**MACHINE LEARNING-BASED CROP RECOMMENDATION SYSTEM FOR ENHANCED YIELD PRODUCTION**

**ABSTRACT**

Agriculture plays a critical role in ensuring that there is enough food to feed our ever-growing global population. With the challenges posed by climate change and its uncertainties, it has become increasingly important to adopt sustainable practices in agriculture. One crucial aspect of this is optimizing how we use our land, water resources, and fertilizers while also increasing crop yields. Traditionally, farmers relied on their own experience, local knowledge, and trial-and-error methods to decide which crops to grow in their specific regions and soil types. Unfortunately, this subjective approach often led to uncertainties, reduced productivity, and economic losses. Inappropriate crop choices were also common, resulting in suboptimal yields and wasted resources. To address these issues, we are working on developing a Machine Learning-based Crop Recommendation System. This system holds immense promise in transforming agriculture by offering farmers data-driven insights to make informed decisions about their crop selection and management practices. The system uses a wealth of data, including historical crop performance, soil characteristics, and climate patterns, to empower farmers with intelligent recommendations. By analysing this vast array of information, the system guides farmers in making data-driven choices, reducing the risk of crop failure, and increasing profitability. Furthermore, it promotes the adoption of sustainable farming methods, thereby minimizing environmental impacts and preserving precious natural resources for future generations. Implementing this recommendation system can have far-reaching benefits. It can lead to economic advantages for farmers, ensuring food security, and encouraging the adoption of sustainable agricultural practices to protect our environment. By providing valuable insights, the system empowers farmers to make informed decisions that ultimately lead to improved crop yields, reduced resource wastage, and increased profitability. With the potential to revolutionize crop selection, this novel approach paves the way toward a more sustainable and food-secure future for all of us.

**CHAPTER – 1**

**INTRODUCTION**

**1.1 Overview**

Agriculture is the cornerstone of global food security and sustenance, making it an industry of paramount importance. However, the efficiency and productivity of agricultural practices are often constrained by factors such as soil quality, climate conditions, and crop choice. In this context, the development of a "Machine Learning-Based Crop Recommendation System for Enhanced Yield Production" emerges as a pivotal endeavour that holds the potential to revolutionize the way crops are cultivated and harvested. At its core, this research aims to bridge the gap between traditional agricultural practices and cutting-edge technology [1]. It leverages the power of machine learning, a subset of artificial intelligence, to optimize crop selection decisions made by farmers. This innovation is of immense significance given the growing challenges faced by the agriculture sector, including the need to feed an expanding global population, mitigate climate change impacts, and reduce resource wastage.

The primary motivation behind this research lies in addressing several pressing issues. Firstly, the unpredictable effects of climate change have significantly altered weather patterns and precipitation levels, making it increasingly difficult for farmers to make informed crop choices based on historical data alone [2]. Additionally, soil quality can vary dramatically even within a single farm, necessitating a more nuanced approach to crop selection. Traditional methods, which often rely on anecdotal knowledge and experience, are no longer sufficient to ensure optimal yield and resource efficiency. Machine learning offers a novel and data-driven solution to this dilemma. By analyzing a myriad of factors, including historical weather data, soil composition, and past crop performance, machine learning algorithms can provide personalized and accurate crop recommendations. This not only optimizes yield but also promotes sustainable farming practices by reducing the unnecessary use of water, pesticides, and fertilizers [3].

Furthermore, the research recognizes the global imperative of sustainable agriculture. In a world grappling with issues of food security and environmental conservation, the ability to recommend crops that are not only high-yielding but also environmentally responsible is crucial. Machine learning models can factor in environmental impact considerations, helping farmers make choices that align with conservation and resource preservation goals [4].

In this introductory overview, we will delve into the key components and objectives of this research. Firstly, we will explore the challenges faced by farmers and the limitations of conventional crop selection methods. Next, we will introduce the concept of machine learning-based crop recommendation systems, highlighting their potential to revolutionize agriculture. We will also discuss the role of data sources, including historical data, satellite imagery, and soil assessments, in training these models. Additionally, ethical considerations, such as data privacy and equitable access to technology, will be addressed [5]. The research on a "Machine Learning-Based Crop Recommendation System for Enhanced Yield Production" seeks to empower farmers with data-driven decision-making tools that can optimize crop selection, enhance yield, and promote sustainable agricultural practices. By harnessing the capabilities of machine learning, this research endeavours to usher in a new era of precision agriculture that is not only more productive but also environmentally responsible and equitable [6].

**1.2 Motivation**

The research motivation for a "Machine Learning-Based Crop Recommendation System for Enhanced Yield Production" is deeply rooted in the critical challenges facing agriculture today. As the global population continues to grow, the pressure on the agricultural sector to produce more food intensifies. This demand necessitates not only increased agricultural productivity but also a more efficient approach to crop selection [7]. Traditional farming methods often rely on historical practices and lack the adaptability required to optimize crop choices in the face of evolving climate conditions and soil quality. This research is motivated by the urgent need to bridge this gap between traditional farming practices and cutting-edge technology, as it holds the potential to significantly enhance global food security.

Climate change further compounds the challenges in agriculture, introducing unpredictability into weather patterns and exacerbating the uncertainty surrounding crop selection. Farmers require adaptive strategies to cope with these changing conditions, and machine learning-based crop recommendation systems provide a promising solution by analyzing historical weather data and climate projections to offer data-driven guidance. Additionally, resource inefficiency in agriculture, including the excessive use of water, fertilizers, and pesticides, poses significant environmental and economic concerns [8]. By recommending crops that align with local conditions and resource availability, this research aims to optimize resource utilization, reduce waste, and promote sustainable farming practices, thereby mitigating environmental degradation [9].

Sustainability is a global imperative, and agriculture plays a pivotal role in this pursuit. The research is motivated by the goal of fostering sustainable farming practices that address environmental impact issues such as soil erosion, water pollution, and habitat destruction. Machine learning-based recommendations not only prioritize high-yield crops but also consider their environmental footprint, contributing to responsible land management and conservation efforts. Moreover, this research is driven by the desire to empower smallholder farmers, who often lack access to advanced agricultural technology and expertise. By providing accessible and user-friendly crop recommendation systems, it seeks to empower these farmers, improve their livelihoods, and reduce poverty [10].

In an era of big data, agriculture stands to benefit significantly from data-driven decision-making. The research motivation lies in harnessing vast datasets, including historical agricultural data, satellite imagery, and soil information, to enhance crop recommendations. This approach promises to usher in a more efficient and informed era of precision agriculture, where every decision is underpinned by comprehensive data analysis. Finally, ethical considerations underpin the research, as ensuring that machine learning-based crop recommendations are developed and deployed ethically is paramount. This involves addressing issues related to data privacy, fairness, and ensuring equitable access to technology to ensure that the benefits of these systems are accessible to all, irrespective of their resources or geographical location. In summary, the research on a "Machine Learning-Based Crop Recommendation System for Enhanced Yield Production" is motivated by the urgent need to address food security, adapt to climate change, promote resource efficiency, and foster sustainability in agriculture, while also empowering farmers and adhering to ethical principles in technology deployment.

**1.3 Problem Statement**

Modern agriculture faces multifaceted challenges that impede its ability to meet the growing global demand for food, ensure sustainability, and adapt to changing environmental conditions. One of the pivotal challenges is the suboptimal selection of crops for cultivation. Traditional crop selection methods often rely on historical practices, anecdotal knowledge, and limited consideration of factors such as soil quality, climate variability, and resource availability. These methods are becoming increasingly inadequate in the face of climate change-induced weather unpredictability, variations in soil health even within small geographic areas, and the imperative of resource efficiency. The consequence of suboptimal crop selection is diminished yield potential, resource wastage, increased vulnerability to climate-related risks, and an unsustainable environmental footprint.

The core problem, therefore, centers on the need for an innovative and data-driven approach to crop selection that leverages the capabilities of machine learning. This approach should harness diverse datasets, including historical agricultural data, weather records, soil composition information, and crop performance metrics, to provide farmers with precise, real-time, and personalized crop recommendations. However, this challenge extends beyond the technical domain; it also encompasses ethical considerations such as data privacy, fairness, and equitable access to technology. It is imperative to ensure that the benefits of machine learning-based crop recommendation systems are accessible to a wide range of farmers, irrespective of their geographical location, economic resources, or technological proficiency. Furthermore, the ethical framework must safeguard data privacy rights and prevent the unintentional reinforcement of biases within the technology. In essence, the problem is twofold: to develop a robust machine learning-based crop recommendation system that optimizes yield production while addressing sustainability and resource efficiency, and to ensure that its deployment adheres to ethical principles, promoting inclusivity and responsible technology adoption within the agricultural sector.

**1.4 Applications**

Precision Agriculture: The primary application is in precision agriculture, where machine learning algorithms can provide farmers with tailored crop recommendations based on real-time data, including soil conditions, weather forecasts, and historical performance. This promotes resource-efficient and sustainable farming practices.

