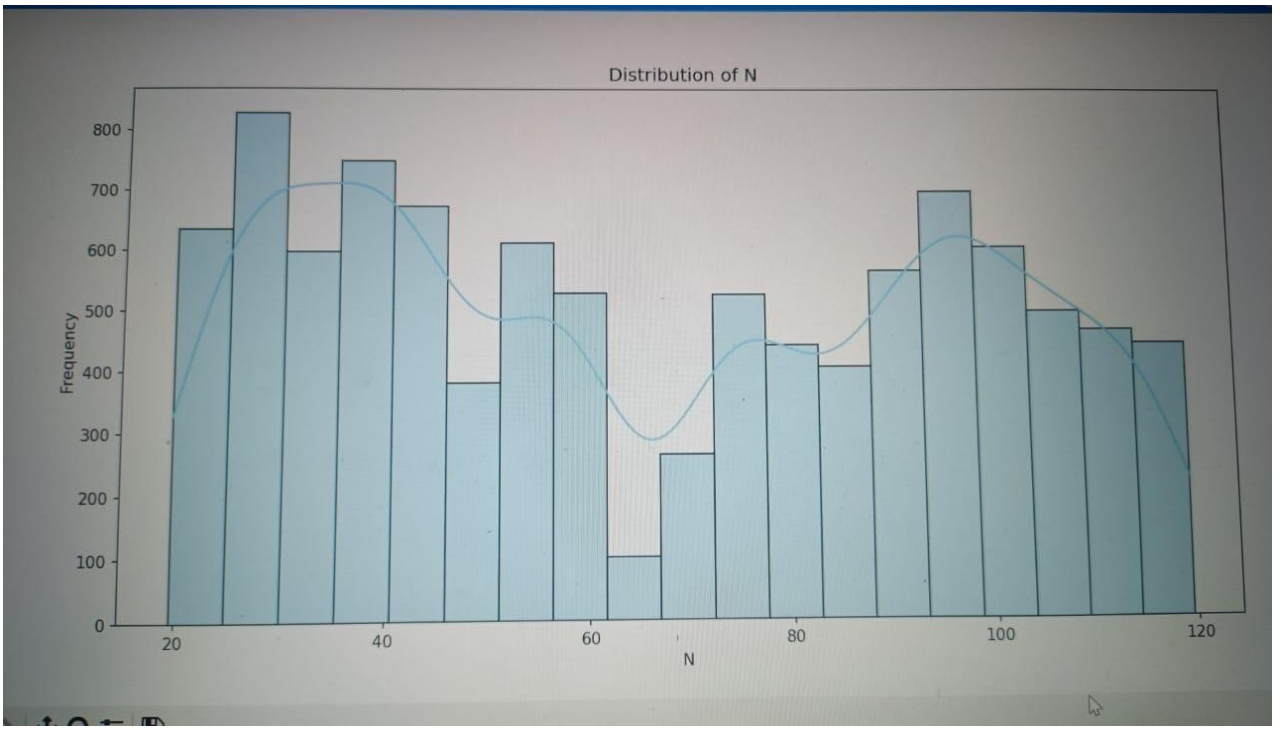
Set up a React project with create-react-app. Design a form for user inputs (e.g., "N", "P", "K", etc.) and a button to submit. Use axios or fetch for asynchronous calls to the Flask API. Validate inputs, send data to /predict-crop, and display predictions in a result component. Manage state with useState and handle API responses or errors. Style the app with CSS or frameworks like Material-UI. Run the app locally and connect it to the Flask backend for seamless interaction

