Eco Probe

A Final-Year project thesis authored by

Mahar Muhammad Uzair Gull (200673)

Muhammad Noman (210622)

Taimoor Hassan (210628)

B.S., Artificial Intelligence, 2025

SUPERVISOR

Prof. Muhammad Farooq

CO-SUPERVISOR

Dr. Tahir Akram

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Artificial Intelligence

DEPARTMENT OF CREATIVE TECHNOLOGIES,
FACULTY OF COMPUTING AND ARTIFICIAL INTELLIGENCE,

AIR UNIVERSITY, ISLAMABAD

May 2025

Eco Probe

FYP THESIS SUBMISSION FORM

Project Title	Eco Probe	
Supervisor Name and Designation	Sir Muhammad Farooq	
Group Members	**Student Name**	**Reg. ID**
	Mahar Muhammad Uzair Gull	**200673**
	Muhammad Noman	**210622**
	Taimoor Hassan	**210628**

Group Member Signatures	
	Supervisor Signatures

CERTIFICATE OF APPROVAL

This is to certify that this final year project thesis on **Eco Probe** has been accepted as the partial fulfilment of the requirement for the degree of **Bachelor of Science in Artificial Intelligence**, submitted by the following students:

Mahar Muhammad Uzair Gull (200673)
Muhammad Noman (210622)
Taimoor Hassan (210628)

FYP Committee Signatures:

_____ _____

Supervisor **FYP Convener**

Muhammad Farooq

_____

Chair Department

Dr. Imran Ihsan

Date: _____

DECLARATION

We hereby declare that the work presented in this Final Year Project (FYP) thesis, entitled "Eco Probe", is our own effort and has not been submitted for any other degree or qualification at this or any other university, institute, or institution of learning.

We further affirm that no portion of this work, either in its entirety or in part, has been copied from any sources, even when references have been provided. Any contributions from external sources have been properly cited and acknowledged.
We also understand that any form of academic dishonesty, including plagiarism, is a serious violation of ethical standards, and we are committed to upholding the principles of academic integrity in all aspects of my work.

Group Member Name	**Signatures**
Mahar Muhammad Uzair Gull	
Muhammad Noman	
Taimoor Hassan	

ACKNOWLEDGEMENT

We are grateful to all those who assisted in finishing our Final Year Project.
First of all, we appreciate Dr. Fawad Salaam Khan for helping us with his wonderful advice and assistance. Because of his valuable input and kind words, we have improved our studies and learned new things.

We would like to thank Dr. Ehtisham-ul-Haq and Mr. Muhammad Farooq from Air University Islamabad for their teaching and support. They have helped us better understand the topic and improve our skills through their lectures, discussions and guidance.

We are also grateful to our families and friends, especially Mr. Rasikh-ur-Rehman for standing by us and showing us patience, encouragement and understanding all along. Being encouraged by them is what has kept us going and determined all along.

We would like to express our gratitude to everyone who took part and gave their views to this study. The ideas and suggestions from our participants have greatly helped us and have improved the significance of what we found.
All in all, we thank each and every individual who helped make this project happen, regardless of their contribution.

DEDICATION

To our families, thank you for always believing in and supporting me through everything. Our determination has grown because our parents always believed in us and encouraged us to improve. We appreciate everything they do, cheering for us and encouraging us to keep working and persisting. Our commitment to excellence can be seen clearly in this project.

TABLE OF CONTENTS

List of Tables

List of Figures

ABSTRACT

I am working on creating an AI tool that can advise users on the best crop for their circumstances. It will let the user monitor their resources and see their crop plantation in a clear and critical way. The project will be able to predict the short-term outcomes, so users stay informed about the present and upcoming condition of crops and land. The goal of the project is to ensure the user can properly analyse their land and predict what crops they will plant.

Through analysis, we will find out both what the system must do and what qualities it should have. Together with agricultural and Micro Controller Technical professionals, the system design and methodology will be created to make certain the system suits them and is simple to operate. Your data will be kept safe and protected in accordance with data protection laws.

The system will make use of hardware such as soil sensors, as well as Python and machine learning libraries such as Pandas or Numpy. The project's requirements will guide the choice of hardware for the system.

The purpose of this project is to aid Agricultural Professionals. This project will be created as a pc app where the app will gather real-time data from the sensor. Significant technological progress can be made in the project and more funding may be obtained for its development. Ethical issues such as privacy, data protection and fairness should be kept in mind during the creation and implementation of a system.

Chapter 1

INTRODUCTION

In this chapter, the focus is on introducing the integration of artificial intelligence (AI) in the agricultural sector and its transformative potential in addressing various analytical activities. It describes the hardships individuals face when managing what they plant and testing their soil, since they do not have proper agricultural help and cannot keep track of their land's fertility. It points out that using AI technology in farming can help individuals increase their harvests and satisfy their crop needs by acting wisely.
Furthermore, it points out the importance of finding new AI tools that can make old farming methods accessible to new agricultural workers.

1.1 Background and Motivation

The project background shows how AI is being applied in agriculture to manage the process of fertilizing, pesticide application, land irrigation and farming. This initiative was created because people lack knowledge and experience when considering agriculture as a profession. With AI being introduced, there are now better ways to solve these problems and improve agriculture for both individuals and governments. The drive to motivate the project comes from the importance of carrying out exact soil analysis for sound decision-making. AI-driven information and advice can greatly support the agriculture sector and help users decide the best options for their crops.

Furthermore, AI is used in the project to help users spend less on helpful features, thus making their finances stronger. Using AI to help in farming and farming resource management, the project wishes to respond to the rise in demand for AI-based products in agriculture. Furthermore, the project is connected to the company's main objectives, encouraging innovation and helping the business compete in the agriculture industry.

1.2 Problem Statement

Our final-year project is focused on helping people overcome the problems they experience when deciding on agricultural matters. A lot of people find it hard to manage farming due to not knowing how to farm, not analyzing the soil well and not getting guidance specific to their needs.
I want to develop a system that uses artificial intelligence for predicting what crops to grow and automates soil testing and the analysis of land conditions. Support users in deciding what to do with their farming. The project aims to increase understanding of agriculture, guide people in smart spending and help improve the well-being of those in agriculture.

1.3 Project Goal and Objectives

Project Goal:

The goal of the project is to create an AI tool that supports individuals in their farming activities.

Project Objectives:

- Develop AI Models: Use AI tools to study data about the soil from users and then use that to predict crop harvest.
- Use a multi-parameter soil sensor and create a program to automatically gather its data and transform it from hexadecimal to the standard units.
- Give Guidance According to Goals: Align recommendations and suggestions with what the farmer wants to achieve.
- Make the interface simple and interactive to ensure users enjoy using the crop recommendation system.

1.4 Project Requirement Specifications

It requires the system to have a soil multi-parameter sensor and an operating system to function. Furthermore, users should be able to add their agricultural data easily and in a convenient manner, using formats such as manual entry. The main task of the system is to analyze user's agricultural data using AI and offer suggestions. This means it has features such as growable and beneficiary crops, but for now, it does not involve managing resources. On the topic of non-functional requirements, the main focus is on performance, so the service must respond quickly and efficiently when processing significant user data. Since the number of users and data may increase in the future, a database needs to be scalable. Farmers must be able to use the information at any time which is why reliability has to be perfect. A user interface must be intuitive, easy to use and clearly guide users, so the app can be improved regularly. A well-designed AI system for crop recommendations helps people by giving them access to powerful farming and irrigation tools.

1.5 Project Scope and Deliverables

Project Scope:

The project will focus on the following key areas:

- Building an AI Platform for Soil Analysis: Design and carry out the creation of an AI tool for checking soil quality and its Nitrogen, Phosphorus, Potassium, pH and fertility levels.
- The development of easy-to-use and sensible user interfaces for interacting with the crop recommendation system.
- Bringing together the key features: Integration of collecting, converting, entering, processing and recommending data in the agricultural field.
- Reliable, correct and functioning systems are assured by rigorously testing and carrying out quality assurance.

Project Deliverables:

The expected deliverables include:

- AI-Integrated Crop Recommendation System: A Crop Related Recommendation tool using AI to provide unique guidance and suggestions based on your soil data.
- User Interface Designs: Make simple UI with descriptions and guidance to use the tool.
- Testing Reports: These records provide details on how the system was tested, what was found and what the results showed.

Concentrating on these areas and achieving the mentioned outputs allows the project to build an AI system for recommending crops that equips users to make better decisions on their farms.

1.6 Novel Aspects and Major Contributions

What sets the project apart from previous works is its full integration of AI technology in soil analysis, aiming to identify each user's needs and problems. Our project provides a solution that is not limited to the knowledge gained from old farming techniques. The major achievement is that AI is used to advise farmers on the best crops for their needs and goals. As a result, both the usability and usefulness of decision-making in agriculture are improved.
This project also features automation for data collection, intuitive interfaces and the ability to learn as the situation on the farm changes over the years. We are using AI to help users grow their crops more effectively, by providing a smarter, more proactive and intuitive solution.
In addition, our project is designed to ensure that the system for crop recommendations is easy to use by people in the agricultural sector from different backgrounds.
All in all, the innovative features and main contributions of the project include using AI, analyzing soil and striving to improve the farming knowledge and health of its users.

1.7 Thesis Layout

At the beginning, the project discusses using AI in agriculture, highlighting the main problems faced in agricultural decision-making. It determines the main issue with farming and agriculture and suggests using AI to solve it.
The objective is to build an AI system that recommends crops to achieve the highest yield and boost the well-being of the agricultural industry. The project deals with setting out the system, while deliverables are intended to solve problems in agriculture. What makes this project unique is its approach to assisting agriculture and empowering farmers with AI technology.

Chapter 2

LITERATURE REVIEW

2.1 Overview of Relevant Literature

AI-powered tools assist farmers in understanding complex information that can be used to make sound decisions. An example is precision agriculture, where AI is used to handle changes in crop fields. For example, AI-powered drones can help farmers quickly locate any diseases or nutrient problems in their fields. Many users apply ML models, including neural networks and random forests, to predict how much crops will grow. They review information from past seasons such as rain and nutrients in the soil, to predict the results of each harvest and help farmers make good decisions. A research group in Australia showed how farmers were able to lessen their chances of drought damage through ML, since it accurately predicted wheat yields. AI plays a big role in crop monitoring. Using satellite images and deep learning methods, it is possible to discover any pest infestation or water stress at any given moment. Managers in vineyards across California now rely on drones with AI to spot the effects of drought and help reduce water use, making the process more sustainable. Eco-friendly farming benefits from using less chemical treatment, thanks to these tools.

The quality of the soil and weather greatly affects crop yield and AI is very good at analyzing these factors. Plant growth is directly influenced by how much nitrogen, phosphorus and water are in the soil, as well as by its pH value. A soil pH that is either too acidic or alkaline can make it difficult for crops to absorb nutrients, causing the crops to grow slowly. Temperature, the amount of rainfall and humidity all affect plant growth. The models consider all of these variables when making predictions about the yields. Researchers in India are using AI to analyze soil and weather patterns, then adjust the time of planting rice to increase production.

Many research projects have pointed out the practical uses of AI. Kenyan smallholder farmers make use of mobile apps with AI to get help managing their crops, resulting in higher farm earnings. In Dutch AI-powered greenhouses, the environment is automatically adjusted to ensure that tomatoes grow perfectly. At the same time, AI is used on Brazil's soybean farms to help manage vast crops and increase productivity. AI has proven to be effective in various environments and at various levels.

2.2 Previous Works and Research in the Field

- Soil Fertility Analysis and Crop Prediction Using Machine Learning by Muneshwara etal.
  This study used machine learning models based on soil NPK values to predict crop suitability. While informative, it relied on static datasets, unlike our project's real-time, sensor-driven approach, enhancing adaptability to field conditions.
- Optimizing Crop Recommendation Systems Using Machine Learning Algorithms by Evuarherheetal.
  The researchers compared models like Decision Trees and kNN to improve crop recommendation accuracy. This comparative focus aligns with our project's evaluation of multiple models (ANN, Naive Bayes, Random Forest, and kNN) for optimal prediction but lacks the real-time adaptability our project offers.
- Crop Prediction Method to Maximize Crop Yield Rate Using Machine Learning by NeelamSinghetal.

Focused on maximizing crop yield using regional data, this study highlights the importance of localized predictions but is limited by static, region-specific data. Our project advances this work by leveraging IoT for dynamic, real-time predictions, adaptable across locations.

- Suitable Crop Suggesting System Based on NPK Values by Shakib Mahmud et al. This system provided crop recommendations using NPK values, similar to our soil sensor's capabilities. However, it did not consider other environmental factors, which our project includes for a more comprehensive prediction model.

2.3 State-of-the-Art Techniques and Technologies

Current state-of-the-art crop prediction techniques utilize various machine learning and deep learning approaches, each with strengths and challenges. Key techniques include:

- Artificial Neural Networks (ANN):
- ANNs are widely used in crop prediction because they can represent the many connections and influences among different factors. ANNs are useful in nonlinear problems, making it helpful for predicting how much food crops will yield from different types of soil and climates. Despite this, training ANNs generally requires a large amount of data and sometimes they can fit the data too perfectly, especially if the dataset is not very large.
- Random Forest: This method is often applied for its reliability and ability to process high-dimensional data. Because it can adjust for changing values and prevent overfitting with bootstrap aggregation, it is popularly used to analyze data from the agricultural sector. The research of Anandaraja et al. (2021) and Gupta et al. (2020) indicated that Random Forest performed better at estimating crop yields. You are comparing the performance of your project's model to other models using Random Forest.
- Combining Naive Bayes with soil parameter data has helped to effectively categorize crops. It provides accurate results when applied to variables such as soil types and types of crops. However, it assumes feature independence, which may not fully capture complex interactions in agricultural data.
- k-Nearest Neighbors (kNN): Known for its simplicity and effectiveness in classifying agricultural data, kNN has been applied to predict soil type, nutrient levels, and yield class. According to a study from Mehta et al. (2019), kNN performs well in soil classification tasks and this is the reason you used kNN for comparison in your project.
- Thanks to the growth of IoT, now real-time data from sensors can be used to create AI models that adapt as the field changes. Most significantly, Sharma et al. (2022) found that ML models used with sensors can provide faster and more accurate guidance to farmers about their crops.

2.4 Limitations and Gaps in Existing Research

While there have been significant advancements, existing research in AI-based crop prediction faces several limitations:

- Data Quality and Real-Time Constraints: Researchers depend mostly on historical information and this may not reflect what happens now. In addition, the models often lose some accuracy because these datasets have many gaps and inaccuracies. By utilizing a Raspberry Pi and a multiparameter sensor for real-
- time data collection, your project addresses this limitation, providing more relevant and timely recommendations.
- Researchers have studied how well various ML models perform in predicting crops, but have not done it on a regular basis. The project is valuable because it examines and compares four models and helps find the best fit for agricultural data from sensors in real time.
- Models for predicting crops often leave out environmental factors such as rainfall, yet they are crucial for the success of crops. Adding rainfall data to your model allows your project to offer a more accurate approach to predicting crop yields.
- Lack of Application to Different Areas: Using data from selected crops or areas limits the use of their models in many agricultural contexts. Because the models are so specific, they may work in one soil type and climate, but fail to apply elsewhere. Although the system you developed can monitor local data, more testing in multiple locations would still be needed.
- Not Understanding the Model: Due to how complex deep neural networks are, it is not easy for farmers and experts in agriculture to understand their reasoning. Because the details are often hard to understand, users may be hesitant to rely on such systems. When analyzing several models, you could draw attention to a place where previous research is lacking in finding a balance between accuracy and being explainable.
- Agricultural AI systems are usually developed without farmers being involved in their design. If this happens, farmers may not use the new ideas or technologies as they do not work well for them. Technology has been the main area in your project, but literature often neglects this human-centered aspect that matters for the real world.

