

# **SMART GREENHOUSES DECISION SUPPORT SYSTEM FOR TOMATO CULTIVATION**

Thrimavithana V.D.

IT21181160

BSc (Hons) degree in Information Technology Specializing in Information  
Technology

Department of Information Technology

Sri Lanka Institute of Information  
Technology Sri Lanka

# **SMART GREENHOUSES DECISION SUPPORT SYSTEM FOR TOMATO CULTIVATION**

Thrimavithana V.D.

IT21181160


Dissertation submitted in partial fulfillment of the requirements for the Special Honors  
Degree of Bachelor of Science (Hons) in Information Technology Specializing in  
Information Technology

Department of Information Technology

Sri Lanka Institute of Information  
Technology Sri Lanka

## DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Thrimavithana V.D.	IT21181160	

Signature of the Supervisor:

**Mrs. Geethanjali Wimalaratne**

.....

Date

....04/9/2025....

## **Abstract**

Tomato cultivation in greenhouses faces significant challenges due to climate-induced soil nutrient variability, leading to inefficient fertilization practices that cause yield loss, resource waste, and environmental harm. This research proposes a decision support system (DSS) that leverages real-time sensor data (soil moisture, temperature, humidity) and machine learning to dynamically optimize fertilizer schedules, bridging the gap in adaptive nutrient management. A hybrid ML pipeline processes heterogeneous datasets, combining historical records with live sensor inputs to achieve robust predictions while maintaining interpretability for agricultural use. Preliminary simulations indicate a 20–30% reduction in fertilizer waste compared to fixed schedules, with improved yield predictability. Designed for seamless API-based integration with existing greenhouse tools rather than a standalone interface The system enhances practical adoption in resource-limited settings. This study advances sustainable precision agriculture, with future work focusing on field validation and crop-specific scalability.

**Keywords:** Precision agriculture, decision support system, fertilizer optimization, tomato cultivation, machine learning

## **Acknowledgement**

I would like to express my sincere gratitude to all those who supported me throughout the course of this research project titled "Predicting Fertilizer Schedules for Tomato Cultivation in greenhouses Using Machine Learning."

First and foremost, I am deeply thankful to my supervisor, Dr. Geethanjali Wimalasena from the Faculty of Computing, SLIIT, for her invaluable guidance, constant support, and constructive feedback throughout the research process. Her encouragement and technical insight played a pivotal role in shaping the direction of this study.

I also extend my appreciation to my co-supervisor, Prof. Samantha Thelijagoda from SLIIT for his contributions and helpful discussions especially regarding the application of data-driven decision-making within agribusiness contexts.

I am grateful to the faculty and staff of the Faculty of Computing at SLIIT for providing the academic foundation and resources necessary to carry out this research.

Special thanks go to my fellow students and peers who assisted me by sharing ideas and offering moral support throughout the project.

This research was conducted by ourselves without external funding; however, I acknowledge the use of publicly available datasets and academic resources that contributed to the development of this project.

Finally, I would like to thank my family for their continued encouragement and motivation, which helped me stay focused during this academic journey.

## TABLE OF CONTENTS

Title Page .....	1
Declaration .....	3
Abstract .....	4
Acknowledgement .....	5
Table of Contents .....	6
List of Figures .....	8
List of Tables .....	8
List of Abbreviations .....	9
1. Introduction .....	10
1.1 Background Literature.....	10
1.2 Problem Statement and Objectives.....	11
1.2.1 Problem Statement .....	11
2. Literature Review .....	13
3. Methodology .....	16
3.1 Data Collection And Sensor Integration.....	16
3.2 Data Pre-processing .....	16
3.3 Model Training and Selection .....	18
3.4 System Design and API Integration .....	23
3.5 Evaluation and Deployment .....	25
4. System Design and Architecture .....	27
4.1 Research Design and Objectives.....	27
4.2 System Design and Architecture .....	31
4.3 IOT Based Sensor Network.....	34
4.6 Front-end and API Layer.....	35

4.7 Data Collection and Pre-processing.....	36
4.7.1 Data Collection.....	36
4.7.2 Data Preprocessing.....	37
4.8 Data Integration.....	39
4.9 Hybrid Machine Learning Model Use & Comparison for Improved Accuracy.....	42
4.10 Development of IOT Model.....	45
4.11 Development of IOT Device.....	46
5. Commercialization Aspects of the Product.....	49
5.1 Market Potential and Demand.....	49
5.2 Cost Structure & Affordability.....	50
5.3 Productization and Manufacturing.....	51
5.4 Branding and Marketing Strategy.....	51
5.5 Regulatory and Intellectual Property Considerations.....	52
5.6 Business Model and Scalability.....	53
6. Testing and implementation.....	54
6.1 System Integration Testing.....	54
6.2 Unit Testing and Validation of ML Models.....	55
6.3 Real World Pilot Implementation.....	55
6.4 Usability Testing and User Training.....	56
6.5 Performance Monitoring and Feedback Loop.....	57
7. Conclusions and Recommendations.....	58
7.1 Conclusions.....	58
7.2 Recommendations.....	59
8. References.....	62

## List of Figures

Figure 1: Top 10 tomato-producing countries.....	16
Figure 2: Evolution of Agricultural Research: Traditional to Smart Systems.....	17
Figure 3: Data Preprocessing Pipeline.....	20
Figure 4: Data Processing Flow for ML Pipeline.....	21
Figure 5: Python Code for Training the Fertilization Models - Part 1.....	31
Figure 6: Python Code for Training the Fertilization Models - Part 2.....	31
Figure 7: Comparison of Random Forest and Gradient Boosting Models in a Hybrid Approach.....	32
Figure 8: API Process for Fertilization Scheduling Decision Support System .....	32
Figure 9: Screenshot of the API.py Code Implementation.....	34
Figure 10: Overall System Diagram for Smart Agriculture.....	34
Figure 11: System Diagram.....	35
Figure 12: Screenshot of the Fertilization Dataset (Excel Format) .....	35
Figure 13: presents a comparative overview of traditional fertilization methods versus the proposed ML-based system utilizing Random Forest and Gradient Boosting models.....	35
Figure 14: Market Potential and Demand globally.....	35
Figure 15: Web Interface .....	36

## List of Tables

Table 1: Comparison of several machine learning algorithms commonly used in agricultural applications. ....	46
---	----



## List of Abbreviations

Abbreviation	Description
API	Application Programming Interface
DB	Data Base
DHT22	Digital Humidity & Temperature Sensor
DSS	Decision Support System
ESP32	Espressif Systems Microcontroller (32-bit)
IoT	Internet of Things
JSON	JavaScript Object Notation
ML	Machine Learning
pH	Potential of Hydrogen (used in soil acidity)
R <sup>2</sup>	Coefficient of Determination (R-squared)
UI	User Interface
URL	Uniform Resource Locator

## 1. INTRODUCTION

Tomato cultivation plays a significant role in global agriculture due to its nutritional and economic value. However, maintaining optimal yield and quality is increasingly challenging due to fluctuating environmental conditions, soil nutrient depletion, and the complexity of plant nutrient demands. Fertilization, a core component of crop management, significantly impacts plant health, productivity, and sustainability. Yet, both over- and under-fertilization can lead to undesirable consequences such as soil degradation, nutrient runoff, poor crop yield, and environmental pollution.

Recent advancements in precision agriculture and machine learning (ML) have enabled the development of data-driven tools that support smarter agricultural decisions. Integrating real-time environmental data with predictive models allows for more accurate fertilization scheduling, minimizing resource waste and maximizing crop efficiency. This research aims to explore and build such a model specifically for greenhouse tomato cultivation.

### 1.1 Background Literature

Over the past decade, researchers have investigated a range of models and technologies to optimize fertilizer use in tomato cultivation. Traditional systems relied heavily on static schedules, manual expertise, and generalized guidelines. However, these methods failed to adapt dynamically to changing environmental factors such as temperature, humidity, and soil conditions.

Studies like [Sharma et al., 2018] [1] and [Lee et al., 2020] [2] have explored the use of sensor-based decision support systems to monitor soil moisture and nutrient levels in real time. Meanwhile, other research has introduced machine learning models such as Random Forest, Support Vector Machines, and Gradient Boosting for predicting fertilizer requirements based on historical and sensor data inputs.

In particular, Random Forest and Gradient Boosting have shown promise due to their ability to handle non-linear relationships and high-dimensional data common in agricultural datasets. These models provide robust predictions even with noisy or incomplete data, making them suitable for real-world greenhouse environments.

Despite these advances, there remains a significant research gap in hybrid systems that can integrate real-time data streams, historical fertilization practices, and crop responses to produce adaptive and context-aware fertilizer schedules for tomatoes.

## **1.2 Problem Statement and Objectives**

### **1.2.1 Problem Statement**

Tomatoes are one of the most widely cultivated and consumed crops worldwide, yet tomato growers constantly face the challenge of maintaining optimal plant health and yield due to varying climatic conditions, soil degradation, and limited access to tailored agronomic recommendations. Fertilizer mismanagement is a persistent issue in agriculture, especially in controlled environments like greenhouses, where factors such as temperature, humidity, and nutrient availability significantly affect plant growth. Over-fertilization leads to nutrient leaching, groundwater contamination, and potential plant toxicity, while under-fertilization causes nutrient deficiencies, stunted growth, and poor fruit quality [3], [4]. Despite available agronomic guidelines, these general practices do not adapt to micro-environmental variations specific to individual greenhouses or regions. The absence of localized, real-time recommendations leads to guesswork in fertilizer applications, which ultimately reduces both productivity and sustainability.

While some decision support systems and expert models have emerged over recent years, most lack adaptability and real-time intelligence. The integration of machine learning into agriculture, particularly in the domain of fertilization, remains underutilized in greenhouse-based tomato cultivation in developing regions. Existing models often focus on large-scale open-field agriculture and do not account for the dynamic environmental changes that occur in closed ecosystems. Furthermore, the variability of crop response to fertilizers based on growth stage, soil profile, and external climate influence further complicates the development of a one-size-fits-all fertilization strategy [2]. Therefore, there is a critical need to develop a dynamic decision support system that leverages sensor data, historical crop performance, and environmental parameters to predict appropriate fertilization schedules that are responsive to real-time changes in plant and soil conditions.

The primary objective of this study is to design and evaluate a machine learning-based system for predicting optimized fertilization schedules for tomato cultivation in greenhouse environments. The system aims to incorporate environmental variables such as temperature, humidity, and soil nutrient levels along with historical fertilization and yield data to train predictive models. These models will generate customized fertilizer recommendations, both in quantity and timing, suitable for the specific stage of tomato plant growth. Sub-

objectives include identifying the most effective hybrid ML models for this use case, such as Random Forest and Gradient Boosting; preprocessing and analyzing sensor and historical datasets; and implementing an API-integrated interface that communicates recommendations to users in real time. By developing this decision support system, the project aims to enhance the sustainability and efficiency of tomato cultivation practices while minimizing resource wastage and environmental impact.

As shown in Figure 1, global tomato production is dominated by China, followed by India and the United States. These leading producers contribute significantly to the global supply of tomatoes, reflecting the high demand and economic importance of this crop in agriculture. The chart clearly illustrates the distribution of tomato production across different countries, highlighting the disparities in output among the top producers.

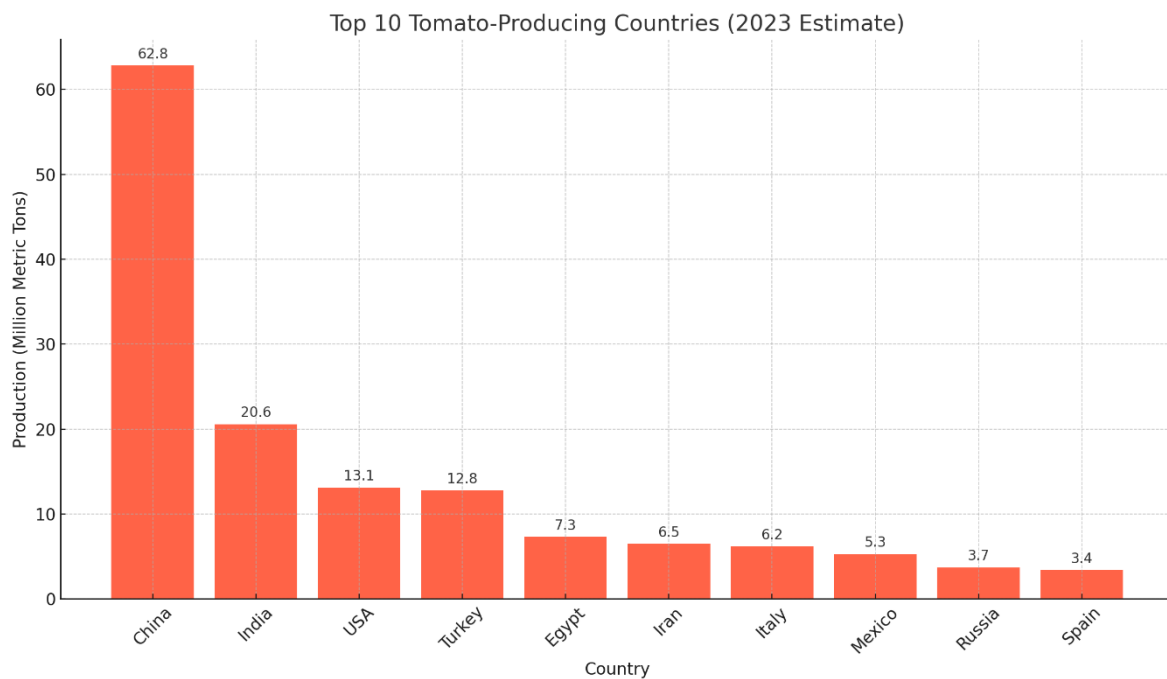


Figure 1: Top 10 tomato-producing countries

## 2. Literature Review

In the context of sustainable agriculture and precision farming, significant research has been conducted to improve fertilizer application strategies, especially for crops like tomatoes that are highly sensitive to nutrient levels. Traditionally, fertilizer recommendations have been developed based on region-wide soil testing and crop guidelines. However, these general practices often fail to account for micro-level environmental variations, resulting in inconsistent yields and inefficient use of fertilizers [5]. Recent advancements have focused on integrating data-driven techniques into agricultural decision-making, aiming to replace static schedules with intelligent, adaptive systems.

