

A

**Project Report**

on

**OPTIMIZED CROP AND FERTILIZER RECOMMENDATION  
SYSTEM USING MACHINE LEARNING FOR SUSTAINABLE  
AGRICULTURE**

Submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR,  
ANANTAPURAMU**

in partial fulfillment of the requirements for the award of the Degree of

**BACHELOR OF TECHNOLOGY**

**In**

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

**Submitted by**

<b>N. DIVYA GANGOTRI</b>	<b>(219E1A3370)</b>
<b>M.HARI KRISHNA</b>	<b>(219E1A3358)</b>
<b>M.ABHIRAM REDDY</b>	<b>(219E1A3362)</b>
<b>P.KEERTHANA</b>	<b>(219E1A3384)</b>
<b>P.HARISH</b>	<b>(219E1A3383)</b>

**Under the Guidance of**

**Dr. J. Sarada**

**Associate Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
SRI VENKATESWARA ENGINEERING COLLEGE**

**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)  
Karakambadi Road, TIRUPATI – 517507**

**2021-2025**

# **SRI VENKATESWARA ENGINEERING COLLEGE**

**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)  
Karakambadi Road, TIRUPATI – 517507**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



## **CERTIFICATE**

*This is to certify that the project report entitled “**OPTIMIZED CROP AND FERTILIZER RECOMMENDATION USING MACHINE LEARNING FOR SUSTAINABLE AGRICULTURE**”  
a bonafide record of the project work done and submitted by*

<b>N.DIVYA GANGOTRI</b>	<b>(219E1A3370)</b>
<b>M.HARI KRISHNA</b>	<b>(219E1A3358)</b>
<b>M.ABHIRAM REDDY</b>	<b>(219E1A3362)</b>
<b>P.KEERTHANA</b>	<b>(219E1A3384)</b>
<b>P.HARISH</b>	<b>(219E1A3383)</b>

for the partial fulfillment of the requirements for the award of B. Tech Degree in  
**COMPUTER SCIENCE AND ENGINEERING (AI & ML)**, JNT University Anantapur,  
Ananthapuramu.

**GUIDE**

**Head of the Department**

External Viva-Voce Exam Held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **DECLARATION**

We hereby declare that the project report entitled “**OPTIMIZED CROP AND FERTILIZER RECOMMENDATION USING MACHINE LEARNING FOR SUSTAINABLE AGRICULTURE**” done by us.

It Is submitted in partial fulfillment of the requirements for the award of the Bachelor’s degree in **Computer Science and Engineering(AI & ML)**. This project is the result of our own effort and it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above.

<b>DATE:</b>	<b>N.DIVYA GANGOTRI</b>	<b>(219E1A3370)</b>
<b>PLACE:</b>	<b>M.HARI KRISHNA</b>	<b>(219E1A3358)</b>
	<b>M.ABHIRAM REDDY</b>	<b>(219E1A3362)</b>
	<b>P.KEERTHANA</b>	<b>(219E1A3384)</b>
	<b>P.HARISH</b>	<b>(219E1A3383)</b>

## ACKNOWLEDMENT

We are thankful to our guide **Dr. J Sarada** for her valuable guidance and encouragement. Her helping attitude and suggestions have helped us in the successful completion of the project.

We would like to express our gratefulness and sincere thanks to **Dr. R.Swathi**, Head of the Department of COMPUTER SCIENCE AND ENGINEERING(AI & ML), for her kind help and encouragement during the course of our study and in the successful completion of the project work.

We have great pleasure in expressing our hearty thanks to our beloved Principal **Dr. C. Chandrasekhar**, for spending his valuable time with us to complete this project.

Successful completion of any project cannot be done without proper support and encouragement. We sincerely thank to the **Management** for providing all the necessary facilities during the course of study.

We would like to thank our parents and friends, who have the greatest contributions in all our achievements, for the great care and blessings in making us successful in all our endeavors.

<b>N.DIVYA GANGOTRI</b>	<b>(219E1A3370)</b>
<b>M.HARI KRISHNA</b>	<b>(219E1A3358)</b>
<b>M.ABHIRAM REDDY</b>	<b>(219E1A3362)</b>
<b>P.KEERTHANA</b>	<b>(219E1A3384)</b>
<b>P.HARISH</b>	<b>(219E1A3383)</b>

## TABLE OF CONTENTS

Chapter No.	Description	Page No.
	<b>Abstract</b>	<b>i</b>
	<b>List of Figures</b>	<b>ii</b>
	<b>List of Tables</b>	<b>iii</b>
	<b>List of Abbreviations</b>	<b>iv</b>
<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Project Description</b>	<b>2</b>
	2.1 Problem Definition	2
	2.2 Project Details	3
<b>3</b>	<b>Computational Environment</b>	<b>4</b>
	3.1 Software Specification	4
	3.2 Hardware Specification	4
	3.3 Software Features	8
<b>4</b>	<b>Feasibility Study</b>	<b>9</b>
	4.1 Technical Feasibility	9
	4.2 Social Feasibility	9
	4.3 Economical Feasibility	9
<b>5</b>	<b>System Analysis</b>	<b>10</b>
	5.1 Existing System	10
	5.1.1 Drawbacks of existing system	11
	5.2 Proposed System	11
	5.2.1 Advantages of proposed system	12
<b>6</b>	<b>System Design</b>	<b>13</b>

	6.1 UML Diagrams	14
	6.1.1 Class Diagram	15
	6.1.2 Use case Diagram	16
	6.1.3 Sequence Diagram	19
	6.1.4 Activity Diagram	20
	6.1.5 Deployment Diagram	22
<b>7</b>	<b>System Implementation</b>	<b>24</b>
	7.1 Implementation Process	24
	7.2 Modules	26
<b>8</b>	<b>Testing</b>	<b>31</b>
	8.1 Unit Testing	31
	8.2 Integration Testing	32
	8.3 System Testing	33
	8.4 Acceptance Testing	33
<b>9</b>	<b>Sample Source Code</b>	<b>34</b>
<b>10</b>	<b>Screen Layouts</b>	<b>41</b>
<b>11</b>	<b>Conclusion and Future Scope</b>	<b>46</b>
<b>12</b>	<b>Bibliography</b>	<b>48</b>
	12.1 References	48
	12.2 Websites	49

## **ABSTRACT**

Efficient agricultural practices are essential to meet the growing demands for food production while preserving environmental resources. This project focuses on developing a machine learning-based system that provides accurate crop and fertilizer recommendations tailored to specific regions and soil characteristics. By analyzing historical and regional data, the system identifies complex relationships between environmental factors, crop types, and fertilizer requirements, enabling farmers to make data-driven decisions. Advanced algorithms, including CatBoost and Artificial Neural Networks (ANN), are utilized for classification and prediction, ensuring high accuracy and reliability. CatBoost enhances performance by efficiently handling categorical and numerical agricultural data, while ANN captures complex non-linear relationships for improved recommendations. The proposed system promotes precision agriculture, helping farmers optimize resource usage, reduce costs, and enhance crop productivity. Furthermore, this approach contributes to sustainable farming by minimizing the overuse of fertilizers and mitigating environmental degradation. By bridging technology and agriculture, the project addresses critical challenges in food security, resource management, and ecological balance, fostering long-term agricultural sustainability.

## LIST OF FIGURES

Figure No.	Title	Description	Page No.
Fig 6.1	Class Diagram	Shows the static structure of system classes and their relationships	16
Fig 6.2	Use Case Diagram	Depicts system functionalities and user interactions	17
Fig 6.3	Sequence Diagram	Illustrates object interactions and the order of messages	20
Fig 6.4	Activity Diagram	Displays the flow of control between various system activities	22
Fig 6.5	Deployment Diagram	Represents physical deployment of software components on hardware nodes	23
Fig 8.1	Unit Testing Diagram	Visual representation of Unit testing process and results	32
Fig 8.2	Integration Diagram	Shows the integration of the machine learning system with the user interface	32
Fig 8.3	System Diagram	Overview of testing at the system level to ensure all components function properly	33



## LIST OF TABLES

Table No.	Title	Description	Page No.
Table 3.1	Software Specifications	Lists the operating System, Coding language, tools used	14
Table 3.2	Hardware Specifications	Details the hardware environment required to run the system	14
Table 7.1	Modules List	Outlines all functional modules implemented in the system	26
Table 8.1	Unit Test Cases	Provides sample inputs and expected outputs for unit testing	32
Table 8.2	Integration Testing Overview	Describes the integration of model and interface components	32

## LIST OF ABBREVIATIONS

S.NO	TERM	ABBREVIATIONS
1	Artificial Intelligence	AI
2	Machine Learning	ML
3	Artificial Neural Network	ANN
4	Random Forest	RF
5	Extreme gradient Boosting	XG Boost
6	Light Gradient Boosting Machine	Light GBM
7	User Interface	UI
8	Hyper Text Markup Language	HTML
9	Structured Query Language	SQL
10	Uniform Resource Locator	URL

## 1.INTRODUCTION

Agriculture is a vital component of global development, serving as the primary source of livelihood for millions and a key contributor to food security and economic stability. However, traditional farming practices often struggle to keep pace with the growing demand for food, rising input costs, and the adverse effects of climate change. One of the major challenges farmers face today is making informed decisions about crop selection and fertilizer application. These decisions are typically based on conventional knowledge, generalized guidelines, or trial-and-error methods, which can result in inefficient use of resources, poor crop yields, and environmental degradation.

In recent years, advancements in machine learning (ML) have opened new possibilities for transforming agriculture into a more precise, data-driven, and sustainable industry. Machine learning enables the analysis of vast datasets containing environmental, soil, and crop information to uncover patterns and relationships that are not easily detectable through manual analysis. Leveraging this capability, the present project aims to develop an intelligent system that offers optimized crop and fertilizer recommendations by analyzing region-specific data, such as soil properties, weather patterns, and historical agricultural trends.

The core objective of this project is to provide personalized and actionable recommendations that can help farmers increase productivity while preserving natural resources. To achieve this, the system employs advanced ML algorithms—specifically CatBoost and Artificial Neural Networks (ANN)—known for their high performance in classification and predictive tasks. These models are trained on diverse agricultural datasets to ensure robust predictions that adapt to various soil types, climates, and cropping patterns.

Furthermore, this project promotes the concept of precision agriculture, where data-driven techniques guide farming decisions to optimize input usage and minimize environmental harm modern technology.