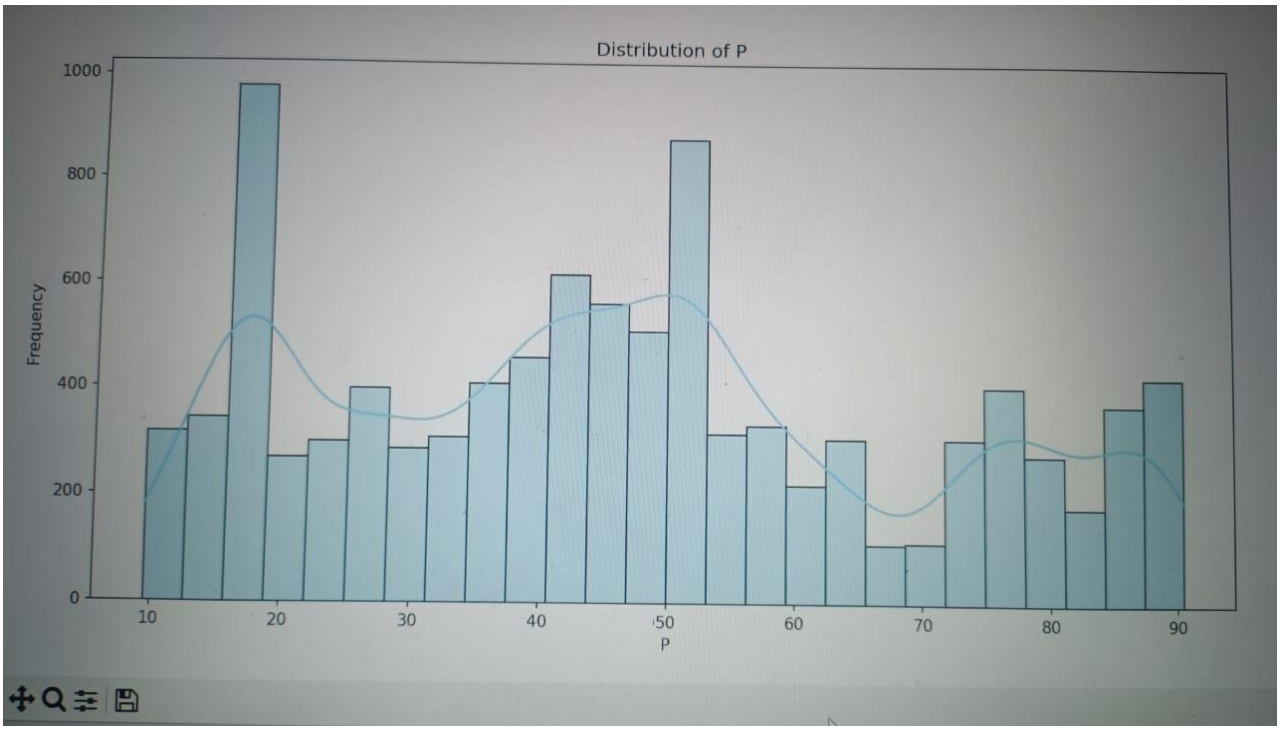
6.9 Deploy the Model on a Server

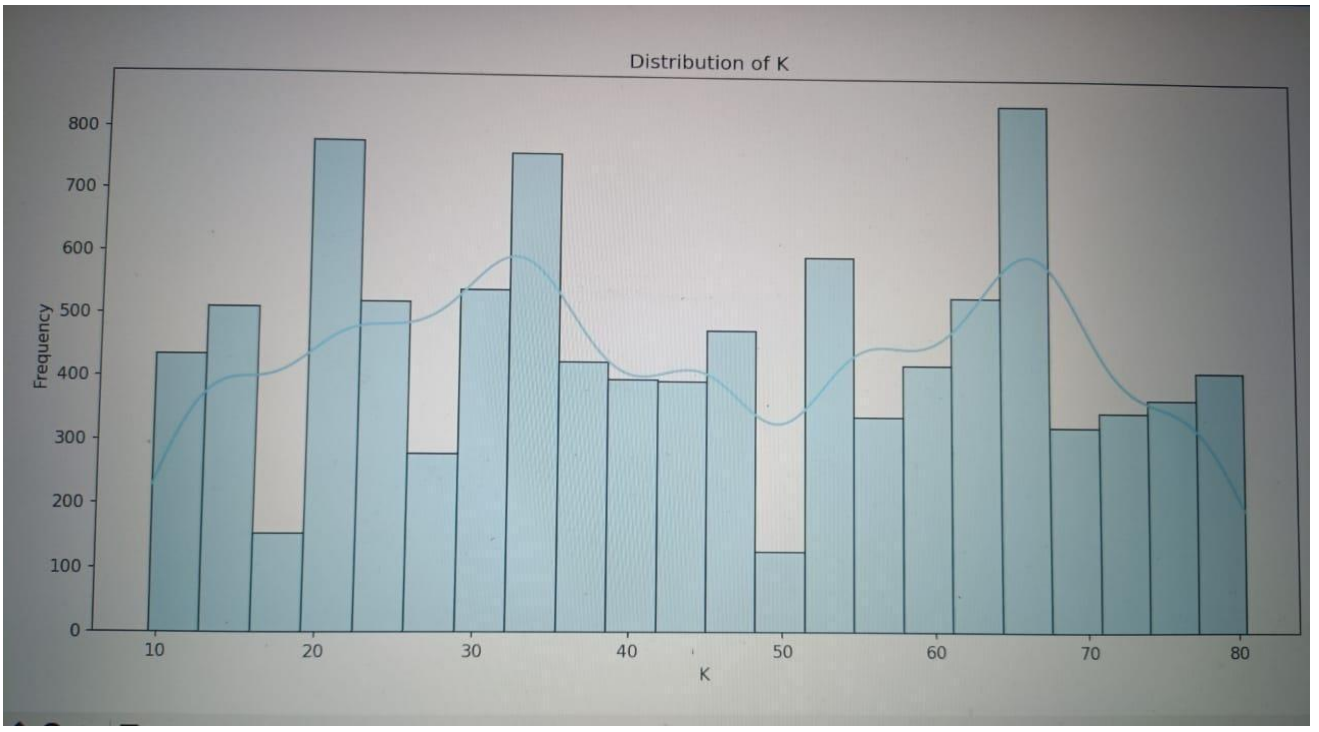
Host the trained model and Flask application on a cloud server like AWS, Azure, or Heroku. Install required dependencies on the server and transfer your project files. Configure the Flask app to listen to the server's host and port. Use a web server like Gunicorn or uWSGI for production. Set up a reverse proxy with Nginx for better scalability and security. Finally, test the deployed API endpoint to ensure seamless accessibility for clients.

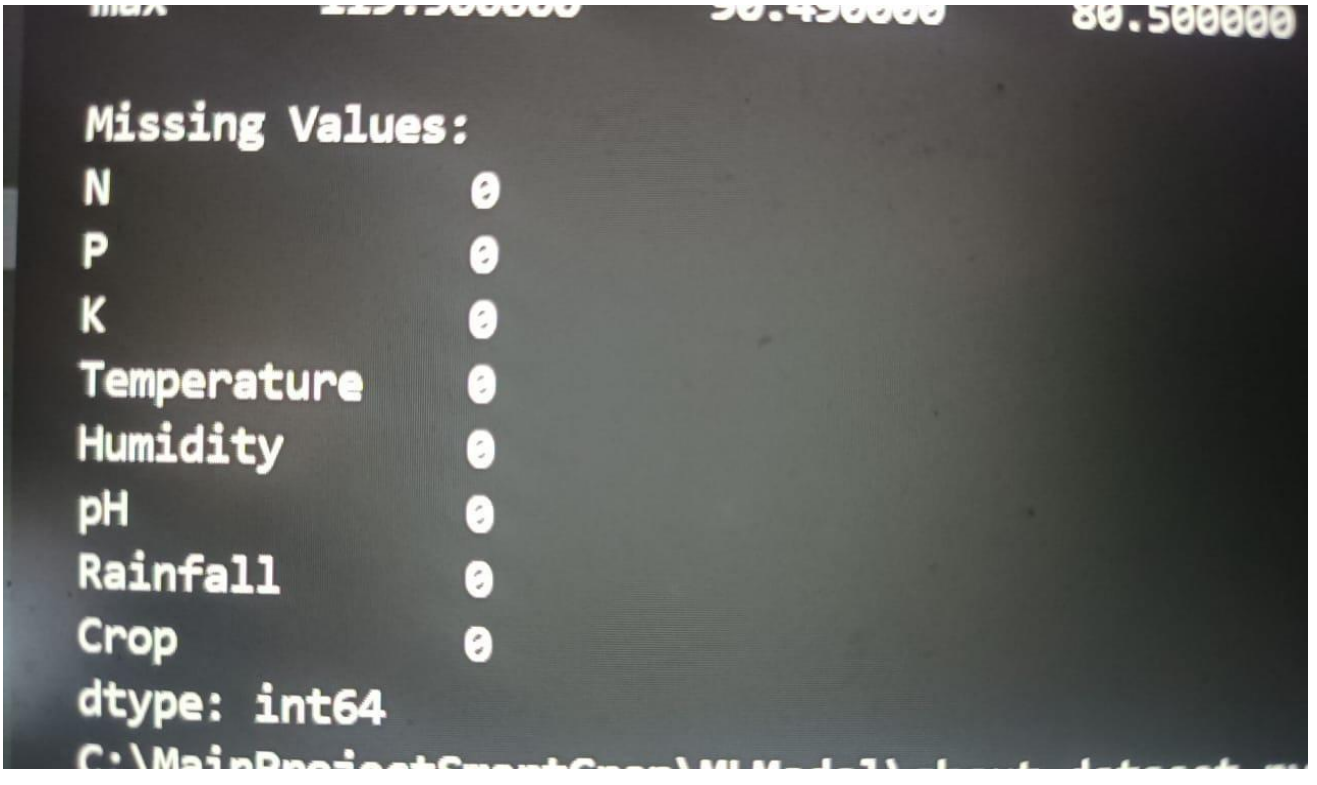
DataAnalysis (EDA):