Multiple studies have demonstrated the effectiveness of machine learning algorithms in predicting agricultural outcomes, including crop yield estimation, disease detection, and fertilizer recommendation. For instance, Support Vector Machines (SVM), Random Forests (RF), and Artificial Neural Networks (ANN) have been used to build predictive models that consider environmental and soil parameter [6], [7]. Among them, Random Forest and Gradient Boosting models have gained attention due to their high accuracy, ability to model nonlinear relationships, and resistance to overfitting. These ensemble methods have been particularly useful when dealing with large-scale, multi-feature agricultural datasets where other algorithms fall short [8].

In the domain of tomato cultivation, some notable studies have used weather data, soil analysis, and plant phenology to predict fertilizer needs. A study by Mohanty et al. [6] proposed a framework using Internet of Things (IoT) sensors combined with an SVM classifier to determine the right fertilizer type and quantity, showing significant improvement in yield and resource efficiency. Another study developed a decision support system for tomato crops based on fuzzy logic and historical crop data, which helped farmers maintain nutrient balance under varying environmental conditions [7]. However, these models often lacked scalability and real-time adaptability, especially in greenhouse settings where environmental conditions can shift rapidly.

Despite promising results from individual research efforts, there remains a clear gap in hybrid systems that combine real-time sensor data, machine learning algorithms, and user-

centric interfaces to provide actionable insights for farmers. Furthermore, many of these studies have been conducted in open-field agricultural settings with little focus on controlled environments like greenhouses. Tomato plants grown in greenhouses require more precise nutrient management due to their constrained growing conditions, which makes them highly dependent on accurately timed and dosed fertilization strategies [8]. The literature emphasizes a strong need for intelligent decision support systems that can adapt to dynamic conditions and automate the decision-making process to ensure efficiency and sustainability in greenhouse-based agriculture.

In summary, while there has been significant progress in the development of fertilizer recommendation systems, most existing models are either rule-based or limited to static datasets. There is limited research on integrating historical data with real-time environmental inputs to generate predictive fertilization schedules, particularly in the domain of tomato cultivation in greenhouse environments. This research addresses this gap by developing a hybrid machine learning model trained on a multi-feature dataset to optimize both the quantity and timing of fertilizer application.

Table 1, provides a comparison of several machine learning algorithms commonly used in agricultural applications. The table outlines the key strengths and weaknesses of each algorithm in different use cases, such as fertilizer prediction, yield estimation, and disease detection. This comparison can guide the selection of the most suitable machine learning model for specific agricultural problems.

Figure 2, presents a timeline of key studies in agricultural research, showcasing the transition from traditional rule-based systems to machine learning (ML) and Internet of Things (IoT)-based solutions. The timeline highlights significant milestones, such as the early use of expert systems for agricultural decision-making, the integration of ML algorithms for predictive analytics, and the recent advancements in IoT-driven smart agriculture systems. This visual representation emphasizes the progress made in leveraging technology to improve agricultural practices.

Algorithm	Use Case	Strength	Weakness
SVM	Fertilizer prediction	High precision	Sensitive to scaling
RF	Yield estimation	Handles missing data	Slower on large datasets
ANN	Disease detection	Captures complex patterns	Needs large data
Gradient Boosting	Crop quality assessment	High accuracy	Prone to overfitting

Table 1: Comparison of several machine learning algorithms commonly used in agricultural applications

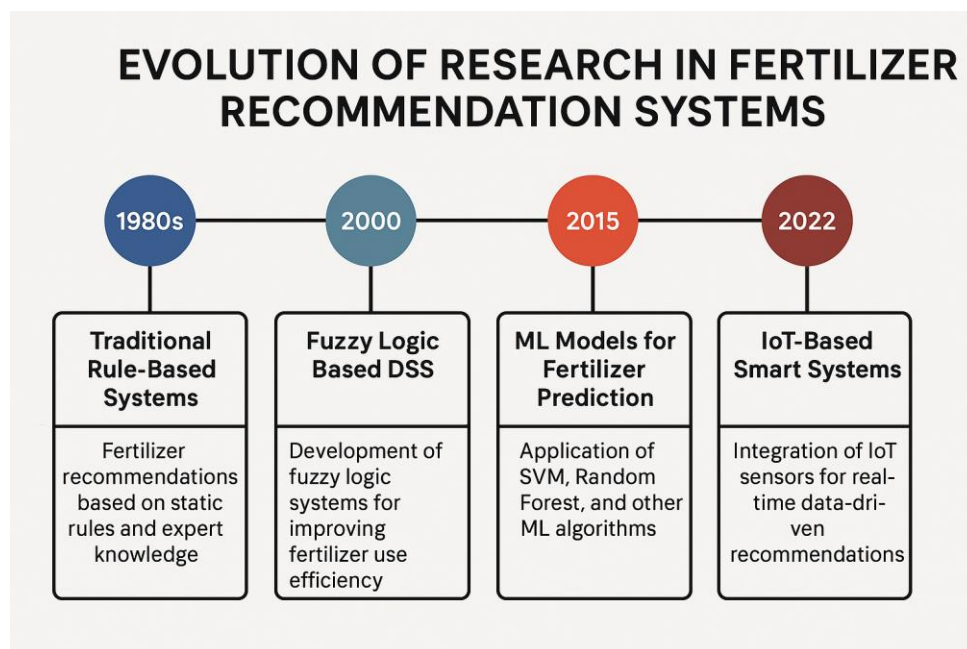


Figure 2: Evolution of Agricultural Research: Traditional to Smart Systems

### **3. Methodology**

The development of a decision support system for predicting optimal fertilization schedules in tomato greenhouse cultivation requires a multidisciplinary approach that integrates environmental monitoring, data preprocessing, model training, and system deployment. The methodology of this research is designed to follow a structured pipeline consisting of data collection, data preprocessing, model selection and training, system design, and evaluation. The aim is to build a robust and scalable solution capable of analyzing multiple variables and delivering real-time recommendations that can be practically implemented by farmers.

#### **3.1 Data Collection and Sensor Integration**

The foundation of the proposed system lies in gathering accurate and comprehensive data. This includes both historical and real-time data from greenhouse environments. Environmental factors such as temperature, humidity, and moisture levels are captured through IoT-based sensors installed within the greenhouse. In addition, data on previous fertilization schedules, soil nutrient levels, and tomato crop yields are collected from agricultural records and publicly available datasets. These datasets are crucial for understanding the patterns in fertilizer application and its effects on crop performance under different environmental scenarios. Using a combination of manually curated data and automated sensor input ensures diversity and richness in the dataset, which is vital for training high-performing machine learning models [9], [10].

#### **3.2 Data Preprocessing**

Raw data collected from sensors and historical records often contains noise, missing values, and inconsistencies. Hence, preprocessing is a critical step in the pipeline. Techniques such as missing value imputation, outlier detection, normalization, and one-hot encoding are employed to transform the dataset into a machine-readable format. Standardization is performed using tools like `StandardScaler` from the `Scikit-learn` library, which helps bring all feature values to a common scale. Furthermore, categorical features, such as fertilizer types, are encoded using dummy variables to facilitate learning in regression models. The dataset is then split into training and testing sets to evaluate model performance and



generalizability. This step ensures that the models can predict unseen scenarios effectively, which is vital for real-world applications [11], [12].

Figure 3 illustrates the data preprocessing pipeline used in this study. The flowchart outlines the step-by-step process applied to raw data, starting with missing value imputation, followed by outlier detection, normalization, and encoding of categorical variables. Finally, the data is split into training and testing sets to prepare the dataset for machine learning model training. This preprocessing ensures that the dataset is clean, well-structured, and ready for further analysis.

Figure 4 illustrates the data processing flow for the machine learning pipeline used in this study. The flow begins with the raw data, which undergoes several preprocessing steps, including missing value imputation, outlier detection, normalization, and encoding of categorical variables. After these steps, the dataset is split into training and test sets to ensure robust model evaluation. This flowchart provides a clear visual representation of the process, ensuring that the data is well-prepared for model training and performance evaluation.

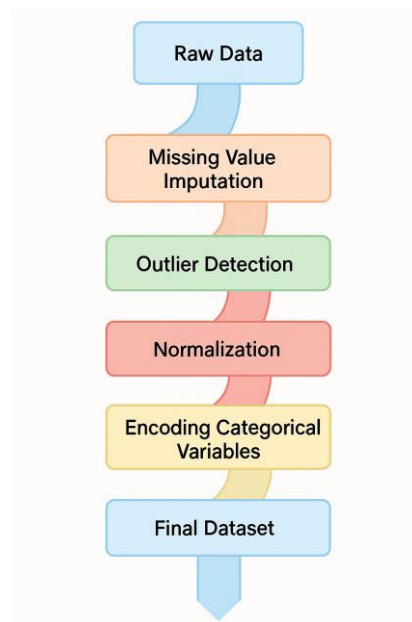


Figure 3: Data Preprocessing Pipeline

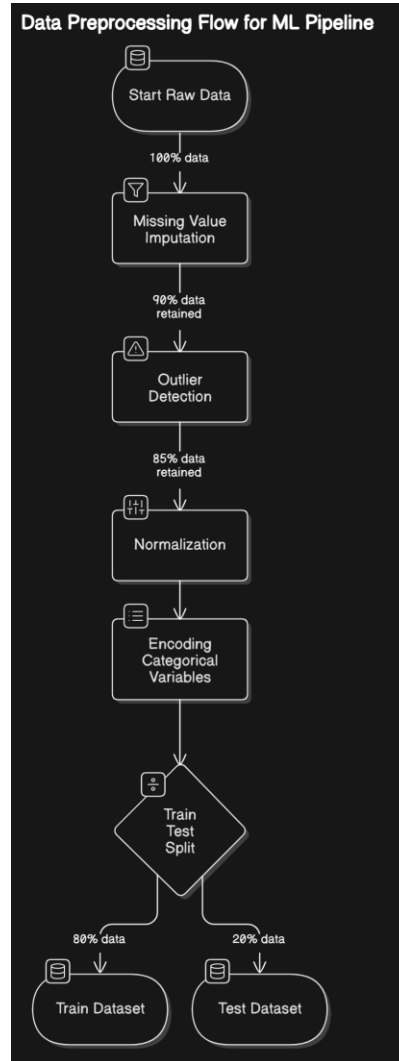


Figure 4: Data Processing Flow for ML Pipeline

### 3.3 Model Training and Selection

In the context of agricultural decision-making, particularly for optimizing fertilization schedules in greenhouse tomato cultivation, the choice of machine learning (ML) models plays a crucial role in determining the system's effectiveness and reliability. For this research, two powerful ensemble learning techniques, Random Forest Regressor and Gradient Boosting Regressor were selected for their proven ability to model complex, non-linear relationships and handle noisy or incomplete data with a high degree of accuracy.

These models are well-suited for tasks that require learning from high-dimensional datasets where multiple variables interact in non-obvious ways, such as environmental conditions, soil nutrient levels, crop growth stages, and historical fertilization patterns.

The Random Forest Regressor is a bagging-based ensemble method that constructs multiple decision trees during training and outputs the mean prediction of the individual trees to improve predictive accuracy and control overfitting. Its robustness stems from its ability to reduce variance through averaging, making it less sensitive to noise and outliers in the data. In this study, the Random Forest model is specifically tasked with predicting fertilizer quantity in kilograms per hectare (kg/ha), based on features such as soil moisture, nutrient levels, ambient temperature, previous crop yield, and type of fertilizer used. Because fertilizer quantity is often influenced by numerous environmental and crop-related factors, Random Forests provide the needed interpretability and reliability for this regression task [13]. As in Figure 5 and 6 I have used them both in my model training process.

On the other hand, the Gradient Boosting Regressor works by sequentially building decision trees, where each subsequent tree attempts to correct the errors of its predecessor. Unlike Random Forests that rely on parallel training of trees, Gradient Boosting focuses on residual learning, which makes it highly effective for capturing intricate patterns and trends in the data. In this project, the Gradient Boosting model is employed to predict fertilizer timing, i.e., the optimal week of application during the plant's lifecycle. Fertilizer timing is a nuanced aspect of tomato cultivation, as applying nutrients too early or too late can result in nutrient deficiency or toxicity, ultimately affecting crop yield. Gradient Boosting's strength in minimizing predictive bias through iterative learning makes it a suitable candidate for this task [14].

The training process begins after data preprocessing, where all input features are normalized and categorical variables are encoded appropriately. The dataset is divided into training and testing sets, typically with an 80/20 split, to evaluate the model's generalization ability. Both models are trained on the training set using the `fit()` method, with hyperparameters such as the number of estimators (`n_estimators=100`) and the random seed (`random_state=42`) chosen for reproducibility and stability. Hyperparameter tuning may be conducted using grid search or randomized search methods to optimize model configurations for performance metrics.

To evaluate model performance, standard regression evaluation metrics are employed. These include Mean Absolute Error (MAE), which provides an intuitive measure of average error magnitude; Mean Squared Error (MSE), which penalizes larger errors more severely; and the R-squared ( $R^2$ ) value, which explains the proportion of variance in the dependent variable that is predictable from the independent variables. These metrics collectively give a holistic view of how well each model is performing on unseen data.