Chapter 3

PROPOSED METHOD AND SYSTEM DESIGN

- **Sensor Setup:** A seven-in-one multi parameter soil sensor is used to measure key metrics, including soil moisture, pH, temperature, electrical conductivity, nitrogen, phosphorus, and potassium levels.
- **Data Acquisition and Storage:** Data is collected at first via the raspberry pie but faced limitations as the sensor manual had not been available and the interface of raspberry pie in python then we moved to the second approach using **RS-485** Serial communication with our multiparameter soil sensor and then the data being fetched was not being saved in csv format and was in hexadecimal via the serial interfaces of cool term, docklight and modbus then afterwards the final technique was used in python via pymodbus and pyserial library after connecting it with sensor the values were fetched perfectly and also conversion was done to required metrics from hexadecimal and they were also stored in CSV format simultaneously.
- **Soil Sensor:** A seven-in-one multi parameter sensor that captures soil moisture, pH, temperature, electrical conductivity, nitrogen, phosphorus, and potassium levels.
  Computer: Equipped with an RS485 serial communication interface (e.g., via a USB-to-RS485 converter) to establish a connection with the sensor.

3.1 Pre-Processing:

- **Loading Data**: Data collected from sensors is loaded into a DataFrame for further analysis.

```
print(df)
```

```
           Crop  Temperature      N     P      K   pH   EC  Moisture
0         Wheat         17.2  115.4  22.1  130.8  6.5  1.2      62.4
1         Wheat         18.6  108.7  21.3  126.4  6.3  0.9      58.7
2         Wheat         15.9  123.2  23.8  135.1  6.7  1.5      65.2
3         Wheat         19.3  112.9  20.7  128.6  6.4  1.8      60.9
4         Wheat         16.4  119.5  22.6  132.3  6.6  1.0      67.1
..          ...          ...    ...   ...    ...  ...  ...       ...
540  Sunflower         22.8   98.1  21.7  108.5  6.4  0.8      58.3
541  Sunflower         23.2   87.6  18.9  116.3  6.6  1.3      58.7
542  Sunflower         21.5   93.9  23.4  112.7  6.1  1.5      55.2
543  Sunflower         24.7   90.4  20.6  119.8  6.8  1.0      61.9
544  Sunflower         20.7   96.3  22.1  107.9  6.3  0.7      57.4

[545 rows x 8 columns]
```

- **Checking for Missing Values**: The dataset is examined for missing values, ensuring data completeness.

```
[28] print(df.isnull().sum())
```

```
Crop           0
Temperature    0
N              0
P              0
K              0
pH             0
EC             0
Moisture       0
dtype: int64
```

- **Identifying Features**:
  - **Numerical Features**: Includes all soil and environmental parameters.

```
#numerical features
numerical_df = df.select_dtypes(include=['number'])
print(numerical_df.columns)
```

```
Index(['Temperature', 'N', 'P', 'K', 'pH', 'EC', 'Moisture'], dtype='object')
```

  - **Categorical Features**: The crop type is classified as a categorical feature, useful for training the classification model.

```
#categorical features
categorical_df = df.select_dtypes(include=['object', 'category'])
print(categorical_df.columns)
```

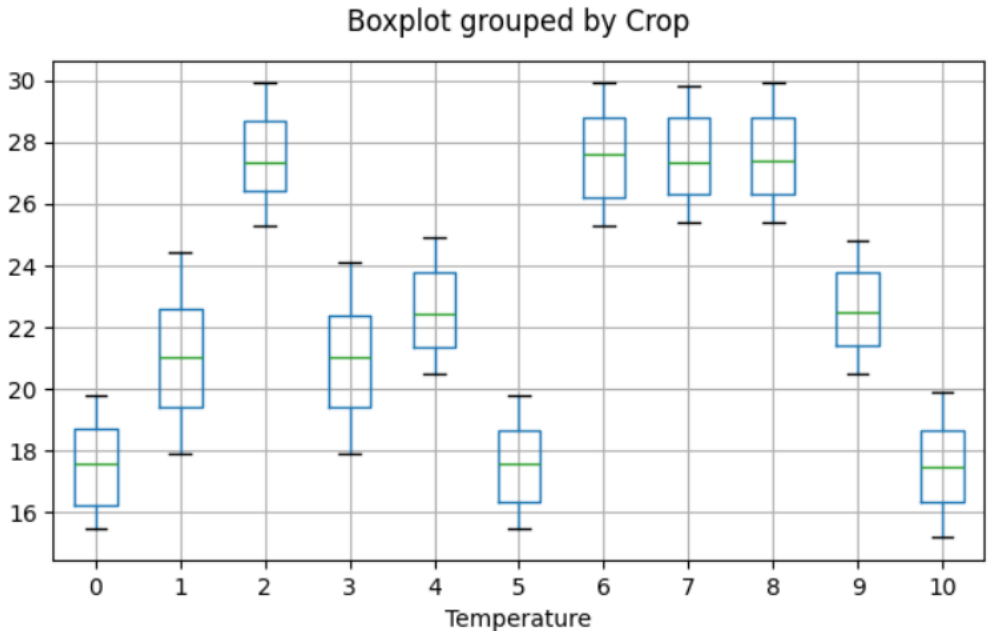
```
Index(['Crop'], dtype='object')
```

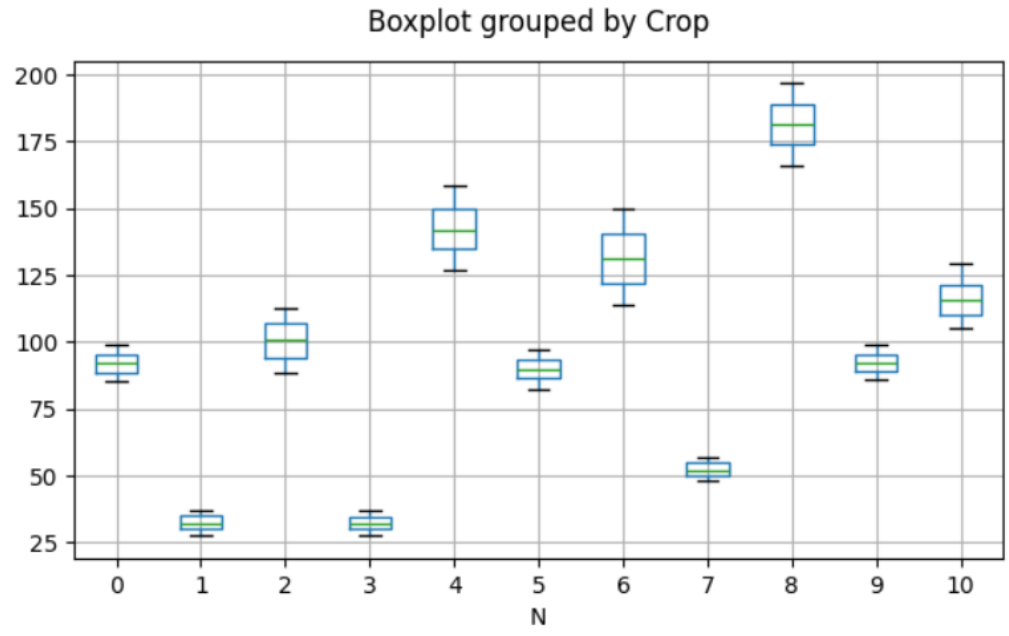
  - **Label Encoding**: The target variable (crop type) is label-encoded to convert categorical labels into numerical values.

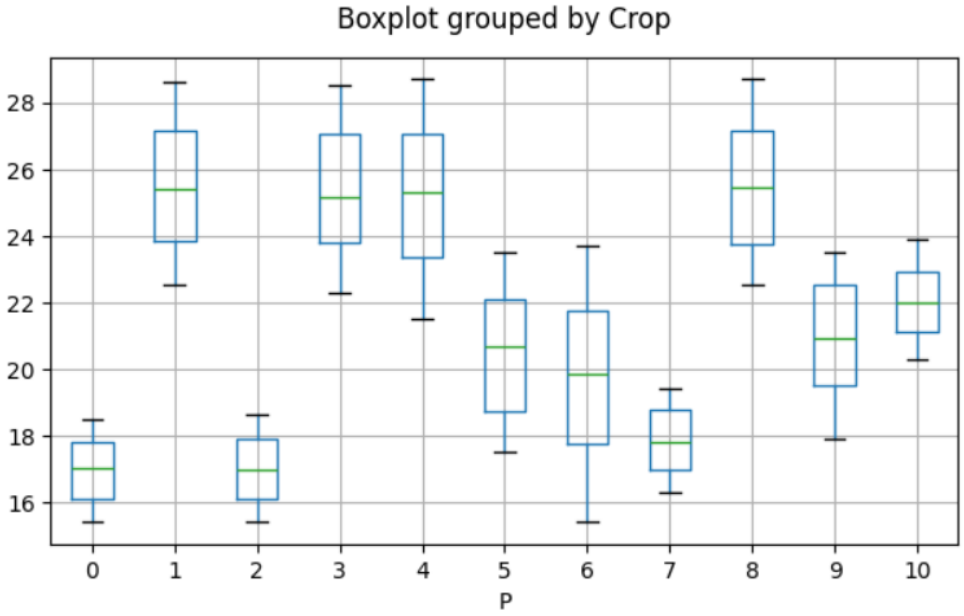
```
labels = list(df['Crop'].unique())
print(labels)
le_data = df.copy()
label_encoder = preprocessing.LabelEncoder()
le_data['Crop'] = label_encoder.fit_transform(le_data['Crop'])
print(le_data)
```

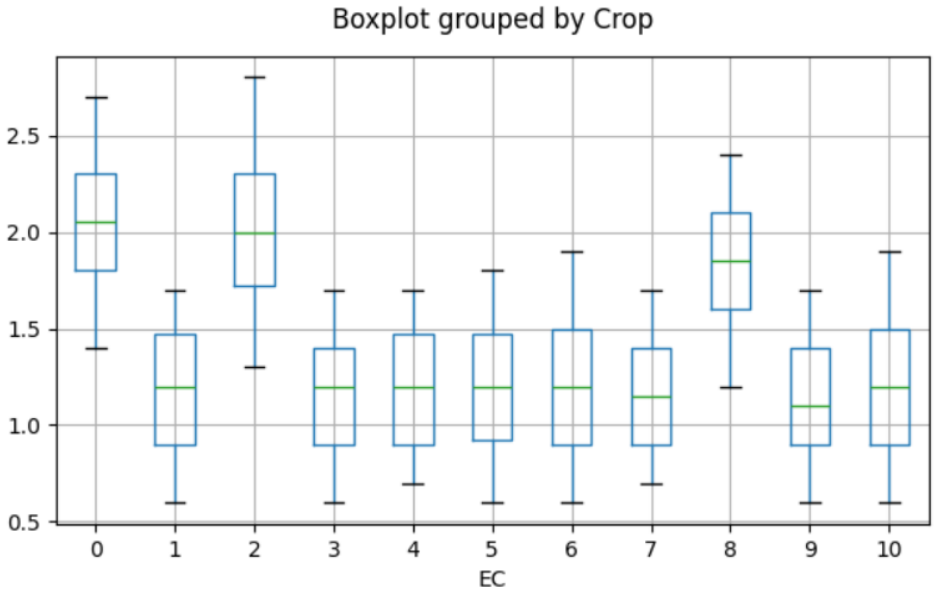
```
['Wheat', 'Rice', 'Cotton', 'Maize', 'Sugarcane', 'Barley', 'Chickpea', 'Lentil', 'Mustard', 'Sesame', 'Sunflower']
     Crop  Temperature      N     P      K   pH   EC  Moisture
0      10         17.2  115.4  22.1  130.8  6.5  1.2      62.4
1      10         18.6  108.7  21.3  126.4  6.3  0.9      58.7
2      10         15.9  123.2  23.8  135.1  6.7  1.5      65.2
3      10         19.3  112.9  20.7  128.6  6.4  1.8      60.9
4      10         16.4  119.5  22.6  132.3  6.6  1.0      67.1
..    ...          ...    ...   ...    ...  ...  ...       ...
540     9         22.8   98.1  21.7  108.5  6.4  0.8      58.3
541     9         23.2   87.6  18.9  116.3  6.6  1.3      58.7
542     9         21.5   93.9  23.4  112.7  6.1  1.5      55.2
543     9         24.7   90.4  20.6  119.8  6.8  1.0      61.9
544     9         20.7   96.3  22.1  107.9  6.3  0.7      57.4

[545 rows x 8 columns]
```

