UNIVERSITY OF RAJSHAHI

CROP RECOMMENDATION SYSTEM USING MACHINE LEARNING

Project Part-II (CSE 4292)

A Project Paper

Submitted for the partial fulfillment of the requirements

for the degree of B.Sc. in Computer Science and Engineering

Submitted by

Dipa Rani Sarker

Student ID: 2037820121

Supervised by

Md. Hasibul Alam

Lecturer, Dept. of CSE.

Department of Computer Science and Engineering

University of Rajshahi

November, 2025.

CANDIDATES' DECLARATION

I hereby declare that the project presented in this report is the result of my own work, carried out under the supervision of Md. Hasibul Alam, Department of Computer Science and Engineering, TMSS Engineering College, Bogura, Bangladesh. The work was conducted over two final semester courses — CSE 4192: Project (Part I) and CSE 4292: Project (Part II) — in accordance with the curriculum requirements of the Bachelor of Science in Computer Science and Engineering.

It is also declared that neither this project nor any part of it has been submitted elsewhere for the award of any degree, diploma, or other qualification.

_____

Dipa Rani Sarker

ID: 2037820121

Session: 2019- 2020

CERTIFICATION

This project, titled “Crop Recommendation System Using Machine Learning,” submitted by me with the details given below, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering.

Dipa Rani Sarker
ID: 2037820121
Session: 2019- 2020

Md. Hasibul Alam
Lecturer & Supervisor
Department of Computer Science and Engineering,
TMSS Engineering College.

External Examiner
Name:
Designation:
Department of Computer Science and Engineering,
Rajshahi University.

ABSTRACT

Enhancing agricultural productivity is critical for ensuring food security and promoting sustainable development. However, farmers frequently encounter challenges in selecting the most suitable crops for cultivation due to diverse soil characteristics, fluctuating climatic conditions, and limited access to expert agricultural advice. This project proposes the development of a Machine Learning-based Crop Recommendation System designed to assist farmers in making data-driven crop selection decisions. The system leverages key agricultural parameters—such as soil nutrient levels (Nitrogen, Phosphorus, Potassium), temperature, humidity, pH, and rainfall—to analyze environmental suitability. Utilizing a trained classification algorithm, the system provides accurate and region-specific crop recommendations. This approach aims to improve decision-making in agriculture, enhance productivity, and ensure efficient use of natural resources. A user-friendly web application has been built using Flask, making it easy and intuitive for farmers to use. Through this portal, farmers can simply enter information about their soil and local environmental conditions. Based on these inputs, the system instantly provides crop recommendations, allowing farmers to make informed decisions.

Keywords: Machine Learning, Crop recommendation, Correlation, Gaussian Naïve Bayes, Random Forest.

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to the Almighty for granting me the strength to complete the project undertaken as part of the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science & Engineering.. This project could not have been completed without the continuous and invaluable support of several beloved and respected individuals.

I would like to express my heartfelt gratitude to my project supervisor, Md. Hasibul Alam, for his constant guidance, insightful knowledge, and unwavering support throughout every stage of this project. His valuable suggestions, skillful direction, constructive feedback, and continuous encouragement have been instrumental in the successful completion of this work.

I am also deeply grateful to my family for their moral support and to my friends for their helpful feedback and suggestions, which greatly contributed to the improvement of my project as a whole.

Finally, I would like to extend my sincere thanks to TMSS Engineering College for providing me with this wonderful opportunity to undertake and complete my B.Sc. in Computer Science & Engineering project.

CONTENTS

List of Figures

List of Tables

Chapter-1

Introduction

1.1 Project Overview

A machine learning based Crop Recommendation System is an useful tool that can be used by farmers and agricultural authorities finding the best crop cultivation options for a specific set of soil, environmental, weather conditions. Through the application of advanced algorithms and machine learning, data driven decision making is made available to match the regional specifics which support better forethought in agricultural planning and investment. Due to its wide choice of libraries for scientific and machine learning tasks, Python is therefore an ideal language to build such systems in. The application on real-world agricultural related data in this project has shown that high prediction accuracy and practical use are possible. These findings demonstrate the potential of machine learning-based crop recommendation system as a powerful decision support tool for driving modern data driven and resource optimized farming.

This project is all about building a web application that recommends the best crops to grow with the help of machine learning and the Flask framework. It is designed to make crop selection smarter and easier by using data to guide better farming decisions. The system is trained on a reliable dataset from Kaggle and uses key agricultural factors—such as soil nutrients (Nitrogen, Phosphorus, Potassium), temperature, humidity, pH, and rainfall—to determine the most suitable crop for given conditions. Several machine learning models are tested, and the one with the best performance is integrated into an easy-to-use web interface. By bringing together machine learning and modern web technologies, this project aims to give farmers and researchers a simple, fast, and affordable way to find the most suitable crops for their land and conditions.

1.2 Context

Choosing the right crop is crucial for a good harvest, but it’s not always easy to know what will grow best. Using machine learning and web technologies, this system offers a simple, fast way to suggest the most suitable crops based on real soil and weather conditions.

1.3 Motivation

This project was inspired by the idea of using technology to make farming easier and more productive. Choosing the right crop can be difficult, especially with changing weather and different soil conditions, and many farmers don’t always have the information they need. The goal of this system is to provide simple, fast, and accessible crop recommendations through an easy-to-use web application. By combining machine learning with a Flask

application, it helps farmers make smarter decisions and plan their crops more effectively. In addition, the system aims to reduce guesswork in farming and minimize potential losses.

1.4 Problem Description

Given a dataset D containing multiple field profiles $F=\{f1,f2,...,fn\}$, where each profile f_i consists of key soil and environmental features such as Nitrogen, Phosphorus, Potassium, temperature, humidity, pH, and rainfall, the goal is to use data mining and machine learning techniques to predict the most suitable crop for each field.

1.5 Objectives

- The main objective is to develop a system that recommends crops based on input data (e.g., soil nutrients, temperature, humidity, pH etc.).
- Use machine learning models for prediction and better selection.
- Integrate the chosen model into a Flask web app so that it's easy to use and accessible to anyone through a simple web interface.
- Provide a user-friendly web interface for users.

Chapter-2

Literature Review

2.1 Introduction

Studies on crop recommendation systems show that machine learning can guide farmers in picking the right crops by analyzing the soil quality and weather conditions. Using algorithms like KNN, Decision Trees, Random Forests, SVM etc. these systems make it easier to plan for better harvests and use resources wisely. Studies using real agricultural data illustrate that such tools can be very helpful, especially when available through simple web or mobile apps.

2.2 Related Works

Ersin Elbasi et al. (2023), emphasizes the critical role of Machine Learning (ML) in advancing smart farming practices, aiming to maximize both crop yield and farmer profitability. Existing studies primarily employ supervised ML techniques (such as Naïve Bayes, Random Forest, and SVM) to tackle key agricultural challenges, including crop selection, yield forecasting, and early disease detection, using inputs like soil quality and weather patterns. However, the review identified a significant shortcoming in the prior art: a frequent reliance on non-real-world data and inconsistent reporting of model accuracy. To address these gaps, the present study developed an enhanced model using genuine, practical data. The resulting analysis showed that the Bayes Net algorithm delivered the best performance, achieving a remarkably high classification accuracy of 99.59% [1].

Pedina Sasi Kiran et al. (2024) proposes an ML-based Decision Support System (DSS) for crop recommendation, utilizing parameters like NPK, pH, temperature, and rainfall. Comparing algorithms like GNB, SVM, RF, and DT, the Gaussian Naïve Bayes (GNB) classifier achieved the highest performance with a 99% accuracy. The proposed system, accessible via a Streamlit GUI , proved effective, yielding a 20% improvement in crop yield over traditional methods. This success supports the hypothesis that combining ML with soil and environmental parameters significantly increases crop selection accuracy [2].

Dhruvi Gosaiet al. (2024) proposed an intelligent crop recommendation system by testing six different machine learning algorithms: Decision Tree, Naïve Bayes, Support Vector Machine, Logistic Regression, Random Forest, and XGBoost. These models used key soil and weather parameters like Nitrogen (N), Phosphorous (P), Potassium (K), pH value, Humidity, Temperature, and Rainfall to suggest the best crop for a given area. Ultimately, the XGBoost algorithm stood out, achieving the highest accuracy at approximately 99.31%, making it the most reliable method for farmers to choose the right crop, increase their yield, and boost the country's overall profit [4].

Vandana Kale and Badri Narayan Mohapatra (2024) proposed a crop recommendation system suggesting the best crop from 22 types based on soil and climate data. The system tested three machine learning algorithms, with the Random Forest (RF) model proving most accurate at 99.22%, outperforming KNN (97.95%) and SVM (97.85%). This predictive model, which also has the fastest training time, is accessible to farmers through a user-friendly web application built with HTML, CSS, and Flask. By entering their field's parameters, farmers receive an accurate crop suggestion to increase their cultivation and maximize profits [5].

Biplob Dey et al. (2024) tested five ML models to see which is best at recommending 21 crops based on soil nutrients and climate. The main takeaway is to keep farming types separate—Agricultural and Horticultural—for much more accurate predictions. The XGBoost model absolutely crushed the competition, proving itself as the most effective tool among all models tested. With near-perfect 99% accuracy, XGBoost is highly recommended to power a smart, cloud-based crop recommendation system for farmers [6].

Ajay Lokhande Prof. Manish Dixit (2022) in their research shows that using Machine Learning (ML) helps farmers select the best crops by analyzing factors like soil nutrients and weather conditions. Models like Random Forest consistently achieve high accuracy, often above 99%, in recommending the most suitable crop or predicting yield. This integration of data science and agriculture is effectively moving farmers toward precision agriculture, leading to smarter decisions, reduced waste, and better overall harvests [8].

Madhuri Shripathi Rao et al. (2022) compared three popular machine learning methods—K-Nearest Neighbor (KNN), Decision Tree, and Random Forest—to see which was best at predicting the right crop based on soil nutrients and weather conditions. The clear winner was the Random Forest Classifier, which achieved an impressive accuracy of 99.32%. This high accuracy means it's a very reliable tool for helping farmers make smart choices about what to plant, significantly outperforming KNN and Decision Tree. Essentially, Random Forest is the best digital advisor for boosting farming efficiency and improving yields [9].

Farida Siddiqi Prity et al. (2024) didn't waste time on the usual old models. This study really dug deep, testing nine different machine learning approaches to figure out the best way to tell farmers which crops to plant. They fed the system historical data on everything from soil health to weather patterns, and the results were incredible. The powerful Random Forest method blew the others out of the water, achieving an amazing 99.31% accuracy. This breakthrough means we can now give farmers incredibly reliable, personalized advice to boost their harvests [11].

Mahmudul Hasan et al. (2023) understood that the previous research often struggled to accurately predict crop yields, especially in the context of Bangladesh, relying on simpler models that lacked precision. To tackle this, a new ensemble machine learning model called KRR (combination of K-nearest Neighbor, Random Forest and Ridge Regression algorithms) was created to forecast the output of five essential crops like rice and potato. The result analysis showed that KRR was clearly superior to five standard methods,

achieving a remarkable 99% accuracy (R^2) and a very low error rate in predicting Aus rice production. This confirms the KRR model's reliability, making it a robust foundation for building an effective crop recommendation system [12].

Guna Sekhar Sajja et al. (2021) proposed a system using machine learning (ML) to predict the best crops for the farmer's land. They tested three ML algorithms—ID3, Random Forest, and Support Vector Machine (SVM)—on a dataset of 750 crop details. The results showed that SVM was the most accurate tool, proving it's the best digital assistant to help farmers make smarter, more profitable decisions [13].

Chapter-3

Proposed Methodology

3.1 Block Diagram Showing the Overall Methodology

In this study, a machine learning based system is built to suggest the most suitable crops based on soil and environmental conditions. As shown in Fig. 1, the process follows a step-by-step approach—from preparing the data to integrating the final model into a user-friendly Flask web application. The method begins by cleaning the data and handling any missing information. Various machine learning algorithms are then tested to find the one that makes the most accurate predictions, and the model is evaluated using standard methods to ensure it is reliable and effective for practical use.

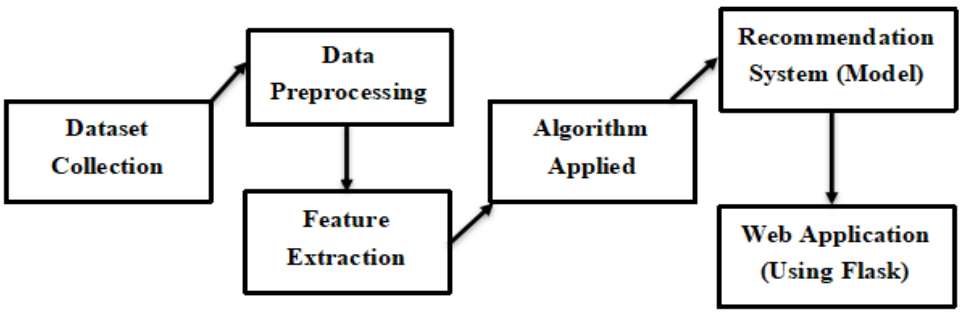


Fig. 1: Overview of the Proposed Methodology.

