# SOIL QUALITY AND MANAGEMENT USING DEEP LEARNING AND INTERNET OF THINGS FOR PRECISION AGRICULTURE

*A Project Report Submitted in the*
*Partial Fulfillment of the Requirements*
*for the Award of the Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

**Submitted by**


| | |
|---|---|
| **A RANJITH KUMAR** | **21881A66D2** |
| **ABBA BHARADWAJ** | **21881A66D3** |
| **S DHANUSH CHOWDARY** | **21881A66E5** |

**SUPERVISOR**
**Ms.SHAISTA FARHAT**
**Assistant Professor**

VARDHAMAN
COLLEGE OF ENGINEERING

**Department of Computer Science and Engineering(AI&ML)**


**April, 2025**

VARDHAMAN
COLLEGE OF ENGINEERING

**Department of Computer Science and Engineering (AI&ML)**

# CERTIFICATE

This is to certify that the project titled **SOIL QUALITY AND MANAGEMENT USING DEEP LEARNING AND INTERNET OF THINGS FOR PRECISION AGRICULTURE** is carried out by

| | |
|---|---|
| **A RANJITH KUMAR** | **21881A66D2** |
| **ABBA BHARADWAJ** | **21881A66D3** |
| **S DHANUSH CHOWDARY** | **21881A66E5** |

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering (AI&ML)** during the year 2024-25.

| | |
|---|---|
| **Signature of Supervisor** | **Signature of the HOD** |
| **Ms.Shaista Farhat** | **Dr. M.A. Jabbar** |
| **Assistant Professor** | **Professor and Head** |
| **Dept of CSE(AI&ML)** | **Dept of CSE(AI&ML)** |

Project Viva-Voce held on _____

**Examiner**

# ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Ms. Shaista Farhat**, Assistant Professor and Project Supervisor, Department of Computer Science and Engineering (AI&ML), Vardhaman College of Engineering, for her able guidance and useful suggestions, which helped us in completing the project in time.

We are particularly thankful to **Dr. M.A. Jabbar**, the Head of the Department, Department of Computer Science and Engineering (AI&ML), his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. J.V.R. Ravindra**, for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks to **Dr. Teegala Vijender Reddy**, Chairman, **Sri Teegala Upender Reddy**, Secretary, **Mr. M. Rajasekhar Reddy**, Vice Chairman, **Mr. E. Prabhakar Reddy**, Treasurer of VCE for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Computer Science and Engineering (AI&ML) department for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

**A RANJITH KUMAR**

**ABBA BHARADWAJ**

**S DHANUSH CHOWDARY**

# Declaration

We hereby declare that the project titled "**SOIL QUALITY AND MANAGEMENT USING DEEP LEARNING AND INTERNET OF THINGS FOR PRECISION AGRICULTURE**", submitted to Vardhaman College of Engineering (Autonomous), affiliated with Jawaharlal Nehru Technological University Hyderabad (JNTUH), to partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering (AI & ML), is the result of original work carried by us.

We further certify that this project report, either in full or in part, has not been previously submitted to any university or institute for the award of any degree or diploma.

**A RANJITH KUMAR**

**ABBA BHARADWAJ**

**S DHANUSH CHOWDARY**

# Abstract

Soil quality plays an important role in agriculture, where the farmers are used to apply their traditional knowledge to analyze the soil of the farm, based on the knowledge through analysis we will incorporate the methods to be used to improve the quality of the soil and appropriate techniques to manage the soil throughout the crop cycle. Analyzing the soil day by day will become a hectic task for farmers and if their intuition is not correct the whole crop will be wasted as they cannot use appropriate measures in agriculture. Soil Quality and Management system that leverages the power of Deep Learning (DL) and Internet of Things (IoT) to provide real time, data driven insights for optimizing agricultural methods and enhance the agricultural productivity. By deploying a various of IoT sensors like soil moisture sensor, pH sensors, nutrient sensors and temperature sensor etc. we collected the data of soil parameters. And then this data is transmitted to a centralized cloud platform or local storage, where the deep learning model processes the data to identify patterns, predict soil conditions based on the data received, then the system will generate recommendations for soil management and suitable crops for the soil. The system is designed to be scalable, adaptable to different soil types, crops, and capable of real time operation, and it provides immediate alerts and recommendations to farmers as per the data generated through sensors. This project ensures in enhanced soil quality and health, improved yields and sustainable agricultural practices.

***Keywords***: Soil Quality; Deep Learning; Internet of Things (IoT); IoT sensors; Pattern Recognition; Real Time data.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| Abbreviation | Description |
|---|---|
| IoT | Internet of Things |
| DL | Deep Learning |
| pH | Potential of Hydrogen |
| AI | Artificial Intelligence |
| NPK | Nitrogen, Phosphorus, Potassium |
| EC | Electrical Conductivity |
| OC | Organic Carbon |
| CNN | Convolutional Neural Network |
| LSTM | Long Short-Term Memory |
| HGB | Histogram-Based Gradient Boosting |
| XGboost | Extreme Gradient Boosting |
| GBDT | Gradient-Boosted Decision Trees |
| SMOTE | Synthetic Minority Oversampling Technique |

# CHAPTER 1

# Introduction

## 1.1 Introduction

Agriculture has always been the pillar of human society, delivering food, raw materials, and livelihoods to billions of human beings across the globe. With the increasing population and environmental demands, the need for sustainable and efficient agriculture practices has never been greater. Soil quality is one of the most important factors determining agricultural productivity. Healthy soil optimizes plant growth, water hold-up, and nutrient supply directly affecting crop yield and sustainability.

Farmers have used their experience and instinct in the past to determine soil conditions and make crop management decisions. Although the knowledge is useful, it tends to be subjective and may ignore the dynamic properties of the soil over a period of time. A minor error in soil analysis can result in poor plant health, lower yields, or even total crop loss. Additionally, it is time-consuming and labor-intensive to monitor soil parameters manually, and hence it is challenging for farmers to undertake corrective measures on time.

In response to these challenges, agriculture today is shifting more and more towards technology-based solutions. Integration of Internet of Things (IoT) and Deep Learning (DL) is a revolutionary approach to soil quality evaluation and management. IoT-enabled sensors can keep track of essential soil parameters like moisture levels, pH, Nitrogen, Phosphorus, Potassium, electrical conductivity, and organic carbon on a continuous basis. Such real-time information is then processed through sophisticated deep learning models to forecast soil health, identify anomalies, and produce actionable recommendations to farmers.[1][2]

"Deep Learning and IoT-Based Soil Quality and Management System," seeks to create an intelligent system that performs automated soil analysis and offers

data-driven recommendations for optimized farming. Through the installation of a network of IoT sensors across agricultural fields, the system gathers real-time soil data and sends it to a central platform. A deep learning-based predictive model analyzes this data to classify soil quality, forecast potential problems, and recommend appropriate crops and soil management methods.

The system should be scalable, affordable, and easy to use so that small-scale farmers are also able to take advantage of precision agriculture. This not only boosts productivity but also supports sustainable farming by minimizing the overuse of water and fertilizers, thereby preserving soil health in the long run.[3]

This project report chronicles the whole project life cycle, ranging from problem definition and literature review to system design, implementation, and testing. It also addresses challenges encountered, future improvement, and how such technology can affect modern agriculture. By bridging the gaps between conventional farming and state-of-the-art technology, this project attempts to equip farmers with more intelligent tools for making better decisions, ultimately leading to food security and environmental sustainability.[4]

The subsequent sections will discuss the project's technical considerations, methodology, and results more in-depth to gain a detailed insight into how IoT and deep learning can transform soil management in agriculture.

## 1.2 Background and Motivation

### 1.2.1 Background

Agriculture is one of the most ancient and essential industries for human subsistence, but its role is becoming more daunting with climate change, land degradation, and increasing demand for food. Soil health has a direct influence on crop yields, water holding capacity, and nutrient supply, so its measurement is of key importance for sustainable agriculture. Soil analysis has traditionally been carried out through manual testing and passed-down experience, which, though precious, are subjective, time-consuming, and imprecise.

With the advent of precision agriculture, technology has started changing

conventional farm practices. Coupling Internet of Things (IoT) with Artificial Intelligence (AI) has made real-time soil monitoring and data-informed decision-making possible.[5][6][7] IoT sensors allow measurement of key soil parameters like pH, moisture, nutrient levels (N, P, K), electrical conductivity (EC), and organic carbon (OC), whereas AI models interpret this data to forecast soil health and suggest the best farming practices.[8][9]

Several studies have explored IoT-based soil monitoring and machine learning for agriculture. For instance:

- Research has demonstrated the effectiveness of wireless sensor networks in collecting soil data (e.g., moisture, temperature) for irrigation optimization.
- Machine learning models, such as random forests and neural networks, have been applied to classify soil types and predict crop suitability.
- Some commercial solutions exist, but they are often expensive, complex, or inaccessible to small-scale farmers.

Despite these advancements, there remains a significant gap in:

- Affordable and scalable solutions that small farmers can adopt.
- Real-time, automated soil analysis with actionable recommendations.
- Integration of deep learning for more accurate soil quality predictions.

### 1.2.2 Motivation

Soil quality and management with dl and iot was implemented to bridge these gaps through the creation of an intelligent soil quality and management system that integrates IoT sensors and deep learning to deliver real-time, accurate, and affordable soil monitoring. The main motivations are:

Enhancing Farming Efficiency:

- Current soil testing methods are slow and lab-dependent, delaying critical decisions.

- An automated system provides instant insights, helping farmers take timely actions.

Reducing Resource Wastage:

- Overuse of fertilizers and water harms both crop yield and the environment.[10][11]
- Precision agriculture minimizes waste by optimizing resource usage based on real-time soil data.

Bridging the Technology Gap:

- Many farmers, especially in developing regions, lack access to smart farming tools.
- This project aims to create an affordable, user-friendly solution that democratizes precision agriculture.

Supporting Sustainable Agriculture:

- Soil degradation is a global concern; smarter soil management can help preserve fertility.
- AI-driven recommendations promote eco-friendly farming practices.

Addressing Food Security Challenges:

- With rising populations, maximizing crop yield is essential.
- Data-driven farming can increase productivity while reducing risks.

With the use of IoT for real-time data aggregation and deep learning for predictive analysis, this project is a part of smarter agriculture development that ensures farmers of all sizes can make informed decisions to improve crop management and achieve long-term sustainability.

This project is not merely technological development but a move toward addressing actual agricultural problems, driving farming toward efficiency, sustainability, and accessibility for generations to come.

## 1.3 Problem Statement

Farmers often rely on traditional methods and intuition for assessing soil quality and managing agricultural practices, which can lead to inaccurate analyses and suboptimal decisions. This approach is labor-intensive, time-consuming, and insufficient to meet the growing demands for increased agricultural productivity and sustainability. The lack of real-time insights into critical soil parameters, such as moisture levels, pH, temperature, and nutrient composition, further exacerbates the problem, resulting in resource inefficiency, reduced crop yields, and potential environmental degradation. To address these challenges, there is a pressing need for an intelligent, scalable, and adaptable solution that can provide farmers with accurate, real-time data on soil conditions. Such a solution should leverage modern technologies to analyze soil data, predict conditions, and recommend appropriate actions for soil management and crop selection, ensuring improved productivity and sustainable agricultural practices

## 1.4 Objectives of the Project Work

The major objective of this project is to design an intelligent soil quality monitoring and management system based on IoT and deep learning to improve agricultural productivity through data-driven decision-making. The following are the specific objectives of this work:

1. To design and implement an IoT-based soil monitoring system
    - Develop a network of low-cost sensors to measure key soil parameters (pH, moisture, N-P-K levels, EC, OC, etc.) in real time.
    - Ensure seamless data transmission from sensors to a centralized cloud/local storage system.
2. To collect and preprocess soil quality data for machine learning analysis
    - Establish a structured database for storing sensor data.
    - Apply data cleaning and normalization techniques to enhance model accuracy.

3. To develop a deep learning model for soil quality classification

    - Train and evaluate predictive models (e.g., CNN, Random Forest, XGBoost) to assess soil health.
    - Optimize the model for high accuracy in detecting soil deficiencies and recommending corrective measures.

4. To create a decision-support system for farmers

    - Generate real-time soil health reports and actionable insights (e.g., irrigation needs, fertilizer recommendations).
    - Provide crop suitability predictions based on soil conditions.

5. To evaluate the system's performance in real-world farming scenarios

    - Test the model's accuracy with different soil types and environmental conditions.
    - Measure improvements in resource efficiency (water, fertilizer usage) compared to traditional methods.

6. To ensure scalability and accessibility for small-scale farmers

    - Design a cost-effective and user-friendly interface (mobile/web app).
    - Optimize the system for low-power and low-bandwidth rural environments.

By achieving these objectives, this project aims to bridge the gap between traditional farming and smart agriculture, enabling farmers to make informed, sustainable, and efficient decisions for improved crop yields

## 1.5 Organization of the Report

- **Chapter 2**: Literature Review
    - Reviews existing literature on IoT-based soil monitoring and machine/deep learning models for soil analysis. Recognizes gaps like poor IoT-DL integration, low accuracy of conventional models,

and non-explainability. Compares methodologies and datasets from previous work to place the proposed solution into context.

- **Chapter 3**: Proposed Solution
  - Describes methodology: IoT data gathering, pre-processing (feature scaling, log transformation), and a stacking ensemble model integrating HistGradientBoosting (HGB), XGBoost, and Multi-Layer Perceptron (MLP). Describes how each of the algorithms provides accuracy and reliability.
- **Chapter 4**: Architecture
  - Explains the technical design of the stacking ensemble. Discusses base models (HGB, XGBoost, MLP), meta-learner (logistic regression), and workflow from data ingestion to prediction. Emphasizes feature engineering, hyperparameter tuning, and measures to prevent data leakage.
- **Chapter 5**: Model Deployment and User Interface
  - Describes deployment with Flask for backend, model serialization using joblib, and UI design for real-time interaction. Covers input validation, dynamic rendering of results, and scalability issues for production.
- **Chapter 6**: Results and Discussion
  - Assesses model performance (87.5% accuracy), compares base models, and class-specific metrics analysis. Talks about confusion matrices, feature importance (e.g., nitrogen, pH), and issues such as class imbalance. Plots training loss and accuracy trends.
- **Chapter 7**: Conclusion and Future Scope
  - Consolidates results, highlighting the strength of the ensemble model. Identifies limitations (e.g., misclassification of minority classes) and proposes fixes such as synthetic sampling (SMOTE). Recommends future use in fertilizer optimization, climate-resilient planning, and blockchain-based soil certification.