## 2. PROJECT DESCRIPTION

### PROBLEM DEFINITION

With the global population continuing to rise and environmental concerns becoming more pressing, agriculture is under increasing pressure to produce more food using fewer resources. Traditional farming practices often rely on outdated methods and generalized recommendations for crop selection and fertilizer usage. These conventional techniques fail to consider local variations in soil composition, climate conditions, and crop requirements, leading to poor crop yields, unnecessary expenditure on fertilizers, and long-term degradation of soil health.

In many rural and semi-urban regions, farmers still make crucial agricultural decisions based on intuition, trial and error, or generic government advisories. This approach often results in inefficient use of water, fertilizer, and other critical inputs, contributing to environmental pollution and economic loss. Moreover, improper fertilizer application can cause soil nutrient imbalance, groundwater contamination, and adverse effects on plant growth. With climate variability and unpredictable weather patterns further complicating farming activities, there is a clear need for smarter, region-specific solutions.

Existing recommendation systems are either manually operated or use basic computational tools that fail to account for the complex interplay of environmental, biological, and chemical factors influencing agriculture. These systems lack adaptability, precision, and scalability—making them less effective for real-time, data-driven decision-making at the grassroots level. To overcome these challenges, this project proposes a machine learning-based solution that delivers optimized recommendations for both crop selection and fertilizer usage based on real-time and historical data. By utilizing algorithms like **CatBoost** for handling structured data and **Artificial Neural Networks (ANN)** for identifying non-linear relationships, the system is capable of generating highly accurate, customized outputs.

## PROJECT DETAILS

This project, titled “**Optimized Crop and Fertilizer Recommendations Using Machine Learning for Sustainable Agriculture,**” aims to create a smart, data-driven system to assist farmers in making informed decisions regarding crop selection and fertilizer application. With agriculture facing increasing challenges from climate change, soil degradation, and inefficient input use, there is a growing need for intelligent systems that can enhance productivity while ensuring environmental sustainability. The core goal is to provide customized recommendations based on regional soil characteristics, historical crop performance, and climatic conditions.

To address this, the project leverages advanced machine learning techniques—primarily **CatBoost** and **Artificial Neural Networks (ANN)**—to analyze diverse agricultural datasets. CatBoost, a gradient boosting algorithm, is particularly effective for handling categorical features common in agricultural data such as soil type, crop variety, and seasonal cycles. ANN complements this by capturing complex non-linear interactions among multiple features, allowing the model to produce more accurate predictions and recommendations.

The workflow begins with the **collection of agricultural data** from government repositories, field surveys, and online agricultural databases. The dataset includes variables such as soil nutrient levels (N, P, K), pH, temperature, humidity, rainfall, and previous crop yields. The **preprocessing phase** involves handling missing values, encoding categorical data, normalization, and outlier detection to prepare a high-quality dataset for training.

The machine learning models are trained on this data using **supervised learning techniques**, with the dataset split into training and testing sets to validate model performance. The system evaluates predictions using metrics such as **accuracy, precision, and F1-score**, ensuring reliability in real-world scenarios.

### 3. COMPUTATIONAL ENVIRONMENT

#### SOFTWARE SPECIFICATION

- Operating system : Windows 7 Ultimate.
- Coding : Python.  
Language
- Front-End : Python.
- Back-End : Django-ORM
- Designing : Html, css, javascript.
- Data Base : MySQL (WAMP Server).

#### HARDWARE SPECIFICATION

- Processor : I3/Intel processor
- RAM : 8 GB
- Hard Disk : 160 GB
- Key Board : Standard Windows
- Mouse : Two or Three Button
- Mouse Monitor : SVGA

#### SOFTWARE FEATURES

When it comes to Python software for Machine Learning (ML) and Deep Learning (DL), several libraries and frameworks provide the necessary tools for building, training, and deploying models. Python is the go-to programming language for data science, offering a rich ecosystem of tools that support various aspects of machine learning and deep learning. Below are some of the most widely used Python libraries for ML and DL.

##### 1. Scikit-Learn

Scikit-learn is one of the most popular and easy-to-use libraries for traditional machine learning algorithms. It provides tools for classification, regression, clustering,

dimensionality reduction, and model selection. It is built on top of NumPy, SciPy, and matplotlib, which makes it a good fit for data analysis and visualization. Scikit-learn is known for its simplicity and efficiency, making it a top choice for beginners and experts alike.

Key features include:

- Various algorithms for classification (e.g., SVM, decision trees, k-NN), regression (e.g., linear regression, decision trees), clustering (e.g., kmeans), and dimensionality reduction (e.g., PCA).
- Model evaluation and selection tools, such as cross-validation and grid search.
- Integration with other libraries like Pandas for data manipulation and Matplotlib for data visualization.

## **2. TensorFlow**

Developed by Google, TensorFlow is one of the most powerful and widely used deep learning frameworks. It is an open-source library that supports both machine learning and deep learning applications, making it suitable for a range of tasks from simple ML models to complex deep neural networks. TensorFlow is designed to be flexible and scalable, which allows it to be used in both research and production environments.

Key features of TensorFlow include:

- Support for both CPU and GPU computation, allowing for fast model training.
- A comprehensive ecosystem of tools, such as TensorFlow Lite for mobile devices and TensorFlow.js for running models in the browser.
- A high-level API, Keras, which simplifies the development of neural networks by providing an easy-to-use interface.
- 

## **3. Keras**

Keras is a high-level API built on top of TensorFlow (and previously Theano and CNTK). It provides a simplified interface for building and training deep learning

models. Keras allows you to quickly prototype models and experiment with different architectures without needing to write much low-level code. This makes it a favorite among both beginners and researchers who need to develop deep learning models efficiently.

Key features of Keras include:

- Easy-to-use, modular design that allows you to stack layers and define the model structure intuitively
- Support for both convolutional neural networks (CNNs) and recurrent neural networks (RNNs).
- Integration with TensorFlow, allowing for access to advanced features such as distributed training and custom model building.

#### **4. PyTorch**

PyTorch is another highly popular deep learning framework developed by Facebook's AI Research lab. It is known for its dynamic computational graph, which allows for greater flexibility in building complex models. PyTorch's ease of use and flexibility make it a great choice for research and experimentation.

Key features of PyTorch include:

- Dynamic computational graphs, making it easier to modify the model during runtime (useful for research).
- Strong GPU support for accelerated computation.
- A large and active community that continuously contributes to a wealth of pre-trained models and tutorials.
- Integration with Python libraries like NumPy and SciPy, making it easier to manipulate data.

#### **5. Pandas**

While not directly a machine learning or deep learning library, Pandas is an essential tool for data manipulation and preprocessing. It provides data structures like DataFrames that make it easy to load, clean, transform, and analyze large datasets. It is often used in conjunction with ML and DL frameworks to prepare data before training models.



Key features of Pandas include:

- Easy handling of missing data, data transformations, and aggregation.
- Powerful indexing and data alignment capabilities.

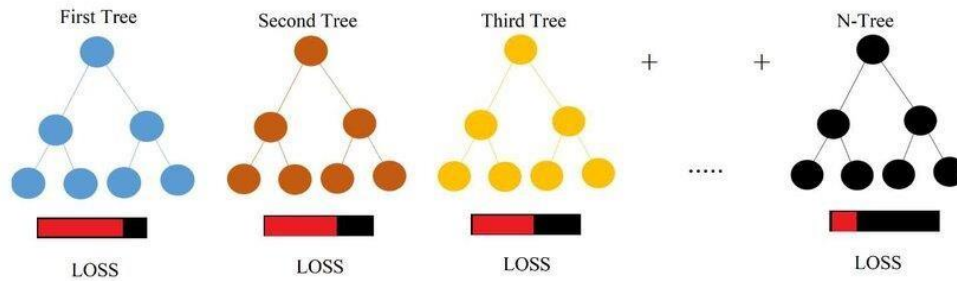
### 1. CatBoost Algorithm

CatBoost (Categorical Boosting) is a high-performance machine learning algorithm developed by Yandex, specifically designed to handle categorical features efficiently. It is a gradient boosting algorithm based on decision trees and is widely recognized for its superior accuracy, speed, and ease of use in classification and regression tasks. In the context of this project, CatBoost is employed to generate optimized crop and fertilizer recommendations by learning from complex agricultural datasets that include both numerical and categorical variables such as soil type, climate region, and crop names. One of the most significant strengths of CatBoost is its **native support for categorical features** without the need for explicit preprocessing like one-hot encoding or label encoding. It converts categorical variables internally using ordered statistics and combinations, which enhances model accuracy and prevents overfitting. This is particularly useful in agriculture, where datasets often contain numerous non-numeric attributes.

Another advantage of CatBoost is its **ordered boosting technique**, which ensures more stable training and eliminates the prediction shift problem typically encountered in traditional gradient boosting. This ordered boosting uses permutations of the dataset to prevent target leakage, resulting in more reliable models even on smaller datasets. CatBoost also includes **efficient handling of missing values** and provides built-in regularization, making it robust against noisy data. The model can learn non-linear relationships between inputs and outputs, which is essential for capturing the intricate dependencies between soil nutrients, crop types, and fertilizer needs.

In summary, CatBoost enhances the prediction accuracy for region-specific crop and fertilizer recommendations. Its ability to automatically process categorical data,

coupled with its fast and accurate learning capabilities, makes it a powerful tool for promoting precision agriculture and sustainable farming practices.



## 6. Theano

Although Theano is no longer actively developed (as of 2017), it was one of the first deep learning frameworks and still plays a role in the development of newer frameworks like TensorFlow and Keras. Theano is used for numerical computation and optimization, making it a suitable tool for building deep learning models.

Key features of Theano include:

- Optimization of mathematical expressions, making training more efficient.
- GPU support for accelerated computation.

While its development has ceased, Theano paved the way for modern deep learning libraries.

Python offers a wide variety of libraries and frameworks for machine learning and deep learning, each catering to different needs and levels of complexity. Scikit-learn is great for traditional machine learning models, while TensorFlow, Keras, and PyTorch are powerful choices for deep learning applications. Pandas remains indispensable for data manipulation, and libraries like XGBoost and LightGBM offer specialized tools for gradient boosting. By leveraging these tools, data scientists and researchers can build, train, and deploy robust models that power everything from simple applications to cutting-edge AI systems.

## **4.FEASIBLE STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY
- ECONOMICAL FEASIBILITY

### **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it..

### **ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system.

## SYSTEM ANALYSIS

### EXISTING SYSTEM

Current agricultural recommendation systems are predominantly based on traditional methods or generic advisory platforms that lack precision and adaptability. In many regions, crop and fertilizer suggestions are provided through agricultural officers, printed manuals, or outdated web portals that use static guidelines irrespective of location-specific variables like soil health, weather patterns, or prior crop history. While these conventional systems offer baseline support, they fail to deliver accurate, real-time, and personalized solutions for modern farming needs.