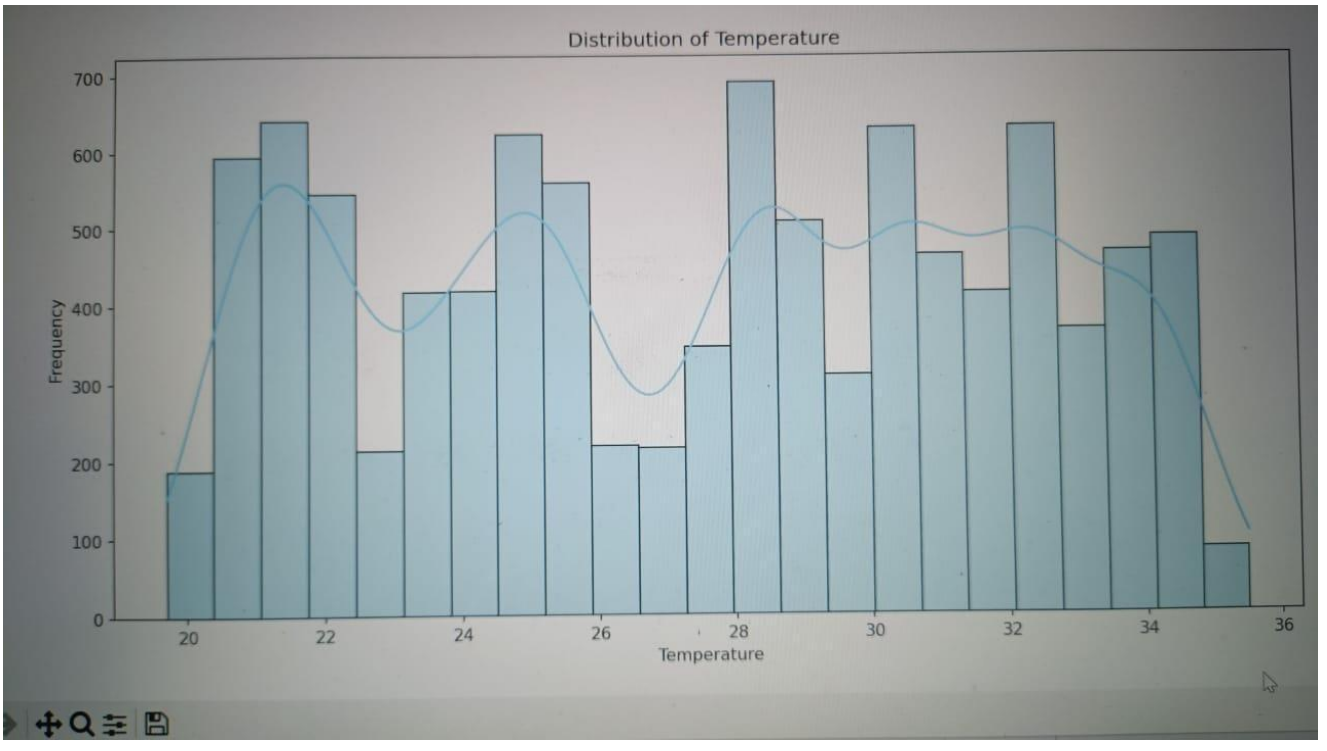

Distribution of N
Frequency
0
100
200
300
400
500
600
700
800
20
40
60
80
100
120
N

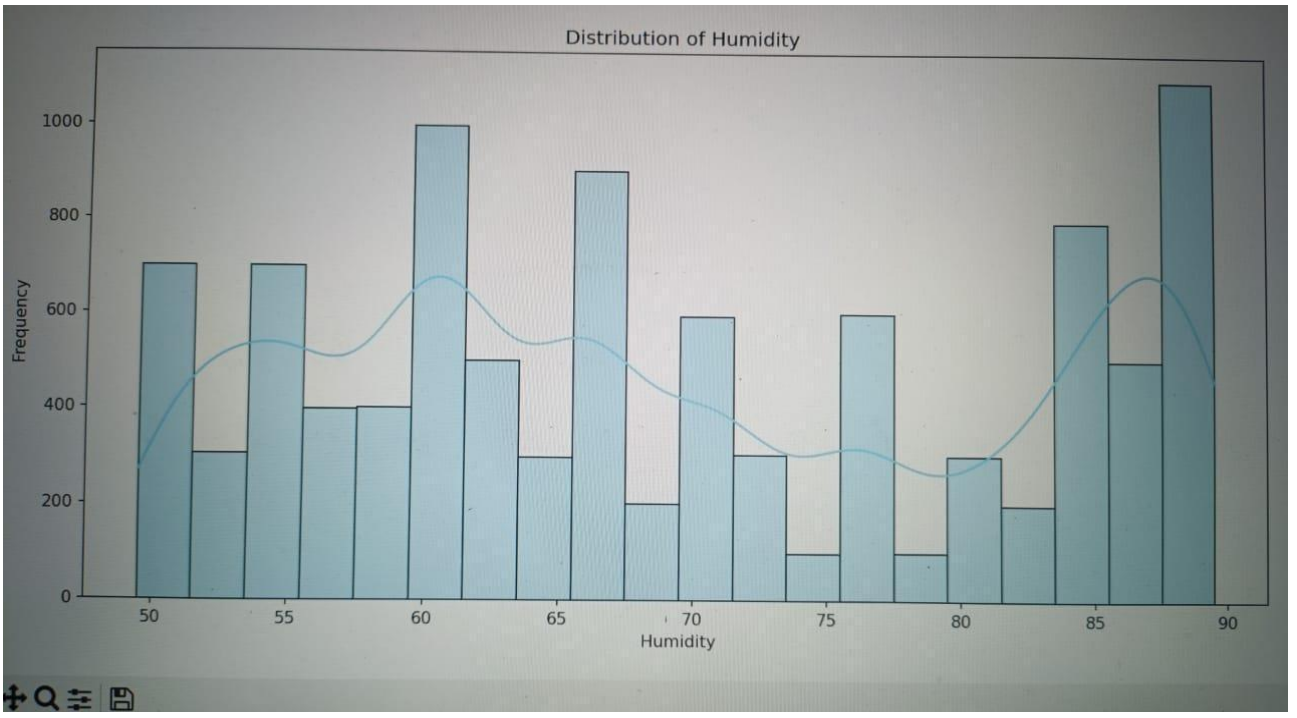

Distribution of P
Frequency
0
200
400
600
800
1000
10
20
30
40
50
60
70
80
90
P

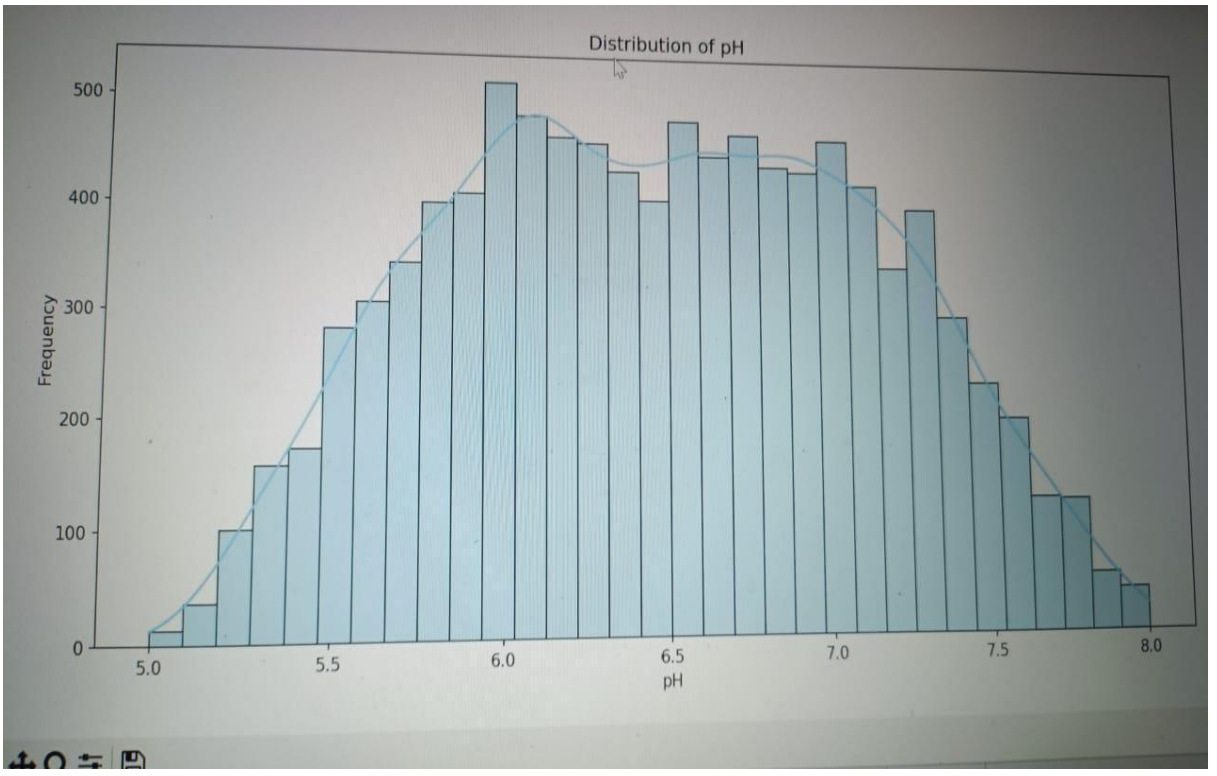

Distribution of K
Frequency
K
0
100
200
300
400
500
600
700
800
10
20
30
40
50
60
70
80