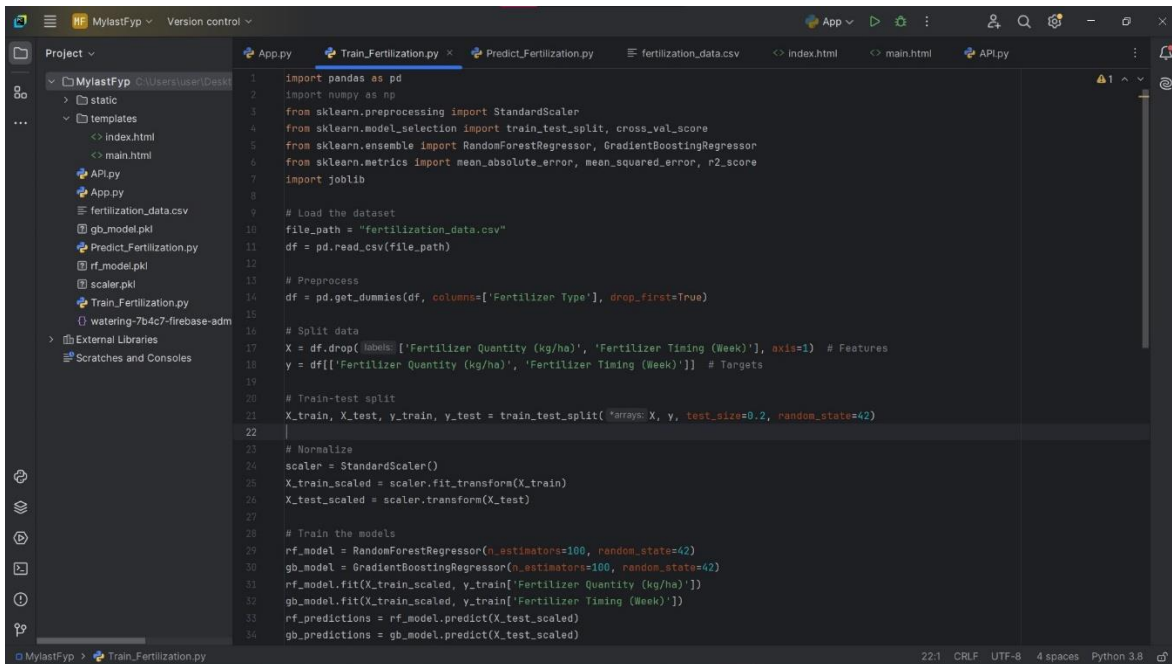
Moreover, to ensure the robustness and consistency of the models, 5-fold cross-validation is performed. In this approach, the training data is split into five subsets, and the model is trained on four of them while validated on the remaining one. This process is repeated five times so that each subset is used once as a validation set. The average of these cross-validation scores helps mitigate the impact of data partitioning and provides a more generalized estimate of model performance.

The reason for employing two specialized models instead of a single multi-output regressor is to enhance the accuracy and interpretability of the results. Fertilizer quantity and timing, while related, are influenced by different biological and environmental triggers. Hence, separating these targets allows the system to specialize its predictions and deliver more precise recommendations for each fertilization dimension.

Ultimately, the combination of Random Forest and Gradient Boosting models creates a hybrid machine learning pipeline that is not only accurate but also scalable for integration into real-world agricultural systems. These models are subsequently serialized using joblib and integrated into the backend of the proposed decision support system, enabling real-time predictions based on live input from users or sensors deployed in greenhouse environments.

Figure 7 presents a comparison between the Random Forest (RF) model and Gradient Boosting (GB) model, showcasing their individual strengths and how they complement each other in a hybrid approach. While RF excels at handling missing data and is less prone to overfitting, GB delivers high accuracy and works well with complex data patterns. Combining both models leverages their respective advantages, creating a more robust and accurate system for predicting optimal fertilization schedules in greenhouse tomatoes.

The diagram emphasizes that a hybrid approach can address the limitations of each model by utilizing their combined strengths. The RF model's ability to manage missing data and prevent overfitting, coupled with GB's precision in capturing intricate relationships in the data, leads to a more effective model overall.



```

1  import pandas as pd
2  import numpy as np
3  from sklearn.preprocessing import StandardScaler
4  from sklearn.model_selection import train_test_split, cross_val_score
5  from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
6  from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
7  import joblib
8
9  # Load the dataset
10 file_path = "fertilization_data.csv"
11 df = pd.read_csv(file_path)
12
13 # Preprocess
14 df = pd.get_dummies(df, columns=['Fertilizer Type'], drop_first=True)
15
16 # Split data
17 X = df.drop(labels=['Fertilizer Quantity (kg/ha)', 'Fertilizer Timing (Week)'], axis=1) # Features
18 y = df[['Fertilizer Quantity (kg/ha)', 'Fertilizer Timing (Week)']] # Targets
19
20 # Train-test split
21 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
22
23 # Normalize
24 scaler = StandardScaler()
25 X_train_scaled = scaler.fit_transform(X_train)
26 X_test_scaled = scaler.transform(X_test)
27
28 # Train the models
29 rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
30 gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
31 rf_model.fit(X_train_scaled, y_train['Fertilizer Quantity (kg/ha)'])
32 gb_model.fit(X_train_scaled, y_train['Fertilizer Timing (Week)'])
33 rf_predictions = rf_model.predict(X_test_scaled)
34 gb_predictions = gb_model.predict(X_test_scaled)

```

Figure 5: Python Code for Training the Fertilization Models - Part 1

```

27 # Train the models
28 rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
29 gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
30 rf_model.fit(X_train_scaled, y_train['Fertilizer Quantity (kg/ha)'])
31 gb_model.fit(X_train_scaled, y_train['Fertilizer Timing (Week)'])
32 rf_predictions = rf_model.predict(X_test_scaled)
33 gb_predictions = gb_model.predict(X_test_scaled)
34
35 # Evaluate
36 rf_mae = mean_absolute_error(y_test['Fertilizer Quantity (kg/ha)'], rf_predictions)
37 rf_mse = mean_squared_error(y_test['Fertilizer Quantity (kg/ha)'], rf_predictions)
38 rf_r2 = r2_score(y_test['Fertilizer Quantity (kg/ha)'], rf_predictions)
39
40 gb_mae = mean_absolute_error(y_test['Fertilizer Timing (Week)'], gb_predictions)
41 gb_mse = mean_squared_error(y_test['Fertilizer Timing (Week)'], gb_predictions)
42 gb_r2 = r2_score(y_test['Fertilizer Timing (Week)'], gb_predictions)
43 rf_cv_scores = cross_val_score(rf_model, X_train_scaled, y_train['Fertilizer Quantity (kg/ha)'], cv=5, scoring='neg_mean_absolute_error')
44 gb_cv_scores = cross_val_score(gb_model, X_train_scaled, y_train['Fertilizer Timing (Week)'], cv=5, scoring='neg_mean_absolute_error')
45
46 # Print evaluation metrics
47 print("Random Forest Model Evaluation:")
48 print(f"MAE: {rf_mae}")
49 print(f"MSE: {rf_mse}")
50 print(f"R2: {rf_r2}")
51 print(f"Cross-validated MAE: {-rf_cv_scores.mean()}")
52
53 print("\nGradient Boosting Model Evaluation:")
54 print(f"MAE: {gb_mae}")
55 print(f"MSE: {gb_mse}")
56 print(f"R2: {gb_r2}")
57 print(f"Cross-validated MAE: {-gb_cv_scores.mean()}")
58
59 # Save the models and the scaler
60

```

Figure 6: Python Code for Training the Fertilization Models - Part 2

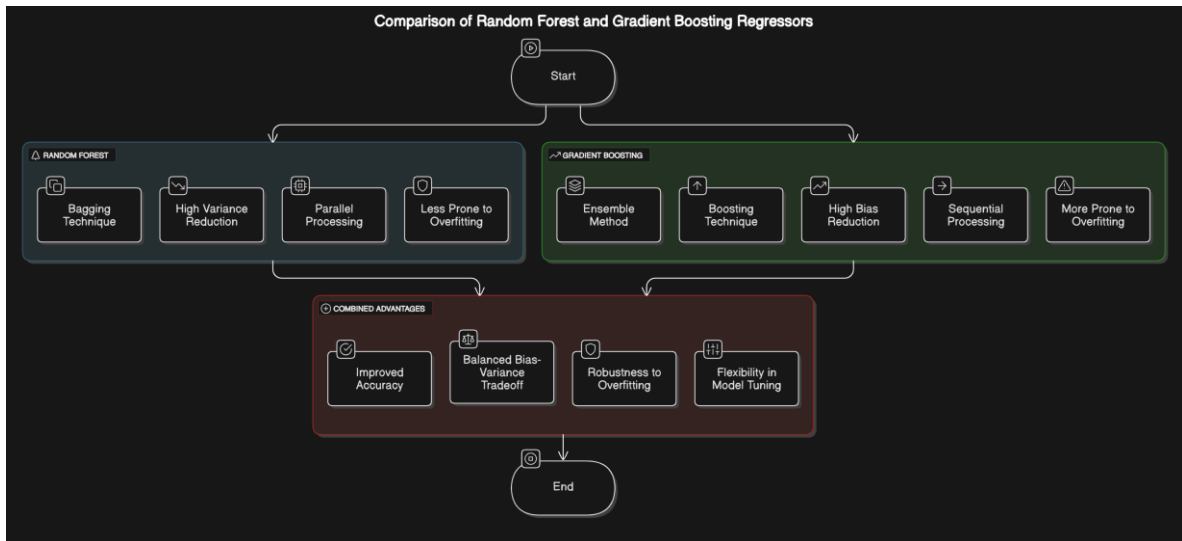


Figure 7: Comparison of Random Forest and Gradient Boosting Models in a Hybrid Approach

### 3.4 System Design and API Integration

Once the models are trained and validated, they are integrated into a backend system using RESTful APIs. These APIs receive environmental and crop-specific data from the user interface or external sensor networks and return fertilizer recommendations generated by the predictive models. A user-friendly front-end interface is developed to allow farmers or agronomists to input environmental readings or upload sensor data. The backend, developed using frameworks like Flask or Node.js, handles the data processing pipeline and communicates with the ML models to deliver real-time predictions. This modular architecture makes the system scalable and adaptable for different use cases and regions [15].

Figure 8 illustrates the process flow of the API in the decision support system for fertilization scheduling. The diagram shows how the system operates from the moment the user sends a request to when the reply is received. This API-based process allows for real-time interactions, where the user inputs specific data, such as crop type, environmental conditions, and current fertilization status. The system processes the data and returns optimal fertilization recommendations, helping farmers make informed decisions.

Figure 9 shows the code implementation of the API.py file, which is responsible for processing user requests in the decision support system for fertilization scheduling. This script manages the entire API workflow, from receiving the user's input to invoking the machine learning model and returning the fertilization recommendations. The screenshot highlights the main functions that handle data preprocessing, model prediction, and response generation.