Increased Crop Yields: Farmers can use the recommendations to optimize their crop selection, leading to increased yields and improved agricultural productivity. This application is particularly crucial for feeding the growing global population.

Resource Efficiency: Machine learning-based recommendations can help reduce resource wastage, such as water, fertilizers, and pesticides. This contributes to environmental sustainability by minimizing the ecological footprint of agriculture.

Climate Resilience: Farmers can adapt to changing climate conditions by receiving timely advice on crop selection that is better suited to evolving weather patterns and climate variability.

Diversification of Crops: The system can encourage crop diversification, promoting biodiversity and reducing the risk of crop failures due to pests, diseases, or adverse weather events.

Farm Management: Farm managers and agricultural consultants can use these recommendations to make informed decisions about crop rotations, land-use planning, and farm management strategies.

Agricultural Extension Services: Agricultural extension officers can incorporate machine learning-based recommendations into their services to provide local farmers with valuable insights and guidance.

Educational Tools: Educational institutions and agricultural training programs can use these systems as teaching tools to educate future generations of farmers about data-driven agricultural practices.

Research and Development: Researchers in agriculture can use the data generated by these systems to study crop performance, evaluate the impact of different agricultural practices, and develop new insights into crop management.

Environmental Conservation: By recommending crops that are well-suited to local conditions, including soil types and precipitation levels, these systems can contribute to sustainable land use and conservation efforts.

Policy Development: Governments and policymakers can use the data collected from these systems to inform agricultural policies, subsidies, and incentives aimed at promoting sustainable farming practices.

Global Food Security: Machine learning-based crop recommendations can play a crucial role in improving food security by enhancing agricultural efficiency and productivity, especially in regions facing food shortages and agricultural challenges.

Smallholder Agriculture: These systems can empower smallholder farmers by providing them with access to advanced agricultural technology and knowledge, helping them increase their income and improve their livelihoods.

Community and Cooperative Farming: Agricultural cooperatives and community farming initiatives can use these recommendations to make collective decisions about crop selection, leading to improved overall outcomes for the community.

**CHAPTER – 2**

**LITERATURE SURVEY**

Prakash, et al. [11] proposed a Machine learning-based crop suggestion system. an intelligent model in crop suggestion predicts the right kind of crop that suits the soil and other environmental factors by using a machine learning algorithm. Temperature, wetness of the soil, environment dampness and pH are the essential information for crop suggestion in the agriculture field. Many existing works of IoT in the field of agriculture are related to yield prediction and few related to crop suggestion use fewer parameters and learning algorithms.

Indira, et al. [12] proposed A Machine Learning based New Recommendation System to the Farmer. An algorithm called MobileNet uses an image of a leaf to identify the disease present in a plant. The XGBoost model predicts a suitable crop based on the local soil nutrients and rainfall. Random Forest [RF] model was used to propose fertilizer and develop ideas for improving soil fertility depending on nutrients present in the soil. When compared to other approaches, the proposed model delivers a high level of accuracy.

Jeevaganesh, et al. [13] proposed a machine learning algorithm: AdaBoost to predict the yield of crops based on the parameters like state, district, area, seasons, rainfall, temperature, and area. To enhance the yield, this research study also suggests a fertilizer based on soil conditions like NPK values, soil type, soil PH, humidity, and moisture. Fertilizer recommendation is primarily done by using the Random Forest [RF] algorithm.

Wen, et al. [14] proposed Optimizing machine learning-based site-specific nitrogen application recommendations for canola production. This work employed four machine learning models, namely, support vector machine (SVM), gradient boosting (GB), random forest (RF), and ridge regression (RR) to predict the site-specific EONR values of canola crops, based on a 22-site-year field study across eastern Canada.

SSL, et al. [15] proposed An Intelligent Crop Recommendation System using Deep Learning. The purpose of this work is to examine the approaches employed in extracting the water bodies utilizing the mode of satellite remote sensing. The goal of the proposed work is to collect the data of temperature and humidity and utilize the algorithm of clustering with the method of k-Nearest Neighbor to find out the patterns which are all hidden in them with a help of huge amount of dataset.

Chowdhary, et al. [16] proposed An Ensemble Model to Predict Crops using Machine Learning Algorithms. The authors tried to form an ensemble model using various machine-learning algorithms for better rice production. Crop production prediction utilizing AI Strategies aims to deliver improved outcomes, but the ensemble model provides better predictive results compared to the individual algorithm. we tried to use a combination of symmetric machine-learning algorithms to form an ensemble model for better prediction.

Suresh, et al. [17] proposed a solution based on IoT and deep learning for improving paddy output. The system collects data through sensors and transfers it to the cloud in order to diagnose plant stress caused by soil fertility, environmental imbalance, and crop diseases (AWS EC2 Server). The proposed system is implemented in three stages. The stages of production are divided into three categories: the start stage (planting to panicle initiation), the middle stage (panicle initiation to flowering), and the end stage (flowering to maturity).

Raviraja, et al. [18] proposed machine learning-based mobile applications for autonomous fertilizer suggestion. In this work, the smartphone app will be used to construct an autonomous fertilizer suggestion system for farmers. The dataset was gathered from the website Kaggle. The encoding technique is used to make the data entries numeric. Three Machine Learning (ML) models K-Nearest Neighbour (KNN), Random Forest (RF), and Decision Tree (DT) are employed to predict fertilizer based on environmental and soil characteristics.

Mamatha, et al. [19] proposed Machine learning-based crop growth management in a greenhouse environment using hydroponics farming techniques. In this proposed system, in hydroponics for germination organic coconut coir medium is used rather than rock wool, because rock wool is not bio-degradable and is composed of volcanic materials. In the proposed research of hydroponics, the system is automated on a wide scale covering the entire green house with different crops produced in different climatic conditions.

Oikonomidis, et al. [20] proposed Hybrid deep learning-based models for crop yield prediction. The authors had developed deep learning-based models to evaluate how the underlying algorithms perform with respect to different performance criteria. The algorithms evaluated in our study are the XGBoost machine learning (ML) algorithm, Convolutional Neural Networks (CNN)-Deep Neural Networks (DNN), CNN-XGBoost, CNN-Recurrent Neural Networks (RNN), and CNN-Long Short-Term Memory (LSTM).

Apat, et al. [21] proposed proposed an IoT-HELE-based smart farming prediction and intelligent agriculture analytics model and a decision support system that effectively predicts crop production by utilizing cutting-edge machine learning and deep learning techniques. In this model, ensemble voting results in a more efficient, sustainable, and profitable agriculture enterprise. The multi-source dataset from the National Research Council (CNR), an ISTAT, and an IoT sensor will be analyzed.

Kumar, et al. [22] proposed Crop recommendations using machine learning algorithms. The proposed system in which ML is used for crop recommendation is based on previously recorded measurements of soil parameters. This technique lessens the possibility of soil degradation and aids in crop health maintenance. Temperature, pH, and N, P, K, humidity are analyzed using machine learning algorithms such as random forest, Naive Bayes, KNN, decision tree, Logistic regression on which suggestions are made for growing a suitable crop.

Sandhya, et al. [23] proposed Crop Recommendation System Using Ensembling Technique. This work focuses on figuring out the best crop to cultivate in order to get optimum yield based on the site-specific parameters. Our proposed model takes the data of soil characteristics, environmental characteristics of a farm and the appropriate crop recommendations are given to the farmer based on the parameter values. Crop Recommendation is done through an Ensemble model using KNN, Random Forest, Gaussian Naïve Bayes, Logistic regression, SVM as base learners.

Muhammed, et al. [24] proposed a user-friendly crop recommendation system called UACR. The users only need to insert their farm location and the system can automatically obtain other necessary information (e.g., rain level, soil type) based on the requested location and recommends the best crop(s) for growing in the location. To support and implement the proposed system, an architecture based on Artificial Intelligence of Things (AIoT) technology, as one of the main enablers for smart agriculture, is also proposed.

Venkatesan, et al. [25] proposed a machine learning-based prediction model for peak energy use by analyzing energy-related data collected from various environmental and growth devices in a smart paprika farm of the Jeonnam Agricultural Research and Extension Service in South Korea between 2019 and 2021. The proposed model may contribute to the development of various applications for environmental energy usage in a smart farm, such as a notification service for energy usage peak time or an energy usage control for each device.

**CHAPTER 3**

**EXISITING TECHNOLOGY**

**Logistic Regression**

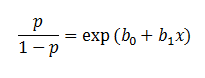
Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

* A linear regression will predict values outside the acceptable range (e.g. predicting probabilities
* outside the range 0 to 1)
* Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

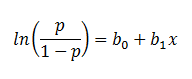
On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the “odds” of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

****

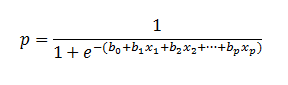
In the logistic regression the constant (b0) moves the curve left and right and the slope (b1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.



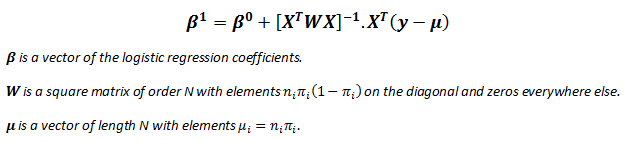
Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (*b1*) is the amount the logit (log-odds) changes with a one unit change in *x*.



