Crop Yield Prediction Using Machine Learning Techniques

A Project Report submitted

in fulfillment of the requirements for the degree of

Bachelors of Technology in

Computer Science and Engineering

Submitted by

Levid Kumar (202025017)

Surjyadeep Baruah (202025031)

Tanzimur Rohman (202025034)

B. Tech CSE, 8th Semester

Royal School of Engineering and Technology

Under the guidance of

Dr. Israfil Hussain

Associate Professor

Royal School of Engineering and Technology
THE ASSAM ROYAL GLOBAL UNIVERSITY

GUWAHATI: 781035

Session: 2023-24

ROYAL GLOBAL UNIVERSITY
GUWAHATI

CERTIFICATE OF APPROVAL

It is certified that the work contained in the report entitled **" Crop Yield Prediction Using Machine Learning Techniques"** by **Levid Kumar** bearing Roll No 202025017, **Surjyadeep Baruah** bearing Roll No 202025031 and **Tanzimur Rohman** bearing Roll No 202025034 of **B. Tech CSE, 8th Semester** under the **Department of Computer Science and Engineering**, **Royal School of Engineering and Technology**, The Assam Royal Global University, Guwahati, Assam for the fulfilment of the degree of **Bachelors of Engineering** has been carried out under my supervision and that work has not been submitted elsewhere for a degree.

Project Guide:

Dr. Israfil Hussain

(Associate Professor)

Signature of the External Examiner

Name of the External Examiner

Date:

Place: Guwahati

ROYAL GLOBAL UNIVERSITY
GUWAHATI

FORWARDING CERTIFICATE

It is certified that the work contained in the report entitled "**Crop Yield Prediction using Machine Learning Techniques**" by **Levid Kumar** bearing Roll No 202025017, **Surjyadeep Baruah** bearing Roll No 202025031 and **Tanzimur Rohman** bearing Roll No 202025034 of **B. Tech CSE, 8th Semester** under the **Computer Science and Engineering**, **Royal School of Engineering and Technology**,under the guidance of **Dr. Israfil Hussain, Associate Professor** has been presented in a manner satisfactory to permit its acceptance as a prerequisite to the degree for which has been submitted.

Date:
Place: Guwahati

Dr. Ishita Chakraborty
(Associate Professor)
Head of Department, Royal School Of Engineering and Technology

DECLARATION

We, **Levid Kumar** bearing Roll No 202025017, **Surjyadeep Baruah** bearing Roll No 202025031 and **Tanzimur Rohman** bearing Roll No 202025034 hereby declare that this project work entitled "**Crop Yield Prediction Using Machine Learning Techniques**" was carried out by us under the guidance & supervision of **Dr. Israfil Hussain, Associate Professor**. This project work is submitted during the academic session 2023-24. This work or no part of it has been submitted elsewherefor any other purpose till date.

Date:

Place: Guwahati

Signature

Levid Kumar	**Surjyadeep Baruah**	**Tanzimur Rohman**
Roll No. 202025017	**Roll No. 202025031**	**Roll No. 202025034**

ACKNOWLEDGEMENT

We take the opportunity to express our sincere gratitude to all those who supported us throughout this project/dissertation work. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work.

We convey our special thanks to Dr. Israfil Hussain for their support and guidance at during our project/dissertation work.

Our special thanks also go to our Faculty Guide and other faculty members for their kind support for the successful completion of our project/dissertation.

Thank you

Levid Kumar (202025017)

Surjyadeep Baruah (202025031)

Tanzimur Rohman(202025034)

ABSTRACT

Globally, agriculture has an important responsibility to improve the economic contribution of the nation. However, most agricultural fields are still underdeveloped due to a lack of ecosystem management techniques because of these problems, crop production does not improve, which affects the agricultural economy. Therefore, the development of agricultural based on the yield forecast. To avoid this problem, agricultural sectors need to predict the yield of a given data set using machine learning techniques. Data exploration with coordinated ML techniques. A comparative study was conducted between machine learning algorithms to determine which algorithm was most accurate in predicting the best performance. Here we predict the yield when a particular yield is chosen differently. predict the yield of all crops using the ph level, rainfall, temperature, nitrogen, phosphorous and potassium.

Machine learning is an important decision support tool for crop yield prediction, including supporting decisions on what crops to grow and what to do during the growing season of the crops. Several machine learning algorithms have been applied to support crop yield prediction research. In this study, we performed a Literature Review to extract and synthesize the algorithms and features that have been used in crop yield prediction studies. The main focus of these algorithms is to help optimize crop production and reduce waste through informed decisions regarding planting, watering, and harvesting crops.

The results show that we can achieve a classification accuracy of 99.9% using the Random Forest Classifier and 99.8% using Decision Tree algorithms. These results will indicate an increase in production rates and reduce the effective cost for the farms, leading to more resilient infrastructure and sustainable environments. Moreover, the findings we obtained in this study can also help future farmers increase crop production efficiency.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In developing countries, agriculture is considered the most important source of income for many people. Today, the growth of agriculture is linked by several innovations, environments, technologies and civilizations. In addition, the use of information technology can change the decision-making conditions and thus farmers can produce in the best way. Machine learning techniques related to agriculture are used in the decision making process. Today we have used machine learning which is developed to predict yield or crop yield as there are various types of data in agriculture such as soil, yield and weather data. Plant growth prediction has been proposed for effective plant production monitoring using machine learning techniques. It is also suitable for automatic cultivation which is suitable for farmers who are looking for experts to take suggestions about the crop suitable for the specific location of their land and do not want. forgetting all the cultivation steps during the process. Although expert opinion is the most convenient way, this application is designed to provide an accurate solution in the fastest possible way. The main goal of this research is to bring the cultivation process one step closer to the digital platform.

1.1 Domain Overview

The goal of machine learning is to predict the future based on past data. Machine learning (ML) is a type of artificial intelligence (AI) that gives computers the ability to learn without special programming. Machine learning focuses on the development of computer programs that can change when exposed to new information and the basics of machine learning, the implementation of a simple machine learning algorithm using python. The training and prediction process involves the use of special algorithms. It feeds the training data to the algorithm, and the algorithm uses this training data to predict new test data. Machine learning can be broadly divided into three categories. There is supervised learning, unsupervised learning and reinforcement learning. The guided learning program receives both input data and response records for learning, the data must be named by a person in advance. Unsupervised learning is not labels. It worked for the learning algorithm. This algorithm needs to figure out the grouping of the input data. Finally, reinforcement learning dynamically interacts with its environment and receives positive or negative feedback to improve its performance. Data scientists use many different machine learning algorithms to find patterns in Python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on how they "learn" from the data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Categories are sometimes called items/tags or categories. Predictive modelling of classification must be done to approximate a mapping function from input variables (X) to discrete output variables (y). In machine learning and statistics, classification is a supervised learning method where a computer program learns from a given data input and then uses that learning to classify a new observation.

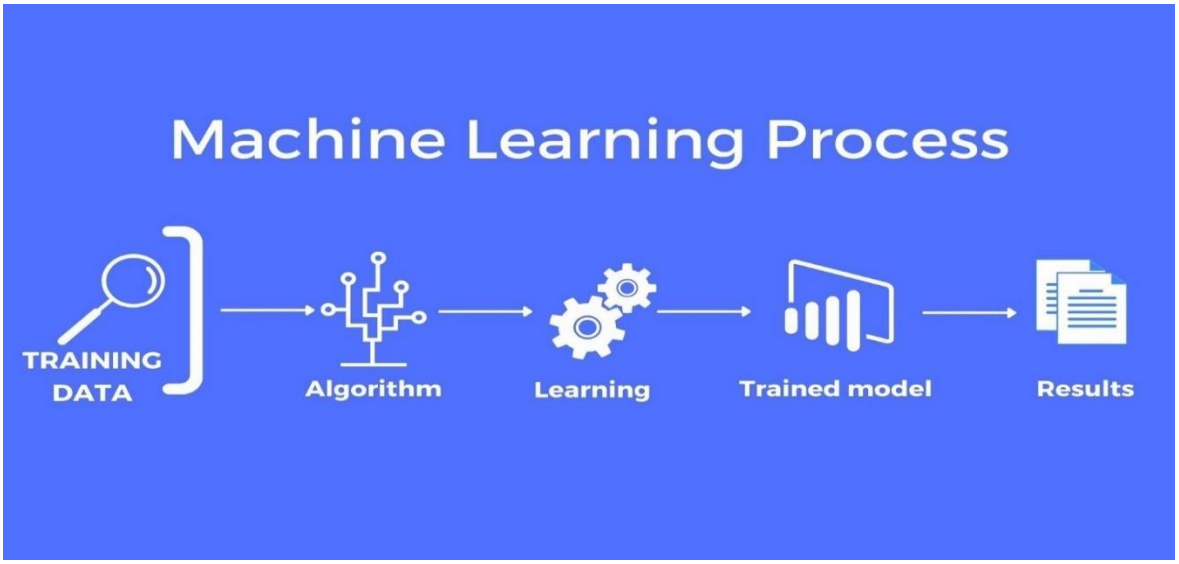

Fig 1.1 Process of Machine Learning

Supervised machine learning is the bulk of practical machine learning that uses supervised learning. Supervised learning means that the input variables (X) and the output variables (y) are y = f(X). The goal is to approximate the mapping function so well that when you have new input data (X) you can predict the data output variables (y). Supervised machine learning algorithm techniques include logistic regression, multi-class classification, decision trees and support vector machines, etc. Supervised learning assumes that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be grouped into classification problems. The purpose of this task is to create a summary model that can predict the value of a dependent attribute based on the attribute variables. The difference between these two tasks is that the dependent attribute is numeric for classification purposes. A classification model attempts to make inferences from observed values. Based on one or more inputs, a classification model attempts to predict the value of one or more outcomes. A classification problem is when the outcome variable is a category such as "red" or "blue". Agriculture is one of the most important occupations in our country. It is the largest economic sector and plays an important role in the overall development of the country. About 60% of the country is used for agriculture to meet the needs of 1.2 billion people. So, modernization of agriculture is very important and will lead the farmers of our country to profit. Data analytics (DA) is the process of examining data sets to draw conclusions from the information they contain, increasingly using specialized systems and software. The forward yield forecast was made taking into account the farmer's experience with a particular field and crop. However, because the conditions are changing very quickly every day, farmers have to grow more and more crops. In the current situation, many of them do not have enough knowledge about new crops and are not fully aware of the benefits of growing them. Farm productivity can also be increased by understanding and predicting crop performance under different environmental conditions. Thus, the proposed system takes the soil nutrients such as nitrogen, phosphorus, potassium and rainfall. This static data is information obtained from different websites about crop production and requirements of different crops. It uses machine learning and predictive algorithm to detect a pattern in the data and then process it according to the input conditions.