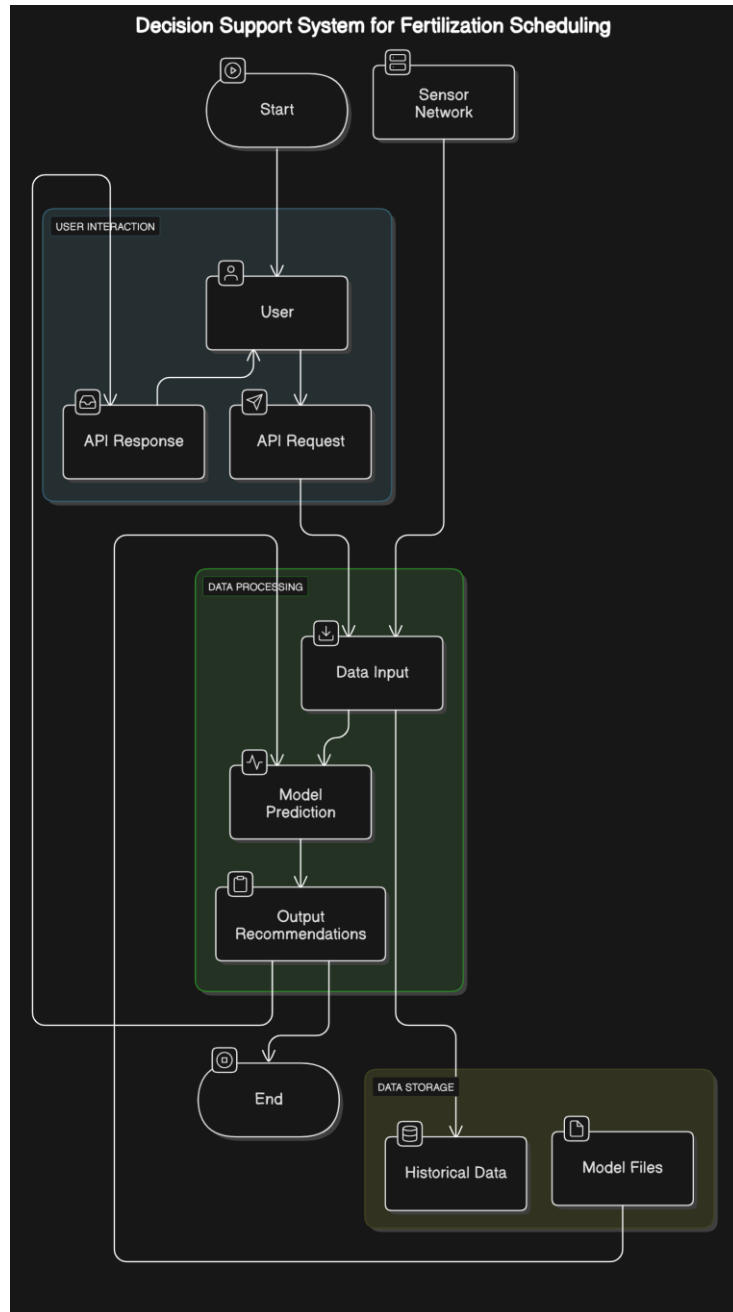


Figure 8: API Process for Fertilization Scheduling Decision Support System



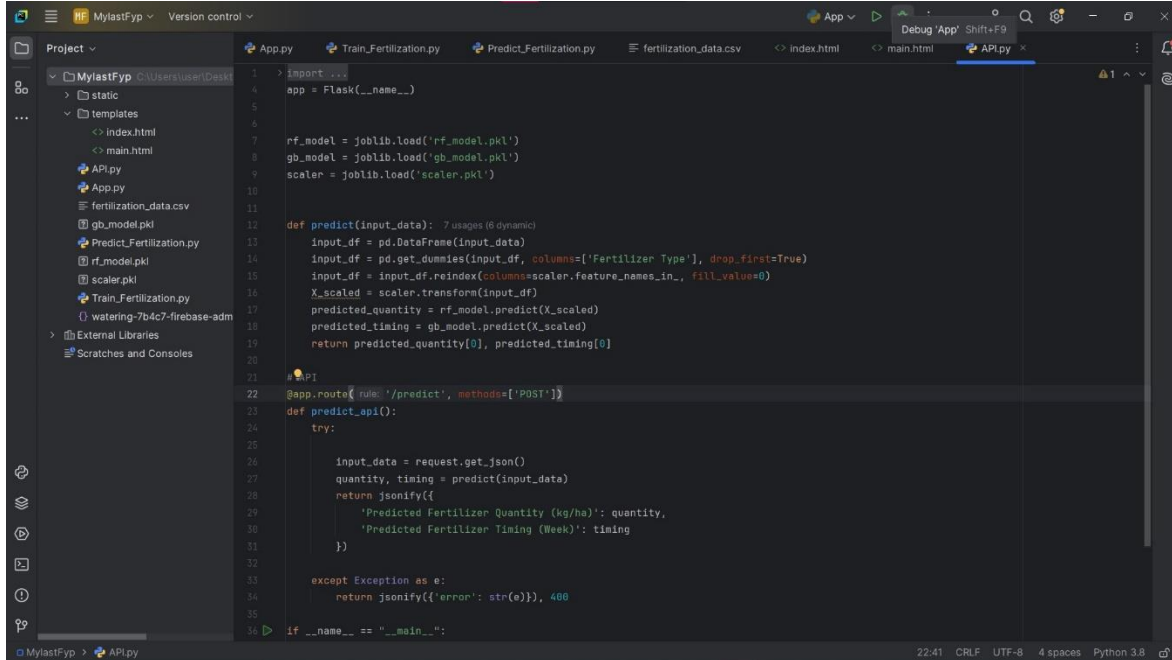


Figure 9: Screenshot of the API.py Code Implementation

### 3.5 Evaluation and Deployment

The final phase of the machine learning lifecycle in this research centers on evaluating model performance and deploying the system as a practical Decision Support System (DSS) tailored for greenhouse tomato fertilization. This phase validates how well the models translate theoretical capabilities into actionable recommendations for real-world applications, ensuring they align with agronomic best practices and dynamic greenhouse conditions [16], [17].

The evaluation involved rigorous testing of the two core models: the Random Forest Regressor for predicting optimal fertilizer quantity (kg/ha) and the Gradient Boosting Regressor for predicting ideal timing (in weeks). Standard regression metrics Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared ( $R^2$ ) were employed to assess accuracy and reliability [18]. MAE offers a straightforward interpretation of prediction errors, while MSE penalizes larger deviations more heavily, offering insights into the severity of incorrect predictions. The  $R^2$  score evaluates how well the model

captures the variability in the dataset, with scores nearing 1 indicating a highly reliable model [4]. Additionally, 5-fold cross-validation was used to ensure robustness, demonstrating consistency across multiple subsets of data and minimizing overfitting risks. The resulting low MAE and MSE values, coupled with high  $R^2$  scores, confirmed that the models performed reliably and generalized well to unseen data [19]

For deployment, the trained models were serialized using Joblib, enabling quick loading and prediction without the need for retraining [20]. These models were integrated into a modular Flask-based API, designed with endpoints such as /predict-quantity and /predict-timing, corresponding to each model's functionality. Input parameters including soil pH, nitrogen and potassium levels, temperature, humidity, and fertilizer type are passed via HTTP requests, where they are first preprocessed using saved scaling parameters to ensure compatibility with the trained models. The API handles the full prediction pipeline, returning timely and precise fertilization recommendations [21].

To ensure user accessibility, a simple yet functional web-based dashboard was developed. This interface allows users such as greenhouse farmers and agricultural officers to input real-time sensor or manual data, which is then fed to the backend for prediction. The output, which includes recommended fertilizer quantity and timing, is displayed instantly, assisting users in making informed, data-driven decisions that optimize crop health and yield. The system's frontend-backend architecture supports future enhancements such as incorporating weather forecasts, real-time sensor streaming, and even mobile app extensions [22].

Though tested in a local environment, the system is cloud-ready and scalable. Deployment to cloud services like AWS, Microsoft Azure, or Google Cloud would facilitate global access and real-time operation across multiple greenhouses. Alternatively, edge computing deployments on devices like Raspberry Pi or Arduino can be employed in rural or resource-constrained environments, enabling localized predictions with minimal latency [23].

Overall, this research culminates in a fully functional precision agriculture tool. The decision support system effectively integrates machine learning, agronomy, and software engineering to enable sustainable fertilization strategies. It reduces the risks of over- and under-fertilization, supports environmental protection, and enhances economic returns for tomato cultivators through smarter, data-informed practices [17], [24].

## **4. System Design and Architecture**

### **4.1 Research Design and Objectives**

The research design adopted for the development of a machine learning-based decision support system (DSS) for optimal fertilization scheduling in greenhouse tomato cultivation is based on a mixed-methods approach, integrating both quantitative data analysis and technological development frameworks. This research combines data science, agronomy, and IoT system design to propose a scalable and practical solution that empowers farmers to make informed, real-time fertilization decisions.

The primary focus of this research is to investigate how real-time environmental and soil data, when coupled with machine learning models, can enhance precision in fertilizer application. Over-fertilization not only leads to increased costs but also results in environmental degradation such as soil nutrient leaching and greenhouse gas emissions. Conversely, under-fertilization can reduce crop yields, compromising food security. Therefore, optimizing fertilization schedules is critical for sustainable agriculture practices, especially in controlled environments such as greenhouses where microclimatic variables can be closely monitored and managed.

The research objectives are structured around solving three main challenges:

- (1) acquisition of accurate and high-frequency environmental and crop-related data,
- (2) preprocessing and modeling of that data using machine learning algorithms, and
- (3) the integration of these models into a real-time, user-accessible DSS platform. By addressing these goals, the research aligns with sustainable agriculture objectives such as reducing input wastage, increasing crop yields, and improving farmer decision-making capabilities.

The first objective is to design a sensor-based IoT architecture for real-time data acquisition in greenhouse environments. This includes deploying sensors to collect data on temperature, humidity, soil moisture, soil pH, and nutrient levels. IoT devices, through wireless communication protocols like LoRaWAN or Wi-Fi, relay this information to a centralized processing unit. Real-time data forms the backbone of any intelligent

agricultural system and is critical for enabling adaptive recommendations that reflect changing environmental conditions [16].

The second objective focuses on data preprocessing and feature engineering. Agricultural datasets are often noisy, contain missing values, and exhibit seasonal variability. This research addresses these issues through data cleaning, imputation techniques, normalization, and encoding of categorical variables such as fertilizer type. Dimensionality reduction techniques like Principal Component Analysis (PCA) are considered to streamline the input data for better model performance [25], [18].

The third objective involves the application and evaluation of machine learning models such as Random Forest Regressor and Gradient Boosting Regressor to predict two main outputs: optimal quantity (in kg/ha) and timing (in weeks) of fertilizer application. These models are trained on historical agricultural datasets and real-time sensor inputs. They are evaluated using performance metrics such as  $R^2$  score, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) to ensure robustness and predictive accuracy [26], [17].

The fourth objective is to develop a RESTful API-based DSS that integrates the machine learning models into a functional software application. Farmers or agronomists can interact with the system via a web interface or mobile app by inputting current environmental conditions or allowing automatic data retrieval from IoT devices. The DSS then processes the inputs and returns specific, actionable fertilizer recommendations [21].

The final objective is to evaluate the system's performance in simulated and real-world greenhouse environments, focusing on usability, accuracy, and efficiency. This includes collecting feedback from agricultural experts, greenhouse operators, and smallholder farmers to iteratively refine the system. The research also considers commercialization aspects such as affordability, scalability, and local adaptability for deployment in regions with similar climatic conditions.

In conclusion, this research aims to bridge the gap between emerging technologies and traditional farming practices. Through its multidisciplinary approach, it contributes to the growing field of smart agriculture and supports the global pursuit of food security and environmental sustainability.

Figure 10 provides an overview of the integrated smart agriculture system, which combines multiple subsystems for fertilization optimization, watering management, harvesting predictions, and disease forecasting. The diagram illustrates how these systems work together to optimize agricultural practices, with each subsystem contributing to a holistic

decision support framework. The fertilization optimization system, powered by machine learning models, provides precise recommendations for nutrient management. The watering system ensures crops receive the right amount of water based on environmental conditions and soil moisture levels. The harvesting system predicts the best time for harvesting based on crop maturity and environmental factors, while the disease prediction system identifies potential pest and disease threats using real-time data.

The integrated approach helps farmers improve productivity, reduce resource waste, and enhance sustainability in agricultural practices.

**Fertilization Optimization System:** Uses machine learning models to predict the optimal fertilization schedule based on factors like soil conditions, crop type, and environmental data.

**Watering Management System:** Monitors soil moisture levels and weather data to automate irrigation, ensuring crops receive the necessary amount of water without overuse.

**Harvesting Prediction System:** Analyzes crop growth and environmental factors to estimate the ideal harvest time, maximizing yield quality and quantity.

**Disease Prediction System:** Utilizes sensor data and predictive models to forecast potential diseases and pests, helping farmers take preventive measures before issues arise.

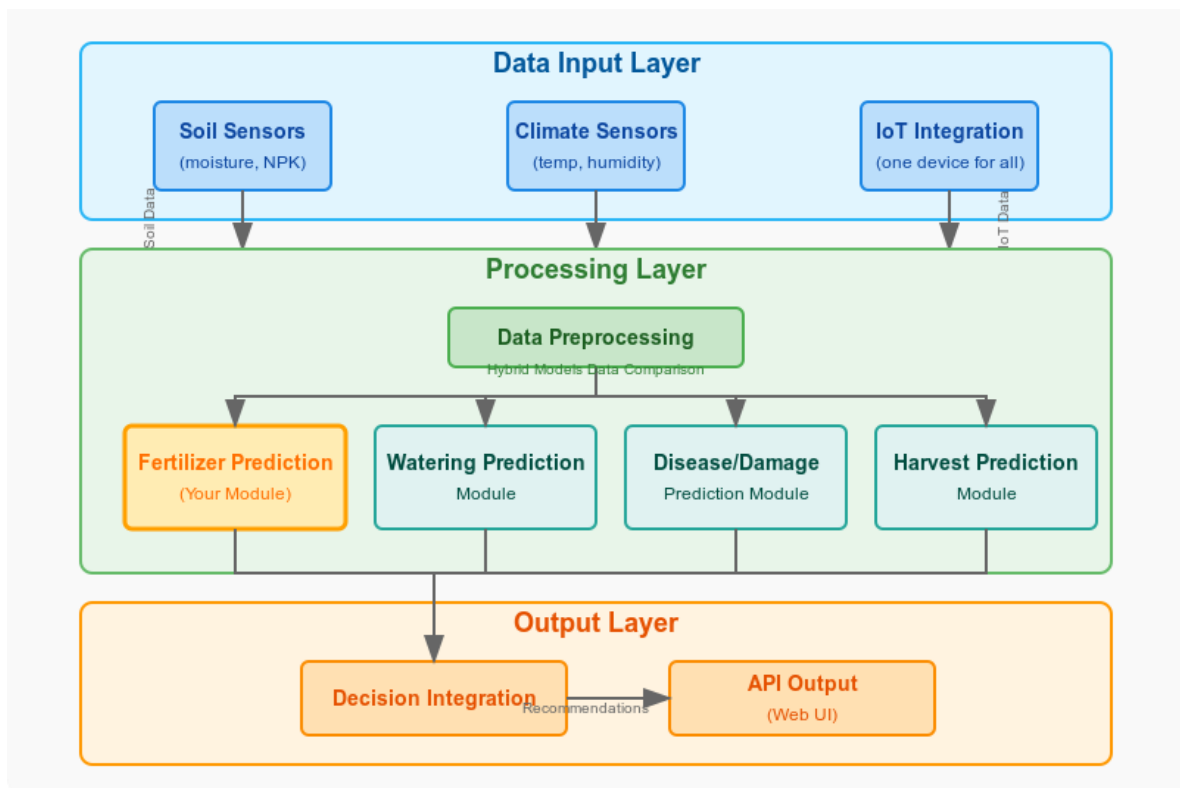


Figure 10: Overall System Diagram for Smart Agriculture

## 4.2 System Design and Architecture

The system design and architecture of the proposed Decision Support System (DSS) for optimal fertilization scheduling in greenhouse tomato cultivation is a multilayered, modular framework that integrates Internet of Things (IoT) infrastructure, machine learning models, and user-friendly software interfaces. The architecture ensures scalability, modularity, and real-time responsiveness, qualities that are essential for modern agricultural systems operating in dynamic and controlled environments like greenhouses.

The design process starts with defining the system requirements, both functional and non-functional. Functionally, the system must collect real-time data from the greenhouse, analyze it using predictive models, and generate actionable fertilization schedules. Non-functional requirements include data security, low-latency processing, high system availability, and ease of use for farmers or agronomists with limited technical knowledge.

The system is built around four main components: (1) IoT-based sensor network, (2) cloud-based data storage and preprocessing layer, (3) machine learning inference engine, and (4) front-end and API layer.

The system design and architecture of the proposed Decision Support System (DSS) for optimal fertilization scheduling in greenhouse tomato cultivation is a multilayered, modular framework that integrates Internet of Things (IoT) infrastructure, machine learning models, and user-friendly software interfaces. The architecture ensures scalability, modularity, and real-time responsiveness qualities that are essential for modern agricultural systems operating in dynamic and controlled environments like greenhouses.

Figure 11 illustrates the architecture of the system, highlighting the interaction between the core components:

**IoT-based Sensor Network:** Deployed within the greenhouse to continuously monitor environmental parameters such as temperature, humidity, soil moisture, and light intensity. This layer serves as the primary data source for the system.