3.2 Dataset Collection

For this project, a dataset is considered from Kaggle which contains 2200 instances. To facilitate the training and evaluation of the crop recommendation model, the dataset will be divided into two parts:

- **Training Set:** The training set will receive 80 percent of the data, which amounted to 1760 instances.
- **Test Set:** 20 percent of the data will be allocated to the test set, comprising 440 instances.

3.3 Dataset Attributes

The dataset chosen for this project (taken from Kaggle) is reliable and it contains 8 attributes and 2200 instances. The description of the dataset attributes is shown in Table 1.

Table 1. Descriptions of the attributes of the dataset.

Attributes	**Meaning**
N	The ratio of Nitrogen content in soil.
P	The ratio of Phosphorous content in soil.
K	The ratio of Potassium content in soil.
temperature	The temperature in degree Celsius.
humidity	Relative humidity in %.
ph	The pH value of the soil.
rainfall	Rainfall in mm.
label	The target attribute, specifying the recommended crop.

As it can be seen from Table 1, the dataset contains key information about several factors that influence agricultural conditions. Each attribute is described further with value range:

1. **Nitrogen (N):** Indicates the amount of nitrogen present in the soil, ranging from 0 to 140 kg per hectare. Nitrogen is an essential nutrient that supports plant growth and productivity.
2. **Phosphorus (P):** Represents the amount of phosphorus in the soil, ranging from 5 to 145 kg per hectare. Phosphorus plays a vital role in root development and energy transfer within plants.
3. **Potassium (K):** Shows the quantity of potassium in the soil, with values between 5 and 205 kg per hectare. Potassium helps strengthen plants, improves disease resistance, and enhances crop quality.
4. **Temperature:** Reflects the temperature conditions, ranging from 8.83 to 43.68 (recorded in Celsius). This factor directly affects seed germination, growth rate, and overall crop health.
5. **Humidity:** Indicates the level of moisture in the air, ranging from 14.26% to 99.98%. Adequate humidity is essential for maintaining soil moisture and supporting plant transpiration.
6. **pH:** Measures the acidity or alkalinity of the soil, with values between 3.50 and 9.94.
7. **Rainfall:** Represents the amount of rainfall received, measured in millimeters, ranging from 20.21 to 298.56 mm. Rainfall is a crucial factor for irrigation and determines the water availability for crops.
8. **Label:** Indicates a categorical variable that specifies the type of crop recommended for the given environmental conditions. There are 22 unique crops - rice, maize, chickpea, kidneybeans, pigeonpeas, mothbeans, mungbean, blackgram, lentil,

pomegranate, banana, mango, grapes, watermelon, muskmelon, apple, orange, papaya, coconut, cotton, jute, and coffee.

3.4 Data frame of the Dataset

Fig. 2 shows the initial data frame of the dataset that is not processed. It shows that the dataset consists of 8 attributes and 2200 instances. Notably, all attribute values except "label" attribute are already in numerical format.

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

2200 rows × 8 columns

Fig. 2: Initial data frame of the dataset.

3.5 Data Preprocessing

These are the key steps involved in data preprocessing:

- Identifying and removing duplicate records.
- Identifying and handling null and zero (0) values.
- Checking and handling missing values and outliers.
- Encoding categorical feature into a numeric format.

3.5.1 Dataset Information Summary

Fig. 3 shows a quick overview of the structure of the dataset. It provides an overview of - number of rows and columns, column names, data types of each column, number of non-null values, memory usage etc.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

Fig. 3: Dataset information summary.

3.5.2 Descriptive Statistics of the Dataset

Fig. 4 illustrates the summary statistics of the numerical features in the dataset, offering key insights into their distribution, central tendency, and variability—an essential part of Exploratory Data Analysis (EDA).

	count	mean	std	min	25%	50%	75%	max
N	2200.000000	50.551818	36.917334	0.000000	21.000000	37.000000	84.250000	140.000000
P	2200.000000	53.362727	32.985883	5.000000	28.000000	51.000000	68.000000	145.000000
K	2200.000000	48.149091	50.647931	5.000000	20.000000	32.000000	49.000000	205.000000
temperature	2200.000000	25.616244	5.063749	8.825675	22.769375	25.598693	28.561654	43.675493
humidity	2200.000000	71.481779	22.263812	14.258040	60.261953	80.473146	89.948771	99.981876
ph	2200.000000	6.469480	0.773938	3.504752	5.971693	6.425045	6.923643	9.935091
rainfall	2200.000000	103.463655	54.958389	20.211267	64.551686	94.867624	124.267508	298.560117

Fig. 4: Descriptive statistics of the dataset.

3.5.3 Visualizing Missing Data

Fig. 5 presents a bar chart of the dataset's missing data which gives a quick view of columns which have missing values. It confirms that there are no missing values in the dataset. Also the dataset has no null value or duplicate value. So imputing missing or null value is not required.

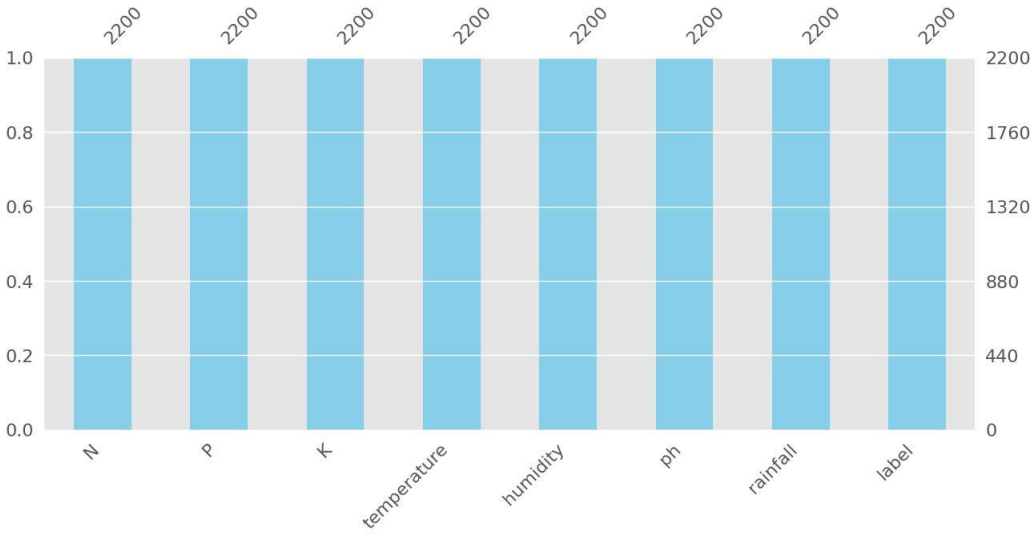
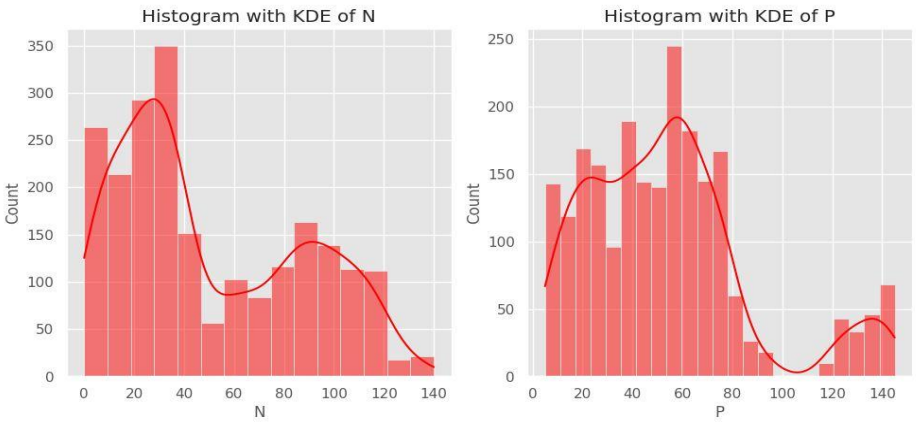


Fig. 5: Bar chart of the missing values in the dataset.

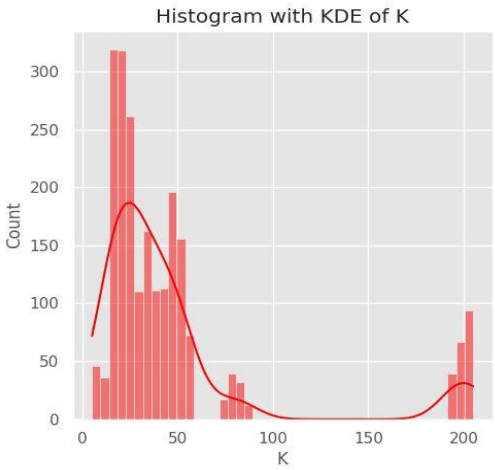
3.5.4 Feature Analysis Using Histogram

A Histogram with KDE (Kernel Density Estimate) is a very common visualization method used in feature analysis. It provides insight into the distribution of a numerical feature by displaying:

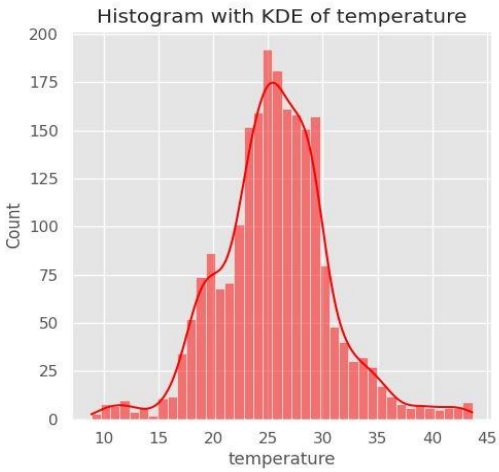
- The count of data points in each bin (histogram).
- A smooth curve representing the estimated probability density of the data (KDE).



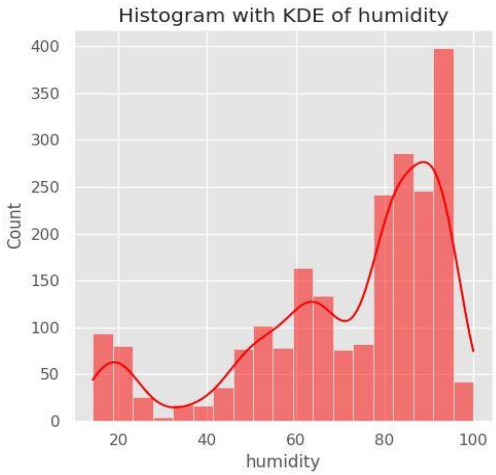
(a) N (b) P



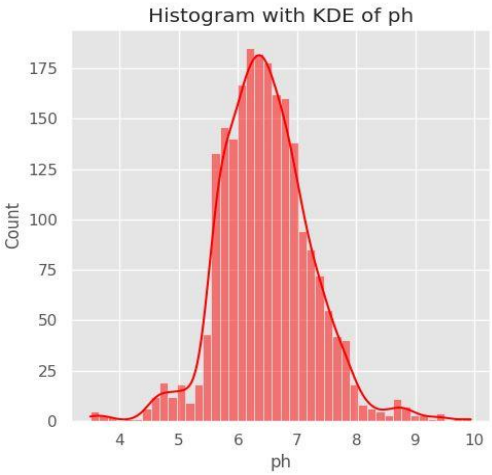
(c) K



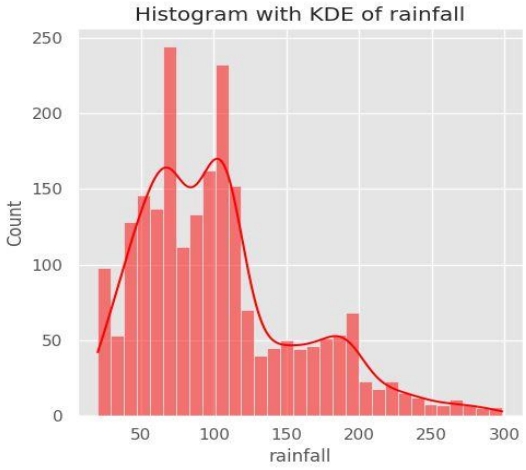
(d) temperature



(e) humidity



(f) ph



(g) rainfall

Fig. 6: Histogram with KDE of (a) N, (b) P, (c) K, (d) temperature, (e) humidity, (f) ph and (g) rainfall.

3.5.5 Encoding “label” Column

The target column “label” is a categorical column (or attribute). This column represents 22 crop names such as rice, maize, banana, coconut etc. These categorical crop names are encoded into numeric values using a dictionary mapping as shown in Table 2. Here, the encoding dictionary matches the exact order of crops in the dataset.

Table 2. Encoding “label” column using dictionary mapping.

CROP NAME	LABEL
Rice	1
Maize	2
Chickpea	3
Kidney beans	4
Pigeon peas	5
Moth beans	6
Mungbean	7
Black gram	8
Lentil	9
Pomegranate	10
Banana	11
Mango	12
Grapes	13
Watermelon	14
Muskmelon	15
Apple	16
Orange	17
Papaya	18
Coconut	19
Cotton	20
Jute	21
Coffee	22

Fig. 7 presents the data frame of the dataset after encoding “label” column to “crop_number” after using the dictionary mapping.

