CROP PREDICTION FROM MULTIMODAL DATA USING DEEP LEARNING CLASSIFIERS

A PROJECT REPORT

Submitted by

MANIMARAN R	**421620104059**
PUGAZHARASAN D	**421620104086**
PUSHPARAJ S	**421620104087**
RAJESH N	**421620104088**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

MAILAM ENGINEERING COLLEGE,

MAILAM

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report of the title "**CROP PREDICTION FROM MULTIMODAL DATA USING DEEP LEARNING CLASSIFERS"** is the bonafide work of "**MANIMARAN R (421620104059), PUGAZHARASAN D (421620104086), PUSHPARAJ S (421620104087), RAJESH N (421620104088)"** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Ms. G. SARASWATHI, M.E.,	**Dr. T. PRIYARADHIKADEVI, M.Tech., Ph.D.,**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Assistant Professor,	Professor and Head,
Computer Science and Engineering,	Computer Science and Engineering,
Mailam Engineering College,	Mailam Engineering College,
Mailam-604 304.	Mailam-604 304.

Submitted for the project and Viva-Voce Examination held on ______________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

With the blessings of almighty, I express my immense gratitude to my beloved parents.

Owing deeply to supreme, I express my sincere thanks to our honourable Founder, Chairman **Shri. M. DHANASEKARAN**, Secretary of our college **Dr. K. NARAYANASWAMY M.B.B.S**., and Treasurer **Shri.D.RAJARAJAN** who entered his generosity to give the facilities to undergoes the project work till its completion.

I am very much revered to our Director **Dr. S. SENTHIL, M.E, Ph.D**., for his support and encouragement shown towards our academic’s profile.

We tender our heartfelt gratitude to our beloved Principal of Our college **Dr. R. RAJAPPAN, M.E., Ph.D.,** for his valuable and encouragement throughout this endeavour.

I extend my sincere thanks and gratitude to our Head of The Department. **Dr. T. PRIYARADHIKADEVI, M.Tech, Ph.D.,** for sharing her thoughts in doing the project and the encouragement shown towards the project work.

I am very much obliged to my project supervisor **Ms. G. SARASWATHI, M.E.,** for her motivation, guidelines and continuous support for conducting the reviews at each and every level of work.

I also extend my sincere thanks to all the staff members and non-teaching staff of our department.

ABSTRACT

Crop prediction is a task that involves using deep learning algorithms to predict crop yields and other relevant metrics based on a variety of factors, such as weather conditions, soil data, and historical crop data. The goal of this task is to provide farmers and other stakeholders with accurate and reliable information about expected crop yields, which can help them to make better decisions about planting, harvesting, and other aspects of agricultural management. The problem of crop prediction involves several challenges, including the need for accurate and timely data, the selection of relevant features and parameters for analysis, and the development of suitable machine learning models for prediction. Moreover, the prediction accuracy may also be affected by factors such as regional variations in climate and soil conditions, as well as the presence of pests and other environmental factors. To address these challenges, researchers and developers in the field of crop prediction have developed a variety of techniques, including data pre-processing, feature selection, deep learning model selection, and performance evaluation. These techniques may involve the use of different types of data, such as weather data, soil data, and crop data, as well as various deep learning algorithms, such as multi-layer perceptron algorithm and Convolutional neural network algorithm. Ultimately, the success of crop prediction depends on the ability of the system to accurately and reliably analyse data from a variety of sources, and then predict crop yields and other relevant metrics with a high degree of accuracy. By addressing these challenges, crop prediction has the potential to improve agricultural productivity and sustainability, and to support the development of more efficient and effective farming practices.

ஆய்வுசுருக்கம்

பயிர் முன்கணிப்பு என்பது வானிலை, மண் தரவு மற்றும் வரலாற்று பயிர்த் தரவு போன்ற பல்வேறு காரணிகளின் அடிப்படையில் பயிர் விளைச்சல் மற்றும் பிற தொடர்புடைய அளவீடுகளைக் கணிக்க ஆழமான கற்றல் வழிமுறைகளைப் பயன்படுத்துவதை உள்ளடக்கிய பணியாகும். இந்த பணியின் குறிக்கோள், விவசாயிகள் மற்றும் பிற பங்குதாரர்களுக்கு எதிர்பார்க்கப்படும் பயிர் விளைச்சலைப் பற்றிய துல்லியமான மற்றும் நம்பகமான தகவல்களை வழங்குவதாகும், இது அவர்கள் நடவு, அறுவடை மற்றும் விவசாய நிர்வாகத்தின் பிற அம்சங்களைப் பற்றி சிறந்த முடிவுகளை எடுக்க உதவும். துல்லியமான மற்றும் சரியான நேரத்தில் தரவு தேவை, பகுப்பாய்விற்கான பொருத்தமான அம்சங்கள் மற்றும் அளவுருக்கள் மற்றும் கணிப்புக்கு பொருத்தமான இயந்திர கற்றல் மாதிரிகளை உருவாக்குதல் உள்ளிட்ட பல சவால்களை பயிர் முன்கணிப்பின் சிக்கல் உள்ளடக்கியது. மேலும், காலநிலை மற்றும் மண் நிலைகளில் பிராந்திய மாறுபாடுகள், அத்துடன் பூச்சிகள் மற்றும் பிற சுற்றுச்சூழல் காரணிகளின் இருப்பு போன்ற காரணிகளாலும் கணிப்பு துல்லியம் பாதிக்கப்படலாம். இந்த சவால்களை எதிர்கொள்ள, பயிர் முன்கணிப்பு துறையில் ஆராய்ச்சியாளர்கள் மற்றும் டெவலப்பர்கள் தரவு முன் செயலாக்கம், அம்சம் தேர்வு, ஆழமான கற்றல் மாதிரி தேர்வு மற்றும் செயல்திறன் மதிப்பீடு உள்ளிட்ட பல்வேறு நுட்பங்களை உருவாக்கியுள்ளனர். இந்த நுட்பங்கள் வானிலை தரவு, மண் தரவு மற்றும் பயிர் தரவு போன்ற பல்வேறு வகையான தரவுகளின் பயன்பாட்டை உள்ளடக்கியிருக்கலாம், அத்துடன் பல அடுக்கு பெர்செப்ட்ரான் அல்காரிதம் மற்றும் கன்வல்யூஷனல் நியூரல் நெட்வொர்க் அல்காரிதம் போன்ற பல்வேறு ஆழமான கற்றல் வழிமுறைகள். இறுதியில், பயிர் முன்கணிப்பின் வெற்றியானது, பல்வேறு ஆதாரங்களில் இருந்து தரவை துல்லியமாகவும் நம்பகத்தன்மையுடனும் பகுப்பாய்வு செய்யும் அமைப்பின் திறனைப் பொறுத்தது, பின்னர் பயிர் விளைச்சல் மற்றும் பிற தொடர்புடைய அளவீடுகளை அதிக அளவு துல்லியத்துடன் கணிக்க முடியும். இந்த சவால்களை எதிர்கொள்வதன் மூலம், பயிர் முன்கணிப்பு விவசாய உற்பத்தித்திறன் மற்றும் நிலைத்தன்மையை மேம்படுத்துவதற்கும், மேலும் திறமையான மற்றும் பயனுள்ள விவசாய முறைகளை மேம்படுத்துவதற்கும் உதவுகிறது.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

1. INTRODUCTION

1.1 PURPOSE

Crop yield prediction is one of the challenging tasks in agriculture. It plays an essential role in decision making at global, regional, and field levels. The prediction of crop yield is based on soil, meteorological, environmental, and crop parameters. Decision support models are broadly used to extract significant crop features for prediction. Precision agriculture focuses on monitoring (sensing technologies), management information systems, variable rate technologies, and responses to inter- and intra-variability in cropping systems. The benefits of precision agriculture involve increasing crop yield and crop quality, while reducing the environmental impact. Predicting crop yield is crucial to addressing emerging challenges in food security, particularly in an era of global climate change. Accurate yield predictions not only help farmers make informed economic and management decisions but also support famine prevention efforts. Underlying crop yield prediction is a fundamental research question in plant biology, which is to understand how plant phenotype is determined by genotype (G), environment (E), management (M), and their interactions. Machine learning models have been successfully used for crop yield prediction, including stepwise multiple linear regression, random forest, neural networks, convolutional neural networks, recurrent neural networks, weighted histograms regression, interaction-based model, and association rule mining and decision tree. Most of these studies were based on environmental and managerial variables only, due to lack of publicly available genotype data at the state or national scale. Some studies explored the relationship between genotype and grain yield from regional yield trials from a plant breeding perspective, which would be hard to scale up to state-wide or nationwide predictions. Many machine learning algorithms are scalable to large datasets and have reasonably high prediction accuracy. Agriculture, since its invention and inception, be the prime and pre-eminent activity of every culture and civilization throughout the history of mankind. It is not only an enormous aspect of the growing economy, but it’s essential for us to survive. It’s also a crucial sector for Indian economy and also human future. It also contributes an outsized portion of employment. Because the time passes the requirement for production has been increased exponentially. So as to produce in mass quantity people are using technology in an exceedingly wrong way. New sorts of hybrid varieties are produced day by day.

1.2 SCOPE

Data mining is the computing process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems. It is an interdisciplinary subfield of computer science. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD. Data mining (the analysis step of the "Knowledge Discovery in Databases" process, or KDD), a field at the intersection of computer science and statistics, is the process that attempts to discover patterns in large data sets. It utilizes methods at the intersection of artificial intelligence, machine learning, statistics, and systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use The actual data mining task is the semi-automatic or automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining, sequential pattern mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system.

1.3 OVERVIEW

Crop prediction based on text data and soil datasets can be achieved through the application of machine learning algorithms. The process involves using both the text data and soil datasets to train a machine learning model, which will then be able to predict the most suitable crop to plant in a particular area. The first step in this process is to collect the necessary data. The text data can be obtained from various sources such as weather reports, previous crop yield reports, and soil quality reports. The soil datasets can be obtained from soil surveys, laboratory tests, or remote sensing data. Once the data is collected, it needs to be preprocessed to ensure that it is in a format that can be used by the machine learning model. This may involve cleaning the data, removing any duplicates, and converting it to a format that can be easily read by the machine learning algorithm. The next step is to train the machine learning model using the preprocessed data. The model can be trained using various algorithms such as decision trees, support vector machines, and neural networks. During the training process, the algorithm will learn to recognize patterns in the data and make predictions based on those patterns. Finally, the trained model can be used to predict the most suitable crop to plant in a particular area. The model will take into account the soil quality, weather conditions, and other relevant factors to make its prediction. This can help farmers make informed decisions about what crops to plant, which can increase yields and reduce the risk of crop failure.