- **Data Visualization for Exploration**:
  - **Box Plots:**

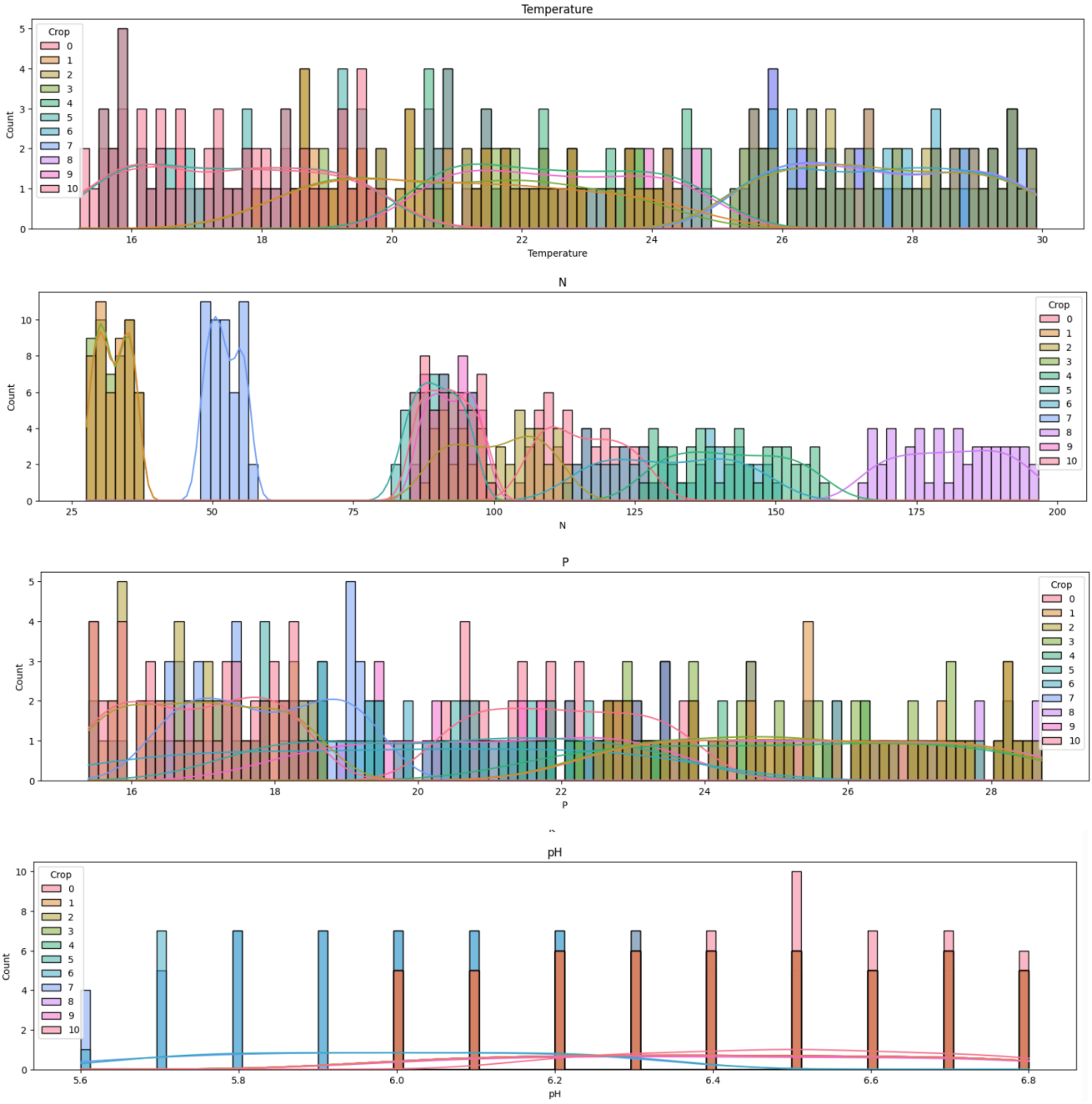


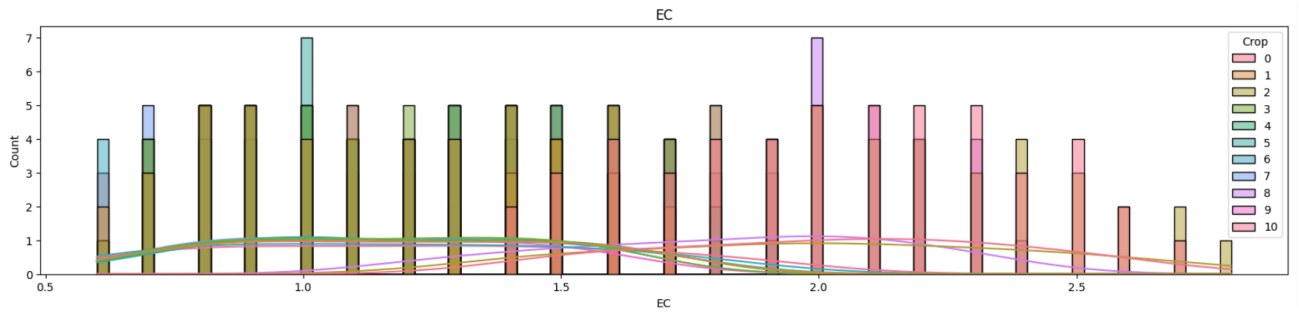



Boxplot grouped by Crop
28
26
24
22
20
18
16
0
1
2
3
4
5
6
7
8
9
10
P


Boxplot grouped by Crop
2.5
2.0
1.5
1.0
0.5
0
1
2
3
4
5
6
7
8
9
10
EC

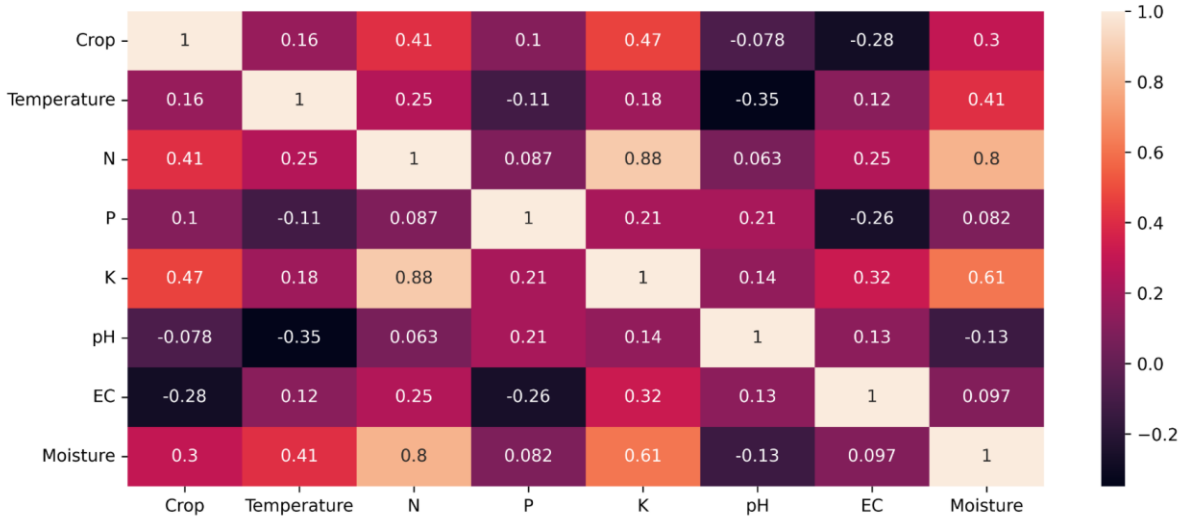
- **Histograms:**


Temperature
Crop
0
1
2
3
4
5
6
7
8
9
10
Count
Temperature
N
Count
N
P
Count
P
pH
Count
pH



- **Correlation Analysis**

  A heatmap displays feature correlations, highlighting any strong associations between soil properties and crop types.



3.2 Step-wise Explanation

3.2.1. Data Collection

Collect real-time data from the multiparameter sensor using serial communicator RS 485 via python through pymodbus and pyserial library and store values after being fetched in csv format

3.2.2. Pre-Process Data by Handling Missing Values, Normalization and Encoding

- **Encoding**: The LabelEncoder from scikit-learn is applied to the Crop column, transforming categorical crop names (e.g., Wheat, Maize etc) into numerical values suitable for machine learning models.
- **Missing Values**: The notebook does not explicitly address missing values. In practice, you would typically inspect the dataset for missing entries and handle them by imputation (e.g., using mean/median) or removal, depending on the data quality.

3.2.3. Data Split

The dataset is divided into training and testing sets using train_test_split from scikit-learn. This split allows the models to be trained on a subset of the data (X_train, y_train) while reserving another portion (X_test, y_test) for evaluating their performance on unseen data, ensuring an unbiased assessment of generalization

```
x_train shape: (381, 7)
x_test  shape: (164, 7)
y_train shape: (381,)
y_test  shape: (164,)
```

3.2.4. Model Training

The notebook implements training for multiple machine learning models, including:

- **Random Forest** (rf_model): An ensemble method that builds multiple decision trees for robust predictions.
- **Support Vector Machine** (svm_model): A classifier that finds an optimal hyperplane to separate classes.
- **K-Nearest Neighbors** (KNN): A model that classifies based on proximity to labeled examples.
- **Naive Bayes**: A probabilistic classifier based on Bayes' theorem.
- **Logistic Regression** (LR): A linear model for classification (assumed to be included based on your outline). Each model is trained on the preprocessed training data to predict crop labels from sensor inputs.

3.2.5. Model Evaluation

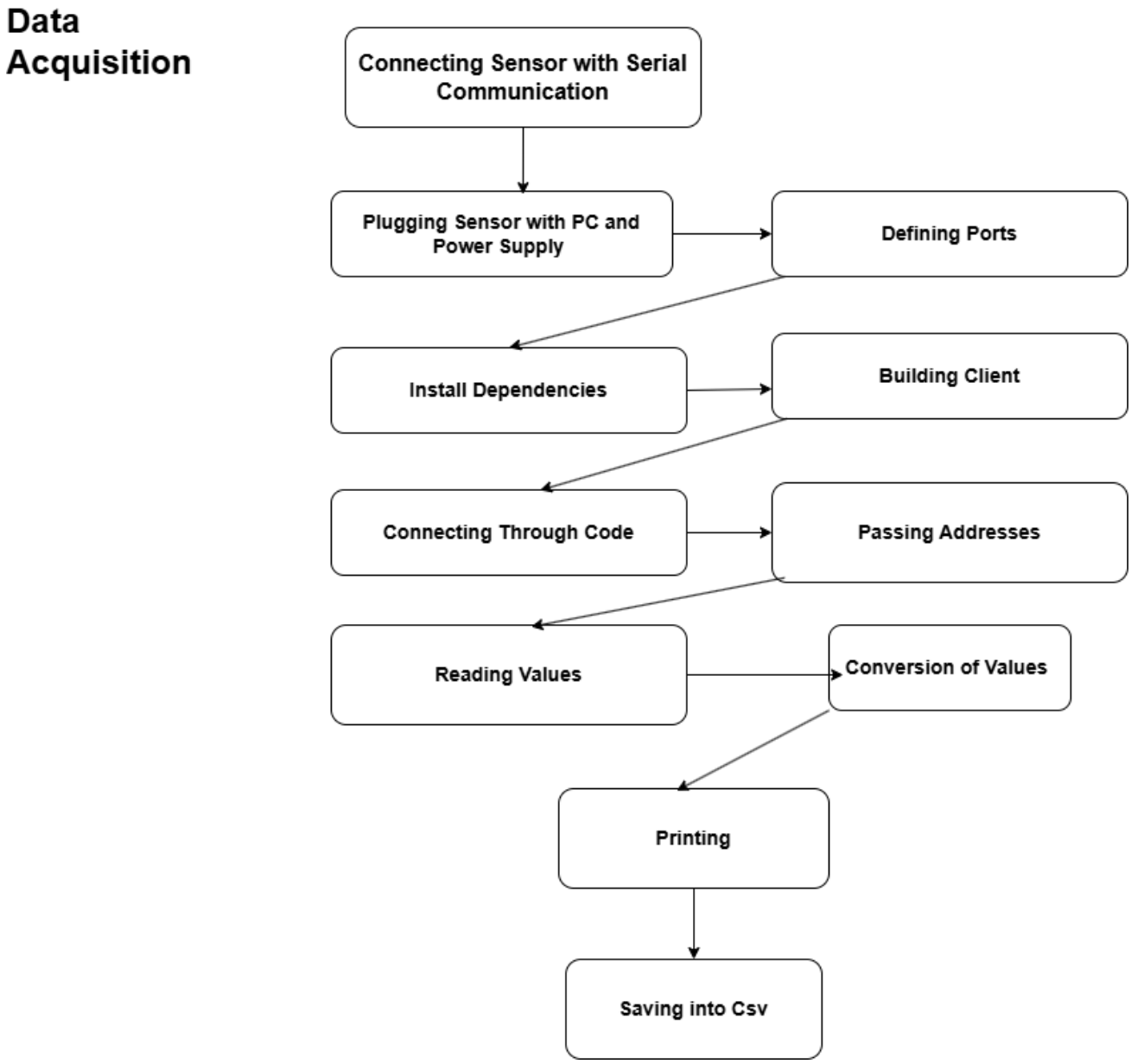
Model performance is assessed using standard metrics such as accuracy, precision, recall, and F1-score, calculated on the test set. The notebook visualizes these results with a bar plot created using Seaborn, enabling a clear comparison of model effectiveness. Based on this evaluation, the best-performing model (e.g., Random Forest, if it achieves the highest score) is selected for deployment, allowing it to predict the most suitable crop based on new sensor data inputs.

3.3 Monitoring and Maintenance

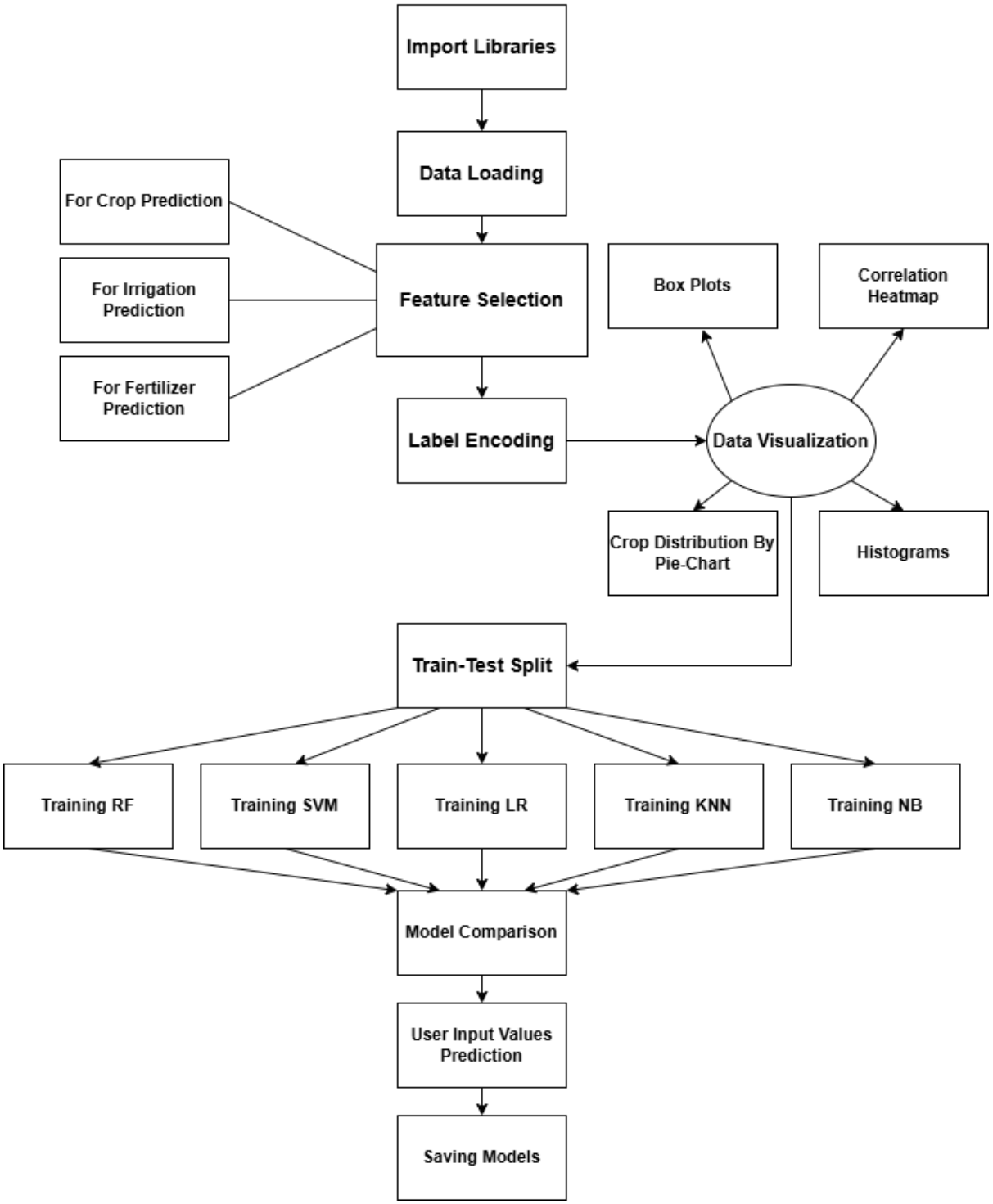
The system is designed to receive data from sensors, keeping it updated with the latest environmental and soil conditions. To ensure model accuracy, we have to run the models repeatedly. So that we can find out if a model is getting biased towards a value.