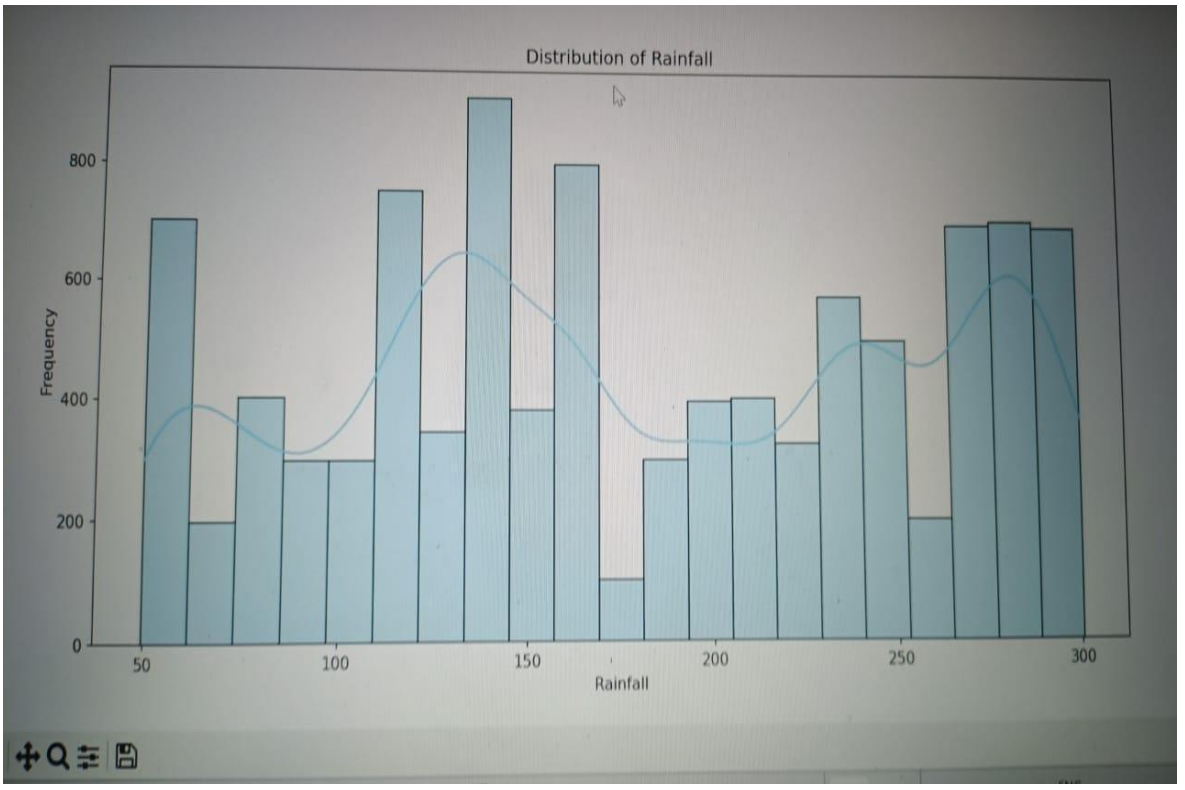

80.500000
Missing Values:
N 0
P 0
K 0
Temperature 0
Humidity 0
pH 0
Rainfall 0
Crop 0
dtype: int64

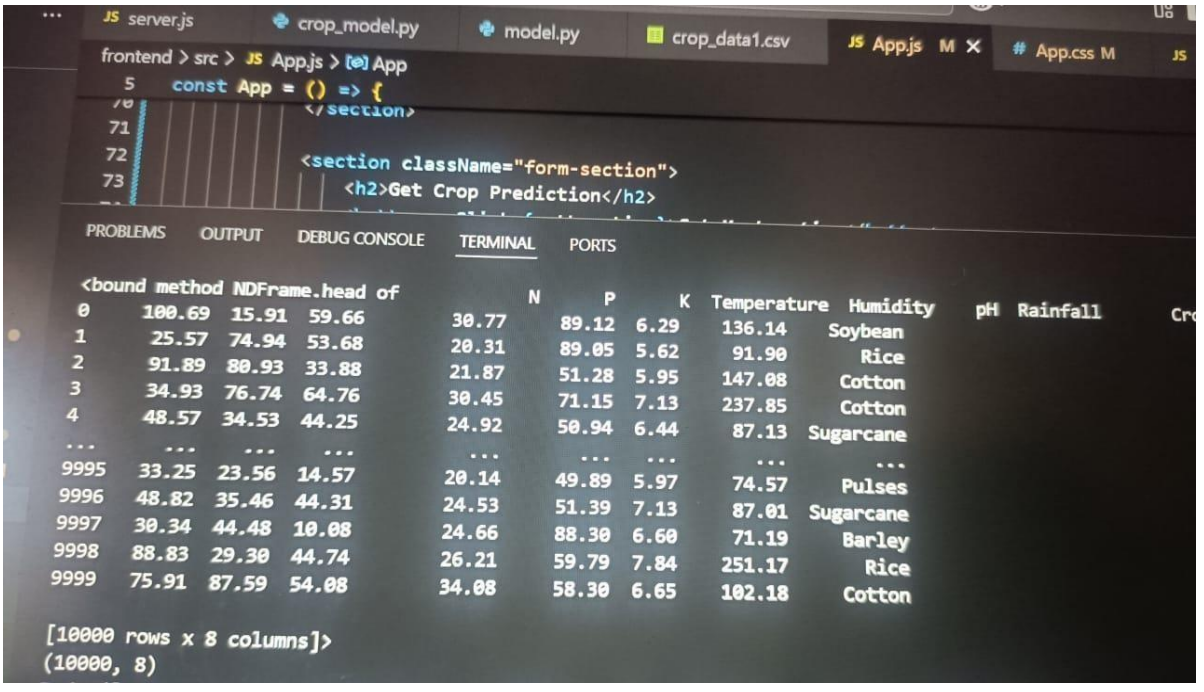
```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   N            10000 non-null  float64
 1   P            10000 non-null  float64
 2   K            10000 non-null  float64
 3   Temperature  10000 non-null  float64
 4   Humidity     10000 non-null  float64
 5   pH           10000 non-null  float64
 6   Rainfall     10000 non-null  float64
 7   Crop         10000 non-null  object
dtypes: float64(7), object(1)
```