As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.



There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses [maximum likelihood estimation](https://www.saedsayad.com/further_readings.htm) (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

****

A pseudo R2 value is also available to indicate the adequacy of the regression model. Likelihood ratio test is a test of the significance of the difference between the likelihood ratio for the baseline model minus the likelihood ratio for a reduced model. This difference is called "model chi-square“. Wald test is used to test the statistical significance of each coefficient (*b*) in the model (i.e., predictors contribution).

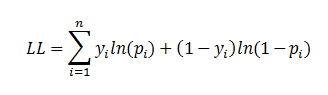
Pseudo R2

There are several measures intended to mimic the R2 analysis to evaluate the goodness-of-fit of logistic models, but they cannot be interpreted as one would interpret an R2 and different pseudo R2 can arrive at very different values. Here we discuss three pseudo R2measures.

|  |  |  |
| --- | --- | --- |
| **Pseudo R2** | **Equation** | **Description** |
| **Efron's** |  | ***'p'* is the logistic model predicted probability. The model residuals are squared, summed, and divided by the total variability in the dependent variable.** |
| **McFadden's** |  | **The ratio of the log-likelihoods suggests the level of improvement over the intercept model offered by the full model.** |
| **Count** |  | **The number of records correctly predicted, given a cutoff point of .5 divided by the total count of cases. This is equal to the**[**accuracy**](https://www.saedsayad.com/model_evaluation.htm)**of a classification model.** |

**Likelihood Ratio Test**

The likelihood ratio test provides the means for comparing the likelihood of the data under one model (e.g., full model) against the likelihood of the data under another, more restricted model (e.g., intercept model).



where '*p'* is the logistic model predicted probability. The next step is to calculate the difference between these two log-likelihoods.



The difference between two likelihoods is multiplied by a factor of 2 in order to be assessed for statistical significance using standard significance levels (Chi2 test). The degrees of freedom for the test will equal the difference in the number of parameters being estimated under the models (e.g., full and intercept).

**Disadvantages of Logistic Regression:**

* Limited to binary classification problems.
* Assumes a linear relationship between predictors and the log-odds of the outcome.
* Sensitive to outliers in the data.
* Not suitable for capturing complex, non-linear relationships.
* Requires complete data without missing values.
* Assumes observations are independent of each other.
* Prone to overfitting, especially with many predictors.
* Requires numeric data, may struggle with categorical variables.
* Assumes a linear decision boundary.
* Complex to interpret probabilities in multi-class problems.

**CHAPTER 4**

**PROPOSED METHODOLOGY**

**4.1 Overview**

The Python code that implements a graphical user interface (GUI) using the Tkinter library for a machine learning-based crop recommendation system. Here's a breakdown of the main components and functionalities of the code

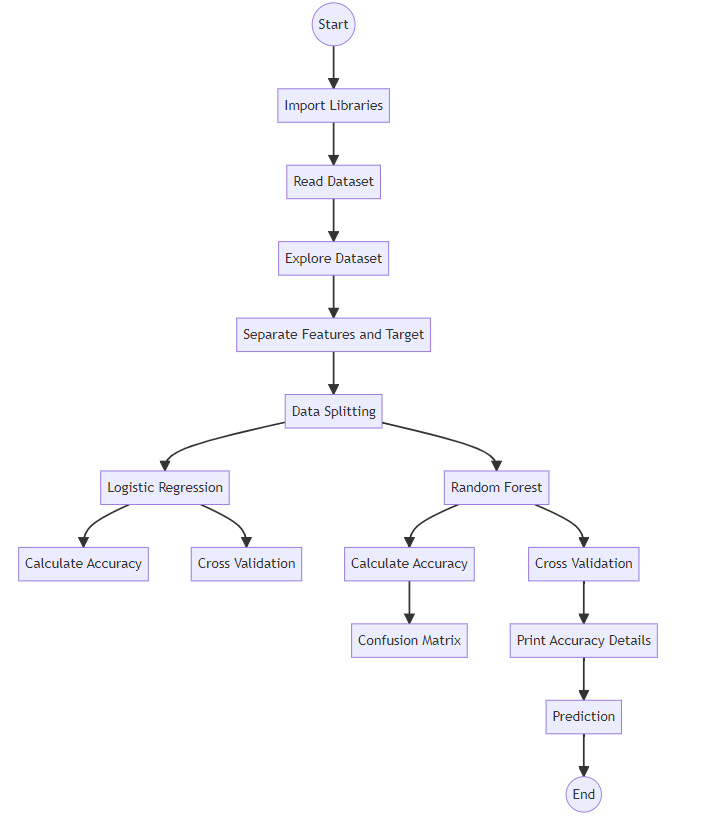


Figure 4.1. Proposed System model.

Importing Libraries: The script starts by importing necessary libraries such as Tkinter, NumPy, Pandas, Seaborn, Scikit-learn, Matplotlib, etc.

Global Variables: Global variables are declared to store dataset information, model evaluation metrics, and other necessary data.

Tkinter Setup: The main Tkinter window is created with a specific title and geometry.

Function Definitions:

uploadDataset(): Allows the user to upload a dataset file.

preprocessDataset(): Preprocesses the uploaded dataset, which includes handling missing values, encoding labels, splitting data into training and testing sets, and displaying basic statistics about the dataset.

rocGraph(): Plots the Receiver Operating Characteristic (ROC) curve for the given test data and predictions.

custom\_knn\_classifier(): Trains a K-Nearest Neighbors (KNN) classifier on the dataset, evaluates its performance, and displays the results.

Randomforestclassifier(): Trains a Random Forest classifier on the dataset, evaluates its performance, and displays the results.

Predict(): Allows the user to upload a test dataset file and makes predictions using the trained Random Forest classifier.

graph(): Compares the performance of KNN and Random Forest classifiers based on various evaluation metrics and displays the results using a bar graph.

close(): Closes the Tkinter window.

GUI Components:

Labels, buttons, and a text widget are created to provide a user-friendly interface for dataset upload, preprocessing, model training, prediction, and result display.

Main Loop: The script enters the Tkinter event loop to handle user interactions and GUI updates.

**4.2 Data Preprocessing**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding Missing Data
* Encoding Categorical Data
* Splitting dataset into training and test set

**4.3 Dataset Splitting**

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

**Training** **Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test** **set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

**4.4 Random Forest Algorithm**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Diagram

Description automatically generated

Fig. 4.2: Random Forest algorithm.

**4.4.1 Random Forest algorithm**

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

**4.4.2 Important Features of Random Forest**

* **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
* **Immune** **to** **the** **curse** **of** **dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
* **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
* **Train-Test** **split**- In a random forest we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
* **Stability**- Stability arises because the result is based on majority voting/ averaging.

**4.4.3 Assumptions for Random Forest**

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

* There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
* The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.

**4.4.4 Types of Ensembles**

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

**Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

**Diagram

Description automatically generated**

Fig. 4.3: RF Classifier analysis.

**Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

Diagram

Description automatically generated

Fig. 4.4: Boosting RF Classifier.

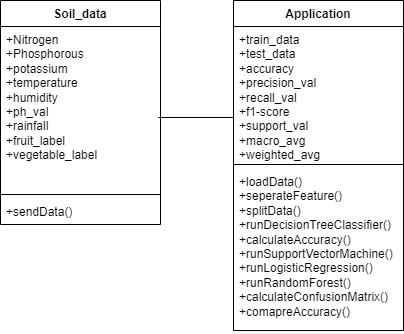
**4.5 Advantages of proposed system**

* It can be used in classification and regression problems.
* It solves the problem of overfitting as output is based on majority voting or averaging.
* It performs well even if the data contains null/missing values.
* Each decision tree created is independent of the other thus it shows the property of parallelization.
* It is highly stable as the average answers given by a large number of trees are taken.
* It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.
* It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.

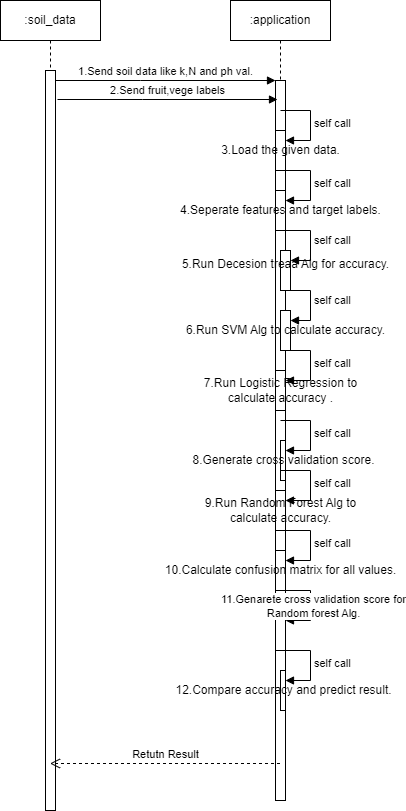
**CHAPTER 5**

**UML DIAGRAMS**

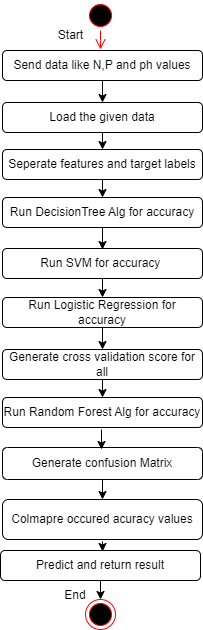
**Class Diagram**: Class diagram is a static diagram. It represents the static view of an application.