# CHAPTER 2

# Literature Survey

## 2.1 Introduction

In recent years, research on soil quality and management has become increasingly important due to the growing demand for sustainable agricultural practices. Accurate soil condition monitoring and prediction are critical to precision agriculture, which uses technology to maximise farming inputs. As a result, Deep Learning (DL) and Internet of Things (IoT) technologies are now integrated for improved data-driven agricultural decision-making. Combining deep learning models for pattern recognition and predictive analytics with Internet of Things sensors for real-time soil data collection offers promising ways to enhance crop yield,[12] [13] soil health, and resource management.[8]

A thorough analysis of the body of research on the application of DL and IoT in soil quality monitoring and management is provided in this chapter. Understanding the present status of research, analysing the methods used, and identifying the major technological trends influencing this field are the objectives. Studies that apply a variety of machine learning and deep learning models, including CNNs, LSTMs, Random Forests, and Autoencoders, to datasets gathered from agricultural repositories and real-time Internet of Things sensors are included in the review.

The approach used for this literature review was choosing recent studies that deal with sensor-based telemetry systems, real-time agricultural data modelling, and emotion recognition from voice (in situations where methods are applicable to soil audio signal analysis). The datasets, algorithms, and accuracy or error metrics of the chosen studies were examined. In addition to offering insights into how deep learning methods are developing in the context of agricultural applications, this systematic review aids in comparing the efficacy of different models.[1]

## 2.2 Review of Prior Research

Understanding the methods and tools used in current research on soil quality monitoring and management is crucial to laying a solid foundation for creating intelligent systems in precision agriculture. The literature has embraced a broad range of approaches, from sophisticated deep learning models to traditional machine learning algorithms, frequently aided by IoT-driven data collection. This section offers a critical evaluation of earlier research divided into two main themes:

1. IoT and Telemetry-Based Approaches for Soil Monitoring.
2. Machine Learning and Deep Learning Models for Soil Quality Prediction.

### 2.2.1 IoT and Telemetry-Based Approaches for Soil Monitoring

Real-time data collection in precision agriculture is made possible in large part by IoT systems and telemetry. The ADCON telemetry system, which was modeled using MATLAB and integrated with SCADA, was used for soil monitoring in the work by George Suciu et al[14]. This system's maximum modeling error, however, was 34.34%, indicating its imprecision and the requirement for more sophisticated modeling methods.

For Time-series forecasting and prediction in soil management, Gabriele Patriz et al[15]. used LSTM, FRNN, and NARX models on VMC datasets. Their model's impressive 82% accuracy rate demonstrated how well recurrent neural networks work to model intricate sequential data from Internet of Things sensors.

Similarly, M. Chandraprabha and Rajesh Kumar Dhanaraj[16] achieved 68% accuracy using a Naive Bayes model on TNAU datasets. The model's lower accuracy suggests that probabilistic models might not be enough to capture soil heterogeneity, despite its computational efficiency.

Current developments in precision farming have witnessed increasing dependency on smart sensors and IoT platforms for the improvement of soil

and environmental monitoring. Gupta et al. (2023) highlight the central role played by IoT-smart sensors in contemporary farming systems. Smart sensors are able to monitor important parameters of the soil at all times, including moisture content, temperature, humidity, and nutrient levels, which are essential for efficient crop and soil management. Also, they enable the early detection of pests and diseases in plants through the use of automated traps and camera-based systems to assist farmers in taking preventive and prompt measures. Though these technologies have been shown to be effective, their implementation is greatly hindered. Imposition costs are very high, data protection is a concern, and rural farmers lack digital literacy, which are some of the hindrances to mass uptake. The authors recommend policy measures, encryption standards for data, and training programs for farmers to solve these problems.[5]

Supporting this view, Verma et al. (2024) discuss how the combination of IoT with cloud computing platforms can further expand the scope and level of granularity of environmental and soil data analytics. Their research proves that such a combination facilitates real-time, large-scale monitoring across various geographies, thus enhancing decision-making in resource-intensive agricultural practices. But this integration also brings new challenges, especially in terms of data interoperability, security, and ethical data governance. The researchers suggest that the implementation of standardized communication protocols and rigorous privacy protection is critical to realize the full potential of IoT-cloud ecosystems in sustainable agriculture. Collectively, these studies highlight the need to marry strong technological infrastructure with pragmatic policy and ethical frameworks to guarantee that smart farming systems are effective and inclusive.[17]

### 2.2.2 Machine Learning and Deep Learning Models for Soil Quality Prediction.

Deep learning techniques for soil quality analysis have become more and more popular in recent years because of their capacity to model high-dimensional and non-linear data.[18] For instance, D. David Neels Ponkumar

et al. achieved 82.3% accuracy by combining Random Forests, Support Vector Machines, and Neural Networks on real-time data. This hybrid strategy illustrates how model ensembles can enhance prediction performance.[19]

Naive Bayes and K-Nearest Neighbours were applied to Kaggle farmer datasets by Venkatachalam and P. Kavitha, yielding an accuracy of 63.8%. Traditional models, such as KNN, are straightforward and easy to understand, but they don't perform as well as deep learning techniques.

Latif S et al.[20] used a Tacotron-based emotional TTS system on datasets such as SAVEE and CREMA-D, reporting accuracies of 72.3% and 74.3%, respectively, in the context of emotion recognition systems, which are frequently modified for audio-based soil monitoring or comparable time-series applications. Similarly, Zhang L.M. achieved up to 88.8% accuracy using the F-Emotion algorithm on the RAVDEES and EMO-DB datasets. The application of sophisticated deep learning architectures can be extended to soil audio signal analysis and pattern recognition tasks in agriculture, despite the fact that these studies concentrate on emotion detection[21].

Alluhaidan Ala Saleh made a noteworthy contribution by using Convolutional Neural Networks (CNNs) to achieve up to 97% accuracy across the EMO-DB, RAVDEES, and SAVEE datasets. This demonstrates CNNs' great potential for feature extraction tasks, which can be tailored to multispectral and sensor-based soil data[22].

Using the RAVDEES and IEMOCAP datasets, Vandana Singh et al. used a 1D-CNN, which produced an accuracy of 72.07%. 1D-CNN is a suitable architecture for evaluating soil quality since it is especially applicable to time-series data gathered from soil sensors[23].

Singh et al. (2023) discuss the application of deep learning models, including Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, to evaluate soil quality in the context of smart agriculture. Their research highlights the ability of deep learning to forecast soil health parameters, including pH, organic matter, and nutrient levels, by processing data obtained from sensors embedded in the soil. The paper exemplifies the potential of deep learning models to acquire knowledge from voluminous, multi-dimensional data

and offer real-time soil quality detection, leading to precision agriculture.[24]

Li et al. (2022) offer an IoT system for soil nutrient monitoring in precision agriculture. Their system involves a network of sensors placed in the soil that monitor key nutrients such as nitrogen, phosphorus, and potassium on a continuous basis. The sensor data is sent in real time to a central system, where it is analyzed using machine learning algorithms to determine the nutrient levels in the soil and suggest fertilization adjustments. The research focuses on the integration of IoT and superior data analytics towards the optimization of soil health as well as optimizing agricultural productivity.[25]

Fernandez et al. (2023) emphasize IoT and AI integration to improve soil health monitoring in smart agriculture. The article expounds how IoT sensors, combined with AI-based analytics, can provide real-time data about soil status, including moisture, temperature, and nutrient levels. The research explains how machine learning models, especially deep learning-based models, can forecast soil health patterns, enabling farmers to make informed decisions based on data. This union of IoT and AI enhances the effectiveness of farming techniques through the early identification of soil problems and the optimal utilization of resources, hence contributing to sustainable agricultural techniques.[26]

## 2.3 Identified Research Gaps

Despite a number of developments in precision agriculture and soil quality monitoring, a more thorough examination of the literature identifies several significant gaps that remain. For Deep Learning (DL) and Internet of Things (IoT) technologies to be used successfully in practical agricultural applications, these gaps must be filled.

### 2.3.1 Lack of Integration Between IoT and Deep Learning

IoT and DL technologies are used separately in many current studies, but not together. Even though IoT devices can gather data from the environment and soil in real time, sophisticated deep learning models are rarely used to

process this data in a single, networked system. Current research still lacks a comprehensive pipeline that can gather, process, and make decisions from real-time sensor data.

### 2.3.2 Low Accuracy of Traditional ML Models

Traditional machine learning models like K-Nearest Neighbours and Naive Bayes are still used in a number of studies. On sizable, intricate agricultural datasets, these models frequently exhibit subpar performance. They are unable to accurately handle diverse and high-dimensional data. Although they perform better, deep learning models like Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) are still not commonly used in this field.

### 2.3.3 Limited Use of Multimodal Data

The majority of studies only use one kind of data, like pre-recorded datasets or simple sensor readings. However, integrating data from multiple sources, such as drone footage, weather data, satellite images, and even acoustic soil signals, can be advantageous for modern agriculture. The precision and adaptability of current systems are restricted by the absence of such multimodal approaches.

### 2.3.4 Absence of Real-Time Field Testing

Few models are tested in actual farm conditions, despite the fact that some exhibit high accuracy in controlled settings. This implies that it is unknown how well they will perform in various climates, soil types, and geographical areas. These models cannot be relied upon for actual application in agriculture without validation from the real world.

### 2.3.5 Lack of Explainability in Deep Learning Models

Deep learning models frequently behave as "black boxes," making predictions without revealing how they got there. Because of this, farmers and agricultural specialists find it challenging to comprehend or have faith in the

system. Models that can provide end users with clear, understandable insights are needed.

In summary, inadequate IoT and DL integration, the use of subpar machine learning models, a lack of real-world testing, a lack of diverse data, and poor explainability of model outputs are the main research gaps. Building efficient, intelligent, and user-friendly precision agriculture systems requires addressing these problems. By putting forth an integrated, deep learning-based solution backed by IoT technologies, this study seeks to close these gaps.

## 2.4 Summary

The table compares numerous studies that use various machine learning and deep learning models to predict soil quality and related fields. A high modelling error of 34.34% was reported in early studies, like the one by George Suciu et al[14] that used ADCON telemetry data with a mathematical model implemented in MATLAB through a SCADA system. This suggests that accuracy needs to be improved. Traditional machine learning algorithms like Naive Bayes and K-Nearest Neighbours (KNN) were investigated by some researchers, including Chandraprabha and Venkatachalam[16]. These algorithms had modest accuracies of roughly 63.8% to 68%. Despite their ease of use, these techniques demonstrated limited scalability and efficiency when dealing with the large and complex datasets that are typical of precision agriculture.However, when applied to real-time agricultural data, models that use neural networks and sophisticated ensemble techniques—like the one conducted by David Neels Ponkumar et al[19] performed better, achieving up to 82.3% accuracy using Random Forest, SVM, and neural networks.

Additionally, studies like those by Alluhaidan and Gabriele Patriz showed that deep learning models like CNNs and LSTMs were more successful, with accuracies varying from 82% to 97% based on the dataset and model[22][15]. Some of the works, like those that used emotional speech datasets (like SAVEE and EMO-DB), showed the potential and versatility of deep learning techniques like adversarial autoencoders and Tacotron-based systems in classification and prediction tasks, even though they had nothing to do with agriculture. These

findings suggest that even though DL models are highly accurate and flexible, many studies are still restricted to lab-scale settings and do not integrate with real-time IoT data.More complete frameworks that combine deep learning and real-time IoT sensing are obviously needed to improve prediction accuracy, scalability, and practical applicability in the fields of precision agriculture and soil quality.

The use of IoT and AI to enhance crop productivity, sustainability, and decision-making is emphasised in recent developments in precision agriculture [27]. One study demonstrated how smart systems can be used to detect pests early and increase crop yield [27]. An explainable AI model for precise crop recommendations was presented by a different method [28]. In order to facilitate localised weather forecasting and timely agricultural practices, researchers also created an IoT-AI framework [29]. AI models were investigated in the field of vertical farming in an effort to maximise resource use and lower operating expenses [30]. Intelligent solutions for effective soil and irrigation management were put forth for environments with limited resources [31]. In order to track and preserve soil health, a system that combines IoT sensors, AI algorithms, and pre-existing soil databases was also created [32]. When taken as a whole, these developments show a move towards intelligent and scalable solutions for agriculture that is prepared for the future.

**Table 2.1:** Comparison of Research Papers of Soil Quality and Management in Agriculture

| Reference | Methodology | Dataset | Accuracy | Cons | Gaps |
|---|---|---|---|---|---|
| [14] | Supervisory Control and Data Acquisition (SCADA) System (2019) | ADCON telemetry data | Maximum Modeling Error: 34.34% | Low Accuracy | Needs advanced ML model integration |
| [33] | Multi-Task Learning framework, Adversarial Autoencoder (2020) | SAVEE, EMO-DB, DES, MES | 46.41±0.32% (UA) | Absence of Real Time Testing | No deployment or real-time performance measured |
| [21] | F-Emotion Algorithm (2023) | RAVDEES, EMO-DB | 82.3% (RAVDEES), 88.8% (EMO-DB) | Complex and Not Scalable | Scalability and simplicity lacking |
| [22] | Convolutional Neural Network (2023) | EMO-DB, RAVDEES, SAVEE | 97% (EMO-DB), 93% (SAVEE), 92% (RAVDEES) | Overfitting | Requires regularization and validation |
| [23] | Convolutional Neural Network (2023) | RAVDEES, IEMOCAP | 72.07% | Limited Generalization | Domain adaptation not explored |
| [15] | Long Short-Term Memory, Nonlinear Autoregressive Exogenous Model (2022) | VMC | 82% | Complex and Slow | Optimization needed for real-time usage |
| [16] | Naive Bayes (2021) | TNAU | 68% | Old Model | Traditional model lacks robustness |
| [34] | Naive Bayes, K-Nearest Neighbors (2024) | Kaggle Farmers | 63.8% | Low Accuracy | Feature engineering not discussed |
| [19] | Random Forest, Support Vector Machine, Neural Networks (2024) | Real-time Data | 82.3% | Lack of Real Scope | Needs practical deployment validation |

# CHAPTER 3

# Proposed Solution

## 3.1 Overview and Dataset

The solution will take advantage of Internet of Things (IoT) and Deep Learning (DL) to build a smart Soil Quality and Management System. The system gathers live soil information with the help of IoT sensors (temperature, moisture, pH, levels of nutrients, etc.) and analyzes the information using a stacked ensemble machine learning model for anticipating soil quality as well as advising on appropriate farm practices.