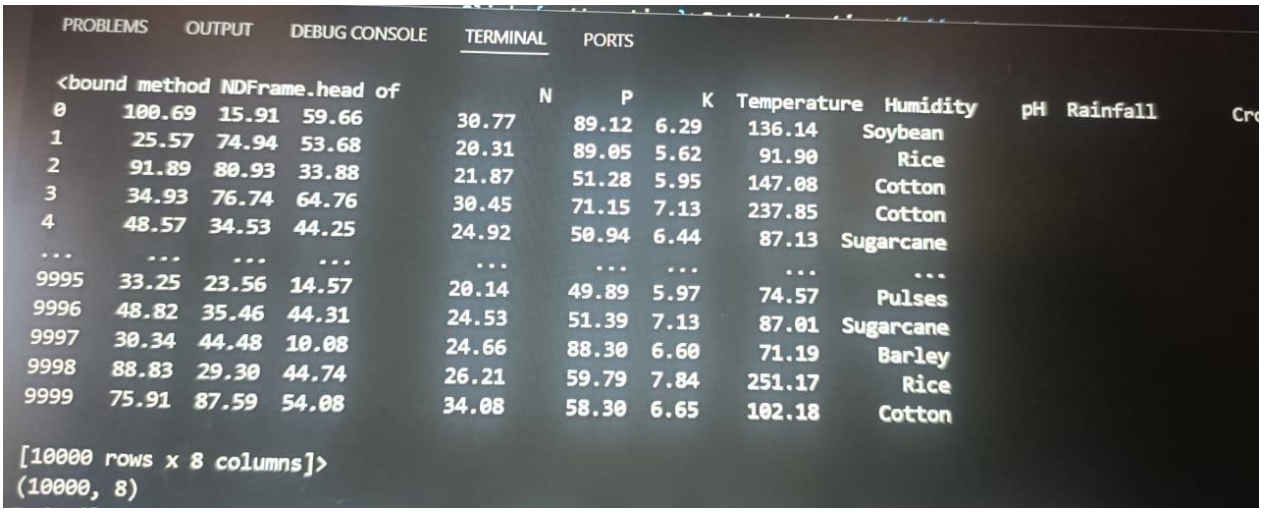



Distribution of Humidity
Frequency
1000
800
600
400
200
0
50
55
60
65
70
75
80
85
90
Humidity


Distribution of pH
Frequency
500
400
300
200
100
0
5.0
5.5
6.0
6.5
7.0
7.5
8.0
pH


Distribution of Rainfall
Frequency
Rainfall
0
200
400
600
800
50
100
150
200
250
300

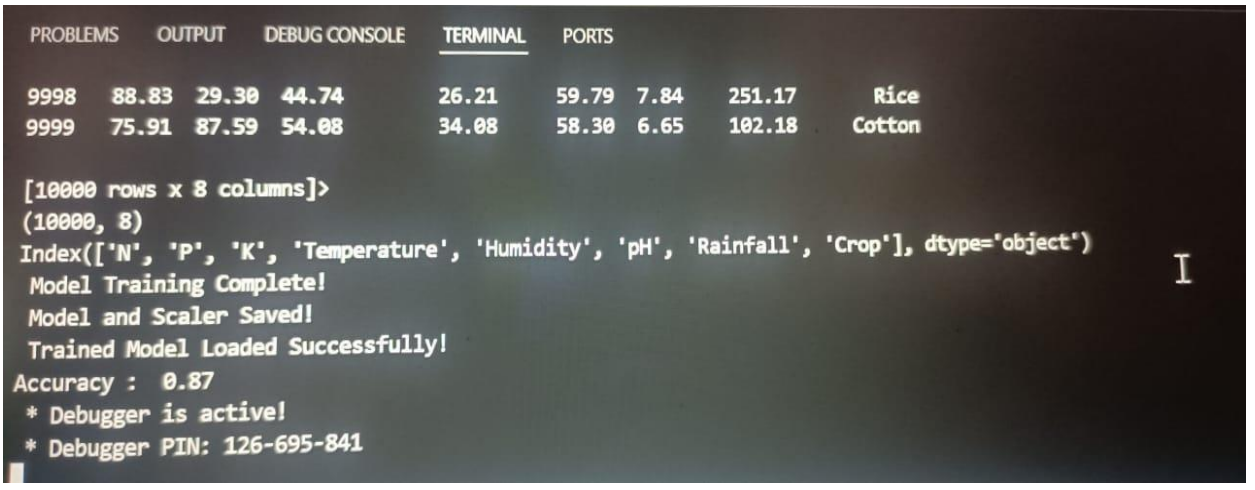

server.js
crop_model.py
model.py
crop_data1.csv
App.js M
App.css M
frontend > src > App.js > App
const App = () => {
</section>
<section className="form-section">
<h2>Get Crop Prediction</h2>
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
<bound method NDFrame.head of
N P K Temperature Humidity pH Rainfall
0 100.69 15.91 59.66 30.77 89.12 6.29 136.14 Soybean
1 25.57 74.94 53.68 20.31 89.05 5.62 91.90 Rice
2 91.89 80.93 33.88 21.87 51.28 5.95 147.08 Cotton
3 34.93 76.74 64.76 30.45 71.15 7.13 237.85 Cotton
4 48.57 34.53 44.25 24.92 50.94 6.44 87.13 Sugarcane
... ...
9995 33.25 23.56 14.57 20.14 49.89 5.97 74.57 Pulses
9996 48.82 35.46 44.31 24.53 51.39 7.13 87.01 Sugarcane
9997 30.34 44.48 10.08 24.66 88.30 6.60 71.19 Barley
9998 88.83 29.30 44.74 26.21 59.79 7.84 251.17 Rice
9999 75.91 87.59 54.08 34.08 58.30 6.65 102.18 Cotton
[10000 rows x 8 columns]>
(10000, 8)



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

<bound method NDFrame.head of        N      P      K  Temperature  Humidity    pH  Rainfall      Cr
0      100.69  15.91  59.66        30.77     89.12  6.29     136.14   Soybean
1       25.57  74.94  53.68        20.31     89.05  5.62      91.90      Rice
2       91.89  80.93  33.88        21.87     51.28  5.95     147.08    Cotton
3       34.93  76.74  64.76        30.45     71.15  7.13     237.85    Cotton
4       48.57  34.53  44.25        24.92     50.94  6.44      87.13  Sugarcane
...       ...    ...    ...          ...       ...   ...        ...       ...
9995    33.25  23.56  14.57        20.14     49.89  5.97      74.57    Pulses
9996    48.82  35.46  44.31        24.53     51.39  7.13      87.01  Sugarcane
9997    30.34  44.48  10.08        24.66     88.30  6.60      71.19    Barley
9998    88.83  29.30  44.74        26.21     59.79  7.84     251.17      Rice
9999    75.91  87.59  54.08        34.08     58.30  6.65     102.18    Cotton

[10000 rows x 8 columns]>
(10000, 8)
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

9998    88.83  29.30  44.74        26.21     59.79  7.84     251.17      Rice
9999    75.91  87.59  54.08        34.08     58.30  6.65     102.18    Cotton

[10000 rows x 8 columns]>
(10000, 8)
Index(['N', 'P', 'K', 'Temperature', 'Humidity', 'pH', 'Rainfall', 'Crop'], dtype='object')
 Model Training Complete!
 Model and Scaler Saved!
 Trained Model Loaded Successfully!
Accuracy :  0.87
 * Debugger is active!
 * Debugger PIN: 126-695-841
```

CODE

```
import pandas as pd
import numpy as np import joblib
from flask import Flask, request, jsonify from
flask_cors import CORS
from sklearn.model_selection import
train_test_split from sklearn.ensemble import
RandomForestClassifier from
sklearn.preprocessing import StandardScaler
 try:
      df = pd.read_csv("crop_data1.csv") # Ensure the file is
      present p r i n t ( " Dataset Loaded Successfully!")
      except FileNotFoundError:
       print(" Error: Dataset file 'crop_data.csv' not found


 exit()

# Define Features and Target
features = ["N", "P", "K", "Temperature", "Humidity", "pH", "Rainfall"]
target = "Crop"

# Check if required columns exist
missing_columns = [col for col in features + [target] if col not in
df.columns] if missing_columns:
 print(f"Error: Missing columns in dataset: {missing_columns}")
  exit()
X =df[features]
y = df[target]

# Scale the Features
scaler =
StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into Training and Testing Sets
X_train, X_test, y_train, y_test =train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train Model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
p r in t ( "Model Training Complete!")