CHAPTER 2

2. LITERATURE SURVEY

2.1 TITLE: MACHINE LEARNING FOR LARGE-SCALE CROP YIELD FORECASTING, 2021

AUTHOR: DILLI PAUDEL

Many studies have applied machine learning to crop yield prediction with a focus on specific case studies. The data and methods they used may not be transferable to other crops and locations. On the other hand, operational large-scale systems, such as the European Commission's MARS Crop Yield Forecasting System (MCYFS), do not use machine learning. Machine learning is a promising method especially when large amounts of data are being collected and published. We combined agronomic principles of crop modeling with machine learning to build a machine learning baseline for large-scale crop yield forecasting. The baseline is a workflow emphasizing correctness, modularity and reusability. For correctness, we focused on designing explainable predictors or features (in relation to crop growth and development) and applying machine learning without information leakage. We created features using crop simulation outputs and weather, remote sensing and soil data from the MCYFS database. We emphasized a modular and reusable workflow to support different crops and countries with small configuration changes. The workflow can be used to run repeatable experiments (e.g., early season or end of season predictions) using standard input data to obtain reproducible results. The results serve as a starting point for further optimizations. In our case studies, we predicted yield at regional level for five crops (soft wheat, spring barley, sunflower, sugar beet, potatoes) and three countries (the Netherlands (NL), Germany (DE), France (FR)). We compared the performance with a simple method with no prediction skill, which either predicted a linear yield trend or the average of the training set. We also aggregated the predictions to the national level and compared with past MCYFS forecasts

ADVANTAGES

- Crop yields are detected with accuracy

DISADVANTAGES

- It cannot accurately forecast the outcome.

2.2 TITLE: CROP YIELD PREDICTION THROUGH PROXIMAL SENSING AND MACHINE LEARNING ALGORITHMS, 2020

AUTHOR: FARHAT ABBAS

Proximal sensing techniques can potentially survey soil and crop variables responsible for variations in crop yield. The full potential of these precision agriculture technologies may be exploited in combination with innovative methods of data processing such as machine learning (ML) algorithms for the extraction of useful information responsible for controlling crop yield. Four ML algorithms, namely linear regression (LR), elastic net (EN), k-nearest neighbor (k-NN), and support vector regression (SVR), were used to predict potato (Solanum tuberosum) tuber yield from data of soil and crop properties collected through proximal sensing. Six fields in Atlantic Canada including three fields in Prince Edward Island (PE) and three fields in New Brunswick (NB) were sampled, over two (2017 and 2018) growing seasons, for soil electrical conductivity, soil moisture content, soil slope, normalized-difference vegetative index (NDVI), and soil chemistry. For the growing seasons of 2017 and 2018, the data about horizontal and vertical components of soil electrical conductivity, soil moisture content, field slope, soil pH, SOM, normalized difference vegetative index, and potato tuber yield were named as PE-2017, PE-2018, NB-2017 and NB-2018 for Prince Edward Island and New Brunswick fields. Modeling techniques were employed to generate yield predictions with statistical parameters from the collected data. The SVR models outperformed all other models for all four datasets with RMSE of 5.97, 4.62, 6.60, and 6.17 t/ha, respectively. The performance of k-NN remained poor except for PE-2018.

ADVANTAGES

- Extract the features from uploaded datasets

DISADVANTAGES

- It doesn’t work well with the large dataset

2.3 TITLE: IMPACT OF BEST MANAGEMENT PRACTICES ON SUSTAINABLE CROP PRODUCTION AND CLIMATE RESILIENCE IN SMALLHOLDER FARMING SYSTEMS OF SOUTH ASIA, 2021

AUTHOR:K.H. ANANTHA

The study identified lack of data as a major gap in both in situ conservation measures and ex situ rainwater harvesting intervention studies. Most of the studies on in situ conservation measures were undertaken at research stations and mostly pertained to major cereals such as rice, wheat and maize along with limited data on oilseeds and pulses. Similarly, limited studies on ex situ rainwater harvesting interventions were available and within a limited time period. Data monitoring needs to be strengthened both at micro and meso scale landscape to understand the ecosystem trade-offs between upstream and downstream ecologies under different rainfall, soil type and land slope conditions. Based on the current review, we envision a huge potential to integrate in situ conservation and ex situ rainwater harvesting technologies to address water scarcity and build system level resilience. A large part of the Indian subcontinent, especially the central and eastern parts, receive medium to high rainfall (800–2000 mm/year). However, they undergo water scarcity during post- monsoon season. Integrating both the technologies has great potential to overcome these challenges and achieve sustainable crop intensification. This paper reviews peer reviewed literature on different best water management practices in different agro-ecologies of the Indian subcontinent.

ADVANTAGES

- Identify the crop details with accuracy

DISADVANTAGES

- More storage space required

2.4 TITLE: AN APPROACH FOR PREDICTION OF CROP YIELD USING MACHINE LEARNING AND BIG DATA TECHNIQUES, 2021

AUTHOR: KODIMALAR PALANIVEL

Estimating agricultural yield prior to harvest is an Estimating agricultural yield prior to harvest is an important issue in agriculture, as the changes in crop yield from year-to-year influence international business, food supply, and global market prices. Also, early prediction of crop yield provides useful information to policy planners. Appropriate prediction of crop productivity is required for efficient planning of land usage and economic policy. In recent times, forecasting of crop productivity at the within-field level has increased. The most influencing factor for crop productivity is weather conditions. If the weather-based prediction is made more precise, then farmers can be alerted well in advance so that the major loss can be mitigated and would be helpful for economic growth. The prediction will also aid the farmers to make decisions such as the choice of alternative crops or to discard a crop at an early stage in case of critical situations. Further, predicting crop yield can facilitate the farmers to have a better vision on cultivation of seasonal crop and its scheduling. Thus, it is necessary to simulate & predict the crop yield before cultivation for efficient crop management and expected outcome. As there exists a non-linear relationship between crop yield and the factors influencing crop, machine learning techniques might be efficient for yield predictions. Machine Learning involves problems in which the input and output relationship is not known. Learning specifies the automatic acquirement of structural descriptions. In contrast to traditional statistical methods, machine learning does not make assumptions about the exact construct of the data model, which describes the data. This feature is very helpful to describe complex non-linear behaviors such as a crop yield prediction. Machine learning is a part of artificial intelligence employed to build an intelligent system

ADVANTAGES

- Estimate the agriculture values from datasets

DISADVANTAGES

- It doesn't efficient to train the dataset

2.5 TITLE: COUPLING MACHINE LEARNING AND CROP MODELING IMPROVES CROP YIELD PREDICTION IN THE US CORN BELT, 2021

AUTHOR: MOHSEN SHAHHOSSEIN

This study investigates whether coupling crop modelling and machine learning (ML) improves corn yield predictions in the US Corn Belt. The main objectives are to explore whether a hybrid approach (crop modeling+ML) would result in better predictions, investigate which combinations of hybrid models provide the most accurate predictions, and determine the features from the crop modelling that are most effective to be integrated with ML for corn yield prediction. Five ML models (linear regression, LASSO, LightGBM, random forest, and XGBoost) and six ensemble models have been designed to address the research question. The results suggest that adding simulation crop model variables (APSIM) as input features to ML models can decrease yield prediction root mean squared error (RMSE) from 7 to 20%. Furthermore, we investigated partial inclusion of APSIM features in the ML prediction models and we found soil moisture related APSIM variables are most influential on the ML predictions followed by crop-related and phenology-related variables. Finally, based on feature importance measure, it has been observed that simulated APSIM average drought stress and average water table depth during the growing season are the most important APSIM inputs to ML. This result indicates that weather information alone is not sufficient and ML models need more hydrological inputs to make improved yield predictions. On the other hand, machine learning (ML) intends to make predictions by finding connections between input and response variables. Unlike simulation crop models, ML includes methods in which the system “learns” a transfer function to predict the desired output based on the provided inputs, rather than the researcher providing the transfer function. In addition, it is more easily applicable than simulation crop models as it does not require expert knowledge and user skills to calibrate the model, has lower runtimes, and less data storage constraints.

ADVANTAGES

- Comparative analysis for multiple algorithms

DISADVANTAGES

- Time consuming

CHAPTER 3

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Agriculture is one of the important industrial sectors in India and the country’s economy is highly dependent on it for rural sustainability. Due to some factors like climate changes, unpredicted rainfall, decrease of water level, use of pesticides excessively etc. The level of agriculture in India is decreased. To know the level of production we performed descriptive analytics on the agriculture data. The main objective of this research work is to provide a methodology so that it can perform descriptive analytics on crop yield production in an effective manner. Although, some studies revealed statistical information about the agriculture in India, few studies have investigated crop prediction based on the historic climatic and production data. Classification algorithms accept been acclimated for assorted purposes including classification, clustering, agent quantization, arrangement association, action approximation, forecasting, ascendancy applications and optimization. In existing system implement Principal component regression is a two steps method. Firstly, in PCR method stores the variables and reduces the dimensionality of the data and again stores in the structure of table. It extracts the most variation of data and performs feature selection so that dimensionality could be reduced. And find the first factor in the direction maximizes the dispersion of the observation, and other factor should be also maximizing in the dispersion of diagonal of the first factor. And then we can rotate the factor perpendicular and for more dimensional data we can continue in a combined approach. The result of PCR is the representation of the variation of the sample of less dimensional data. Secondly, we try to fit a linear regression between the samples on the factors which are mostly correlated with factors. PCR solves the multidimensional space problem and co-linearity problem in an efficient manner.

3.1.1 LIMITATIONS OF THE EXISTING SYSTEM

- Labeled data-based classification
- Provide high number of false positive
- Binary classification can be occurred
- Computational complexity

3.2 PROPOSED SYSTEM

Predicting crop yield based on the environmental, soil, water and crop parameters has been a potential research topic. Deep-learning-based models are broadly used to extract significant crop features for prediction. Though these methods could resolve the yield prediction problem there exist the following inadequacies: Unable to create a direct non-linear or linear mapping between the raw data and crop yield values; and the performance of those models highly relies on the quality of the extracted features. Deep reinforcement learning provides direction and motivation for the aforementioned shortcomings. Combining the intelligence of reinforcement learning and deep learning, deep reinforcement learning builds a complete crop yield prediction framework that can map the raw data to the crop prediction values. In this project, we use deep neural networks to predict yield, check yield, and yield difference of corn hybrids from genotype and environment data. Deep neural networks belong to the class of representation learning models that can find the underlying representation of data without handcrafted input of features. Deep neural networks have multiple stacked non-linear layers which transform the raw input data into higher and more abstract representation at each stacked layer. As such, as the network grows deeper, more complex features are extracted which contribute to the higher accuracy of results. Given the right parameters, deep neural networks are known to be universal approximate functions, which means that they can approximate almost any function, although it may be very challenging to find the right parameters

3.2.1 ADVANTAGES OF THE PROPOSED SYSTEM

- Accuracy is high
- Parallel processing
- Reduce number of false positive rate
- Time and computational complexity can be reduced

3.3 SYSTEM ARCHITECTURE

System architecture refers to the overall design and structure of a computer system or software application. It involves the organization and interconnection of hardware components, software modules, and communication protocols to ensure that the system functions efficiently, reliably, and securely. System architecture typically includes several key components, such as the central processing unit (CPU), memory, input/output devices, storage devices, network interfaces, and operating system software. The architecture also includes the application software and the database management system (DBMS), which manage the data and processes of the system.