	N	P	K	temperature	humidity	ph	rainfall	crop_number
0	90	42	43	20.879744	82.002744	6.502985	202.935536	1
1	85	58	41	21.770462	80.319644	7.038096	226.655537	1
2	60	55	44	23.004459	82.320763	7.840207	263.964248	1
3	74	35	40	26.491096	80.158363	6.980401	242.864034	1
4	78	42	42	20.130175	81.604873	7.628473	262.717340	1
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	22
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	22
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	22
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	22
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	22

2200 rows × 8 columns

Fig. 7: Data frame of the dataset after encoding “label” column to “crop_number”.

3.5.6 Correlation Matrix

Correlation matrix is a table that displays the pairwise correlation coefficients between several variables or features. It measures the strength and direction of linear relationships, with values ranging from -1 (perfect negative) to +1 (perfect positive). In machine learning, it helps identify feature dependencies, multicollinearity, and redundant variables.

Fig. 8 shows the correlation matrix of the dataset. By examining the correlation matrix, we can choose which features to keep or remove to improve model performance. It's usually calculated using Pearson’s correlation and can be visualized with heatmaps.

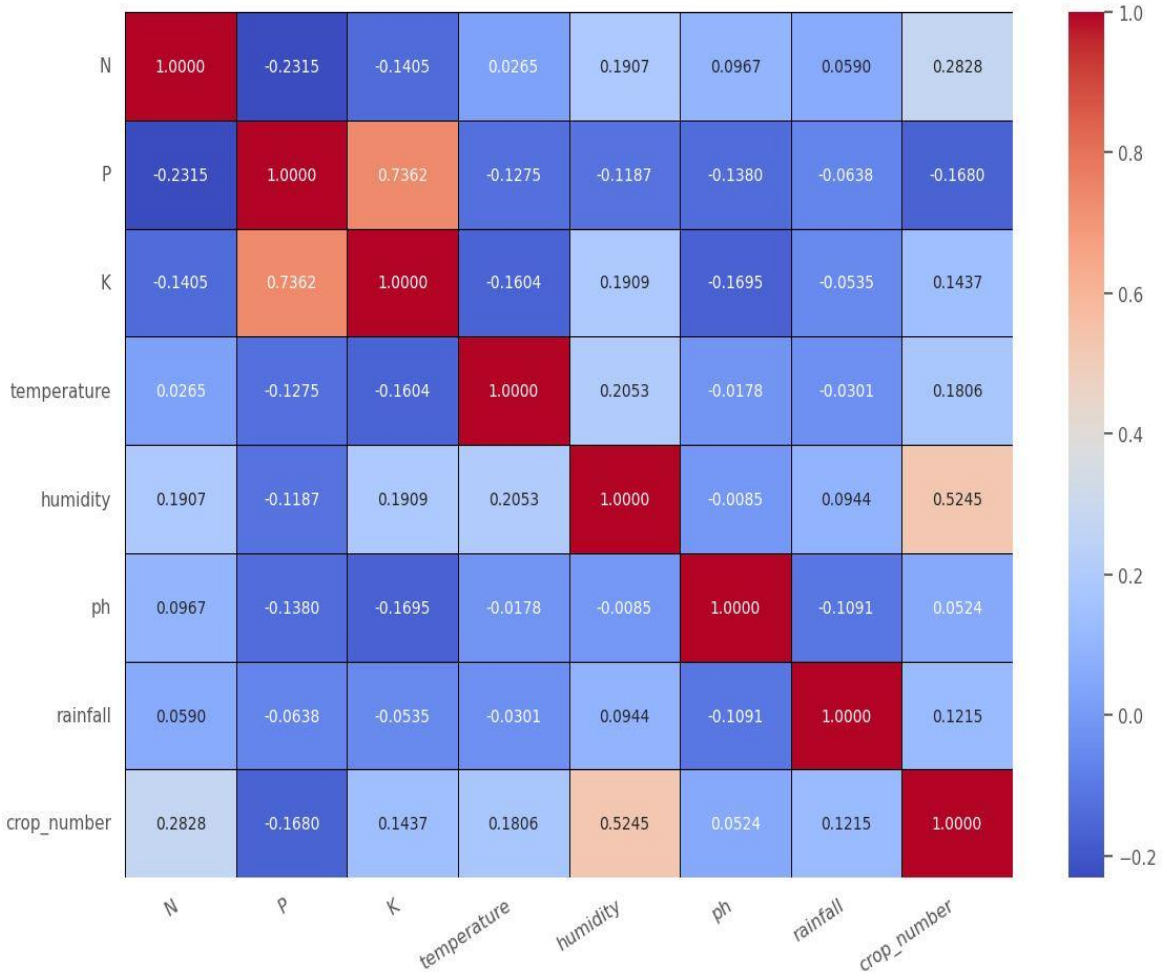


Fig. 8: Correlation matrix.

3.5.7 Pair Plot

A pair plot is a way to visually explore a dataset by looking at all the features together. It creates a grid of small graphs where each feature is compared with every other feature. On these graphs, we can see patterns, relationships, or differences between features. The diagonal usually shows how each feature is distributed on its own.

Selecting the most suitable algorithm for this type of crop dataset can be challenging. To gain insight into the data, a pair plot (Fig. 9) was generated using the Seaborn library. However, the features show a high degree of overlap, making it difficult to distinguish the different crop classes visually.

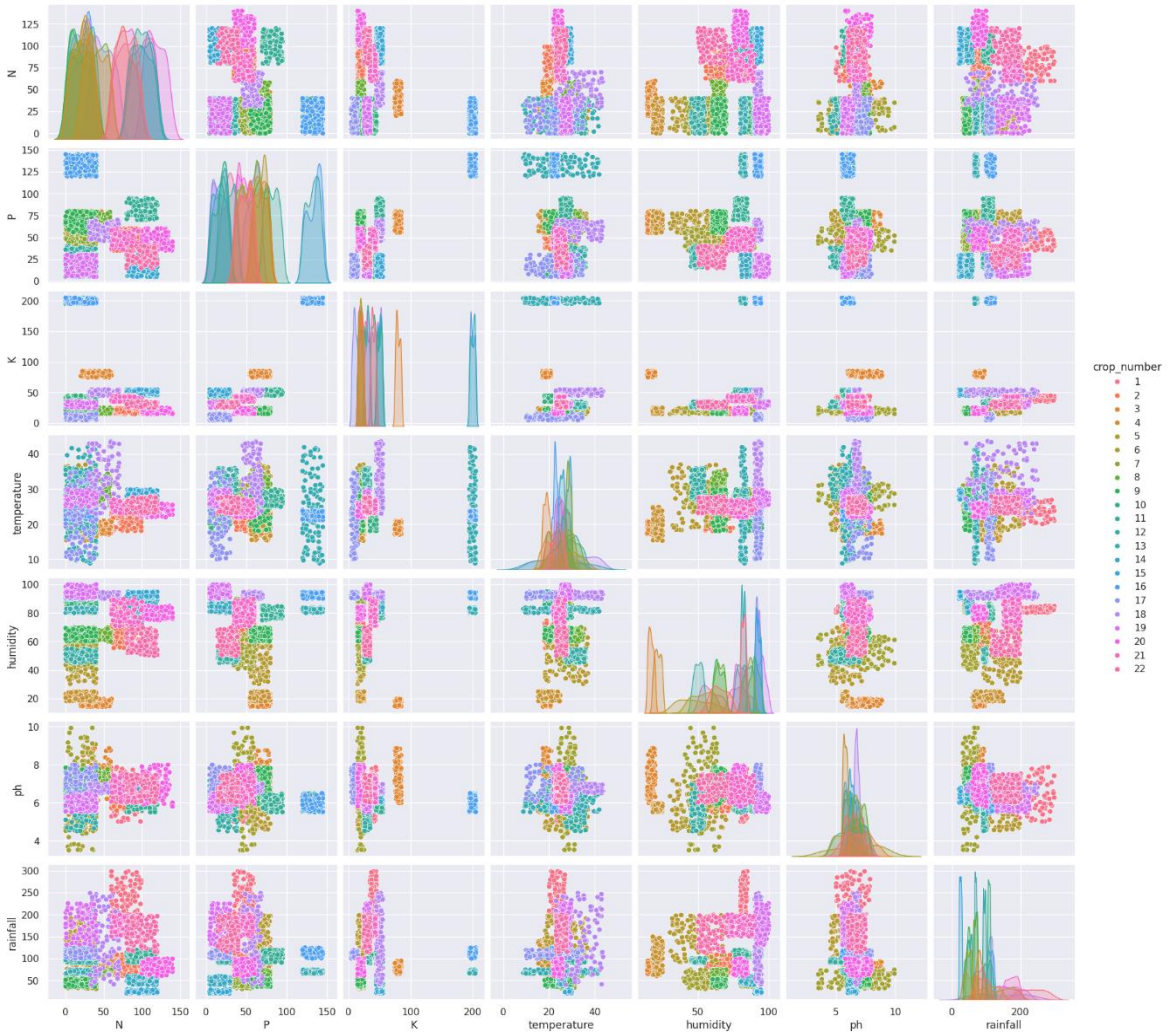


Fig. 9: Pair Plot of the dataset.

3.6 Various Classifier Algorithms for Training and Testing

In this project, six efficient classifiers were employed for model training and testing. These classifiers include Logistic Regression (LR), K-Nearest Neighbors (KNN), Gaussian Naive Bayes (GNB), Support Vector Machine (SVM), Decision Tree (DT) and Random Forest (RF). In this section, a brief description of each algorithm is presented.

3.6.1 Logistic Regression

Logistic Regression is a supervised machine learning algorithm mainly used for classification tasks. Unlike Linear Regression, which predicts continuous numerical values, Logistic Regression predicts the probability that a given input belongs to a particular category. It's especially useful for binary classification problems, where the outcome can be one of two options—such as Yes or No, True or False, or 0 and 1. The algorithm uses a sigmoid function to transform the input values into a probability score between 0 and 1, helping determine the most likely class for the given data.

The sigmoid function (also known as logistic function) is the heart of Logistic Regression. It helps turn any numerical value (no matter how big or small) into a number between 0 and 1, which we can interpret as a probability. In simple terms, the sigmoid function decides how strongly an input belongs to a certain class. The formula looks like this:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Where:

- $\sigma(z)$ is the output of the sigmoid function (the predicted probability).
- z is the input to the function (a linear combination of features, i.e., $z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$)
- e is the base of the natural logarithm.

Here, z represents the combined effect of all input features and their weights.

- If z is a large positive number, the function's output is close to **1**, meaning the model is confident that the input belongs to the positive class.
- If z is a large negative number, the output is close to **0**, meaning it likely belongs to the negative class.
- If z is around **0**, the output is near **0.5**, showing uncertainty between the two classes.

3.6.2 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a straightforward, supervised machine learning algorithm that can be used for both classification and regression, and it's also commonly applied to fill in missing values. The core idea is simple: data points that are close to each other are likely to be similar. So, when we encounter a new or unknown data point, we can make predictions based on the values of its nearest neighbors.

In K-Nearest Neighbors, the "K" represents the number of nearest neighbors to consider when making a prediction. For classification, the algorithm looks at these neighbors and uses a majority vote to decide which class the new data point should belong to. Choosing a larger K often helps make the model more stable and less sensitive to outliers. For example, using K = 3 is usually better than K = 1, because a single neighbor might lead to unpredictable or misleading results. One of the neat things about KNN is that it doesn't make any assumptions about the shape or distribution of the data, which makes it a flexible, non-parametric, and instance-based approach—we are essentially letting the data itself decide the outcome.

K-Nearest Neighbors is often called a "lazy learner" because, unlike other algorithms, it doesn't try to learn patterns from the training data right away. Instead, it simply stores the entire dataset and waits until it needs to make a prediction. When a new data point comes

in, that's when KNN does all the computation, looking at the nearest neighbors to decide the outcome.

3.6.3 Gaussian Naïve Bayes

Gaussian Naive Bayes is a version of the Naive Bayes algorithm designed to work with continuous data, assuming that the features follow a Gaussian (normal) distribution. The "naive" part comes from the assumption that all features are independent of each other, which may not always be true but makes the calculations much simpler. This simplicity makes the model fast, efficient, and easy to implement. It's also popular because it often performs well even on smaller datasets and produces results that are straightforward to interpret.

$$P(x_i|y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where:

- x_i is the value of the feature.
- μ is the mean of this feature for the class y_k.
- σ is the standard deviation of this feature for the class y_k.
- π is a constant (≈ 3.14159).
- e is the base of the natural logarithm, used in the exponential function.