# Save Model and Scaler
joblib.dump(model,
"crop_model.pkl")
joblib.dump(scaler, "scaler.pkl")
print( " Modeland Scaler Saved!")
```

```
try:
model= joblib.load("crop_model.pkl")
scaler = joblib.load("scaler.pkl")
    print ( "Trained ModelLoaded Successfully!") except
Exception as e:
print(f"Error loading model: {e}")
exit()

# Initialize Flask App
app = Flask(_name_)
CORS(app, resources={r"/*": {"origins": "http://localhost:3000"}}

@app.route("/predict-crop", methods=["POST"]) def
predict():
try:
data = request.get_json() print("
Received Data:", data)

# Validate input features
for feature in features:
if feature not in data:
return jsonify({"error": f"+ Missing feature: {feature}"}), 400

# Convert input to numpy array and scale it
input_data = np.array([[data[feature] for feature in features]])
input_scaled = scaler.transform(input_data)

# Predict Crop
prediction = model.predict(input_scaled)
print("Predicted Crop:", prediction[0])

    return jsonify({"crop": prediction[0]})

except Exception as e:
print(f"+ Prediction Error: {e}") return
jsonify({"error": str(e)}), 500

if _name_ == "_main_":
app.run(host="0.0.0.0", port=5003, debug=True
```

CHAPTER– 7

TESTING

7. Testing

7.1 Introduction to Testing

The goal of testing is to find flaws. Testing is the practise of attempting to find every possible flaw or vulnerability in a work product. It enables the testing of components, sub- assemblies, assemblies, and/or full products. It is the process of testing software to ensure that it meets its requirements and meets user expectations and does not fail in an unacceptable way. There are several types of tests. Each test type is designed to fulfil a unique testing requirement.

7.2 Test Objective

- All field entries must be filled properly.
- Pages must be activated in every level.
- The messages and responses must not be delayed.

7.3 Test Strategies

7.3.1 Unit Testing

The process of designing test cases for unit testing ensures that the core logic of the programme is operating correctly and that programme inputs result in legitimate outputs. Validation should be done on all internal code flows and decision branches. It is the testing of the application's separate software components. Prior to integration, it is completed following the conclusion of a single unit. This is an intrusive structural test that depends on an understanding of its structure. Unit tests evaluate a particular application, system configuration, or business process at the component level. Unit tests make assurance that every distinct path in a business process has inputs and outputs that are well-defined and that it operates precisely according to the stated specifications.

7.3.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more

concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction,shown by successfully unit testing, the combination of components is correct and consistent.

7.3.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

- Valid Input: identified classes of valid input must be accepted.
- Invalid Input: identified classes of invalid input must be rejected.
- Functions: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.

7.3.4 Acceptance Testing

Acceptance testing is a type of testing used to assess whether the software system has complied with the requirements. This test's primary goal is to assess the system's adherence to the business requirements and confirm that it has fulfilled the prerequisites for end user delivery.

"**Functional Testing**":

Valid Input

Field	Input	Expected Output
Test Case ID	TC-01	