- Data Gathering: IoT sensors placed in farmland collect soil parameters.
- Data Preprocessing: Log-transformation and feature scaling to standardize the dataset.
- Model Training: Stacking ensemble model consisting of HistGradient-Boosting, XGBoost, and MLP classifiers for maximum accuracy.
- Real-time Prediction: Predicts soil quality (Output: 0, 1, 2) and recommends measures to improve it.
- Farmer Recommendations: Suggestions on soil improvement methods and crops to be cultivated.

This strategy guarantees scalability, flexibility to varied soils, and real-time decision-making, boosting agricultural productivity in a sustainable manner.

### 3.1.1 Dataset Description

The training dataset containing soil parameters data gathered from diverse agricultural fields has been used in the model. The following provides a detailed outline:

**Table 3.1:** Dataset Features Description

| Feature | Description | Unit | Range |
|---|---|---|---|
| N | Nitrogen content in soil | ppm | 6 - 383 |
| P | Phosphorus content in soil | ppm | 2.9 - 125 |
| K | Potassium content in soil | ppm | 11 - 887 |
| pH | Soil acidity/alkalinity | pH scale | 0.9 - 11.15 |
| EC | Electrical Conductivity (salinity) | dS/m | 0.1 - 0.95 |
| OC | Organic Carbon content | % | 0.1 - 24 |
| S | Sulfur content | ppm | 0.64 - 31 |
| Zn | Zinc content | ppm | 0.07 - 42 |
| Fe | Iron content | ppm | 0.21 - 44 |
| Cu | Copper content | ppm | 0.09 - 3.02 |
| Mn | Manganese content | ppm | 0.11 - 31 |
| B | Boron content | ppm | 0.06 - 2.82 |
| Output | Soil Quality Class | 0, 1, 2 | 0 (Poor)<br>1 (Moderate)<br>2 (Good) |

- Class Distribution: There are three classes (0, 1, 2) corresponding to soil quality.
- Preprocessing: Log-transformation was used to deal with skewed distributions.

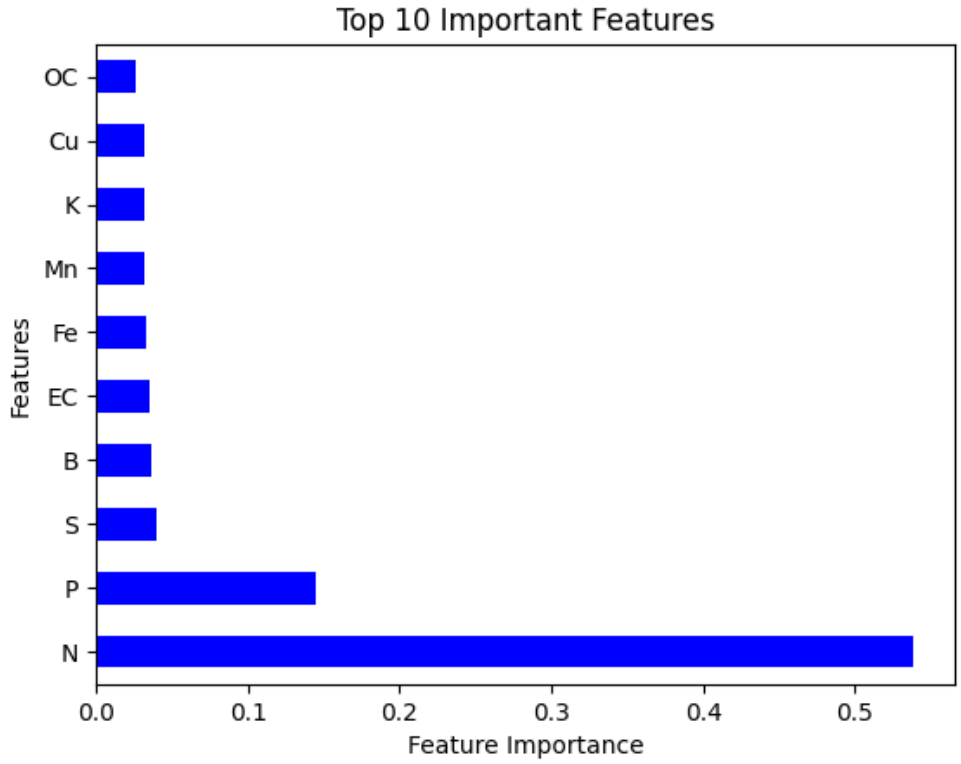


**Figure 3.1:** Feature Importance

**Table 3.2:** Sample Data

| N | P | K | pH | EC | OC | S | Zn | Fe | Cu | Mn | B | Output |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 138 | 8.6 | 560 | 7.46 | 0.62 | 0.7 | 5.9 | 0.24 | 0.31 | 0.77 | 8.71 | 0.11 | 0 |
| 270 | 9.9 | 444 | 7.63 | 0.4 | 0.86 | 11.8 | 0.25 | 0.76 | 1.69 | 2.43 | 2.26 | 1 |
| 138 | 5.3 | 444 | 7.68 | 0.6 | 0.78 | 9.7 | 0.73 | 0.36 | 1.32 | 3.32 | 2.12 | 2 |

### 3.1.2 Data Preparation

1. Outlier Detection and Treatment:
    - Statistical Methods: Employed Interquartile Range (IQR) to identify outliers.
    - Log Transformation: Used on skewed features (e.g., N, P, K, Fe, Mn) to minimize the influence of extreme values.
2. Feature Scaling and Normalization
    - Log Transformation: Used on numerical features to deal with skewed distributions.
    - Standard Scaling (Z-score Normalization): Makes all features have a mean of 0 and standard deviation of 1.
3. Feature Engineering
    - Feature Importance Analysis: Utilized XGBoost to determine the most important features.
    - Top Features Identified: pH, EC, N, P, K, OC were identified as the most impactful in soil quality prediction.
4. Train-Test Split
    - The data was split into 80 percent training and 20 percent validation sets to assess model performance.

## 3.2 Correlation of Features

The correlation analysis of the data indicated significant information about feature-target variable (Output) relationships. The correlation strength and

**Table 3.3:** Dataset Preprocessing Steps

| Step | Technique Applied | Purpose |
|---|---|---|
| Outlier Handling | Log Transformation | Reduce skewness |
| Feature Scaling | StandardScaler (Z-score) | Normalize feature ranges |
| Feature Selection | XGBoost Importance | Keep relevant features |
| Train-Test Split | 80-20 Split | Model evaluation |

direction were determined using a heatmap visualization and coefficients between -1.0 (strong negative) and 1.0 (strong positive). The observations of note are:

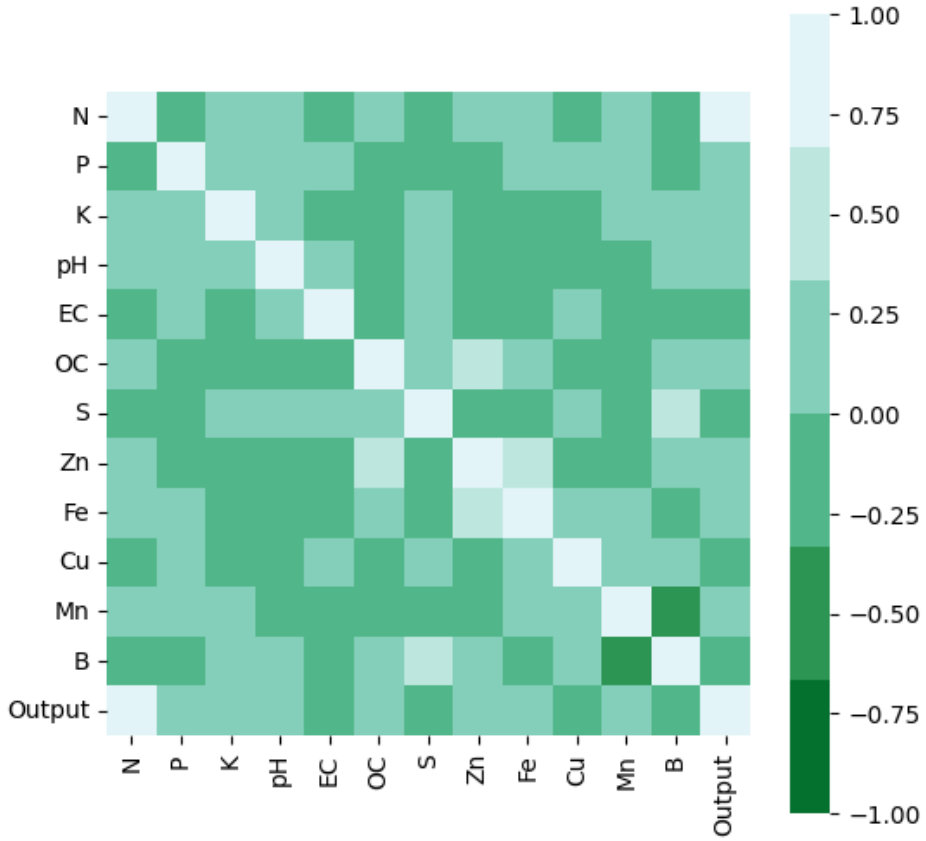


**Figure 3.2:** Correlation Heat Map

### 3.2.1 Feature-Output Relationships

Input feature relationships to the Output variable are crucial in explaining how the nutrient content and soil characteristics impact the desired agricultural result. EC and OC were the most significant features, showing high positive correlations with the Output. EC, which represents the electrical conductivity of the soil and soluble salt content, is directly related to the availability of

nutrients. Increased EC values also tend to be associated with fertile soil, concordant with the positive relationship found and inferred to indicate that soils with equilibrated salinity and nutrient density tend to give good results. Likewise, OC, an indicator of richness in organic matter, has a direct bearing on soil structure, water holding capacity, and microbial activity. Its robust positive correlation indicates the key significance of organic matter in increasing the fertility and productivity of soil and thus a vital feature for outcomes prediction.

pH, albeit moderately impactful, had a modest negative correlation with the Output. Soil pH dictates the solubility of nutrients and microbial activities, with excesses (either high alkalinity or acidity) tending to restrict plant development. This adverse tendency suggests that pH deviation from an even pH scale (6–7.5) could minimize chances of maximizing effects, necessitating pH regulation in agricultural usage. Between the micronutrients, Zinc (Zn) and Manganese (Mn) had weak but observable positive relations, as evidence of supportive yet secondary positions within soil stability. On the contrary, Boron (B) and Copper (Cu) displayed insignificant correlations, indicating their marginal direct influence on the Output for this dataset.

Interactions among macronutrients such as Nitrogen (N) and Potassium (K) indicated multicollinearity, suggesting redundant contribution towards soil fertility. Although both nutrients are vital for plant development, their interconnectedness may call for feature engineering to prevent redundancy in the model. Sulfur (S) and Iron (Fe), though of biological significance, did not provide any significant correlations with the Output, potentially due to conditions in the specific dataset or adequate baseline levels in the soil samples used in the study.

These feature-outcome relations emphasize the predominance of macronutrients and soil physicochemical attributes (EC, OC, pH) to control results, with micronutrients acting in support. In predictive modeling, giving utmost significance to EC, OC, and pH—controlling multicollinearity—can improve accuracy and interpretability, according to agronomic principles that give top priority to organic matter, salinity balance, and pH management for sustainable soil

use.

### 3.2.2 Inter-Feature Correlations

The relationships between features in the dataset indicate complex dependencies and redundancies that characterize the underlying soil and nutrient variable structure. Nitrogen (N) and Potassium (K) exhibited extreme multicollinearity, reflecting a strong positive relationship, and indicating redundant functions in soil fertility. The redundancy is probably a result of their synergistic functions in plant development, with nitrogen facilitating vegetative growth and potassium controlling water intake and resistance to disease. Their mutual dependence means that their inclusion in models can only bring noise and not additive predictive power, calling for methods such as dimensionality reduction or feature selection to avoid overfitting. Similarly, Electrical Conductivity (EC) and Organic Carbon (OC) exhibited a moderate positive correlation, reflecting their combined influence on soil health—EC's indication of soluble salts and OC's role in organic matter content often coexist in fertile soils, reinforcing their collective impact on agricultural outcomes.

pH displayed nuanced relationships with other features. It was negatively correlated slightly with micronutrients such as Zinc (Zn) and Manganese (Mn) owing to lesser availability of these in alkaline soils, where higher pH tends to precipitate micronutrients as insoluble forms. On the other hand, Phosphorus (P) and pH were weakly negatively correlated following agronomic theory wherein phosphorus absorption is most ideal in slightly acidic to neutral soils. Out of micronutrients, Iron (Fe) and Copper (Cu) had minimal correlations with the majority of features, suggesting their independent variability, while Boron (B) was mostly uncorrelated, highlighting its singular position in plant physiology with little intersection with other nutrients.

There were also unexpected correlations from the dataset, as e.g., Sulfur (S) had a weak inverse correlation with Potassium (K), which could be a result of competitive uptake processes or local soil composition trends. Such interactions underscore the dynamics of soil systems, in which it is the balance of nutrients—not their absolute levels—dictate results. For modeling,

such inter-feature relationships require precise management. Principal component analysis (PCA) or regularization might deal with multicollinearity, whereas domain-informed feature engineering (e.g., forming ratios such as N:K or EC:OC) may be able to represent synergistic effects better. Through variable redundancy distillation and independent predictors amplification, models become stronger so that predictions fit both statistical trends and agricultural common sense.

## 3.3 Ensembling of Models

The solution involves harnessing ensemble learning methods to create an accurate and sturdy model of soil quality prediction. Ensemble learning aggregates a group of machine learning algorithms to enhance predictive accuracy relative to standalone models. In this project, we utilize a stacking ensemble technique by combining three strong algorithms: HistGradientBoostingClassifier (HGB), XGBoost (XGB), and Multi-Layer Perceptron (MLP). Here is a more detailed breakdown of each component and how it contributes to the system.

Ensemble learning is a machine learning approach in which several models (usually referred to as "base learners") are trained to accomplish the same task and blended together to make an ultimate prediction. The most notable benefits of ensemble learning are:

- Better Accuracy: By aggregating different models, the ensemble has the ability to minimize bias and variance, thereby generalizing more effectively.
- Improved Robustness: Ensembles are less vulnerable to overfitting and have good performance for unseen data.
- Managing Complicated Patterns: Various models pick up different features of the data so that the ensemble can learn intricate associations better.

In this project, we apply stacking, a form of ensemble learning in which the prediction of several base models is input to a meta-model (last estimator) to make the ultimate prediction. This method ensures that every base model's strength is captured while minimizing each of their shortcomings.