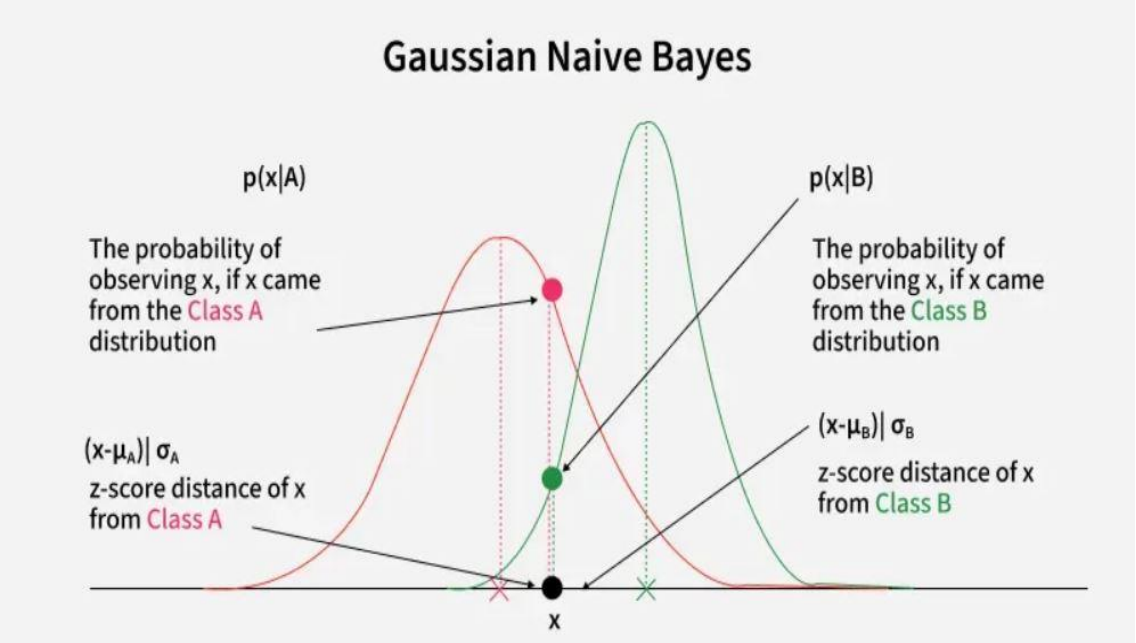


Fig. 10: Gaussian Naïve Bayes.

Gaussian Naive Bayes works really well with continuous data because it assumes that each feature follows a Gaussian, or normal, distribution. When this assumption is reasonably accurate, the algorithm can make very reliable predictions. For instance, in applications like spam detection, medical diagnosis, or predicting house prices, features such as age, income, or height often follow a normal distribution. In these cases, Gaussian Naive Bayes can efficiently and accurately classify or predict outcomes.

3.6.4 Support Vector Machine (SVM)

Support Vector Machine, or SVM, is a type of machine learning algorithm that helps a computer classify things or make predictions. Support Vector Machines (SVM) work by finding the best possible line or decision boundary that separates different classes in an n-dimensional space. This makes it easy to classify new data points by seeing which side of the boundary they fall on.

Key Components of SVM:

1. **Hyperplane:** This is the boundary that the SVM draws to separate different classes in the feature space.
2. **Support Vectors:** These are the key data points that sit closest to the hyperplane. They essentially "hold up" the boundary, influencing exactly where it's placed.
3. **Margins:** The margin is the distance between the hyperplane and the support vectors. A wider margin usually means the model can generalize better to new, unseen data.

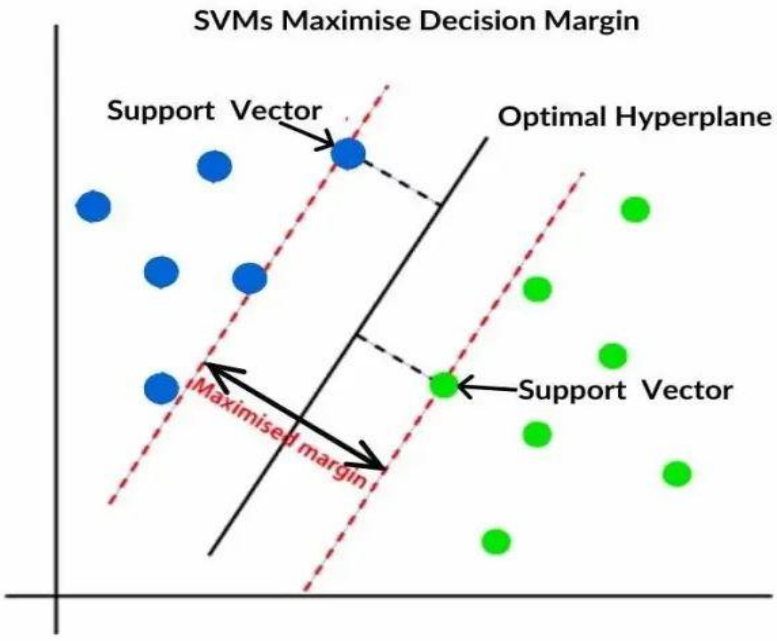


Fig. 11: Support Vector Machine (SVM).

The ideal boundary is called a hyperplane, and its equation is:

$$w \cdot x + b = 0$$

Here's how it works:

- *If* $w \cdot x + b > 0$, the data point is classified as positive.
- *If* $w \cdot x + b < 0$, the data point is classified as negative.

In simple terms, the hyperplane acts like a dividing line, and the SVM uses it to decide which class a point belongs to based on which side it lands on.

3.6.5 Decision Tree

A decision tree is a type of supervised learning algorithm that can be used for both classification and regression tasks. It has a tree-like structure, starting from a root node and branching out through internal nodes, eventually ending at leaf nodes. One of the biggest advantages of a decision tree is that it's easy to visualize and understand, much like a flowchart. It is made up of the following key components:

- **Root Node:** This is the topmost node, representing the entire dataset. From here, the tree begins to branch out.
- **Decision Node:** These are internal nodes where the dataset is split based on certain conditions. Arrows flow into and out of these nodes to show the decision paths.
- **Leaf Node:** Located at the bottom of the tree, leaf nodes represent the final outcome or decision. They do not have any outgoing arrows.
- **Branches:** These are the arrows that connect different nodes, showing the flow of decisions based on feature values.

Fig. 9 presents a simplified diagram of a decision tree that includes all the key components – Root Node, Decision Node, Leaf Node and Branches.

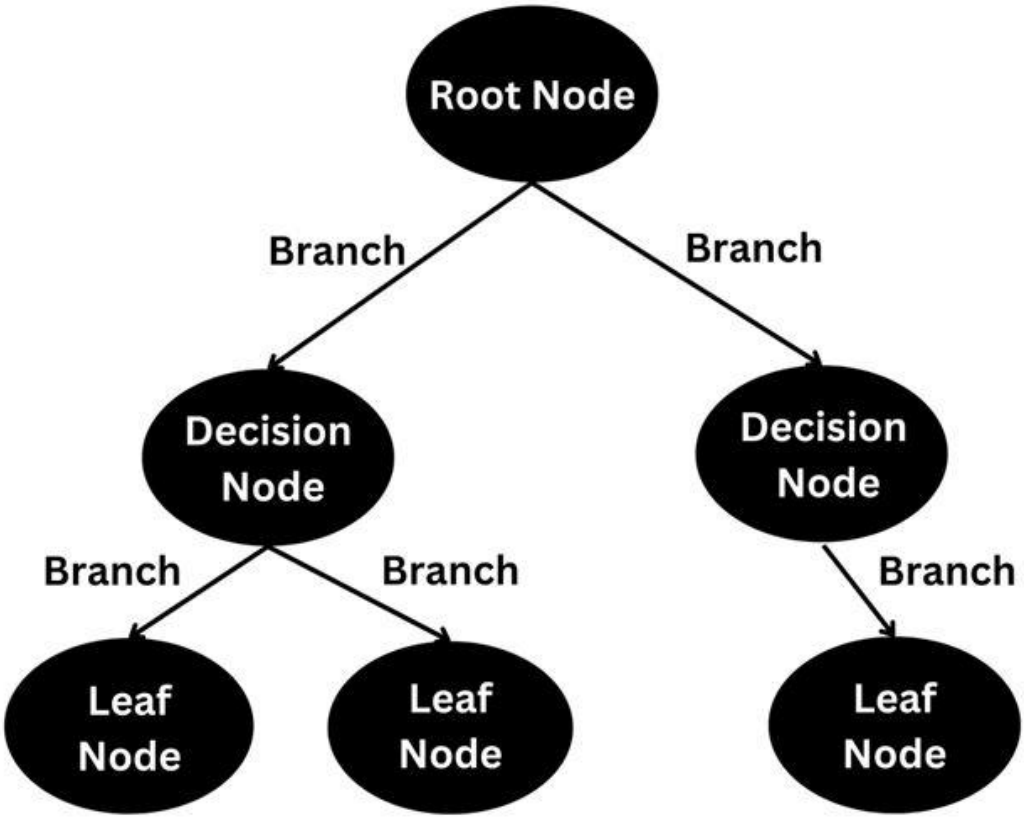


Fig. 12: A simple block diagram of Decision Tree.

3.6.6 Random Forest

Random Forest is a supervised machine learning algorithm used for both classification and regression tasks. It is an ensemble method, which means it combines multiple decision trees to make more accurate predictions.

The algorithm works by building many decision trees independently, with each tree trained on a random subset of the data, a process known as bootstrapping. At each split in a tree, a random subset of features is considered to determine the best split, ensuring that the trees are diverse and not overly similar.

Once all trees are built, the forest aggregates their predictions: for classification, it uses majority voting, and for regression, it takes the average of all tree outputs. Fig. 10 illustrates a simplified diagram of Random Forest classifier.

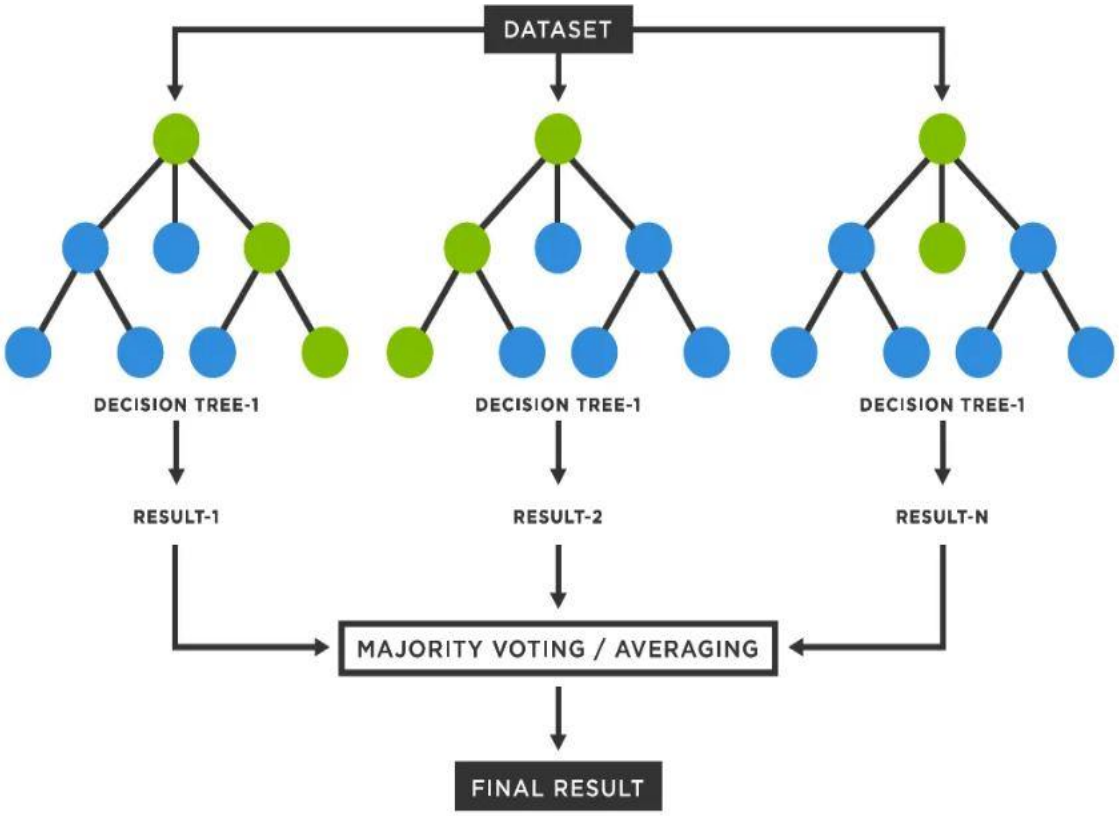


Fig. 13: A simplified diagram of Random Forest Classifier.

This approach helps to reduce overfitting compared to a single decision tree. Random Forest is capable of handling large, high-dimensional datasets and is robust to noise and missing values. Additionally, it provides feature importance scores, highlighting which variables are most influential. Its combination of accuracy, flexibility, and interpretability makes Random Forest one of the most widely used and reliable algorithms in machine learning.

Chapter-4

Model Training and Testing

4.1 Performance Evaluation of Various Measures

Because machine learning models can behave very differently depending on the task and data, finding the best one isn't always straightforward. That's why it's important to have reliable tools to measure how well each model performs. Common evaluation metrics include accuracy, precision, recall, and the F1-score—these are key indicators of a model's overall quality. All of these metrics are derived from the confusion matrix, which provides the essential data for performance assessment.

Confusion Matrix

The confusion matrix itself is not a performance metric, but rather a foundation for calculating multiple performance metrics.