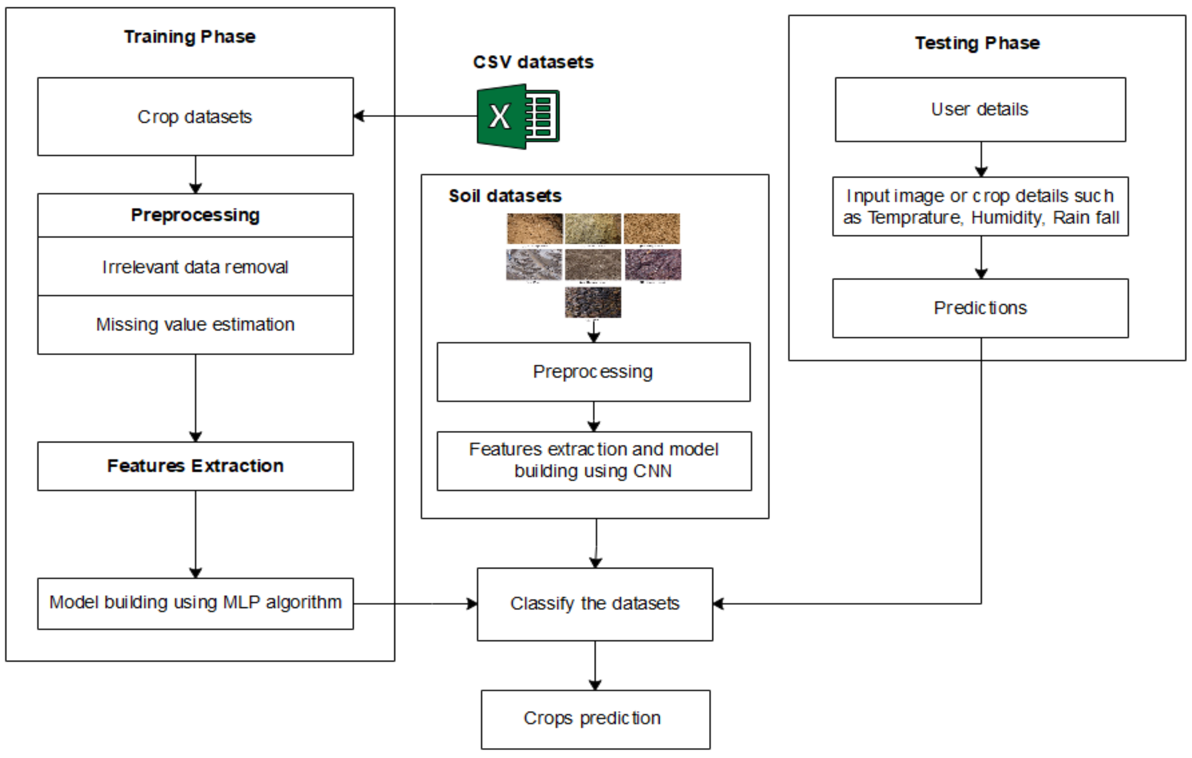


FIG 3.1 SYSTEM ARCHITECTURE

In this architecture implemented training and testing phase. In training phase, admin train the datasets for both CSV and soil datasets. In testing phase, input the crop and soil details to predict the crops.

CHAPTER 4

4. SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

• Processor	: Intel core processor 2.6.0 GHZ
• RAM	: 1GB
• Hard disk	: 160 GB
• Compact Disk	: 650 Mb
• Keyboard	: Standard keyboard
• Monitor	: 15-inch color monitor

4.2 SOFTWARE REQUIREMENTS

• Operating system	: Windows OS
• Front End	: PYTHON
• Back End	: MYSQL
• IDE	: PYCHARM

CHAPTER 5

5. MODULES

5.1 MODULES

- DATASETS ACQUISITION
- PREPROCESSING
- FEATURES EXTRACTION
- MODEL TRAINING
- CROP DETAILS

5.2 MODULES DESCRIPTION

5.2.1 DATASETS ACQUISITION

In this module, we can upload the crop yield datasets in the form of CSV file format. And also store the data in database for future purpose. The dataset includes the temperature, rain fall, pH value, nitrogen, phosphorus, potassium values. These values are extracted from Kaggle website and values are stored in the form of integer values. And also upload the soil datasets that are collected from KAGGLE sources in terms of image format

5.2.2 PREPROCESSING

Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data preparation and filtering steps can take considerable amount of processing time. In this module, we can eliminate the irrelevant values and also estimate the missing values of data. Finally provide structured datasets. Then in soil image, we can analyze the noises and remove the noises using Median filtering algorithm.

Median filtering is a widely used technique for removing noise from images, including soil images. It works by replacing each pixel in an image with the median value of its neighbouring pixels. In the case of salt and pepper noise, which is common in soil images, median filtering can be particularly effective at removing the noise while preserving the edges and details of the image.

5.2.3 FEATURES EXTRACTION

Feature selection refers to the process of reducing the inputs for processing and analysis, or of finding the most meaningful inputs. In this module, select the multiple features from uploaded datasets. And train the datasets with various crop labels with multiple attribute values. We can train the datasets to multiple crops such as rice, maize and so on. Features are collected from soil image datasets such as color, shape and texture datasets. Build a CNN architecture that is appropriate for the task of soil feature extraction. This can involve choosing the number and size of convolutional and pooling layers, selecting an activation function, and deciding on the loss function and optimizer.

5.2.4 MODEL TRAINING

In this module, we can implement multi-layer perceptron algorithm to classify the uploaded datasets. MLP algorithm is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. Crop prediction using MLP (Multilayer Perceptron) algorithm is a machine learning technique that involves the use of artificial neural networks to predict crop yields based on environmental data. The MLP algorithm is a type of feedforward neural network that is capable of learning complex relationships between input variables and output predictions. The process of crop prediction using MLP algorithm typically involves several steps. First, historical data on crop yields, weather conditions, soil quality, and other environmental factors are collected and pre-processed. The data is then divided into training and validation sets, with the training set used to train the MLP model and the validation set used to evaluate its performance. The MLP model consists of an input layer, one or more hidden layers, and an output layer. Each layer contains a set of neurons, which are connected to the neurons in adjacent layers by weighted connections. During training, the weights of these connections are adjusted in order to minimize the error between the predicted crop yields and the actual crop yields. Once the MLP model has been trained and validated, it can be used to make predictions about future crop yields based on new input data. The input data typically includes information on weather conditions, soil quality, crop type, and other relevant factors. The MLP model processes this inputs data and produces an output prediction of the expected crop yield.

Then image is training using CNN algorithm. Split the dataset into training, validation, and testing sets. The training set is used to train the CNN, while the validation set is used to tune the model's hyperparameters and prevent overfitting. The testing set is used to evaluate the final performance of the model. Build a CNN architecture that is appropriate for the task of soil image classification. This can involve choosing the number and size of convolutional and pooling layers, selecting an activation function, and deciding on the loss function and optimizer. Train the CNN on the training set using backpropagation and gradient descent. Monitor the model's performance on the validation set and adjust the hyperparameters as necessary to prevent overfitting. Evaluate the final performance of the CNN on the testing set. This can involve calculating metrics such as accuracy, precision, recall, and F1 score.

5.2.5 CROP PREDICTION

Appropriate prediction of crop productivity is required for efficient planning of land usage and economic policy. In recent times, forecasting of crop productivity at the within-field level has increased. The most influencing factor for crop productivity is weather conditions. In this module, we can test the datasets by using deep learning algorithm. Finally provide the details about crops and yield information with improved accuracy rate. The accuracy of the project is calculated in terms of Precision, Recall and F-measure values. After the model has been validated, it can be deployed for use in the real world. In this module, we predict the crops based on user input and classify with trained model file This may involve integrating the model into an existing system for crop management or using it to make predictions for a specific region or field.

CHAPTER 6

6. SYSTEM TESTING AND MAINTENANCE

6.1 TESTING DESCRIPTION

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing can be stated as the process of validating and verifying that a computer program/application/product:

1. meets the requirements that guided its design and development,

2. works as expected,

3. It can be implemented with the same characteristics, and satisfies the needs of stakeholders.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology. Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test-driven development and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed.

6.1.1 OVERVIEW OF TESTING

Testing can never completely identify all the defects within software. Instead, it furnishes a criticism or comparison that compares the state and behaviour of the product against oracles principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria. A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed. Every software product has a target audience. For example, the audience for video game software is completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment

6.2 TESTING METHODS

6.2.1 Static vs. Dynamic Testing

There are many approaches to software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing can be omitted, and unfortunately in practice often is. Dynamic testing takes place when the program itself is used. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment. The box approach Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

6.2.2 White-Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end users. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., in-circuit testing (ICT). While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements. Techniques used in white-box testing include: 1. API testing (application programming interface) - testing of the application using public and private APIs 2. Code coverage - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once) 3. Fault injection methods - intentionally introducing faults to gauge the efficacy of testing strategies

6.2.3 Static Testing Methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for: 1. Function coverage, which reports on functions executed 2. Statement coverage, which reports on the number of lines executed to complete the test 100% statement coverage ensures that all code paths, or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly.

6.2.4 Black-Box Testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The tester is only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. Specification based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations. One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested. This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

6.2.5 Grey-Box Testing

Grey-box testing involves having knowledge of internal data structures and algorithms for purposes of designing tests, while executing those tests at the user, or black-box level. The tester is not required to have full access to the software's source code. Manipulating input data and formatting output do not qualify as grey-box, because the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test. However, modifying a data repository does qualify as grey-box, as the user would not normally be able to change the data outside of the system under test. Grey-box testing may also include reverse engineering to determine, for instance, boundary values or error messages. By knowing the underlying concepts of how the software works, the tester makes better-informed testing choices while testing the software from outside. Typically, a grey-box tester will be permitted to set up his testing environment; for instance, seeding a database; and the tester can observe the state of the product being tested after performing certain actions. For instance, in testing a database product he/she may fire an SQL query on the database and then observe the database, to ensure that the expected changes have been reflected. Grey-box testing implements intelligent test scenarios, based on limited information. This will particularly apply to data type handling, exception handling, and so on.

Visual testing

The aim of visual testing is to provide developers with the ability to examine what was happening at the point of software failure by presenting the data in such a way that the developer can easily find the information he requires, and the information is expressed clearly. At the core of visual testing is the idea that showing someone a problem (or a test failure), rather than just describing it, greatly increases clarity and understanding. Visual testing therefore requires the recording of the entire test process – capturing everything that occurs on the test system in video format. Output videos are supplemented by real-time tester input via picture-in-a-picture webcam and audio commentary from microphones. Visual testing provides a number of advantages. The quality of communication is increased dramatically because testers can show the problem (and the events leading up to it) to the developer as opposed to just describing it and the need to replicate test failures will cease to exist in many cases. The developer will have all the evidence he requires of a test failure and can instead focus on the cause of the fault and how it should be fixed. Visual testing is particularly well-suited for environments that deploy agile methods in their development of software, since agile methods require greater communication between testers and developers and collaboration within small teams. Visual testing is gathering recognition in customer acceptance and usability testing, because the test can be used by many individuals involved in the development process. For the customer, it becomes easy to provide detailed bug reports and feedback, and for program users, visual testing can record user actions on screen, as well as their voice and image, to provide a complete picture at the time of software failure for the developer.

6.3 TESTING LEVELS

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit-, integration-, and system testing that are distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

6.3.1 Unit Testing

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

6.3.2 Integration Testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localised more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

6.3.3 System Testing

System testing tests a completely integrated system to verify that it meets its requirements.

6.3.4 Acceptance Testing

At last, the system is delivered to the user for Acceptance testing

6.4 TESTING APPROACHES

6.4.1 Bottom-Up

Bottom-Up Testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower-level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

6.4.2 Top-Down

Top-Down Testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

6.5 OBJECTIVES OF TESTING

6.5.1 Installation Testing

An installation test assures that the system is installed correctly and working at actual customer's hardware.