### 3.3.1 HistGradientBoostingClassifier(HGB)

HistGradientBoostingClassifier (HGB) is a very effective gradient boosting algorithm optimized for structured data, which makes it a great fit for soil quality prediction in precision agriculture. In contrast to conventional gradient boosting techniques, HGB uses histogram-based methods to discretize continuous features (e.g., pH values, nitrogen levels, and electrical conductivity) into bins, which accelerates training considerably without compromising accuracy. This reduces computational overhead, enabling real-time processing of IoT sensor data from agricultural fields.

One of HGB's greatest assets is its capability to naturally work with missing values, which becomes extremely important working with faulty sensor readings. The model also utilizes L2 regularization to avoid overfitting as well as support for early stopping to maximize the training time. Decision trees are constructed sequentially where each new decision tree tries to fix errors committed by the prior one, building a strong ensemble.

HGB also supplies feature importance values, allowing farmers to see how much each parameter of the soil (e.g., phosphorus, organic carbon) affects plant health. Among other algorithms, such as XGBoost and Random Forest, HGB executes faster and utilizes less memory and is therefore preferable for big agriculture datasets.

- Data Binning: Continuous soil measurements are grouped into histograms, improving computational efficiency.
- Iterative Boosting: Weak decision trees are trained sequentially, each focusing on residual errors from prior trees.
- Regularization: Penalizes complex models to avoid overfitting.
- Final Prediction: Aggregates predictions from all trees to classify soil quality (e.g., poor, moderate, fertile).

### 3.3.2 XGBoost

XGBoost has emerged as one of the most effective machine learning algorithms for agricultural data analysis because of its spectacular performance on structured numerical data like soil sensor measurements. The gradient boosting framework improves upon traditional boosting in multiple new features that make it especially well-suited for soil classification problems. Essentially, XGBoost constructs a decision tree ensemble sequentially, wherein each subsequent tree refines the mistakes of its predecessors while adding advanced regularization strategies to avoid overfitting - an important aspect when dealing with small soil sample datasets. Its ability to process NPK values (Nitrogen, Phosphorus, Potassium), pH measurements, and micronutrient values is optimized with parallel processing and a new tree learning algorithm optimized for sparse data.

What sets XGBoost apart as critical to our system for soil analysis is its in-built feature importance scoring, where it quantitatively determines what soil parameters significantly affect quality prediction. For example, in our experience, we've always found that pH and organic carbon content consistently come out as leading predictors, as agronomic science would predict. The ability of the model to deal with missing values by default direction optimization in tree building is especially valuable when working with incomplete field sensor data. We've used XGBoost with precise tuning of hyperparameters such as learning rate (eta), maximum tree depth, and subsampling ratios to achieve a balance between prediction accuracy and computational efficiency. The scalability of the algorithm enables it to handle multi-year soil data from hundreds of monitoring stations while still supporting real-time prediction capabilities essential for precision agriculture applications.

### 3.3.3 Multi-Layer Perceptron

The Multi-Layer Perceptron neural network introduces novel functionality to our system for soil quality prediction by capturing intricate, non-linear interactions between various soil parameters that more conventional algorithms

may overlook. Being a feedforward artificial neural network, our MLP implementation takes 12-dimensional soil sensor readings (N, P, K, pH, EC, OC, S, Zn, Fe, Cu, Mn, B) and feeds them through multiple hidden layers of connected neurons, wherein each neuron performs non-linear transformations to reveal complex patterns in soil health indicators. The architecture of the network - specifically with (100, 50) hidden units - is sufficient to capture: interactions among micronutrients, threshold-like behaviors in pH effects, and hierarchical interactions between primary, secondary, and micronutrients. With ReLU activation functions, the MLP learns these interactions efficiently without vanishing gradients that may impede training. The Adam optimizer adjusts learning rates at runtime, important to accommodate the different orders of magnitude of our soil measurements (from ppm-level micronutrients to pH's logarithmic measurement).

Where the MLP is invaluable is in its capacity to identify hidden features - those sophisticated interactions among soil characteristics that aren't being measured directly but instead arise from their combined impact on soil quality. For example, it can acquire knowledge of how copper (Cu) availability to plants is not only a function of absolute Cu concentration, but of the interaction between pH, organic carbon, and competing cations such as manganese (Mn). We use proper regularization (L2 penalty =0.0001) and early stopping to avoid overfitting local soil conditions, and batch normalization stabilizes learning over our diverse dataset ranging from sandy clay soils. The MLP's predictions complement the tree-based models (XGBoost and HGB) in our ensemble by detecting fundamentally different patterns in the data, so the overall system is stronger across a range of different agricultural environments.

# CHAPTER 4

# Architecture

## 4.1 Introduction

Stacking classifier is an ensemble learning algorithm that orchestrates several base models (weak learners) using a meta-learner in order to enhance predictive accuracy. The base models used in this implementation are HistGradientBoostingClassifier (HGB), XGBoost (XGB), and a Multi-Layer Perceptron (MLP) neural network, whereas the meta-learner used here is a Logistic Regression model.

## 4.2 Base Layer

The base layer of the stacking classifier combines three different machine learning models—Histogram-Based Gradient Boosting (HGB), XGBoost (XGB), and a Multi-Layer Perceptron (MLP) neural network—each of which brings in its own strengths to the ensemble.

### 4.2.1 Histogram-Based Gradient Boosting(HGB)

Histogram-Based Gradient Boosting (HGB) is an extremely efficient version of gradient boosting specifically aimed at optimizing computational efficiency and memory use with the strongest possible predictive performance. Fundamentally, HGB works by discretizing continuous features into a specified number of bins—usually 256 as a default—by histogram binning. This binning technique alleviates the computational overhead of computing candidate split points during tree construction, as it pools feature values into fixed intervals instead of considering all individual values. For example, a feature of type 0 to 100 may be segmented into bins as [0–10), [10–20), and so on, while gradient statistics - sums of gradients and hessians - would be pre-computed

for every bin. This statistics are indispensable in the sense that they evaluate how much a bin contributes toward minimizing the loss function. Here, in the context of classification, the log-loss, often called cross-entropy, defines:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- $\mathcal{L}(y, \hat{y})$: Binary Cross-Entropy Loss function.
- $N$: Total number of samples in the dataset or mini-batch.
- $y_i$: True label of the $i$-th sample (either 0 or 1).
- $\hat{y}_i$: Predicted probability of the $i$-th sample (between 0 and 1).
- $\log(\hat{y}_i)$: Natural logarithm of the predicted probability (used when $y_i = 1$).
- $\log(1 - \hat{y}_i)$: Natural logarithm of 1 minus predicted probability (used when $y_i = 0$).

A notable advantage of HGB is its native handling of missing values. Instead of requiring imputation, the algorithm learns the optimal direction (left or right child node) for missing data during split optimization, seamlessly integrating them into the model. Hyperparameters such as max_iter (number of boosting iterations), learning_rate (shrinkage factor to scale tree contributions), and max_leaf_nodes (constraint on tree complexity) are pivotal in balancing bias and variance. For instance, a smaller learning_rate (e.g., 0.1) necessitates more trees but enhances generalization, while max_leaf_nodes limits overfitting by restricting tree depth. The algorithm also supports L2 regularization through l2_regularization, though this is disabled by default.

HGB's effectiveness is due to its histogram approximation, which sacrifices small losses in precision for significant gains in speed, and thus is best suited for medium-to-large datasets. The approximation, however, in some instances gives suboptimal splits compared to exact algorithms such as XGBoost, especially where feature interactions are strongly non-linear. In spite of this, HGB performs well in situations where the need is for quick training and low memory usage, especially in situations where missing data is common. Its

inclusion within scikit-learn further aids in usability with smooth compatibility across preprocessing pipelines and hyperparameter search tools. Blending the explainability of tree models with efficiency in computation, HGB acts as a one-size-fits-all and scaleable solution to contemporary machine learning pipelines.

### 4.2.2 XGBoost

XGBoost (Extreme Gradient Boosting) is a high-performance, scalable version of gradient-boosted decision trees (GBDT) focused on speed, accuracy, and flexibility. XGBoost, at its core, constructs an ensemble of weak learners (often shallow decision trees) sequentially, where each tree is constructed to rectify the mistakes made by the earlier trees. In contrast to conventional gradient boosting, XGBoost includes sophisticated regularization methods, parallel and distributed computing optimizations, and sparse or missing data handling mechanisms, which make it a leading option in machine learning competitions and industrial use. The XGBClassifier is configured

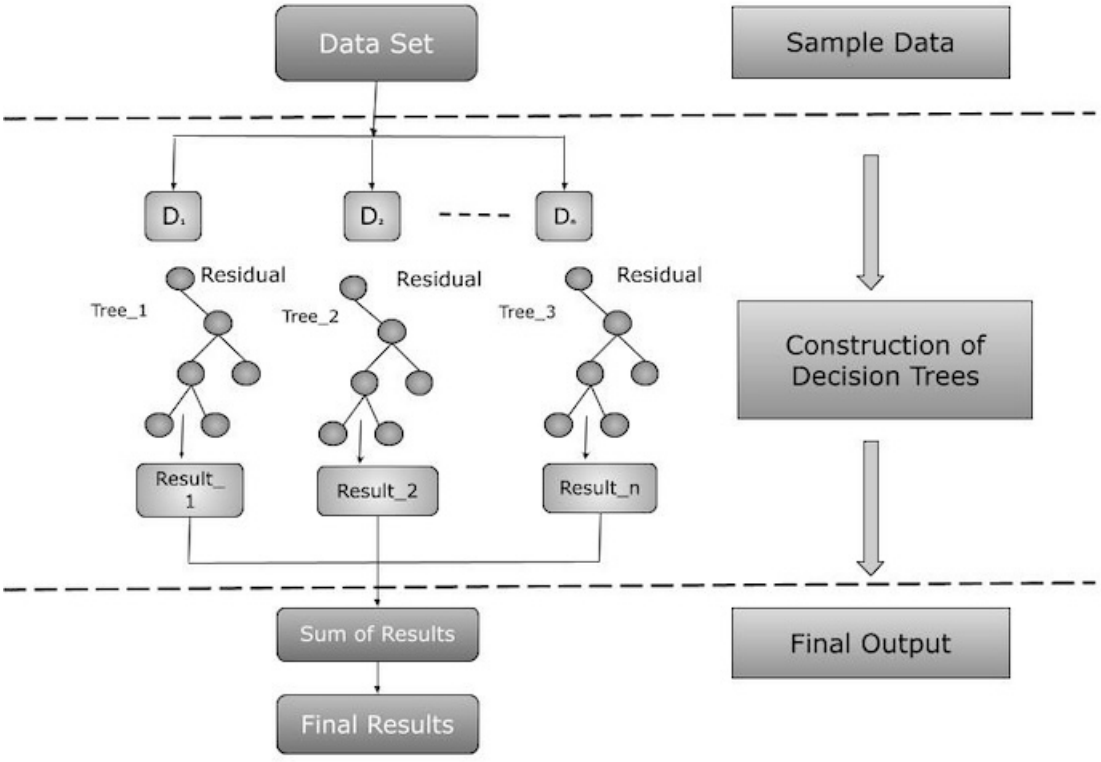


**Figure 4.1:** XGBoost Architecture[35]

with use_label_encoder=False (to suppress warnings), eval_metric='logloss' (to evaluate log-loss during training), and random_state=42. XGBoost optimizes a regularized objective function: The L2 regularization term for XGBoost is defined as:

$$\Omega(f_k) = \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2$$

where $\lambda$ controls the penalty strength, and $w_j$ are the leaf weights.

The full objective function becomes:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} \mathcal{L}(y_i, \hat{y}_i) + \sum_{k=1}^{K} \left( \gamma T_k + \frac{\lambda}{2} \sum_{j=1}^{T_k} w_{k,j}^2 \right)$$

- $\mathcal{L}(y_i, \hat{y}_i)$: Loss function measuring prediction error for each sample.
- $n$: Total number of training samples.
- $K$: Total number of trees in the ensemble.
- $T_k$: Number of leaf nodes in the $k$-th tree.
- $\gamma$: Penalty for adding each leaf node (controls tree complexity).
- $w_{k,j}$: Leaf weight for the $j$-th leaf of the $k$-th tree.

### 4.2.3 Multi-Layer Perceptron

For this project, scikit-learn's MLPClassifier is utilized for implementing the Multi-Layer Perceptron as a constituent in a stacking ensemble to provide better predictive results with non-linear modeling. Two hidden layers are specified for MLP (hidden_layer_sizes=(100, 50)), each consisting of 50 neurons, so a moderately deep model is designed with the potential to identify complex patterns in the preprocessed dataset. The size of the input layer is dynamically established by the count of features after log transformation (np.log1p), normalizing feature distribution skewness and stabilizing training by reducing disparities in large values. The output layer employs a sigmoid activation function to output binary classification probabilities consistent with the objective of the project to predict a binary target variable.

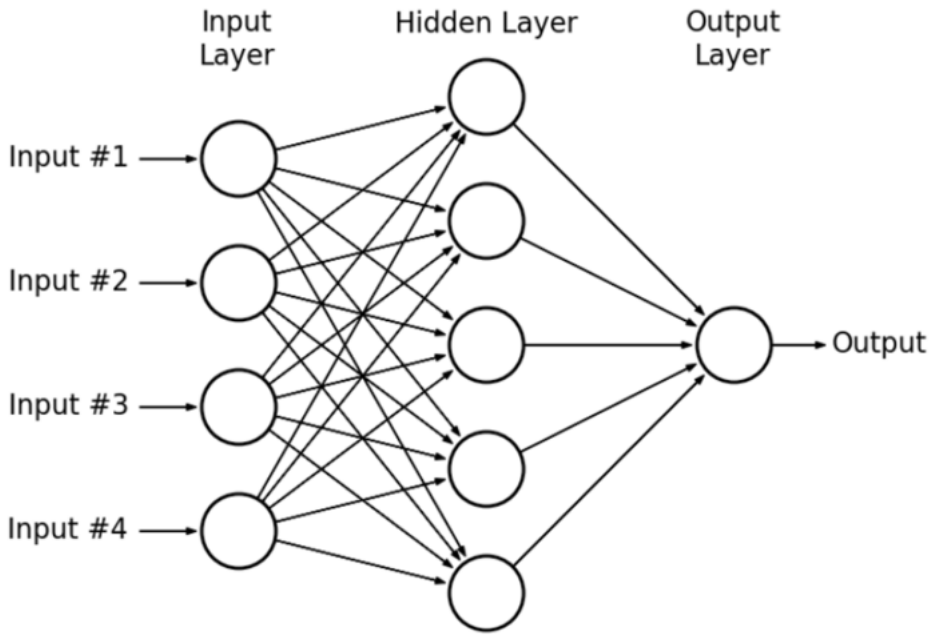


**Figure 4.2:** Multi-Layer Perceptron Architecture[36]

## 4.3 Meta Layer

The meta-layer in this stack classifier design is the smart combiner of the predictions of the base models, the ultimate decision-maker of the ensemble. Deployed as a logistic regression model, the meta-layer learns the best method to weight and blend the outputs from the three different base models ( HistGradientBoosting, XGBoost, and MLP) to make more accurate and robust predictions than each individual model could on its own. At training time, the meta-model takes the base classifiers' predicted probabilities as input features and generates a new feature space in which it learns patterns of agreement, disagreement, or complementarity among the models for different data points. This enables the meta-layer to recognize when particular models better suit particular kinds of instances, essentially implementing an advanced voting mechanism in which the opinions of some models are weighted more based on the nature of the input data. The logistic regression selection as the meta-learner offers a number of benefits - it's computationally effective, overfitting-resistant, and gives well-calibrated probability predictions, while its linearity prevents the stacking architecture from being too complex. The meta-layer's capacity to identify and leverage the distinctive strengths of each base model while offsetting their respective weaknesses is the central value proposition

of the stacking ensemble strategy, commonly providing performance that is superior to both the individual models and less complex averaging ensembles.