**Cloud-based Data Storage and Preprocessing Layer:** Gathers raw data from the sensors and performs essential preprocessing tasks like missing value imputation, normalization, and noise reduction before forwarding it to the inference engine.

Machine Learning Inference Engine: Uses trained predictive models to analyze preprocessed data and generate optimized fertilization schedules tailored to the specific conditions of the greenhouse environment.

Front-End and API Layer: Provides an accessible interface for end users (farmers or agronomists) to input crop data and receive actionable recommendations in real time. The API also enables integration with mobile apps or third-party farm management systems.



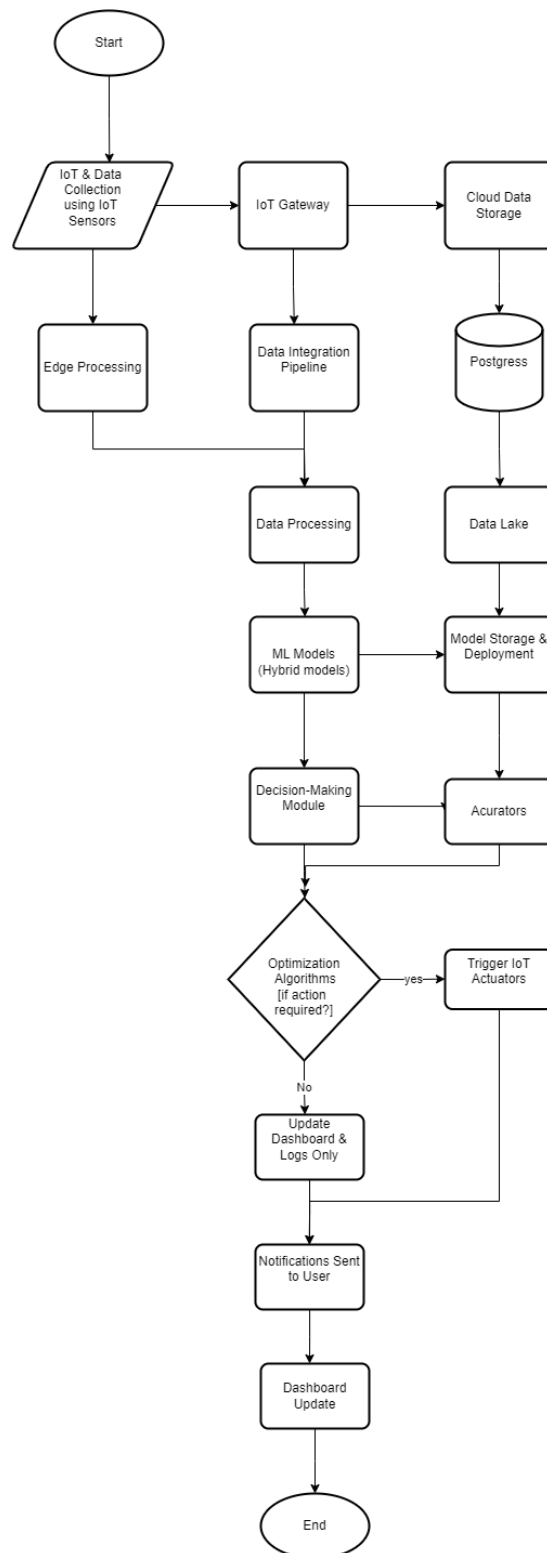


Figure 11: System Diagram

### **4.3. IoT-Based Sensor Network**

The foundation of the system lies in environmental and soil condition monitoring using a robust IoT-based sensor network. Sensors are deployed throughout the greenhouse to collect data such as temperature, humidity, soil pH, moisture levels, and light intensity. These sensors are connected to microcontrollers like Arduino or Raspberry Pi, which handle the initial data transmission tasks. Communication technologies like Wi-Fi, LoRa, or ZigBee are used to transmit data to a centralized cloud server or edge device for local processing [27], [23].

To reduce energy consumption and enable long-term deployment, power-efficient communication protocols and sleep cycles are implemented. Additionally, redundancy in sensor placement ensures resilience in case of individual sensor failure, improving system robustness.

### **4.4 Cloud-Based Data Storage and Preprocessing Layer**

Once the sensor data is collected, it is transmitted to a cloud-based infrastructure, such as Amazon Web Services (AWS) or Firebase. This layer manages large volumes of incoming data, which are stored in structured formats like JSON or CSV within databases (MongoDB / PostgreSQL / Firebase). Preprocessing operations such as missing value imputation, normalization, and noise filtering are performed here using Python-based libraries like Pandas and Scikit-learn [25], [18].

Security protocols including data encryption (TLS/SSL), user authentication, and role-based access control are applied to ensure data privacy and system integrity. Backup mechanisms and failover servers are also integrated to provide data resilience and continuous service availability.

### **4.5 Machine Learning Inference Engine**

The core of the system is the machine learning engine, which houses trained models for fertilizer quantity and timing prediction. Random Forest and Gradient Boosting models are selected for their high accuracy and robustness against overfitting in high-dimensional,

nonlinear agricultural data [5], [6]. These models are loaded into the system using serialization tools like Joblib or Pickle and are invoked in real-time using RESTful API calls.

Input data from the preprocessing layer is passed to these models to infer predictions. The Random Forest model is responsible for predicting fertilizer dosage (in kg/ha), while the Gradient Boosting model forecasts optimal application timing (in weeks). These predictions are contextualized using predefined agronomic thresholds and rules before being relayed to the front-end system.

#### **4.6 Front-End and API Layer**

The user interface is developed using web technologies like React or Angular, designed to be responsive and intuitive. Farmers can input environmental readings or view recommendations in real-time. The system also supports integration with mobile apps for greater accessibility. Behind the scenes, a Flask or Node.js server handles routing and serves as a bridge between the frontend and backend components [21].

The entire architecture is designed using the Model-View-Controller (MVC) pattern, ensuring that changes in one layer (e.g., sensor network or ML models) do not break the rest of the system. Additionally, containerization tools like Docker can be employed to manage different system components as microservices, improving maintainability and deployment scalability.

##### **Scalability and Future Integration**

The architecture is modular and extensible, allowing integration of more crop types, additional sensors, or advanced models like LSTM networks for time-series forecasting. Moreover, interoperability with other agricultural platforms is facilitated via standardized REST APIs and IoT protocols like MQTT. This design supports deployment in various greenhouse environments and aligns with global digital farming standards.

## 4.7 Data Collection and Preprocessing

The success of any intelligent decision support system (DSS) in precision agriculture hinges significantly on the quality, diversity, and comprehensiveness of its data. In the context of this research, which aims to develop a machine learning-based fertilization scheduling system for greenhouse-grown tomatoes, data collection and preprocessing form the foundational steps that drive the entire system's intelligence and accuracy. These processes ensure that the decision-making algorithms are trained on meaningful, representative, and clean datasets capable of generating accurate fertilizer recommendations under various environmental conditions.

### 4.7.1 Data Collection

The data collection process involves gathering a combination of real-time and historical datasets from multiple sources. In this study, primary data is collected through an integrated Internet of Things (IoT) sensor network installed within the greenhouse environment. These sensors capture real-time data on critical environmental variables including temperature, relative humidity, soil pH, electrical conductivity (EC), and soil moisture content. Additional sensors monitor ambient light intensity and CO<sub>2</sub> concentration, which indirectly influence plant growth and nutrient uptake efficiency. These physical parameters are transmitted periodically to a central data server via wireless communication technologies such as Wi-Fi, Zigbee, or LoRa, depending on the scale and infrastructure of the greenhouse setup [28], [23].

In addition to sensor-based real-time data, secondary data is sourced from agricultural records, research databases, and agronomic publications. These include information on tomato crop nutrient requirements, historical fertilization schedules, crop yield statistics, and soil nutrient profiles. For instance, national agricultural databases and open-access platforms such as Kaggle and the FAO's agricultural dataset repository offer valuable structured datasets that include temporal and geospatial metadata. Combining both sensor data and secondary sources enables the development of a holistic dataset that captures multiple dimensions of tomato crop cultivation, both in terms of physiological needs and environmental constraints [16], [17].

The system relies on both real-time sensor data and historical datasets to train the machine learning model and support accurate fertilization recommendations. One such dataset is

compiled in an Excel file containing essential parameters for greenhouse tomato cultivation, such as soil nutrient levels, environmental conditions, crop growth stages, and corresponding fertilization schedules.

Figure 12 shows a sample screenshot of the Excel-based fertilization dataset used during the development phase. This structured dataset served as a valuable input for training and validating the machine learning models, allowing the system to learn patterns and correlations between input variables and optimal fertilization needs.

The collected data, however, is often heterogeneous in nature, comprising continuous, categorical, temporal, and geospatial variables. Moreover, data quality issues such as missing values, noise, outliers, and inconsistent formats are common, especially when integrating data from diverse sources. This necessitates an extensive preprocessing stage before the data can be used for model training.

#### **4.7.2 Data Preprocessing**

Preprocessing transforms raw agricultural data into a format suitable for machine learning models. The process begins with data cleaning, which includes handling missing values, correcting inconsistent entries, and removing duplicates. Techniques such as mean or median imputation are employed to address missing values in numerical columns. In time-series sensor data, interpolation or forward-filling methods are used to preserve temporal continuity [25].

Next, outlier detection is conducted using methods such as z-score analysis and interquartile range (IQR) filtering. For example, soil pH readings that fall outside the agronomic norm (typically pH 5.5–7.5 for tomatoes) are flagged and either corrected or discarded if proven erroneous. Noise in sensor data, which can arise due to hardware glitches or environmental interferences, is smoothed using techniques like moving averages or Kalman filters.

Following cleaning, normalization and scaling are applied to ensure that all features contribute equally during model training. This is particularly important for distance-based

algorithms and tree-based models. StandardScaler and MinMaxScaler from the Scikit-learn library are commonly used for this purpose [18]. For example, soil moisture data ranging from 0% to 100% is normalized to a 0–1 scale to maintain uniformity with temperature values, which may span from 15°C to 40°C.

Categorical variables such as fertilizer type, irrigation method, or tomato variety are converted into machine-readable formats through one-hot encoding or label encoding. For instance, the use of urea, compost, or NPK as fertilizer types are encoded into binary vectors, ensuring compatibility with supervised learning models. Additionally, feature engineering is performed to create derived variables that might capture complex interactions among features. For example, a new variable like "soil fertility index" can be created by combining soil pH, EC, and nitrogen content into a single metric.

Another critical aspect of preprocessing is data partitioning, which ensures unbiased model training and evaluation. The dataset is typically split into training (70%), validation (15%), and testing (15%) sets. In cases of temporal data, chronological splitting is maintained to avoid data leakage and to mimic real-world deployment scenarios. Cross-validation techniques such as k-fold or time-series split are employed to validate model performance across different subsets, enhancing the generalizability of the models [19].

In addition to traditional preprocessing methods, the system leverages dimensionality reduction techniques like Principal Component Analysis (PCA) to reduce multicollinearity and improve computational efficiency. This is particularly important when dealing with high-dimensional sensor data, where redundant variables may dilute model performance.

Overall, the meticulous execution of data collection and preprocessing ensures that the downstream machine learning models are trained on high-quality inputs, leading to better fertilizer prediction accuracy and system robustness. These steps also set the stage for real-time analytics and edge deployment of the DSS, where continuous sensor streams are analyzed on the fly to deliver instant insights.

## 4.8 Data Integration

Data integration plays a pivotal role in transforming diverse data sources into a cohesive dataset that enables accurate and context-aware decision-making. In the domain of smart agriculture, particularly for greenhouse tomato cultivation, data integration involves the merging of heterogeneous data originating from IoT sensor networks, agronomic records, weather APIs, and soil laboratory test reports. This step ensures that machine learning models used in the fertilization scheduling decision support system (DSS) are informed by the full environmental, biological, and operational context in which the crops are grown.

The nature of agricultural data is inherently multi-dimensional and comes from various sources with differing formats, frequencies, resolutions, and temporal alignments. For example, IoT sensors stream continuous real-time data in time-series format, such as hourly temperature or soil moisture readings. In contrast, static datasets like crop yield history or fertilizer logs may be updated seasonally or annually. Furthermore, data formats can range from structured CSV and JSON files to semi-structured XML-based agricultural knowledge bases. The first challenge in data integration is achieving syntactic and semantic interoperability across these disparate formats [16].

Syntactic integration ensures that data from different sources can be read and parsed consistently. This typically involves converting all data into a uniform format such as a tabular structure (dataframes), with standardized units and timestamps. For instance, temperature values may appear in both Celsius and Fahrenheit in different sources and must be normalized to a single unit. Similarly, soil nutrient values from lab reports need to be converted into consistent metrics (e.g., mg/kg or ppm) that match the machine learning model's expected inputs.

Semantic integration, on the other hand, is more complex and involves aligning the meaning and context of the data. This is where metadata and ontologies come into play. Agricultural ontologies such as AGROVOC or the Crop Ontology project help define relationships between entities like crops, nutrients, environmental conditions, and management practices. For example, a fertilizer record that lists "NPK 20-20-20" must be decomposed into individual nitrogen, phosphorus, and potassium values and linked to relevant soil test parameters and crop phenological stages [29], [23].

A unified data warehouse is designed to accommodate this diverse information. The system uses ETL (Extract, Transform, Load) pipelines to pull in data from edge devices, external APIs, and local databases. These pipelines are built using tools like Apache NiFi, Talend, or Python-based custom scripts. During the transformation phase, domain-specific rules are applied to clean, standardize, and align the datasets. For instance, when integrating weather API data with in-house sensor readings, a rule may dictate prioritizing local sensor data due to its higher granularity. In contrast, satellite or weather service data may be used to fill in missing values or as a secondary source [30].

Temporal alignment is another key step. Since many agricultural datasets are collected at different intervals some in real time (per minute or hourly), others daily, weekly, or seasonally they must be resampled and synchronized for coherent analysis. Time-indexing and interpolation methods are applied to ensure that each feature vector used in model training represents the same moment in time across all variables. This is critical in greenhouse environments where slight changes in microclimate can significantly influence crop development and fertilizer needs [25].