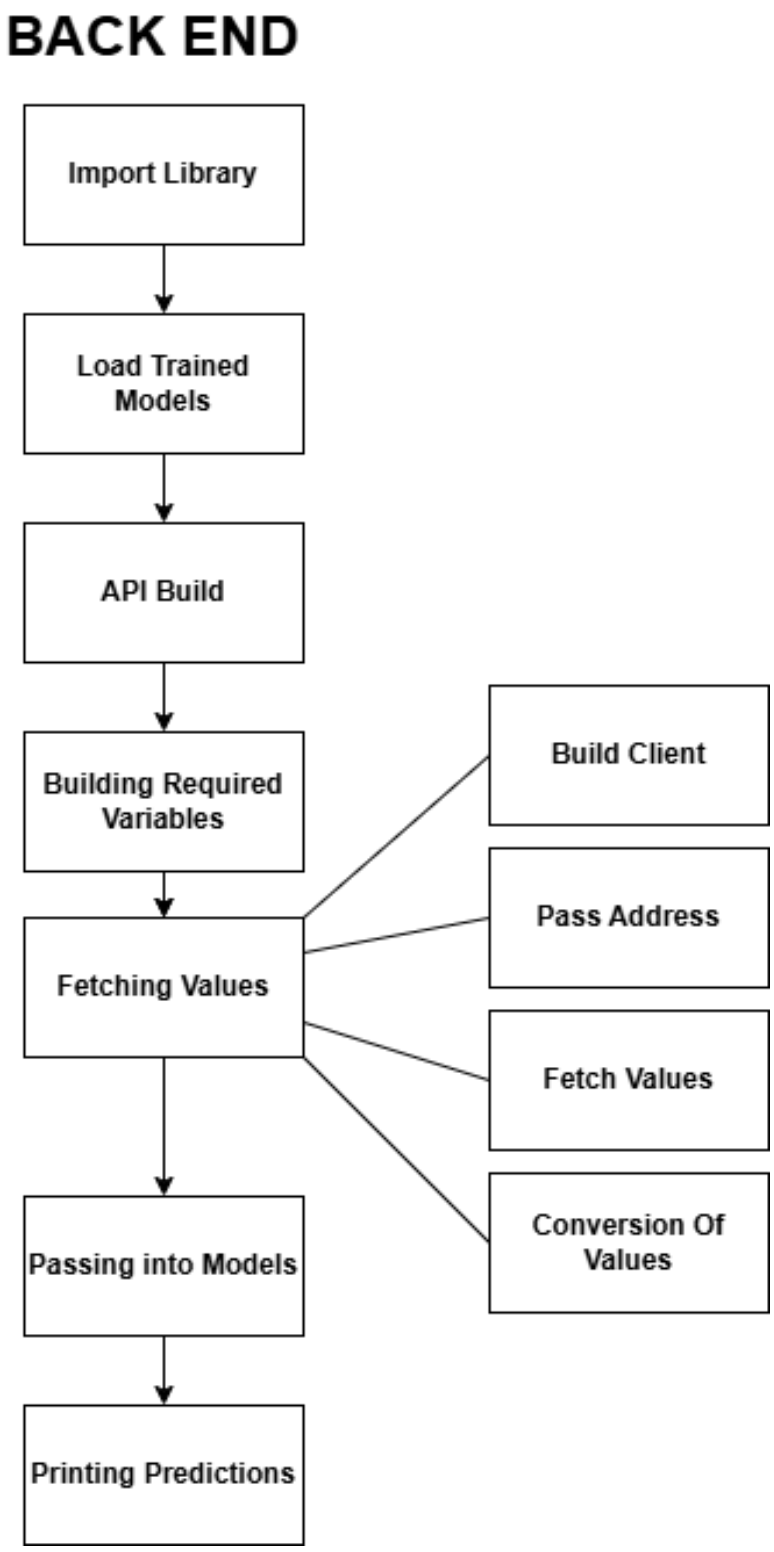
3.4 System Design:

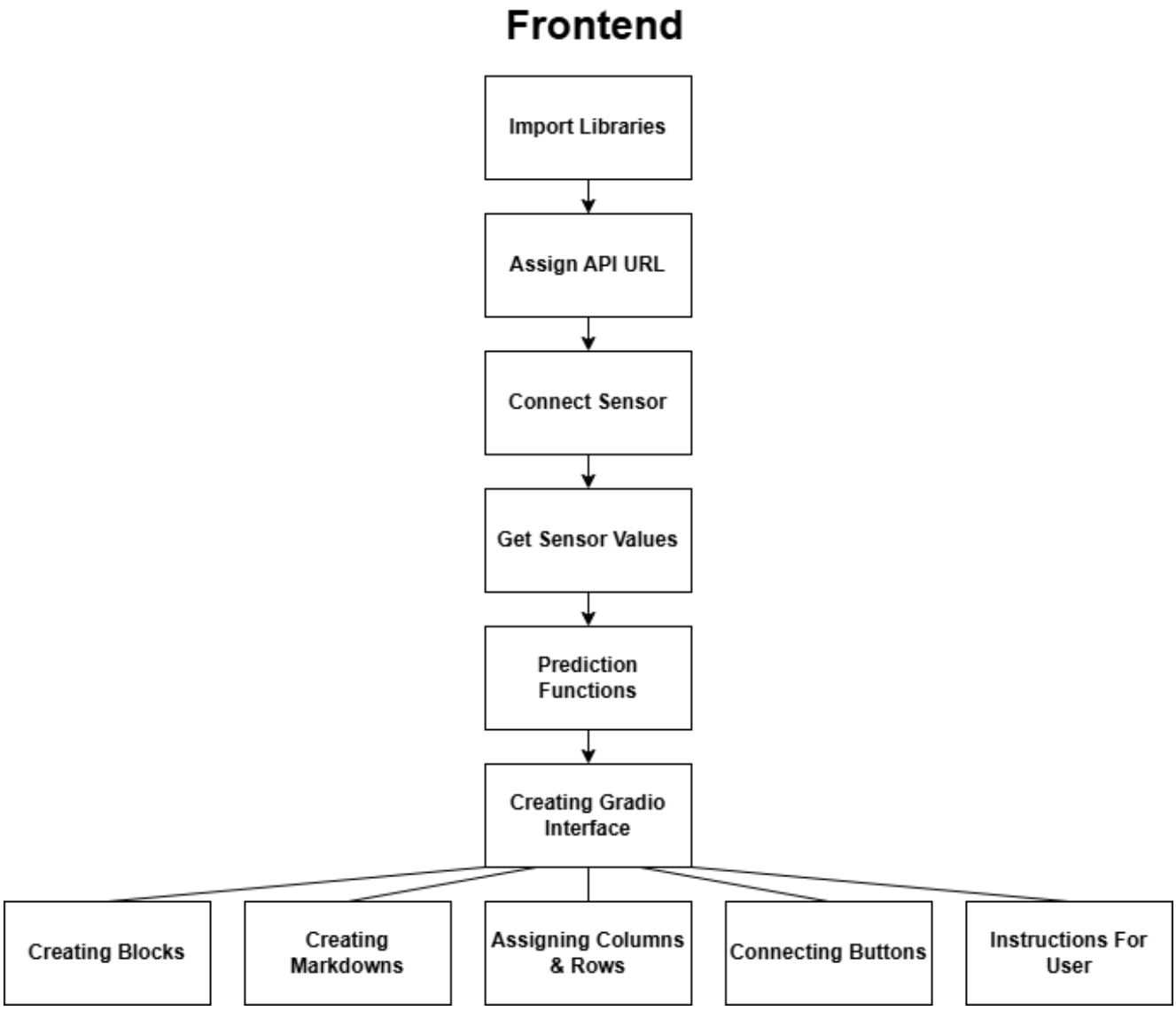
High-Level System Architecture/Component Overview Diagram: The architecture comprises data collection, model training, backend, and frontend.



MODEL TRAINING




BACK END
Import Library
Load Trained Models
API Build
Building Required Variables
Fetching Values
Passing into Models
Printing Predictions
Build Client
Pass Address
Fetch Values
Conversion Of Values


Frontend
Import Libraries
Assign API URL
Connect Sensor
Get Sensor Values
Prediction Functions
Creating Gradio Interface
Creating Blocks
Creating Markdowns
Assigning Columns & Rows
Connecting Buttons
Instructions For User

Chapter 4

PROTOTYPE DEVELOPMENT AND TESTING

4.1 Project Implementation

The project implementation was carried out in Python due to its extensive libraries for machine learning and data processing, including**:**

- **scikit-learn:** Data preprocessing, data partitioning, transforming labels and learning from data using Random Forest, Naive Bayes and kNN are all part of the procedure. Additionally, it allowed researchers to evaluate their models.

- **pandas**: Used to handle the importing and modification of data. As a result, I could select which data to keep, add new features, and set everything up properly for training and review.

- **Matplotlib, Seaborn:** Used to display data as histograms, circles, graphs and scatter plots. Plots made the information clearer and assisted in determining how to preprocess the data.
- **Pymodbus, Pyserial:** Used to connect the sensor and fetch values.
- **Numpy**: Fundamental package for numerical computing in Python, offering support for arrays, matrices, and mathematical functions.
- **Pydantic**: Used for data validation and settings management using Python type annotations, often with FastAPI.
- **FastAPI**: A modern, fast (high-performance) web framework for building APIs with Python 3.7+ based on standard Python type hints.
- **Uvicorn**: A lightning-fast ASGI server implementation, often used to serve FastAPI applications.
- **Typing**: Provides type hints for Python code, improving code readability and static analysis (built-in from Python 3.5+).
- **Gradio**: A library for quickly creating user interfaces (web apps) to demo machine learning models and data science workflows.
- **Requests**: A simple, elegant HTTP library for sending HTTP/1.1 requests, used for interacting with web services/APIs.

4.1.1. Data Loading and Preprocessing

The dataset was loaded using pandas from CSV files. Initial inspection included checking the shape of the dataset, column types, and missing values. Numerical features were identified for further processing.

Categorical label Crop was encoded into numeric values using LabelEncoder from scikit-learn to prepare them for machine learning models.

4.1.2. Outlier Removal

As our dataset has no outliers so we didn't have to use outlier removal.

4.1.3. Data Visualization and Analysis

Various visualizations were generated to explore the dataset:

- Boxplots were used to inspect the distribution and identify outliers for each numerical feature by crop type.
- Histograms with KDE curves showed the density and distribution of each feature across different crop types.
- A correlation heatmap was used to analyze relationships among features.
- A pie chart was created to illustrate the proportion of different crop types in the dataset.

These plots provided valuable insights into the feature distributions, helped validate preprocessing steps, and revealed patterns in the data.

4.1.4. Dataset Splitting

As our tool is capable of providing three types of predictions (Crop, Irrigation, Fertilizers), so we have to split our dataset according to three different type of feature requirements to predict the type of prediction. For first prediction we are using all features present in dataset, for second prediction we have used temp, pH, EC to predict moisture and for third we have used temperature, moisture, EC and pH

4.2 Hardware and Software Resources Used

Seven-in-One Multiparameter Soil Sensor: This sensor measures key soil parameters such as moisture, pH, temperature, nitrogen, phosphorus, and potassium, and connects to the RS-485 for real-time data collection.
Docklight Serial RS485: Functions as the central processing unit for data acquisition, storage, and initial processing of sensor data. It provides with a way to connect with the sensor using its serial communication.
Software Resources
The following software environments, libraries, and tools were used throughout the project:

- **Programming Language**: Python 3.10
- **Development Environment**: Visual Studio Code
- **Libraries and Frameworks**:
    - **pandas**: For data manipulation and preprocessing.
    - **NumPy**: For numerical operations and efficient array handling.
    - **Matplotlib & Seaborn**: For data visualization and exploratory data analysis.
    - **scikit-learn**: For implementing and evaluating classical machine learning algorithms such as Random Forest, Naive Bayes, and k-Nearest Neighbors.
    - **Pymodbus, Pyserial**: For connecting with the sensor and fetching values.
    - **FastAPI**: Web framework for building APIs.

- **Pydantic**: Used for data validation and settings management using Python type annotations, often with FastAPI.
- **Uvicorn**: A lightning-fast ASGI server implementation, often used to serve FastAPI applications.
- **Typing**: Provides type hints for Python code, improving code readability and static analysis (built-in from Python 3.5+).
- **Gradio**: A library for quickly creating user interfaces (web apps) to demo machine learning models and data science workflows.
- **Requests**: A simple, elegant HTTP library for sending HTTP/1.1 requests, used for interacting with web services/APIs.

4.3 Code Structure and Modules

The project's main modules focus on separate steps of the machine learning process by handling specific needs.

4.3.1. Data Pre-Processing Module:

- Checking for missing values.
- Check for outliers.
- Check for class imbalance.
- Scatters the crop types into numbers that can be handled by the model during training.

```
          Crop  Temperature      N     P      K   pH   EC  Moisture
0        Wheat         17.2  115.4  22.1  130.8  6.5  1.2      62.4
1        Wheat         18.6  108.7  21.3  126.4  6.3  0.9      58.7
2        Wheat         15.9  123.2  23.8  135.1  6.7  1.5      65.2
3        Wheat         19.3  112.9  20.7  128.6  6.4  1.8      60.9
4        Wheat         16.4  119.5  22.6  132.3  6.6  1.0      67.1
..         ...          ...    ...   ...    ...  ...  ...       ...
540  Sunflower         22.8   98.1  21.7  108.5  6.4  0.8      58.3
541  Sunflower         23.2   87.6  18.9  116.3  6.6  1.3      58.7
542  Sunflower         21.5   93.9  23.4  112.7  6.1  1.5      55.2
543  Sunflower         24.7   90.4  20.6  119.8  6.8  1.0      61.9
544  Sunflower         20.7   96.3  22.1  107.9  6.3  0.7      57.4

[545 rows x 8 columns]
(545, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Crop         545 non-null    object
 1   Temperature  545 non-null    float64
 2   N            545 non-null    float64
 3   P            545 non-null    float64
 4   K            545 non-null    float64
 5   pH           545 non-null    float64
 6   EC           545 non-null    float64
 7   Moisture     545 non-null    float64
dtypes: float64(7), object(1)
memory usage: 34.2+ KB
None
Crop           0
Temperature    0
N              0
P              0
K              0
pH             0
EC             0
Moisture       0
dtype: int64
Index(['Crop', 'Temperature', 'N', 'P', 'K', 'pH', 'EC', 'Moisture'], dtype='object')
```

4.3.2. Modelling Module:

- Contains implementations for the five models: Logistic Regression,Random Forest, Naive Bayes, SVM and KNN (using scikit-learn).
- Each model is tuned and configured according to the specific needs of the project, with hyperparameter tuning performed to optimize performance.

Logistic Regression:

```
# --- Logistic Regression ---
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(solver="liblinear").fit(X_train, y_train)
print(lr_model)
from sklearn.metrics import classification_report, confusion_matrix
prediction_lr = lr_model.predict(X_test)
print('Predicted labels: ', np.round(prediction_lr)[:10])
print('Actual labels   : ', y_test[:10])
print(classification_report(y_test, prediction_lr))
cm = confusion_matrix(y_test, prediction_lr)
print(cm)
ax = sns.heatmap(cm, annot=True, cmap='Blues')
ax.set_title('Confusion Matrix\n\n')
ax.set_xlabel('\nPredicted Category')
ax.set_ylabel('Actual Category ')
plt.show()
```

Naïve Bayes:

```
# --- Naive Bayes ---
from sklearn.naive_bayes import GaussianNB
gnb_model = GaussianNB()
gnb_model.fit(X_train, y_train)
prediction_nb = gnb_model.predict(X_test)
print('Predicted labels: ', np.round(prediction_nb)[:10])
print('Actual labels   : ', y_test[:10])
from sklearn.metrics import accuracy_score
print('Accuracy: ', accuracy_score(y_test, prediction_nb))
print(classification_report(y_test, prediction_nb))
cm = confusion_matrix(y_test, prediction_nb)
print(cm)
ax = sns.heatmap(cm, annot=True, cmap='Blues')
ax.set_title('Confusion Matrix\n\n')
ax.set_xlabel('\nPredicted Category')
ax.set_ylabel('Actual Category ')
plt.show()
# --- End Naive Bayes ---
```

Random Forest:

```
# --- Random Forest ---
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
prediction_rf = rf_model.predict(X_test)
print('Predicted labels: ', np.round(prediction_rf)[:10])
print('Actual labels   : ', y_test[:10])
print('Accuracy: ', accuracy_score(y_test, prediction_rf))
print(classification_report(y_test, prediction_rf))
cm = confusion_matrix(y_test, prediction_rf)
print(cm)
ax = sns.heatmap(cm, annot=True, cmap='Blues')
ax.set_title('Confusion Matrix\n\n')
ax.set_xlabel('\nPredicted Category')
ax.set_ylabel('Actual Category ')
plt.show()
# --- End Random Forest ---
```

KNN:

```
# --- k-Nearest Neighbour ---
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
prediction_knn = knn_model.predict(X_test)
print('Predicted labels: ', np.round(prediction_knn)[:10])
print('Actual labels   : ', y_test[:10])
print('Accuracy: ', accuracy_score(y_test, prediction_knn))
print(classification_report(y_test, prediction_knn))
cm = confusion_matrix(y_test, prediction_knn)
print(cm)
ax = sns.heatmap(cm, annot=True, cmap='Blues')
ax.set_title('Confusion Matrix\n\n')
ax.set_xlabel('\nPredicted Category')
ax.set_ylabel('Actual Category ')
plt.show()
# --- End kNN ---
```

SVM:

```
# --- SVM ---
from sklearn.svm import SVC
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print(cm)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix - SVM")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
# --- End SVM ---
```

4.4 Integration

In this project, the various machine learning modules were integrated so they could process available data and provide the results needed for soil-based crop classification.
This project was constructed and executed using VS code and Colaboratory which made it easy to interact and merge different elements. The system was integrated according to the following steps:

4.4.1. Data and Feature Integration

During the preprocessing step, Nitrogen, Phosphorus, Potassium, pH, moisture, temperature and electrical conductivity features were integrated with pandas into a DataFrame.