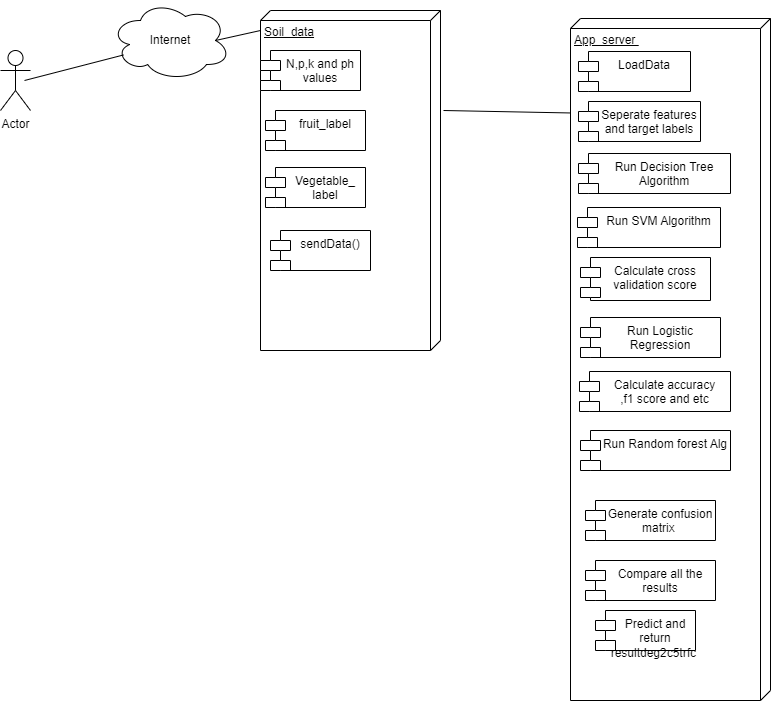
**Sequence Diagram** : Sequence diagram is an interaction diagram that details how operations are carried out.



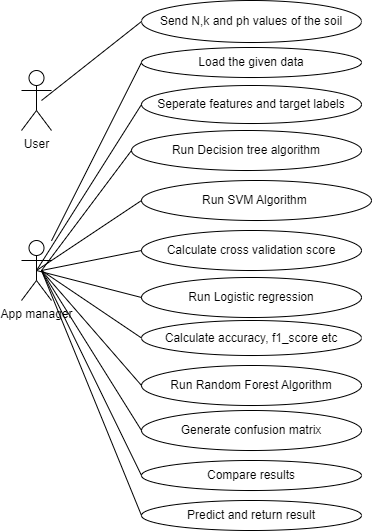
**Activity diagram :** Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.



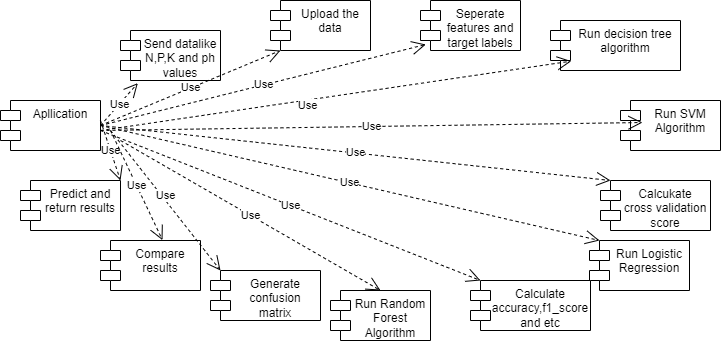
**Deployment diagram** : The deployment diagram visualizes the physical hardware on which the software will be deployed.



**Usecase diagram:** The purpose of use case diagram is to capture the dynamic aspect of a system.



**Component diagram**: Component diagram describes the organization and wiring of the physical components in a system.



**CHAPTER 6**

**SOFTWARE ENVIRONMENT**

**What is Python?**

Below are some facts about Python.

* Python is currently the most widely used multi-purpose, high-level programming language.
* Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
* Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
* Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

* Machine Learning
* GUI Applications (like Kivy, Tkinter, PyQt etc. )
* Web frameworks like Django (used by YouTube, Instagram, Dropbox)
* Image processing (like Opencv, Pillow)
* Web scraping (like Scrapy, BeautifulSoup, Selenium)
* Test frameworks
* Multimedia

**Advantages of Python**

Let’s see how Python dominates over other languages.

**1.Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

**Advantages of Python Over Other Languages**

**1.Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

**1.Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**Modules Used in Project**

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and Ipython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with Ipython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www>.python.org

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

**Verify the Python Installation**

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

**Check how the Python IDLE works**

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**CHAPTER 7**

**SYSTEM REQUIREMENTS**

**Software Requirements**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* Python IDLE 3.7 version (or)
* Anaconda 3.7 (or)
* Jupiter (or)
* Google colab

**Hardware Requirements**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

* Operating system : Windows, Linux
* Processor : minimum intel i3
* Ram : minimum 4 GB
* Hard disk : minimum 250GB

**CHAPTER 8**

**FUNCTIONAL REQUIREMENTS**

**Output Design**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**Output Definition**

The outputs should be defined in terms of the following points:

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**Input Design**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**Error Detection**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Clasified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

**Chapter 9**

**SOURCE CODE**

from tkinter import \*

import tkinter

import tkinter as tk

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

from tkinter import simpledialog

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

import os

import matplotlib.pyplot as plt

from sklearn.metrics import roc\_curve

from sklearn.metrics import roc\_auc\_score

from sklearn import metrics

main = tkinter.Tk()

main.title("MACHINE LEARNING-BASED CROP RECOMMENDATION SYSTEM FOR ENHANCED YIELD PRODUCTION")

main.geometry("1000x650")

global filename

global x\_train,y\_train,x\_test,y\_test

global X, Y

global le

global dataset

accuracy = []

precision = []

recall = []

fscore = []

global classifier

global cnn\_model

def uploadDataset():

global filename

global dataset

filename = filedialog.askopenfilename(initialdir = "Dataset")

text.delete('1.0', END)

text.insert(END,filename+' Loaded\n')

dataset = pd.read\_csv(filename)

text.insert(END,str(dataset.head())+"\n\n")

def preprocessDataset():

global X, Y

global le

global dataset

global x\_train,y\_train,x\_test,y\_test

le = LabelEncoder()

text.delete('1.0', END)

dataset.fillna(0, inplace = True)

text.insert(END,str(dataset.head())+"\n\n")

# Create a count plot

sns.set(style="darkgrid") # Set the style of the plot

plt.figure(figsize=(8, 6)) # Set the figure size

# Replace 'dataset' with your actual DataFrame and 'Drug' with the column name

ax = sns.countplot(x='label', data=dataset, palette="Set3")

plt.title("Count Plot") # Add a title to the plot

plt.xlabel("Crop Categories") # Add label to x-axis

plt.ylabel("Count") # Add label to y-axis

# Annotate each bar with its count value

for p in ax.patches:

ax.annotate(f'{p.get\_height()}', (p.get\_x() + p.get\_width() / 2., p.get\_height()),

ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),

textcoords='offset points')

plt.show() # Display the plot

le = LabelEncoder()

dataset['label'] = le.fit\_transform(dataset['label'])

X=dataset.iloc[:,:-1].values

y=dataset.iloc[:,-1].values

text.insert(END,"Total records found in dataset: "+str(X.shape[0])+"\n\n")

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2, random\_state=0)

text.insert(END,"Total records found in dataset to train: "+str(x\_train.shape[0])+"\n\n")

text.insert(END,"Total records found in dataset to test: "+str(x\_test.shape[0])+"\n\n")

print(x\_train)

def rocGraph(testY, predict, algorithm):

random\_probs = [0 for i in range(len(testY))]

p\_fpr, p\_tpr, \_ = roc\_curve(testY, random\_probs, pos\_label=1)

plt.plot(p\_fpr, p\_tpr, linestyle='--', color='orange',label="True classes")

ns\_fpr, ns\_tpr, \_ = roc\_curve(testY, predict,pos\_label=1)

plt.plot(ns\_fpr, ns\_tpr, linestyle='--', label='Predicted Classes')

plt.title(algorithm+" ROC Graph")

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive rate')

plt.show()

def custom\_knn\_classifier():

global x\_train, y\_train

KNN = KNeighborsClassifier(n\_neighbors=10,leaf\_size=30,metric='minkowski',) # Create an instance of KNeighborsClassifier