- It is a table that summarizes the performance of a classification model.
- It shows the counts of true and false predictions for each class.
- The matrix has four components:
    - **TP** (True Positive): Correctly predicted positive cases
    - **TN** (True Negative): Correctly predicted negative cases
    - **FP** (False Positive): Incorrectly predicted as positive
    - **FN** (False Negative): Incorrectly predicted as negative

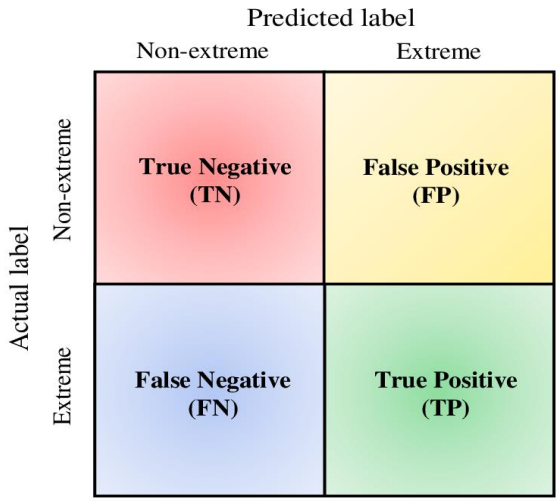


Fig. 14: A standard block diagram of Confusion Matrix.

The following are a few widely used performance metrics:

1. Accuracy

- The proportion of correct predictions out of all predictions made.
- Formula: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

2. Precision

- The ratio of correctly predicted positive observations to total predicted positives.
- Formula: $Precision = \frac{TP}{TP+FP}$

3. Recall (Sensitivity / True Positive Rate)

- The proportion of actual positive cases that were correctly identified.
- Formula: $Recall = \frac{TP}{TP+FN}$

4. F1-Score

- The harmonic mean of precision and recall.
- Formula: $F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

4.2 Performance Analysis

Following data preprocessing each model was trained on the training set, and its performance was evaluated on testing data. This approach allowed for the assessment of potential overfitting or underfitting of the models without applying hyperparameter tuning. For model training, all attributes were used.

4.2.1 Performance Analysis of Logistic Regression Model

Fig. 15 presents the confusion matrix of the Logistic Regression model. The key performance metrics of the Logistic Regression model were evaluated using the test set of the dataset as shown in Table 3.

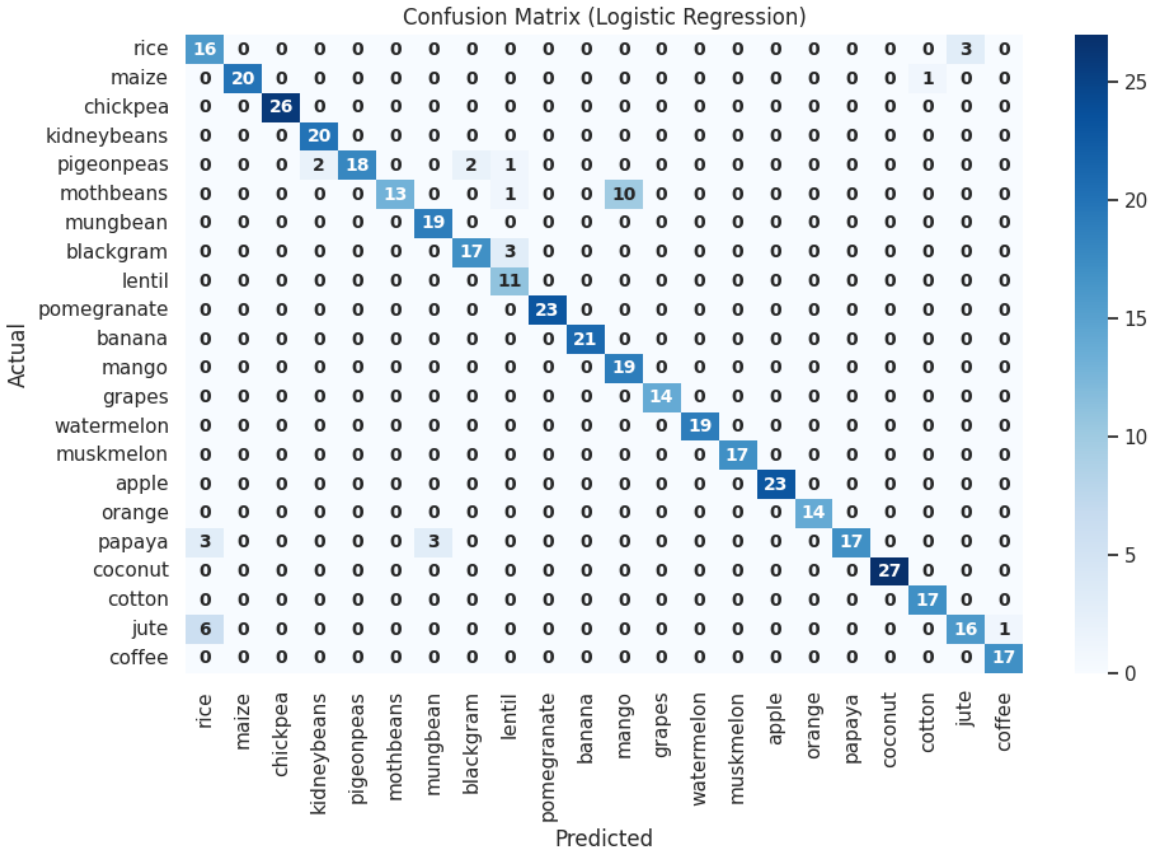


Fig. 15: Confusion matrix of Logistic Regression model.

The model achieved a training accuracy of 94.77% and a test accuracy of 91.82%. Its precision of 91.82% means it's quite accurate when predicting positive cases, while the recall of 91.82% indicates it effectively identifies most actual instances. The F1-score of 91.82% highlights a well-balanced performance between precision and recall.

Table 3: Performance metrics of Logistic Regression model.

Performance Metrics	**Score (%)**
Train Accuracy	94.77
Test Accuracy	91.82
Precision	91.82
Recall	91.82
F1-Score	91.82

4.2.2 Performance Analysis of K-Nearest Neighbors (KNN) Model

Fig. 16 presents the confusion matrix of the K-Nearest Neighbors (KNN) model. The key performance metrics of this model were evaluated using the test set of the dataset as shown in Table 4.

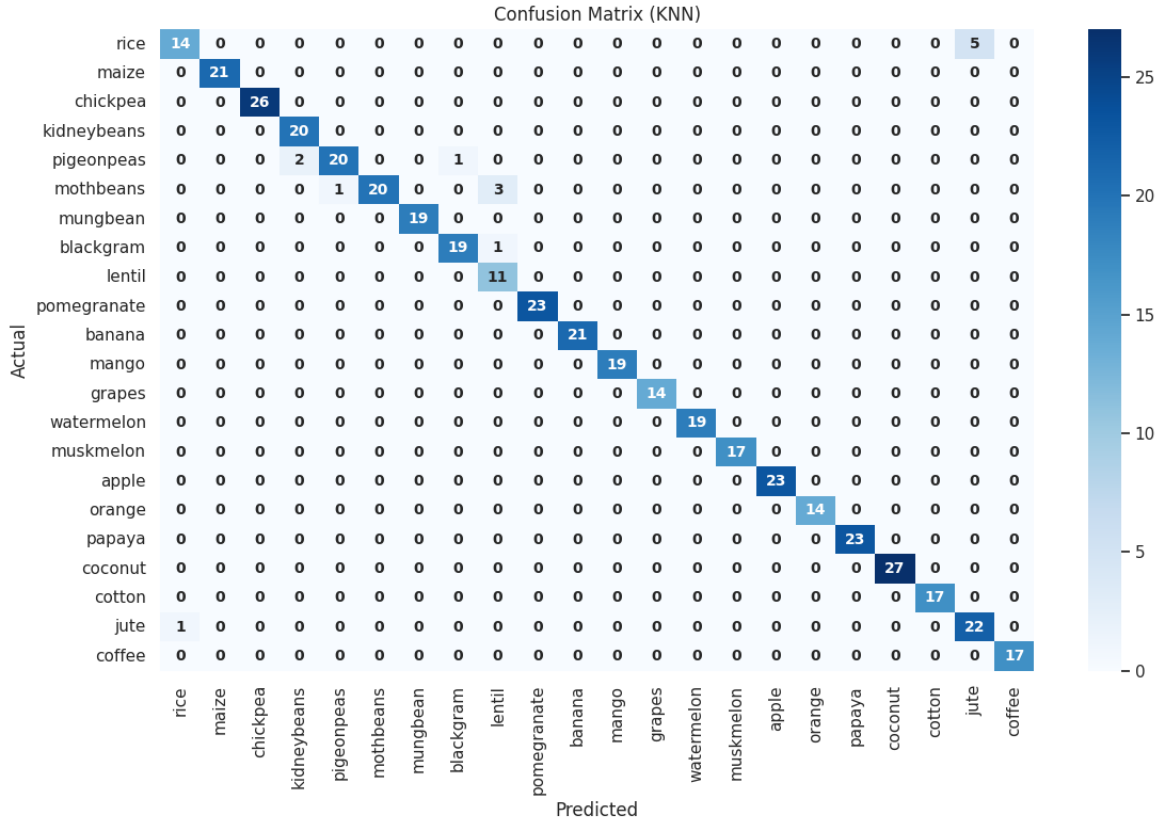


Fig. 16: Confusion matrix of KNN model.

The model achieved a training accuracy of 99.09% and a test accuracy of 96.82%. Its precision of 96.82% means it's quite accurate when predicting positive cases, while the

recall of 96.82% indicates it effectively identifies most actual instances. The F1-score of 96.82% highlights a well-balanced performance between precision and recall.

Table 4: Performance metrics of KNN model.

Performance Metrics	**Score (%)**
Train Accuracy	99.09
Test Accuracy	96.82
Precision	96.82
Recall	96.82
F1-Score	96.82

4.2.3 Performance Analysis of Gaussian Naïve Bayes Model

Fig. 17 presents the confusion matrix of the Gaussian Naïve Bayes (GNB) model. The key performance metrics of this model were evaluated using the test set of the dataset as shown in Table 5.

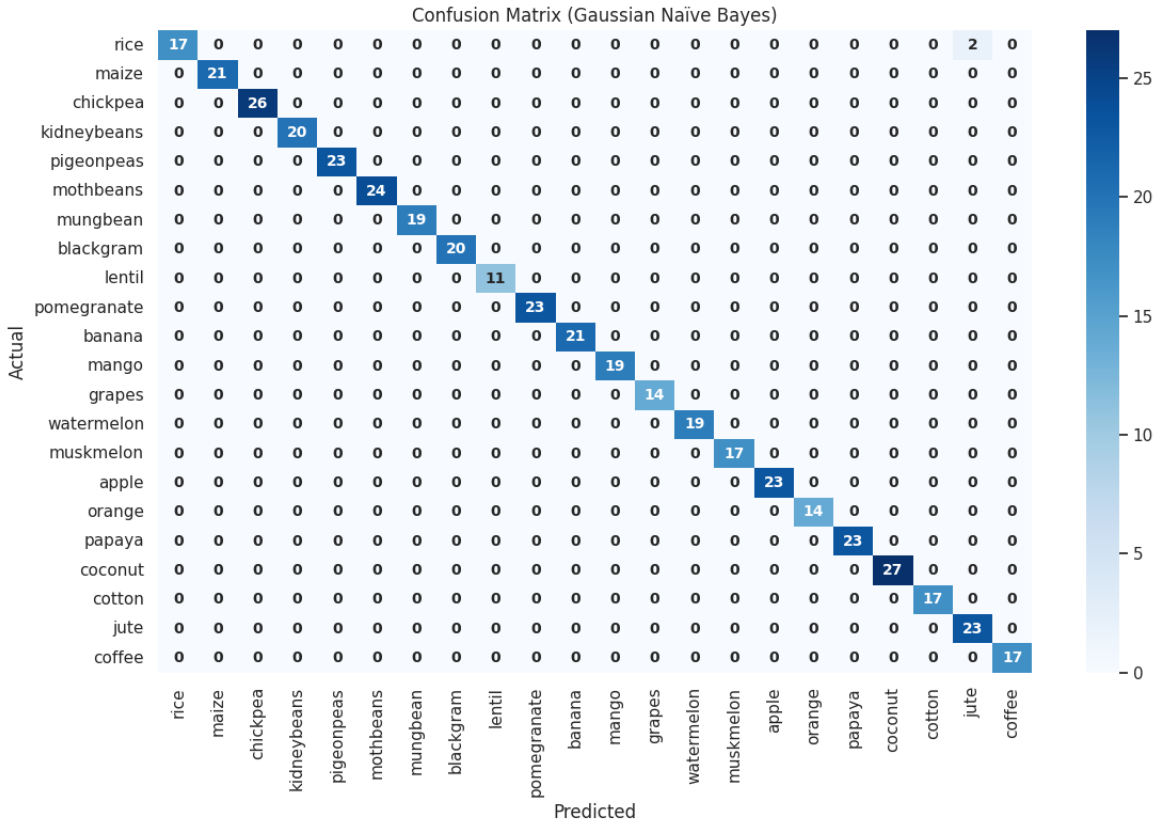


Fig. 17: Confusion matrix of Gaussian Naïve Bayes model.