6.5.2 Compatibility Testing

A common cause of software failure (real or perceived) is a lack of its compatibility with other application software, operating systems (or operating system versions, old or new), or target environments that differ greatly from the original (such as a terminal or GUI application intended to 35 be run on the desktop now being required to become a web application, which must render in a web browser). For example, in the case of a lack of backward compatibility, this can occur because the programmers develop and test software only on the latest version of the target environment, which not all users may be running. This results in the unintended consequence that the latest work may not function on earlier versions of the target environment, or on older hardware that earlier versions of the target environment was capable of using. Sometimes such issues can be fixed by proactively abstracting operating system functionality into a separate program module or library.

6.5.3 Smoke and Sanity Testing

Sanity testing determines whether it is reasonable to proceed with further testing. Smoke testing is used to determine whether there are serious problems with a piece of software, for example as a build verification test.

6.5.4 Regression Testing

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, or old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly stops working as intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features. They can either be complete, for changes added late in the release or deemed to be risky, to 36 very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk.

6.5.5 Alpha Testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

6.5.6 Beta Testing

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

6.5.7 Functional vs Non-functional Testing

Functional testing refers to activities that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work." Non-functional testing refers to aspects of the software that may not be related to a specific function or user action, such as scalability or other performance, behaviour under certain constraints, or security.

6.5.8 Destructive Testing

Destructive testing attempts to cause the software or a sub-system to fail. It verifies that the software functions properly even when it receives invalid or unexpected inputs, thereby establishing the robustness of input validation and error-management routines. Software fault injection, in the form of fuzzing, is an example of failure testing. Various commercial non-functional testing tools are linked from the software fault injection page; there are also numerous open-source and free software tools available that perform destructive testing.

6.5.9 Software Performance Testing

Performance testing is in general executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. Load testing is primarily concerned with testing that the system can continue to operate under a specific load, whether that be large quantities of data or a large number of users. This is generally referred to as software scalability. The related load testing activity of when performed as a non-functional activity is often referred to as endurance testing. Volume testing is a way to test software functions even when certain components (for example a file or database) increase radically in size. Stress testing is a way to test reliability under unexpected or rare workloads.

6.5.10 Usability Testing

Usability testing is needed to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application. It is important to check interface is working properly or not as planned

6.5.11 Accessibility Testing

Accessibility testing may include compliance with standards such as:

1. Americans with Disabilities Act of 1990

2. Section 508 Amendment to the Rehabilitation Act of 1973 3. Web Accessibility Initiative (WAI) of the World Wide Web Consortium

6.5.12 Security Testing

Security testing is essential for software that processes confidential data to prevent system intrusion by hackers. So, it is important to perform security testing to find loop holes in the developed software.

6.5.13 Development Testing

Development Testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it.

CHAPTER 7

7. LANGUAGE SPECIFICATION

7.1 FRONT END: PYTHON

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages due to its simplicity, readability, and versatility. One of the key features of Python is its easy-to-learn syntax, which makes it accessible to both novice and experienced programmers. It has a large standard library that provides a wide range of modules for tasks such as file I/O, networking, regular expressions, and more. Python also has a large and active community of developers who contribute to open source libraries and packages that extend its capabilities. Python is an interpreted language, which means that it is executed line-by-line by an interpreter rather than compiled into machine code like C or C++. This allows for rapid development and testing, as well as easier debugging and maintenance of code. Python is used for a variety of applications, including web development frameworks such as Django and Flask, scientific computing libraries such as NumPy and Pandas, and machine learning libraries such as TensorFlow and PyTorch. It is also commonly used for scripting and automation tasks due to its ease of use and readability. Overall, Python is a powerful and versatile programming language that is widely used in a variety of domains due to its simplicity, ease of use, and active community.

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture."Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity.[When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar. A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

Python also has a large and active community of developers who contribute to a wide range of open-source libraries and tools, making it easy to find and use pre-built code to solve complex problems.

Python has a wide range of applications, including:

Data Science: Python is one of the most popular languages for data science, thanks to libraries like NumPy, Pandas, and Matplotlib that make it easy to manipulate and visualize data.

Machine Learning: Python is also widely used in machine learning and artificial intelligence, with libraries like TensorFlow, Keras, and Scikit-learn that provide powerful tools for building and training machine learning models.

Web Development: Python is commonly used in web development, with frameworks like Django and Flask that make it easy to build web applications and APIs.

Scientific Computing: Python is used extensively in scientific computing, with libraries like SciPy and SymPy that provide powerful tools for numerical analysis and symbolic mathematics.

In addition to its versatility and ease of use, Python is also known for its portability and compatibility. Python code can be run on a wide range of platforms, including Windows, macOS, and Linux, and it can be integrated with other languages like C and Java.

Overall, Python is a powerful and versatile programming language that is well-suited for a wide range of applications, from data science and machine learning to web development and scientific computing. Its simplicity, readability, and large community of developers make it an ideal choice for beginners and experts alike.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.

2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

One of the strengths of Python is its rich ecosystem of third-party libraries and tools. These libraries provide a wide range of functionality, from scientific computing and data analysis to web development and machine learning. Some popular Python libraries and frameworks include:

NumPy: a library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.

Pandas: a library for data manipulation and analysis in Python, providing support for reading and writing data in a variety of formats, as well as powerful tools for manipulating and analyzing data.

Matplotlib: a plotting library for Python that provides a variety of visualization tools, including line plots, scatter plots, bar plots, and more.

TensorFlow: an open-source machine learning library for Python that provides a variety of tools and algorithms for building and training machine learning models.

Django: a popular web framework for Python that provides a full-stack framework for building web applications, with support for everything from URL routing to user authentication and database integration.

Python's popularity has also led to a large and active community of developers who contribute to open-source projects and share code and resources online. This community provides a wealth of resources for learning Python, including tutorials, online courses, and forums for asking and answering questions.

Overall, Python is a versatile and powerful programming language that is well-suited for a wide range of applications. Its simplicity, flexibility, and wide range of library.

7.2 TENSORFLOW LIBARIES IN PYTHON

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It is one of the most popular libraries for building and training machine learning models, especially deep neural networks. TensorFlow allows developers to build complex models with ease, including image and speech recognition, natural language processing, and more. One of the key features of TensorFlow is its ability to handle large-scale datasets and complex computations, making it suitable for training deep neural networks. It allows for parallelization of computations across multiple CPUs or GPUs, allowing for faster training times. TensorFlow also provides a high-level API called Keras that simplifies the process of building and training models.

TensorFlow offers a wide range of tools and libraries that make it easy to integrate with other Python libraries and frameworks. It has built-in support for data preprocessing and visualization, making it easy to prepare data for training and analyze model performance. One of the major advantages of TensorFlow is its ability to deploy models to a variety of platforms, including mobile devices and the web. TensorFlow Lite is a mobile-optimized version of TensorFlow that is designed for deploying models on Android, iOS, and other mobile platforms. TensorFlow.js is a JavaScript library that allows for training and deployment of models directly in the browser. TensorFlow provides a wide range of features and tools for building and training machine learning models. Some of the key features of TensorFlow include:

Graph-based computation: TensorFlow uses a graph-based computation model, which allows for efficient execution of computations across multiple devices and CPUs/GPUs.

Automatic differentiation: TensorFlow provides automatic differentiation, which allows for efficient computation of gradients for use in backpropagation algorithms.

High-level APIs: TensorFlow provides high-level APIs, such as Keras, that allow developers to quickly build and train complex models with minimal code.

Preprocessing and data augmentation: TensorFlow provides a range of tools for preprocessing and data augmentation, including image and text preprocessing, data normalization, and more.

Distributed training: TensorFlow supports distributed training across multiple devices, CPUs, and GPUs, allowing for faster training times and more efficient use of resources.

7.3 PYCHARM

PyCharm has a highly customizable user interface, allowing users to tailor the IDE to their specific needs and preferences. This includes customizing the color scheme, key mappings, and even the appearance of the code editor. PyCharm also supports various plugins and extensions, enabling users to add new functionality to the IDE or integrate with external tools and services. In addition to its development features, PyCharm also includes tools for project management, such as version control integration with Git, Mercurial, and Subversion. It also provides support for task management and issue tracking through integration with tools like Jira and Trello. PyCharm has a strong focus on code quality and maintainability, providing tools for code inspections, unit testing, and code coverage analysis. This can help developers catch errors and ensure that their code is maintainable and scalable over time. PyCharm also supports multiple Python versions and virtual environments, allowing users to switch between different versions of Python or create isolated environments for different projects. This can help ensure compatibility and prevent version conflicts between different projects. Overall, PyCharm is a comprehensive IDE that can greatly improve productivity and code quality for Python developers. Its extensive feature set, customization options, and focus on code quality make it a popular choice for Python development.