#x\_train\_reshaped = np.array(x\_train).reshape(-1, 1)

#x\_test\_reshaped = np.array(x\_test).reshape(-1, 1)

KNN.fit(x\_train, y\_train)

predict = KNN.predict(x\_test)

p = precision\_score(y\_test, predict, average='macro') \* 100

r = recall\_score(y\_test, predict, average='macro') \* 100

f = f1\_score(y\_test, predict, average='macro') \* 100

a = accuracy\_score(y\_test, predict) \* 100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

text.insert(END, "KNN Precision : " + str(p) + "\n")

text.insert(END, "KNN Recall : " + str(r) + "\n")

text.insert(END, "KNN FMeasure : " + str(f) + "\n")

text.insert(END, "KNN Accuracy : " + str(a) + "\n\n")

rocGraph(y\_test, predict, "KNN")

# Compute confusion matrix

cm = confusion\_matrix(y\_test,predict)

# Compute classification report

report = classification\_report(y\_test,predict)

# Display confusion matrix in the Text widget

text.insert(END, "Confusion Matrix:\n")

text.insert(END, str(cm) + "\n\n")

# Display classification report in the Text widget

text.insert(END, "Classification Report:\n")

text.insert(END, report)

def Randomforestclassifier():

global x\_train, y\_train, x\_test, y\_test

global rf

rf=RandomForestClassifier()

rf.fit(x\_train, y\_train)

predict = rf.predict(x\_test)

p = precision\_score(y\_test, predict, average='macro', zero\_division=0) \* 100

r = recall\_score(y\_test, predict, average='macro', zero\_division=0) \* 100

f = f1\_score(y\_test, predict, average='macro', zero\_division=0) \* 100

a = accuracy\_score(y\_test, predict) \* 100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

# Display precision, recall, F1-score, and accuracy in the Text widget

text.insert(END, "RF Precision: " + str(p) + "\n")

text.insert(END, "RF Recall: " + str(r) + "\n")

text.insert(END, "RF FMeasure: " + str(f) + "\n")

text.insert(END, "RF Accuracy: " + str(a) + "\n\n")

rocGraph(y\_test, predict, "RFC")

# Compute confusion matrix

cm = confusion\_matrix(y\_test, predict)

# Compute classification report

report = classification\_report(y\_test, predict)

# Display confusion matrix in the Text widget

text.insert(END, "Confusion Matrix:\n")

text.insert(END, str(cm) + "\n\n")

# Display classification report in the Text widget

text.insert(END, "Classification Report:\n")

text.insert(END, report)

crop\_classes = ['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',

'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',

'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',

'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee']

def Predict():

filename = filedialog.askopenfilename(initialdir="Dataset")

Dataset = pd.read\_csv(filename)

Dataset=Dataset.values

prediction=rf.predict(Dataset)

for i in range(len(prediction)):

if prediction[i] == 0:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'rice'\n\n")

elif prediction[i] == 1:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'maize'\n\n")

elif prediction[i] == 2:

text.insert(END,"Test Data : "+str(Dataset[i])+" ===> Class: 'chickpea'\n\n")

elif prediction[i] == 3:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'kidneybeans'\n\n")

elif prediction[i] == 4:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'pigeonpeas'\n\n")

elif prediction[i] == 5:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'mothbeans'\n\n")

elif prediction[i] == 6:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'mungbean'\n\n")

elif prediction[i] == 7:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'blackgram'\n\n")

elif prediction[i] == 8:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'lentil'\n\n")

elif prediction[i] == 9:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'pomegranate'\n\n")

elif prediction[i] == 10:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'banana'\n\n")

elif prediction[i] == 11:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'mango'\n\n")

elif prediction[i] == 12:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'grapes'\n\n")

elif prediction[i] == 13:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'watermelon'\n\n")

elif prediction[i] == 14:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'muskmelon'\n\n")

elif prediction[i] == 15:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'apple'\n\n")

elif prediction[i] == 16:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'orange'\n\n")

elif prediction[i] == 17:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'papaya'\n\n")

elif prediction[i] == 18:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'coconut'\n\n")

elif prediction[i] == 19:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'cotton'\n\n")

elif prediction[i] == 20:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'jute'\n\n")

elif prediction[i] == 21:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> Class: 'coffee'\n\n")

else:

text.insert(END,"Test Data : "+str(Dataset[i])+" ====> UNKNOWN\n\n")

def graph():

# Create a DataFrame

df = pd.DataFrame([

['KNN', 'Precision', precision[0]],

['KNN', 'Recall', recall[0]],

['KNN', 'F1 Score', fscore[0]],

['KNN', 'Accuracy', accuracy[0]],

['rf', 'Precision', precision[-1]],

['rf', 'Recall', recall[-1]],

['rf', 'F1 Score', fscore[-1]],

['rf', 'Accuracy', accuracy[-1]],

], columns=['Parameters', 'Algorithms', 'Value'])

# Pivot the DataFrame and plot the graph

pivot\_df = df.pivot\_table(index='Parameters', columns='Algorithms', values='Value', aggfunc='first')

pivot\_df.plot(kind='bar')

# Set graph properties

plt.title('Classifier Performance Comparison')

plt.ylabel('Score')

plt.xticks(rotation=0)

plt.tight\_layout()

# Display the graph

plt.show()

def close():

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='MACHINE LEARNING-BASED CROP RECOMMENDATION SYSTEM FOR ENHANCED YIELD PRODUCTION', justify=LEFT)

title.config(bg='lavender blush', fg='black')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=100,y=5)

title.pack()

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Dataset", command=uploadDataset)

uploadButton.place(x=200,y=100)

uploadButton.config(font=font1)

preprocessButton = Button(main, text="Preprocess Dataset", command=preprocessDataset)

preprocessButton.place(x=500,y=100)

preprocessButton.config(font=font1)

knnButton = Button(main, text="KNeighborsClassifier", command=custom\_knn\_classifier)

knnButton.place(x=500,y=150)

knnButton.config(font=font1)

LRButton = Button(main, text="Randomforestclassifier", command=Randomforestclassifier)

LRButton.place(x=200,y=150)

LRButton.config(font=font1)

predictButton = Button(main, text="Prediction", command=Predict)

predictButton.place(x=500,y=200)

predictButton.config(font=font1)

graphButton = Button(main, text="Comparison Graph", command=graph)

graphButton.place(x=200,y=250)

graphButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)

exitButton.place(x=500,y=250)

exitButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=120)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=300)

text.config(font=font1)

main.config(bg='LightSteelBlue1')

main.mainloop()

**CHAPTER 10**

**RESULTS AND DISCUSSION**

**10.1 IMPLEMENTATION DESCRIPTION**

Importing Libraries:

The script imports necessary libraries like Tkinter for GUI, NumPy and Pandas for data manipulation, Seaborn and Matplotlib for data visualization, Scikit-learn for machine learning algorithms, and other utility libraries.

Global Variables:

Global variables are declared to store essential information like the filename of the dataset, training and testing data, label encoder, dataset, model evaluation metrics (accuracy, precision, recall, fscore), and the trained classifier.

Tkinter Setup:

The main Tkinter window is created with a specific title and geometry.

Buttons and labels are placed within this window to provide a user interface for various actions.

Function Definitions:

uploadDataset(): This function is called when the user clicks the "Upload Dataset" button. It opens a file dialog to allow the user to select a dataset file. Once selected, the dataset is loaded into memory and displayed in the text widget.

preprocessDataset(): This function preprocesses the uploaded dataset. It handles missing values by filling them with zeros, encodes categorical labels using LabelEncoder, displays a count plot of the label distribution, and splits the data into training and testing sets.

rocGraph(): This function plots the Receiver Operating Characteristic (ROC) curve for the given test data and predictions. It helps visualize the performance of the classifier.

custom\_knn\_classifier(): This function trains a K-Nearest Neighbors (KNN) classifier on the preprocessed dataset. It then evaluates the classifier's performance using various metrics like precision, recall, F1-score, accuracy, and displays the confusion matrix and classification report.

Randomforestclassifier(): This function trains a Random Forest classifier on the preprocessed dataset. It evaluates the classifier's performance similar to the KNN classifier and displays the results.

Predict(): This function allows the user to upload a test dataset file and make predictions using the trained Random Forest classifier. It displays the predicted class for each test data point.

graph(): This function compares the performance of the KNN and Random Forest classifiers based on various evaluation metrics. It creates a bar graph to visualize and compare the performance of the two algorithms.

close(): This function closes the Tkinter window when the user clicks the "Exit" button.

GUI Components:

Buttons, labels, and a text widget are created and placed within the Tkinter window to provide a user-friendly interface for dataset upload, preprocessing, model training, prediction, and result display.

Main Loop:

The script enters the Tkinter event loop using the main.mainloop() function. This loop handles user interactions and GUI updates, ensuring the smooth functioning of the application.