1.2 Motivation

Combining soil composition and environmental parameters with machine learning algorithms enables real-time, tailored crop suggestions. Key factors considered include nitrogen, potassium, phosphorus, pH, rainfall, humidity, and temperature. Increased Food Demand With the global population rising, there is a heightened need to increase food production. Accurate yield predictions help optimize agricultural practices to meet this demand.

Machine learning can analyze vast amounts of data to provide insights on the best use of resources like water, fertilizers, and pesticides, leading to more efficient and sustainable farming practices. As climate conditions become more unpredictable, machine learning models can help farmers anticipate and adapt to changes, minimizing the impact on crop yields. Accurate yield predictions can help farmers make better financial decisions, reducing waste and increasing profitability through better planning and market strategies. Machine learning facilitates precision agriculture, allowing for more precise interventions tailored to specific crop needs, improving yield and reducing environmental impact. Governments and organizations can use yield predictions to inform policy-making and strategic planning, ensuring food security and stability.

By leveraging machine learning, the agriculture industry can enhance productivity, sustainability, and resilience in the face of various challenges.

1.3 Proposed System

To make it simple and which can be directly used by the farmer this work uses simple factors of soil nutrients like Nitrogen, Phosphorus, Potassium and Rainfall.

The system prepared predict crops yield of almost all kinds of crops that are planted in India. The model makes the usage of simple parameters like Nitrogen, Potassium, Phosphorus etc. and the user can predict the yield of the crop he or she wants to.

We have to enter the parameters such as Nitrogen, pH value, Phosphorus. After submitting the inputs, it will output the best crop to be planted.

STEPS

- Define a problem
- Preparing data
- Evaluating algorithms
- Improving results
- Predicting results

1.4 OBJECTIVE

- **Data Validation**

Data validation is the process of verifying and validating data that is collected before it is used. Any type of data handling task, whether it is gathering data, analyzing it, or structuring it for presentation, must include data validation to ensure accurate results. It catches changes in the data coming into the machine learning pipeline before it reaches the time-consuming preprocessing and training steps. Our goal is to automate our machine learning model updates, validating our data is essential.

- **Data Cleaning/ Preparing**

Data preparation is the process of preparing raw data so that it is suitable for further processing and analysis. Key steps include collecting, cleaning, and labeling raw data into a form suitable for machine learning (ML) algorithms and then exploring and visualizing the data.

- **Data Visualization**

Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

- **Using algorithms**

1.) Random Forest Classifier

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.

2.) Decision Tree Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

3.) Logistic Regression Classifier

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyze the relationship between two data factors. Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value.

It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

4.) K-Nearest Neighbour Classifier

KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

1.5 HARDWARE AND SOFTWARE REQUIREMENTS

1.5.1 Software Requirements

Operating System: Windows

Tool: Anaconda with Jupyter Notebook

1.5.2 Hardware requirements

Processor: Pentium IV/III

Hard disk: minimum 80 GB

RAM: minimum 2 GB

CHAPTER 2
LITERATURE SURVEY

2.1 GENERAL

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

2.2 REVIEW OF LITERATURE SURVEY:

Title:A GNN-RNN Approach for Harnessing Geospatial and Temporal Information: Application to Crop Yield Prediction

Author: Joshua Fan (Cornell University); Junwen Bai (Cornell University); Zhiyun Li (Cornell University); Ariel Ortiz-Bobea (Cornell); Carla P Gomes (Cornell University)

Year: 2023

Description:

The authors developed a model that simultaneously leverages spatial dependencies and temporal patterns in the data. This approach helps in capturing the complex relationships and interactions between different geographical regions and their historical crop yield data. GNNs are employed to process the spatial data, representing different regions as nodes in a graph and capturing the spatial dependencies among them. This enables the model to understand how the crop yields in one region can influence yields in neighbouring regions. Recurrent Neural Network (RNNs) are used to handle the temporal aspect of the data, analyzing the sequence of crop yields over time. This helps in recognizing trends, seasonal patterns, and temporal dependencies that affect crop yields. The model was trained and tested on a large dataset that includes crop yield data from 2000 counties in the United States, spanning the years from 1981 to 2019. The data also incorporated various environmental and climatic variables that impact crop production.

The proposed GNN-RNN model demonstrated superior performance in crop yield prediction compared to traditional models. The results indicated that the integration of spatial and temporal information significantly improves the accuracy of yield predictions. This approach can be applied to various crops and regions, providing valuable insights for farmers, policymakers, and researchers. It supports better decision-making in agricultural management, resource allocation, and planning by providing more accurate yield forecasts.

Title: Rice Crop Yield Prediction in India using Support Vector Machines, BayesNet, and Multilayer Perceptron

Author: Niketa Gandhi, Leisa J. Armstrong, Owaiz Petkar, and Amiya Kumar Tripathy.

Year: 2021

Description: The primary objective of this research is to enhance the accuracy of rice yield predictions by employing different machine learning models, namely Support Vector Machines (SVM), BayesNet, and Multilayer Perceptron (MLP). Accurate predictions are crucial for improving agricultural productivity and planning. The study uses historical data on rice yields along with various climatic and environmental factors. These factors include temperature, rainfall, soil properties, and other agronomic parameters, which are critical determinants of crop yield

The collected data is divided into training and validation sets to build and test the predictive models. Each model is trained on the training dataset and validated against the validation dataset to evaluate its performance. The models are assessed using various performance metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2) to determine the accuracy and robustness of the predictions. The paper reports that each of the three models has its strengths and weaknesses in predicting rice yields.

Title: Distinguishing Heavy-Metal Stress Levels in Rice Using Synthetic Spectral Index Responses to Physiological Function Variations

Author: Ming Jin, Xiangnan Liu, Ling Wu, and Meiling Liu

Year: 2016

Description:

Accurately assessing the heavy-metal contamination in crops is crucial to food security. This study provides a method to distinguish heavy-metal stress levels in rice using the variations of two physiological functions as discrimination indices, which are obtained by assimilation of remotely sensed data with a crop growth model. Two stress indices, which correspond to dailytotalCO2 assimilationanddry-matterconversioncoefficient were incorporated into the World Food Study (WOFOST) crop growth model and calculated by assimilating the model with leaf area index (LAI), which was derived from time-series HJ1- CCD data. The stress levels are not constant with rice growth; thus, to improve the reliability, the two stress indices were obtained at both the first and the latter half periods of rice growth.

To compare the stress indices of different stress levels, a synthetic stress index was

established by combining the two indices; then, three types of stress index discriminant spaces based on the synthetic index of different growth periods were constructed, in which the two-dimensional discriminant space based on two growth periods showed the highest accuracy, with a misjudgement rate of 4.5%. When the discrimination rules were applied at a regional scale, the average correct discrimination rate was 95.0%.

Title: Design and Characterization of a Fringing Field Capacitive Soil Moisture Sensor

Author: Manash Protim Goswami, Babak Montazer, and Utpal Sarma, Member, IEEE

Year: 2018

Description:

The optimization and implementation of a fringing field capacitive soil moisture sensor using the printed circuit board technology. It includes the analysis of a novel configuration of an interdigital sensor for measuring soil moisture with two existing configurations. The optimized designs were simulated by using a 3-D finite-element method and fabricated by using a copper clad board. The performance of the fabricated sensors was evaluated using four soil samples collected from different locations. The observations were compared with the standard gravimetric method to evaluate the soil water content of the samples. The characterization method and the results of the whole sensing system are discussed in terms of calibration, dynamic test, and repeatability.

Title: Improving Spring Maize Yield Estimation at Field Scale by Assimilating Time-Series HJ-1 CCD Data into the WOFOST Model Using a New Method with Fast Algorithms

Author: Zhiqiang Cheng, Jihua Meng * and Yiming Wang

Year: 2016

Description:

Field crop yield prediction is crucial to grain storage, agricultural field management, and national agricultural decision-making. Currently, crop models are widely used for crop yield prediction. However, they are hampered by the uncertainty or similarity of input parameters when extrapolated to field scale. Data assimilation methods that combine crop models and remote sensing are the most effective methods for field yield estimation. In this study, the World Food Studies (WOFOST) model is used to simulate the growing process of spring maize. Common assimilation methods face some difficulties due to the scarce, constant, or similar nature of the input parameters. For example, yield spatial heterogeneity simulation, coexistence of common assimilation methods and the nutrient module, and time cost are relatively important limiting factors. To address the yield simulation problems at field scale, a simple yet effective method with fast algorithms is presented for assimilating the time-series HJ-1 A/B data into the WOFOST model in order to improve the spring maize yield simulation. First, the WOFOST model is calibrated and validated to obtain the precise mean yield. Second, the time-series leaf area index (LAI) is calculated from the HJ data using an empirical regression model. Third, some fast algorithms are developed to complete assimilation. Finally, several experiments are conducted in a large farmland (Hongxing) to evaluate the yield simulation results. In general, the results indicate that the proposed

method reliably improves spring maize yield estimation in terms of spatial heterogeneity simulation ability and prediction accuracy without affecting the simulation efficiency.

Title: A generalized regression-based model for forecasting winter wheat yields in Kansas and Ukraine using MODIS data

Author: Becker-Reshef, E. Vermote, M. Lindeman

Year: 2010

Description:

As new remote sensing instruments and data become available their utility for improving established terrestrial monitoring tasks need to be evaluated. An empirical, generalized remotely sensed based yield model was developed and successfully applied at the state level in Kansas using daily, high quality 0.05° NDVI time series data to drive the regression model, a percent crop mask as a filter to identify the purest winter wheat pixels, and USDA NASS county crop statistics for model calibration. The model predictions of production in Kansas closely matched the USDA/NASS reported numbers with a 7% error. This empirical regression model that was developed in Kansas was successfully applied directly in Ukraine. The model forecast winter wheat production in Ukraine six weeks prior to harvest with a 10% error of the official production numbers. In 2009 the model was run in real-time in Ukraine and forecast production within 7% of the official statistics which were released after the harvest. Wheat is one of the key cereal crops grown worldwide, providing the primary caloric and nutritional source for millions of people around the world. In order to ensure food security and sound, actionable mitigation strategies and policies for management of food shortages, timely and accurate estimates of global crop production are essential. It combines a new BRDF-corrected, daily surface reflectance dataset developed from NASA's Moderate resolution Imaging Spectro-radiometer (MODIS) with detailed official crop statistics to develop an empirical, generalized approach to forecast wheat yields. The first step of this study was to develop and evaluate a regression-based model for forecasting winter wheat production in Kansas. This regression-based model was then directly applied to forecast winter wheat production in Ukraine. The forecasts of production in Kansas closely matched the USDA/NASS reported numbers with a 7% error. The same regression model forecast winter wheat production in Ukraine within 10% of the official reported production numbers six weeks prior to harvest. Using new data from MODIS, this method is simple, has limited data requirements, and can provide an indication of winter wheat production shortfalls and surplus prior to harvest in regions where minimal ground data is available.