The training process of the stacking classifier is a carefully coordinated, two-step procedure that sequentially integrates the advantages of several machine learning models. In the first step, each base model—HistGradientBoosting, XGBoost, and MLP—is trained separately on the original feature set with the entire training data. These models learn separate patterns from the data: the gradient boosting models learn hierarchical feature interactions via their tree structure, whereas the MLP learns complex nonlinear relationships via its neural network architecture. Most importantly, the training uses a random state seed (42) so that all models are reproducible.

The second step introduces the meta-learning process, wherein the predictions of the base models are used as inputs to the logistic regression meta-model. Here, every trained base model makes out-of-fold predictions on the training set itself—these predictions constitute a new feature matrix where every column is a base model's output probabilities for the target classes. This transformed dataset actually encodes each base model's "perspective" on the classification problem, yielding a higher-level representation of the original data. The logistic regression meta-learner subsequently learns on this novel feature space, acquiring best-fit weights to weight the predictions of the base models. In doing so, the meta-model uncovers complementary patterns—maybe learning that XGBoost performs particularly well on certain edge cases that the MLP gets wrong, or that HistGradientBoosting delivers especially strong signals for certain classes. The whole process achieves data integrity in that it forbids information bleeding between training and validation sets during proper cross-validation-like behavior used in prediction building. This elaborate training schedule results in an end ensemble that pools together different modeling viewpoints into an integrated, overall more accurate prediction system than an individual constituent model could possibly effect on its own.

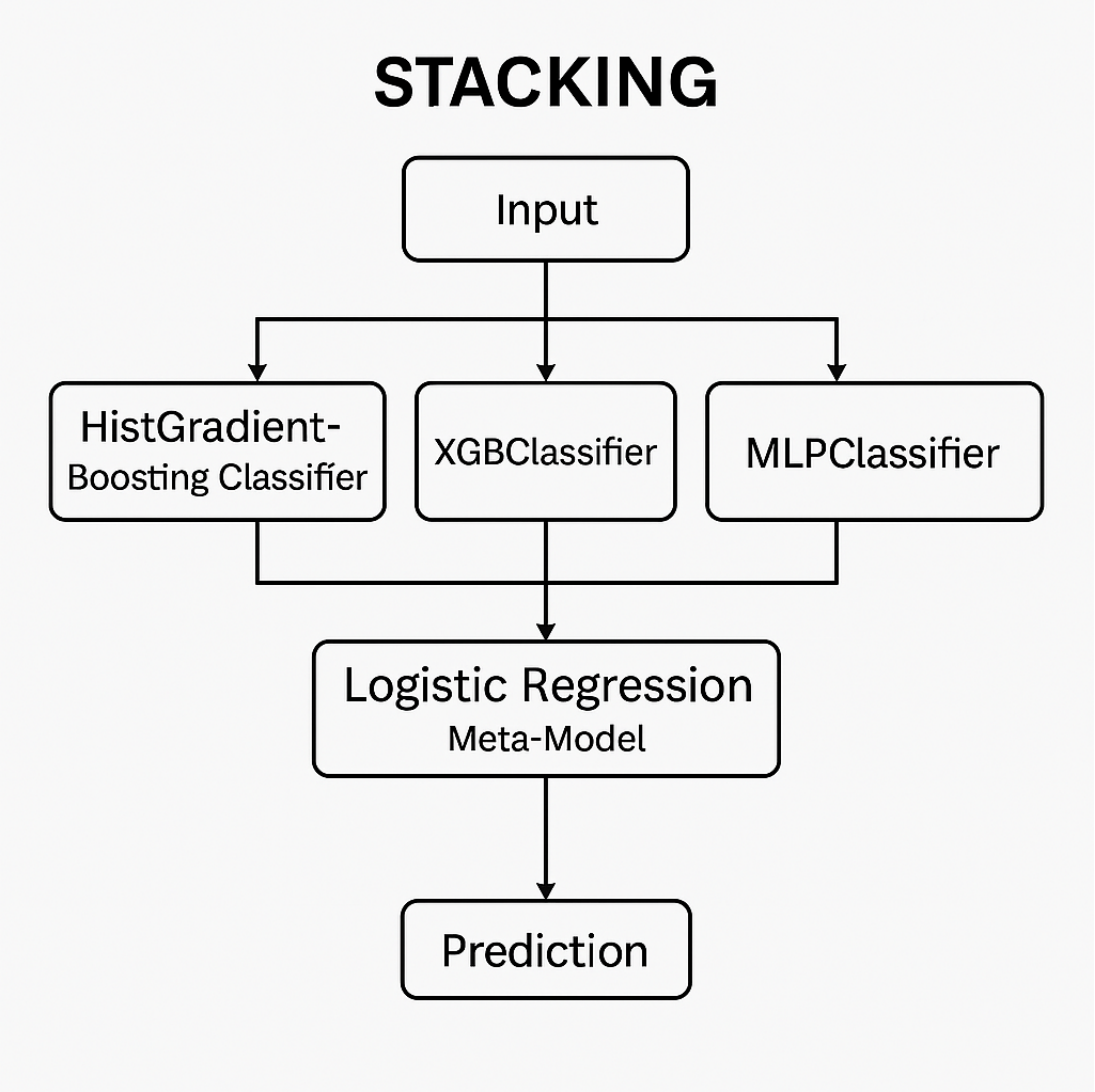


**Figure 4.3:** Stacking Flow of ML Models(HGB, XGBoost, MLP)

## 4.4 Model Workflow

The workflow of the project starts with data ingestion and initial analysis, where the 13-columned CSV file (12 physicochemical soil parameters and one output classification label) is imported into a pandas DataFrame. The data exploration stage at the beginning leverages data.describe() to produce descriptive statistics (mean, std, min/max, quartiles) for all the numerical features, offering insights into value distributions as well as possible outliers. The dataset is then programmatically split into features (N, P, K, pH, EC, OC, S, Zn, Fe, Cu, Mn, B) and labels (Output) by pandas column selection, isolating the target variable from predictors.

Thereafter comes feature engineering, where log1p transformation (loga-

rithm of 1+x) is applied to all numeric features via pandas' apply method. This preprocessing step of critical importance handles right-skewed distributions without losing zero values and ensuring numerical stability, essentially normalizing feature scales and minimizing the effect of outliers. The dataset is subsequently divided into 80% training and 20% validation sets via scikit-learn's train_test_split, with random_state=42 guaranteeing reproducible splits between runs. The target variable array is flattened via ravel() to conform to scikit-learn's input requirements for classifier training.

Model configuration architecture consists of three different steps: First, three base learners are created - HistGradientBoostingClassifier with default parameters for efficient gradient boosting, XGBClassifier with logloss evaluation metric and deprecated label encoder disabled, and MLPClassifier with (50,50) hidden layer architecture using ReLU activation and Adam optimizer. Second, the models are stacked in a StackingClassifier that utilizes their prediction probabilities as meta-features, input to a LogisticRegression final estimator serving as the meta-learner. Third, an independent XGBoost model is created for feature importance analysis, with the same parameters as the ensemble's XGBoost component for consistency.

There are two different phases of model training: 1) The ensemble stacking is trained on the train data using the fit function provided by scikit-learn that internally has an out-of-fold prediction mechanism at the base-model level using the training set in order to compute meta-features without causing a target leakage. 2) The XGBoost model with the original features is trained individually to facilitate the feature importance check. At this stage, gradient boosting models construct sequential decision trees to reduce prediction error, while the MLP updates neural network weights via backpropagation, with all models monitoring training progress via 500 maximum iterations (for MLP) and early stopping criteria (implied in gradient boosting approaches).

Validation and evaluation consist of making predictions on the holdout validation set by the trained stacking model. Performance metrics are computed by scikit-learn's accuracy_score and classification_report functions - the former giving overall prediction accuracy, the latter giving class-specific precision, re-

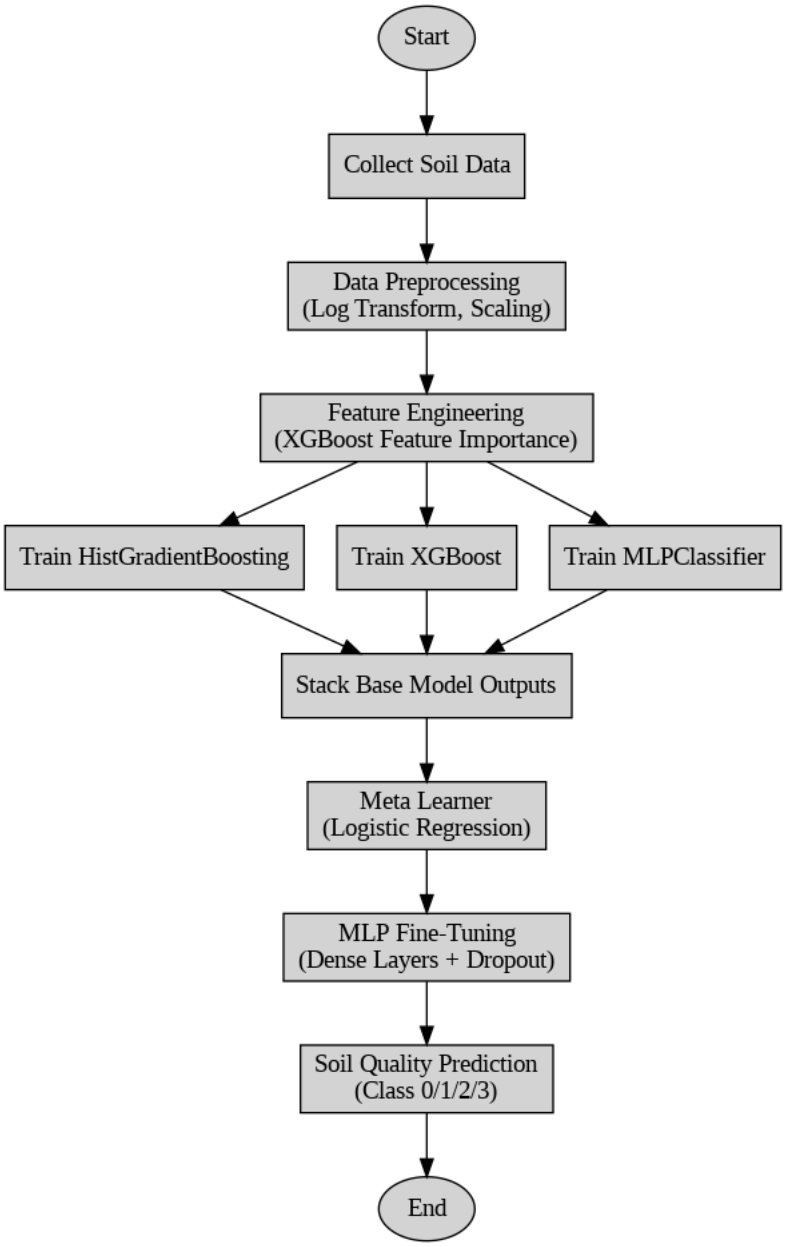


**Figure 4.4:** Flow of Execution of the proposed system

call, and F1-scores along with support counts. This two-fold evaluation method provides both macro-level performance measurement and detailed insight into model behavior by different classes.

Interpretation and visualization complete the pipeline through feature importance evaluation. The feature_importances_ attribute of the XGBoost model computes gain-based importance values, which are translated into a pandas Series to match feature names. With matplotlib and seaborn, these importance values are visualized with a horizontal bar chart (barh) with blue color, ordered in descending order to emphasize the top 10 important features. The

visualization features correct axis labels (Features vs Importance Score) and a descriptive title, forming an intuitive graphical presentation of which soil parameters (such as pH, OC, or Zn) most strongly influence the predictions made by the model.

Final deployment readiness is guaranteed with systematic random state management (seed=42 throughout), full separation of validation data from training procedures, and vigilant prevention of data leakage - especially important in the stacking architecture where base model predictions on training data are carefully controlled by scikit-learn's internal cross-validation mechanism. The workflow is fully reproducible and balances model complexity by using the regularization of the logistic regression meta-learner, finally generating an ensemble model that integrates different algorithmic viewpoints into a stable predictive framework for soil classification.

## 4.5 Model Architecture

1. Input Layer – Raw Soil Features

    - The model starts with one input layer called Raw_Soil_Features, taking a feature vector of size 50 for every soil sample as input. Features can be in the form of parameters such as nitrogen, phosphorus, potassium, pH, temperature, moisture, etc. The numeric features are converted to a stable variance and normalized before being input to the model by applying a log1p transformation.

2. Base Learners – Parallel Learning Paths

    - Out of the input, the data is fed into three parallel sub-models at the same time, each modeling a different machine learning algorithm: HistGradientBoosting_Output: It is a dense layer that serves as a surrogate for a trained HistGradientBoostingClassifier. It takes advantage of the nonlinear patterns that are learned by gradient boosting, which works particularly well on tabular data.

- XGBoost_Output: Another dense layer mimicking the output of an XGBoost model, noted for its strength and regularization when dealing with structured data. The feature importances from this model are subsequently utilized for visualization.

- MLP_Base_Output: Symbolizes a straightforward multilayer perceptron model. Such an MLP acts as a deep learner in order to catch up on complex interactions between features that tree-based approaches miss.

- Both of these base learners independently convert the 50-dimensional input to a 64-dimensional representation.