**10.2 DATASET DESCRIPTION**

The dataset contains agricultural-related information with various features related to the cultivation of different crops. Here are columns in the dataset:

* N: This column appears to represent the nitrogen content in the soil, which is an essential nutrient for plant growth.
* P: This column likely represents the phosphorus content in the soil, another crucial nutrient for plant development.
* K: This column could represent the potassium content in the soil, which is important for various plant processes.
* Temperature: This column represents the temperature of the environment in which the crops are grown. Temperature significantly affects plant growth and development.
* Humidity: This column likely represents the humidity level of the environment. Humidity can influence plant transpiration and overall moisture conditions.
* pH: This column represents the pH level of the soil. Soil pH affects nutrient availability to plants.
* Rainfall: This column could represent the amount of rainfall in the region where the crops are cultivated. Rainfall is a critical factor in determining irrigation needs and water availability.
* Label: This column appears to be the target variable or the label that indicates the type of crop being grown. It seems to be a categorical variable representing different types of crops, such as "rice" and "coffee."

The dataset is organized with rows corresponding to different instances or observations, each with values for the different features (N, P, K, Temperature, Humidity, pH, Rainfall) and a label indicating the type of crop. This kind of dataset is used for various machine learning tasks, such as classification (predicting the crop type) or regression (predicting some numerical outcome based on the features).

**10.3 RESULTS AND DESCRIPTION**

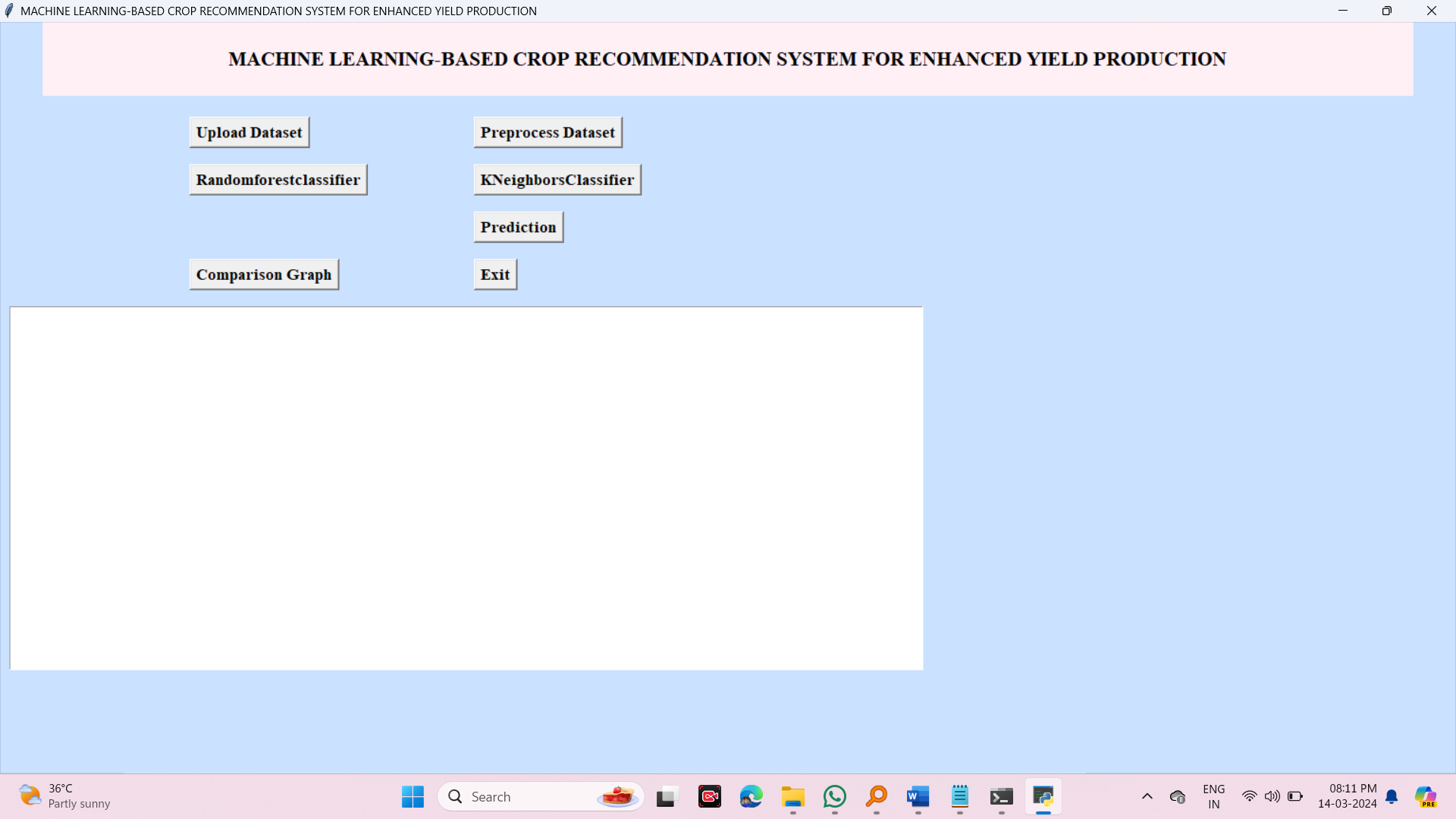


Figure 1: Sample UI used for Crop Recommendation System.

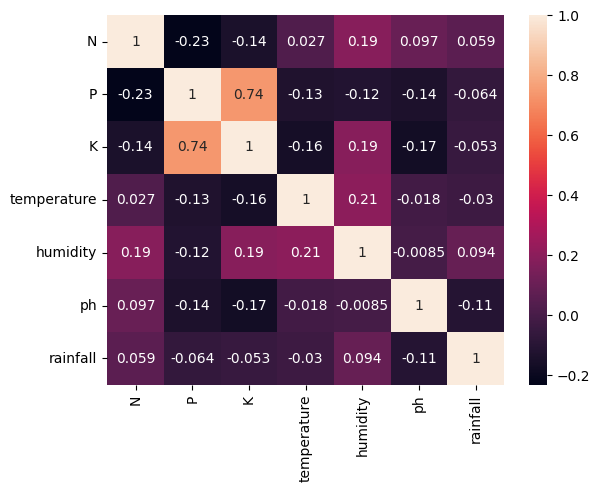


Figure 2: correlation between numerical columns in the Data Frame

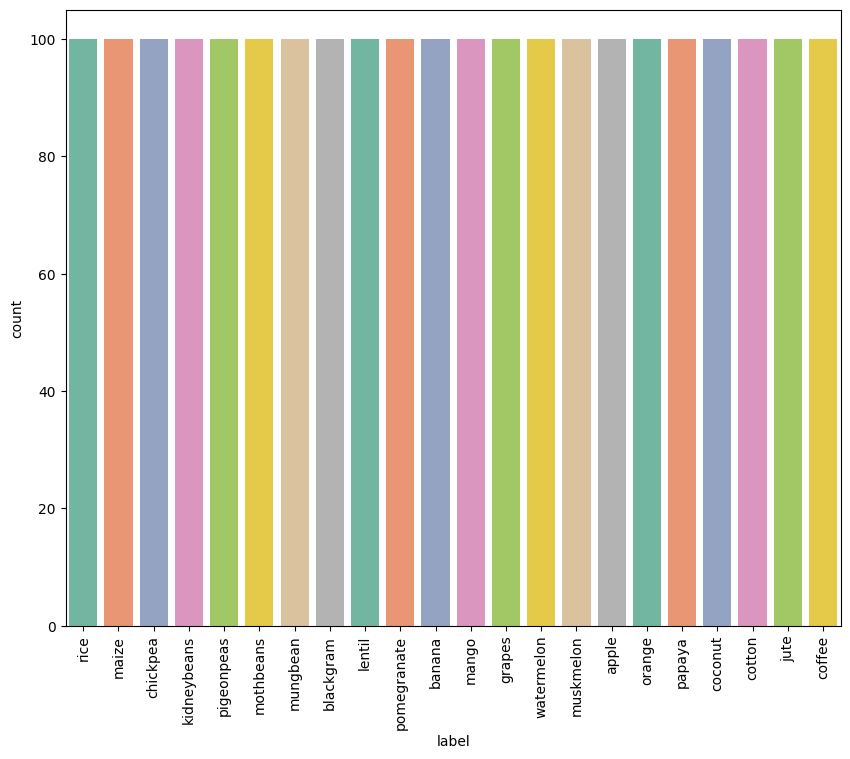


Figure 3: showcase the distribution of categories in the "label" column of the Data Frame