Title: Machine Learning Approaches to Corn Yield Estimation Using

Satellite Images and Climate Data: A Case of Iowa State

Author: Kim, NariLee, Yang-Won
Year: 2016
Description:

Machine learning, which is an efficient empirical method for classification and prediction, is another approach to crop yield estimation. It described the corn yield estimation in Iowa

State using four machine learning approaches such as RF (Random Forest), ERT (Extremely Randomized Trees) and DL (Deep Learning). Also, comparisons of the validation statistics among them were presented. To examine the seasonal sensitivities of the corn yields, three period groups were set up: (1) MJJAS (May to September), (2) JA (July and August) and (3) OC (optimal combination of month). In overall, the DL method showed the highest accuracies in terms of the correlation coefficient for the three period groups. The accuracies were relatively favourable in the OC group, which indicates the optimal combination of month can be significant in statistical modelling of crop yields. The differences between our predictions and USDA (United States Department of Agriculture) statistics were about 6-8 %, which shows the machine learning approaches can be a viable option for crop yield modelling. Monitoring crop yield is important for many agronomy issues such as farming management, food security and international crop trade. Because South Korea highly depends on imports of most major grains except for rice, reasonable estimations of crop yields are more required under recent conditions of climate changes and various disasters. Remote sensing data has been widely used in the estimation of crop yields by employing statistical methods such as regression model. It conducted multivariate regression analyses to estimate corn and soybean yields in Iowa using MODIS (Moderate Resolution Imaging Spector radiometer) NDVI (Normalized Difference Vegetation Index), climate factors and soil moisture and presented regression models for the estimation of winter wheat yields using MODIS NDVI and weather data in Shandong, China. It estimated corn and soybean yields using several MODIS products and climatic variables for Midwestern United States (US) and represented prediction errors of about 10 %. It built multiple regression models using MODIS NDVI and weather data to estimate rice yields in North Korea and showed the RMSE of 0.27 ton/ha. Most of the previous studies are based on the multivariate regression analysis using the relationship between crop yields and agro-environmental factors such as vegetation index, climate variables and soil properties.

Title: Estimation of OrganicMatter Content in Coastal Soil Using Reflectance Spectroscopy Research

Author: ZHENG Guanghui1, Dongryeol RYU2, ∗, JIAO Caixia1 and HONG Changqiao1

Year: 2015

Description:

Rapid determination of soil organic matter (SOM) using regression models based on soil reflectance spectral data serves an important function in precision agriculture. “Deviation of arch” (DOA)-based regression and partial least squares regression (PLSR) are two modelling approaches to predict SOM. However, few studies have explored the accuracy of the DOA based regression and PLSR models. Therefore, the DOA-based regression and PLSR were applied to the visible near-infrared (VNIR) spectra to estimate SOM content in the case of various dataset divisions. A two-fold cross-validation scheme was adopted and repeated 10000 times for rigorous evaluation of the DOA-based models in comparison with the widely used PLSR model. Soil samples were collected for SOM analysis in the coastal area of northern Jiangsu Province, China. The results indicated that both modelling methods provided reasonable estimation of SOM, with PLSR outperforming DOA-based regression

in general. However, the performance of PLSR for the validation dataset decreased more noticeably. Among the four DOA-based regression models, a linear model provided the best estimation of SOM and a cut-off of SOM content (19.76 g kg−1), and the performance for calibration and validation datasets was consistent. As the SOM content exceeded 19.76 g kg−1, SOM became more effective in masking the spectral features of other soil properties to a certain extent. This work confirmed that reflectance spectroscopy combined with PLSR could serve as a non-destructive and cost-efficient way for rapid determination of SOM when hyper spectral data were available. The DOA-based model, which requires only 3 bands in the visible spectra, also provided SOM estimation with acceptable accuracy.

Title: Preliminary Study of Soil Available Nutrient Simulation Using a Modified WOFOST Model and Time-Series Remote Sensing Observations

Author: Zhiqiang Cheng 1,2 ID, Jihua Meng 1,*, Yanyou Qiao 1, Yiming Wang 1,2, Wenquan Dong 1 and Yanxin Han 1,2

Year: 2017

Description: The approach of using multispectral remote sensing (RS) to estimate soil available nutrients (SANs) has been recently developed and shows promising results. This method overcomes the limitations of commonly used methods by building a statistical model that connects RS-based crop growth and nutrient content. However, the stability and accuracy of this model require improvement. In this article, we replaced the statistical model by integrating the World Food Studies (WOFOST) model and time series of remote sensing (TRS) observations to ensure stability and accuracy. Time series of HJ-1 A/B data was assimilated into the WOFOST model to extrapolate crop growth simulations from a single point to a large area using a specific assimilation method. Because nutrient-limited growth within the growing season is required and the SAN parameters can only be used at the end of the growing season in the original model, the WOFOST model was modified. Notably, the calculation order was changed, and new soil nutrient uptake algorithms were implemented in the model for nutrient-limited growth estimation. Finally, experiments were conducted in the spring maize plots of Hongxing Farm to analyze the effects of nutrient stress on crop growth and the SAN simulation accuracy. The results confirm the differences in crop growth status caused by a lack of soil nutrients. The new approach can take advantage of these differences to provide better SAN estimates. In general, the new approach can overcome the limitations of existing methods and simulate the SAN status with reliable accuracy.

2.3 Literature Review Summary

S.No	Paper Title	Year of Publication	Author Name	Contribution
1	A GNN-RNN Approach for Harnessing Geospatial and Temporal Information: Application to Crop Yield Prediction	2023	Joshua Fan (Cornell University), Junwen Bai (Cornell University), Zhiyun Li (Cornell University), Ariel Ortiz-Bobea (Cornell), Carla P Gomes (Cornell University)	This paper introduces a novel graph-based recurrent neural network (GNN-RNN) that incorporates both geographical and temporal data for crop yield prediction. The method was tested across 2000 counties in the US from 1981 to 2019, demonstrating significant improvements over existing models by leveraging the spatial correlations and temporal dynamics in the data.
2	Rice Crop Yield Prediction in India using Support Vector Machines, BayesNet, and Multilayer Perceptron	2021	Niketa Gandhi, Leisa J. Armstrong, Owaiz Petkar, and Amiya Kumar Tripathy.	This paper investigates the use of machine learning algorithms such as Support Vector Machines (SVM), BayesNet, and Multilayer Perceptron (MLP) for predicting rice yields in India. By incorporating various climatic and environmental factors, the study aims to enhance the accuracy of yield predictions.
3	Distinguishing Heavy-Metal Stress Levels in Rice Using Synthetic Spectral Index Responses to Physiological Function Variations	2016	Ming Jin, Xiangnan Liu, Ling Wu, and Meiling Liu	This study presents a method to assess heavy-metal contamination in rice using stress indices derived from physiological functions and remotely sensed data. By integrating these indices into the WOFOST crop growth model with leaf area index (LAI) data, the study distinguishes stress levels at different growth periods. A synthetic stress index improves accuracy, achieving a misjudgement rate of 4.5% and a regional correct discrimination rate of 95.0%.

4	Design and Characterizatio n of a Fringing Field Capacitive Soil Moisture Sensor	2018	Manash Protim Goswami, Babak Montazer, and Utpal Sarma, Member, IEEE	This study optimizes and implements a fringing field capacitive soil moisture sensor using printed circuit board technology. It introduces a new interdigital sensor design and compares it with existing configurations. The designs were simulated with 3-D finite-element methods and fabricated on a copper clad board. Performance was tested on soil samples from various locations and compared to the gravimetric method. The study covers calibration, dynamic testing, and repeatability of the sensor system.
5	Improving Spring Maize Yield Estimation at Field Scale by Assimilating Time-Series HJ-1 CCD Data into the WOFOST Model Using a New Method with Fast Algorithms	2016	Zhiqiang Cheng, Jihua Meng and Yiming Wang	This study improves spring maize yield prediction by integrating the WOFOST crop model with remote sensing data. The WOFOST model is calibrated and validated, then assimilates time-series leaf area index (LAI) data from HJ-1 A/B satellites using fast algorithms. Field experiments demonstrate enhanced spatial heterogeneity and prediction accuracy without reducing efficiency.
6	A generalized regression-based model for forecasting winter wheat yields in Kansas and Ukraine using MODIS data	2010	Becker-Reshef, E. Vermote, M. Lindeman	This study developed an empirical regression model to predict winter wheat yields using high-quality NDVI time series data and USDA NASS crop statistics. The model, tested in Kansas, showed a 7% error margin compared to official reports. It was successfully applied in Ukraine, forecasting yields within 10% accuracy six weeks before harvest, and achieving a 7% error in real-time predictions. This method, utilizing MODIS data, requires minimal ground data and offers timely, accurate estimates of wheat production to support food security and management strategies.

7	Machine Learning Approaches to Corn Yield Estimation Using Satellite Images and Climate Data: A Case of Iowa State	2016	Kim, NariLee, Yang-Won	This study explores machine learning methods for corn yield estimation in Iowa, using Random Forest (RF), Extremely Randomized Trees (ERT), and Deep Learning (DL). It compares validation statistics and examines seasonal sensitivities of corn yields across different period groups. DL showed the highest accuracy, particularly in the optimal combination of months (OC). The study's machine learning predictions deviated by about 6-8% from USDA statistics, demonstrating the viability of these methods. It highlights the importance of accurate crop yield monitoring for agronomy, food security, and international trade, especially under climate change conditions.
8	Estimation of Organic Matter Content in Coastal Soil Using Reflectance Spectroscopy Research	2015	ZHENG Guanghui1, Dongryeol RYU2, JIAO Caixia1 and HONG Changqiao1	This study compares the accuracy of predicting soil organic matter (SOM) using "Deviation of arch" (DOA)-based regression and partial least squares regression (PLSR) on VNIR spectral data. A two-fold cross-validation scheme, repeated 10,000 times, was used for comparison. Soil samples from northern Jiangsu Province, China, showed both models estimated SOM well, with PLSR generally outperforming DOA. The study concludes that PLSR is effective for rapid, non-destructive SOM determination, while the DOA-based model also provided acceptable accuracy using fewer spectral bands.

9	Preliminary Study of Soil Available Nutrient Simulation Using a Modified WOFOST Model and Time-Series Remote Sensing Observations	2017	Zhiqiang Cheng 1,2 ID, Jihua Meng 1,*, Yanyou Qiao 1, Yiming Wang 1,2, Wenquan Dong 1 and Yanxin Han 1,2	This paper introduces a novel method for estimating soil available nutrients (SANs) using remote sensing and the integration of the WOFOST model with time series data. By replacing statistical models, this approach improves stability and accuracy. Experiments on spring maize plots show significant improvements in SAN estimation accuracy by considering nutrient stress on crop growth. Overall, this method offers more reliable SAN estimates compared to traditional techniques.