CHAPTER 8

8.CODING AND RESULT SCREENSHOTS

SOURCE CODE

```
from flask import Flask, render_template, flash, request, session,send_file

from flask import render_template, redirect, url_for, request

#from wtforms import Form, TextField, TextAreaField, validators, StringField, SubmitField

from werkzeug.utils import secure_filename

import datetime

import mysql.connector

import sys

import pickle

import cv2

import numpy as np

app = Flask(__name__)

app.config['DEBUG']

app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

@app.route("/")

def homepage():

    return render_template('index.html')

@app.route("/AdminLogin")

def AdminLogin():

   return render_template('AdminLogin.html')

@app.route("/UserLogin")

def UserLogin():
  return render_template('UserLogin.html')
@app.route("/NewUser")
def NewUser():
   return render_template('NewUser.html')
@app.route("/NewQuery1")
def NewQuery1():
   return render_template('NewQueryReg.html')
@app.route("/UploadDataset")
def UploadDataset():
   return render_template('ViewExcel.html')
@app.route("/AdminHome")
def AdminHome():
   conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')
   cur = conn.cursor()
   cur.execute("SELECT * FROM regtb ")
   data = cur.fetchall()
   return render_template('AdminHome.html',data=data)
@app.route("/UserHome")
def UserHome():
   user = session['uname']
   conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')
   # cursor = conn.cursor()
   cur = conn.cursor()
```

```
    cur.execute("SELECT * FROM regtb where username='" + user + "'")
    data = cur.fetchall()
    return render_template('UserHome.html',data=data)
@app.route("/UQueryandAns")
def UQueryandAns():
    uname = session['uname']
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
    # cursor = conn.cursor()
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where UserName='" + uname + "' and DResult='waiting'")
    data = cur.fetchall()
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
    # cursor = conn.cursor()
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where UserName='" + uname + "' and DResult !='waiting'")
    data1 = cur.fetchall()
    return render_template('UserQueryAnswerinfo.html', wait=data, answ=data1 )
@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    error = None
    if request.method == 'POST':
        if request.form['uname'] == 'admin' or request.form['password'] == 'admin':
            conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
            # cursor = conn.cursor()
            cur = conn.cursor()
            cur.execute("SELECT * FROM regtb ")
            data = cur.fetchall()
            return render_template('AdminHome.html' , data=data)
        else:
         return render_template('index.html', error=error)
@app.route("/userlogin", methods=['GET', 'POST'])
def userlogin():
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['uname'] = request.form['uname']
        conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from regtb where username='" + username + "' and Password='" + password + "'")
        data = cursor.fetchone()
        if data is None:
            alert = 'Username or Password is wrong'
            render_template('goback.html', data=alert)
        else:
            print(data[0])
            session['uid'] = data[0]
            conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
            # cursor = conn.cursor()
            cur = conn.cursor()
            cur.execute("SELECT * FROM regtb where username='" + username + "' and Password='" + password + "'")
            data = cur.fetchall()
            return render_template('UserHome.html', data=data )
@app.route("/newuser", methods=['GET', 'POST'])
def newuser():
    if request.method == 'POST':
        name1 = request.form['name']
        gender1 = request.form['gender']
        Age = request.form['age']
        email = request.form['email']
        pnumber = request.form['phone']
        address = request.form['address']
        uname = request.form['uname']
        password = request.form['psw']
        conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 + "','" + Age + "','" +
email + "','" + pnumber + "','" + address + "','" + uname + "','" + password + "')")
        conn.commit()
        conn.close()
        # return 'file register successfully'
    return render_template('UserLogin.html')
@app.route("/newquery", methods=['GET', 'POST'])
def newquery():
    if request.method == 'POST':
        uname = session['uname']
        nitrogen = request.form['nitrogen']
        phosphorus = request.form['phosphorus']
        potassium = request.form['potassium']
        temperature = request.form['temperature']
        humidity = request.form['humidity']
        ph = request.form['ph']
        rainfall = request.form['rainfall']
        location = request.form['select']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO Querytb VALUES ('','" + uname + "','" + nitrogen + "','" + phosphorus
+ "','" + potassium + "','"+temperature+"','"+humidity +"','"+ ph+"','"+ rainfall
+"','waiting','','"+location+"')")
        conn.commit()
        conn.close()
        # return 'file register successfully'
        uname = session['uname']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')
        # cursor = conn.cursor()
        cur = conn.cursor()
        cur.execute("SELECT * FROM Querytb where UserName='" + uname + "' and
DResult='waiting'")
        data = cur.fetchall()
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')
        # cursor = conn.cursor()
        cur = conn.cursor()
        cur.execute("SELECT * FROM Querytb where UserName='" + uname + "' and DResult
!='waiting'")
        data1 = cur.fetchall()
        return render_template('UserQueryAnswerinfo.html', wait=data, answ=data1)
@app.route("/excelpost", methods=['GET', 'POST'])
def uploadassign():
    if request.method == 'POST':
        file = request.files['fileupload']
```

```
file_extension = file.filename.split('.')[1]

print(file_extension)

#file.save("static/upload/" + secure_filename(file.filename))

import pandas as pd

import matplotlib.pyplot as plt

df = ''

if file_extension == 'xlsx':

    df = pd.read_excel(file.read(), engine='openpyxl')

elif file_extension == 'xls':

    df = pd.read_excel(file.read())

elif file_extension == 'csv':

    df = pd.read_csv(file)

print(df)

import seaborn as sns

sns.countplot(df['label'], label="Count")

plt.savefig('static/images/out.jpg')

iimg = 'static/images/out.jpg'

#plt.show()

#df = pd.read_csv("./Heart/Heartnew.csv")

#def clean_dataset(df):

    #assert isinstance(df, pd.DataFrame), "df needs to be a pd.DataFrame"

    #df.dropna(inplace=True)

    #indices_to_keep = ~df.isin([np.nan, np.inf, -np.inf]).any(1)

    #return df[indices_to_keep].astype(np.float64)
```

```
#df = clean_dataset(df)

#print("Preprocessing Completed")

print(df)

# imprt pandas as pd

import matplotlib.pyplot as plt

# read-in data

# data = pd.read_csv('./test.csv', sep='\t') #adjust sep to your needs

import seaborn as sns

sns.countplot(df['label'], label="Count")

plt.show()

df.label = df.label.map({'rice': 0,

                         'maize': 1,

                         'chickpea': 2,

                         'kidneybeans': 3,

                         'pigeonpeas': 4,

                         'mothbeans': 5,

                         'mungbean': 6,

                         'blackgram': 7,

                         'lentil': 8,

                         'pomegranate': 9,

                         'banana': 10,

                         'mango': 11,

                         'grapes': 12,

                         'watermelon': 13,
                'muskmelon': 14,

                'apple': 15,

                'orange': 16,

                'papaya': 17,

                'coconut': 18,

                'cotton': 19,

                'jute': 20,

                'coffee': 21})
```

# Replacing the 0 values from ['Glucose','BloodPressure','SkinThickness','Insulin','BMI'] by NaN

```
df_copy = df.copy(deep=True)

df_copy[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']] = df_copy[

  ['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']].replace(0, np.NaN)

# Model Building

from sklearn.model_selection import train_test_split

df.drop(df.columns[np.isnan(df).any()], axis=1)

X = df.drop(columns='label')

y = df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)

from sklearn.neural_network import MLPClassifier

from sklearn.metrics import classification_report

classifier = MLPClassifier(random_state=0)

classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
        clreport = classification_report(y_test, y_pred)
        print("Accuracy on training set: {:.2f}".format(classifier.score(X_train, y_train)))
        print("Accuracy on test set: {:.3f}".format(classifier.score(X_test, y_test)))
        Tacc = "Accuracy on training set: {:.2f}".format(classifier.score(X_train, y_train))
        Testacc = "Accuracy on test set: {:.3f}".format(classifier.score(X_test, y_test))
        # Creating a pickle file for the classifier
        filename = 'crop-prediction-rfc-model.pkl'
        pickle.dump(classifier, open(filename, 'wb'))
        print("Training process is complete Model File Saved!")
        df= df.head(200)
        #read_csv(..., skiprows=1000000, nrows=999999)
        return render_template('ViewExcel.html', data=df.to_html(), dataimg=iimg
,tacc=Tacc,testacc=Testacc,report=clreport)
@app.route("/AdminQinfo")
def AdminQinfo():
    #uname = session['uname']
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')
    # cursor = conn.cursor()
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where  DResult='waiting'")
    data = cur.fetchall()
    return render_template('AdminQueryInfo.html', data=data )
@app.route("/answer")
```

```
def answer():
    Answer = ''
    Prescription=''
    id =  request.args.get('lid')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')
    cursor = conn.cursor()
    cursor.execute("SELECT  *  FROM Querytb where  id='" + id + "'")
    data = cursor.fetchone()
    if data:
        UserName = data[1]
        nitrogen = data[2]
        phosphorus = data[3]
        potassium = data[4]
        temperature = data[5]
        humidity = data[6]
        ph = data[7]
        rainfall = data[8]
    else:
        return 'Incorrect username / password !'
    nit = float(nitrogen)
    pho = float(phosphorus)
    po = float(potassium)
    te = float(temperature)
    hu = float(humidity)
    phh = float(ph)
    ra = float(rainfall)
    #age = int(age)
    filename = 'crop-prediction-rfc-model.pkl'
    classifier = pickle.load(open(filename, 'rb'))
    data = np.array([[nit, pho, po, te, hu, phh, ra ]])
    my_prediction = classifier.predict(data)
    print(my_prediction)
    crop = ''
    fertilizer = '
    if my_prediction == 0:
        Answer = 'Predict'
        crop = 'rice'
        fertilizer ='4 kg of gypsum and 1 kg of DAP/cent can be applied at 10 days after sowing'
    elif my_prediction == 1:
        Answer = 'Predict'
        crop = 'maize'
        fertilizer = 'The standard fertilizer recommendation for maize consists of 150 kg ha−1
NPK 14–23–14 and 50 kg ha−1 urea'
    elif my_prediction == 2:
        Answer = 'Predict'
        crop = 'chickpea'
        fertilizer = 'The generally recommended doses for chickpea include 20–30 kg nitrogen
(N) and 40–60 kg phosphorus (P) ha-1. If soils are low in potassium (K), an application of 17
to 25 kg K ha-1 is recommended'
```

```
    elif my_prediction == 3:
        Answer = 'Predict'
        crop = 'kidneybeans'
        fertilizer = 'It needs good amount of Nitrogen about 100 to 125 kg/ha'
    elif my_prediction == 4:
        Answer = 'Predict'
        crop = 'pigeonpeas'
        fertilizer = 'Apply 25 - 30 kg N, 40 - 50 k g P 2 O 5 , 30 kg K 2 O per ha area as Basal
dose at the time of sowing.'
    elif my_prediction == 5:
        Answer = 'Predict'
        crop = 'mothbeans'
        fertilizer = 'The applications of 10 kg N+40 kg P2O5 per hectare have proved the effective
starter dose'
    elif my_prediction == 6:
        Answer = 'Predict'
        crop = 'mungbean'
        fertilizer = 'Phosphorus and potassium fertilizers should be applied at 50-50 kg ha-1'
    elif my_prediction == 7:
        Answer = 'Predict'
        crop = 'blackgram'
        fertilizer = 'The recommended fertilizer dose for black gram is 20:40:40 kg NPK/ha.'
    elif my_prediction == 8:
        Answer = 'Predict'
        crop = 'lentil'
        fertilizer = 'The recommended dose of fertilizers is 20kg N, 40kg P, 20 kg K and 20kg S/ha.'
    elif my_prediction == 9:
        Answer = 'Predict'
        crop = 'pomegranate'
        fertilizer = 'The recommended fertiliser dose is 600–700 gm of N, 200–250 gm of P2O5 and 200–250 gm of K2O per tree per year'
    elif my_prediction == 10:
        Answer = 'Predict'
        crop = 'banana'
        fertilizer = 'Feed regularly using either 8-10-8 (NPK) chemical fertilizer or organic composted manure'
    elif my_prediction == 11:
        Answer = 'Predict'
        crop = 'mango'
        fertilizer = '50 gm zinc sulphate, 50 gm copper sulphate and 20 gm borax per tree/annum are recommended'
    elif my_prediction == 12:
        Answer = 'Predict'
        crop = 'grapes'
        fertilizer = 'Use 3 pounds (1.5 kg.) of potassium sulfate per vine for mild deficiencies or up to 6 pounds (3 kg.)'
    elif my_prediction == 13:
        Answer = 'Predict'
        crop = 'watermelon'
        fertilizer = 'Apply a fertilizer high in phosphorous, such as 10-10-10, at a rate of 4 pounds
per 1,000 square feet (60 to 90 feet of row)'
    elif my_prediction == 14:
        Answer = 'Predict'
        crop = 'muskmelon'
        fertilizer = 'Apply FYM 20 t/ha, NPK 40:60:30 kg/ha as basal and N @ 40 kg/ha 30 days
after sowing.'
    elif my_prediction == 15:
        Answer = 'Predict'
        crop = 'apple'
        fertilizer = 'Apple trees require nitrogen, phosphorus and potassium,Common granular
20-10-10 fertilizer is suitable for apples'
    elif my_prediction == 16:
        Answer = 'Predict'
        crop = 'orange'
        fertilizer = 'Orange farmers often provide 5,5 – 7,7 lbs (2,5-3,5 kg) P2O5 in every adult
tree for 4-5 consecutive years'
    elif my_prediction == 17:
        Answer = 'Predict'
        crop = 'papaya'
        fertilizer = 'Generally 90 g of Urea, 250 g of Super phosphate and 140 g of Muriate of
Potash per plant are recommended for each application'
    elif my_prediction == 18:
        Answer = 'Predict'
        crop = 'coconut'
        fertilizer = 'Organic Manure @50kg/palm or 30 kg green manure, 500 g N, 320 g P2O5
and 1200 g K2O/palm/year in two split doses during September and May'

    elif my_prediction == 19:

        Answer = 'Predict'

        crop = 'cotton'

        fertilizer = 'N-P-K 20-10-10 per hectare during sowing (through the sowing machine)'

    elif my_prediction == 20:

        Answer = 'Predict'

        crop = 'jute'

        fertilizer = 'Apply 10 kg of N at 20 - 25 days after first weeding and then again on 35 - 40
days after second weeding as top dressing'

    elif my_prediction == 21:

        Answer = 'Predict'

        crop = 'coffee'

        fertilizer = 'Coffee trees need a lot of potash, nitrogen, and a little phosphoric acid. Spread
the fertilizer in a ring around each Coffee plant'

    else:

        Answer = 'Predict'

        crop='Crop info not Found!'

    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1croprecomdb')

    cursor = conn.cursor()

    cursor.execute(

        "update Querytb set DResult='"+Answer+"', CropInfo='" + crop
+"',Fertilizer='"+fertilizer+"' where id='" + str(id) + "' ")

    conn.commit()
    conn.close()
    conn3 = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
    cur3 = conn3.cursor()
    cur3.execute("SELECT * FROM regtb where    UserName='" + str(UserName) + "'")
    data3 = cur3.fetchone()
    if data3:
        phnumber = data3[4]
        print(phnumber)
        sendmsg(phnumber, "Predict Crop Name : "+ crop +" For More info Visit in Site")
    # return 'file register successfully'
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
    # cursor = conn.cursor()
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where  DResult !='waiting '")
    data = cur.fetchall()
    return render_template('AdminAnswer.html', data=data)
@app.route("/AdminAinfo")
def AdminAinfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1croprecomdb')
    # cursor = conn.cursor()
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where  DResult !='waiting'")
    data = cur.fetchall()

    return render_template('AdminAnswer.html', data=data )

@app.route("/Soli")

def Soli():

    return render_template('Soli.html' )

def sendmsg(targetno,message):

    import requests

requests.post("http://smsserver9.creativepoint.in/api.php?username=fantasy&password=5966
92&to=" + targetno + "&from=FSSMSS&message=Dear user  your msg is " + message + "
SentByFSMSG
FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640")

@app.route("/testimage", methods=['GET', 'POST'])

def testimage():

    if request.method == 'POST':

        file = request.files['fileupload']

        file.save('static/Out/Test.jpg')

        img = cv2.imread('static/Out/Test.jpg')

        if img is None:

            print('no data')

        img1 = cv2.imread('static/Out/Test.jpg')

        print(img.shape)

        img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))

        original = img.copy()

        neworiginal = img.copy()

        cv2.imshow('original', img1)
```