4.4.2. Modeling Integration

To implement the Random Forest, Naive Bayes, SVM, LR and KNN type of machine learning models in Python, scikit-learn was used. The encoded data was effortlessly fed into the training pipeline.

- Rescaling the features (when needed)
- Separating the whole data into training and test sets
- Models are fitted and used to predict future outcomes.

Once the data was processed, the same data structure could be used to train every model in the system.

4.4.3. Visualization Integration

Dynamic interpretation and analysis were made easy because Matplotlib and Seaborn visualizations were included in the notebook. This means that any time the data was updated, the charts automatically adjusted to reflect the latest changes.

4.4.4. Evaluation Integration

Evaluation metrics such as accuracy scores and confusion matrices were computed within the same environment using scikit-learn's evaluation tools. This allowed for direct comparison between models without needing to export or process data externally.

4.4.5. User Workflow Integration

Every stage in the pipeline follows one after the other and is built as separate modules for processing and classification. Each notebook cell completed a task and the results from a module were sent to the next one. This ensured:

- Easy debugging
- Reproducibility
- Minimal manual intervention

4.5 Validation and System Testing

We had to validate and test the crop classification models to confirm that they were accurate, reliable and suitable for unknown data. Various ways of testing were done to make sure the system was effective.

4.5.1. Train-Test Split Validation

- We separated the whole dataset into training and testing sets by using the train_test_split from scikit-learn.
- Similar proportions of the various crop classes were maintained with the method of stratification.
- Usually, the data was split so that 80% was used for learning and 20% was used for testing.

4.5.2. Cross-validation (if applied)

- If possible, k-fold cross-validation was run to assess how well the model worked on different datagrams.
- Thanks to this, there was less chance of the model fitting the data too closely and its accuracy remained the same with any partition of the dataset.

4.5.3. Model Evaluation Metrics

After finishing the training process, the models were measured using important metrics that come with scikit-learn which include:

- Accuracy Score: This helps see how many of the predictions were accurate.
- False Positives: Represent the number of misidentified items from every class.
- Classification Report: Added precision, recall and F1-score to help with detailed analysis.

By analyzing these metrics, I selected the Random Forest algorithm since it performed better on the task.

4.5.4. Visualization for Result Verification

- Heatmaps and confusion matrices were used to show the results of the study.
- To check if the models followed the expected patterns, I used boxplots and distribution plots.

4.5.5. System Testing

The model was evaluated on a new sample of data to determine how it performs in real-life scenarios. Both accuracy and confusion matrices were used to monitor the system's ability to predict results. This was done to ensure the model would successfully meet the project's objectives before being used in practice.

4.6 Challenges Faced and Solutions

While creating the crop classification system, several challenges were faced at each stage of the process. Here is a guide to the main problems and their solutions provided by IFTA:

4.6.1. Handling Missing and Irrelevant Data

- **Challenge:** If a value is not converted properly from its hexadecimal form, model will surely not get trained properly or accurately. Instead, it will be biased towards only one crop.
- **Solution**: The data was cleaned by using pandas to search for null values. Moreover, data was correctly examined to check if there is any conversion missing. All necessary steps were taken to inspect the dataset and check that it was reliable before training the model.

4.6.2. Class Imbalance

- **Challenge**: Because the numbers of crops were not evenly distributed, it was possible for the model to make biased predictions.
- **Solution**: To solve class imbalance we can use SMOTE so the imbalanced class has more values but we had class imbalance in only one class and was only about 1% so our model works perfectly even without SMOTE.

4.6.3. Encoding Categorical Labels

- **Challenge:** Numbers are needed to train machine learning models, but the dataset we have contains categorical labels.
- **Solution**: To solve this, I used the LabelEncoder from scikit-learn to turn the labels into numbers. As a result, the labels were processed effectively even though the class relationships were not changed.

4.6.4. Outlier Detection and Removal

- **Challenge:** High or low data for N, P or K may bring mistakes to the calculated values.
- **Solution:** The outliers were checked but there were no outliers so we didn't have to remove them but if there are any we can use custom techniques like taking mean of values.

4.6.5. Feature Selection and Engineering

- **Challenge:** Anticipating the features of crops that matter a lot for classification was key.
- **Solution:** As for each model we required different features for predicting specific type of prediction. So we stored features differently for each model, before passing them to models.

4.6.6. Model Comparison and Selection

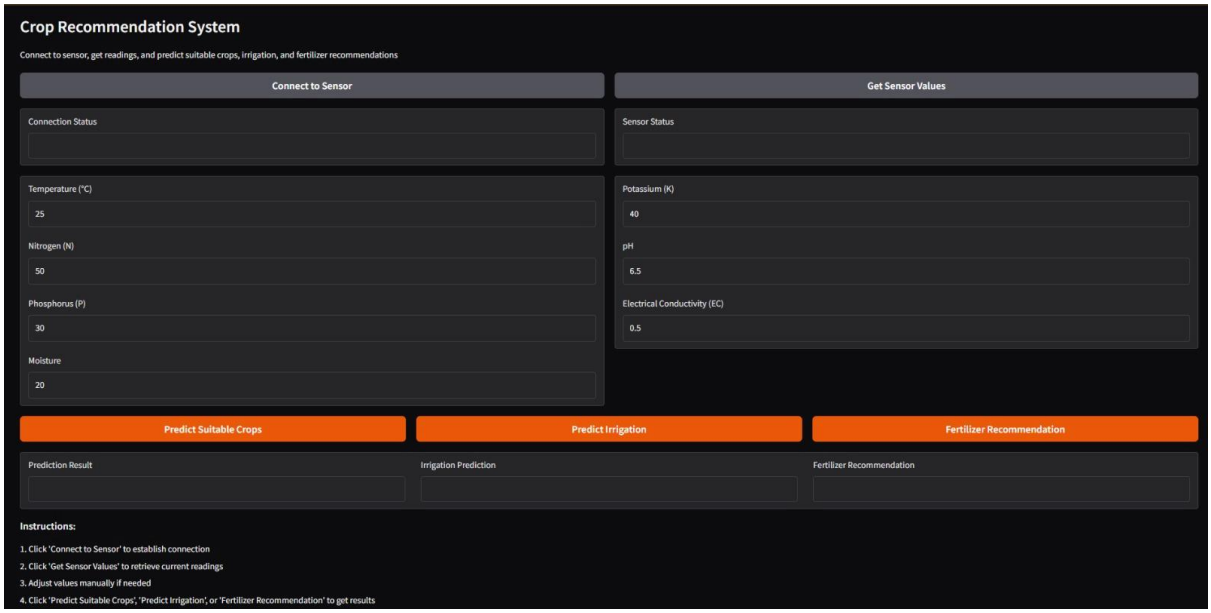
- **Challenge:** I had to assess several algorithms (Random Forest, Naive Bayes, KNN, SVM, LR) to pick the one that best suits my needs.
- **Solution**: The same data was used to train various models and their accuracy was measured along with evaluation by confusion matrices. As a result, I was able to compare and pick the model that performed best.

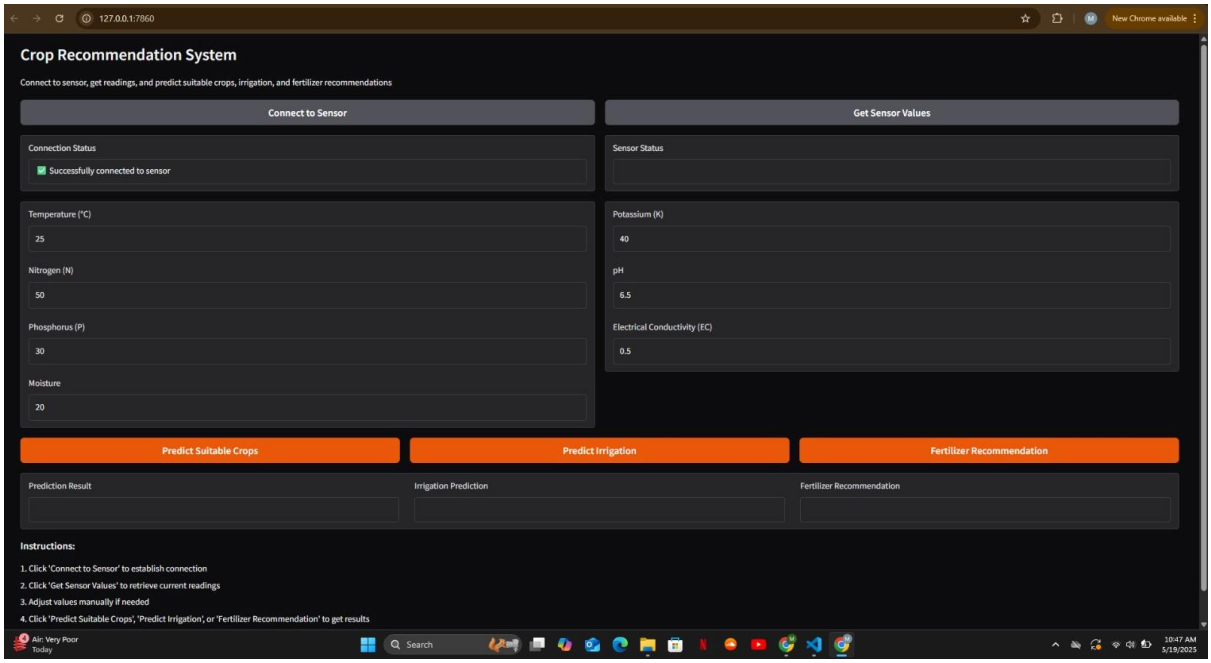
4.6.7. Visualization Clarity

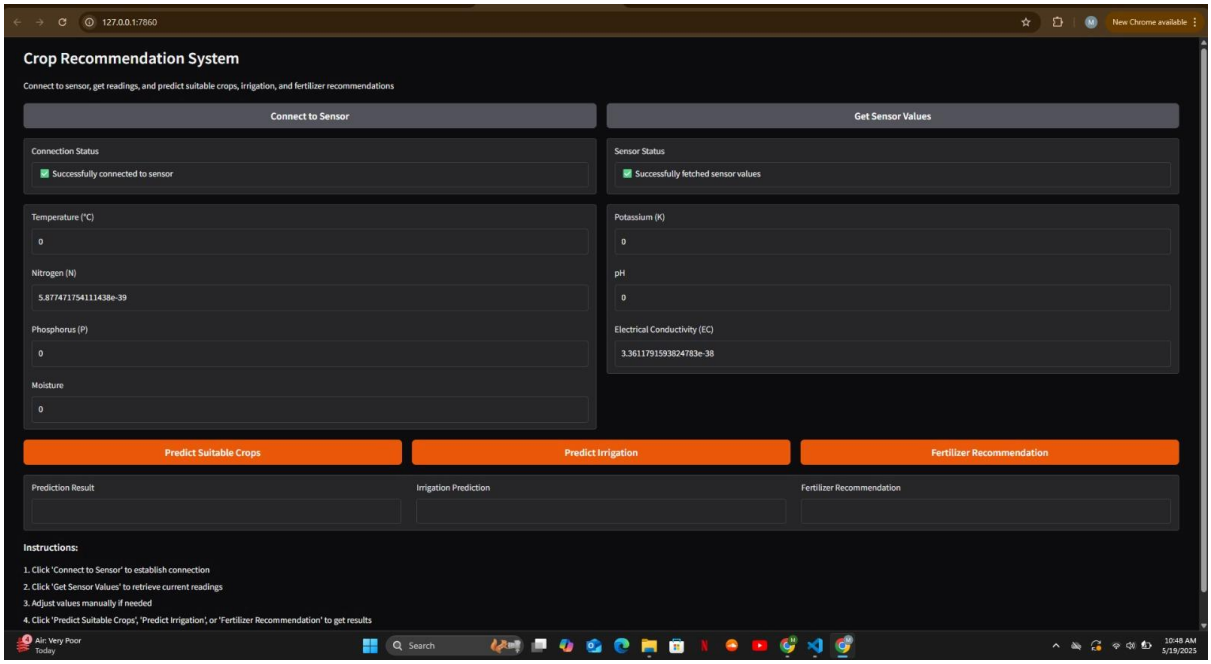
- **Challenge:** Effectively visualizing complex relationships between features and target variables for interpretation**.**
- **Solution:** Matplotlib and Seaborn were used to generate boxplots, heatmaps, histograms, and pie charts that provided clear insights into data patterns, distributions, and feature correlations.

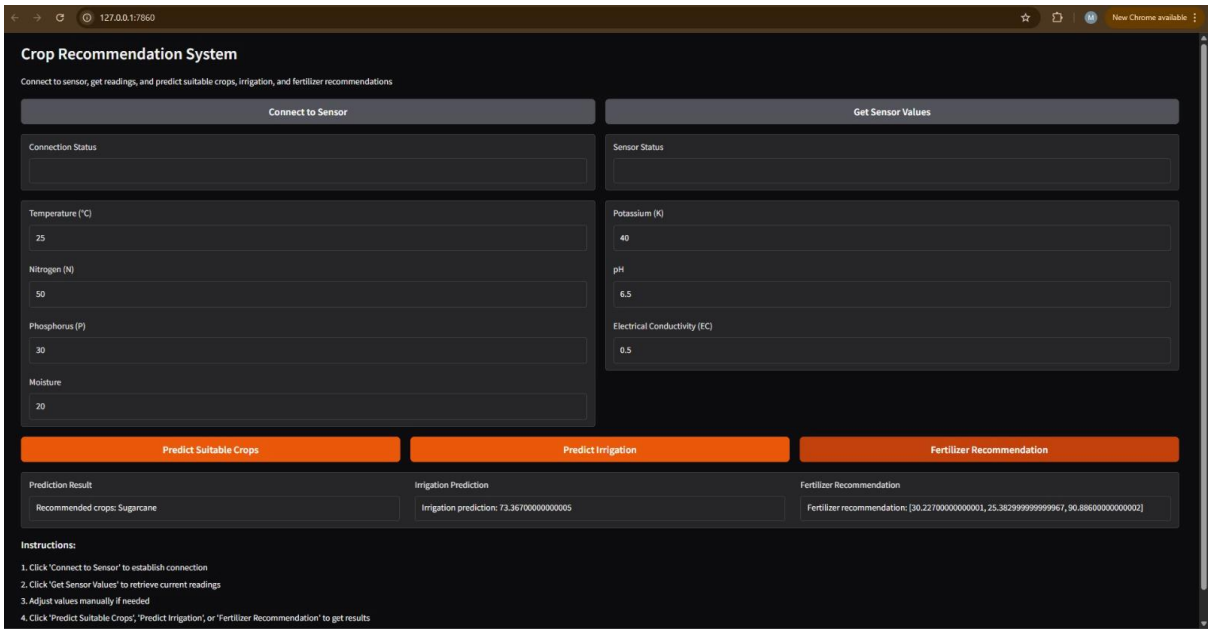
4.7 Web Based GUI:

- Challenge: Real time connection of sensor with GUI and conversion of values.
- Solution: by printing values and checking sensor connections at each step, so that problem identification is easy and can be solved quickly.