CHAPTER 3

LOW LEVEL DESIGN

3.1 Methodology

In our research, we used a comprehensive set of 10 different machine learning algorithms to build models based on agricultural data, including 2200 records containing 22 different crop markers. These models provide farmers with recommendations on the most suitable crops. Our crop analysis methodology follows standard data analysis steps. A significant improvement is the inclusion of multiple classifiers that are fine-tuned and evaluated to identify the most suitable input data. our method is important to highlight relevant features that improve crop detection and help farmers choose the most suitable features for accurate predictions.

Our models have the following steps to predict returns-

Data collection: Collecting data from IoT devices in the farm is necessary for machine learning analysis. By gathering important information about crop use, crop type, water requirements and cropping methods, we can significantly increase the productivity of smart farms.

Data modelling: The accuracy of the data analysis was tested through experiments that involved changing the labels. We analyzed a dataset of crops using machine learning using seven different features to classify them. Additionally, we determined the minimum number of features required for accurate learning and prediction.

Model evaluation and interpretation: We want to achieve accurate prey detection by choosing appropriate features for our machine learning algorithms. To ensure optimal results we took into account the characteristics of the parameters of the relevant properties. This enabled us to obtain accurate and reliable information about our agricultural activities. During the experiments, we analysed the effect of changing the labels on the accuracy of our data analysis algorithm. This helped us better understand the impact of small tag changes and optimize our approach for more accurate results. To succeed in classifying plants, it is important to use more comprehensive notations. It is very important to research and ensure the most effective classification techniques in the field.

3.2 SCOPE

The scope of this project is to investigate a dataset of crop records for agricultural sector using machine learning technique. To identifying crop predicting by farmer is more difficult. We try to reduce this risk factor behind selection of the crop.

3.3 PROJECT REQUIREMENTS

3.3.1 Functional requirements

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

3.3.2 Non-Functional Requirements

Process of functional steps,
Problem define
Preparing data
Evaluating algorithms
Improving results
Prediction the result

3.4 SOFTWARE DESCRIPTION

Anaconda is a free and open distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system "Conda". So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator.
Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- Jupyter Lab
- Jupyter Notebook
- Visual Studio Code

3.4.1Conda

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-languages. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository

3.4.2 The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

3.4.3 Notebook Document

Notebook documents (or "notebooks", all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

3.4.4 Jupyter Notebook App

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

3.4.5 Kernel

A notebook kernel is a "computational engine" that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

3.4.6 Notebook Dashboard

The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown). The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files.

3.5 Modules

- Data validation and Data cleaning
- Data Standardization
- Exploration data analysis of visualization and normalizing the data
- Training a model by given attributes and using Random Forest algorithm
- Performance measurements of KNN and Decision Tree
- Performance measurements of Logistic Regression

3.5.1 Module-01: Variable Identification Process / data validation process and Data cleaning:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the

training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers uses this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model. For example, time series data can be analysed by regression algorithms; classification algorithms can be used to analyse discrete data. (For example, to show the data type format of given dataset)

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Fig 3.1 Given Data Frame

3.5.1.1. Data cleaning:

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

3.5.2 Module-02: Exploration data analysis of visualization and normalization:

3.5.2.1 Exploration data analysis of visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end. Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data. How to chart time series data with line plots and categorical quantities with bar charts. How to summarize data distributions with histograms and box plots. How to summarize the relationship between variables with scatter plots.

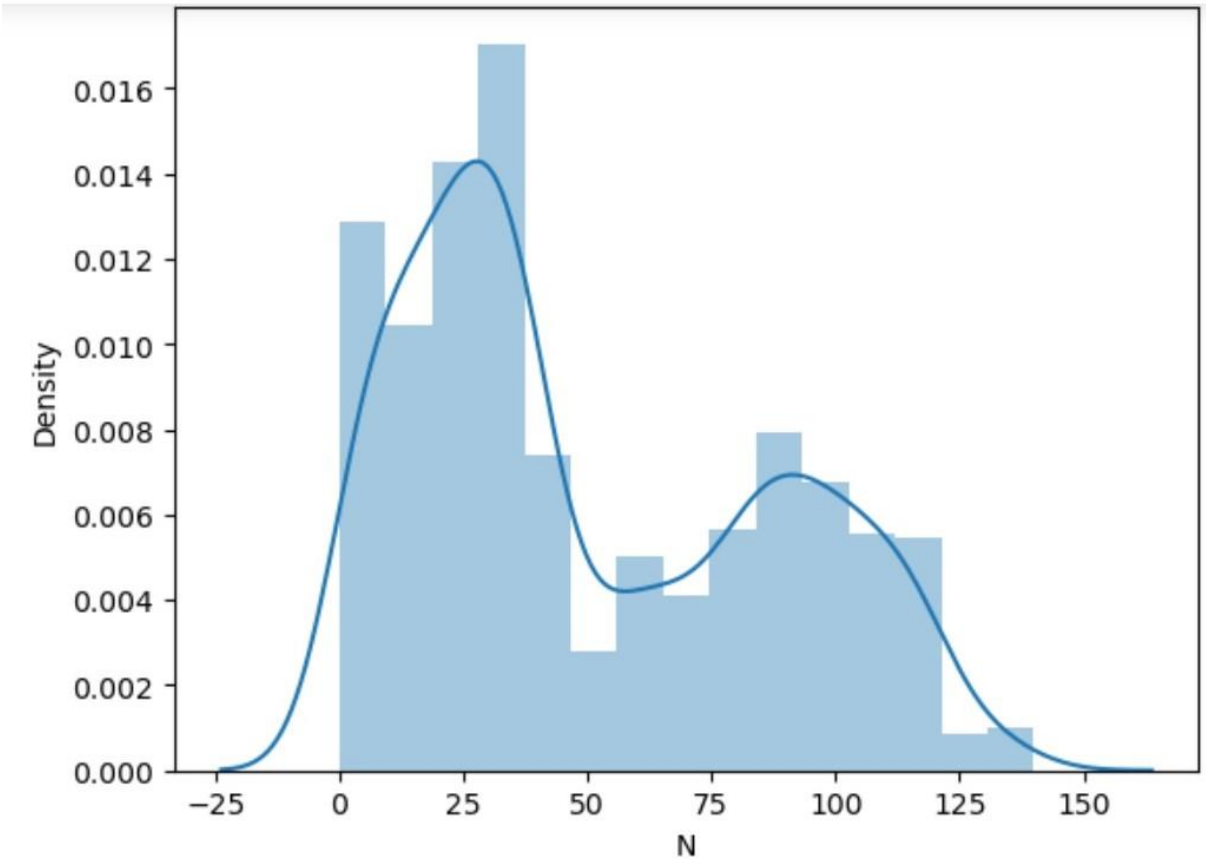

Fig 3.2: Distplot

Many machine learning algorithms are sensitive to the range and distribution of attribute values in the input data. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results. Even before predictive models are prepared on training data, outliers can result in misleading representations and in turn misleading interpretations of collected data. Outliers can skew the summary 22 distribution of attribute values in descriptive statistics like mean and standard deviation and in plots such as histograms and scatterplots, compressing the body of the data. Finally, outliers can represent examples of data instances that are relevant to the problem such as anomalies in the case of fraud detection and computer security.

It couldn’t fit the model on the training data and can’t say that the say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

3.5.2.2 Data Normalization

To train data with string datatype we use label encoder to convert them to numeric data. Label encoding simply assigns each data point a number starting from the attributes we are label encoding is district name, season and crop. Then to scale each data attribute we use two types of scaling techniques which are MinMax transform and normalization. The attributes district name, season, crop is scaled by MinMax transform and production per area is scaled by normalization. MinMax transform is done by subtracting each datapoint from minimum among datapoints then dividing by maximum- minimum. normalization is done by dividing datapoint with standard deviation. We choose normalization to production per area as many data points are low and min max transform is yielding good results.

```
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()

X_train = ms.fit_transform(X_train)
X_test = ms.transform(X_test)
```

Fig3.3: MinMax Scaler

3.5.2.3 Data Standardization

Data standardization converts data into a standard format that computers can read and understand. This is important because it allows different systems to share and efficiently use data. Without data standardization, it would not be effortless for different approaches to communicate and exchange information. Data standardization is also essential for preserving data quality. When data is standardized, it is much easier to detect errors and ensure that it is accurate. This is essential for making sure that decision-makers have access to accurate and reliable information. Overall, data standardization is critical to ensuring that data is usable and accessible. Without it, we would be unable to use and manage data effectively. Data standardization is essential because it allows different systems to exchange data consistently. Without standardization, it would be challenging for computers to communicate with each other and exchange data. Standardization also makes it easier to process and analyze data and store it in a database. With this approach, businesses can make better decisions based on their data. When data is standardized, companies can compare and analyze it more easily to make insights that they can use to improve their operations. Data standardization has many benefits, but one of the most important is that it helps businesses avoid making decisions based on inaccurate or incomplete data. Data standardization ensures that companies have a complete and accurate picture of their data, allowing them to make better decisions to improve their bottom line.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()


sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

Fig 3.4: Data Standardization

3.5.3 Module 3: Importing Required Modules and using Random Forest algorithm

At First, we import all the required modules. Then we use appropriate cleaning methods to clean our data. Then we split our data according to the required output. We split our dataset into train and test data using train_test_split technique. The X prefix indicates the element esteems and y prefix indicates target esteems. This splits the dataset into train and test information in the proportion which we mentioned. Here we are using a proportion of 80:20. At this point we epitomize any calculation. Here we fit our preparation information into this calculation so our system can get prepared utilizing this information. Here the training part is finished.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

Fig 3.5 Importing Modules

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig 3.6: Splitting the Data

3.5.3.1 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks. The following are the basic steps involved in performing the random forest algorithm pick N random records from the dataset. Build a decision tree based on these N records. Choose the number of trees you want in your algorithm and repeat steps 1 and 2. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

3.5.4 Module 4: Performance measurements of KNN and Decision Tree

3.5.4.1 KNN

K-Nearest Neighbor (KNN) K-Nearest Neighbor is a supervised machine learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors) and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors. In the distance-weighted nearest neighbor algorithm, it weights the contribution of each of the k neighbors according to their distance using the following query giving greater weight to the closest neighbors.

Usually KNN is robust to noisy data since it is averaging the k-nearest neighbors. The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labeled points and uses them to learn how to label other points. To label a new point, it looks at the labeled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the "k" is the number of neighbors it checks). Makes predictions about the validation set using the entire training set. KNN makes a prediction about a new instance by searching through the entire set to find the k "closest" instances. "Closeness" is determined using a proximity measurement (Euclidean) across all features.

3.5.4.2 Decision Tree

It is one of the most powerful and popular algorithms. Decision-tree algorithm falls under the category of supervised learning algorithms. It 24 works for both continuous as well as categorical output variables.