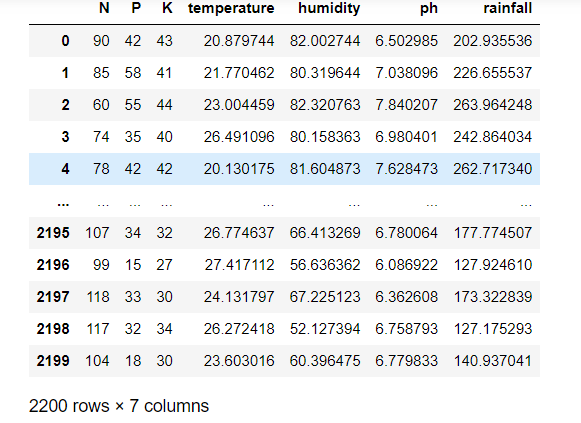


Figure 4: Features columns of the Data Frame

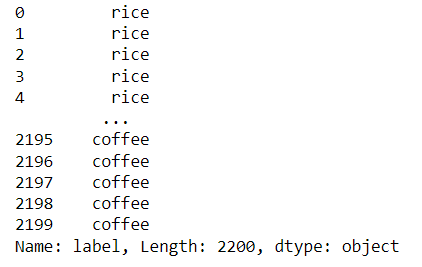


Figure 5: Target column of the Data Frame

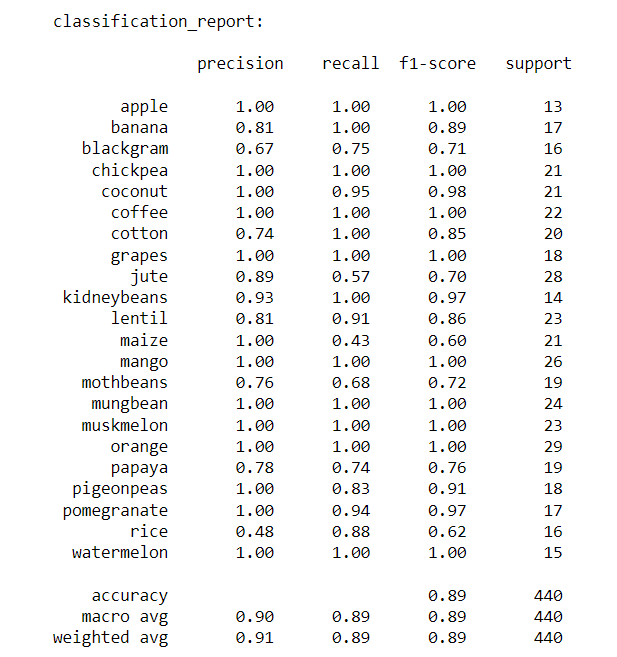


Figure 6: Classification report of Logistic Regression

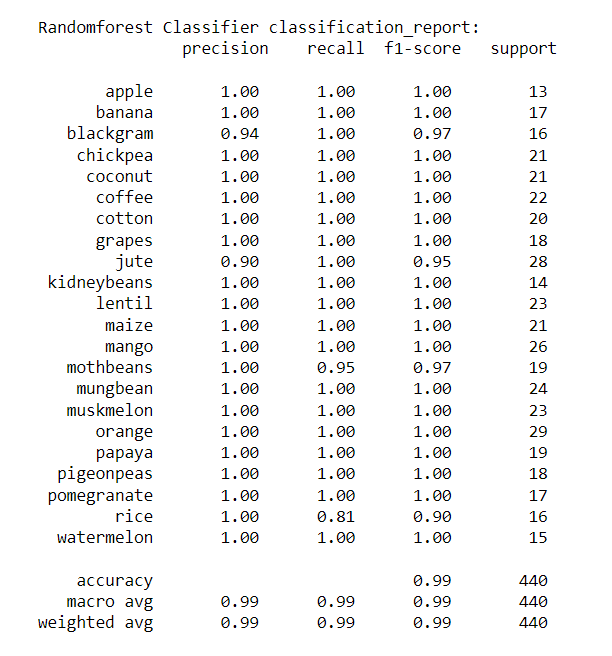


Figure 7: Classification report of RandomForestClassifier

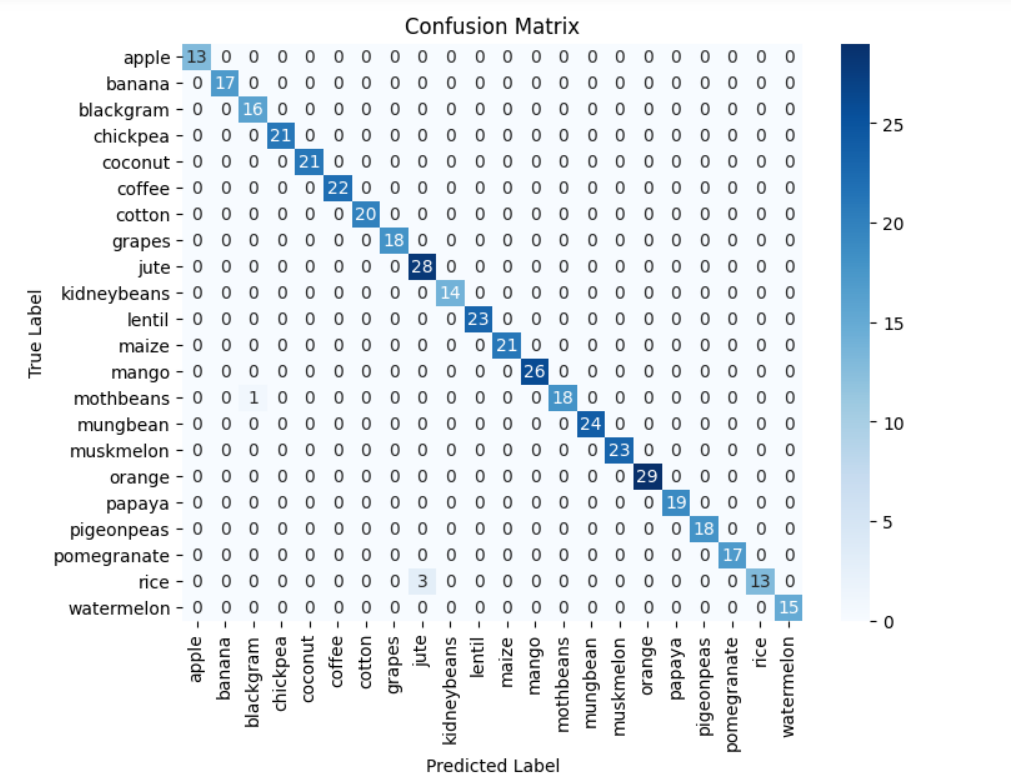


Figure 8: Confusion matrix of RandomForestClassifier

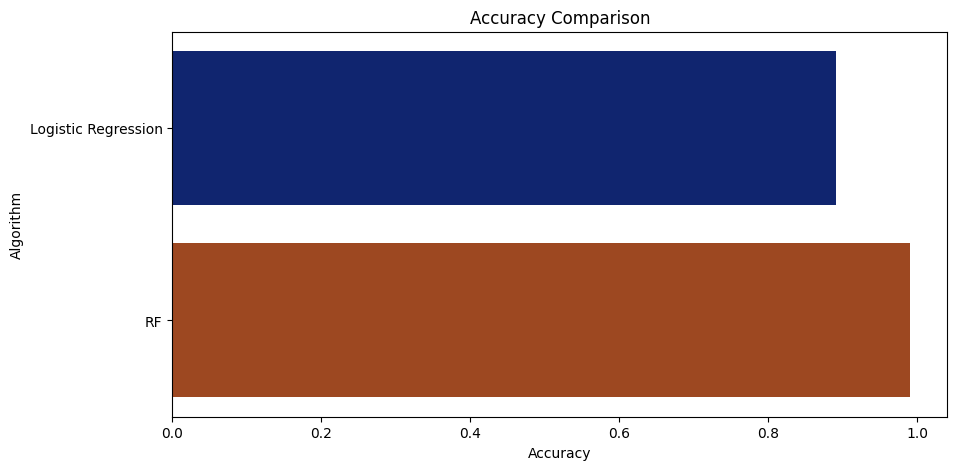


Figure 9: Accuracy Comparison Graph for ml models

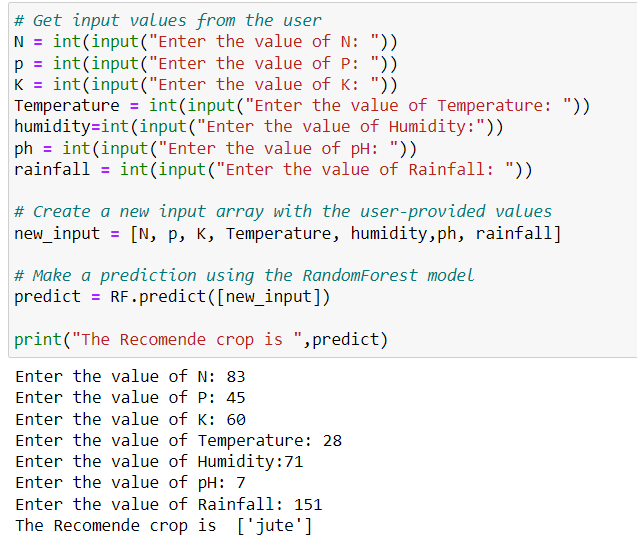


Figure 9: Prediction results for Random Forest classifiers

Table 2: Overall performance comparison of ML models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model name | Accuracy (%) | Precision (%) | Recall (%) | F1-score |
| Random Forest | 99 | 99 | 99 | 99 |
| Logistic Regression | 89 | 90 | 89 | 89 |

**Chapter 11**

**Conclusion and Future scope**

**11.1 Conclusion**

In conclusion, the "Machine Learning-based Crop Recommendation System for Enhanced Yield Production" represents a comprehensive approach to empower farmers with data-driven insights for making informed decisions about crop selection. This system operates on fundamental principles of data science and machine learning. Beginning with data import and preparation, the system recognizes the critical role of data in its operations. It leverages libraries like Pandas and NumPy for efficient data handling, ensuring that historical information about various crops' growth requirements is readily accessible. Exploratory Data Analysis (EDA) forms the foundation for understanding the dataset's characteristics. By utilizing functions like **df.tail()**, **df.size**, and **df.shape**, it gains insights into the data's structure and size. Furthermore, examining the dataset's columns with **df.columns** sets the stage for feature selection and target variable identification.