Farmers often rely on their intuition or historical practices to decide which crops to grow and how much fertilizer to apply. This trial-and-error approach frequently leads to inefficient resource use, poor crop yields, and long-term degradation of soil fertility. Moreover, blanket fertilizer recommendations contribute to overuse or underuse, resulting in soil imbalances, reduced productivity, and environmental harm such as groundwater contamination and greenhouse gas emissions.

Some modern platforms incorporate basic decision support tools or rule-based models, but they lack the capacity to analyze large datasets or adapt to the complexity of real-world agricultural conditions. These tools are generally incapable of learning from new data or understanding non-linear relationships between crop types, nutrient requirements, and climatic influences.

Additionally, these systems do not scale effectively for broader deployment across different regions with unique agro-ecological characteristics. They also require significant manual effort to update recommendations as agricultural patterns evolve. The lack of real-time adaptability, integration with local data sources, and intelligent learning capabilities renders existing systems inadequate for the dynamic demands of sustainable agriculture.

### Drawbacks of Existing System

- Relies on static, generalized recommendations without accounting for region-specific variations.
- Heavy dependence on manual expertise or outdated guidelines, reducing accuracy
- Lacks the intelligence to model complex relationships between soil, climate, and crop dynamics.
- No integration with real-time environmental or sensor-based data for dynamic predictions.
- Poor scalability and adaptability to diverse agricultural regions and conditions.
- Inability to learn from new data, limiting continuous improvement and evolution of suggestions.
- Inefficient fertilizer usage leading to soil degradation, increased costs, and environmental risks.
- No personalization, resulting in sub-optimal crop choices and decreased productivity.

### PROPOSED SYSTEM

In this project, the proposed system introduces a machine learning-based intelligent recommendation framework for crop and fertilizer selection. By analyzing soil parameters, climate data, and crop history, the system provides highly accurate and customized suggestions using advanced algorithms like **CatBoost** and **Artificial Neural Networks (ANN)**. This modern, data-driven approach addresses the shortcomings of conventional methods by leveraging automation, predictive analytics, and adaptive learning. The system begins by collecting diverse agricultural datasets including soil nutrient content (NPK values), pH, temperature, rainfall, and past crop yields. This data is then subjected to preprocessing steps such as normalization, missing value treatment, and encoding of categorical features to prepare it for model training. The **CatBoost algorithm** is employed for its capability to handle categorical and numerical data without extensive preprocessing, while the **ANN** model captures intricate non-linear relationships that influence yield and fertilizer needs. These

models are trained and validated on real-world agricultural data to ensure robust performance. Once trained, the system allows farmers to input their specific field parameters, and it outputs optimized crop choices along with precise fertilizer quantities. The user interface, built using Django, ensures ease of access even for users with limited technical knowledge. The proposed system is designed to be scalable, user-friendly, and adaptable to new data, enabling real-time recommendations that evolve with changing agricultural trends.

### **Advantages of Proposed System**

- Provides region-specific and soil-specific recommendations, increasing decision accuracy.
- Utilizes advanced ML algorithms (CatBoost and ANN) for superior predictive performance.
- Reduces reliance on manual judgment, promoting data-driven decision-making.
- Supports real-time crop and fertilizer recommendation through a userfriendly interface.
- Easily scalable and adaptable to different regions and evolving datasets.
- Minimizes fertilizer waste and supports environmental conservation efforts.
- Contributes to precision agriculture and long-term sustainable farming goals

## 6. SYSTEM DESIGN

System design is the foundational phase where the structure, architecture, components, data flow, and interactions within the system are defined to meet specific functional and non-functional requirements. In the context of this project - **“Optimized Crop and Fertilizer Recommendations Using Machine Learning for Sustainable Agriculture”**—the design process is crucial for integrating diverse agricultural data sources, machine learning models, and user interfaces into a cohesive and efficient solution.

The architecture of the proposed system consists of multiple interdependent modules: **data collection, preprocessing, feature extraction, model training using CatBoost and Artificial Neural Networks (ANN), prediction generation, user interaction through the web interface, and result logging for future feedback**. Each module is developed to fulfill a specific function, contributing to the end goal of delivering region-specific, accurate crop and fertilizer recommendations.

The system design begins with the **data input module**, which collects agricultural datasets including soil parameters (like NPK values, pH), weather patterns, historical crop yields, and fertilizer usage records. This data is gathered from government portals, sensor devices, and field reports. The collected data is routed to the **preprocessing module**, where missing values are handled, categorical features are encoded, and numerical data is normalized to ensure consistency.

The **core of the system involves two machine learning models**: CatBoost and ANN. The CatBoost model excels in processing structured tabular data and categorical inputs, while the ANN model captures complex non-linear patterns across multiple variables. These models are trained on labeled datasets using supervised learning, allowing them to predict the most suitable crop and fertilizer for a given set of environmental inputs. The training process includes splitting data into training and testing sets, model evaluation, and performance tuning.

The **prediction module** interacts with the frontend, developed using the **Django framework**, where users input their soil and climate data. Based on the model's

inference, the output is displayed as an optimized crop recommendation and precise fertilizer composition (e.g., urea, DAP, potash). The predictions are also stored in a backend database for future enhancement.

A **modular design approach** ensures that each component can be updated or scaled independently, making the system adaptable to changing agricultural trends and larger datasets. Clear interfaces and APIs ensure smooth data flow and system integration. The overall system design thus provides a **robust, scalable, and intelligent framework** for enabling data-driven decisionmaking in agriculture. It not only increases yield and cost-efficiency but also supports sustainability through optimized fertilizer usage and smart farming practices.

## UML DIAGRAMS

The Unified Modelling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modelling Language (UML) was



designed to respond to these needs. Simply, Systems design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

### Contents of UML

In general, a UML diagram consists of the following features:

- **Entities:** These may be classes, objects, users or systems behaviors.
- **Generalization** -- a solid line with an arrow that points to a higher abstraction of the present item.
- **Association** -- a solid line that represents that one entity uses another entity as part of its behaviour.
- **Dependency** -- a dotted line with an arrowhead that shows one entity depends on the behaviour of another entity.

In this project four basic UML diagrams have been explained

- a. Class Diagram
- b. Use Case Diagram
- c. Sequence Diagram
- d. Activity Diagram
- e. Deployment Diagram

### Class Diagram

UML class diagrams model static class relationships that represent the fundamental architecture of the system. Note that these diagrams describe the relationships between classes, not those between specific objects instantiated from those classes. Thus the diagram applies to all the objects in the system.

A class diagram consists of the following features:

- **Classes:** These titled boxes represent the classes in the system and contain information about the name of the class, fields, methods and access specifiers. Abstract roles of the Class in the system can also be indicated.
- **Interfaces:** These titled boxes represent interfaces in the system and contain information about the name of the interface and its methods. Relationship Lines that model the relationships between classes and interfaces in the system.
- **Dependency:** A dotted line with an open arrowhead that shows one entity depends on the behavior of another entity. Typical usages are to represent that one class instantiates another or that it uses the other as an input parameter
- **Aggregation:** Represented by an association line with a hollow diamond at the tail end. An aggregation models the notion that one object uses another object without "owning" it and thus is not responsible for its creation or destruction.
- **Inheritance:** A solid line with a solid arrowhead that points from a sub-class to a super class or from a sub-interface to its super-interface.
- **Implementation:** A dotted line with a solid arrowhead that points from a class to the interface that it implement
- **Composition:** Represented by an association line with a solid diamond at the tail end. A composition models the notion of one object "owning" another and thus being responsible for the creation and destruction of another object.

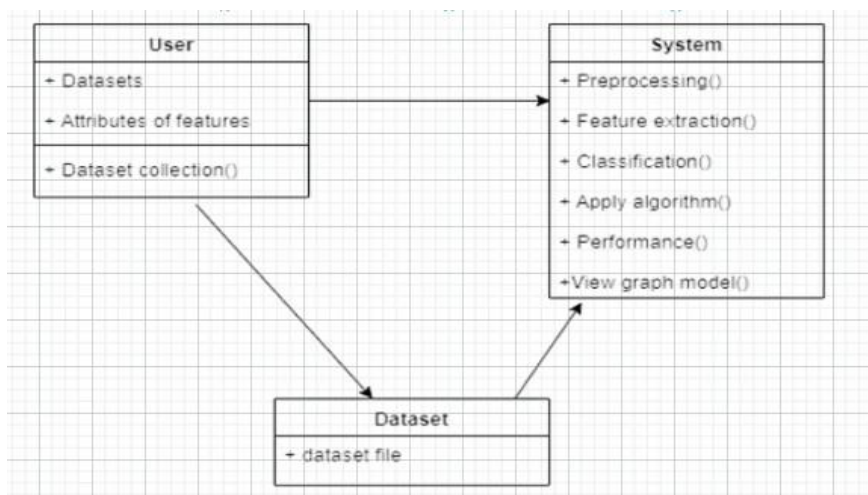


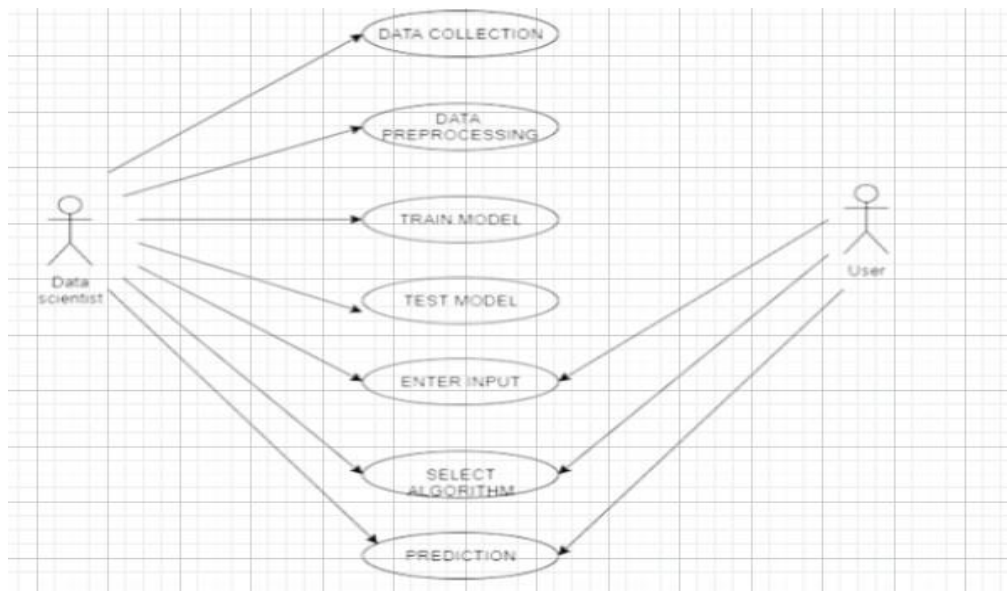
Fig 6.1: Class Diagram

### Use case diagram

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms.