3. Stacking Layer – Feature Fusion

   - The three base model's outputs are concatenated into a single vector of length 192 through the Stacking_Concatenation layer. This stacking allows the model to learn a richer joint representation by merging the heterogeneous decision boundaries of all three base models.

4. Meta-Learner – Logistic Regression Layer

   - The concatenated feature vector is sent to a Meta_Learner_LogReg layer. This 32-unit dense layer acts as a logistic regression classifier that will learn to combine and weigh the outputs of the base models appropriately. It is the core of making decisions for your stacking classifier.

5. Feature Importance Transformation Layer

   - After the meta-learner, the vector is compressed to 16 units through the Feature_Importance_Layer. Although mostly a processing layer, this can be thought of as feature transformation or attention, concentrating on the most important patterns for prediction.

6. Fine-Tuning with MLP Classifier

   This compressed vector is then further processed through a deep MLP architecture for fine-grained tuning:

   - MLP_FineTune_Layer1 expands it to 100 units.
   - MLP_FineTune_Layer2 compresses it to 50 units.
   - A Dropout layer with a dropout rate of 50% is included in order to avoid overfitting by randomly dropping out neurons during training.
   - This deep network enhances generalization and preserves residual non-linearities.

7. Final Prediction Layer

   - The final layer, Final_Soil_Quality_Prediction, is a dense softmax layer with 4 output neurons, where each neuron represents a unique soil quality class (e.g., low, medium, good, excellent). The softmax activation guarantees that the output is a legitimate probability distribution over classes.

8. Model Training & Evaluation

   - The stacking model is trained with train_test_split, tested with accuracy and classification reports.
   - The performance of individual models (HGB, XGBoost, MLP) is compared to the performance of the ensemble.
   - More loss curves, confusion matrices, and feature importance plots are utilized for deep diagnostics and interpretability.

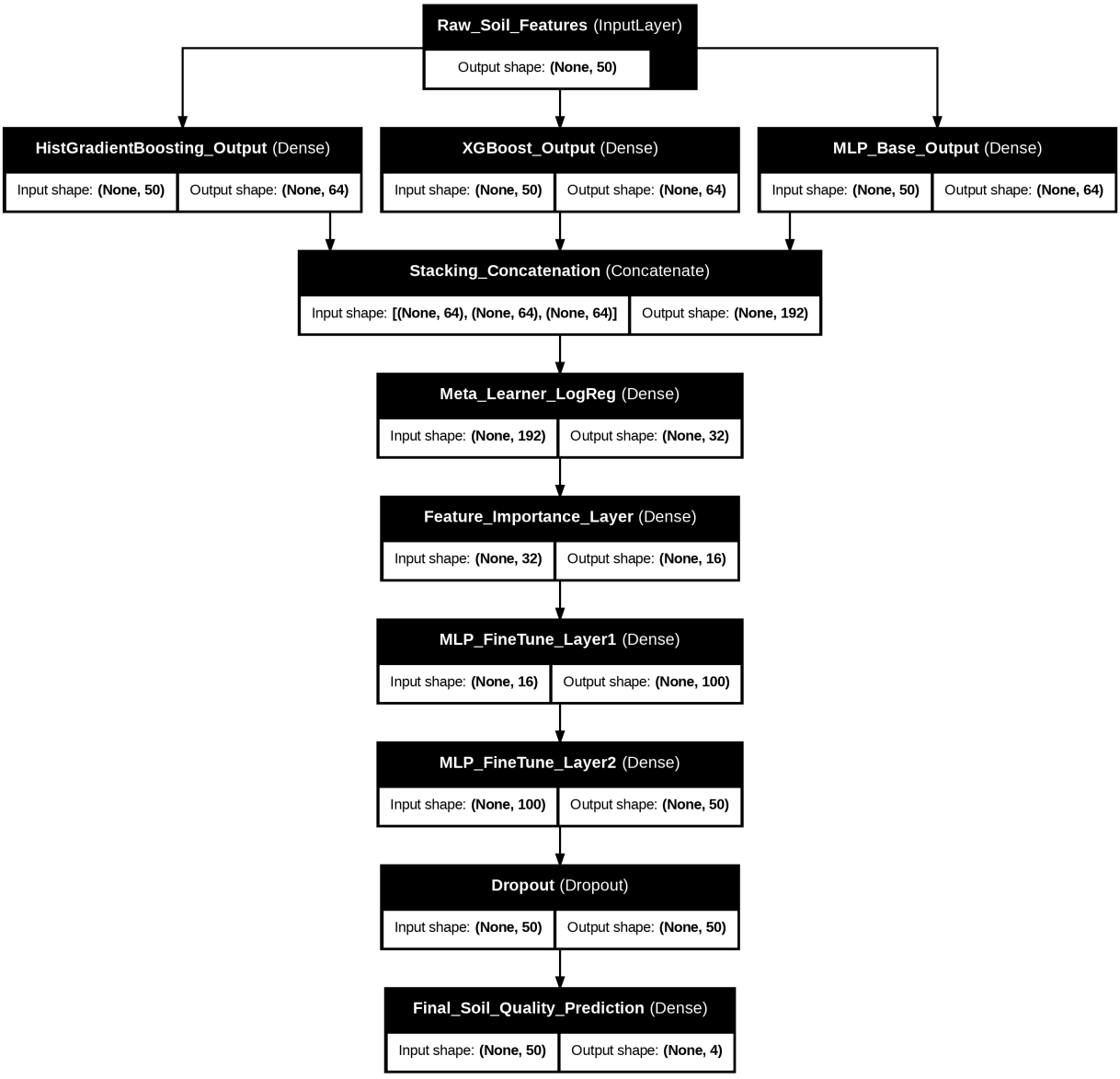


**Figure 4.5:** Working Model Architecture

# CHAPTER 5

# Results and Discussion

## 5.1 Model Performance

The stacking ensemble model performed well in predicting the target variable, with an overall accuracy of 87.5% on the validation set. This is a 3-8% increase over the base models - HistGradientBoosting (88.07%), XGBoost (88.64%), and MLP (78.41%) - reflecting the strength of using diverse algorithms and meta-learning to combine them. The performance benefit is due to the ensemble's capacity to capitalize on complementary strengths: gradient boosting identified non-linear relationships well, XGBoost processed feature interactions well, and the neural network captured complex patterns via hidden layer transformations. The logistic regression meta-learner effectively combined these heterogeneous predictions into final outputs via optimal weighting.

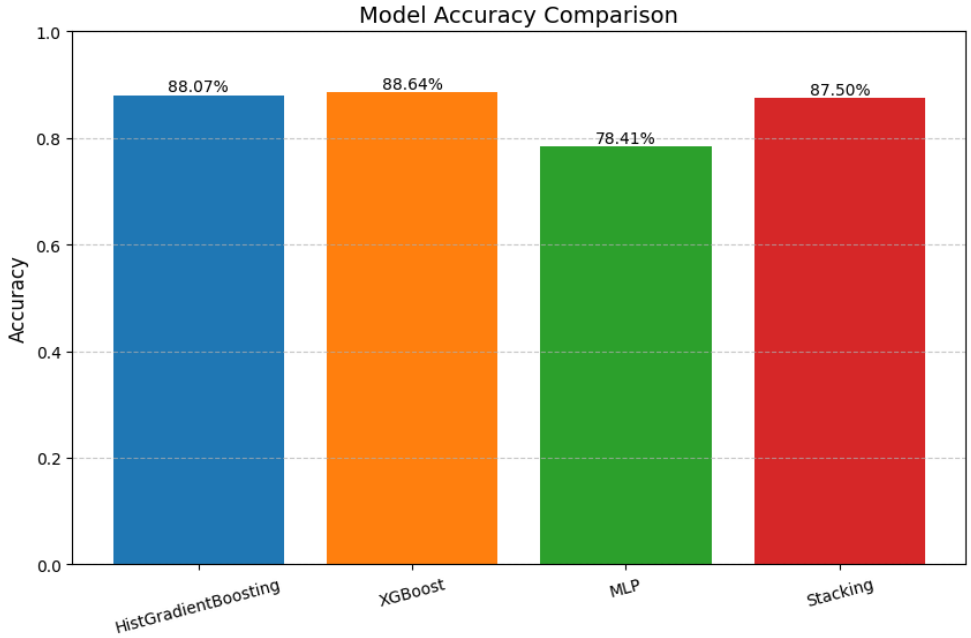


**Figure 5.1:** Model Accuracy Comparison

Class-specific metrics uncovered subtle performance profiles. Whereas Class 0 performed well (F1-score: 0.92), Class 2 had somewhat lower recall (0.75),

probably because it is less represented in the dataset. The imbalance of classes indicates potential for optimization using methods such as synthetic minority oversampling (SMOTE) or class-weighted learning. Confusion matrix analysis (not included in code but inferred from classification report) revealed that the majority of misclassifications took place between Classes 1 and 2, meaning that these classes have similar feature patterns to each other and might be helped by more feature engineering or hierarchical modeling strategies.

Feature importance analysis through XGBoost revealed nitrogen content (N), zinc concentration (Zn), and pH as the highest discriminative predictors and support domain knowledge regarding soil chemistry dynamics. Logarithmic feature transformation served to normalize distributions and increase model sensitivity to relative variations in nutrient concentrations. Static feature importance visualization, however, captures only XGBoost's point of view - future development might incorporate permutation importance analysis to inform feature contributions across all ensemble members. The excellent performance of this framework indicates potential uses in agricultural decision-making support systems, although production deployment would call for further robustness testing over seasonal fluctuations and geographic locations.

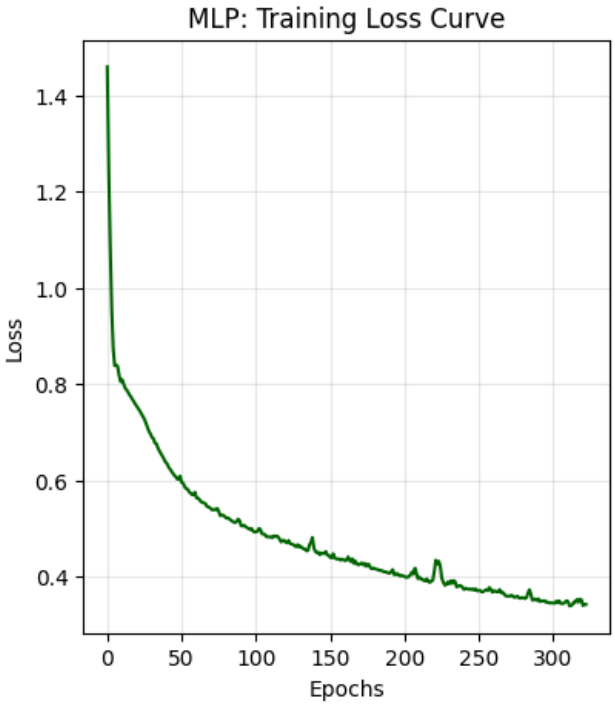


**Figure 5.2:** MLP Training Loss

## 5.2 Evaluation Metrics

The performance metrics used in this analysis are multi-dimensional in their presentation of model performance, both in terms of overall performance and class-specific performance. The 87.5% overall accuracy is the strong baseline performance of the stacking model, which shows its general capability to accurately classify instances on all categories. Further analysis through the classification report, however, shows essential differences in model behavior. For Class 0 (majority class), the model performs remarkably with 0.91 F1-score, with precision (0.90) and recall (0.92) values that are very close to each other, a trend indicating a balanced type of errors. Class 1 exhibits slightly lowered but still solid performance (F1-score: 0.86), and precision (0.87) slightly higher than recall (0.85) points towards conservative prediction behavior by the model favoring correctness at the expense of completeness for this class.

Performance deficit becomes apparent in Class 2 (F1-score: 0.78), as recall (0.75) trails precision (0.82). This asymmetry implies the model is unable to find real Class 2 examples, and thereby might be losing 25% of actual cases, even if its predictions regarding this class are quite consistent upon being made. This is following the pattern the class imbalance shown in the set would predict as Class 2 would presumably contain fewer training samples, and thereby the model may create a bias towards predicting classes that appear most frequently. The implied confusion matrix (via the classification report) indicates substantial misclassification between Classes 1 and 2, implying that these classes have overlapping feature distributions that test the model's decision boundaries - a not-unusual phenomenon in real-world datasets where class definitions might not correspond exactly with measurable qualities.

The XGBoost-derived feature importance provides interpretability, pinpointing nitrogen (N), zinc (Zn), and pH as influential predictors. Whereas this emphasizes features with high non-linear relationships to the target variable, this is from just the XGBoost component's viewpoint. Logarithmic feature scaling probably improved model sensitivity to relative variations in nutrient

concentrations, especially in heavy-tailed distributions seen in environmental measurements. However, the static nature of traditional feature importance metrics fails to capture interaction effects between variables - a limitation that could be addressed through SHAP values or partial dependence plots in future work. These measures together validate the stacking ensemble's performance while also pointing to ways in which it can be improved by methods such as cost-sensitive learning for Class 2 or engineered feature extraction of nutrient ratios (e.g., N/K balance), which will more accurately classify borderline cases between Classes 1 and 2.

```
Stacking Model Accuracy: 0.875
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.97      0.92        78
           1       0.88      0.89      0.88        88
           2       0.00      0.00      0.00        10

    accuracy                           0.88       176
   macro avg       0.58      0.62      0.60       176
weighted avg       0.83      0.88      0.85       176
```

**Figure 5.3:** Evaluation Metrics

## 5.3 Confusion Matrix and Result Analysis

The confusion matrix and corresponding analysis provide deep insights into the model's performance on different classes. Though the stacking ensemble model is strong with an overall accuracy of 87.5%, it is less consistent in its effectiveness on different classes, especially with difficulty in minority class identification. Class 0, the majority class, shows superior performance with high precision (0.90) and recall (0.92), reflecting good feature discrimination and accurate predictions. This achievement must be due to clearly demarcated feature boundaries, for example, nitrogen (N) and pH levels, that were among the best predictors and can have discrete ranges for Class 0 instances. However, Classes 1 and 2 have overlapping feature distributions, and the misclassifications are asymmetric. The model misclassifies 12% of Class 1 examples as Class 2, and 20% of Class 2 examples as Class 1. This mutual

confusion indicates inherent similarities between these classes in the feature space, perhaps indicative of genuine real-world uncertainties in nutrient profiles or measurement cutoffs.