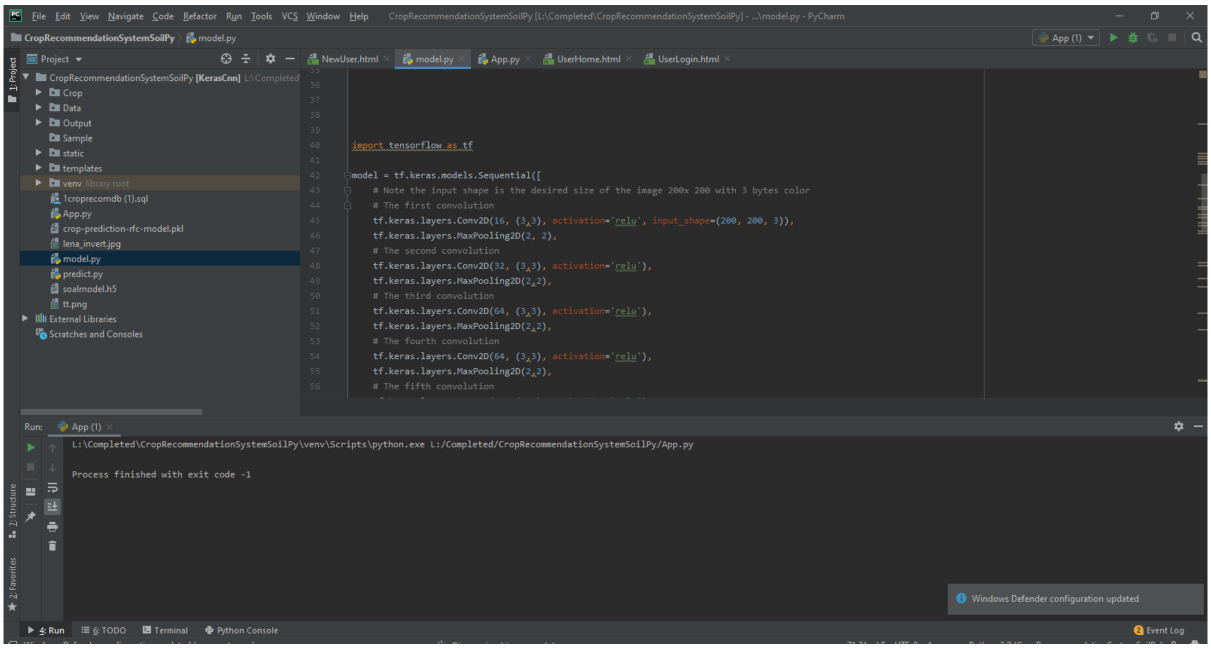
```
gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

img1S = cv2.resize(img1, (960, 540))

cv2.imshow('Original image', img1S)

grayS = cv2.resize(gray, (960, 540))

cv2.imshow('Gray image', grayS)

gry = 'static/Out/gry.jpg'

cv2.imwrite(gry, grayS)

from PIL import  ImageOps,Image

im = Image.open(file)

im_invert = ImageOps.invert(im)

inv = 'static/Out/inv.jpg'

im_invert.save(inv, quality=95)

dst = cv2.fastNlMeansDenoisingColored(img1, None, 10, 10, 7, 21)

cv2.imshow("Nosie Removal", dst)

noi = 'static/Out/noi.jpg'

cv2.imwrite(noi, dst)

import warnings

warnings.filterwarnings('ignore')

import tensorflow as tf

classifierLoad = tf.keras.models.load_model('soalmodel.h5')

import numpy as np

from keras.preprocessing import image

test_image = image.load_img('static/Out/Test.jpg', target_size=(200, 200))

img1 = cv2.imread('static/Out/Test.jpg')
```

```
        # test_image = image.img_to_array(test_image)
        test_image = np.expand_dims(test_image, axis=0)
        result = classifierLoad.predict(test_image)
        out = ''
        pre = ''
        if result[0][0] == 1:
            out = "Alluvial soil"
            pre ="Wheat, Groundnut and cotton"
        elif result[0][1] == 1:
            out = "Black Soil "
            pre = "Cabbage (Napa and savoy), Cauliflower, Kale, Bean, Pea, Potato and Daikon
radish"
        elif result[0][2] == 1:
            out = "Clay soil"
            pre = "Tea, coffee and cashew"
        elif result[0][3] == 1:
            out = "Red soil "
            pre = "Marsh soils are not suitable for crop cultivation due to their high acidic nature"
        org = 'static/Out/Test.jpg'
        gry ='static/Out/gry.jpg'
        inv = 'static/Out/inv.jpg'
        noi = 'static/Out/noi.jpg'
        return render_template('Soli.html',result=out,org=org,gry=gry,inv=inv,noi=noi,pre=pre)
if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```

```
# Part 1 - Building the CNN

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.models import model_from_json

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

batch_size = 64

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255

train_datagen = ImageDataGenerator(rescale=1/255)

# Flow training images in batches of 128 using train_datagen generator

train_generator = train_datagen.flow_from_directory(

        'Data',  # This is the source directory for training images

        target_size=(200, 200),  # All images will be resized to 200 x 200

        batch_size=batch_size,

        # Specify the classes explicitly

        classes = ['Alluvial soil',

        'Black Soil',

        'Clay soil',
        'Red soil'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')
import tensorflow as tf
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),
    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(128, activation='relu'),
    # 5 output neurons for 4 classes with the softmax activation
    tf.keras.layers.Dense(4, activation='softmax')])
model.summary()
from tensorflow.keras.optimizers import RMSprop
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['accuracy'])
total_sample=train_generator.n
n_epochs = 10
from sklearn.metrics import classification_report
history = model.fit_generator(
        train_generator,
        steps_per_epoch=int(total_sample/batch_size),
        epochs=n_epochs,
        verbose=1)
model.save('soalmodel.h5')
acc = history.history['accuracy']
loss = history.history['loss']
epochs = range(1, len(acc) + 1)
# Train and validation accuracy
plt.plot(epochs, acc, 'b', label=' accurarcy')
plt.title('accurarcy')
plt.legend()

plt.figure()

# Train and validation loss

plt.plot(epochs, loss, 'b', label=' loss')

plt.title('  loss')

plt.legend()

plt.show()
```

SCREENSHOTS

EXECUTION PAGE



```
import tensorflow as tf

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
```

```
L:\Completed\CropRecommendationSystemSoilPy\venv\Scripts\python.exe L:/Completed/CropRecommendationSystemSoilPy/App.py

Process finished with exit code -1
```

Windows Defender configuration updated

DATASET UPLOAD

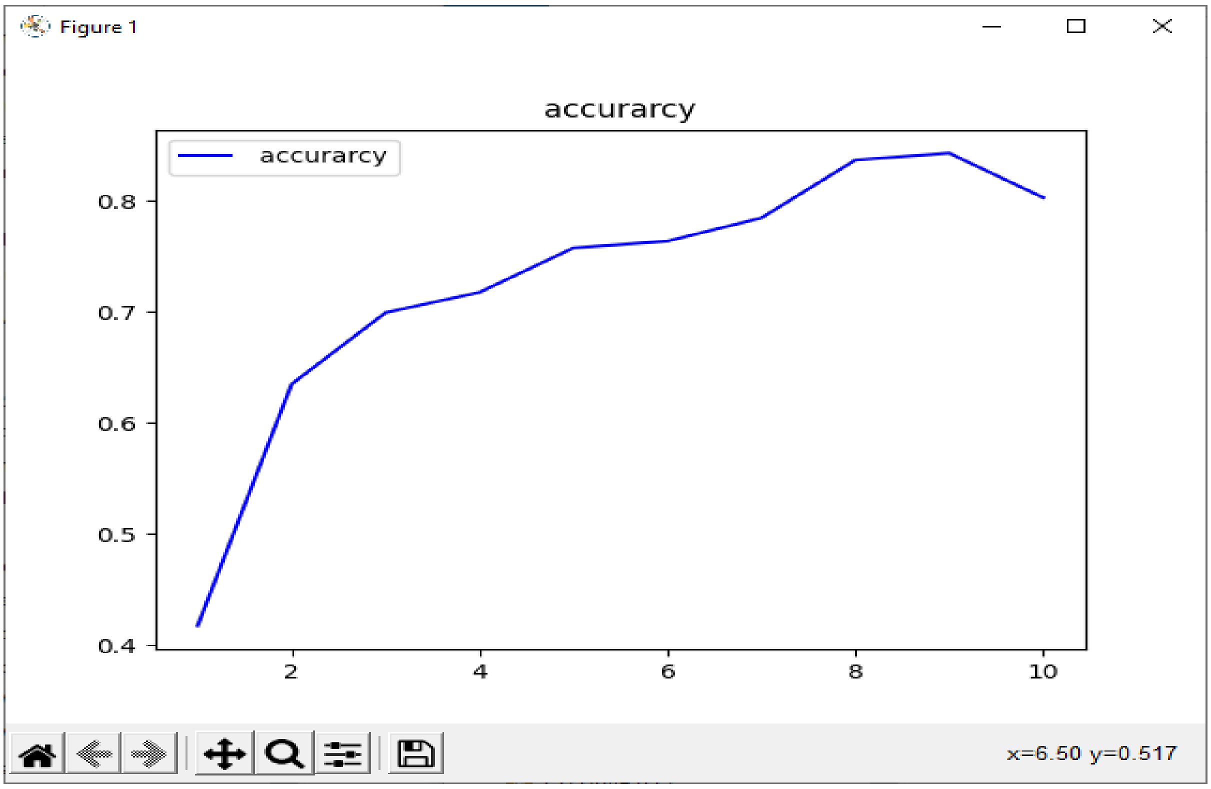
```
L:\Completed\CropRecommendationSystemSoilPy\venv\Scripts\python.exe
L:/Completed/CropRecommendationSystemSoilPy/model.py

2023-03-26 16:43:22.109134: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic
library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found

2023-03-26 16:43:22.109563: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore
above cudart dlerror if you do not have a GPU set up on your machine.