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined.

The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Fig 6.2: Use case Diagram**

## Parts of Use cases

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

- **Actors**

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

- **System boundary boxes (optional)**

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not **Relationships.**

- **Include**

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case".

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviours from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "<<include>>". This usage resembles a macro expansion where the included use case behavior is placed inline in the base use case behavior. There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

- **Extend**

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "<<extend>>".

Depending on the modeler's approach "optional" may mean "potentially not executed with the base use case" or it may mean "not required to achieve the base use case goal".

- **Generalization**

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviours, requirements, constraints, and assumptions with a more general use case. In this case,

describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation).

- **Associations**

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modelled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicate the direction of the initial invocation of the relationship or to indicate the primary actor within the use case. The arrowheads imply control flow and should not be confused with data flow.

## **STEPS TO DRAW USE CASES**

- Identifying Actor
- Identifying Use cases
- Review your use case for completeness

## **Sequence Diagram**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram.

## **Elements of sequence diagram**

The sequence diagram is an element that is used primarily to showcase the interaction that occurs between multiple objects. This interaction will be shown over certain period of time. Because of this, the first symbol that is used is one that symbolizes the object.

- **Lifeline**

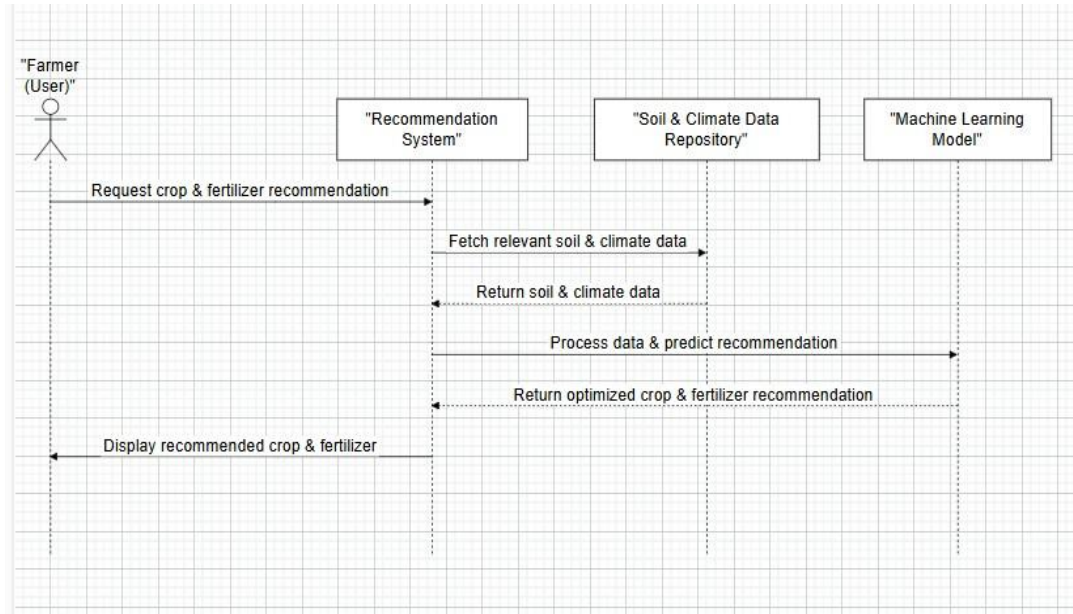
A lifeline will generally be generated, and it is a dashed line that sits vertically, and the top will be in the form of a rectangle. This rectangle is used to indicate both the instance and the class. If the lifeline must be used to denote an object, it will be underlined.

- **Messages**

To showcase an interaction, messages will be used. These messages will come in the form of horizontal arrows, and the messages should be written on top of the arrows. If the arrow has a full head, and it's solid, it will be called a synchronous call. If the solid arrow has a stick head, it will be an asynchronous call. Stick heads with dash arrows are used to represent return messages.

### **Steps to Create a Sequence Diagram**

- Set the context for the interaction, whether it is a system, subsystem, operation or class.
- Set the stage for the interaction by identifying which objects play a role in interaction.
- Set the lifetime for each object.
- Start with the message that initiates the interaction.
- Visualize the nesting of messages or the points in time during actual computation.
- Specify time and space constraints, adorn each message with timing mark and attach suitable time or space constraints.



**Fig 6.3: Sequence Diagram Activity Diagram**

### Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

### How to draw Activity Diagram?

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram are not exactly a flow chart as they have some additional capabilities. These additional capabilities include branching, parallel flow, swim lane etc. Before drawing an activity diagram we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities we need to understand how they are associated with constraints and conditions. So before drawing an activity diagram we should identify the following elements.

- Activities
- Association
- Conditions
- Constraints

The following are the basic notational elements that can be used to make up a diagram:

**Initial state:**

An initial state represents a default vertex that is the source for a single transition to the default state of a composite state. There can be at most one initial vertex in a region. The outgoing transition from the initial vertex may have a behavior, but not a trigger or guard. It is represented by Filled circle, pointing to the initial state.

**Final state :**

A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed. It is represented by Hollow circle containing a smaller filled circle, indicating the final state.

**Rounded rectangle**

It denotes a state. Top of the rectangle contains a name of the state. Can contain a horizontal line in the middle, below which the activities that are done in that state are indicated.

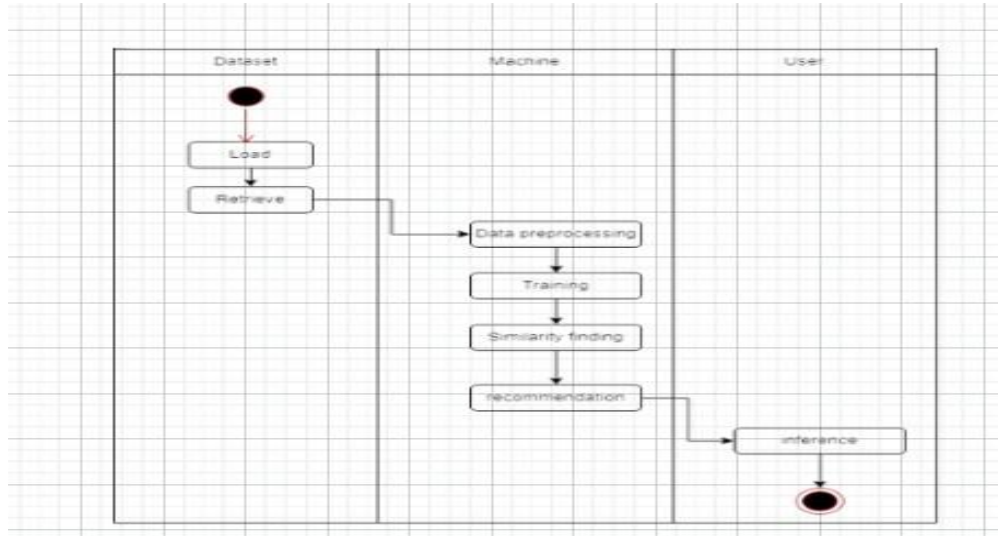
**Arrow**

It denotes transition. The name of the event (if any) causing this transition labels the arrow body.

**Steps To Construct Activity Diagram**

- Identify the preconditions of the workflow
- Collect the abstractions that are involved in the operations
- Beginning at the operation's initial state, specify the activities and actions.
- Use branching to specify conditional paths and iterations
- Use forking & joining to specify parallel flows of control.





**Fig 6.4: Activity Diagram**

### **Deployment Diagram**

A Deployment Diagram is part of the UML (Unified Modeling Language) used to model the physical deployment of artifacts (software components) on hardware nodes. It shows how software systems are deployed in the real world.

#### **Steps to Construct a Deployment Diagram**

##### **a. Understand the System Requirements**

- Identify the physical devices and their roles.
- Understand the software components to be deployed.

##### **b. Identify Nodes**

- Nodes are physical or virtual hardware
- Represent nodes as **3D boxes** (cube-shaped).

##### **c. Identify Artifacts**

- Artifacts are the pieces of software to be deployed
- Artifacts are shown inside nodes using rectangles with the «artifact» keyword.

**d. Map Artifacts to Nodes**

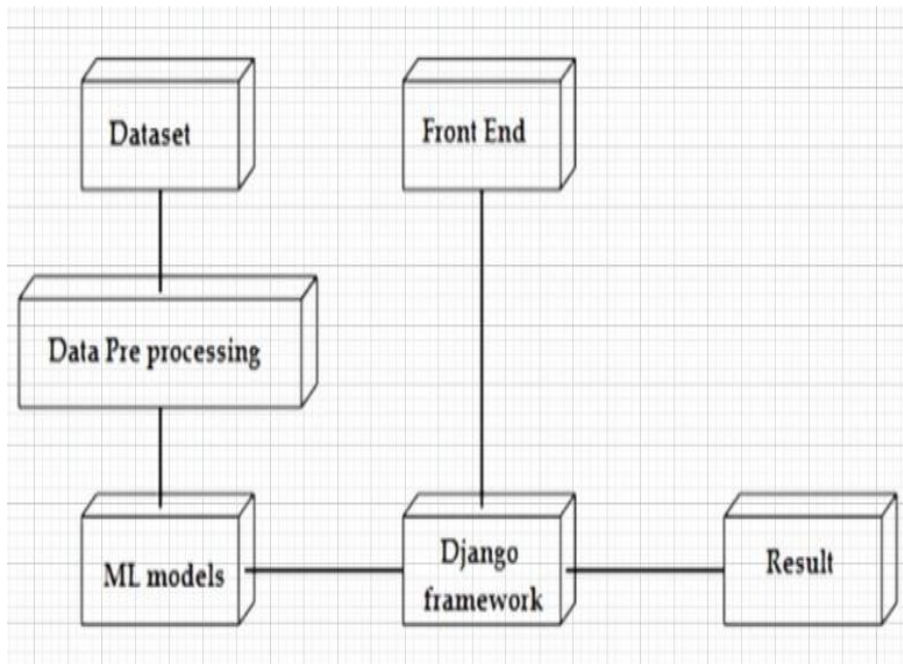
- Show which artifacts run on which nodes.
- Artifacts are drawn **inside** the nodes.