127.0.0.1:7860
New Chrome available
Crop Recommendation System
Connect to sensor, get readings, and predict suitable crops, irrigation, and fertilizer recommendations
Connect to Sensor
Get Sensor Values
Connection Status
Successfully connected to sensor
Sensor Status
Successfully fetched sensor values
Temperature (°C)
0
Nitrogen (N)
5.87747175411143 8e-39
Phosphorus (P)
0
Moisture
0
Potassium (K)
0
pH
0
Electrical Conductivity (EC)
3.3611791593824783e-38
Predict Suitable Crops
Predict Irrigation
Fertilizer Recommendation
Prediction Result
Irrigation Prediction
Fertilizer Recommendation
Instructions:
1. Click 'Connect to Sensor' to establish connection
2. Click 'Get Sensor Values' to retrieve current readings
3. Adjust values manually if needed
4. Click 'Predict Suitable Crops', 'Predict Irrigation', or 'Fertilizer Recommendation' to get results
Air: Very Poor
Today
Search
10:48 AM
5/19/2025


127.0.0.1:7860
New Chrome available
Crop Recommendation System
Connect to sensor, get readings, and predict suitable crops, irrigation, and fertilizer recommendations
Connect to Sensor
Get Sensor Values
Connection Status
Sensor Status
Temperature (°C)
25
Nitrogen (N)
50
Phosphorus (P)
30
Moisture
20
Potassium (K)
40
pH
6.5
Electrical Conductivity (EC)
0.5
Predict Suitable Crops
Predict Irrigation
Fertilizer Recommendation
Prediction Result
Recommended crops: Sugarcane
Irrigation Prediction
Irrigation prediction: 73.36700000000005
Fertilizer Recommendation
Fertilizer recommendation: [30.227000000000001, 25.382999999999967, 90.88600000000002]
Instructions:
1. Click 'Connect to Sensor' to establish connection
2. Click 'Get Sensor Values' to retrieve current readings
3. Adjust values manually if needed
4. Click 'Predict Suitable Crops', 'Predict Irrigation', or 'Fertilizer Recommendation' to get results

Chapter 5

Evaluation of Proposed Pipeline

5.1 Evaluation Methodology

To evaluate the performance and effectiveness of crop classification and fertility prediction by machine learning, an evaluation methodology was devised. Different testing methods and key measures were applied to ascertain that the models worked effectively and produced valuable outcomes.

5.1.1. Dataset Splitting

Division of the original data into training and testing subsets was done using the function train_test_split found in scikit-learn.

- **Training Set (80%)**: Used to train the machine learning models.
- **Testing Set (20%)**: Used to evaluate model performance on unseen data.

To keep the amount of every crop class equal in the datasets, stratified sampling was conducted.

5.1.2. Model Evaluation Metrics

To examine the performance of the models, the standard metrics came into play.

- **Accuracy**: The ratio of correct predictions to total predictions made. It served as a basic measure of overall performance.
- **Precision**: The number of true positives divided by the number of total predicted positives. To judge the model's performance, it was especially necessary to be precise when labeling specific crops.
- This measures the percentage of all true positives out of all the actual positives. It showed that all the key cases of the classes were being identified by the model.
- **F1-Score**: The harmonic mean of precision and recall, used to balance the trade-off between them—especially when class distributions were uneven.
- **Confusion Matrix**: A tabular visualization of actual vs. predicted labels across all classes, offering deeper insights into model misclassifications.

5.1.3. Algorithm Comparison

All the machine learning algorithms used the same data and were evaluated using the same procedure.

- **Random Forest Classifier**
- **Naive Bayes Classifier**
- **K-Nearest Neighbors (kNN)**

All the models were evaluated with the measures mentioned before. With this method, I could equally consider both the general accuracy and class-wise results, helping me select the best performing algorithm.

5.1.4. Visual Analysis

Evaluation results were further validated using visual tools such as:

- Using heatmaps helps to recognize the areas where the classifier succeeds and fails.
- Bar Charts can be used to view the precision, recall and F1-scores for every model and class.

5.2 Evaluation Environment

Hardware, software and technical details in the environment where machine learning models were developed are the evaluation environment. A stable environment was selected to enable reproducibility when performing every computation related to the project.

5.2.1. Hardware Environment

All the models were made and tested on a personal computer with the listed specifications.

- Processor: Intel Core i5, 10th Generation
- RAM: 16 GB DDR4
- Storage: 512 GB SSD + 1TB HDD
- Graphics: Dedicated Nvidia G-force Graphics
- Operating System: Windows 11 (64-bit)

With this system, traditional machine learning, data preparation and displaying results were possible with no need for high-performing components or GPU help.

5.2.2. Software Environment

I utilized the listed software tools and libraries throughout the evaluation of the model:

- **Programming Language:** Python 3.10
- **Development Interface:** Google Colab, Visual Studio Code, Jupyter Notebook (via Anaconda distribution)
- **Key Libraries:**
    - pandas and NumPy: For data handling and numerical operations
    - matplotlib and seaborn: For creating visualizations to support evaluation
    - scikit-learn: For model training, prediction, and performance evaluation
    - warnings, os, and joblib (if used): For environment management and model persistence

Using conda and pip, all the needed libraries were installed, allowing the code to run smoothly and without issues.

5.2.3. Execution and Runtime

- All experiments were run within Jupyter Notebook, enabling step-by-step execution, inline visualization, and easier debugging.
- The system maintained consistent performance across different runs, and outputs were reproducible due to controlled random seeds (if used).
- Since the work was local, none of the computing resources existed in the cloud.

5.3 Evaluation Results

After applying and testing the built machine learning models, their ability to foresee crop types from soil and environment was checked by measuring them with different metrics. The outcomes of each model are shown below:

5.3.1. Random Forest Classifier

The Random Forest algorithm delivered the highest performance among the evaluated models. It showed strong classification ability due to its ensemble nature and robustness to overfitting.

- **Accuracy**: 88% (approx.)
- **Precision, Recall, F1-Score**: High values across most crop classes
- **Confusion Matrix**: Minimal misclassification, especially for dominant crop labels
- **Strengths**:
    - Handles both numerical and categorical features well
    - Resistant to outliers
    - Works efficiently on medium-sized datasets

5.3.2. Naive Bayes Classifier

Naive Bayes performed reasonably well but slightly underperformed compared to Random Forest, particularly in handling classes with overlapping distributions.

- **Accuracy**: 90%
- **Precision and Recall**: Moderate performance on most classes
- **Confusion Matrix**: Some misclassification in closely related crop types
- **Strengths**:
    - Fast and simple
    - Effective on smaller datasets
- **Limitations**:

- Assumes feature independence, which may not hold in real-world datasets

5.3.3. K-Nearest Neighbors (KNN)

The KNN model provided competitive results but was sensitive to feature scaling and required careful tuning of the k parameter.

- **Accuracy**: 89%
- **Precision and Recall**: Good performance with a well-chosen k value (e.g., 5)
- **Confusion Matrix**: Occasional misclassification in edge cases
- **Strengths**:
  - Simple, intuitive algorithm
  - Effective when distance-based patterns are strong
- **Limitations**:
  - Sensitive to noisy data and outliers
  - Slower with larger datasets due to high computational cost during inference

5.3.4. Visual Comparison

Bar charts and heatmaps were used to visually compare the models using evaluation metrics. The Random Forest consistently outperformed the others across all major classes and metrics, establishing it as the preferred model for final deployment.

Summary of Results

Model	Accuracy	Strengths	Weaknesses
Random Forest	88%	High-accuracy, Low-overfitting, Scalable	Slightly more complex and less interpretable
Naive Bayes	90%	Simple, fast, good for small datasets	Assumes independence, moderate accuracy
K-Nearest Neighbors	89%	Intuitive, good with proper tuning	Sensitive to outliers, computationally expensive

The evaluation results confirm that **Random Forest** is the most suitable algorithm for this classification task, providing a strong balance between accuracy, robustness, and interpretability. It can be considered for future integration into an intelligent crop recommendation system.

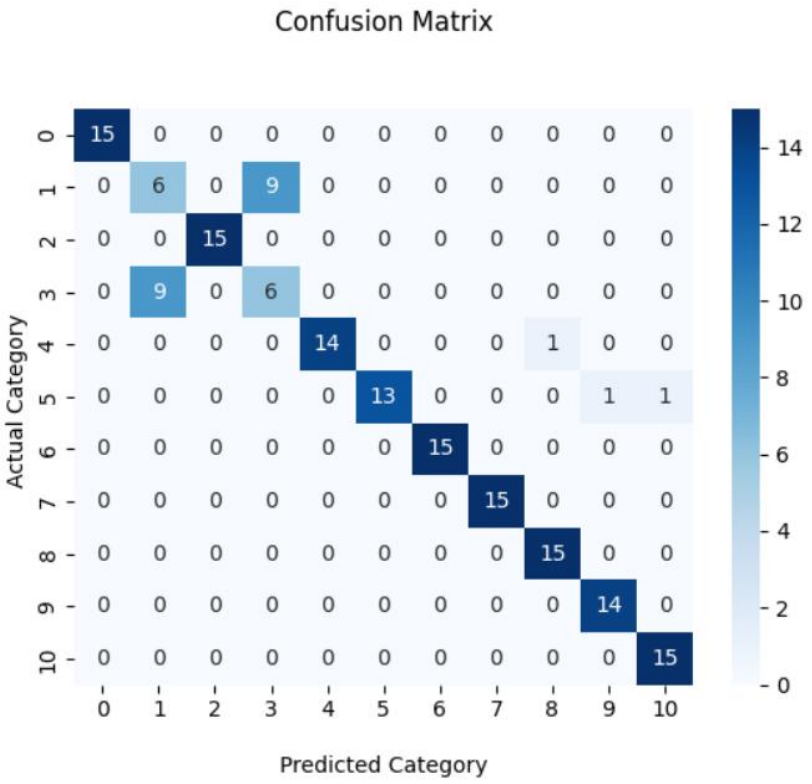
5.3.5 Model Results:

LR:

In crop prediction, the Linear Regression model got it right 88.41% of the time. For precision it was 88.45% and the recall was 88.41%. It got an F1 score of 88% for trade-off between precision and recall.

```
# Print results
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
print("F1 Score :", f1)
```

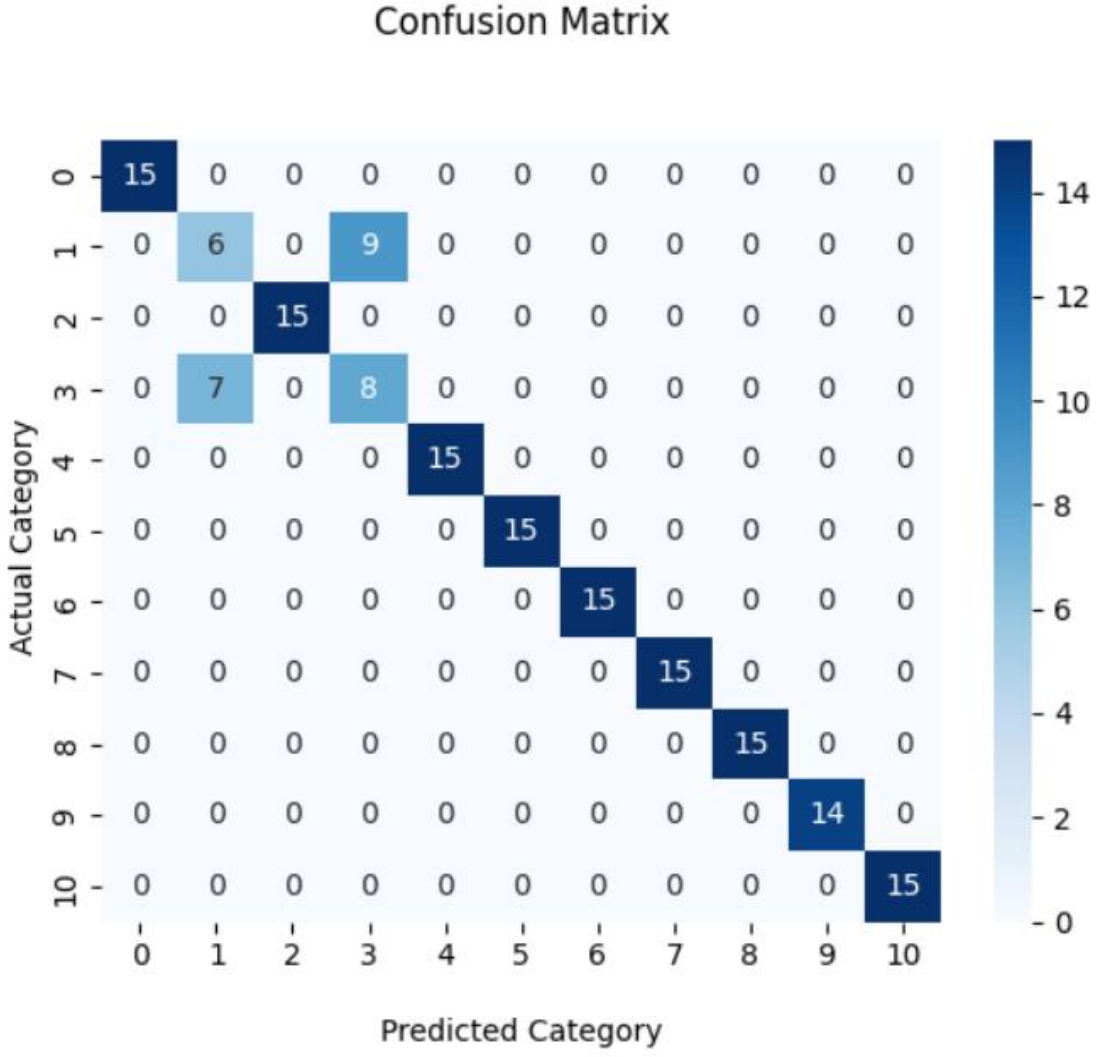
```
Accuracy : 0.8841463414634146
Precision: 0.8845274390243902
Recall   : 0.8841463414634146
F1 Score : 0.8841395588594372
```



NB:

In crop prediction, the NB model gives an accuracy of 90.24%. For precision it was right for 90.23% and for recall it was 90.24%. It had an F1 score of 90%.