Description	Predict crop for valid input	
Input	{'N': 50, 'P': 30, 'K': 20, 'Temperature': 25.5, 'Humidity': 60, 'pH': 6.5, 'Rainfall': 200}	
Expected Output		{'crop': 'Wheat'}

Missing Feature

Field	Input	Expected Output
Test Case ID	TC-02	
Description	Request missing a required feature	
Input	{'N': 50, 'P': 30, 'K': 20, 'Temperature': 25.5, 'Humidity': 60, 'pH': 6.5}	
Expected Output		{'error': 'Missing feature: Rainfall'}

Invalid Data Type

Field	Input	Expected Output
Test Case ID	TC-03	
Description	Input with non-numeric values	
Input	{'N': 'abc', 'P': 30, 'K': 20, 'Temperature': 25.5, 'Humidity': 'xyz', 'pH': 6.5, 'Rainfall': 200}	
Expected Output		{'error': 'Invalid input type'}

pH Out of Range

Field	Input	Expected Output
Test Case ID	TC-05	
Description	pH value greater than 14	
Input	{'N': 50, 'P': 30, 'K': 20, 'Temperature': 25.5, 'Humidity': 60, 'pH': 15, 'Rainfall': 200}	
Expected Output		{'error': 'pH value must be between 0 and 14'}

"**Integration Testing**": {

Missing Model File

Field	Input	Expected Output
Test Case ID	TC-06	
Description	Model file is missing	
Scenario	crop_model.pkl file is deleted	
Expected Output		{'error': 'Error loading model: No such file or directory'}

Missing Scaler File

Field	Input	Expected Output
Test Case ID	TC-07	
Description	Scaler file is missing	
Scenario	scaler.pkl file is deleted	
Expected Output		{'error': 'Error loading scaler: No such file or directory'}

CORS Issue

Field	Input	Expected Output
Test Case ID	TC-08	
Description	API accessed from an unauthorized domain	
Scenario	Request sent from http://untrusted.com	
Expected Output		{'error': 'CORS policy issue'}

"**Performance Testing**

High Load

Field	Input	Expected Output
Test Case ID	TC-09	
Description	Sending 1000+ requests in 10 seconds	
Scenario	API is stress-tested with rapid requests	
Expected Output		API should not crash, response time should be within acceptable range

Slow Response

Field	Input	Expected Output
Test Case ID	TC-10	
Description	Handling complex inputs	
Scenario	Extreme values given as input	
Expected Output		Response time should be under 2 seconds

CHAPTER – 8

SCREENSHOTS

8. ScreenShots

8.1 Home Page

The Home page ofthe application that displays best Practices for cultivation.

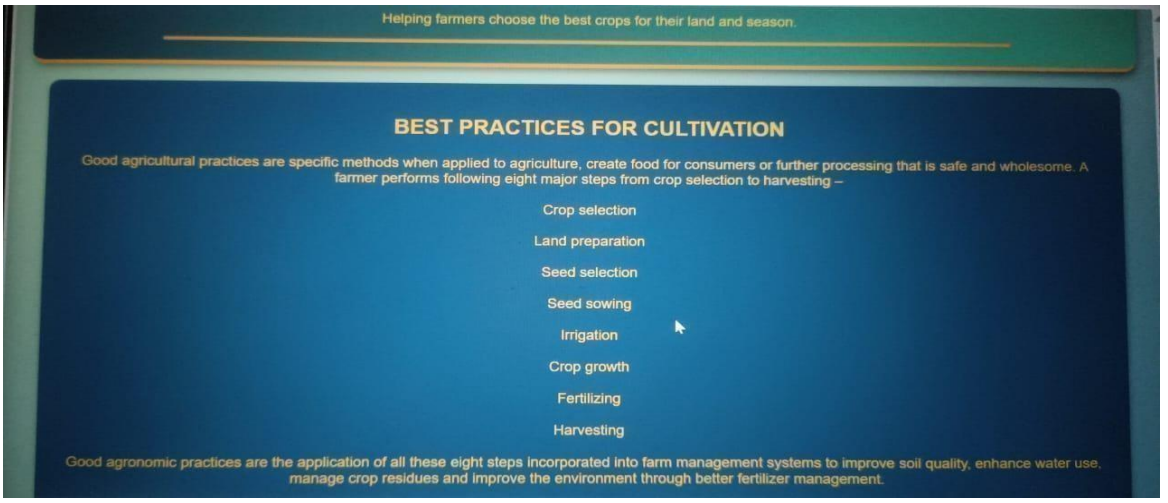


Fig no. 01

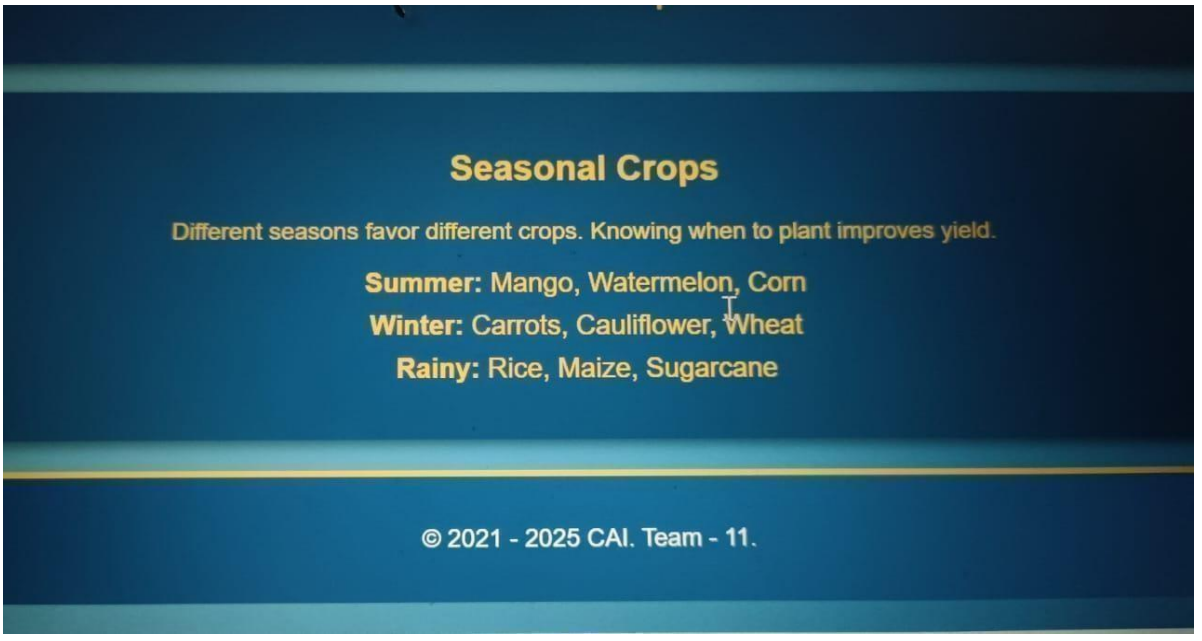
8.2 Predict Page

To Predict Crop we move on to predicition page where we need to enter valid values and numbers of soil nutriens and click predict button.

Fig no.02

This page displays the predicted crop.

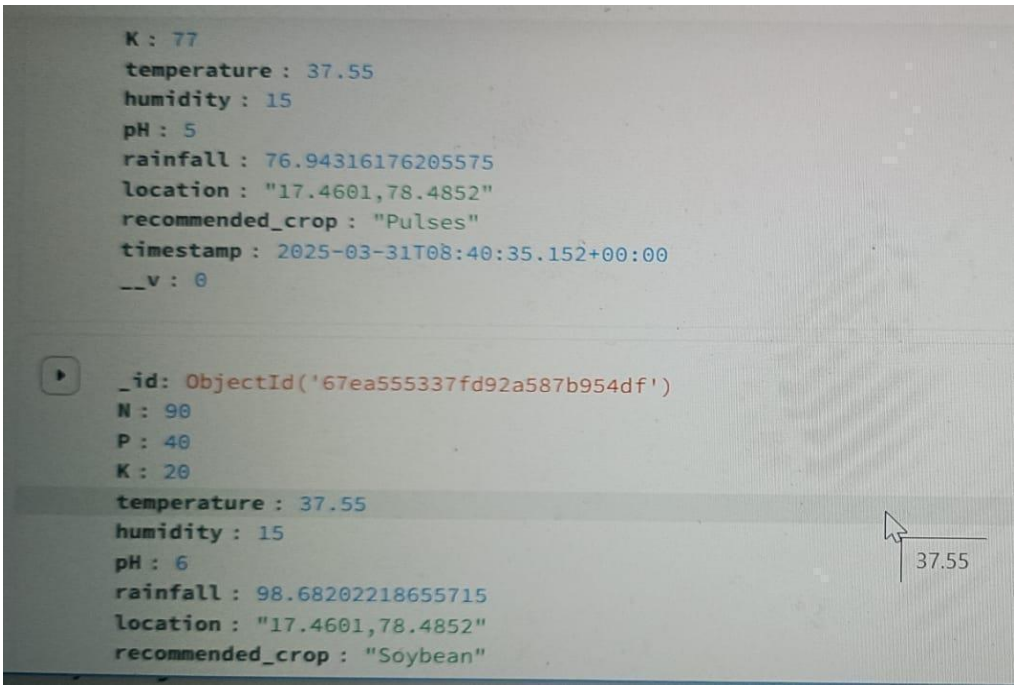
8.3 Footer Page



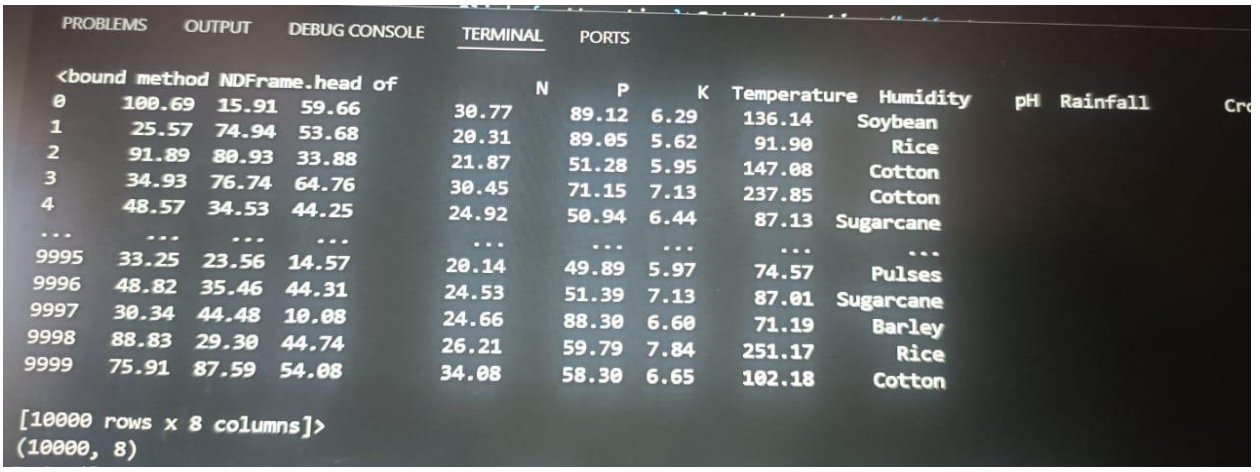
The page of the application that displays seasonal crops.

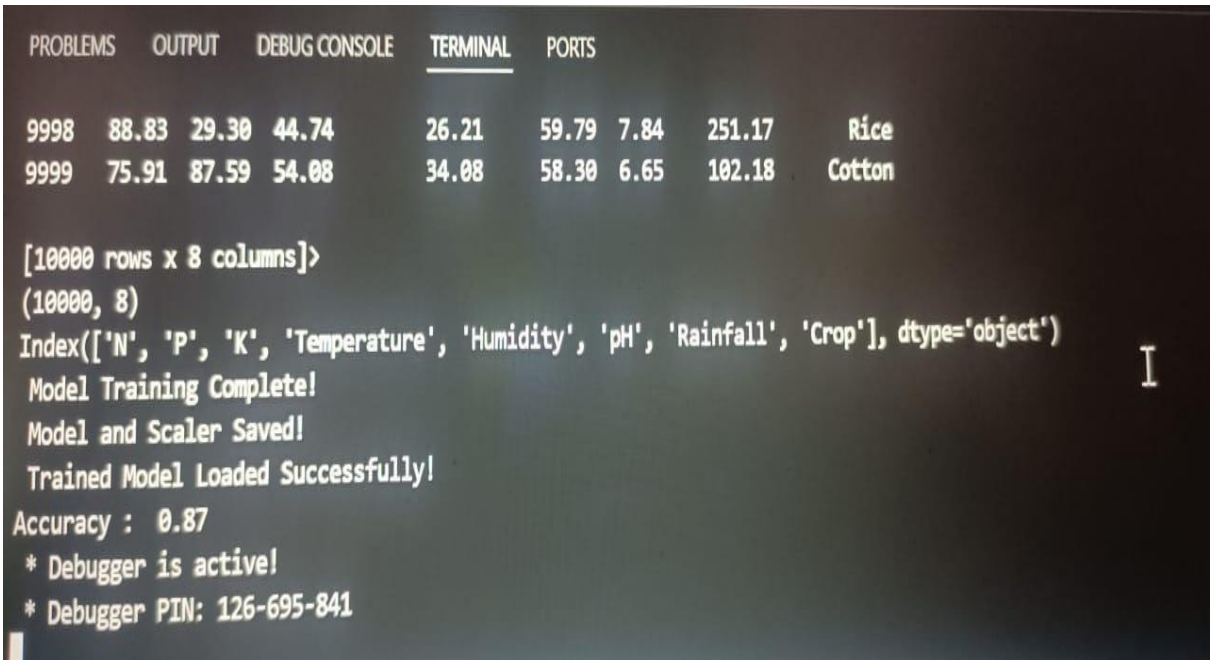
8.4 Predicted Crop Info

The Predicted Crop Info displays the details of all the valid data about the predicted crop.



Console Page:





CHAPTER – 9

CONCLUSION AND FUTURE WORK

9. Conclusion and Future Work

9.1 Conclusion

The Smart Crop Prediction System is a significant step towards data-driven agriculture, helping farmers make informed crop selection decisions. By utilizing Machine Learning algorithms and integrating real-time weather data, the system predicts the most suitable crop based on soil nutrients (N, P, K, pH) and environmental conditions (temperature, humidity, rainfall). The implementation of this system has resulted in more accurate crop predictions, allowing farmers to optimize their yield and sustainability. One of the key strengths of this project is the integration of weather APIs, which provides dynamic insights into environmental factors affecting crops. Additionally, the user-friendly web interface ensures easy access to predictions, making it accessible even for farmers with minimal technical knowledge. By storing data in MongoDB, the system allows tracking of past predictions, enabling continuous improvement and analysis of farming trends. Ultimately, this system serves as a valuable tool in modernizing agricultural practices, ensuring better productivity, resource optimization, and increased profitability for farmers.