Assumptions of Decision tree:

At the beginning, we consider the whole training set as the root. Attributes are assumed to be categorical for information gain, attributes are assumed to be continuous. On the basis of attribute values records are distributed recursively. We use statistical methods for ordering attributes as root or internal node.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers.

3.5.5 Module 5: Performance measurements of Logistic Regression

Logistic regression is useful when the response variable is binary but the explanatory variables are continuous. This would be the case if one were predicting whether or not an customer is a good credit risk, using information on their income, years of employment, age, education, and other continuous variables.
In such applications one uses the model

(1) PY=1=exp$X$Tθ1+exp$X$Tθ,

where $Y = 1$ if the customer is a good risk, $\boldsymbol{X}$ is the vector of explanatory variables for that customer, and θ are the unknown parameters to be estimated from the data. This model is advantageous because, under the transformation
p=lnP[Y=1]1−P[Y=1]

one obtains the linear model $p = \boldsymbol{X}^{\mathrm{T}}\theta$. Thus all the usual machinery of multiple linear regression will apply.
Logistic regression can be modified to handle categorical explanatory variables through definition of dummy variables, but this becomes impractical if there are many categories. Similarly, one can extend the approach to cases in which the response variable is polytomous

(i.e., takes more than two categorical values). Also, logistic regression can incorporate product interactions by defining new explanatory variables from the original set, but this, too, becomes impractical if there are many potential interactions. Logistic regression is relatively fast to implement, which is attractive in data mining applications that have large datasets. Perhaps the chief value of logistic regression is that it provides an important theoretical window on the behavior of more complex classification methodologies

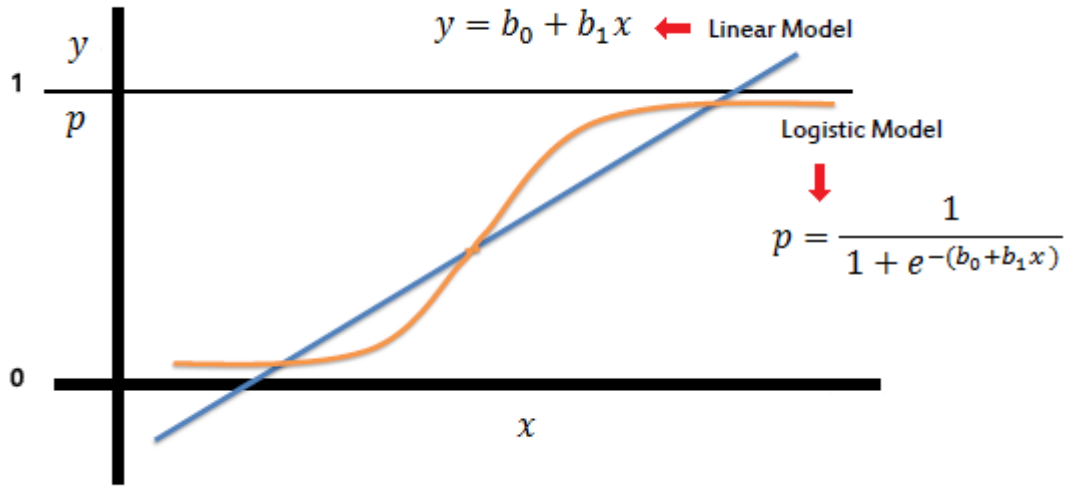

Fig 3.7: Logistic Regression Graph

3.5.7 Parameter calculations

Regression refers to predictive modelling problems that involve predicting a numeric value. It is different from classification that involves predicting a class label. Unlike classification, you cannot use classification accuracy to evaluate the predictions made by a regression model. Instead, you must use error metrics specifically designed for evaluating predictions made on regression problems.

There are four error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

- Mean Squared Error (MSE).
- Mean Absolute Error (MAE)
- R2 Score

3.5.7.1 Mean Squared Error

Mean Squared Error, or MSE for short, is a popular error metric for regression problems. It is also an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. Here "*least squares*" refers to minimizing the mean squared error between predictions and expected values.

The MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset.

- MSE = 1 / N * sum for i to N (y_i – yhat_i)^2

Where *y_i* is the i'th expected value in the dataset and *yhat_i* is the i'th predicted value. The difference between these two values is squared, which has the effect of removing the sign, resulting in a positive error value.

The squaring also has the effect of inflating or magnifying large errors. That is, the larger the difference between the predicted and expected values, the larger the resulting squared positive error. This has the effect of "*punishing*" models more for larger errors when MSE is used as a loss function. It also has the effect of
"*punishing*" models by inflating the average error score when used as a metric.

3.5.7.2 Root Mean Squared Error:

RMSE, is an extension of the mean squared error**.** Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

For example, if your target variable has the units "*dollars*," then the RMSE error score will also have the unit "*dollars*" and not "*squared dollars*" like the MSE.
As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.
The RMSE can be calculated as follows:

- **RMSE = sqrt(1 / N * sum for i to N (y_i – yhat_i)^2)**

Where *y_i* is the i'th expected value in the dataset, *yhat_i* is the i'th predicted value, and *sqrt()* is the square root function**.**

We can restate the RMSE in terms of the MSE as:

- RMSE = sqrt(MSE)

Note that the RMSE cannot be calculated as the average of the square root of the mean squared error values. This is a common error made by beginners and is an
example of jensens inequality.

3.5.7.3 Mean Absolute Error

MAE, is a popular metric because, like RMSE, the units of the error score match the units of the target value that is being predicted.

Unlike the RMSE, the changes in MAE are linear and therefore intuitive.
That is, MSE and RMSE punish larger errors more than smaller errors, inflating or magnifying the mean error score. This is due to the square of the error value. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

As its name suggests, the MAE score is calculated as the average of the absolute error values. Absolute or *abs()* is a mathematical function that simply makes a number positive. Therefore, the difference between an expected and predicted value may be positive or negative and is forced to be positive when calculating the MAE.
The MAE can be calculated as follows:

- MAE = 1 / N * sum for i to N abs (y_i – yhat_i)

Where *y_i* is the i'th expected value in the dataset, *yhat_i* is the i'th predicted value and *abs ()* is the absolute function.

3.5.7.4 R2_score:

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable. R-squared (R^2) is defined as a number that tells you how well the independent variable(s) in a statistical model explain the variation in the dependent variable. It goes from 0 to 1, where 1 indicates a perfect fit of the model to the data. Whereas correlation explains the strength of the relationship between an independent and a dependent variable, R-squared explains the extent to which the variance of one variable explains the variance of the second variable. So, if the R^2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