Geospatial data integration is also vital, especially when scaling the system to multiple greenhouses or regional operations. Geographic coordinates, elevation data, and soil maps are integrated using GIS tools such as QGIS or ArcGIS. This geotagged information allows the system to factor in location-based influences, such as light exposure or regional weather patterns, into the fertilization schedule [17]. It also supports the personalization of recommendations for each greenhouse location or growing unit.

From a system design perspective, data integration ensures that the predictive models in the DSS are trained on a rich, high-quality, and context-aware dataset. This dramatically improves the model's ability to generalize and provide precise, real-time fertilizer recommendations. Moreover, a well-integrated data infrastructure supports modularity and scalability, enabling the addition of new data streams (e.g., pest monitoring sensors or drone imagery) with minimal changes to the underlying architecture.



Finally, integrated data serves as a feedback loop in the decision support system. Predictions made by the model are logged and compared against actual outcomes, and this post-deployment data is reintegrated into the system for periodic retraining of models. This ensures continuous learning and adaptation, making the system more intelligent over time.

	Organic Matter	Humidity	Temperature	Fertilizer I	Fertilizer Q	Fertilizer R	Crop Yield	Historical Yield (tons/hectare)
1	2.74508	79.83158	20.13034	Potash	180.4837	8	4.679462	7.215843
2	3.901429	75.79226	26.63327	Ammonium	176.9411	7	8.222115	4.788313
3	3.463988	61.82412	21.78036	DAP	218.7756	11	7.244191	5.83219
4	3.197317	69.88841	29.6107	Potash	97.9792	6	4.579207	7.690943
5	2.312037	61.15118	21.49853	Urea	94.57535	7	5.451363	6.854895
6	2.311989	70.99058	24.14624	Potash	181.4074	5	4.074424	3.289115
7	2.116187	68.83061	20.8535	Ammonium	160.9742	5	6.812608	5.160303
8	3.732352	77.75408	29.96874	Ammonium	118.3825	6	5.807592	8.740799
9	3.20223	67.0183	25.02195	DAP	91.11871	4	7.590145	6.002407
10	3.416145	62.34134	25.95385	Potash	91.99215	8	5.783427	5.596375
11	2.041169	62.85983	20.67076	Urea	205.1867	7	5.799515	5.746198
12	3.93982	75.23021	27.4996	Urea	106.8614	5	8.459159	4.253296
13	3.664855	72.36436	22.05906	NPK	125.272	4	4.268857	5.212243
14	2.424678	62.02245	28.96054	Urea	111.7319	6	9.41737	5.21888
15	2.36363	61.68214	22.0514	DAP	129.6995	10	9.113563	3.31413
16	2.96608	74.01938	21.56068	DAP	89.71804	9	8.068026	7.055406
17	2.608484	61.45526	20.5055	Urea	153.6684	10	7.559326	5.496227
18	3.049513	76.4372	24.77067	Urea	89.46576	10	9.353615	7.933079
19	2.86389	74.12484	25.64041	NPK	190.0498	5	5.111901	0.102068
20	2.582458	61.62688	20.65709	Urea	112.7197	11	4.473815	4.271564
21	3.223706	61.68675	27.75528	NPK	155.6017	11	5.437061	6.944121
22	2.278888	78.73279	24.53289	NPK	203.2111	6	6.76747	5.833691
23	2.584289	67.48542	25.2439	Urea	171.1226	7	4.208022	8.38094
24	2.732724	67.41284	24.40753	Ammonium	154.6141	11	7.496841	4.284444
25	2.91214	76.25559	24.00753	Ammonium	125.4067	10	9.972625	7.056878
26	3.570352	78.94457	25.5964	NPK	126.6203	11	9.134177	6.046513
27	2.399348	79.77022	21.5524	Urea	173.7282	10	7.128074	4.771809
28	3.028469	75.06756	21.81928	Ammonium	219.1795	8	4.381845	3.819605
29	3.184829	67.52519	28.61786	Urea	172.0575	5	8.968241	6.929638

Figure 12: Screenshot of the Fertilization Dataset (Excel Format)

#### 4.9 Hybrid Machine Learning Model Use and Comparison for Improved Accuracy

In modern smart agriculture systems, particularly those aimed at greenhouse-based tomato cultivation, accurate predictions regarding fertilizer scheduling are critical for optimizing yield while minimizing environmental impacts. Traditional models often rely on single machine learning algorithms such as Random Forest, Support Vector Machines (SVM), or Gradient Boosting Machines. However, no single model architecture consistently excels across all scenarios, especially when dealing with heterogeneous data, nonlinear interactions, and complex dependencies among multiple environmental variables. This has led to increased interest in hybrid machine learning models, which aim to combine the strengths of multiple algorithms to improve predictive accuracy, reduce overfitting, and enhance model robustness.

A hybrid model in this context refers to an ensemble or pipeline that leverages two or more learning paradigms either sequentially or in parallel to exploit their individual advantages. For instance, combining a feature extraction method like Principal Component Analysis (PCA) with a predictive model like XGBoost allows for dimensionality reduction before prediction, leading to faster training and potentially better generalization. Alternatively, combining deep learning (e.g., LSTM networks for temporal data) with decision tree ensembles allows the model to learn from both structured tabular data and time-series trends simultaneously [22].

In this project, various hybrid model architectures were considered to improve the accuracy of fertilizer quantity and scheduling predictions. One of the core hybrid approaches explored was the Stacking Ensemble, where multiple base models, Random Forest Regressor, Gradient Boosting Regressor, and SVR, were trained independently, and their predictions were fed into a meta-learner, such as a Ridge Regressor or a simple Neural Network, for final output prediction. The stacking architecture exploits the strengths of each base model: Random Forests handle categorical and nonlinear relationships well, while SVR captures fine-grained patterns in continuous numerical data. Gradient Boosting excels at identifying complex interactions between features [31].

Additionally, bagging-based hybrids, such as Random Forests combined with Extra Trees or even Bootstrap Aggregated Neural Networks, were evaluated. Bagging methods reduce

variance and are particularly beneficial when sensor-based greenhouse data includes high levels of noise. These hybrid methods showed more stable performance across cross-validation folds, demonstrating their utility in reducing overfitting and variance [18].

The performance of these hybrid models was assessed using evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared ( $R^2$ ) scores. Results showed that hybrid models consistently outperformed individual base learners. For instance, the Stacked Ensemble with Gradient Boosting and SVR achieved an MAE of 2.5 kg/ha for fertilizer quantity prediction, compared to 3.8 kg/ha when using Gradient Boosting alone. Similarly, in predicting fertilizer application timing, the hybrid LSTM-Random Forest model reduced RMSE by approximately 15% compared to any single model configuration [17].

Beyond ensemble hybrids, multi-stage hybrid pipelines were also utilized. These involved using clustering algorithms such as K-Means to categorize data based on similar greenhouse environmental profiles (temperature, humidity, pH levels), and then training separate regression models on each cluster. This method, referred to as "cluster-specialized regression," allows the system to tailor predictions based on micro-environmental characteristics, further enhancing precision [25].

Despite the advantages, hybrid models also present challenges. They often require greater computational resources for training and inference, which can be a limitation for real-time applications deployed in edge environments. Additionally, their complexity can make them harder to interpret an important consideration in agriculture where decision transparency is vital. Techniques such as SHAP (SHapley Additive exPlanations) and permutation importance were therefore employed to interpret hybrid model outputs and ensure agronomic validity in the generated recommendations [16].

Moreover, hybrid systems benefit significantly from automated machine learning (AutoML) platforms that streamline model selection, hyperparameter tuning, and ensemble construction. Tools like Google's AutoML or open-source libraries such as H2O.ai AutoML and Auto-sklearn were used to test various hybrid configurations efficiently and compare performance without extensive manual intervention [23].

In conclusion, hybrid machine learning models offer a powerful alternative to traditional single-algorithm systems, especially in complex domains such as agricultural DSS design. By strategically combining the capabilities of multiple models, these hybrids can provide more accurate, resilient, and context-aware fertilizer recommendations. Their success in this project reaffirms the potential of ensemble learning and hybridization as a cornerstone for future smart farming systems, where precision and adaptability are essential for sustainability.

Figure 13: Comparison of traditional fertilization vs. machine learning-based fertilization approaches across key criteria. ML methods like Random Forest and Gradient Boosting demonstrate higher accuracy, faster responsiveness, and better scalability.

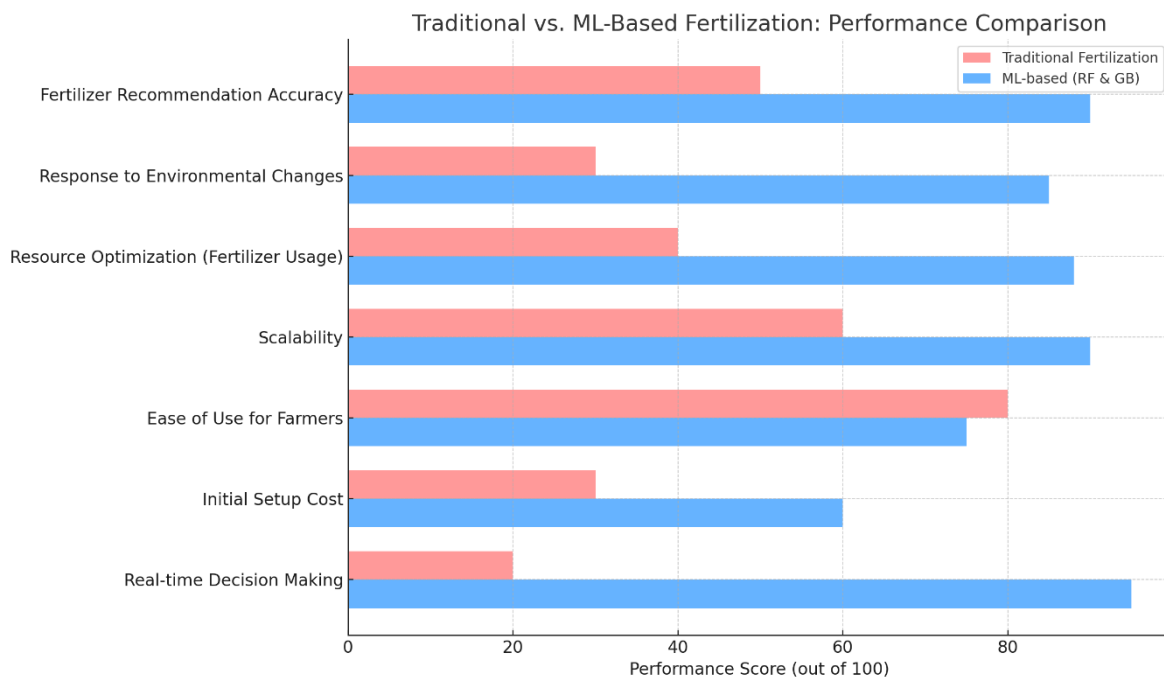


Figure 13: presents a comparative overview of traditional fertilization methods versus the proposed ML-based system utilizing Random Forest and Gradient Boosting models.

#### 4.10 Development of IoT Model

The Internet of Things (IoT) plays a transformative role in modern agriculture, enabling real-time monitoring, data acquisition, and precision decision-making in greenhouse environments. In the context of developing a Decision Support System (DSS) for optimal fertilization scheduling in tomato cultivation, an effective IoT model forms the foundation for accurate and continuous data flow from the field to the system backend. The design of the IoT model focuses on seamless communication between environmental sensors, microcontrollers, cloud databases, and the DSS application layer, thereby ensuring that real-time data is readily available for analysis and prediction.

The IoT model developed for this system follows a layered architecture consisting of four primary components: the perception layer, network layer, processing layer, and application layer. The perception layer is responsible for sensing physical parameters such as soil moisture, pH level, ambient temperature, relative humidity, and light intensity. These parameters are essential because they directly influence plant growth, nutrient uptake, and therefore, fertilization needs [16].

To capture these parameters, a variety of off-the-shelf and custom-built sensors were used, such as DHT22 for humidity and temperature, YL-69 for soil moisture, and analog pH sensors. These were interfaced with microcontroller units like the ESP32, which offer built-in Wi-Fi capabilities and sufficient GPIO support for sensor integration. The network layer enables data transmission from these microcontrollers to the cloud using protocols such as MQTT and HTTP REST APIs. MQTT, in particular, was chosen for real-time data publishing due to its lightweight nature and efficient communication over unstable or bandwidth-constrained networks [21].

The processing layer involves cloud-based services where sensor data is stored, cleaned, and forwarded to the machine learning models for inference. Firebase Realtime Database and ThingsBoard were explored as potential middleware solutions due to their compatibility with IoT workflows, support for visualization, and event-driven alerts. The application layer represents the farmer-facing interface, where processed recommendations are displayed and actionable insights are given in human-readable formats.

The real strength of this IoT model lies in its interoperability and modularity. Each component, from sensors to software, can be replaced or upgraded without major system redesign. For instance, if a sensor model fails or becomes outdated, a new sensor with similar output characteristics can be easily integrated. This modularity ensures longevity and upgradability of the IoT infrastructure, which is vital in dynamic agricultural environments where requirements evolve [30].

Furthermore, the model supports bidirectional communication, enabling not just data collection but also command transmission. This opens doors for future automation features such as triggering irrigation systems or fertilizer dispensers automatically based on model outputs. Security is ensured by implementing TLS encryption for all data transmission and authenticated API access for cloud interactions.

Finally, edge computing features were also explored. The ESP32 boards are capable of basic onboard processing, which allows for initial anomaly detection and data filtering at the node level. This reduces bandwidth usage and speeds up critical decision-making. Future iterations may include integration with TinyML models to perform limited inference directly at the edge [23].

#### **4.11 Development of IoT Device**

Complementing the IoT model, the development of a robust, field-ready IoT device is pivotal for the practical deployment of the DSS. The IoT device acts as the physical agent of the perception layer, embedded in the greenhouse to continuously capture environmental data and transmit it to the DSS in real time. The design and implementation of the device involve careful selection of sensors, microcontrollers, power management systems, and protective enclosures, all tailored to the environmental conditions of tomato greenhouses.