**e. Define Communication Paths**

- Use **solid lines** between nodes to show communication links.
- Label connections if necessary (e.g., HTTP, JDBC).

**f. Add Additional Elements**

- Optional: Add components, devices, or execution environments.
- Add stereotypes like «device», «executionEnvironment», etc.



**Fig 6.5: Deployment Diagram**

## 7. SYSTEM IMPLEMENTATION

### IMPLEMENTATION PROCESS

The implementation of the proposed crop and fertilizer recommendation system using machine learning involves a series of critical phases, including data acquisition, preprocessing, model training, evaluation, and deployment within a user-accessible web interface. This step-by-step approach ensures that the system delivers accurate, personalized, and sustainable agricultural guidance.

#### 1.Data Collectionand Preprocessing:

The first stage involves collecting comprehensive agricultural data from multiple sources, including government agriculture portals, research datasets, and soil health records. The data includes parameters such as nitrogen (N), phosphorus (P), potassium (K) levels, soil pH, moisture content, historical crop yields, temperature, rainfall, and humidity.

Preprocessing techniques are then applied to clean the dataset—this includes handling missing values, encoding categorical data (e.g., crop types, soil classes), normalizing numerical attributes, and removing outliers. These steps ensure that the input data is accurate, consistent, and suitable for machine learning modeling.

#### 2.Model Selection – CatBoost and ANN:

Two powerful machine learning models are selected for this project: **CatBoost** and **Artificial Neural Networks (ANN)**. CatBoost is employed for its efficiency in handling tabular and categorical data without requiring complex preprocessing. It is particularly suited for structured datasets common in agriculture. Meanwhile, ANN is chosen for its ability to model complex, nonlinear relationships between input variables. The architecture of the ANN is customized with multiple hidden layers, ReLU activation functions, and dropout layers to ensure robust learning.

### 3. Training the Models:

Both models are trained using a supervised learning approach. The dataset is split into training, validation, and testing subsets in a 70:15:15 ratio. The

**CatBoost model** is trained using gradient boosting with tuned hyperparameters for depth, learning rate, and iterations.

### 4. Evaluation:

After model training, rigorous evaluation is conducted to assess the predictive performance and generalizability of both the CatBoost and ANN models. The reserved **test dataset** (15% of the original dataset) is used to ensure that the models are evaluated on previously unseen data.

Key performance metrics include:

- **Accuracy:** Measures the overall correctness of the model's predictions.
- **Precision:** Indicates how many of the predicted positive results are actually correct.
- **Recall (Sensitivity):** Reflects the model's ability to identify all relevant cases within the dataset.
- **F1-Score:** A harmonic mean of precision and recall, offering a balanced assessment.
- **R<sup>2</sup> Score (Coefficient of Determination):** Particularly important for regression tasks like fertilizer quantity estimation, indicating how well the model explains the variability of the output.

Visual tools such as **confusion matrices** help to pinpoint where the model performs well or poorly across different crop categories. **Error distribution plots**, **residual analysis**, and **feature importance charts** (especially in CatBoost) are also generated to interpret model decisions and validate the robustness of predictions.

## 5. System Integration:

The final phase focuses on seamlessly integrating the trained models into a functional and user-friendly **Django-based web application**, transforming the research into a practical tool for real-world use by farmers and agricultural professionals.

- **User Interface (UI):** A clean, responsive, and mobile-compatible interface is designed to allow users to input soil characteristics (e.g., NPK levels, pH, moisture) and local weather data (e.g., rainfall, temperature).
- **Model Deployment:** The trained CatBoost and ANN models are serialized using appropriate tools and loaded within the Django framework. API endpoints are created to handle real-time data input, model inference, and result output.
- **Recommendation Output:** Once the input is submitted, the backend invokes the appropriate model to return:
  - The **most suitable crop** for the given conditions.
  - A **customized fertilizer mix** recommendation optimized for yield and soil health.
- **Database Integration:** All inputs and outputs are stored in a relational database. This facilitates:
  - Historical tracking and user feedback loops.
  - Periodic model retraining based on new data.
  - Advanced analytics and reporting features.
- **System Optimization:** The application is designed to function efficiently even in areas with limited internet connectivity. Techniques such as **lazy loading**, **asynchronous requests**, and **lightweight templates** are used to enhance performance on low-bandwidth networks.

Overall, this integration transforms advanced machine learning algorithms into an **intelligent, accessible decision-support tool** that enables farmers to make data-driven choices—boosting productivity, conserving resources, and supporting the broader goals of sustainable and precision agriculture.

## **Modules List :**

### **1. Data Collection Module**

The Data Collection Module is the foundational step of the system, responsible for aggregating relevant and diverse agricultural data from multiple sources. These sources include government agricultural departments, meteorological data centers, agricultural universities, and online datasets. The data primarily consists of soil attributes such as nitrogen (N), phosphorus (P), potassium (K), pH level, moisture content, and texture. In addition, weather related data like temperature, rainfall, and humidity are collected, alongside historical crop yield records and fertilizer usage patterns from various regions. To ensure data reliability, quality checks are conducted to eliminate inconsistencies, duplicates, and incomplete entries. In some cases, real-time data acquisition can be integrated from IoT-based soil sensors or mobile apps that allow farmers to upload localized data. This diverse dataset is essential for training machine learning models capable of learning region-specific agricultural patterns.

The collected data is organized into structured formats like CSV or Excel files and stored in a relational database or data lake for scalable access. The goal is to ensure the dataset is rich, representative, and balanced across different geographic zones, crop types, and soil conditions. This module enables the system to learn complex relationships between input factors and agricultural outcomes, forming the backbone for accurate and personalized recommendations.

### **2. Data Preprocessing Module**

The Data Preprocessing Module is critical for transforming raw agricultural data into a clean, structured, and machine-readable format. Raw data collected from different sources often includes missing values, inconsistent formats, and irrelevant features. This module handles these challenges to ensure the machine learning models receive high-quality input.

Key tasks performed in this module include handling missing values using techniques such as mean/mode imputation or KNN-based filling. Categorical variables such as crop names, soil types, and region names are encoded using

Label Encoding or One-Hot Encoding, depending on the model's requirement. Numerical values are normalized or standardized to bring all features into a similar range, which is especially important for gradient-based learning in ANN. Outlier detection and removal is also performed using statistical methods or clustering to ensure that extreme values do not distort model training. Additionally, feature selection and dimensionality reduction are carried out to retain only the most relevant variables, improving computational efficiency and reducing overfitting.

The data is then split into training, validation, and testing sets, typically in a 70:15:15 ratio. This enables proper evaluation of the model's generalization ability. The module also includes data balancing if the dataset has an uneven distribution of crop types or fertilizer usage.

In essence, this module acts as the bridge between raw data and machine learning by refining the inputs to ensure consistency, reduce noise, and extract meaningful patterns—laying a strong foundation for accurate model training and predictions.

### **3. Feature Engineering Module**

The Feature Engineering Module plays a vital role in enhancing the predictive power of the system by creating meaningful new variables from existing data. This process involves deriving new features, transforming existing ones, and combining variables in innovative ways to help the machine learning models understand the complex relationships within agricultural data.

For instance, the NPK values (Nitrogen, Phosphorus, Potassium) can be combined to calculate a soil fertility index or a nutrient balance ratio. Weather attributes such as rainfall, humidity, and temperature are processed into seasonal indicators or drought stress levels, which directly influence crop suitability. Interactions between soil pH and fertilizer type may also be modeled to highlight specific tolerances or deficiencies.

Domain knowledge from agronomists and agricultural experts is integrated into this module to guide feature construction. For example, crops are categorized into Kharif, Rabi, or Zaid seasons, and region-specific features like average growing season duration or pest resistance levels can also be added.

Feature selection techniques such as correlation analysis, mutual information gain, or Recursive Feature Elimination (RFE) are used to retain only the most impactful

variables. This step ensures that the models focus on relevant inputs and are not misled by noise or redundant data.

Overall, this module transforms the input dataset into a more informative, enriched, and compact feature space, which leads to better training outcomes, improved model interpretability, and stronger predictive performance.

#### **4. Model Building Module (CatBoost & ANN)**

The Model Building Module serves as the foundational computational engine of the system, tasked with initializing, configuring, and compiling two highly capable machine learning models—CatBoost and Artificial Neural Networks (ANNs)—each bringing unique strengths to the analysis of agricultural data. These models are strategically chosen to address the multifaceted nature of agronomic datasets, which often contain a mix of structured numerical values such as soil nutrients, temperature, and rainfall, alongside categorical features like soil type, crop variety, and planting season.

CatBoost, a powerful gradient boosting algorithm developed by Yandex, is particularly well-suited for such structured data, offering built-in support for categorical feature handling without requiring manual preprocessing or external encoding. It utilizes ordered boosting to eliminate target leakage and reduce overfitting, while its native regularization mechanisms ensure stable learning, especially in scenarios with limited or noisy data. The model is fine-tuned through hyperparameters such as learning rate, depth of trees, number of iterations, and regularization coefficients, which are optimized using systematic search techniques like Grid Search, Random Search, or Bayesian Optimization to ensure optimal performance. CatBoost is also efficient in terms of training speed and memory usage, making it ideal for real-world deployment in resource constrained environments. Complementing this, the Artificial Neural Network model is designed to capture complex, non-linear patterns and interactions that may be difficult for tree-based models to detect.

The ANN architecture typically includes an input layer corresponding to all relevant features, followed by one or more hidden layers equipped with ReLU (Rectified Linear Unit) activation functions to introduce non-linearity, and concludes with an output layer designed to produce predictions for either classification (e.g., crop



suitability) or regression (e.g., fertilizer dosage). To enhance the generalization ability of the ANN and prevent overfitting, the architecture incorporates dropout layers that randomly deactivate neurons during training, and batch normalization layers that stabilize learning and accelerate convergence. Both models are compiled with appropriate loss functions based on the problem type—crossentropy loss for classification tasks and mean squared error for regression—ensuring that predictions are both accurate and reliable. Optimization algorithms such as Adam, SGD (Stochastic Gradient Descent), or RMSprop are employed to update the model weights efficiently during training. Once trained, these models become capable of ingesting preprocessed agricultural data and producing highly accurate, context-aware predictions and recommendations.