The model achieved a training accuracy of 99.49% and a test accuracy of 99.55%. Its precision of 99.55% means it's quite accurate when predicting positive cases, while the recall of 99.55% indicates it effectively identifies most actual instances. The F1-score of 99.55% highlights a well-balanced performance between precision and recall.

Table 5: Performance metrics of Gaussian Naïve Bayes model.

Performance Metrics	**Score (%)**
Train Accuracy	99.49
Test Accuracy	99.55
Precision	99.55
Recall	99.55
F1-Score	99.55

4.2.4 Performance Analysis of Support Vector Machine (SVM) Model

Fig. 18 presents the confusion matrix of the Support Vector Machine (SVM) model. The key performance metrics of this model were evaluated using the test set of the dataset as shown in Table 6.

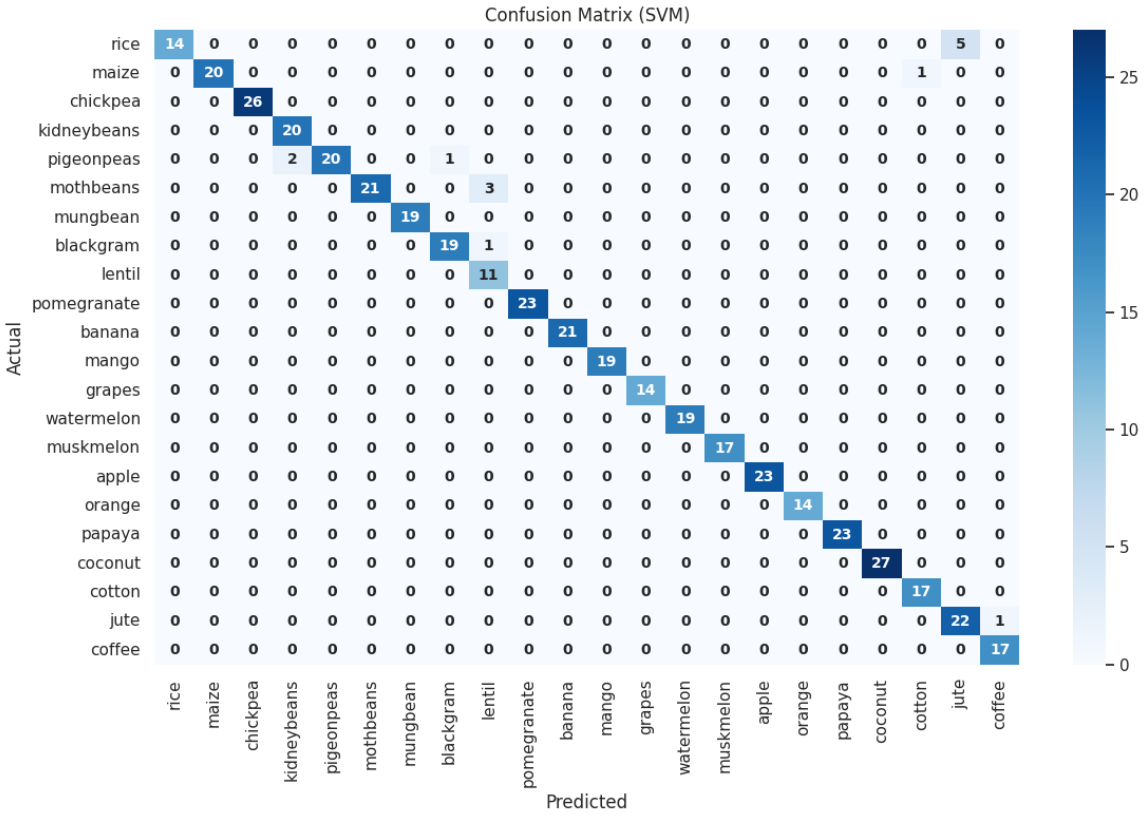


Fig. 18: Confusion matrix of Support Vector Machine (SVM) model.

The model achieved a training accuracy of 98.52% and a test accuracy of 96.82%. Its precision of 96.82% means it's quite accurate when predicting positive cases, while the recall of 96.82% indicates it effectively identifies most actual instances. The F1-score of 96.82% highlights a well-balanced performance between precision and recall.

Table 6: Performance metrics of SVM model.

Performance Metrics	**Score (%)**
Train Accuracy	98.52
Test Accuracy	96.82
Precision	96.82
Recall	96.82
F1-Score	96.82

4.2.5 Performance Analysis of Decision Tree Model

Fig. 19 presents the confusion matrix of the Decision Tree model. The key performance metrics of this model were evaluated using the test set of the dataset as shown in Table 7.

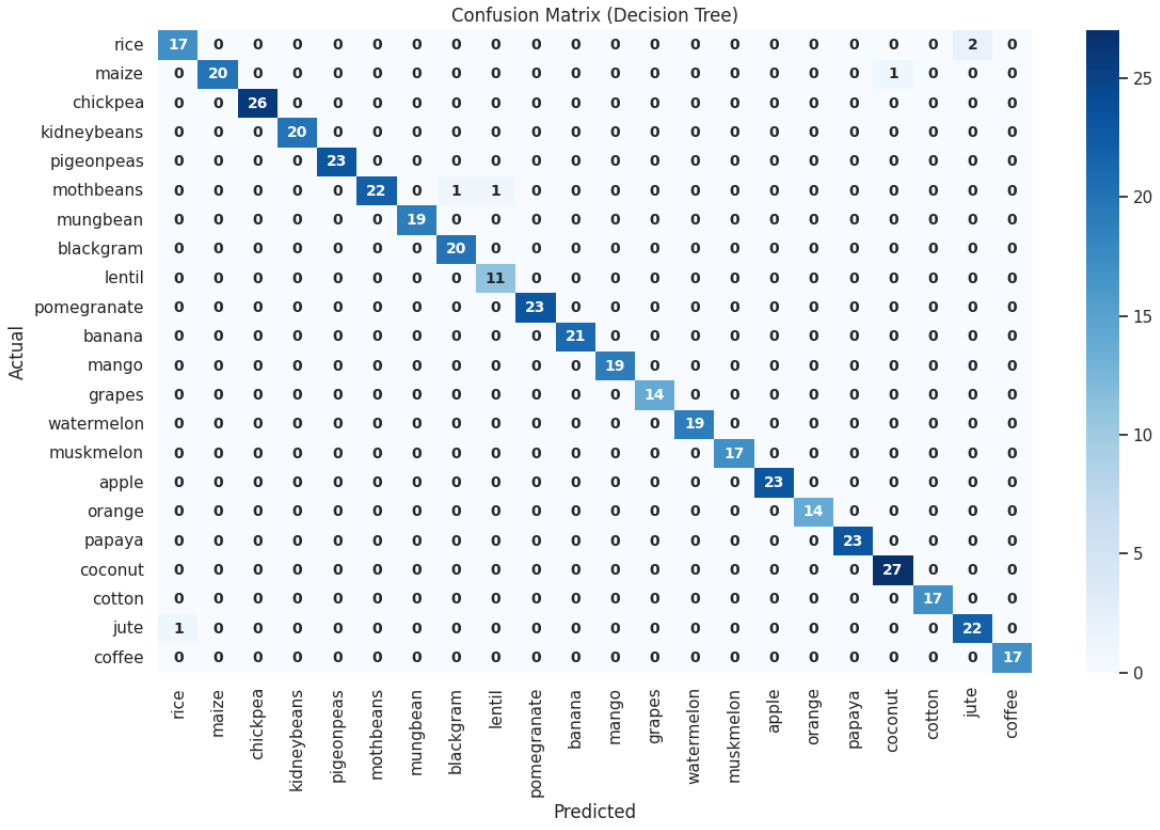


Fig. 19: Confusion matrix of Decision Tree model.

The model achieved a training accuracy of 100.00% and a test accuracy of 98.64%. Its precision of 98.64% means it's quite accurate when predicting positive cases, while the recall of 98.64% indicates it effectively identifies most actual instances. The F1-score of 98.64% highlights a well-balanced performance between precision and recall.

Table 7: Performance metrics of Decision Tree model.

Performance Metrics	**Score (%)**
Train Accuracy	100.00
Test Accuracy	98.64
Precision	98.64
Recall	98.64
F1-Score	98.64

4.2.6 Performance Analysis of Random Forest Model

Fig. 20 presents the confusion matrix of the Decision Tree model. The key performance metrics of this model were evaluated using the test set of the dataset as shown in Table 8.

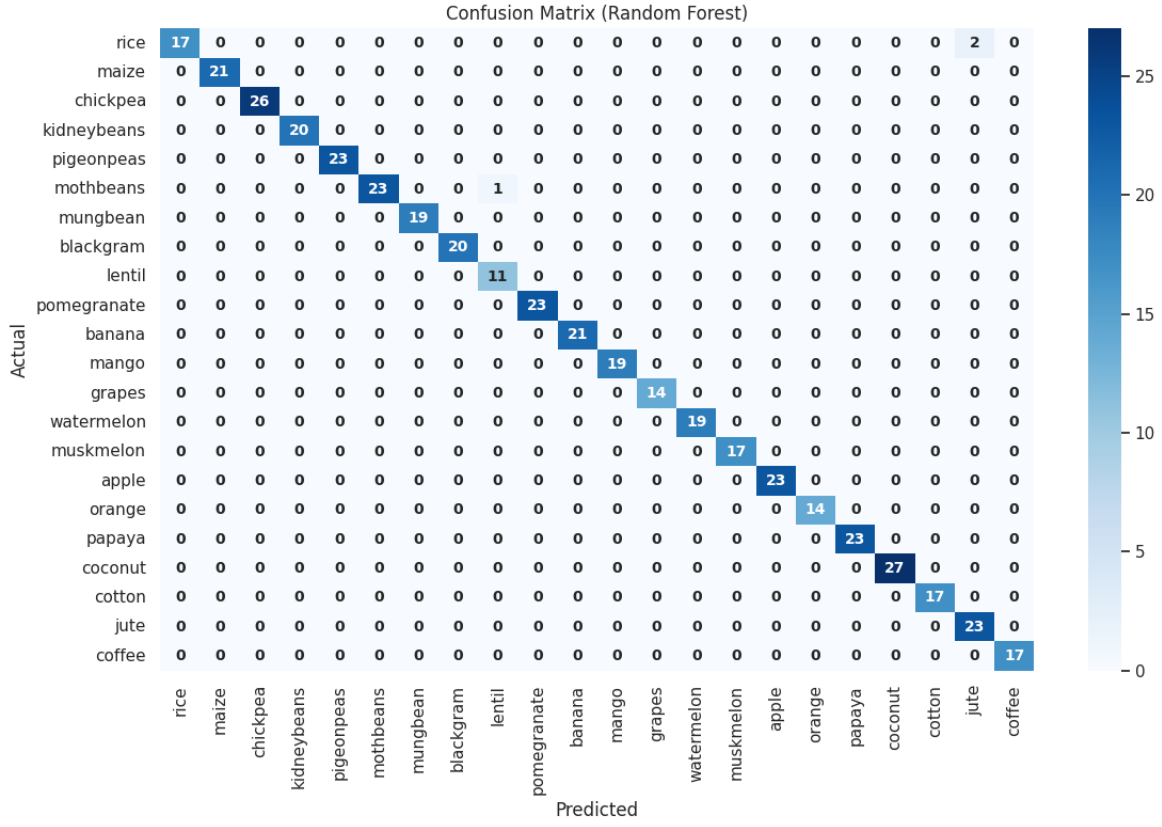


Fig. 20: Confusion matrix of Random Forest model.

The model achieved a training accuracy of 100.00% and a test accuracy of 99.32%. Its precision of 99.37% means it's quite accurate when predicting positive cases, while the recall of 99.32% indicates it effectively identifies most actual instances. The F1-score of 99.32% highlights a well-balanced performance between precision and recall.

Table 8: Performance metrics of Random Forest model.

Performance Metrics	**Score (%)**
Train Accuracy	100.00
Test Accuracy	99.32
Precision	99.37
Recall	99.32
F1-Score	99.32

4.3 Model Accuracy Comparison

To evaluate the performance of different machine learning models, both training and testing accuracies were examined. Comparing these results provides insights into how well each model can generalize to unseen data and helps identify issues like underfitting or overfitting.

Table 9 presents the comparison between train and test accuracies for different classification models applied to the dataset in the descending order of test accuracy.

Table 9: Accuracy (%) comparison on train and test sets.