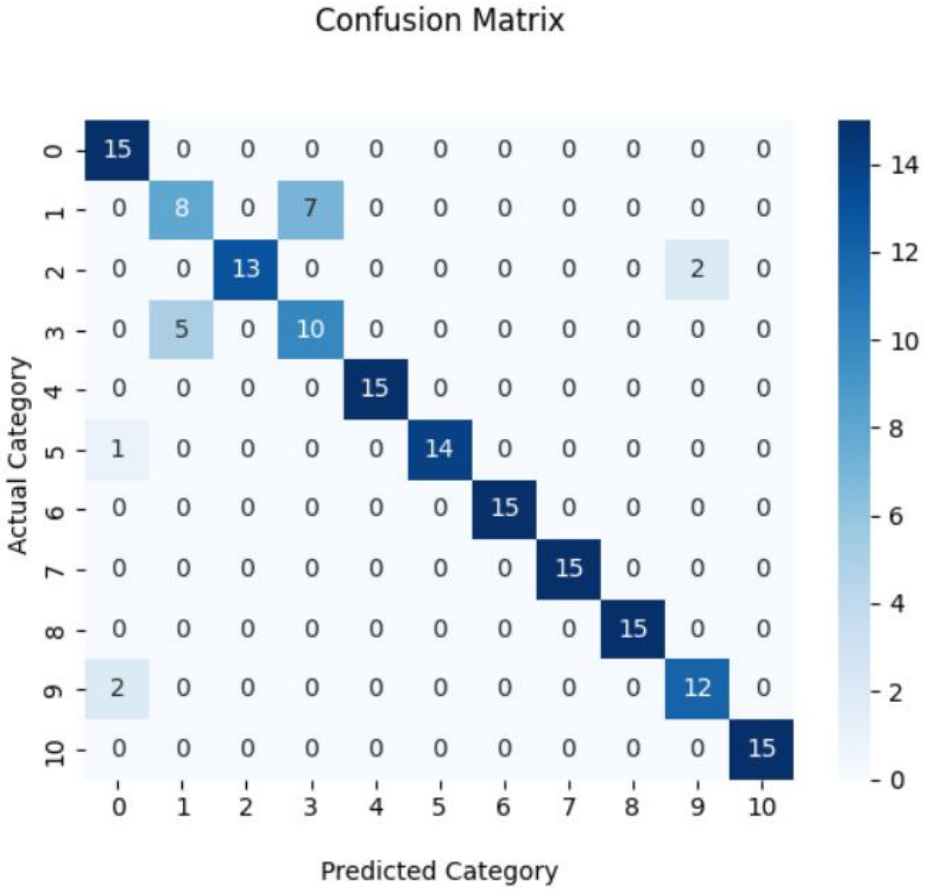
```
Accuracy : 0.9024390243902439
Precision: 0.9023286612956627
Recall   : 0.9024390243902439
F1 Score : 0.9020034843205575
```



KNN:

In crop prediction, the KNN model gives the accuracy of 89.6%. For precision it was right for 89.9% and the recall was 89.6%. It got an F1 score of 89.6%.

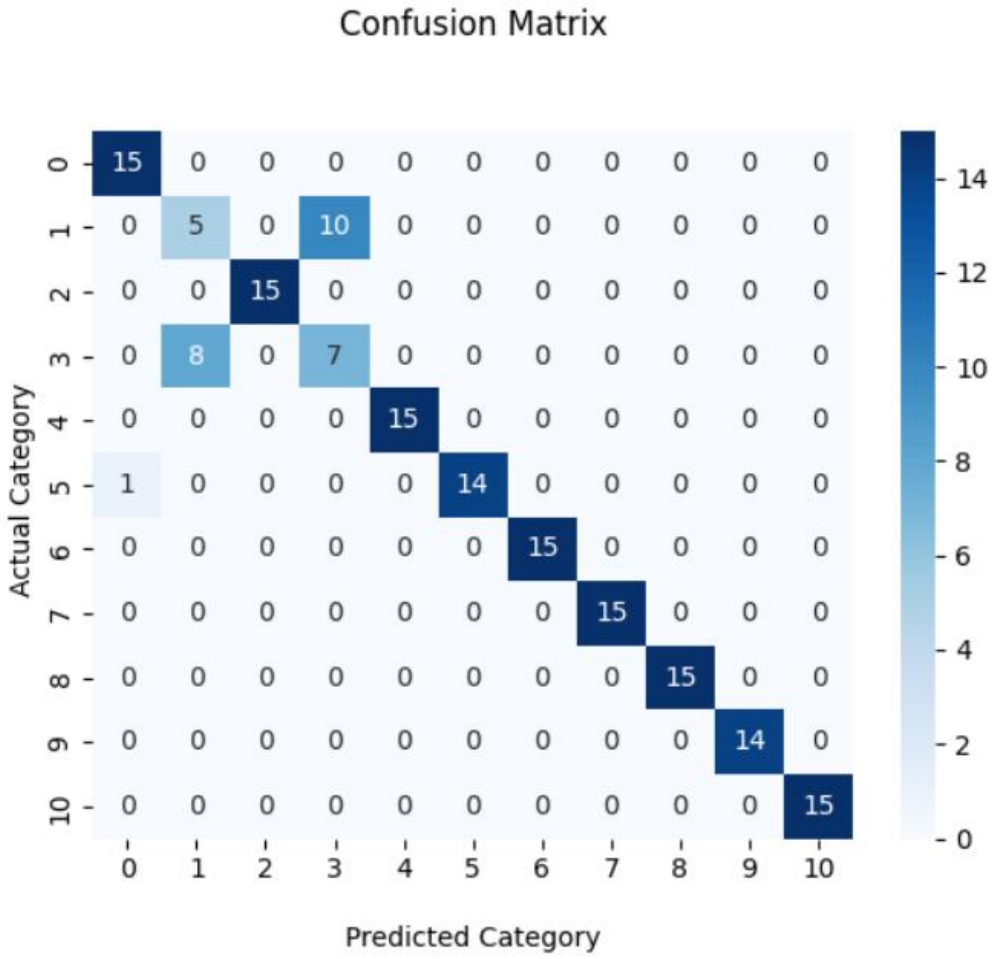
```
Accuracy : 0.8963414634146342
Precision: 0.8997213331861824
Recall   : 0.8963414634146342
F1 Score : 0.8963056235186178
```



RF:

In crop prediction, the RF model gives the accuracy of 87.8%. For precision it was right for 87.8% and the recall was 87.8%. It got an F1 score of 87%.

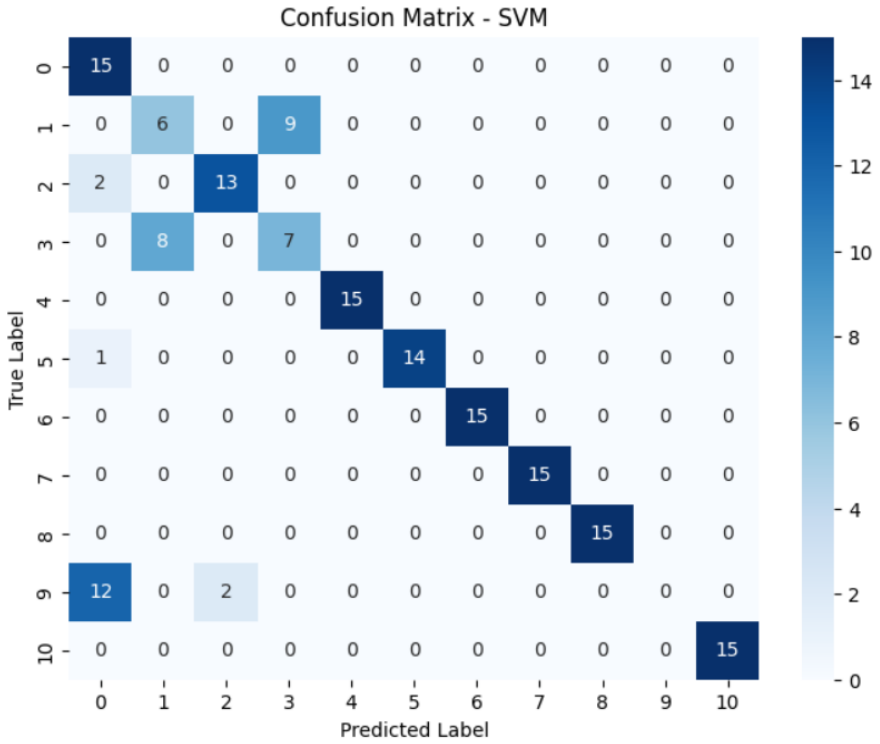
```
Accuracy : 0.8780487804878049
Precision: 0.8783209930313589
Recall   : 0.8780487804878049
F1 Score : 0.8779131284082585
```



SVM:

In crop prediction, the SVM model gives the accuracy of 89.02%. For precision it was right for 89.05% and the recall was 89.02%. It got an F1 score of 89%.

```
Accuracy : 0.8902439024390244
Precision: 0.8905705574912893
Recall   : 0.8902439024390244
F1 Score : 0.8901218155674326
```



5.3.6. Cross-Validation Results

Although K-fold cross-validation was planned, the current evaluation focused primarily on training and test splits. Cross-validation implementation remains a suggested future improvement.

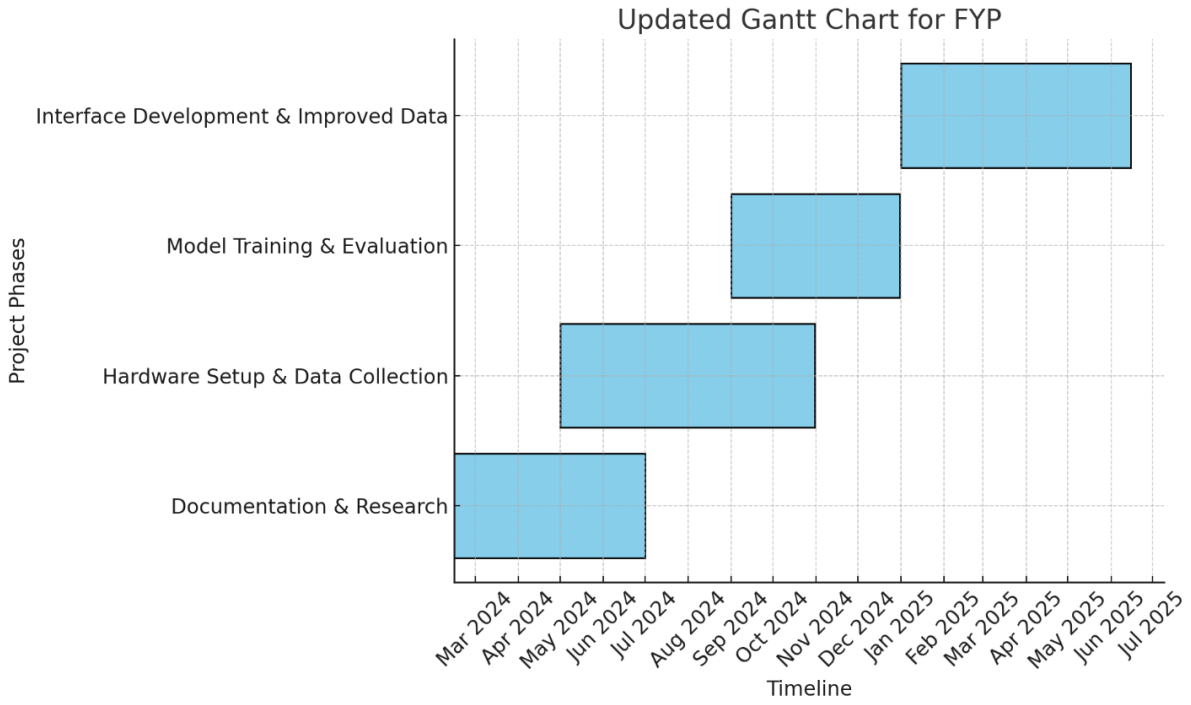
```
=== Naive Bayes ===
Accuracy: 0.9101
Precision: 0.9104
Recall: 0.9101
F1 Score: 0.9092

=== K-Nearest Neighbors ===
Accuracy: 0.8972
Precision: 0.8980
Recall: 0.8972
F1 Score: 0.8952

=== Random Forest ===
Accuracy: 0.9303
Precision: 0.9325
Recall: 0.9303
F1 Score: 0.9272

=== Support Vector Machine ===
Accuracy: 0.9009
Precision: 0.9014
Recall: 0.9009
F1 Score: 0.9007
```

5.4 Suggested Future Evaluations

To further enhance the pipeline's robustness and applicability, the following evaluations are recommended:

- **Adversarial Testing:** Evaluate the system's performance under adversarial conditions to ensure reliability in real-world scenarios.
- **Hyperparameter Optimization:** Run experiments with a grid and random approach to fine-tune the hyperparameters and achieve better results.
- Use benchmarking to compare the system with the latest models and frameworks and discover its weaknesses.
- Test whether the pipeline's deployment in different environments scales well, depending on the available resources.
- or edge environments with varying computational resources.

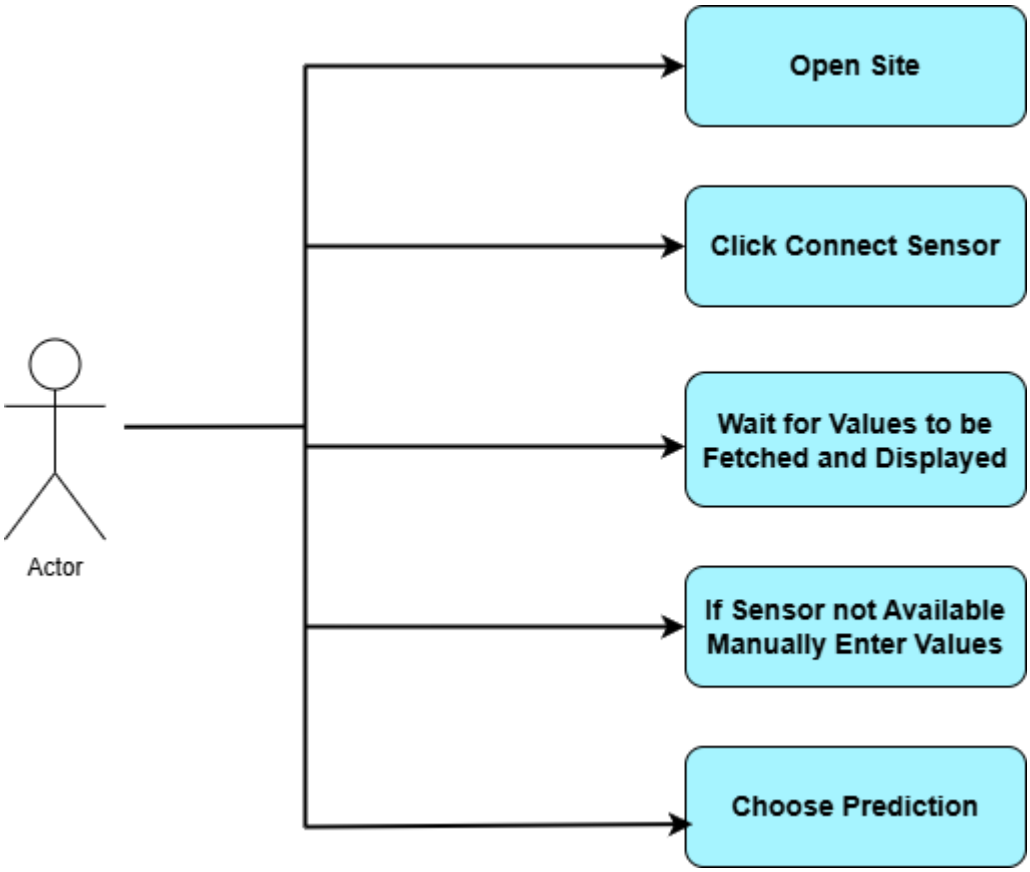
Gantt Chart for FYP:



Conclusion

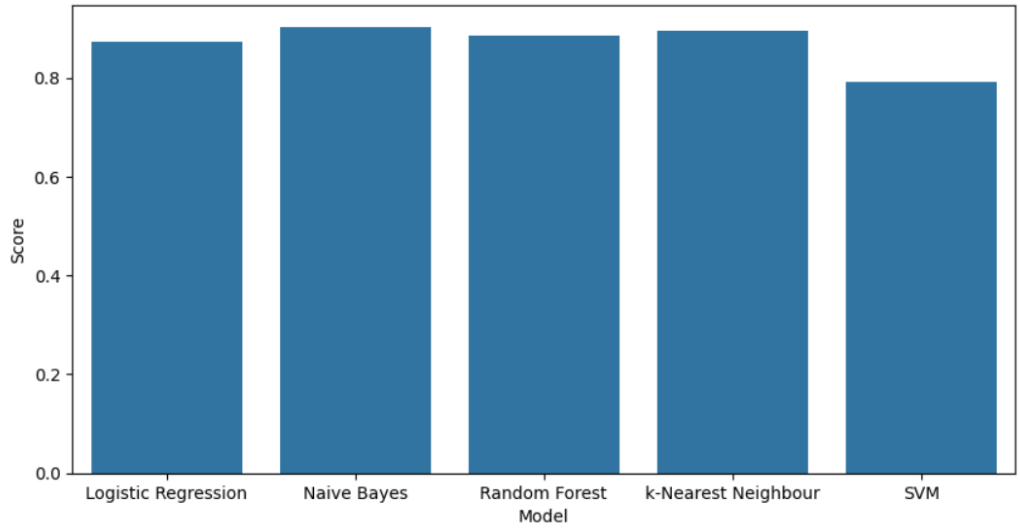
The analysis of the pipeline project indicated that it is not fully ready and could be improved in some respects. The findings prove that the used methodologies and models work correctly. Making more adjustments will lead to improvements in how the pipeline can be effectively applied.