The combination of CatBoost’s ability to interpret structured, feature-rich data and ANN’s capacity for deep pattern recognition results in a hybrid modeling approach that is both powerful and versatile. This dual-model setup allows the system to deliver precise, actionable insights for agricultural planning, such as selecting the most suitable crops for a given environment, estimating optimal fertilizer requirements, or forecasting yield outcomes based on evolving weather patterns. By integrating these models in a single cohesive module, the system is empowered to support data-driven decision-making and promote sustainable, efficient agricultural practices.

## **5. Training Module**

The Training Module is responsible for teaching the machine learning models how to recognize patterns and make accurate predictions based on the agricultural data. It begins by feeding the preprocessed and feature engineered dataset into the CatBoost and ANN models in a controlled training environment.

For CatBoost, the model is trained using supervised learning by iteratively reducing the prediction error on the training set using gradient boosting techniques. It evaluates the contribution of each feature and refines its internal decision trees accordingly. Training involves hundreds or thousands of boosting rounds, with early stopping applied to prevent overfitting.

## 8. TESTING

### TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

#### Types of Testing

The common types of testing are :

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

#### 8.1: UNIT TESTING

Unit testing is testing the smallest testable unit of an application. It is done during the coding phase by the developers. Unit Testing helps in finding bugs early in the development process. Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration.

```
import unittest
import pandas as pd
from sklearn.preprocessing import LabelEncoder

class TestEncoding(unittest.TestCase):
    def test_encoding(self):
        df = pd.DataFrame({'label': ['apple', 'banana']})
        df['label'] = LabelEncoder().fit_transform(df['label'])
        self.assertTrue(df['label'].dtype != 'object')

if __name__ == '__main__':
    unittest.main()
```

Fig 8.1: Unit Testing

## 8.2: INTEGRATION TESTING

In Integration Testing different units, modules or components of a software application are tested as a combined entity. Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. In PHDML, the machine learning prediction system is integrated with the user interface streamlit code.

```
import unittest
import pandas as pd
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

class TestPipeline(unittest.TestCase):
    def test_model(self):
        df = pd.DataFrame({'N':[10,20], 'P':[5,10], 'K':[3,6], 'temp':[25,30], 'humidity':[80,70]})
        X, y = df.drop('label',1), df['label']
        X_train, X_test, y_train, y_test = train_test_split(X, y)
        model = CatBoostClassifier(iterations=5, verbose=0)
        model.fit(X_train, y_train)
        acc = accuracy_score(y_test, model.predict(X_test))
        self.assertGreater(acc, 0)

if __name__ == '__main__':
    unittest.main()
```

Fig 8.2: Integration Testing

### 8.3: SYSTEM TESTING

System Testing evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. The testers do not require more knowledge of programming to carry out this testing. System testing checks the interface, decision logic, control flow, recovery procedures and throughput, capacity and timing characteristics of the entire system.

The screenshot shows a web browser window with the title 'Prediction of Heart Disease'. The address bar shows 'localhost:3001'. The page has a sidebar with links: Prediction, Deaths, About, Activities, Data, and Team. The main content area is titled 'Enter the Data' and contains several input fields:

- Age: A slider ranging from 25 to 35, with a current value of 28.
- Sex (Male/Female): Radio buttons for 0 and 1, with 0 selected.
- Client Data Type (1 values): A dropdown menu with 0 selected.
- Resting Blood Pressure (in mmHg units on admission to the hospital): A slider ranging from 120 to 200, with a current value of 130.
- Serum Cholesterol (in mg/dl): A slider ranging from 174 to 260, with a current value of 200.
- Resting Blood Sugar > 120 mg/dl (True = 1, False = 0): Radio buttons for 0 and 1, with 0 selected.
- Resting Electrocardiographic Results (values 0, 1, 2): A dropdown menu with 0 selected.

Fig 8.3: System Testing

### 8.4: ACCEPTANCE TESTING

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not. Testing here focusses on the external behavior of the system, the internal logic of the program is not emphasized. In Acceptance Testing, the system is tested for various inputs. This testing helps the project team to know the further requirements from the users directly as it involves the users for testing.

#### Test Cases used for Acceptance Testing:

Easy access, More accurate results, Awareness, User friendly.

## 9.SAMPLE SOURCE CODE

**models.py:**

```
from django.db import models
import numpy as np
import pickle
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
import json
```

```
# Load pre-trained models
rf_crop_model_path = "rf_crop_model.pkl"
rf_irrigation_model_path = "rf_irrigation_model.pkl"
xgb_crop_model_path = "xgb_crop_model.pkl"
xgb_irrigation_model_path = "xgb_irrigation_model.pkl"
lgbm_crop_model_path = "lgbm_crop_model.pkl"
lgbm_irrigation_model_path = "lgbm_irrigation_model.pkl"
```

```
rf_crop = pickle.load(open(r"C:\Users\User\Music\CROP_ADVICE\FRONTEND\rf_crop_model.pkl", 'rb'))

rf_irrigation = pickle.load(open(r"C:\Users\User\Music\CROP_ADVICE\FRONTEND\rf_irrigation_model.pkl", 'rb'))

xgb_crop = pickle.load(open(r"C:\Users\User\Music\CROP_ADVICE\FRONTEND\xgb_crop_model.pkl", 'rb'))

xgb_irrigation = pickle.load(open(r"C:\Users\User\Music\CROP_ADVICE\FRONTEND\xgb_irrigation_model.pkl", 'rb'))

lgbm_crop = pickle.load(open(r"C:\Users\User\Music\CROP_ADVICE\FRONTEND\lgbm_crop_model.pkl", 'rb'))

lgbm_irrigation = pickle.load(open(r"C:\Users\User\Music\CROP_ADVICE\FRONTEND\lgbm_irrigation_model.pkl", 'rb'))
```

```
@csrf_exempt
def predict_crop(request):
    if request.method == "POST":
        try:
```

```
data = json.loads(request.body)
features = np.array(data['features']).reshape(1, -1)    model_type = data.get('model',
'xgb')

    if model_type == 'rf':
        prediction = rf_crop.predict(features)[0]        elif model_type == 'lgbm':
        prediction = lgbm_crop.predict(features)[0]      else:
        prediction = xgb_crop.predict(features)[0]

    return JsonResponse({"predicted_crop": str(prediction)}) except Exception as
e:
    return JsonResponse({"error": str(e)})    return JsonResponse({"error": "Invalid
request"})
@csrf_exempt
def predict_irrigation(request):    if request.method == "POST":        try:
        data = json.loads(request.body)                crop_type =
np.array([data['crop_type']]).reshape(1, -1)            model_type = data.get('model', 'xgb')
    if model_type == 'rf':
        prediction = rf_irrigation.predict(crop_type)[0]    elif model_type == 'lgbm':
        prediction = lgbm_irrigation.predict(crop_type)[0]    else:
        prediction = xgb_irrigation.predict(crop_type)[0]

    return JsonResponse({"recommended_irrigation": str(prediction)})    except
Exception as e:
    return JsonResponse({"error": str(e)})    return JsonResponse({"error": "Invalid
request"})
```

**apps.py**

```
from django.apps import AppConfig
```

```
class WebappConfig(AppConfig):
```

```
    name = 'webapp'
```

### **tests.py**

```
from django.test import TestCase
```

### **manage.py**

```
#!/usr/bin/env python
```

```
"""Django's command-line utility for administrative tasks.""" import os import sys
```

```
def main():
```

```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'self.settings')    try:
```

```
        from django.core.management import execute_from_command_line    except
```

```
ImportError as exc:
```

```
    raise ImportError(
```

```
        "Couldn't import Django. Are you sure it's installed and "
```

```
        "available on your PYTHONPATH environment variable? Did you "
```

```
        "forget to activate a virtual environment?"
```

```
    ) from exc
```

```
    execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':
```

```
    main()
```

### **settings.py**

```
"""
```

```
Django settings for self project.
```

```
Generated by 'django-admin startproject' using Django 3.0.8.
```

```
For more Information click on the below link
```

<https://docs.djangoproject.com/en/3.0/topics/settings/>

For the full list of settings and their values , see

<https://docs.djangoproject.com/en/3.0/ref/settings/>

"""

```
import os
```

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
```

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```
# Quick-start development settings - unsuitable for production
```

```
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
```

```
SECRET_KEY = '_v1^jm!7=b**0=vvtp8otw#u_=$+u^4sr_vgf)$+7teon2o3(0'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
```

```
]
```



```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT_URLCONF = 'self.urls'  
  
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [os.path.join(BASE_DIR, 'templates')],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',  
                'django.template.context_processors.request',  
                'django.contrib.auth.context_processors.auth',  
                'django.contrib.messages.context_processors.messages',  
            ],  
        },  
    ],  
]  
  
WSGI_APPLICATION = 'self.wsgi.application'  
  
# Database  
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases  
  
DATABASES = {
```

```
'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
}
}

# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/3.0/howto/static-files/
```

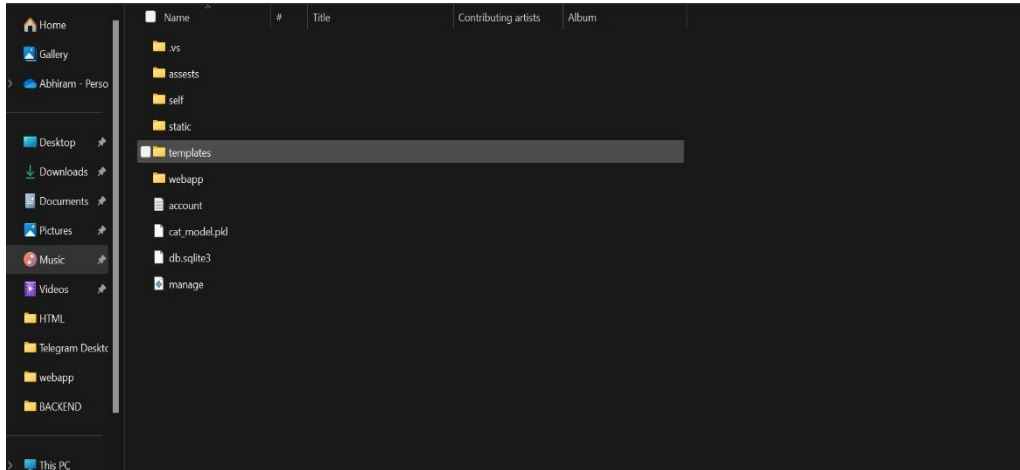
```
STATIC_URL = '/static/'
```

```
STATICFILES_DIRS=[os.path.join(BASE_DIR,'static')]
```

```
STATIC_ROOT=os.path.join(BASE_DIR,'assests')
```