3.6 Flowchart

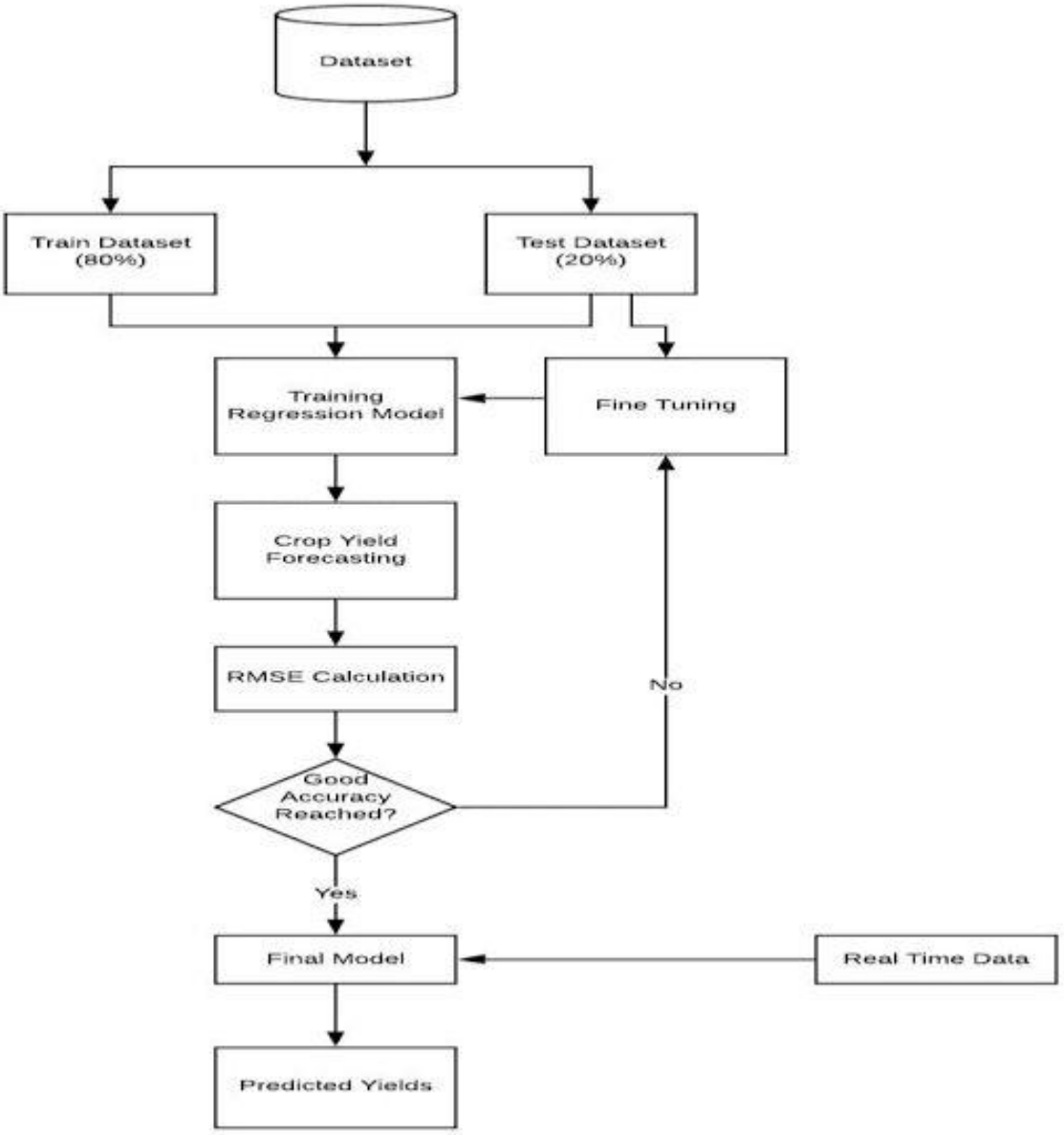

Fig 3.8: FlowChart

CHAPTER 4
HIGH LEVEL DESIGN

4.1 System Architecture

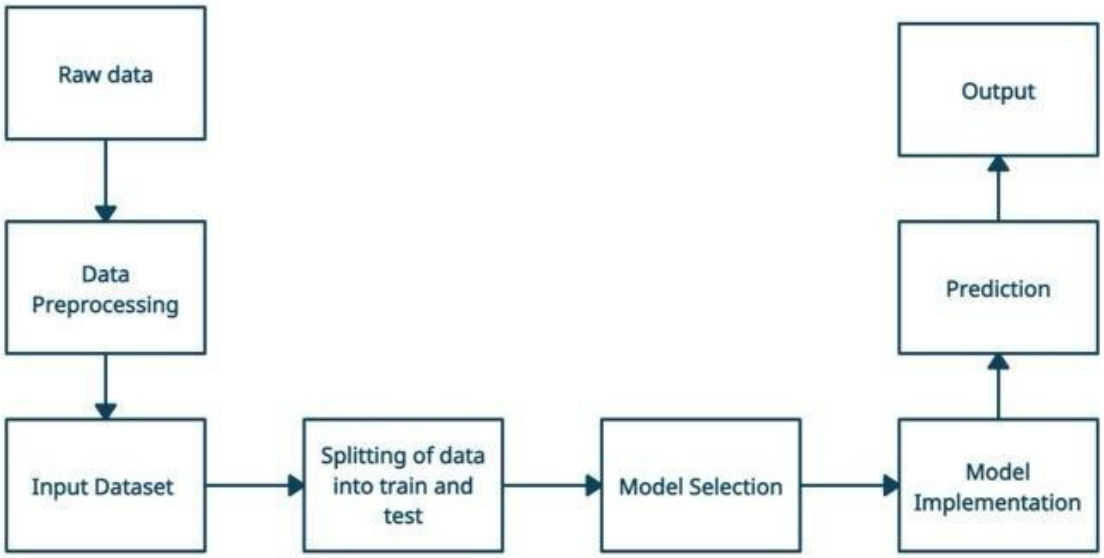

Fig 4.1: System Architecture

4.2 Raw Data

To predict crop yield using machine learning, it's crucial to understand the different types of raw data involved and how they contribute to the model's performance. Here's an explanation of the key data types and their importance:

4.2.1. Weather Data

Temperature: Affects plant growth, development, and maturation.

Precipitation: Determines water availability for crops.

Humidity: Influences transpiration rates and disease prevalence.

4.2.2. Soil Data

Soil Type: Determines water retention capacity and nutrient availability.

pH Level: Affects nutrient solubility and root growth.

Nutrient Content: Essential for plant growth (e.g., nitrogen, phosphorus, potassium).

Moisture Levels: Directly impacts water availability to plants.

4.2.3. Crop Data

Crop Type: Different crops have varying requirements and yield potentials.

Varieties: Specific varieties may have unique growth characteristics and yield potentials.

Planting Dates: Affects growing season length and exposure to weather conditions.

4.3. Data Preprocessing

Data preprocessing 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 preprocessing 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 preprocessing 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.

4.4 Input Dataset

To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the **dataset**.

Dataset may be of different formats for different purposes, such as, if we want to create a machine learning model for business purpose, then dataset will be different with the dataset required for a liver patient. So each dataset is different from another dataset. To use the dataset in our code, we usually put it into a CSV **file.**

CSV stands for "**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs.

```
crop = pd.read_csv("cropnew.csv")
crop.head()
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Fig 4.2: Input Dataset

4.5. Splitting of Data into Train and Test

Whenever we build machine learning models, we will be training the model on the dataset (X and y). Once trained, we want to ensure the trained model is capable of performing well on the unseen test data as well. The train test split is a way of checking if the ML model performs well on data it has not seen. This is applied to supervised learning problems, both **classification and regression.** We split our dataset into train and test data using the train_test_split technique. The x-prefix represents element values and the y-prefix represents object values. This divides the data into train and test data with the ratio we mentioned. Here we use a ratio of 80:20. Here we introduce any calculations. Here we combine our preparation information with this calculation so that our system can prepare using this information. Here the training part is ready.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig 4.3: Splitting the dataset

4.6 Model Selection

Model selection is an essential phase in the development of powerful and precise predictive models in the field of machine learning. Model selection is the process of deciding which algorithm and model architecture is best suited for a particular task or dataset. It entails contrasting various models, assessing their efficacy, and choosing the one that most effectively addresses the issue at hand.

The choice of an appropriate machine learning model is crucial since there are various levels of complexity, underlying assumptions, and capabilities among them. A model's ability to generalize to new, untested data may not be as strong as its ability to perform effectively on a single dataset or problem. Finding a perfect balance between the complexity of models & generalization is therefore key to model selection.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

models = {

    'Logistic Regression': LogisticRegression(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
}
```

Fig 4.4: Models

4.7 Model Implementation

When implementing a model start most of the work in ML is on the data side, so getting a full pipeline running for a complex model is harder than iterating on the model itself. After setting up your data pipeline and implementing a simple model that uses a few features, we can iterate on creating a better model. Simple models provide a good baseline, even if we don't end up launching them. In fact, using a simple model is probably better than you think. Starting simple helps us determine whether or not a complex model is even justified.

```
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
ypred = rfc.predict(X_test)
```

Fig 4.5: Model Selection

4.8 Prediction

The purpose of prediction in machine learning is to project a probable data set that relates back to the original data. This helps organizations predict future behaviours and market changes. Essentially, prediction is used to fit a shape as closely to the data as possible. With machine learning predictions, organizations use proactive decisions to avoid predicted user churn. To gain the most success with prediction in machine learning, organizations need to have infrastructure in place to support the solutions, and high quality data to supply the algorithm.

```
def recommendation(N,P,k,temperature,humidity,ph,rainfal):
    features = np.array([[N,P,k,temperature,humidity,ph,rainfal]])
    transformed_features = ms.fit_transform(features)
    transformed_features = sc.fit_transform(transformed_features)
    prediction = rfc.predict(transformed_features).reshape(1,-1)

    return prediction[0]
```

Fig: 4.6 Prediction

4.9 Output

Model output is the prediction or decision made by a machine learning model based on input data. In supervised learning, the model output is a predicted target value for a given input. In unsupervised learning, the model output may include cluster assignments or other learned patterns in the data. Model output can be a single value, a probability distribution, a class label, or a series of continuous or discrete values, and is often used to evaluate the performance of a machine learning model.

```
N = 40
P = 50
k = 50
temperature = 40.0
humidity = 20
ph = 100
rainfall = 100

predict = recommendation(N,P,k,temperature,humidity,ph,rainfall)


crop_dict = {1: "Rice", 2: "Maize", 3: "Jute", 4: "Cotton", 5: "Coconut", 6: "Papaya", 7: "Orange",
                8: "Apple", 9: "Muskmelon", 10: "Watermelon", 11: "Grapes", 12: "Mango", 13: "Banana",
                14: "Pomegranate", 15: "Lentil", 16: "Blackgram", 17: "Mungbean", 18: "Mothbeans",
                19: "Pigeonpeas", 20: "Kidneybeans", 21: "Chickpea", 22: "Coffee"}

if predict[0] in crop_dict:
    crop = crop_dict[predict[0]]
    print("{} is a best crop to be cultivated ".format(crop))
else:
    print("Sorry are not able to recommend a proper crop for this environment")
```

```
if predict[0] in crop_dict:
    crop = crop_dict[predict[0]]
    print("{} is a best crop to be cultivated ".format(crop))
else:
    print("Sorry are not able to recommend a proper crop for this environment")
```

```
Papaya is a best crop to be cultivated
```

Fig 4.7 Output

CHAPTER 5

IMPLEMENTATION

5.1 Proposed Algorithm

5.1.1 Random Forest Classifier Algorithm

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.
The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below diagram:

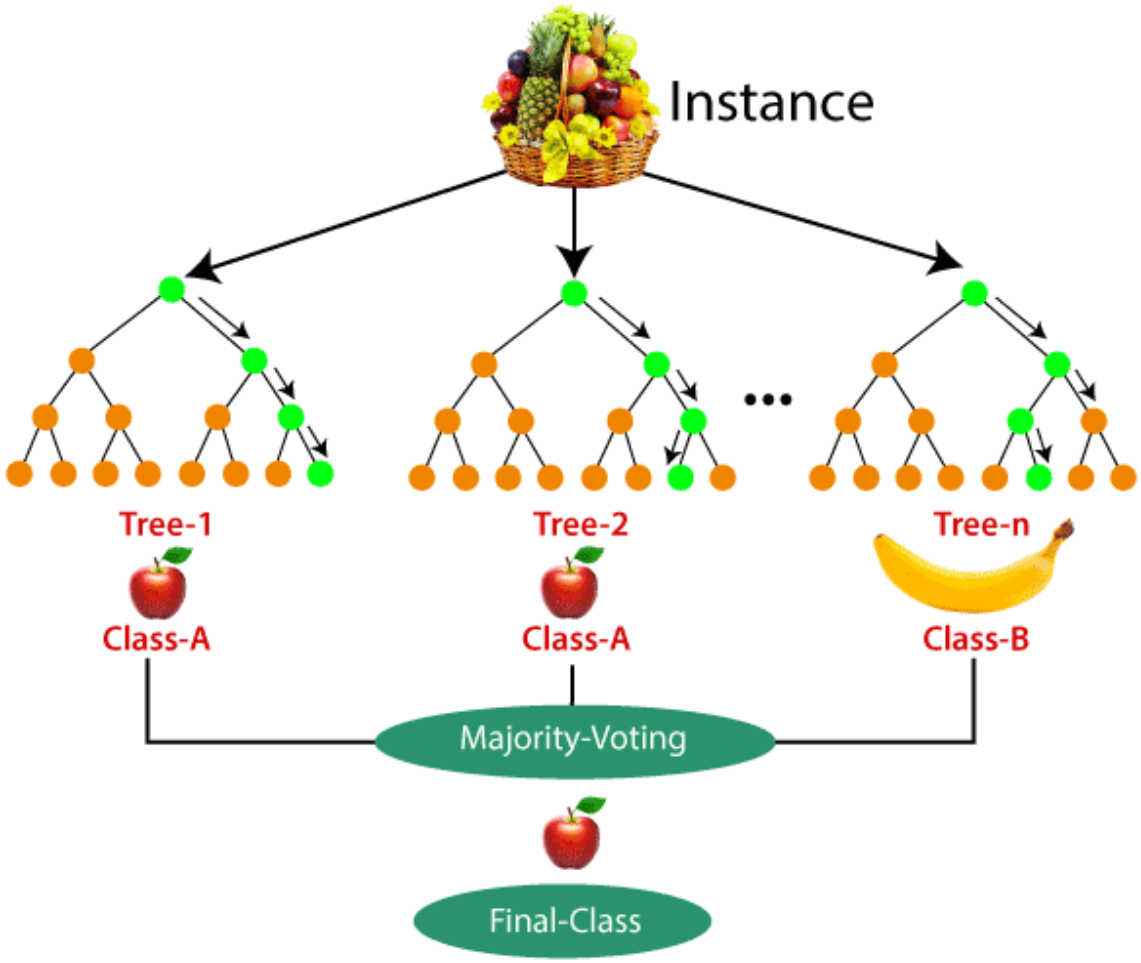

Fig 5.1: Working of Random Forest Classifier

5.1.2 Logistic Regression Classifier

The logistic regression algorithm can be summarized in the following steps:

Step-by-Step Algorithm

1. Initialize Parameters

•Initialize the weights \mathbf{w} and bias b to small random values or zeros.