Data visualization techniques, including count plots and heatmaps, enable a visual understanding of crop distribution and feature correlations. These visuals provide invaluable context, aiding in better-informed decisions during subsequent stages of model development. Feature selection involves extracting relevant factors such as nutrient levels (N, P, K), environmental conditions (temperature, humidity, pH), and rainfall. These features become the basis for training machine learning models, and the target variable ('label') defines the model's prediction goal. The training and evaluation of models, specifically Logistic Regression and Random Forest, embody the heart of this system. Model performance is rigorously assessed using metrics like accuracy, precision, recall, and F1-score. These evaluations ensure that the models can make accurate crop recommendations. A critical aspect is the confusion matrix, which provides a detailed breakdown of prediction results, highlighting true positives, true negatives, false positives, and false negatives. This aids in understanding model strengths and weaknesses. The system also facilitates model selection by comparing accuracies through a bar plot, enabling users to identify the most suitable model for their specific needs. Finally, the ability to make predictions for new data points based on environmental conditions allows farmers to apply the system's insights directly to their farming practices, potentially leading to optimized crop yield production.

**11.2 Future scope**

The "Machine Learning-based Crop Recommendation System for Enhanced Yield Production" holds significant potential for future advancements and broader applications in agriculture and related fields. Here are some potential future scopes and areas of development for this system:

* **Integration of IoT and Sensor Data**: Future iterations of this system can incorporate data from IoT (Internet of Things) sensors placed in farms. These sensors can provide real-time data on soil moisture, temperature, humidity, and other environmental variables. Integrating this data can enhance the accuracy of crop recommendations by accounting for current conditions.
* **Advanced Machine Learning Models**: While Logistic Regression and Random Forest are effective, future research can explore more advanced machine learning algorithms, including deep learning techniques like neural networks. These models may capture more intricate relationships in agricultural data.
* **Personalized Recommendations**: Develop personalized crop recommendations for individual farms. Consider historical data specific to each farm, allowing the system to adapt to unique conditions and farming practices.
* **Crop Disease Detection**: Expand the system's capabilities to include the detection of crop diseases and pests. Machine learning can be used to identify disease symptoms from images of crops and suggest appropriate interventions.
* **Climate Change Adaptation**: As climate change continues to impact agriculture, the system can be adapted to consider changing climate patterns. It can provide recommendations that are resilient to climate variations and extreme weather events.
* **Mobile Applications**: Create user-friendly mobile applications for farmers to access recommendations and input data directly from the field. This would make the system more accessible and practical for everyday use.
* **Data Sharing and Collaboration**: Enable farmers to share their anonymized data with agricultural research institutions, creating a collaborative network that can lead to more comprehensive and accurate recommendations.
* **Crop Rotation Planning**: Extend the system to include recommendations for crop rotation, which can improve soil health and reduce the risk of pests and diseases.
* **Market Integration**: Integrate market data and price predictions to help farmers make decisions not only about what to plant but also when and where to sell their crops for maximum profit.
* **Global Scalability**: Adapt the system for use in different regions and countries by considering local variations in soil types, climate, and crop varieties. Translate the system into multiple languages to make it accessible worldwide.
* **Machine Learning Explainability**: Improve the interpretability of machine learning models used in the system, ensuring that farmers can understand why specific recommendations are made. This can build trust in the system's recommendations.
* **Education and Training**: Offer training and educational resources to farmers on how to use the system effectively and understand the importance of data-driven agriculture.
* **Environmental Sustainability**: Incorporate sustainability metrics into crop recommendations, encouraging environmentally friendly farming practices.

**REFERENCES**

1. Kilaru, Ruthvik, and Kommisetti Murthy Raju. "Prediction of maize leaf disease detection to improve crop yield using machine learning based models." 2021 4th International Conference on Recent Trends in Computer Science and Technology (ICRTCST). IEEE, 2022.
2. Kheir, Ahmed MS, et al. "Machine learning-based cloud computing improved wheat yield simulation in arid regions." Computers and Electronics in Agriculture 203 (2022): 107457.
3. Pham, Hoa Thi, Joseph Awange, and Michael Kuhn. "Evaluation of three feature dimension reduction techniques for machine learning-based crop yield prediction models." Sensors 22.17 (2022): 6609.
4. Khan, Rijwan, et al. "Machine learning-based agriculture." Application of Machine Learning in Agriculture. Academic Press, 2022. 3-27.
5. Manjula, E., and S. Djodiltachoumy. "Efficient prediction of recommended crop variety through soil nutrients using deep learning algorithm." Journal of Postharvest Technology 10.2 (2022): 66-80.
6. Gopi, P. S. S., and M. Karthikeyan. "Red fox optimization with ensemble recurrent neural network for crop recommendation and yield prediction model." Multimedia Tools and Applications (2023): 1-21.
7. Nihar, Ashmitha, N. R. Patel, and Abhishek Danodia. "Machine-Learning-Based Regional Yield Forecasting for Sugarcane Crop in Uttar Pradesh, India." Journal of the Indian Society of Remote Sensing 50.8 (2022): 1519-1530.
8. Li, Kai-Yun, et al. "Toward automated machine learning-based hyperspectral image analysis in crop yield and biomass estimation." Remote Sensing 14.5 (2022): 1114.
9. Sagana, C., et al. "Machine Learning-Based Crop Recommendations for Precision Farming to Maximize Crop Yields." 2023 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2023.
10. Ennaji, Oumnia, Leonardus Vergutz, and Achraf El Allali. "Machine learning in nutrient management: A review." Artificial Intelligence in Agriculture (2023).
11. Prakash, R. Varun, M. Mohamed Abrith, and S. Pandiyarajan. "Machine learning based crop suggestion system." 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2022.
12. Indira, D. N. V. S. L. S., et al. "KRISHI RAKSHAN-A Machine Learning based New Recommendation System to the Farmer." 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2022.
13. Jeevaganesh, R., D. Harish, and B. Priya. "A Machine Learning-based Approach for Crop Yield Prediction and Fertilizer Recommendation." 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI). IEEE, 2022.
14. Wen, Guoqi, et al. "Optimizing machine learning-based site-specific nitrogen application recommendations for canola production." Field Crops Research 288 (2022): 108707.
15. SSL, Durai Arumugam, R. Praveenkumar, and V. Balaji. "An Intelligent Crop Recommendation System using Deep Learning." International Journal of Intelligent Systems and Applications in Engineering 11.10s (2023): 423-428.
16. Chowdhary, Muppala Sunny, et al. "InteliCrop: An Ensemble Model to Predict Crop using Machine Learning Algorithms." 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI). IEEE, 2022.
17. Suresh, Chalumuru, et al. "Paddy Crop Monitoring System Using IoT and Deep Learning." Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2021. Singapore: Springer Nature Singapore, 2022. 791-807.
18. Raviraja, S., et al. "Machine learning based mobile applications for autonomous fertilizer suggestion." 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, 2022.
19. Mamatha, V., and J. C. Kavitha. "Machine learning based crop growth management in greenhouse environment using hydroponics farming techniques." Measurement: Sensors 25 (2023): 100665.
20. Oikonomidis, Alexandros, Cagatay Catal, and Ayalew Kassahun. "Hybrid deep learning-based models for crop yield prediction." Applied artificial intelligence 36.1 (2022): 2031822.
21. Apat, Shraban Kumar, et al. "State of the art of ensemble learning approach for crop prediction." Next Generation of Internet of Things: Proceedings of ICNGIoT 2022 (2022): 675-685.
22. Kumar, Prashant, et al. "Crop recommendation using machine learning algorithms." 2023 International Conference on Disruptive Technologies (ICDT). IEEE, 2023.
23. Sandhya, K. Gnana, et al. "Crop Recommendation System Using Ensembling Technique." 2022 International Conference on Breakthrough in Heuristics And Reciprocation of Advanced Technologies (BHARAT). IEEE, 2022.
24. Muhammed, Dalhatu, et al. "A User-friendly AIoT-Based Crop Recommendation system (UACR): concept and architecture." 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). IEEE, 2022.
25. Venkatesan, SaravanaKumar, et al. "A machine learning based model for energy usage peak prediction in smart farms." Electronics 11.2 (2022): 218.