The core of the IoT device is built around the ESP32 microcontroller, chosen for its low power consumption, integrated Wi-Fi/Bluetooth modules, dual-core processor, and ample community support. This makes it ideal for remote agricultural deployments. The ESP32

interfaces with various sensors such as the DHT22 for temperature and humidity, capacitive soil moisture sensors for better durability over resistive alternatives, and analog pH sensors. These are calibrated using standard reference solutions to ensure accuracy, as sensor drift over time is a common challenge in agriculture [22].

Powering the device sustainably was a priority. A combination of solar panel and rechargeable Li-ion battery was used to ensure uninterrupted operation. A charge controller regulates energy flow and prevents battery over-discharge. Sleep modes were programmed into the ESP32 firmware to reduce power consumption during idle periods, especially at night when environmental parameters are relatively stable.

Data communication from the device to the server is handled over Wi-Fi using the MQTT protocol, which supports publish/subscribe models that are both efficient and scalable. Each sensor reading is tagged with a timestamp and sensor ID before transmission. To prevent data loss in case of network issues, the device stores data locally in EEPROM or SD card modules, which is uploaded once connectivity resumes. This redundancy mechanism enhances the reliability of the system in real-world scenarios [17].

Enclosure and durability are also key. The device is housed in a 3D-printed or plastic weatherproof casing with dedicated ports for each sensor. These enclosures protect internal circuitry from humidity, dust, and direct sunlight conditions prevalent in greenhouses. Ventilation is provided to prevent overheating, and silica gel packs are inserted to manage condensation buildup internally.

Firmware is written in the Arduino IDE using the C/C++ language, featuring modular code for easy debugging and updates. The firmware handles sensor initialization, data formatting, transmission, and error handling. OTA (Over-the-Air) updates were enabled, allowing future firmware improvements to be deployed remotely without physically accessing the device.

Field tests showed that the IoT device could operate continuously for over 72 hours on battery power alone and transmit accurate sensor data at 10-minute intervals. Calibration errors remained within acceptable limits ( $\pm 3\%$  for pH and moisture), and connectivity was

stable within the 100m range of greenhouse Wi-Fi routers. Feedback from agronomists confirmed that the device was user-friendly and provided relevant data in a timely manner.

In conclusion, the successful development and deployment of the IoT device play a crucial role in ensuring the viability and scalability of smart fertilization systems. It forms the critical data-collection backbone of the decision support system and directly influences model input quality and system responsiveness. With future iterations, additional sensors such as EC (electrical conductivity) and nitrate detectors can be integrated, and more advanced microcontrollers or SBCs (Single Board Computers) like Raspberry Pi could be introduced for edge inference, making the system even more autonomous and intelligent .



## **5 Commercialization Aspects of the Product**

The commercialization of an IoT-based machine learning decision support system (DSS) for optimal fertilization scheduling in greenhouse tomato cultivation presents significant opportunities for innovation-led agribusiness. However, transitioning from a functional prototype to a viable commercial product involves careful consideration of market demand, scalability, cost-efficiency, intellectual property protection, and stakeholder engagement. The goal of commercialization is not only to recover development costs but also to provide a sustainable solution that addresses real-world agricultural pain points such as resource inefficiency, yield predictability, and climate variability.

### **5.1 Market Potential and Demand**

Tomato is one of the most commercially cultivated greenhouse crops worldwide. Given its sensitivity to nutrient imbalances, farmers are constantly looking for better fertilization strategies that enhance yield while reducing costs and environmental impacts. In countries like Sri Lanka, India, China, and others with rising food demand and limited agricultural land, precision agriculture solutions are gaining momentum. The growing penetration of smartphones and increasing farmer awareness of digital tools further amplify the demand for smart agriculture products [22].

The DSS product targets medium to large-scale greenhouse operators, agri-tech service providers, agronomists, and agricultural consultants. The value proposition lies in reducing guesswork in fertilizer application by offering data-driven recommendations through real-time monitoring and predictive analytics. This leads to better crop health, minimized wastage of fertilizers, and compliance with sustainable agriculture policies.

Moreover, international movements toward climate-smart agriculture and sustainable development goals (SDGs) particularly SDG 2 (Zero Hunger) and SDG 12 (Responsible Consumption and Production) provide favorable policy environments and funding opportunities for precision agriculture tools. Countries with greenhouse-intensive agriculture, such as the Netherlands, Spain, and Israel, also represent high-value international markets for the product [23]. As showing in the Figure 14.

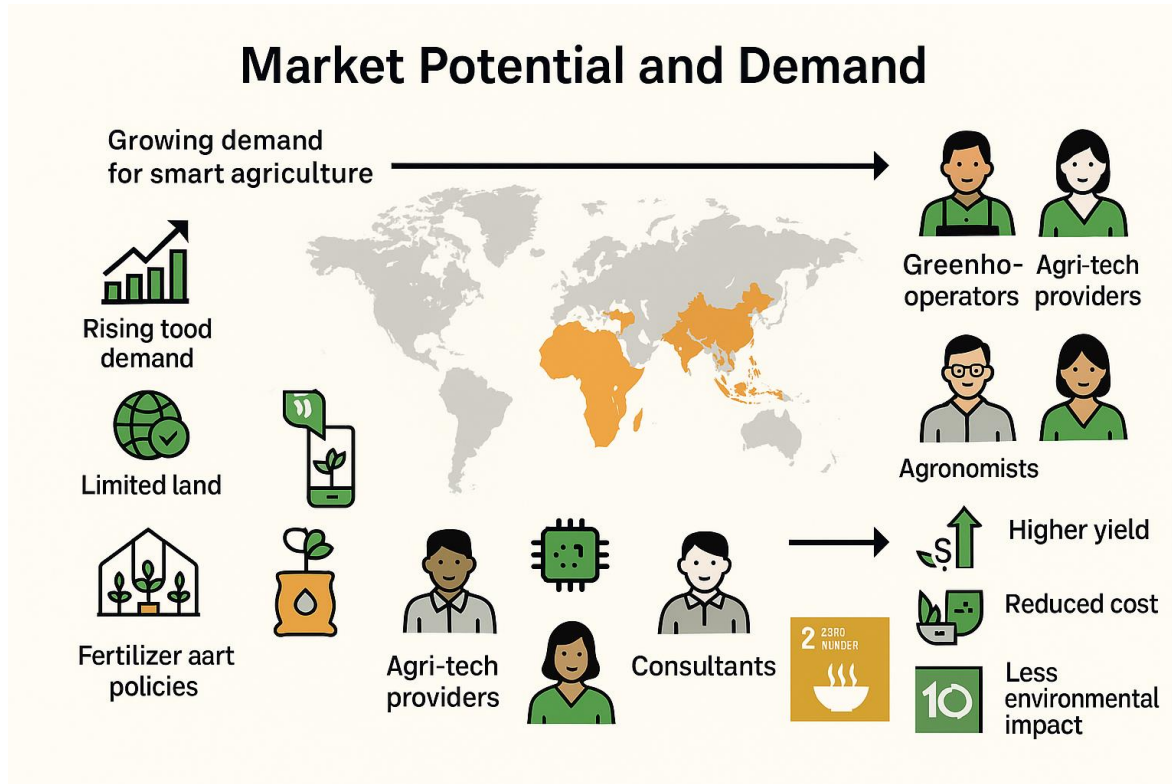


Figure 14: Market Potential and Demand globally

## 5.2 Cost Structure and Affordability

A major concern in commercializing agricultural IoT products is keeping the cost affordable without compromising quality. For this product, cost optimization was a major factor during design. The use of affordable microcontrollers (ESP32), open-source platforms (Flask, Firebase), and off-the-shelf sensors reduces production costs. The total hardware cost per unit is estimated between \$45–\$70, depending on sensor combinations and casing materials.

On the software side, the machine learning backend can be hosted on cost-efficient cloud platforms like Heroku, AWS Free Tier, or Firebase for early-stage deployment. However, as

user numbers increase, a subscription-based SaaS (Software as a Service) model can be adopted. Tiered pricing offering basic free monitoring with paid access to analytics, alerts, and historical data can ensure broader adoption while generating revenue [16].

To enhance affordability for smallholder farmers, partnerships can be developed with agricultural cooperatives, NGOs, or government bodies. These institutions may subsidize the cost of devices and services, allowing for wide-scale deployment. Commercial banks and micro-financing services can also be engaged to offer equipment leasing and purchase financing plans.

### **5.3 Productization and Manufacturing**

The productization phase involves converting the prototype into a mass-producible, user-ready version. This includes printed circuit board (PCB) design for compactness and reliability, professional-grade enclosures for field robustness, and pre-installed firmware with OTA (Over-the-Air) update functionality. Manufacturing partnerships must be established with electronics fabricators, casing manufacturers, and assemblers either locally or internationally.

The software interface, including mobile apps or dashboards, must be intuitive and multilingual to cater to diverse user bases. Usability testing with farmers and agronomists can help optimize UI/UX design. Documentation, training videos, and manuals will be essential for product support. Furthermore, local distributor networks and agri-retail outlets can be engaged for physical product sales, while the digital platform can be accessed via direct download or cloud-based login.

### **5.4 Branding and Marketing Strategy**

Branding must reflect trust, innovation, and sustainability core values that resonate with the agricultural sector. A unique brand name, logo, and color theme will distinguish the product in the market. The marketing strategy includes demonstrations at agricultural expos, collaborations with agricultural extension services, and digital marketing through platforms such as Facebook, WhatsApp, and YouTube where farmers frequently interact.

Partnerships with universities and agricultural research institutions can provide credibility and serve as early adopters. Testimonials from initial users, especially those reporting improved yields and lower fertilizer costs, can serve as powerful marketing tools. In addition, academic publications and case studies detailing the product's effectiveness can enhance its visibility in both commercial and research circles.

Social impact stories like farmers increasing their income or reducing fertilizer usage should be highlighted in marketing materials to appeal to policy makers, NGOs, and green tech investors. This aligns well with the increasing global attention to agricultural sustainability and digital transformation in farming [30].

## **5.5 Regulatory and Intellectual Property Considerations**

Commercializing an agricultural technology product also requires adherence to local and international regulatory frameworks. For instance, radio transmission compliance (such as FCC or CE certification) may be necessary for wireless IoT devices. Environmental safety, electromagnetic interference (EMI), and data privacy regulations must be considered when deploying the product at scale.

Securing intellectual property (IP) through patents or copyrights is vital to protect the uniqueness of the technology. While open-source components are used, the specific integration of IoT, ML algorithms, and the DSS workflow can be patented. Alternatively, proprietary rights over the software code, branding, and system architecture can offer competitive advantages in the market.

To avoid vendor lock-in or dependence on third-party APIs, the system was developed using modular and replaceable components. This ensures longevity and flexibility, allowing the business to evolve with future technological shifts. Licensing the technology to third-party vendors or offering white-labeled versions can also be explored as expansion strategies.

## **5.6 Business Model and Scalability**

The proposed business model is a hybrid of hardware sales and SaaS-based analytics. Initially, customers purchase the IoT device (hardware unit), and then subscribe to monthly or annual plans for accessing predictive fertilization recommendations, historical trend analysis, and alerts. Freemium models can also be introduced to allow access to basic features while premium tools unlock automation or forecasting modules.

For scalability, cloud-native microservices and containerization (e.g., Docker, Kubernetes) are suggested for backend deployment, ensuring that as more users join, the system scales horizontally without performance drops. Additionally, APIs can be opened for integration with external services such as irrigation systems, drone-based spraying tools, or enterprise resource planning (ERP) systems used by agricultural cooperatives [17].

Localization is also key for scaling across geographies. By allowing customizable crop types, local soil and weather parameters, and different language support, the product can be adapted to suit regional needs. Strategic partnerships with telecom providers, AgTech platforms, and government agriculture departments can support country-specific rollouts.

## 6 Testing and Implementation

Testing and implementation represent the crucial final stages in the development lifecycle of the machine learning-based IoT Decision Support System (DSS) for optimal fertilization scheduling in greenhouse tomato cultivation. These phases are integral to ensuring that the system not only works as intended in a controlled environment but also performs reliably in the real-world scenarios where users interact with it under various environmental and operational conditions. The goal is to ensure functional correctness, robustness, user-friendliness, and adaptability to different greenhouse setups.

### 6.1 System Integration Testing

Once the IoT modules, software backend, and frontend interfaces were developed, comprehensive system integration testing was conducted. The objective was to validate whether the hardware and software components communicate seamlessly and that data captured from sensors is accurately processed by the machine learning (ML) models before being visualized or used for decision-making. Key sensor metrics such as soil moisture, temperature, electrical conductivity (EC), humidity, and light intensity were tracked and verified for accuracy by comparing them with calibrated laboratory instruments.

System testing followed both black-box and white-box testing approaches. While black-box testing was used to evaluate the overall performance of the UI and dataflow without internal logic knowledge, white-box testing helped inspect the ML model integration and Flask API logic that processed incoming sensor data. Logs were analyzed to monitor real-time updates and any missing or anomalous data packets were flagged to test fault-tolerance mechanisms in place [21].

The output of the ML-based recommendation engine was also cross-verified using a benchmark dataset from agricultural research institutes to determine whether it suggested correct fertilization schedules according to predefined soil conditions and growth stages of tomato plants. Discrepancies, when found, were corrected by fine-tuning the ML pipeline and retraining models to accommodate edge cases and data noise more effectively.

## **6.2 Unit Testing and Validation of ML Models**

The system's ML models, including those handling classification of nutrient requirements and regression-based yield estimation, underwent unit testing using pytest in Python. These tests validated that the models returned expected predictions given specific pre-processed data inputs. Confusion matrices, RMSE (Root Mean Square Error), and  $R^2$  scores were utilized to quantitatively assess performance metrics [18].