2. Hypothesis Function

•The hypothesis (or model) for logistic regression is the logistic (sigmoid) function:

h_\theta(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}}

3. Cost Function

•Define the cost function using binary cross-entropy:

$$J(\mathbf{w}, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(\mathbf{x}^{(i)})) \right]$$

4. Gradient Descent

•Compute the gradients of the cost function with respect to the weights and bias:

$$\frac{\partial J(\mathbf{w}, b)}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\frac{\partial J(\mathbf{w}, b)}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)})$$

•Update the weights and bias using the gradients:

$$w_j := w_j - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial w_j}$$

$$b := b - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial b}$$

where α is the learning rate.

5. Iterate Until Convergence

•Repeat the gradient descent steps until the cost function converges, i.e., the change in cost function is below a predefined threshold or a maximum number of iterations is reached.

6. Making Predictions

•After training, use the logistic function to make predictions on new data:

$$\hat{y} = \begin{cases}$$

$$1 & \text{if } h_\theta(\mathbf{x}) \geq 0.5 \\$$

$$0 & \text{if } h_\theta(\mathbf{x}) < 0.5$$

$$\end{cases}$$

5.1.3 Decision Tree Algorithm

Decision trees are a versatile machine learning algorithm used for both classification and regression tasks. They work by recursively splitting the data into subsets based on the feature that provides the maximum information gain or minimizes a specific metric. Here's a detailed outline of the decision tree algorithm:

Step-by-Step Algorithm

1. Start with the Entire Dataset

•Begin with the root node containing the entire dataset.

2. Select the Best Feature to Split

•For each feature, calculate the potential split's effectiveness using a criterion like Gini impurity, entropy (information gain), or variance reduction (for regression).

•Gini Impurity (for classification):

Gini(D) = 1 - \sum_{i=1} ^{c} p_i^2

where p_i is the probability of class i in dataset D.

•Entropy (for classification):

Entropy(D) = - \sum_{i=1} ^{c} p_i \log_2(p_i)

•Information Gain (for classification):

Information\ Gain (D, A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy(D_v)

where D_v is the subset of D where feature A has value v.

•Variance Reduction (for regression):

Variance\ Reduction = Variance(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Variance(D_v)

3. Split the Dataset

•Choose the feature that provides the highest information gain or lowest Gini impurity (for classification) or variance (for regression).

•Split the dataset into subsets based on the chosen feature's values.

4. Create Child Nodes

•Assign the subsets to child nodes.

•If a subset is pure (all instances belong to the same class) or meets a stopping criterion (e.g., maximum tree depth, minimum number of samples per node), mark it as a leaf node.

5. Repeat Recursively

•Repeat steps 2 to 4 for each child node using only the subset of data associated with that node.

6. Stopping Criteria

•The recursion stops when one of the following conditions is met:

•All instances in a node belong to the same class.

•The maximum tree depth is reached.

•The number of instances in a node is less than a specified minimum.

•The information gain is below a threshold.

7. Make Predictions

•For classification: Traverse the tree from the root to a leaf node based on the feature values of the instance. The class label of the leaf node is the prediction.

•For regression: The prediction is the mean value of the target variable in the leaf node.

5.1.4 K-Nearest Neighbor Algorithm

The k-Nearest Neighbors (k-NN) algorithm is a simple, non-parametric, and instance-based learning algorithm used for both classification and regression tasks. Here's a detailed outline of the k-NN algorithm:

Step-by-Step Algorithm

1. Data Preparation

•Feature Scaling: Normalize or standardize the feature values to ensure that each feature contributes equally to the distance computation.

•Train-Test Split: Split the dataset into training and testing sets to evaluate the performance of the model.

2. Choose the Number of Neighbors (k)

•k Selection: Select the number of neighbors, (k). This is a hyperparameter that can be tuned using techniques like cross-validation.

3. Distance Metric

•Distance Calculation: Define a distance metric to compute the distance between instances. The most common distance metric is the Euclidean distance:

d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}

where (x) and (y) are instances and (n) is the number of features.

4. Classification or Regression

•For Classification: The predicted class is the mode (most common class) among the k-nearest neighbors.

•For Regression: The predicted value is the mean (average) of the values of the k-nearest neighbors.

•Compute Distances: For a new instance, compute the distance to all instances in the training set.

•Identify Neighbors: Identify the k instances in the training set that are closest to the new instance.

•Aggregate Output: Aggregate the outputs (class labels for classification or numerical values for regression) of the k-nearest neighbors to make a prediction.

5.2 Pseudocode

5.2.1 Pseudocode of KNN

```
import numpy as np
from collections import Counter
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))
def knn_predict(X_train, y_train, X_test, k, task='classification'):
            predictions = []
            for test_instance in X_test:
        #Compute distances between the test_instance and all training instances
            distances = [euclidean_distance(test_instance, x_train) for x_train in X_train]
            # Get the k nearest neighbors (indices)
            k_indices = np.argsort(distances)[:k]
        # Get the labels of the k nearest neighbors
        k_nearest_labels = [y_train[i] for i in k_indices]
        if task == 'classification':
            # Majority vote (classification)
            most_common = Counter(k_nearest_labels).most_common(1)
            predictions.append(most_common[0][0])
        elif task == 'regression':
            # Mean value (regression)
            predictions.append(np.mean(k_nearest_labels))
    return np.array(predictions)
```

```
# Example usage:
# X_train and X_test should be scaled/normalized.
# X_train = np.array([...])
# y_train = np.array([...])
# X_test = np.array([...])
# k = 3
# task = 'classification' or 'regression'
# predictions = knn_predict(X_train, y_train, X_test, k, task)
```

5.2.2 Pseudocode of Decision Tree Algorithm

```
class Node:
    def _init_(self, feature=None, threshold=None, left=None, right=None, *, value=None):
        self.feature = feature
        self.threshold = threshold
        self.left = left
        self.right = right
        self.value = value
def gini_impurity(y):
    m = len(y)
    return 1.0 - sum((np.sum(y == c) / m) ** 2 for c in np.unique(y))
def entropy(y):
    m = len(y)
    p = [np.sum(y == c) / m for c in np.unique(y)]
    return -sum(p[i] * np.log2(p[i]) for i in range(len(p)))
def information_gain(y, y_left, y_right, criterion='gini'):
```

```
    weight_left = len(y_left) / len(y)
    weight_right = len(y_right) / len(y)
    if criterion == 'gini':
        gain = gini_impurity(y) - (weight_left * gini_impurity(y_left) + weight_right * gini_impurity(y_right))
    elif criterion == 'entropy':
        gain = entropy(y) - (weight_left * entropy(y_left) + weight_right * entropy(y_right))
    return gain


def split(dataset, feature, threshold):
    left = np.where(dataset[:, feature] <= threshold)
    right = np.where(dataset[:, feature] > threshold)
    return left, right


def best_split(X, y, criterion='gini'):
    best_feature, best_threshold, best_gain = None, None, -float('inf')
    for feature in range(X.shape[1]):
        thresholds = np.unique(X[:, feature])
        for threshold in thresholds:
            left_idx, right_idx = split(X, feature, threshold)
            if len(left_idx[0]) == 0 or len(right_idx[0]) == 0:
                continue
            gain = information_gain(y, y[left_idx], y[right_idx], criterion)
            if gain > best_gain:
                best_feature, best_threshold, best_gain = feature, threshold, gain
    return best_feature, best_threshold
```

```
def build_tree(X, y, depth=0, max_depth=None, criterion='gini'):
    num_samples_per_class = [np.sum(y == i) for i in np.unique(y)]
    predicted_class = np.argmax(num_samples_per_class)
    node = Node(value=predicted_class)

    if depth < max_depth:
        feature, threshold = best_split(X, y, criterion)
        if feature is not None:
            left_idx, right_idx = split(X, feature, threshold)
            node.feature = feature
            node.threshold = threshold
            node.left = build_tree●
```

5.2.3Pseudocode of Logistic Regression

```
# Initialize parameters
weights = initialize_weights(dimension)
bias = 0
# Set learning rate and number of iterations
alpha = 0.01
num_iterations = 1000
for i in range(num_iterations):
    # Compute the linear combination of inputs and weights
    z = dot_product(weights, X) + bias
    # Apply the sigmoid function
    h = 1 / (1 + exp(-z))
```

```
    # Compute the gradients
    dw = (1 / m) * dot_product(X.T, (h - y))
    db = (1 / m) * sum(h - y)
    # Update the parameters
    weights -= alpha * dw
    bias -= alpha * db
# Making predictions
predictions = 1 / (1 + exp(-dot_product(weights, X_test) + bias))
predicted_labels = predictions >= 0.5
```

Explanation of Pseudocode

1. **Initialization:**

   • We start by initializing the weights and bias. The weights can be initialized to small random values or zeros.

2. **Set Hyperparameters:**

   • The learning rate \alpha controls how much the model parameters are adjusted with respect to the gradient. The number of iterations determines how many times the update process is repeated.

3. **Gradient Descent Loop:**

   • In each iteration, we compute the linear combination of input features and weights plus the bias.

   • The sigmoid function is applied to get the predicted probabilities.

   • The gradients for weights and bias are computed using the difference between predicted probabilities and actual labels.

   • The weights and bias are updated by subtracting the product of the learning rate and the gradients.

4. **Making Predictions:**

   • After training, we can use the learned weights and bias to predict the probability of the positive class for new data points.

   • Predictions are converted to binary labels based on a threshold of 0.5.

This pseudocode represents a basic implementation of the logistic regression algorithm suitable for binary classification problems. For more complex scenarios or larger datasets, further optimizations and considerations may be necessary.