## 9. SCREEN LAYOUT

### Step 1:



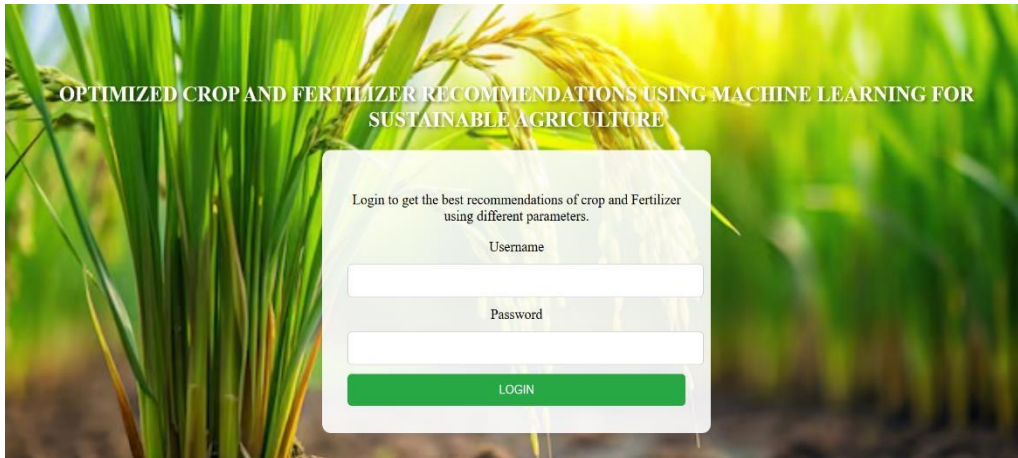
### Step 2:

```
Anaconda Prompt (anaconda) X + -
(base) C:\Users\Abhii>cd C:\Users\Abhii\Music\Crop_Recommendataion\FRONTEND
(base) C:\Users\Abhii\Music\Crop_Recommendataion\FRONTEND>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
April 16, 2025 - 02:00:14
Django version 4.2.20, using settings 'self.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

## Login:



OPTIMIZED CROP AND FERTILIZER RECOMMENDATIONS USING MACHINE LEARNING FOR SUSTAINABLE AGRICULTURE

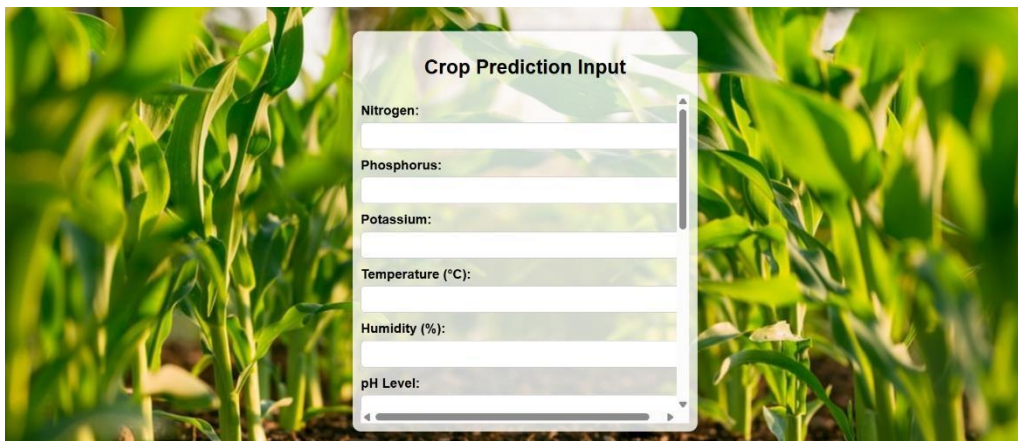
Login to get the best recommendations of crop and Fertilizer using different parameters.

Username

Password

LOGIN

## Credentials:



Crop Prediction Input

Nitrogen:

Phosphorus:

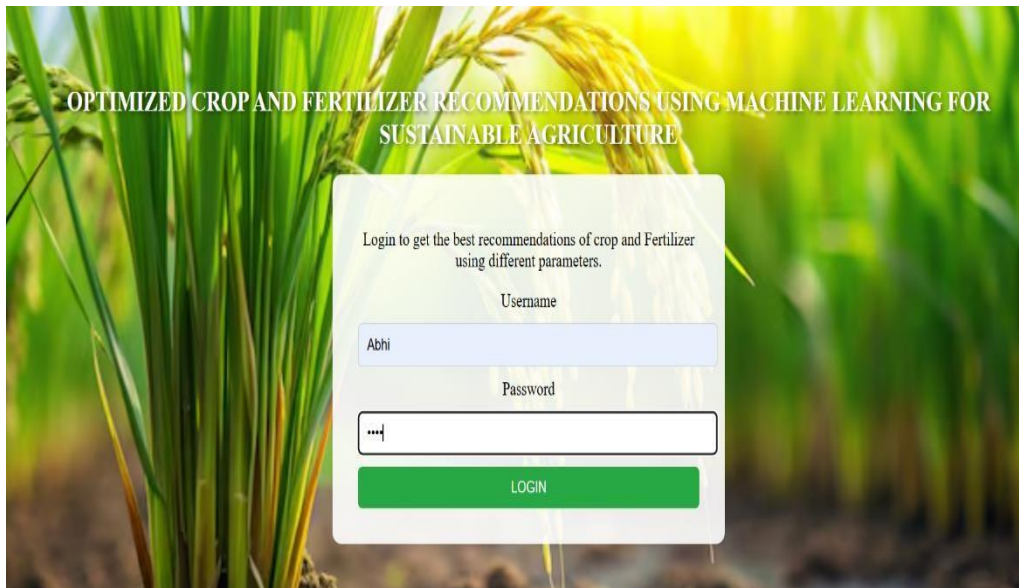
Potassium:

Temperature (°C):

Humidity (%):

pH Level:

## Input Details:



OPTIMIZED CROP AND FERTILIZER RECOMMENDATIONS USING MACHINE LEARNING FOR SUSTAINABLE AGRICULTURE

Login to get the best recommendations of crop and Fertilizer using different parameters.

Username

Abhi

Password

...

LOGIN



Crop Prediction Input

Soil Type:

Sandy

Sunlight Exposure:

Low

Irrigation Frequency:

Low

Crop Density:

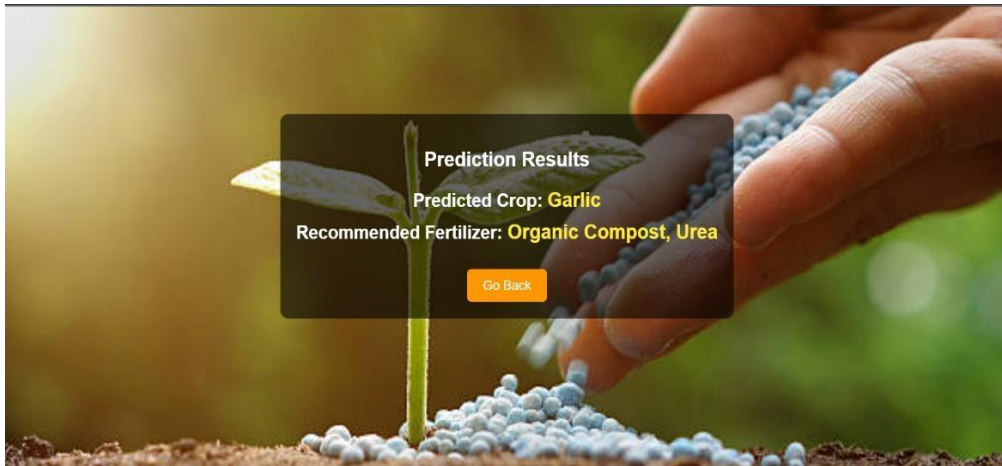
Frost Risk:

Low

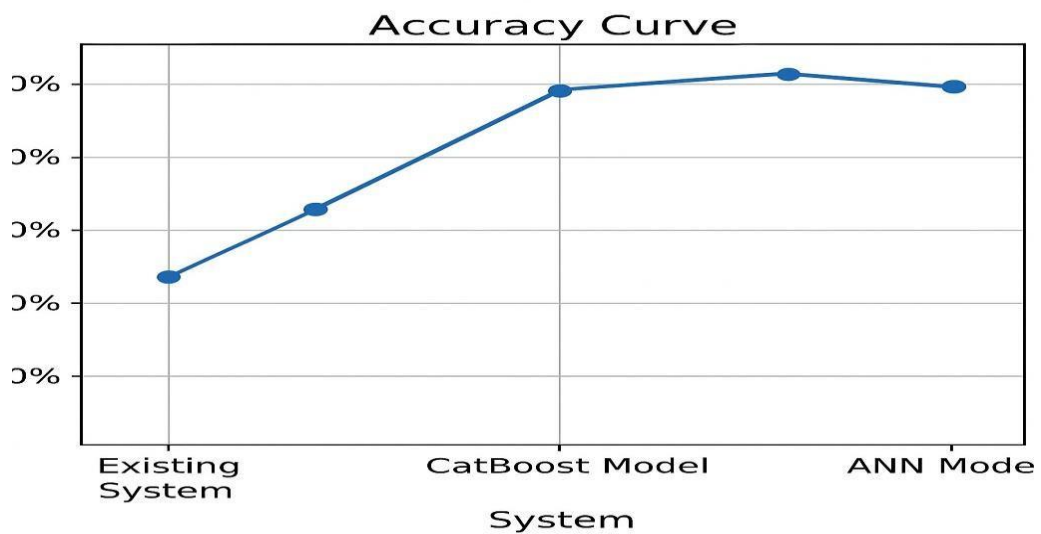
Predict



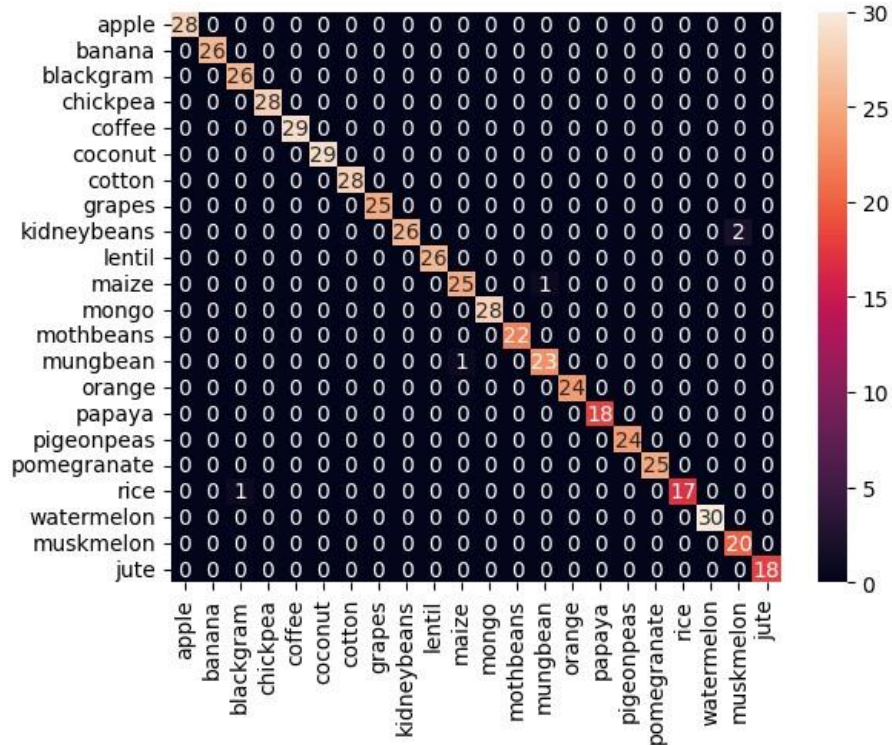
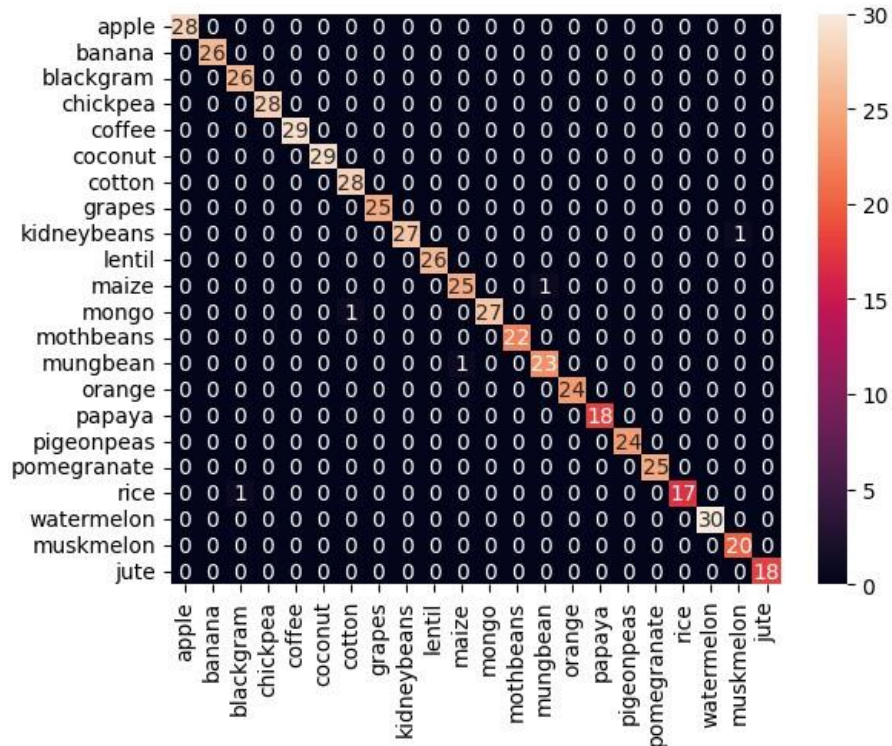
### Predicted output :



### Difference In Accuracy:



## Difference In Confusion Matrix:





## 11. CONCLUSION AND FUTURE ENHANCEMENTS

The project titled “Optimized Crop and Fertilizer Recommendations Using Machine Learning for Sustainable **Agriculture**” presents a transformative approach to modern farming by leveraging data-driven intelligence to enhance decision-making in agriculture. With the ever-growing global population, climate variability, and the depletion of natural resources, there is a pressing need to shift from conventional, intuition-based farming practices to technology-assisted precision agriculture. This project addresses that need by integrating advanced machine learning techniques -namely **CatBoost** and **Artificial Neural Networks (ANN)**—to deliver customized, accurate, and scalable recommendations tailored to soil and environmental conditions.

By analyzing key agricultural inputs such as nitrogen, phosphorus, potassium levels, soil pH, climate patterns, and historical yield data, the system is capable of identifying the most suitable crops for cultivation and determining optimal fertilizer combinations. The models were trained and validated using real-world datasets, achieving strong performance metrics and demonstrating the feasibility of AI-powered agricultural advisory systems. Through the combination of CatBoost’s efficiency in handling categorical and numerical data and ANN’s ability to model non-linear relationships, the system provides holistic recommendations that reflect both statistical accuracy and agricultural relevance.

Furthermore, the system incorporates an intuitive **Django-based web interface** that enables farmers, agronomists, and stakeholders to input their local soil and weather conditions and receive instant, science-backed recommendations. This user-centric design ensures the technology remains accessible and practical, especially for small and medium-scale farmers. The modular design of the system allows for future enhancements such as integration with **IoT-based soil sensors**, **mobile application support**, **satellite weather updates**, and even **blockchain-based traceability** for secure data handling. These enhancements can further increase the system’s adaptability and effectiveness, especially in real-time applications and resource-limited environments.

In conclusion, this project demonstrates that the fusion of machine learning and agriculture can significantly enhance productivity, reduce environmental impact, and promote sustainability. It not only empowers farmers with the tools needed to make informed decisions but also contributes to national goals such as **food security**, **climate resilience**, and **smart village development**. As

agriculture continues to evolve, intelligent systems like the one proposed in this project will play a crucial role in ensuring a more productive and sustainable future for global farming communities.

### Future Enhancement

While the current system for crop and fertilizer recommendation using CatBoost and Artificial Neural Networks (ANN) achieves high accuracy and offers meaningful support to farmers, there is significant scope for enhancement to improve its scalability, intelligence, and real-time usability. Future improvements aim to elevate the system's functionality, increase adoption across rural areas, and support broader goals of smart and sustainable agriculture.