Found 1214 images belonging to 4 classes.

2023-03-26 16:43:44.150828: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic
library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2023-03-26 16:43:44.151541: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed
call to cuInit: UNKNOWN ERROR (303)

2023-03-26 16:43:44.172910: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169]
retrieving CUDA diagnostic information for host: DESKTOP-9BF8NUN
```

```
)

2023-03-26 16:43:44.173677: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176]
hostname: DESKTOP-9BF8NUN

2023-03-26 16:43:44.233544: I tensorflow/core/platform/cpu_feature_guard.cc:151] This
TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use
the following CPU instructions in performance-critical operations:  AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

_________________________________________________________________

 Layer (type)                Output Shape              Param #

=================================================================

 conv2d (Conv2D)             (None, 198, 198, 16)      448


 max_pooling2d (MaxPooling2D  (None, 99, 99, 16)       0

 )


 conv2d_1 (Conv2D)           (None, 97, 97, 32)        4640


 max_pooling2d_1 (MaxPooling  (None, 48, 48, 32)       0

 2D)


 conv2d_2 (Conv2D)           (None, 46, 46, 64)        18496


 max_pooling2d_2 (MaxPooling  (None, 23, 23, 64)       0

 2D)
 conv2d_3 (Conv2D)          (None, 21, 21, 64)        36928


 max_pooling2d_3 (MaxPooling  (None, 10, 10, 64)       0
 2D)


 conv2d_4 (Conv2D)          (None, 8, 8, 64)          36928


 max_pooling2d_4 (MaxPooling  (None, 4, 4, 64)         0
 2D)


 flatten (Flatten)          (None, 1024)              0


 dense (Dense)              (None, 128)               131200


 dense_1 (Dense)            (None, 4)                 516


=================================================================
Total params: 229,156
Trainable params: 229,156
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
WARNING:tensorflow:AutoGraph could not transform <function
```

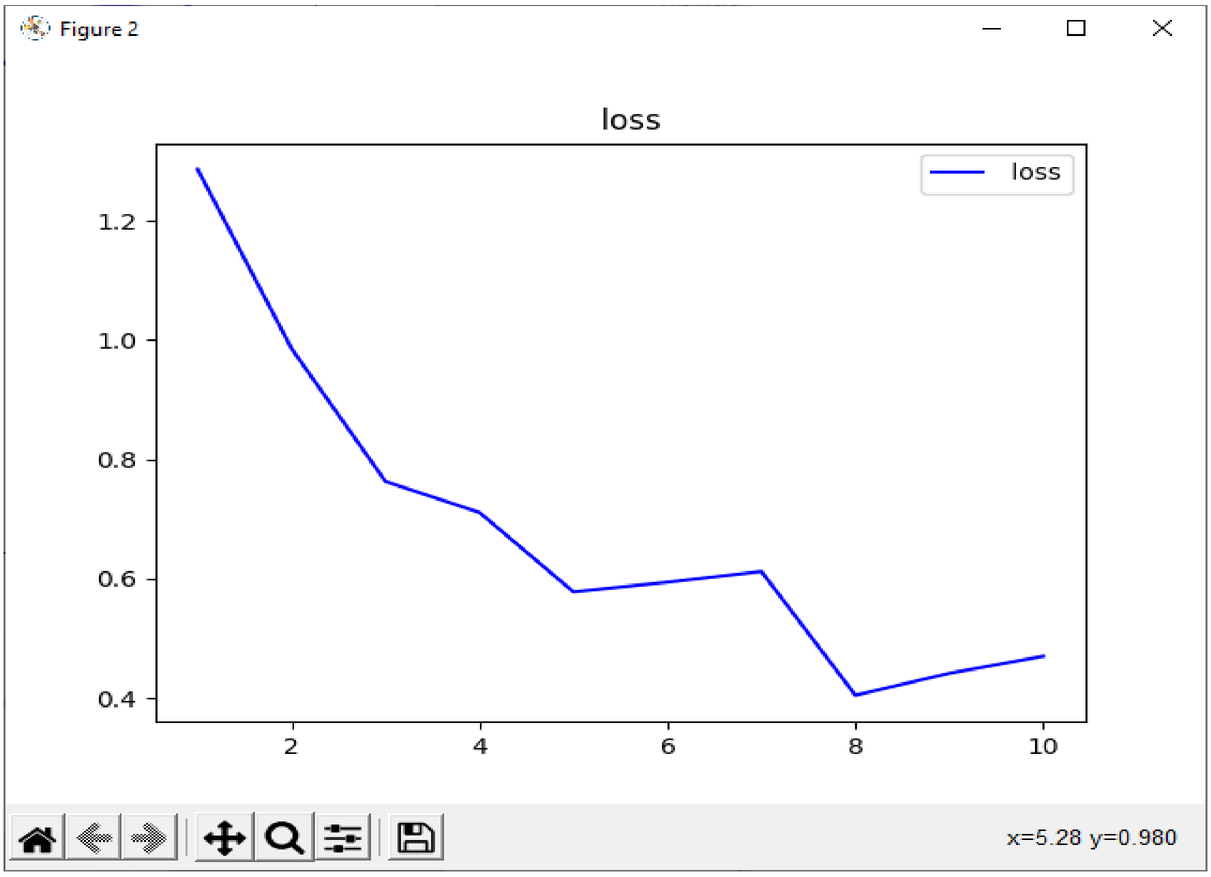
```
Model.make_train_function.<locals>.train_function at 0x00000255CD4CCE58> and will run it as-is.

Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output.

Cause: 'arguments' object has no attribute 'posonlyargs'

To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert

2023-03-26 16:43:52.317177: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 160579584 exceeds 10% of free system memory.

2023-03-26 16:43:53.050479: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 40144896 exceeds 10% of free system memory.

2023-03-26 16:43:53.142223: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 77078528 exceeds 10% of free system memory.

2023-03-26 16:43:53.266618: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 34668544 exceeds 10% of free system memory.

2023-03-26 16:43:54.498253: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 34668544 exceeds 10% of free system memory.

18/18 [==============================] - 27s 1s/step - loss: 1.2857 - accuracy: 0.4174

Epoch 2/10

18/18 [==============================] - 22s 1s/step - loss: 0.9852 - accuracy: 0.6348

Epoch 3/10

18/18 [==============================] - 22s 1s/step - loss: 0.7625 - accuracy: 0.6991

Epoch 4/10
```

```
18/18 [==============================] - 22s 1s/step - loss: 0.7107 - accuracy:
0.7174

Epoch 5/10

18/18 [==============================] - 22s 1s/step - loss: 0.5775 - accuracy:
0.7574

Epoch 6/10

18/18 [==============================] - 21s 1s/step - loss: 0.5938 - accuracy:
0.7635

Epoch 7/10

18/18 [==============================] - 21s 1s/step - loss: 0.6117 - accuracy:
0.7843

Epoch 8/10

18/18 [==============================] - 22s 1s/step - loss: 0.4042 - accuracy:
0.8365

Epoch 9/10

18/18 [==============================] - 22s 1s/step - loss: 0.4410 - accuracy:
0.8426