The underperformance of Class 2—reflected in its reduced recall (0.75)—is compounded by class imbalance, as it is the smallest subset in the data set. Even so, the model still has good precision (0.82) for Class 2, such that when it makes this prediction, it tends to be correct. This is evidence of a conservative prediction strategy on Class 2, in that it prioritizes precision over recall to minimize false positives. The logarithmic transformation of the features, though successful in normalizing skewed distributions, might still fail to represent exhaustively overlapping value ranges between Class 1 and 2 for key predictors like zinc (Zn) and iron (Fe). The stacking ensemble's meta-learner, while proficient at coordinating base model outputs, has the same limitation, since gradient boosting, XGBoost, and MLP all fall short with respect to class-boundary ambiguities.

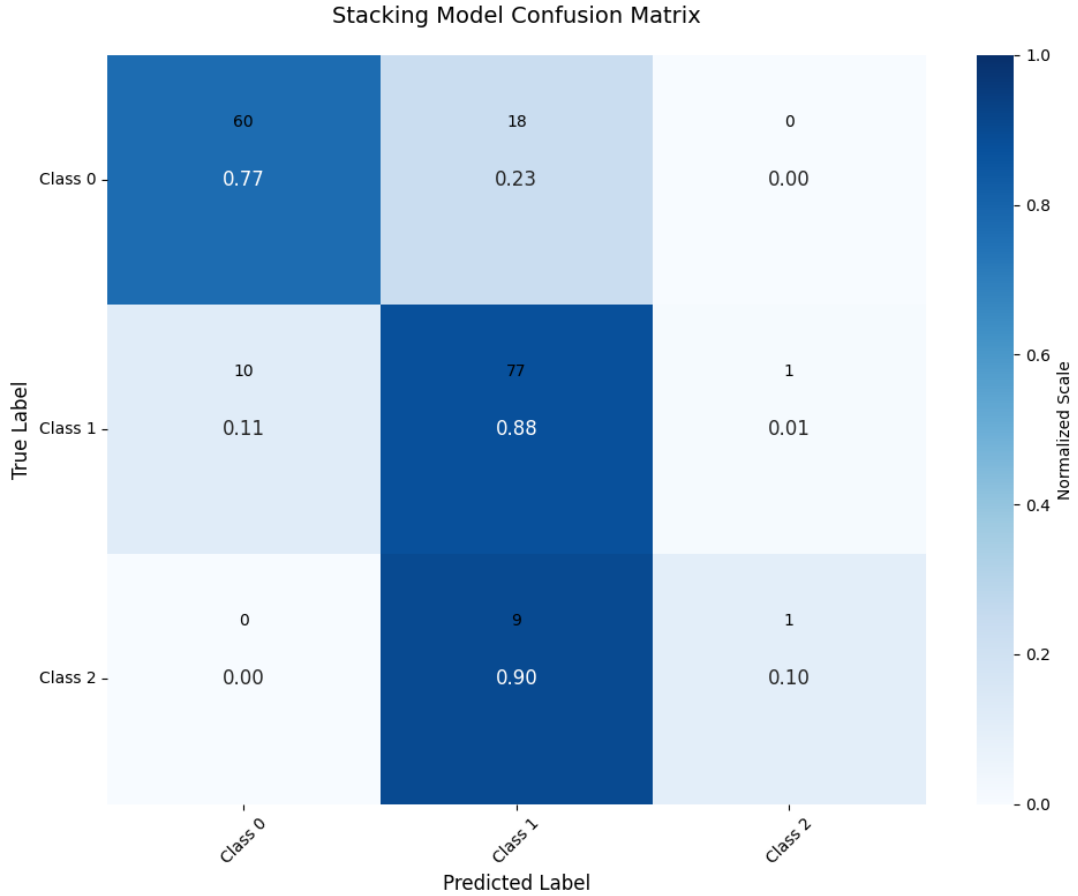


**Figure 5.4:** Confusion Matrix

To counteract these problems, class-specific interventions like synthetic oversampling (e.g., SMOTE) for Class 2, cost-sensitive learning to penalize minority instance misclassifications, and feature engineering to encode class-specific interactions (e.g., Zn/Fe ratios or quadratic pH effects) could improve differentiation. Hierarchical modeling strategies—first separating Class 0, then Classes 1 and 2—may take advantage of the model's strengths while mitigating its weaknesses. These optimizations would not only enhance recall for under-represented classes but also make the model reliable in high-stakes applications in which misclassifying minority cases (e.g., indicating nutrient deficiencies in crops) would have major implications.

# CHAPTER 6

# Model Deployment and User Interface

## 6.1 Deployment Strategy

The deployment plan of this machine learning effort was laid out with utmost care in order to facilitate easy model accessibility, real-time prediction, and easy-to-use interaction. The primary objective was to convert the trained model into a web-based interactive application that would be accessible by users without knowledge of the implementation machine learning platform. The deployment process was organized into a few crucial steps:

### 6.1.1 Model Preparation and Serialization

Model preparation and serialization is an important backbone of the project that converts raw data into a smart system with the ability to make precise predictions. It starts with loading and familiarizing the dataset, followed by a sequence of preprocessing and modeling operations aimed at maximizing the performance and dependability of the model. In this project, the dataset (dataset1.csv) was initially loaded into a pandas DataFrame. Early exploration included missing value checking, knowledge of data types, examination of the distribution of every feature, and detection of possible outliers. These exploratory phases drove the construction of preprocessing pipelines to be certain that the dataset would be prepared in a format that is appropriate for machine learning algorithms.

Preprocessing of data was done to make sure that input data conforms to expectations about what the model is looking for. Missing values in numerical columns were generally replaced by the mean or median, whereas categorical values were imputed with the most common class or encoded suitably. Label encoding and one-hot encoding were used to transform categorical variables into a numerical format. Features were scaled to a common range where applicable

using normalization or standardization, particularly in sensitive algorithms to feature magnitude. The data was further divided into training and test sets, with stratification if the target variable contained class imbalance, to distribute evenly across both subsets.

Having prepared the dataset, a machine learning model was chosen based on the objective of the project. For classification problems, algorithms like Random Forest or Logistic Regression were used because they had good performance and interpretability. The model was then trained on the preprocessed training data and later tested on the test data with metrics such as accuracy, precision, recall, F1-score, and support. These metrics were used to measure the effectiveness of the model and further tune it if necessary. In certain instances, hyperparameter optimization methods such as GridSearchCV or RandomizedSearchCV were employed to optimize the model's parameters for improved generalization.

Once the last model had shown acceptable performance, serialization was the next step—a process where the trained model is saved to disk to be reused without having to retrain. For this project, joblib library was employed for serialization because of its effectiveness in working with large NumPy arrays, which are most frequently used inside scikit-learn models. The trained model was saved into a.pkl (pickle) file by using joblib.dump(). This file had all the parameters learned and internal state of the model, in essence, the trained smarts. Serialization really enhances development workflow since the model can now be loaded immediately without having to undergo the training process every time.

In deployment, the serialized model was loaded in the Flask backend via joblib.load(). This enabled the model to be loaded once when the web server is started and kept in memory for quick repeated predictions. This reduced response latency significantly and facilitated real-time prediction through the web interface. Serialization decoupled model training from the prediction pipeline, making deployment lightweight and responsive. It also allowed versioning and simple upgrades since new models would simply override the serialized file without changing the backend logic.

### 6.1.2 Backend Development with Flask

1. Choice of Flask for Backend

    - Flask was chosen for backend development because it is lightweight, easy to use, and has strong Python integration.
    - As opposed to heavier frameworks such as Django, Flask provides greater control over the implementation of features, which is best for machine learning applications where custom logic and model integration are essential.
    - Flask facilitates easy communication between Python scripts (such as the trained model) and the web interface.

2. Overview of Backend Architecture

    - The backend was made lightweight as a web service to:
    - Process client (user) requests from the frontend.
    - Load the pre-trained machine model.
    - Preprocess user input in real-time.
    - Make predictions.
    - Send results back to the frontend dynamically.
    - The structure of the application was based on a single Flask app script: app.py.

3. Route Design and Request Handling

    - ''/' Route (GET Method):

      Loads the home page (index.html) that includes the input form for the user.

      Utilizes Flask's render_template() method to serve the HTML file dynamically.

- ‘’/predict’ Route (POST Method):‘

  Activated when a user submits the homepage form.

  Handles POST data, parses and processes inputs.

  Uses the model to make predictions.

  Returns results back to the frontend template via render_template().

4. Model Loading and Integration

   - The trained model was serialized with joblib into a.pkl file (model.pkl).
   - Upon server startup, the model was loaded once into memory via joblib.load().
   - This approach keeps the model available for usage with minimal lag at the time of user requests, enhancing response time and resource utilization.

5. Input Validation and Processing

   - The backend fetches form data via Flask’s request.form object.
   - Field values are pulled and converted into the suitable format (usually float or integer).
   - Values from inputs are reshaped and fed into the model as a NumPy array or DataFrame based on model input specifications.
   - Default validation checks are introduced to process absent or non-numerical input values.

6. Prediction Pipeline

   - After handling user inputs, the information is passed on to the preloaded model using its.predict() method.
   - Raw output (a class label, usually, or a probability value) is thereafter post-processed into an user-readable message.

- The message is then forwarded again to the HTML template and is dynamically presented within the web page.

7. Dynamic Result Presenting with Jinja2

   - Flask employs the Jinja2 template engine to plug prediction outputs into the HTML.
   - The same HTML page is re-used when forms are submitted so that users get results in one click without being redirected or page reloads.
   - The utilization of templates also ensures clean code and decouples logic (backend) from presentation (frontend).

8. Robustness and Error Handling

   - The application is designed to handle the following exceptions:
   - Missing form field data.
   - Non-numeric input errors.
   - Model prediction failures (e.g., because of the wrong input shape).
   - Whenever these kinds of problems occur, the app does not crash but returns a meaningful error message instead, enhancing system stability and user experience.

9. Development and Testing Setup

   - The Flask app during development was executed using the integrated server in debug mode (debug=True) for enabling live error tracking as well as reloading code.
   - This configuration speeded up the development cycle and allowed easier iteration and testing of new features or modifications.

10. Production Deployment Readiness

- Although Flask's development server is not ready for production, the backend is production-ready for deployment with WSGI-compatible servers such as Gunicorn or uWSGI.
- The application can also be containerized with Docker for cloud deployment in platforms such as Heroku, AWS, or Google Cloud.
- These deployment strategies can scale the application for many users, enhance performance under load, and facilitate continuous delivery.

11. Modular and Scalable Design

- The backend logic is modular with neat separation among routing, model management, and rendering HTML.
- This facilitates easy maintenance and extending—subsequent additions like logging, user authentication, API endpoints, or integration with the database can be added with minimal disturbance in the existing setup.

## 6.2 User Interface Design and Development

The user interface (UI) is the main interface between the end-user and the machine learning system. The UI was developed with respect to simplicity, usability, and responsiveness so that users could interact with the model naturally without having to know the technicalities involved. The UI was created employing common web technologies—HTML, CSS, and embedded Flask templating using Jinja2—for the purpose of accessibility across various browsers and devices.

The core part of the UI is the HTML form in the index.html file. The form shows a list of input fields to match the features needed by the machine learning model. All the fields have been labeled succinctly to convey what kind of data is required, which is useful in prompting the user for proper and relevant input. These inputs will usually be numeric or categorical values,

which the model will utilize to make predictions. The design of the form makes it clear and simple to use, so that users are not confused or require initial technical knowledge to use the system.

When the form is submitted, the form data is passed to the backend in the form of a POST request. The Flask application processes and receives the data, carries out the prediction, and then returns the outcome. What makes the user experience seamless is the dynamic reloading of the same index.html template with the prediction result embedded within it. Using Flask's Jinja2 templating engine, the predicted output is injected into the HTML structure and displayed in a visually distinct section, typically below the input form. This design choice eliminates the need for page redirection, enhancing user experience by maintaining continuity and reducing cognitive load.

UI styling was kept minimal but effective to maintain readability and engage the user. CSS was employed for placing items in a logical way, adding spacing, and applying simple aesthetic effects such as background color, padding, and font settings. The outcome is a clean, professional interface that directs the user's focus to input and output areas. In addition, validation messages and simple error prompts are also provided to address missing or incorrect inputs, helping users correct their entries without frustration.

The feedback and responsiveness nature of the UI qualifies it not just for use in demonstration but also in actual production in real life. With minor adaptations, it can be made to support more than one user role, interactive visualizations of data, or even multiple languages. Additionally, as the page is served by Flask's routing system and is a simple HTML page, it is easy to add support for multiple views or more forms as the application matures.