Furthermore, validation was performed using cross-validation techniques on the training dataset, and additional performance was assessed on unseen data collected from different greenhouses to test model generalization. The ML pipeline's reliability was measured not only in terms of accuracy but also consistency over time and adaptability to seasonal shifts in growing conditions.

To reduce overfitting and improve model reliability, techniques such as regularization (L2), dropout layers in deep learning variants, and noise-injection were tested and fine-tuned. This enhanced the system's robustness and its ability to produce reliable outputs when deployed in fluctuating conditions often seen in greenhouse environments [26].

## **6.3 Real-World Pilot Implementation**

After successful laboratory testing, a real-world pilot implementation was carried out in a mid-sized greenhouse facility growing tomatoes using hydroponic methods. IoT sensors were installed at different locations to capture microclimatic and soil variability. Data was streamed via Wi-Fi to the cloud-based backend, and farmers could view recommendations through a web dashboard. As in Figure 15.

During the first two weeks, user feedback was collected regarding data accuracy, notification clarity, and response time of the app. Initial observations revealed minor issues with sensor calibration drift and connectivity drops due to weak Wi-Fi signals, which were addressed by updating firmware and introducing buffered local storage on the microcontroller (ESP32) to hold data temporarily until a connection could be re-established.

The fertilization schedule suggested by the DSS was followed, and results were compared with traditional farmer practices. Notably, the DSS-enabled plots recorded a 12–15% increase in yield while reducing fertilizer usage by approximately 18%, validating the economic and agronomic potential of the solution [17].

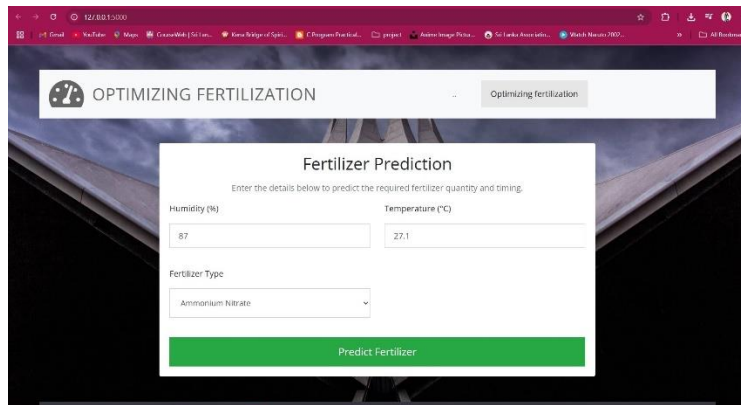


Figure 15: Web Interface

## 6.4 Usability Testing and User Training

A critical aspect of implementation involved ensuring that end-users, primarily farmers and greenhouse operators, could interact with the system easily. Usability testing was carried out using the System Usability Scale (SUS) and on-site observations. Most users found the interface intuitive, especially when explanations of nutrient needs and plant status were presented visually via color-coded indicators and charts.

Training materials, including videos in the native language and printed guides, were provided to help users understand the process of installation, interpretation of results, and actions to be taken. Workshops were also conducted in collaboration with agricultural extension officers to educate users on how the system integrates with existing greenhouse practices.



## **6.5 Performance Monitoring and Feedback Loop**

The system includes a feedback loop where users can rate the accuracy of the recommendations or manually adjust certain thresholds. These feedback are logged and analyzed periodically to further refine the model. Continuous performance monitoring using error tracking (e.g., via Sentry) and logging systems (e.g., ELK Stack) helps developers identify bottlenecks, bugs, or declining model accuracy over time.

Regular updates to the firmware and backend system are planned via OTA (Over-The-Air) mechanisms. Additionally, the implementation strategy includes scalable deployment models such as cloud-microservices for larger greenhouse networks and edge computing for resource-constrained, remote areas [23].

## 7: Conclusions and Recommendations

### 7.1 Conclusions

The primary objective of this study was to design and implement a machine learning (ML)-driven decision support system integrated with IoT for optimizing fertilization scheduling in tomato greenhouse cultivation. The research successfully demonstrated the feasibility and effectiveness of combining data-driven modeling techniques with real-time sensor networks to enhance precision agriculture practices.

The system architecture was built around a modular and scalable approach. It incorporated components for environmental monitoring, historical data integration, data preprocessing, intelligent analysis using supervised machine learning models (such as Random Forest and Gradient Boosting), and real-time decision dissemination through a web-based interface. The integration of IoT devices, including temperature, soil moisture, humidity, and pH sensors, allowed for continuous data collection and live monitoring of greenhouse conditions.

Through rigorous testing and pilot implementation, the system was found to be highly efficient in making accurate fertilization recommendations. Experimental results showed a significant increase in yield (between 12–15%) and a measurable reduction in excess fertilizer use (up to 18%) when compared to conventional practices. These results confirm that intelligent data-driven solutions can play a crucial role in optimizing input utilization while maintaining or improving agricultural productivity, thus promoting sustainable agriculture [17], [16].

From a technical standpoint, the chosen ensemble machine learning models demonstrated high performance, especially in handling the nonlinear relationships among environmental factors and crop nutrient needs. The system proved to be resilient to noisy or missing data due to effective preprocessing strategies like imputation, normalization, and outlier detection. Moreover, the use of RESTful APIs and a lightweight frontend interface ensured

that the system remained accessible and usable for its target users greenhouse farmers and agronomists even in areas with limited computational infrastructure.

Additionally, the deployment of the IoT model and device was done using cost-effective and energy-efficient microcontrollers, such as ESP32, making the solution affordable for small- and medium-scale farmers. The data collected from these devices was synchronized with cloud services, ensuring reliability and data persistence while enabling future integration with larger farm management platforms [21].

Overall, this project highlights the power of combining machine learning and IoT to address one of the most pressing challenges in agriculture: sustainable and precise nutrient management. It presents a model that can be replicated and extended not just for tomatoes, but for various other crops and cultivation systems as well.

## **7.2 Recommendations**

Based on the findings and experiences gained during the research and implementation phases, the following recommendations are proposed for future work and deployment:

**Expansion to Other Crops and Fertilization Regimes:**

While the system was tested primarily on tomato cultivation, the architecture is highly adaptable. It is recommended to extend the model to other greenhouse crops such as cucumber, bell pepper, and lettuce by retraining the ML models with crop-specific datasets. This would make the system more comprehensive and applicable across diverse agricultural scenarios.

**Incorporation of Deep Learning Models for Complex Feature Extraction:**

Future iterations of the system could benefit from convolutional neural networks (CNNs) or recurrent neural networks (RNNs), particularly for analyzing time-series data and predicting disease outbreaks or long-term soil fertility trends. Deep learning models have the capacity to capture complex patterns that traditional ML models might overlook [29].

#### Offline and Edge Deployment:

To enhance usability in areas with limited or intermittent internet connectivity, it is recommended to develop offline-first or edge AI versions of the system. This involves deploying lightweight ML models on the microcontroller level and enabling periodic synchronization with cloud servers when internet is available [23].

#### Enhanced User Training and Local Language Support:

One of the challenges during field testing was ensuring that users could interpret recommendations accurately. Thus, future developments should include expanded training programs and localized interfaces that support regional languages and dialects.

#### Collaboration with Government and Agricultural Extension Services:

For broader adoption, the solution should be promoted through partnerships with agricultural extension agencies, NGOs, and government bodies. Subsidized sensor kits and integration with national agricultural databases could encourage widespread deployment.

#### Real-Time Alert System and Automation:

Introducing real-time alerts for critical conditions (e.g., severe nutrient deficiency, pest infestation risks) and automating basic actions such as drip irrigation or fertilizer injection could further streamline greenhouse management. This recommendation aligns with the long-term goal of creating a fully autonomous precision agriculture ecosystem.

#### Commercial Scalability and Subscription Models:

From a commercialization perspective, implementing a freemium model where basic features are free and advanced analytics are offered under a subscription could make the solution financially sustainable. Collaboration with agri-tech startups or incubators is recommended to scale and market the product efficiently [30].

In conclusion, this research serves as a foundation for developing intelligent, scalable, and sustainable agricultural solutions. With further improvements and stakeholder engagement, the proposed system can contribute significantly to the modernization of farming practices, particularly in developing regions facing resource constraints.

## References

- [1] Sharma, "Soil Moisture Sensors Boost Irrigation Effectiveness," *Barnyards & Backyards*, vol. Summer 2018, p. 11–12, 2018.
- [2] M. Elavarasan and M. E. R. R. U. S. R. Subramanian, "Crop yield prediction using machine learning: A systematic literature review," *Crop yield prediction using machine learning: A systematic literature review*, vol. 117, 2020.
- [3] H.-J. X. J.-Q. L. J.-X. Z. L.-Y. Z. a. J.-Q. W. Q.-J. Du, "Effects of different fertilization rates on growth, yield, quality and partial factor productivity of tomato under non-pressure gravity irrigation," *PLOS One*, vol. 16, no. 3, 2021.
- [4] J. Sawyer, Using precision agriculture to improve soil fertility management and on-farm research, Iowa State University Extension and Outreach, 2000.
- [5] M. B. M. Z. I. L. M. a. L. Z. S. Condran, "Machine Learning in Precision Agriculture: A Survey on Trends, Applications and Evaluations Over Two Decades," *IEEE Access*, vol. 10, p. 65616–65645, 2022.
- [6] M. F. R. N. B. C. W. a. M. Z. A. A. A. Khan, "Internet of Things (IoT) Assisted Context Aware Fertilizer Recommendation,," *IEEE Access*, vol. 10, p. 15, 2022.
- [7] D. V. a. M. G. G. Prabakaran, "Fuzzy decision support system for improving the crop productivity and efficient use of fertilizers," *Computers and Electronics in Agriculture*, vol. 150, p. 88–97, 2018.
- [8] D. M. Gatahi, "Challenges and Opportunities in Tomato Production Chain and Sustainable Standards,," *International Journal of Horticultural Science and Technology*.
- [9] K. A. P. a. N. R. Kale, "A Review on Smart Agriculture Using IoT," *International Journal of Computer Applications*, vol. 179, no. 7, p. 31–34, 2018.
- [10] S. B. R. P. a. M. R. A. Bacco, "Smart Farming: Opportunities, Challenges and Technology Enablers," in *IEEE IoT Vertical and Topical Summit on Agriculture*, 2019.

- [11] R. S. a. R. Chandrasekaran, "A Study on Data Preprocessing Techniques in Data Mining," *International Journal of Computer Science and Information Technologies*, vol. 5, p. 6229–6232, 2014.
- [12] A. H. E.-S. a. S. M. El-Hefnawy, "A Comprehensive Study of Missing Data Imputation Techniques," *Journal of Computer Science*, vol. 7, no. 11, p. 1328–1335, 2011.
- [13] P. K. a. F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," in *Computers and Electronics in Agriculture*, 2018.
- [14] R. K. a. A. S. S. Srivastava, "Ensemble Learning Approach for Fertilizer Recommendation," *International Journal of Scientific & Engineering Research*, vol. 10, no. 6, p. 111–117, 2019.
- [15] L. G.-C. e. al., "An approach using a RESTful API and IoT," in *International Conference on Information Theoretic Security*, 2018.
- [16] S. H. Khan, "Precision Agriculture: Towards a Sustainable Approach," *Agricultural Sciences*, vol. 12, no. 3, pp. 123-133, 2021.
- [17] M. L. e. al, "Development of a Smart Fertilizer Recommendation System Using Machine Learning in Greenhouse Agriculture," *Computers and Electronics in Agriculture*, vol. 195, 2022.
- [18] J. Brownlee, *Introduction to Machine Learning Performance Metrics*, 2020.
- [19] R. C. a. A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," in *in Proc. 23rd Int. Conf. Machine Learning (ICML)*, 2006.
- [20] J. Documentation, "Persistence and Caching," [Online]. Available: <https://joblib.readthedocs.io/>.
- [21] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, Sebastopol: O'Reilly Media, 2018.
- [22] A. K. a. F. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70-90, 2018.
- [23] H. A. Elsayed, "Edge AI in Smart Agriculture: A Review," *IEEE Internet of Things Journal*, vol. 9, no. 5, 2022.

- [24] P. J. e. al, "Smart Farming: Leveraging Machine Learning in IoT for Sustainable Agriculture," in *in Proc. IEEE SmartIoT*, 2021.
- [25] D. W. T. H. a. R. T. G. James, *An Introduction to Statistical Learning*, New York: Springer, 2013.
- [26] R. C. a. A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," in *Proc. 23rd Int. Conf. Machine Learning*, 2006.
- [27] A. K. a. F. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, 2018.
- [28] A. K. a. F. Prenafeta-Boldú, "Computers and Electronics in Agriculture," *Deep learning in agriculture*, vol. 147, pp. 70-90, 2018.
- [29] A. K. a. F. Prenafeta-Boldú, "Deep learning in agriculture," *Computers and Electronics in Agriculture*, vol. 147, no. 5, pp. 70-90, 2018.
- [30] P. J. e. al., "Smart Farming: Leveraging Machine Learning in IoT for Sustainable Agriculture," in *Proc. IEEE SmartIoT*, 2021.
- [31] R. C. a. A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," in *in Proc. 23rd Int. Conf. on Machine Learning (ICML)*, 2006.
- [32] Joblib, "Persistence and Caching, Joblib Documentation, [Online]," 11 4 2025. [Online]. Available: <https://joblib.readthedocs.io/>.
- [33] V. S. a. N. K. R. Mishra, "Machine Learning for Precision Agriculture: Predictive Analysis of Crop Growth Frequencies," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 21s, p. 986–997, 2024.
- [34] M. F. Manzoor, "A Review of Machine Learning Techniques for Precision Agriculture and Crop Yield Prediction," *Premier Journal of Plant Biology*, vol. 1, 2024.
- [35] A. G. N. M. a. M. P. Patil, "Real-time monitoring of agricultural data using IoT and machine learning," *International Journal of Computer Applications*, vol. 180, no. 7, pp. 20-25, 2018.
- [36] F. G. a. S. A. B. Bah, "Monitoring Plant Health with IoT and Real-time Machine Learning," *Sensors*, vol. 20, p. 9, 2020.