9.2 Future Work

To improve the Smart Crop Prediction System, several enhancements can be made to increase its accuracy and usability. Expanding the dataset size with data from diverse regions and using advanced machine learning models like deep learning will refine predictions. Incorporating additional parameters such as soil moisture, sunlight, and wind speed will provide a more detailed crop analysis. A mobile app with regional language support will enhance accessibility for farmers. Integrating IoT-based sensors for real-time data collection will improve monitoring and decision-making. Providing fertilizer recommendations based on soil quality and including market insights for price predictions will help farmers optimize resources and profitability. Climate adaptation features can suggest resilient crops, while real-time weather alerts will support timely decision-making. Blockchain technology can enhance data security and transparency, increasing trust in system. Collaboration with agricultural experts and government agencies can provide valuable support and subsidies. By continuously evolving, the Smart Crop Prediction System can drive efficient crop planning, higher yields, and sustainable farming.

CHAPTER – 10

REFERENCES

10. References

1. Li, L.; Wang, B.; Feng, P.; Liu, D.L.; He, Q.; Zhang, Y.; Wang, Y.; Li, S.; Lu, X.; Yue, C.; et al. Developing machine learning models with multi-source environmental data to predict wheat yield in China. Comput. Electron. Agric. 2022, 194, 106790.
2. van Klompenburg, T.; Kassahun, A.; Catal, C. Crop yield prediction using machine learning: A systematic literature review. Comput. Electron. Agric. 2020, 177, 105709.
3. Kuradusenge, M.; Hitimana, E.; Hanyurwimfura, D.; Rukundo, P.; Mtonga, K.; Mukasine, A.; Uwitonze, C.; Ngabonziza, J.;Uwamahoro, A. Crop Yield Prediction Using Machine Learning Models: Case of Irish Potato and Maize. Agriculture 2023
4. Xu, W.; Kaili, Z.; Tianlei, W. Smart Farm Based on Six-Domain Model. In Proceedings of the IEEE 4th International Conference onElectronics Technology (ICET), Chengdu, China, 7–10 May 2021;
5. Moysiadis, V.; Tsakos, K.; Sarigiannidis, P.; Petrakis, E.G.M.; Boursianis, A.D.; Goudos, S.K. A Cloud Computing web-based application for Smart Farming based on microservices architecture. In Proceedings of the 11th International Conference on Modern Circuits and Systems Technologies (MOCAST),Germany, 8–10 June 2022
6. Ranjan, P.; Garg, R.; Rai, J.K. Artificial Intelligence Applications in Soil & Crop Management. In Proceedings of the IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 21–23 December 2022;
7. 7Oré, G.; Alcântara, M.S.; Góes, J.A.; Oliveira, L.P.; Yepes, J.; Teruel, B.; Castro, V. Crop Growth Monitoring with Drone-Borne DInSAR. Remote Sens. 2020, 12, 615.
8. Gehlot, A.; Sidana, N.; Jawale, D.; Jain, N.; Singh, B.P.; Singh, B. Technical analysis of crop production prediction using Machine Learning and Deep Learning Algorithms. In Proceedings of the International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES),India, 24–25 September 2022.