5.2.4 Pseudocode for Random Forest Algorithm

1. Input:

- Training data: X_train, y_train

- Number of trees: N_trees

- Number of features to consider for best split: M_features

- Maximum depth of trees: max_depth (optional)

- Minimum samples required to split a node: min_samples_split (optional)

- Minimum samples required at a leaf node: min_samples_leaf (optional)

2. Initialize an empty list of trees: forest = []

3. For i = 1 to N_trees:

a. Draw a bootstrap sample from X_train, y_train:

```
X_sample, y_sample = bootstrap_sample(X_train, y_train)
```

b. Build a decision tree on the bootstrap sample:

```
tree = DecisionTree()

tree.train(X_sample, y_sample, M_features, max_depth, min_samples_split,
min_samples_leaf)
```

c. Add the trained tree to the forest:

```
forest.append(tree)
```

4. Define the prediction function for the Random Forest:

```
function predict(X):

 # Initialize an empty list of predictions
```

```
predictions = []

# For each tree in the forest, get the prediction for each sample in X

for tree in forest:

    predictions.append(tree.predict(X))

    # For each sample, use majority voting or average (for classification or regression,
    respectively)

final_predictions = majority_voting(predictions)  # or average(predictions) for regression

return final_predictions
```

5.3 Screenshot of the Output

5.3.1 Predictive Model

```
def recommendation(N,P,k,temperature,humidity,ph,rainfal):
    features = np.array([[N,P,k,temperature,humidity,ph,rainfal]])
    transformed_features = ms.fit_transform(features)
    transformed_features = sc.fit_transform(transformed_features)
    prediction = rfc.predict(transformed_features).reshape(1,-1)

    return prediction[0]
```

Fig 5.2 Predictive Model

5.3.2 Predictive System

```
N = 40
P = 50
k = 50
temperature = 40.0
humidity = 20
ph = 100
rainfall = 100

predict = recommendation(N,P,k,temperature,humidity,ph,rainfall)


crop_dict = {1: "Rice", 2: "Maize", 3: "Jute", 4: "Cotton", 5: "Coconut", 6: "Papaya", 7: "Orange",
                8: "Apple", 9: "Muskmelon", 10: "Watermelon", 11: "Grapes", 12: "Mango", 13: "Banana",
                14: "Pomegranate", 15: "Lentil", 16: "Blackgram", 17: "Mungbean", 18: "Mothbeans",
                19: "Pigeonpeas", 20: "Kidneybeans", 21: "Chickpea", 22: "Coffee"}

if predict[0] in crop_dict:
    crop = crop_dict[predict[0]]
    print("{} is a best crop to be cultivated ".format(crop))
else:
    print("Sorry are not able to recommend a proper crop for this environment")
```

```
if predict[0] in crop_dict:
    crop = crop_dict[predict[0]]
    print("{} is a best crop to be cultivated ".format(crop))
else:
    print("Sorry are not able to recommend a proper crop for this environment")
```

```
Papaya is a best crop to be cultivated
```

Fig 5.3 Predictive System

Chapter 6

Testing

Machine learning testing is the process of evaluating and validating the performance of machine learning models to ensure their correctness, accuracy, and robustness. Unlike traditional software testing, which mainly focuses on code functionality, ML testing includes additional layers due to the inherent complexity of ML models. It ensures that ML models perform as intended, providing reliable results and adhering to industry standards.

6.1 Accuracy Scores

The accuracy score of the different models of the predictive system of Random Forest, Decision Tree, Logistic Regression and KNN algorithms are as follows:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

models = {

    'Logistic Regression': LogisticRegression(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
}


for name, md in models.items():
    md.fit(X_train,y_train)
    ypred = md.predict(X_test)

    print(f"{name}  with accuracy : {accuracy_score(y_test,ypred)}")
```

```
Logistic Regression  with accuracy : 0.9636363636363636
K-Nearest Neighbors  with accuracy : 0.9590909090909091
Decision Tree  with accuracy : 0.9840909090909091
Random Forest  with accuracy : 0.9931818181818182
```

Fig 6.1: Accuracy of the Algorithm

6.2 Mean Absolute Error and r2_score

The Mean Absolute Error and r2 score of the different models of the predictive system of Random Forest, Decision Tree, Logistic Regression and KNN algorithms are as follows:

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error,r2_score, mean_squared_error


models = {
   'KNearest Neighbour Classifier': KNeighborsClassifier(),
    'Decision Tree Regressor':DecisionTreeRegressor(),
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier()
}
for name, md in models.items():
    md.fit(X_train,y_train)
    ypred = md.predict(X_test)

    print(f"{name} : Mean Absolute Error : {mean_absolute_error(y_test,ypred)} \nr2_Score : {r2_score
```

```
KNearest Neighbour Classifier : Mean Absolute Error : 0.1159090909090909
r2_Score : 0.9891031943756515
Decision Tree Regressor : Mean Absolute Error : 0.12045454545454545
r2_Score : 0.9734898609437491
Logistic Regression : Mean Absolute Error : 0.11818181818181818
r2_Score : 0.9756041665126526
Random Forest : Mean Absolute Error : 0.015909090909090907
r2_Score : 0.9990783796238113
```

Fig 6.2: Mean Absolute Error

6.3 Mean Squared Error

The Mean Squared Error and r2 score of the different models of the predictive system of Random Forest, Decision Tree, Logistic Regression and KNN algorithms are as follows:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error,r2_score, mean_squared_error


models = {
   'KNearest Neighbour Classifier': KNeighborsClassifier(),
    'Decision Tree Regressor':DecisionTreeRegressor(),
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier()
}
for name, md in models.items():
    md.fit(X_train,y_train)
    ypred = md.predict(X_test)

    print(f"{name} : Mean Squared Error : {mean_squared_error(y_test,ypred)} \nr2_Score : {r2_score(y_t
```

```
KNearest Neighbour Classifier : Mean Squared Error : 0.45681818181818185
r2_Score : 0.9891031943756515
Decision Tree Regressor : Mean Squared Error : 0.22954545454545455
r2_Score : 0.9945244907061731
Logistic Regression : Mean Squared Error : 1.0227272727272727
r2_Score : 0.9756041665126526
Random Forest : Mean Squared Error : 0.038636363636363635
r2_Score : 0.9990783796238113
```

Fig 6.3 Mean Square Error

Chapter 7
Conclusion and Future Scope

The Machine learning models, which we have properly trained and tuned, has significantly improved the accuracy of crop yield predictions compared to traditional statistical methods. This enables farmers to make more informed decisions regarding crop management and resource allocation. Machine learning provides valuable insights into the factors that most influence crop yields, helping farmers optimize their practices. The analysis process started with data cleaning and processing with removal of missing values, exploratory analysis and finally model building and evaluation. Finally, we predict the performance with a machine learning algorithm with different results. This provides some of the following insights into crop forecasting. Since this system covers most crops, the farmer can learn about crops that may never have been grown and lists all possible crops, helping the farmer decide which crop o grow. In addition, this system takes into account the past production data that helps the farmer to understand the demand and plants in various markets. Machine learning offers a robust framework for predicting crop yields, providing significant benefits in terms of accuracy, efficiency, and sustainability. By overcoming data-related challenges and enhancing model interpretability and adaptability, ML can play a pivotal role in the future of agriculture. The continued evolution of these techniques holds great promise for ensuring food security, promoting sustainable farming practices, and enhancing the economic stability of the agricultural sector. As technology advances, the integration of machine learning in agricultural practices will become increasingly critical, driving innovation and resilience in the face of global challenges.

Future Scope

- Using IoT based devices for the sensors to detect real time analysis of the crop field for better throughput.
- Dataset used in the resource can be improved by taking real time data through IOT devices
- Mobile App can be developed for mobile devices with added services like price estimates in accordance with current market prices
- Paid datasets may bring more reliable and accurate data which in turn might help in model accuracy. They may contain more features which may help correlate more with label.

Social Relevance

The social relevance of crop yield prediction using machine learning is profound, impacting various factors of society. Accurate crop yield predictions help ensure a stable food supply by allowing farmers to optimize their practices and governments to plan better for food distribution. This can mitigate the risks of food shortages and contribute to global food security. It also helps in Economic Stability in the society by Predictive models which enable farmers to maximize their yields and reduce losses, leading to increased income and financial stability. This can positively affect rural economies and reduce poverty in agricultural communities. It also have massive impact in Sustainable Agriculture. Machine learning can promote sustainable farming practices by predicting the optimal use of resources such as water, fertilizers, and pesticides. This reduces environmental impact, conserves natural resources, and ensures long-term agricultural productivity. Also, Adaption in Climate change. With climate change affecting agricultural patterns, machine learning models can help farmers adapt by predicting the impacts of weather changes and suggesting appropriate countermeasures. This resilience is crucial for maintaining crop yields in the face of unpredictable climate conditions. Governments and organizations can use crop yield predictions to formulate policies related to agriculture, trade, and food security. Accurate data supports evidence-based decision-making, leading to more effective and The adoption of advanced technologies in agriculture encourages innovation and modernizes traditional farming methods. It also highlights the importance of education in data science and agricultural technology, fostering a new generation of tech-savvy farmers. Early warnings about potential crop failures due to pests, diseases, or adverse weather conditions can help in preparing for and mitigating the impacts of agricultural disasters. This proactive approach can save livelihoods and prevent widespread economic and social disruption. In summary, crop yield prediction using machine learning not only enhances agricultural productivity but also contributes to broader social goals such as food security, economic stability, environmental sustainability, and disaster resilience.

REFERENCES

[1] Bendre, M. R., Thool, R.C., Thool, V. R., "Big Data in Precision Agriculture : Weather Forecasting for Future Agriculture", 1 st International Conference on Next Generation Computing Technologies, pp.744-750, 2015
.

[2] Grajales, D.F.P., Mosquera, G.J.A, Mejia, F., Piedrahita, L.C., Basurto, C., "Crop-Planning, Making Smarter Agriculture With Climate Data",Fourth International Conference on Agro-GeoInformatics, pp.240-244, 2015.

[3] Hemageetha, N., "A survey on application of data mining techniques to analyze the soil for agricultural purpose", 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp.3112-3117, 2016.

[4] Mayank Champaneri, Chaitanya Chandvidkar , Darpan Chachpara, Mansing Rathod, "Crop yield prediction using machine learning" International Journal of Science and Research ,April 2020.

[5] Pavan Patil, Virendra Panpatil, Prof. Shrikant Kokate, "Crop Prediction System using Machine Learning Algorithms", International Research Journal of Engineering and Technology, Feb 2020.

[6] Raval Agrawal, H., Agrawal, P., "Review on Data Mining Tools", International Journal of Innovative Science, Engineering & Technology, Vol. 1, Issue 2, pp.52-56, 2014.

[7] Sabri Arik, Tingwen Huang, Weng Kin Lai, Qingshan Liu , "Soil Property Prediction: An Extreme Learning Machine Approach" Springer, vol. 3, Issue 4,666-680,2015.

[8] Shivnath Ghosh,Santanu Koley, "Machine Learning for Soil Fertility and Plant Nutrient Management using Back Propagation Neural Networks" IJRITCC, vol. 2, Issue 2,292-297,2014.

[9] Tan, L., "Cloud-based Decision Support and Automation for Precision Agriculture in Orchards", ScienceDirect, pp. 330-335, IFAC–Papers OnLine 49-16, pp.330-335, 2016
.

[10] Zhihao Hong,Z. Kalbarczyk,R. K. Iyer, "A Data-Driven Approach to Soil Moisture Collection and Prediction" IEEE Xplore,vol. 2, Issue 2,292-297,2016.