One major area of enhancement involves the **integration of IoT (Internet of Things) devices**, such as soil moisture sensors, weather stations, and nutrient monitoring devices. These real-time sensors can feed live data into the system, allowing it to make dynamic recommendations based on current field conditions rather than relying solely on static datasets. This real-time monitoring can enable predictive alerts for fertilizer deficiency, irrigation needs, or crop stress due to changing weather patterns.

Another advancement involves the development of a **mobile application** for Android and iOS platforms. This mobile interface can empower farmers in remote areas to access personalized recommendations in their local languages, receive seasonal reminders, and view graphical insights about soil health and yield predictions. Offline capabilities can also be added so that the application can function in areas with poor internet connectivity.

**Blockchain technology** can be integrated to secure and verify data integrity, especially in contexts involving government subsidies, farmer cooperatives, or agricultural supply chains. Blockchain can provide transparent tracking of fertilizer usage, crop performance, and yield outcomes, ensuring further, the system can be enhanced to support **multi-crop rotation planning**, where it recommends the best crop sequences across seasons to maintain soil fertility and maximize profits. Expanding the model to include **pest and disease prediction** using image recognition and environmental data is another promising direction.

Finally, deploying the system on **cloud infrastructure** will allow for scalable access, multi-user support, and integration with external APIs such as government weather forecasts or regional agricultural databases.

## 12. BIBLIOGRAPHY

### REFERENCES

1. Kaur, H., & Singh, D. (2021). Machine learning applications in agriculture: A review. *Journal of Agricultural Informatics*, 12(1), 50–59.
2. Singh, S., & Chauhan, N. (2020). Soil fertility prediction using machine learning algorithms. *International Journal of Scientific & Technology Research*, 9(3), 6142–6147.
3. Mallick, R., & Dey, N. (2021). Precision agriculture using AI techniques. *Advances in Intelligent Systems and Computing*, 1174, 309–317.
4. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31.
5. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
6. Jagielska, I., Matthews, C., & Whitfort, T. (2002). An investigation into the application of neural networks, fuzzy logic, genetic algorithms, and rough sets to automated knowledge acquisition for classification problems. *Neurocomputing*, 24(1–3), 37–54.
7. Dhruv, D., & Kumari, M. (2020). Crop recommendation system using machine learning approach. *International Journal of Advanced Science and Technology*, 29(6), 3544–3550.
8. Jain, D., & Jain, A. (2019). Fertilizer recommendation based on soil testing using machine learning. *Procedia Computer Science*, 167, 376–384.
9. Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147, 70–90.
10. Tripathi, R., & Kumar, A. (2022). Smart farming: A machine learning-based approach for crop and fertilizer prediction. *Journal of Ambient Intelligence and Humanized Computing*, 13(3), 1785–1798.
11. Aggarwal, A., & Gupta, R. (2020). Soil nutrient prediction and fertilizer recommendation using machine learning. *International Journal of Engineering Research & Technology*, 9(6), 916–921.

12. Mehdi, S. M., & Aziz, M. (2019). Data preprocessing techniques in agriculture data for machine learning applications. *International Journal of Computer Applications*, 182(39), 1–6.
13. Kumar, V., & Karthikeyan, K. (2021). Sustainable agriculture: An AI-based approach for yield prediction. *Sustainable Computing: Informatics and Systems*, 30, 100569.
14. Patil, S., & Joshi, R. (2021). An ensemble learning model for predicting soil fertility. *Materials Today: Proceedings*, 47(2), 6483–6489.
15. Wani, M. A., & Kale, S. (2020). Crop recommendation system using hybrid model. *IEEE International Conference on Advances in Computing, Communication & Materials (ICACCM)*.
16. Jat, D. S., & Meena, H. R. (2022). Comparative study of CatBoost and Random Forest for crop yield prediction. *International Journal of Agricultural Statistics and Science*, 18(1), 101–108.
17. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning (ICML)*.
18. Tan, M., & Le, Q. V. (2021). EfficientNetV2: Smaller models and faster training. *Proceedings of the International Conference on Machine Learning (ICML)*.
19. FAO (2020). The State of Food and Agriculture. Food and Agriculture Organization of the United Nations. Available: <https://www.fao.org>