Model	Train Accuracy	Test Accuracy
GNB	99.488636	99.545455
RF	100.000000	99.318182
DT	100.000000	98.636364
KNN	99.090909	96.818182
SVM	98.522727	96.818182
LR	94.772727	91.818182

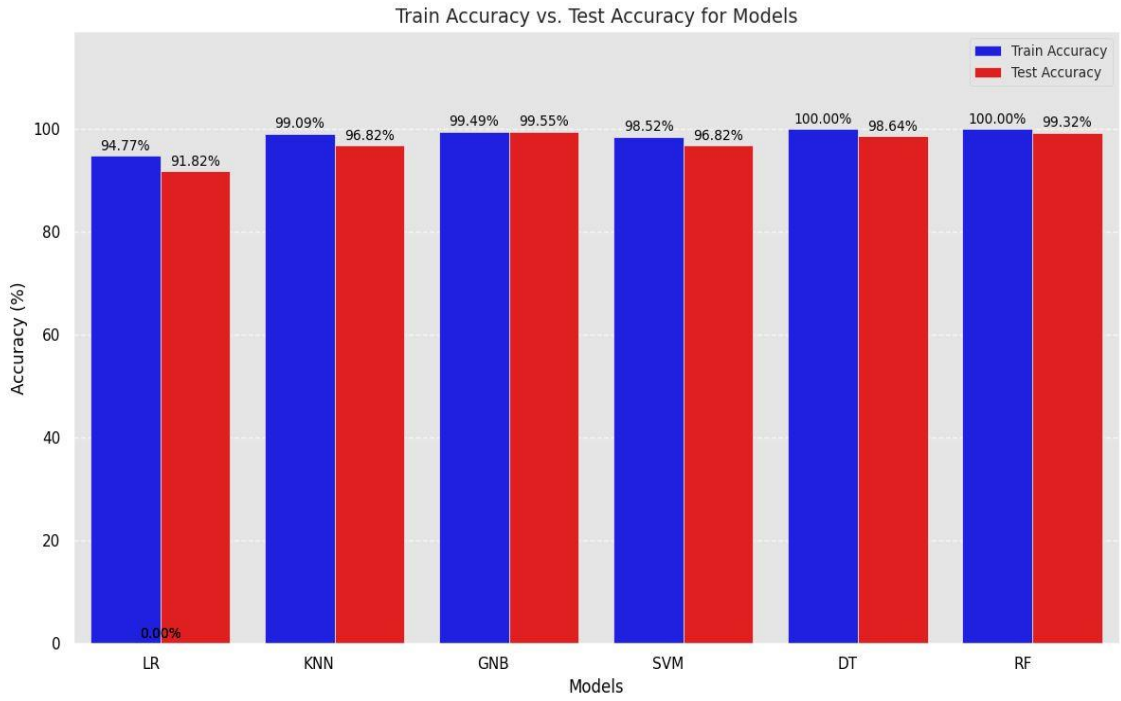


Fig. 21: Accuracy (%) comparison on train and test sets in bar plot.

Similarly, Fig. 21 presents a bar plot illustrating the comparison between training and testing accuracies for various classification models applied to the dataset. Both the Decision Tree (DT) and Random Forest (RF) model achieved perfect training accuracy of 100%. But Gaussian Naïve Bayes model achieved the highest test accuracy of 99.55%. In contrast, Logistic Regression (LR) model achieved both the lowest training and test accuracy of 94.77% and 91.82% respectively.

Chapter-5

Web Application Using Flask

5.1 Simplified Project Block Diagram (After Flask Integration)

Fig. 22 illustrates a simplified block diagram of the project after integrating the Flask framework. It shows how the web interface, server backend, machine learning model, and output display work together, creating a unified and efficient system for crop recommendation.

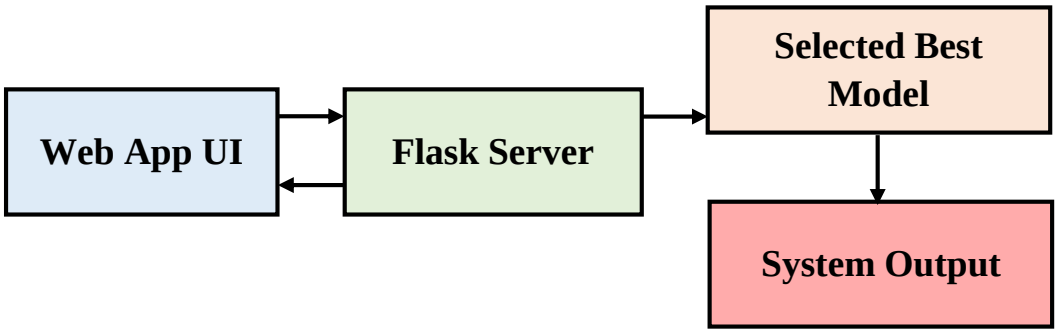


Fig. 22: Simplified block diagram of the project (after flask integration).

The architecture is composed of four main components, described below:

1. **Web App UI:**

   The user interacts with the system through a web page, which provides a simple and user-friendly platform. Here, users can easily enter their information and quickly receive predictions about suitable crops for cultivation.

2. **Flask Server:**

   The Flask server acts as the backbone of the system, connecting the web interface with the machine learning models. It takes care of processing user inputs, handling requests, and running the prediction logic. Being lightweight and flexible, Flask ensures smooth communication and efficient data flow across the application.

3. **Selected Best Model:**

   At the heart of the system is the Gaussian Naïve Bayes model, which achieved the highest test accuracy of 99.55% in recommending the most suitable crop based on different input features.

4. **System Output:**

   The system provides crop recommendations by analyzing the user's input through the Gaussian Naïve Bayes model. Based on factors like soil quality, weather, and other environmental conditions, it suggests the most suitable crops to grow. These recommendations are shown directly on the web interface, making them easy to view and understand, helping farmers make smarter planting decisions.

5.2 Development Tools & Technologies

1. Google Colaboratory (Colab):

- Google Colaboratory (Colab) is a free, cloud-based Jupyter notebook environment provided by Google.
- It lets us write and run Python code directly in your browser without any setup.
- Colab also offers free access to GPUs and TPUs, making it easier to run computations faster.

2. Flask:

- Flask is a lightweight and flexible web framework that lets us turn machine learning models into web apps or APIs.
- It allows the models to send predictions in response to user requests over the web.
- Flask makes it easy to bring machine learning models into real-world applications, whether it's a web interface or a mobile app.

3. Visual Studio Code (VS Code):

- It is a lightweight, open-source code editor developed by Microsoft.
- It supports many programming languages, including Python, and offers intelligent code completion, debugging, and Git integration.
- Widely used for machine learning development with extensions like Python, Jupyter, and TensorFlow tools.

4. Python:

- Python is the leading language for machine learning and AI development.
- It offers rich libraries like NumPy, pandas, scikit-learn, TensorFlow, and PyTorch.
- Its simple syntax makes model building, training, and testing easier.

5. HTML (HyperText Markup Language):

- HTML is the standard language for creating web pages.
- It structures content using elements like headings, paragraphs, and links.
- In ML apps, HTML builds the front-end for user interaction.

6. CSS (Cascading Style Sheets):

- CSS is used to style and design HTML web pages.
- It controls layout, colors, fonts, and responsiveness.
- In ML apps, CSS makes the interface clean and user-friendly.

7. Bootstrap:

- It is a popular CSS framework for building responsive and modern web pages.
- It provides ready-made components like buttons, forms, and navigation bars.
- In ML apps, Bootstrap helps create clean and mobile-friendly interfaces quickly.

5.3 Model Selection

The Gaussian Naïve Bayes model was selected for crop recommendation as it achieved the highest test accuracy (99.55%). The model closest to this accuracy is the Random Forest, which achieved 99.32%.

5.4 Web Application Development

5.4.1 File Structure

Fig. 23 shows the basic file structure of the developed application consists of four main components:

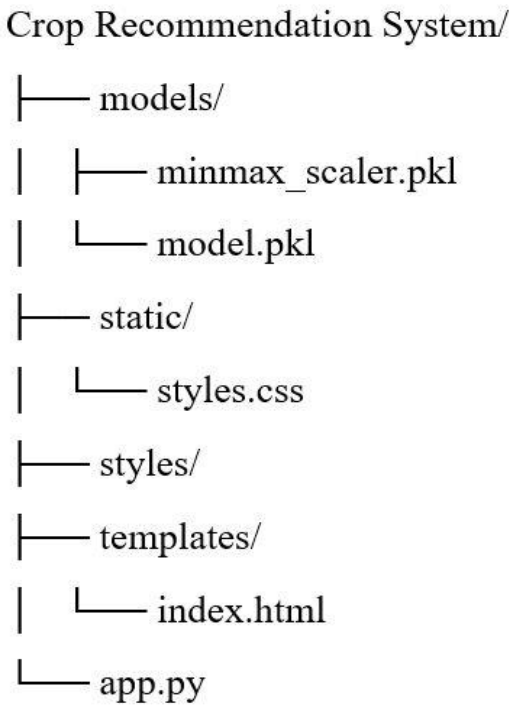


Fig. 23: File structure of the web application.

- **models:** This folder contains the machine learning model used for crop recommendation. Since the Gaussian Naïve Bayes model achieved the highest accuracy of 99.55%, this model is saved here for integration in the application. Additionally, the folder includes the minmix_scaler.pkl file used for scaling input features.

- **app.py:** This script holds the Flask application, which takes crop recommendation-related data from a user-friendly interface, processes it using the saved model, and delivers the prediction results.

- **templates:** This directory contains the HTML file (index.html) that serves as the user interface, letting users enter data for crop recommendations and view the predicted results.

- **static:** This folder contains the CSS file that styles the HTML form, enhancing the overall user experience.

5.4.2 Web Application UI

The front page of the crop recommendation web application is designed using Flask and serves as the user interface for predicting best crop as shown in Fig. 24. This webpage consists of a structured form where users are prompted to input 7 parameters: Nitrogen, Phosphorous, Potassium, Temperature, Humidity, pH and Rainfall.

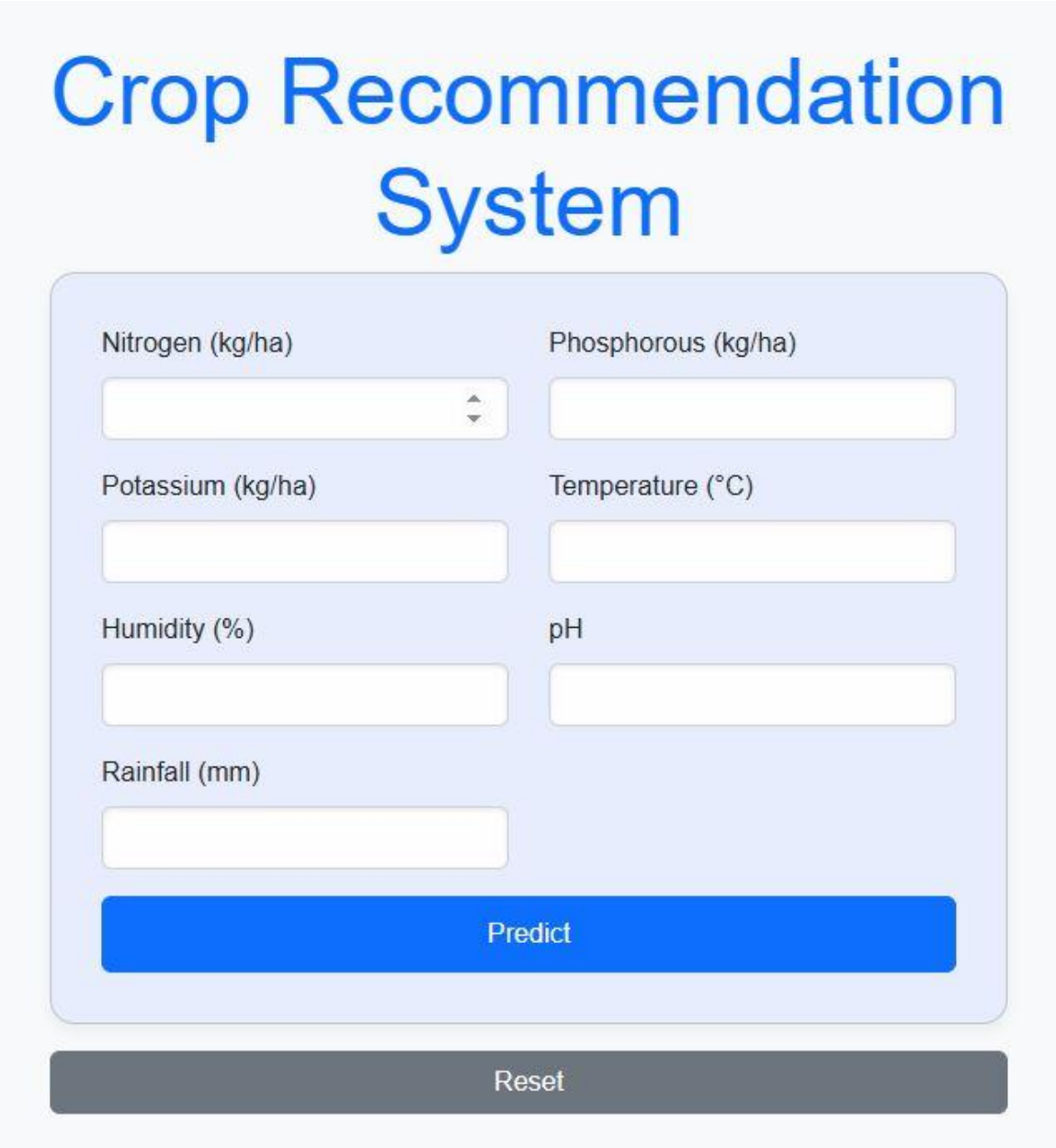


Fig. 24: Front page of the web application.

Once all the input fields are filled, the user clicks the “Predict” button. Upon submission, the application processes the inputs and passes them to the trained machine learning model in the backend. The model then returns a prediction for the best crop. The result is displayed clearly on the same page, offering an intuitive and responsive user experience.