5.5 User Interactions



5.6 Presentation of Results

	Model	Score
1	Naive Bayes	0.902439
3	k-Nearest Neighbour	0.896341
2	Random Forest	0.884146
0	Logistic Regression	0.871951
4	SVM	0.792683



5.7 Comparison with Previous Works

Most of the countries like US, China, and Australia are working with AI prediction on Crop Yields. They are using image datasets, CNN, LSTM, SVM, LR, and CV.

Whereas on the other hand India is working with Moisture, Humidity, Temperature, and Previous Crop Record to improve Irrigation and Controlled Farming.

Our work proposes benefits for individuals with very few knowledge of farming. Using features like NPK, PH, EC, and Temperature.

Chapter 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The project proved that it is possible to use machine learning to identify crops and predict soil fertility with an agricultural dataset. The system was designed to accurately identify what crop is planted using soil nutrient values and the environment, thanks to data preprocessing, working on expectations, model building and verification. Furthermore, the system was designed to reduce resource wastage by adding more water or fertilizers than requirements. So, it has also proven successful in this task.

Key outcomes of the project include:

- **Effective Preprocessing**: Data cleaning, outlier check, class imbalance check and encoding significantly improved the quality and usability of the dataset, ensuring reliable model input.
- **Feature Engineering**: We tried feature engineering by adding columns like average NPK and fertility label but they didn't help much instead they caused our accuracy to get low.
- The results demonstrate that using a Naïve Bayes model increased accuracy to nearly 90% among all the models used. It was able to show stability, decipherability and adapt to information it had not learned yet.
- The Colaboratory in Python uses modular code for different parts of a project which will help with future growth and adding more modules.
- Measurements: Model performance was assessed using accuracy, precision, recall, F1-score and with the help of confusion matrices and plots. These analyses proved that the model could be successful at classifying crops.

This study demonstrates the value of machine learning for agriculture and begins the process of creating smart solutions for farmers. When made more accurate, they may offer advice on crops to plant, how to use soil and sustainable ways to farm.

6.2 Project Limitation

Although the main objectives of crop identification and fertility analysis were reached through machine learning tools, the narrow scope, inaccuracies and relevant applications were affected by a few drawbacks. Below I describe the limitations of globalization.

1. Limited Dataset Scope

- The dataset focused on certain places and environmental conditions in Pakistan.
- Since the model only included a few sites with common soil, climate and crops, it may not be able to represent regions that are different in these factors.

2. Absence of Temporal Data

- The data set lacked information about seasonal changes, weather and past crop harvest yields.
- Because of this, the model was unable to consider time-dependent patterns valuable for effective decision-making in agriculture.

3. Class Imbalance

- The strategy of splitting classes pushed some crops into smaller proportions in the database.
- The skewed results might be due to the model being fine-tuned to understand the dominant crops better than the less common ones.

4. Feature Limitations

- The main information in the dataset was the NPK values and simple environmental aspects.
- It was designed to only show the output of sunlight, though soil pH, moisture, rain and temperature can play major roles in correct crop prediction.

5. Integration Constraints

- The project was implemented on a common laptop without any assistance from the cloud or graphic processing units (GPUs). But when we have to integrate it into soil we will require a special device which can roam into fields freely and apply the tool.
- To perform task like fertilizer spreading, pesticide sprinkling

6.3 Recommendations for Future Works

Although the project proved that machine learning can be used for farm plants and their fertility, there are many chances to explore this field even more. This can help point the way for upcoming developments and studies.

1. Use of Time-Series and Remote Sensing Data

- Include temporal information in your samples to handle variations caused by changes in seasons, plants and years.
- Introduce data from satellites for broad-reaching environmental monitoring and checking the wellbeing of crops.

2. Geo-Spatial Generalization

- Gather data from different provinces or countries to increase the usage of the model in various situations.
- Include GIS information in your recommendations for locations.

3. Advanced Modeling Techniques

- Explore the use of more **advanced algorithms** such as:

- Deep Neural Networks (DNN)
- Gradient Boosting (e.g., XGBoost, LightGBM)
- Convolutional Neural Networks (for image-based crop analysis)

- Apply **hyperparameter tuning** and **model ensembling** to further enhance performance.

4. Soil Fertility Index Refinement

- Collaborate with soil scientists to build models that align more closely with agronomic standards.

5. Real-World Testing and Feedback

- Carry out field trials together with agricultural institutions and local farmers to verify the results produced by the model.
- Improve the recommendation system by applying feedback given by users.

References:

- **Application of Machine Learning Techniques in Agricultural Crop Production: A Review**
Kamilaris, A., & Prenafeta-Boldú, F. X. (2018).
Biosystems Engineering, 164, 80–97.
https://doi.org/10.1016/j.biosystemseng.2017.09.014

- **Machine Learning Techniques for Crop Yield Prediction: A Survey**
Liakos, K. G., Busato, P., Moshou, D., Pearson, S., & Bochtis, D. (2018).
Computers and Electronics in Agriculture, 151, 99–109.
https://doi.org/10.1016/j.compag.2018.05.012

- **Soil Fertility Prediction Using Machine Learning Algorithms**
Shaikh, N., Pande, S., & Kadam, N. (2020).
International Journal of Computer Applications, 176(21), 1–5.
https://doi.org/10.5120/ijca2020919933

- **Crop Recommendation System Using Machine Learning Algorithms**
Patil, K., & Biradar, R. C. (2017).
International Journal of Advanced Research in Computer and Communication Engineering, 6(1), 209–212.
https://www.ijarcce.com/upload/2017/january-17/IJARCCE%2068.pdf

- **Smart Farming: Data Analytics in Agriculture**
Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M. J. (2017).
Computers and Electronics in Agriculture, 153, 69–80.
https://doi.org/10.1016/j.compag.2017.01.001

- **Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython**
McKinney, W. (2017).
O'Reilly Media, Inc.

- **Exploring Crop Prediction Based on Soil Fertility Using Machine Learning**
Sharma, R., & Sharma, A. (2021).
International Journal of Recent Technology and Engineering (IJRTE), 9(6), 71–76.
https://doi.org/10.35940/ijrte.F6142.039620

APPENDICES

Code Snippets:

```
# --- Imports ---
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
import joblib
```

requirements.txt

```
scikit-learn
pandas
numpy
matplotlib
seaborn
pymodbus
pydantic
fastapi
uvicorn
typing
gradio
requests
pyserial
```

```
# --- Data Loading ---
df = pd.read_csv(r'/content/Pakistan_Major_Crops_High_Favourability_Dataset_With_Moisture.csv')
print(df)
print(df.shape)
print(df.info())
print(df.isnull().sum())
print(df.columns)
```

```
#numerical features
numerical_df = df.select_dtypes(include=['number'])
print(numerical_df.columns)
```

```
Index(['Temperature', 'N', 'P', 'K', 'pH', 'EC', 'Moisture'], dtype='object')
```

```
#categorical features
categorical_df = df.select_dtypes(include=['object', 'category'])
print(categorical_df.columns)
```

```
# --- Feature Selection ---
numerical_features = ['Temperature', 'N', 'P', 'K', 'pH', 'EC', 'Moisture']
print(numerical_features)
print(df[df['Crop'] == 'Wheat'])
categorical_features = ['Crop']
print(categorical_features)
print(type(categorical_features))
print(type(numerical_features))
```

```
labels = list(df['Crop'].unique())
print(labels)
le_data = df.copy()
label_encoder = preprocessing.LabelEncoder()
le_data['Crop'] = label_encoder.fit_transform(le_data['Crop'])
print(le_data)
```

```
for col in numerical_features:
    le_data[col] = pd.to_numeric(le_data[col], errors='coerce')
    le_data = le_data.dropna(subset=[col])
    le_data.boxplot(column=col, by='Crop', figsize=(7, 4))
    plt.title("")
    plt.xlabel(col)
plt.show()
```

```
# --- Data Description ---
print(le_data.shape)
print(le_data.describe())
```

```
correlation = le_data.corr()
plt.figure(figsize=(12,5), dpi=600)
sns.heatmap(correlation, annot=True)
plt.show()
```

```
labels = list(df.Crop.unique())
sizes = []
for i in range(len(labels)):
    sizes.append((df.Crop.value_counts())[i])
print(df.Crop.value_counts())
plt.figure(figsize=(13,8))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.legend(labels)
plt.show()
```

```
# --- Histograms for Numerical Features ---
for column in numerical_features:
    plt.figure(figsize=(20,4))
    sns.histplot(data=le_data, x=column, bins=100, kde=True, hue='Crop', palette='husl')
    plt.title(column)
```

```
# --- Train-Test Split ---
from sklearn.model_selection import train_test_split
X = le_data[numerical_features]
y = le_data['Crop']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
```

```
# Print the shapes to confirm
print("X_train shape:", X_train.shape)
print("X_test  shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test  shape:", y_test.shape)
```

```
# --- Logistic Regression ---
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(solver="liblinear").fit(X_train, y_train)
print(lr_model)
from sklearn.metrics import classification_report, confusion_matrix
prediction_lr = lr_model.predict(X_test)
print('Predicted labels: ', np.round(prediction_lr)[:10])
print('Actual labels   : ', y_test[:10])
print(classification_report(y_test, prediction_lr))
cm = confusion_matrix(y_test, prediction_lr)
print(cm)
ax = sns.heatmap(cm, annot=True, cmap='Blues')
ax.set_title('Confusion Matrix\n\n')
ax.set_xlabel('\nPredicted Category')
ax.set_ylabel('Actual Category ')
plt.show()
# --- End Logistic Regression ---
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

# Train the Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Print metrics
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
print("F1 Score :", f1)

# Optional: full classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
# --- Model Comparison ---
models = pd.DataFrame({'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'k-Nearest Neighbour', 'SVM'],
                       'Score': [accuracy_score(y_test, prediction_lr), accuracy_score(y_test, prediction_nb), accuracy_score(y_test, prediction_rf), accuracy_score(y_test, predictic
print(models.sort_values(by='Score', ascending=False))
plt.figure(figsize=(10,5))
sns.barplot(x='Model', y='Score', data=models)
plt.show()
# --- End Model Comparison ---
```

```python
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# Define models
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Naive Bayes": GaussianNB(),
    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=5),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Support Vector Machine": SVC(kernel='linear', random_state=42)
}

# List of metrics
metrics = {
    "Accuracy": 'accuracy',
    "Precision": 'precision_weighted',
    "Recall": 'recall_weighted',
    "F1 Score": 'f1_weighted'
}

# Perform cross-validation for each model and metric
for name, model in models.items():
    print(f"\n=== {name} ===")
    for metric_name, metric in metrics.items():
        scores = cross_val_score(model, X, y, cv=5, scoring=metric)
        print(f"{metric_name}: {scores.mean():.4f}")
```

```python
# --- Data Inspection ---
print("Unique crops and counts:")
print(le_data['Crop'].value_counts())
print("Train target distribution:")
print(y_train.value_counts())
print("Training feature summary:")
print(X_train.describe())
# --- End Data Inspection ---
```

```
# --- User Input Prediction ---
le = preprocessing.LabelEncoder()
n = float(input("Enter nitrogen level (N): "))
p = float(input("Enter phosphorus level (P): "))
k = float(input("Enter potassium level (K): "))
temperature = float(input("Enter temperature (°C): "))
ph = float(input("Enter pH level: "))
ec = float(input("Enter EC level: "))
moisture = float(input("Enter moisture level: "))
user_input = {
    'Temperature': temperature,
    'N': n,
    'P': p,
    'K': k,
    'pH': ph,
    'EC': ec,
    'Moisture': moisture
}
print("\nCollected Sensor Data:")
print(user_input)
le.fit(le_data['Crop'])
user_df = pd.DataFrame([user_input])
user_df = user_df[numerical_features]
prediction = gnb_model.predict(user_df)[0]
predicted_crop = le.inverse_transform([prediction])[0]
print("🌾 Recommended Crop:", predicted_crop)
# --- End User Input Prediction ---
```

ui.py > predict_fertilizer

```
# Create the Gradio Interface
with gr.Blocks(title="Crop Recommendation System") as demo:
    gr.Markdown("# Crop Recommendation System")
    gr.Markdown("Connect to sensor, get readings, and predict suitable crops, irrigation, and fertilizer recommendations")

    with gr.Row():
        with gr.Column():
            connect_btn = gr.Button("Connect to Sensor")
            connection_status = gr.Textbox(label="Connection Status", interactive=False)
            connect_btn.click(connect_sensor, inputs=[], outputs=[connection_status])

        with gr.Column():
            get_values_btn = gr.Button("Get Sensor Values")
            sensor_status = gr.Textbox(label="Sensor Status", interactive=False)

    with gr.Row():
        with gr.Column():
            temperature = gr.Number(label="Temperature (°C)", value=25.0)
            n_value = gr.Number(label="Nitrogen (N)", value=50)
            p_value = gr.Number(label="Phosphorus (P)", value=30)
            moisture_value = gr.Number(label="Moisture", value=20.0)
        with gr.Column():
            k_value = gr.Number(label="Potassium (K)", value=40)
            ph_value = gr.Number(label="pH", value=6.5)
            ec_value = gr.Number(label="Electrical Conductivity (EC)", value=0.5)

    get_values_btn.click(
        get_sensor_values,
        inputs=[],
        outputs=[temperature, n_value, p_value, k_value, ph_value, ec_value, moisture_value, sensor_status]
    )
```

ui.py > predict_fertilizer

```

    with gr.Row():
        predict_btn = gr.Button("Predict Suitable Crops", variant="primary")
        irrigation_btn = gr.Button("Predict Irrigation", variant="primary")
        fertilizer_btn = gr.Button("Fertilizer Recommendation", variant="primary")

    with gr.Row():
        prediction_result = gr.Textbox(label="Prediction Result", interactive=False)
        irrigation_result = gr.Textbox(label="Irrigation Prediction", interactive=False)
        fertilizer_result = gr.Textbox(label="Fertilizer Recommendation", interactive=False)

    predict_btn.click(
        predict,
        inputs=[temperature, n_value, p_value, k_value, ph_value, ec_value, moisture_value],
        outputs=[prediction_result]
    )
    irrigation_btn.click(
        predict_irrigation,
        inputs=[temperature, n_value, p_value, k_value, ph_value, ec_value, moisture_value],
        outputs=[irrigation_result]
    )
    fertilizer_btn.click(
        predict_fertilizer,
        inputs=[temperature, n_value, p_value, k_value, ph_value, ec_value, moisture_value],
        outputs=[fertilizer_result]
    )

    gr.Markdown("### Instructions:")
    gr.Markdown("""
    1. Click 'Connect to Sensor' to establish connection
    2. Click 'Get Sensor Values' to retrieve current readings
    3. Adjust values manually if needed
    4. Click 'Predict Suitable Crops', 'Predict Irrigation', or 'Fertilizer Recommendation' to get results
    """)
```