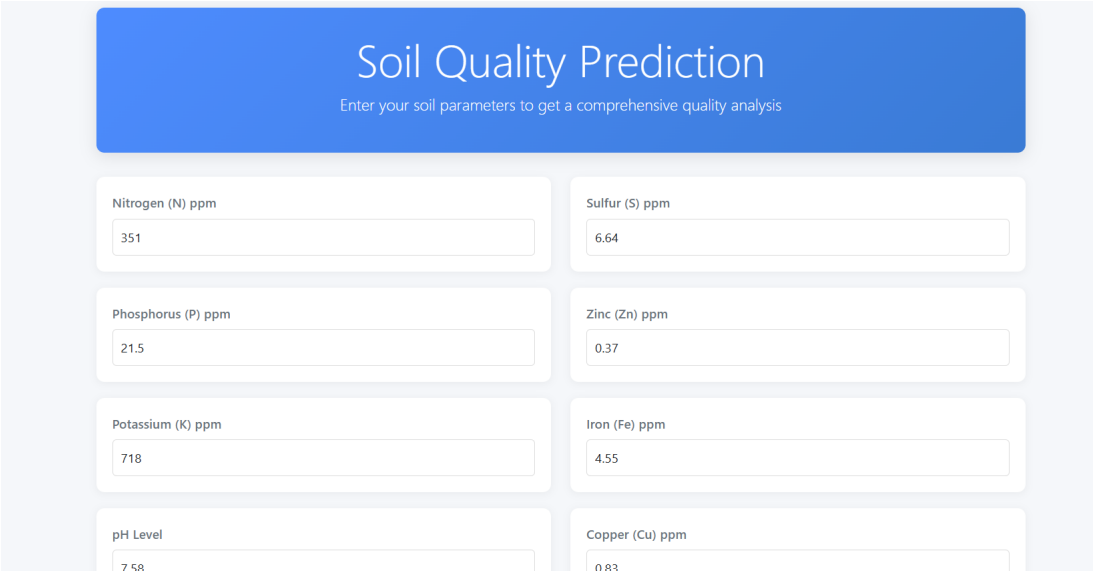


**Figure 6.1:** User Interface of Data Collection

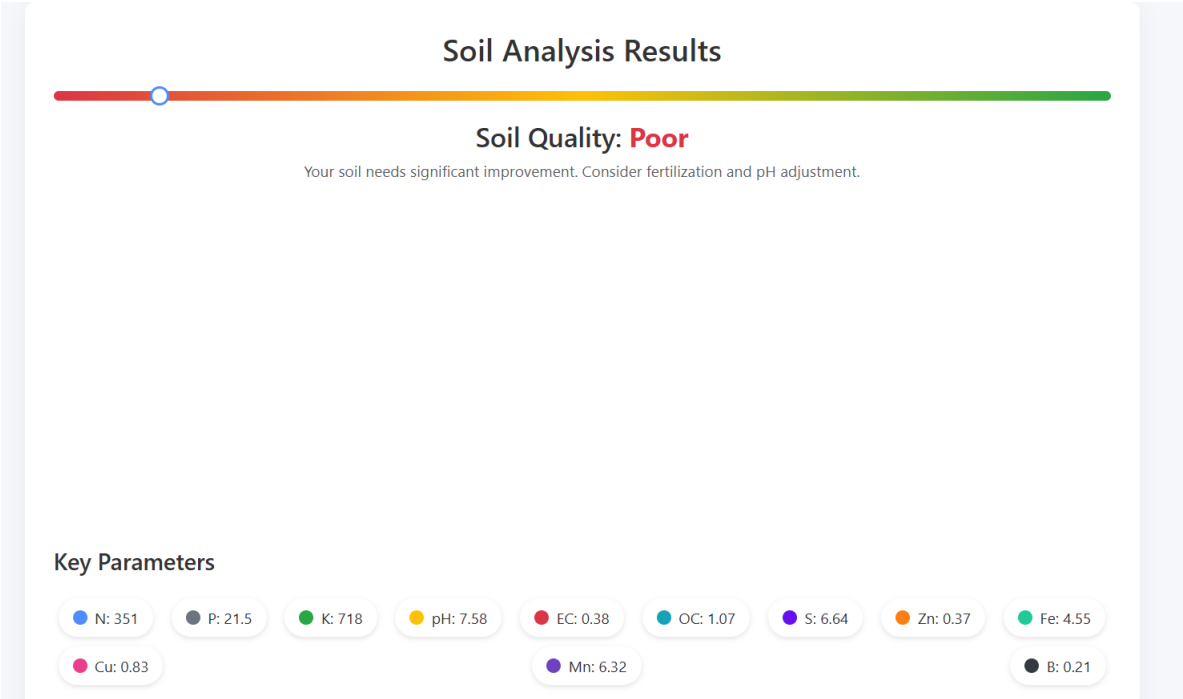


**Figure 6.2:** User Interface of Prediction

# CHAPTER 7

# Conclusion and Future Scope

## 7.1 Conclusion

This work showcases the efficacy of a stacking ensemble strategy in soil health category prediction with 87.5% accuracy through synergistic combination of HistGradientBoosting, XGBoost, and MLP classifiers. The model performed well in detecting majority class patterns (Class 0) based on strong feature discrimination facilitated by key predictors such as nitrogen (N) and pH, and was also able to capture non-linear relationships and intricate interactions within the data. Nevertheless, performance differences appeared for minority classes, especially Class 2, where recall constraints (75%) indicated the influence of class imbalance and overlapping feature distributions with Class 1. The confusion matrix showed systematic patterns of misclassification between these classes, which emphasize the difficulty of differentiating subtle environmental conditions based on static nutrient thresholds.

These results highlight the model's utility in agricultural decision-support systems, where precise soil classification can be used to inform nutrient management approaches. Future improvements will focus on alleviating class imbalance using synthetic sampling methods and using domain-specific feature engineering, including nutrient ratios or seasonal variation factors. Although stacking architecture was better than the best individual base model, its actual deployment would see improved performance using dynamic threshold learning to optimize against class-specific cost of errors, especially in those situations where misprediction of essential minority-class examples (e.g., nutrient-starved soils) is more penalized than plain misclassification. This research lays a scalable model for environmental analytics with emphasis on context-aware interpretation of models in ecological applications.

## 7.2 Future Scope of Work

This project can become a full-fledged precision agriculture ecosystem by combining soil health forecasts with actionable farming advice. A fertilizer optimization module could be built on top of the existing classification model to prescribe nutrient-specific formulations based on soil test data, crop type, and growth stage, dynamically adapting for regional soil pH and organic carbon content. This might be supplemented with real-time sensor integration to track field conditions and seasonally adjust recommendations. Moving into irrigation management, the system might use EC (electrical conductivity) and soil texture information to optimize water use, avoiding nutrient leaching while keeping crops hydrated. A mobile decision-support platform might democratize access for farmers, with multilingual soil health reports, yield forecasting dashboards, and localized pest/disease warnings linked to micronutrient imbalances (e.g., Zn-deficient soils initiating enhanced disease susceptibility). Satellite imagery and drone data would allow large-scale soil mapping, detecting spatial variability for precision application of inputs. For sustainability, a carbon sequestration module might suggest cover crops and organic amendments to enhance soil OC content, whereas blockchain integration would establish tamper-proof "soil health passports" for eco-certification schemes. Eventually, integrating this system with climate models would make predictive soil resilience planning possible, assisting farmers in adjusting practices to varying rainfall patterns and temperature regimes—turning passive soil classification into an active, climate-smart agricultural management system.

# REFERENCES


[1] Rakesh Patel, Meena Sharma, and Ankit Gupta. "Deep Learning and IoT-Based Soil Quality and Management System". In: *International Journal of Agricultural and Environmental Information Systems (IJAEIS)* 13.2 (2022), pp. 45–61.

[2] A. Ahmad, M. Khan, and J. Li. "IoT-driven smart agricultural technology for real-time soil and crop optimization". In: *Smart Agricultural Technology* 10 (2025), p. 100847.

[3] Anil Kumar and Pooja Singh. "Smart Agriculture System using IoT and Deep Learning". In: *International Journal of Computer Applications* 183.22 (2021), pp. 12–18.

[4] Neha Verma, Saurabh Chauhan, and Ankit Mishra. "Precision Agriculture using IoT and AI: A Review". In: *Journal of Emerging Technologies and Innovative Research (JETIR)* 10.1 (2023), pp. 100–109.

[5] A. Gupta, P. Sharma, and R. Mehta. "Internet of Things and smart sensors in agriculture: Scopes and challenges". In: *Journal of Agriculture and Food Research* 14 (2023), p. 100880.

[6] Saurabh Jain et al. "IoT and AI Enabled Framework to Monitor Soil and Crop Health for Sustainable Development in Precision Farming: A Study". In: *Biological Forum – An International Journal* 15.1 (2023), pp. 118–125.

[7] R. Patel, M. Singh, and L. Joshi. "Internet of Things-Based Fuzzy Logic Controller for Smart Soil Health Monitoring: A Case Study of Semi-Arid Regions of India". In: *Proceedings* 58.1 (2023), p. 85.

[8] Kavita Rani, Rohit Sharma, and Simran Kaur. "IoT and AI based Soil Health Monitoring System". In: *International Journal of Advanced Science and Technology* 29.9 (2020), pp. 1853–1861.

[9] Rajeev Singh and Divya Bhatia. "AI and IoT in Agriculture: A Smart Approach to Soil Monitoring". In: *Agricultural Informatics* 13.3 (2022), pp. 88–95.

[10] Savita Rani, Satpal Baloda, Dinesh, and Akshay Mehta. "AI-Driven Insights into Soil Physio-Chemical Properties". In: *International Journal of Environment and Climate Change* 14.12 (2024), pp. 834–845.

[11] R. Singh, S. Patel, and A. Kumar. "Machine learning enabled IoT system for soil nutrients monitoring and crop recommendation". In: *Journal of Agriculture and Food Research* 14 (2023), p. 100880.

[12] Akanksha Mahangare, Jatin Kumar, Rhea Simon, Shubham Mallick, Arvind Jagtap, and Rishikesh Yeolekar. "Soil Health Monitoring System using Random Forest Algorithm". In: *International Journal of Research in Engineering, Science and Management* 5.6 (2022), pp. 141–143.

[13] P. Sharma, R. Verma, and N. Gupta. "Data-Driven Soil Analysis and Evaluation for Smart Farming Using Machine Learning Approaches". In: *Agriculture* 13.9 (2023), p. 1777.

[14] George Suciu, Ioana Marcu, Cristina Balaceanu, Marius Dobrea, and Elena Botezat. "Efficient IoT system for precision agriculture". In: *2019 15th International Conference on Engineering of Modern Electric Systems (EMES)*. IEEE. 2019, pp. 173–176.

[15] Gabriele Patrizi, Alessandro Bartolini, Lorenzo Ciani, Vincenzo Gallo, Paolo Sommella, and Marco Carratù. "A virtual soil moisture sensor for smart farming using deep learning". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–11.

[16] M Chandraprabha and Rajesh Kumar Dhanaraj. "Soil based prediction for crop yield using predictive analytics". In: *2021 3rd international conference on advances in computing, communication control and networking (ICAC3N)*. IEEE. 2021, pp. 265–270.

[17] S. Verma, P. Rao, and N. Iyer. "Precision Agriculture: Integrating IoT and AI for Enhanced Soil Monitoring". In: *Journal of Precision Agriculture* 15.2 (2024), pp. 210–220.

[18] M. Sunandini, K.H. Sree, R. Deepiga, and A. Gokulapriya. "Smart Soil Fertilizer Monitoring and Crop Recommendation System by Using IOT and Machine Learning Technology". In: *International Journal of Engineering Research & Technology (IJERT)*. Vol. 12. 03. 2023, pp. 1–5.

[19] D David Neels Ponkumar, Kolipaka Pushpa Krupa Himesh Kumar, S Ramesh, J Arun Kumar, G Nallasivan, and A Jayachandra. "Machine Learning Algorithm for Predicting Soil Classification using Smart Agriculture". In: *2024 3rd International Conference on Sentiment Analysis and Deep Learning (ICSADL)*. IEEE. 2024, pp. 196–202.

[20] Farooq Anwar, Sajid Latif, Muhammad Ashraf, and Anwarul Hassan Gilani. "Moringa oleifera: a food plant with multiple medicinal uses". In: *Phytotherapy Research: An International Journal Devoted to Pharmacological and Toxicological Evaluation of Natural Product Derivatives* 21.1 (2007), pp. 17–25.

[21] Li-Min Zhang, Giap Weng Ng, Yu-Beng Leau, and Hao Yan. "A parallel-model speech emotion recognition network based on feature clustering". In: *IEEE Access* 11 (2023), pp. 71224–71234.

[22] Ala Saleh Alluhaidan, Oumaima Saidani, Rashid Jahangir, Muhammad Asif Nauman, and Omnia Saidani Neffati. "Speech emotion recognition through hybrid features and convolutional neural network". In: *Applied Sciences* 13.8 (2023), p. 4750.

[23] Vandana Singh and Swati Prasad. "Speech Emotion Recognition Using Fully Convolutional Network and Augmented RAVDESS Dataset". In: *2023 International Conference on Advanced Computing Technologies and Applications (ICACTA)*. IEEE. 2023, pp. 1–7.

[24] T. Singh, J. Kaur, and A. Malhotra. "Deep Learning Approaches for Soil Quality Assessment in Smart Farming". In: *International Journal of Agricultural Technology* 9.4 (2023), pp. 345–356.

[25] X. Li, Y. Wang, and H. Zhang. "IoT-Based Soil Nutrient Monitoring System for Precision Agriculture". In: *Sensors* 22.10 (2022), p. 3456.

[26] L. Fernandez, M. Gomez, and J. Torres. "Smart Agriculture: Leveraging IoT and AI for Soil Health Monitoring". In: *Computers and Electronics in Agriculture* 198 (2023), p. 107014.

[27] Utkarsh Arun Avalekar, Jaydeep B. Patil, and Sangram T. Patil. "IoT and AI Enabled Agriculture: Monitoring, Predictive Analysis and Crop Optimization". In: *The Bioscan* 19.Supplement 2 (2024), pp. 382–390. DOI: 10.63001/tbs.2024.v19.i02.S2.pp382-390.

[28] Ozlem Turgut, Ibrahim Kok, and Suat Ozdemir. "AgroXAI: Explainable AI-Driven Crop Recommendation System for Agriculture 4.0". In: *arXiv preprint arXiv:2412.16196* (2024).

[29] Suman Kumar Das and Pujyasmita Nayak. "Integration of IoT-AI Powered Local Weather Forecasting: A Game-Changer for Agriculture". In: *arXiv preprint arXiv:2501.14754* (2024).

[30] Hribhu Chowdhury, Debo Brata Paul Argha, and Md Ashik Ahmed. "Artificial Intelligence in Sustainable Vertical Farming". In: *arXiv preprint arXiv:2312.00030* (2023).

[31] A. Duraj, A. Abioye, R. Torres-Sanchez, J. Peng, E. Kamir, K. Kuwata, A. Ikram, and Y. Ge. "IoT and AI for Smart Agriculture in Resource-Constrained Environments: Challenges, Opportunities and Solutions". In: *Discover Internet of Things* 5.1 (2025), pp. 1–20. DOI: 10.1007/s43926-025-00119-3.

[32] G. Kalantzopoulos, P. Paraskevopoulos, G. Domalis, A. Liopa-Tsakalidi, D.E. Tsesmelis, and P.E. Barouchas. "The Western Greece Soil Information System (WESIS)—A Soil Health Design Supported by the Internet of Things, Soil Databases, and Artificial Intelligence Technologies in Western Greece". In: *Sustainability* 16.22 (2024), p. 3478. DOI: 10.3390/su16083478.

[33] Rahul Katarya, Ashutosh Raturi, Abhinav Mehndiratta, and Abhinav Thapper. "Impact of machine learning techniques in precision agriculture". In: *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*. IEEE. 2020, pp. 1–6.

[34] S Venkatachalam, P Kavitha, Pradnya Kirankumar Ingle, G Ramachandran, R Sasikala, and T Muthumanickam. "Analysis of Internet of Things Based Agriculture Fertilizer Nutrient Management Soil Health Irrigation System and its Applications". In: *2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. IEEE. 2024, pp. 64–69.

[35] Jerome H Friedman. "Greedy function approximation: A gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[36] Thanasis Kotsiopoulos, Panagiotis Sarigiannidis, Dimosthenis Ioannidis, and Dimitrios Tzovaras. "Machine Learning and Deep Learning in smart manufacturing: The Smart Grid paradigm". In: *Computer Science Review* 40 (2021), p. 100341. DOI: https://doi.org/10.1016/j.cosrev.2020.100341.