There is also a “Reset” button that sends a GET request to the home route, clearing previous inputs and prediction. It reloads the form with default values, resetting the page state. Crop Recommendation System webpage where input is given and output is displayed, is shown in Fig. 25.

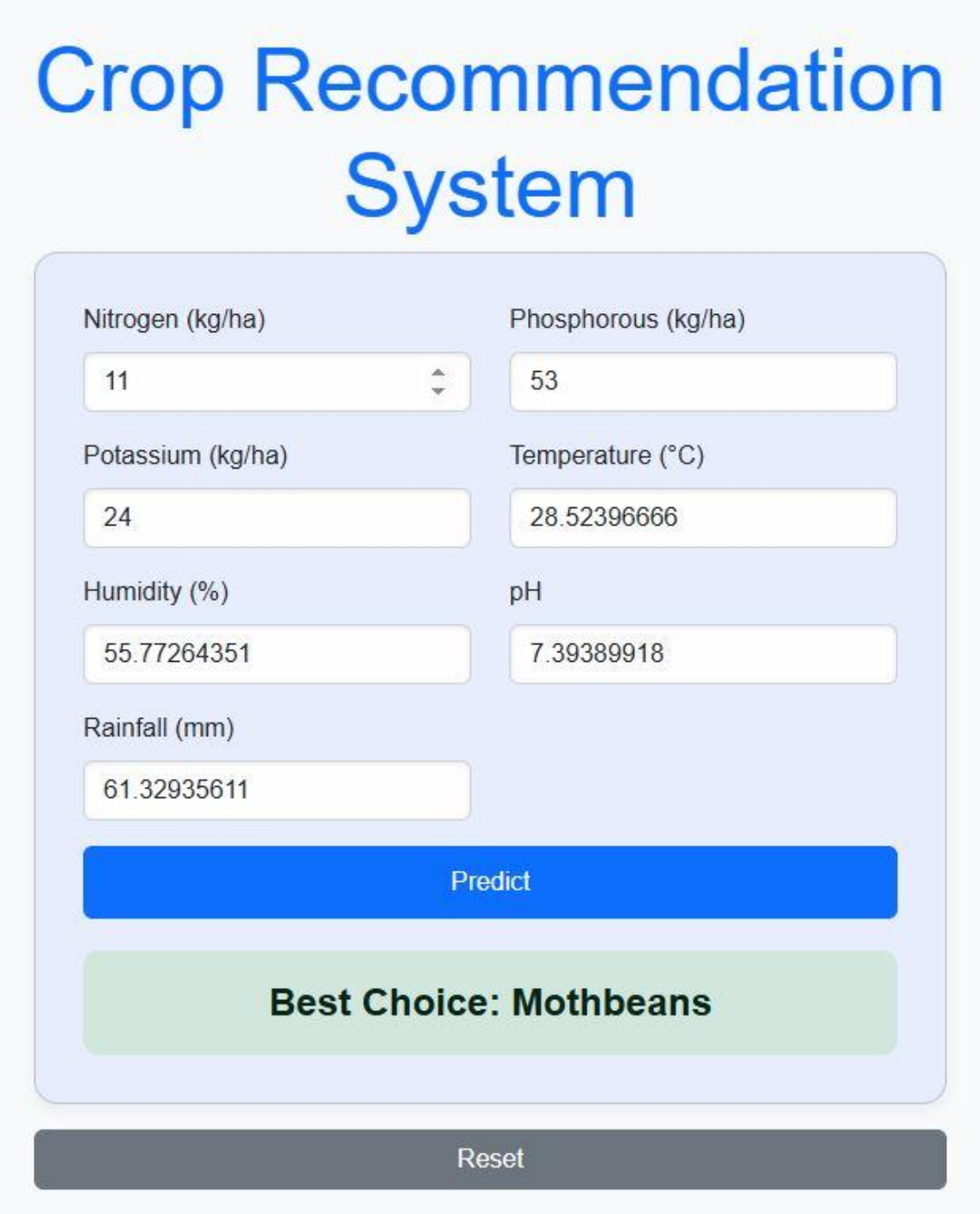


Fig. 25: Prediction result in the web application.

In addition, this web application has the ability to reject the values that are unrealistic for crop recommendation. For example, the pH value can range from 3.5 to 9.9. If the pH value

is less than 3.5 or more than 9.9, then the web application will reject it and ask the user to give a realistic pH value.

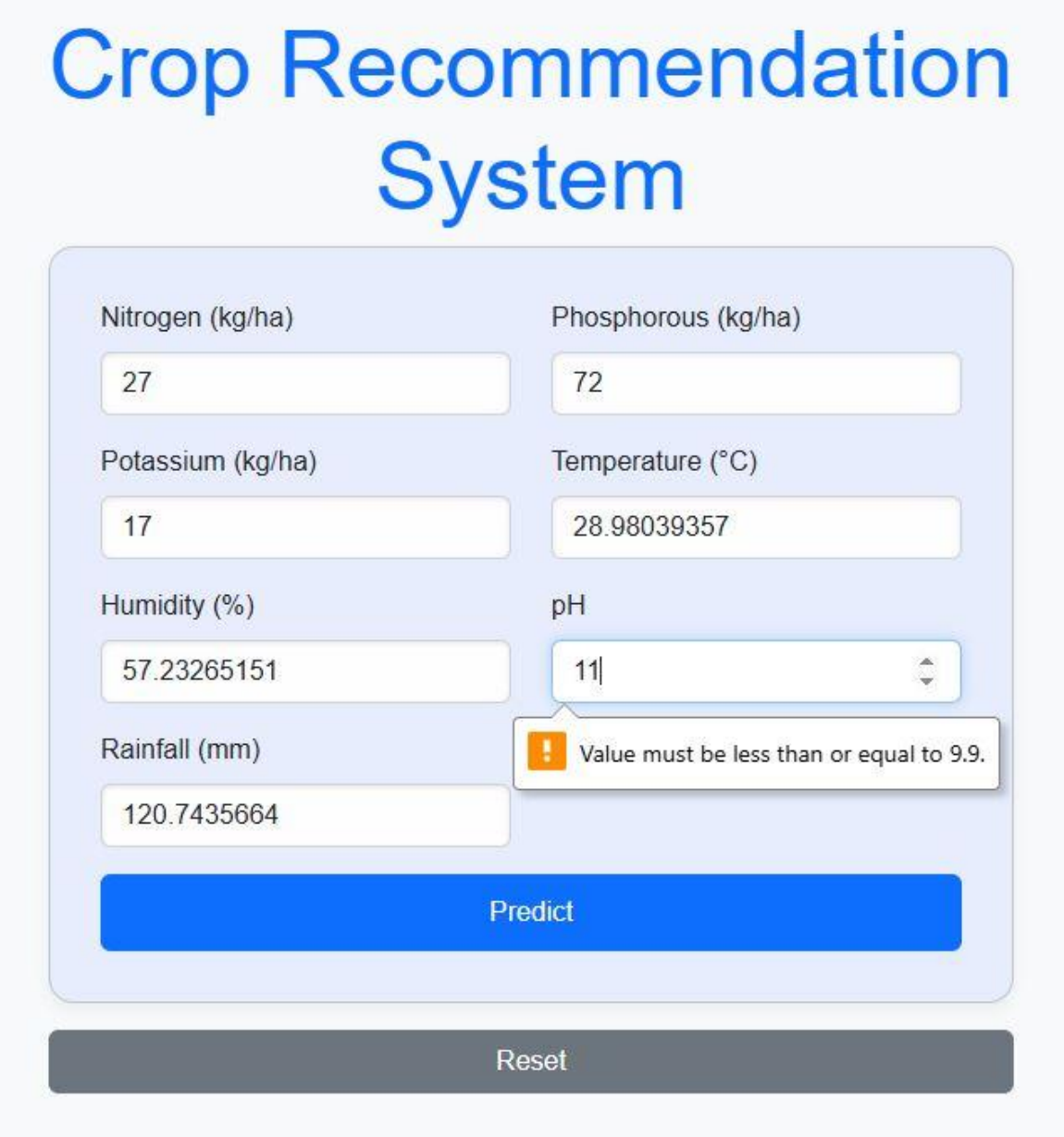


Fig. 26: Each feature requiring a valid real-world value.

Chapter-6

Future Work & Conclusion

6.1 Future Work

This project builds a machine learning–based crop recommendation system to guide farmers in selecting optimal crops. However, it has certain limitations that should be addressed in the future. In the future, the intention is to:

- Work with a large dataset.
- Introduce more models and feature engineering techniques.
- Focus on improving the overall accuracy.

6.2 Conclusion

This project is centered on developing a crop recommendation system with Python and machine learning to assist farmers in selecting the most suitable crops for their fields. By analyzing soil quality, weather conditions, and other environmental factors, the system provides personalized, data-driven suggestions that can boost productivity and minimize resource waste. Farmers and other users can access real-time recommendations, allowing them to make smarter decisions about what to plant. This not only helps increase yields and reduce costs but also lowers risks associated with unpredictable environment. It provides users (farmers and relevant users) with accessible, real-time recommendations, improving crop yield, reducing costs, and mitigating risks related to weather. The proposed Flask web application enables farmers to enter soil and environmental details and get immediate crop recommendations which empowers farmers to make informed decisions, contributing to better agricultural practices.

References

[1] Ersin Elbasi, Chamseddine Zaki, Ahmet E. Topcu, Wiem Abdelbaki, Aymen I. Zreikat, Elda Cina, AhmedShdefat and Louai Saker, *"Crop Prediction Model Using Machine Learning Algorithms,"* Applied Sciences (2023, Volume 13, Issue 9288).

[2] Pedina Sasi Kiran, Gembali Abhinaya, Smaraneeka Sruti, and Neelamadhab Padhy, *"A Machine Learning-Enabled System for Crop Recommendation,"* 3rd International Electronic Conference on Processes (ECP 2024), 29–31 May 2024.

[3] S. Pudumalar, E. Ramanujam, R. Harine Rajashree, C. Kavya, T. Kiruthika, J. Nisha, *"Crop Recommendation System for Precision Agriculture,"* 2016 IEEE Eighth International Conference on Advanced Computing (ICOAC).

[4] Dhruvi Gosai, Chintal Raval, Rikin Nayak, Hardik Jayswal, Axat Patel, *"Crop Recommendation System using Machine Learning,"* International Journal of Scientific Research in Computer Science, Engineering and Information Technology, May-June 2021.

[5] Vandana Kale and Badri Narayan Mohapatra, *"Crop Recommendation System Using Machine Learning,"* ITEGAM-JETIA, Manaus, v.10 n.48, p. 63-68, July/August 2024.

[6] Biplob Dey, Jannatul Ferdous and Romel Ahmed, *"Machine learning based recommendation of agricultural and horticultural crop farming in India under the regime of NPK, soil pH and three climatic variables,"* Heliyon, vol. 10, no. 1, 2024, p. e25112.

[7] Paramweer Kaur and Brahmaleen Kaur Sidhu, *"Crop recommendation using machine learning,"* Applied Data Science and Smart Systems journal (2022).

[8] Ajay Lokhande and Prof. Manish Dixit, *" Crop Recommendation System Using Machine Learning,"* International Research Journal of Engineering and Technology (IRJET), Volume: 09, Issue: 05, May 2022.

[9] Madhuri Shripathi Rao, Arushi Singh, N.V. Subba Reddy, and Dinesh U. Acharya, *" Crop prediction using machine learning,"* Journal of Physics: Conference Series, vol. 2161, no. 1, p. 012033, Jan. 2022.

[10] Mahendra N, Dhanush Vishwakarma, Nischitha K, Ashwini, and Manjuraju M.R, *" Crop Prediction using Machine Learning Approaches,"* International Journal of Engineering Research & Technology (IJERT), 9(08), 23-25, August-2020.

[11] Farida Siddiqi Prity, MD. Mehadi Hasan, Shakhawat Hossain Saif , Md. Maruf Hossain, Sazzad Hossain Bhuiyan, Md. Ariful Islam and Md Tousif Hasan Lavlu, *" Enhancing Agricultural Productivity: A Machine Learning Approach to Crop Recommendations,"* Human-Centric Intelligent Systems, vol. 4, 2024, pp. 497-510.

[12] Mahmudul Hasan, Md Abu Marjan, Md Palash Uddin, Masud Ibn Afjal, Seifedine Kardy, Shaoqi Ma, and Yunyoung Nam, *" Ensemble machine learning-based recommendation system for effective prediction of suitable agricultural crop cultivation,"* Frontiers in Plant Science, 14, Article 1234555, August 2023.

[13] Guna Sekhar Sajja, Subhesh Saurabh Jha, Hicham Mhamdi, Mohd Naved, Samrat Ray and Khongdet Phasinam, *" An Investigation on Crop Yield Prediction Using Machine Learning,"* Third International Conference on Inventive Research in Computing Applications (ICIRCA), Sept. 2021, pp. 916–921.

[14] Thomas van Klompenburg, Ayalew Kassahun and Cagatay Catal, *"Crop yield prediction using machine learning: A systematic literature review,"* Computers and Electronics in Agriculture, Volume 177, Article 105709, August, 2020.