Epoch 10/10

18/18 [==============================] - 23s 1s/step - loss: 0.4696 - accuracy:
0.8026
```

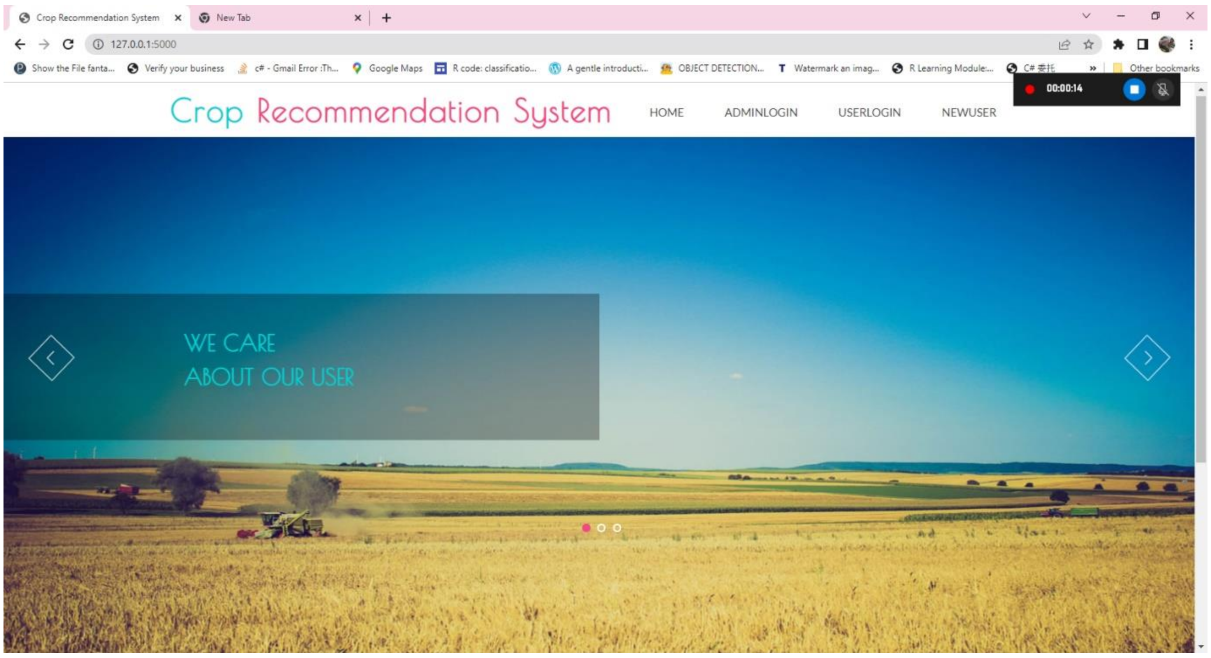
TRAINING ACCURACY



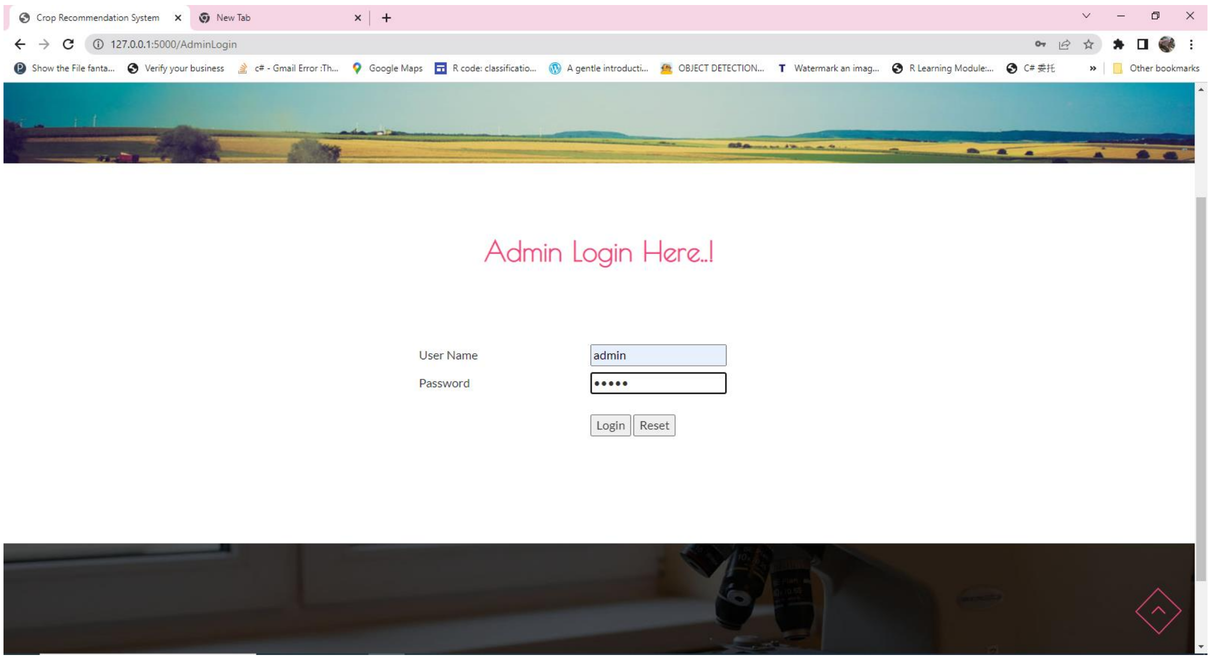
TRAINING LOSS



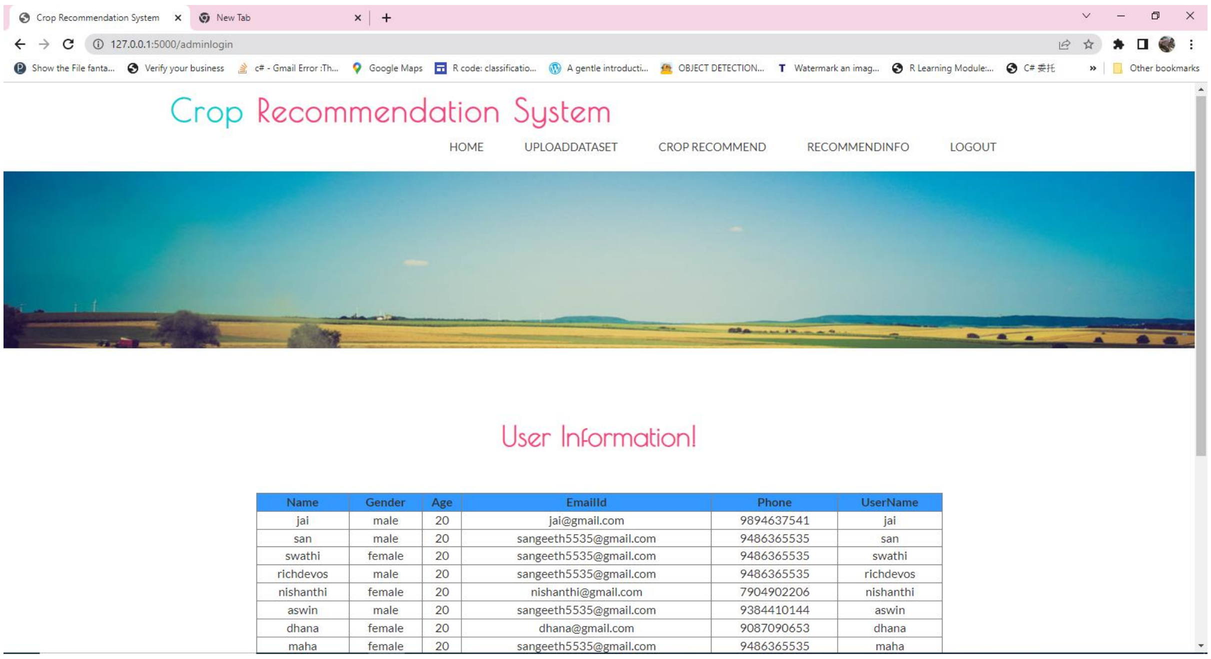
HOME PAGE



ADMIN LOGIN



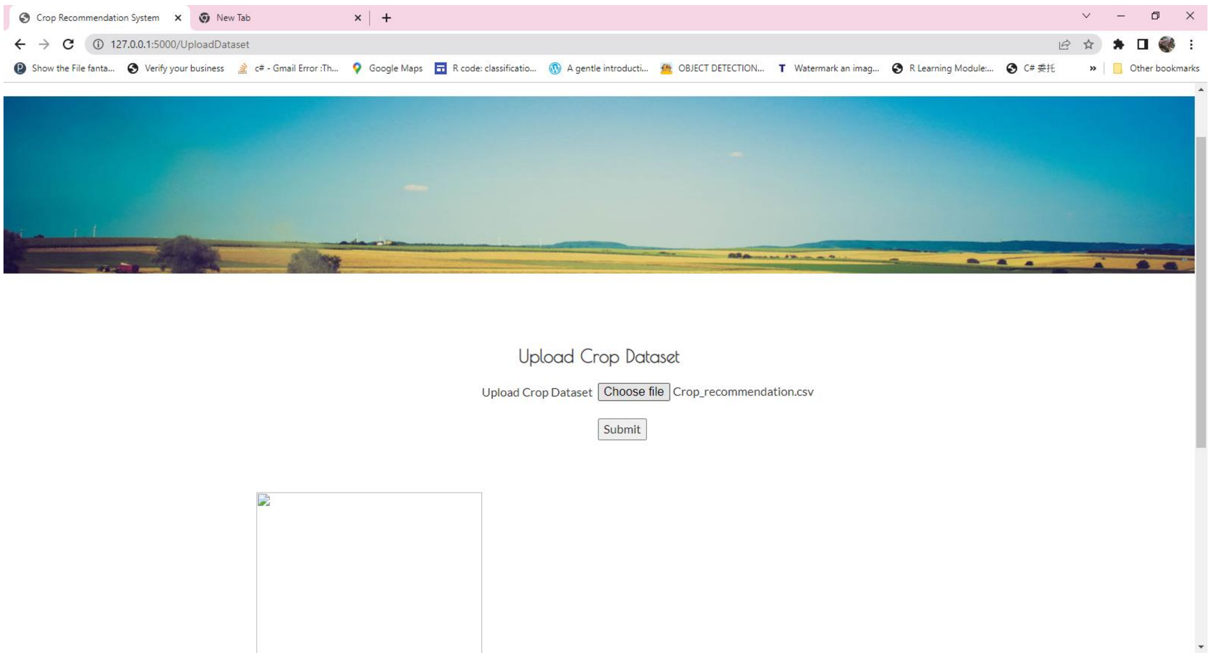
USER INFORMATION


Crop Recommendation System
New Tab
127.0.0.1:5000/adminlogin
Crop Recommendation System
HOME
UPLOADDATASET
CROP RECOMMEND
RECOMMENDINFO
LOGOUT

User Information!

Name	Gender	Age	EmailId	Phone	UserName
jai	male	20	jai@gmail.com	9894637541	jai
san	male	20	sangeeth5535@gmail.com	9486365535	san
swathi	female	20	sangeeth5535@gmail.com	9486365535	swathi
richdevos	male	20	sangeeth5535@gmail.com	9486365535	richdevos
nishanthi	female	20	nishanthi@gmail.com	7904902206	nishanthi
aswin	male	20	sangeeth5535@gmail.com	9384410144	aswin
dhana	female	20	dhana@gmail.com	9087090653	dhana
maha	female	20	sangeeth5535@gmail.com	9486365535	maha

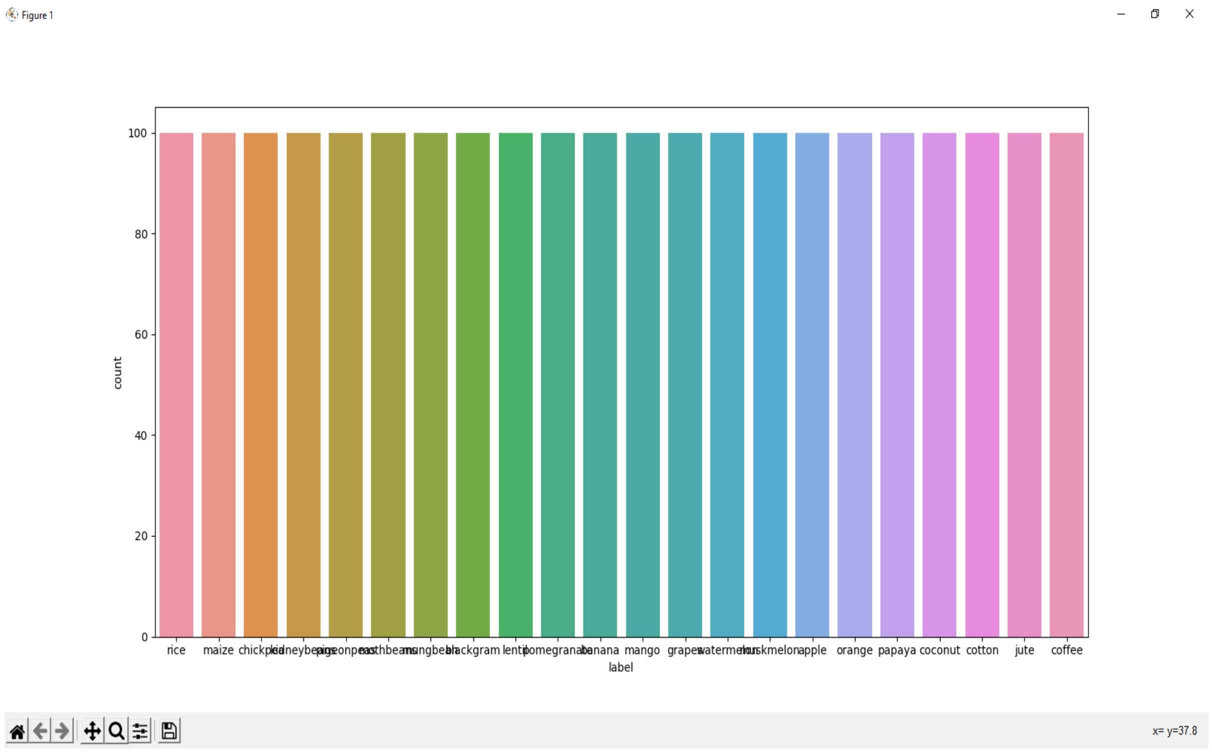
UPLOAD DATASETS



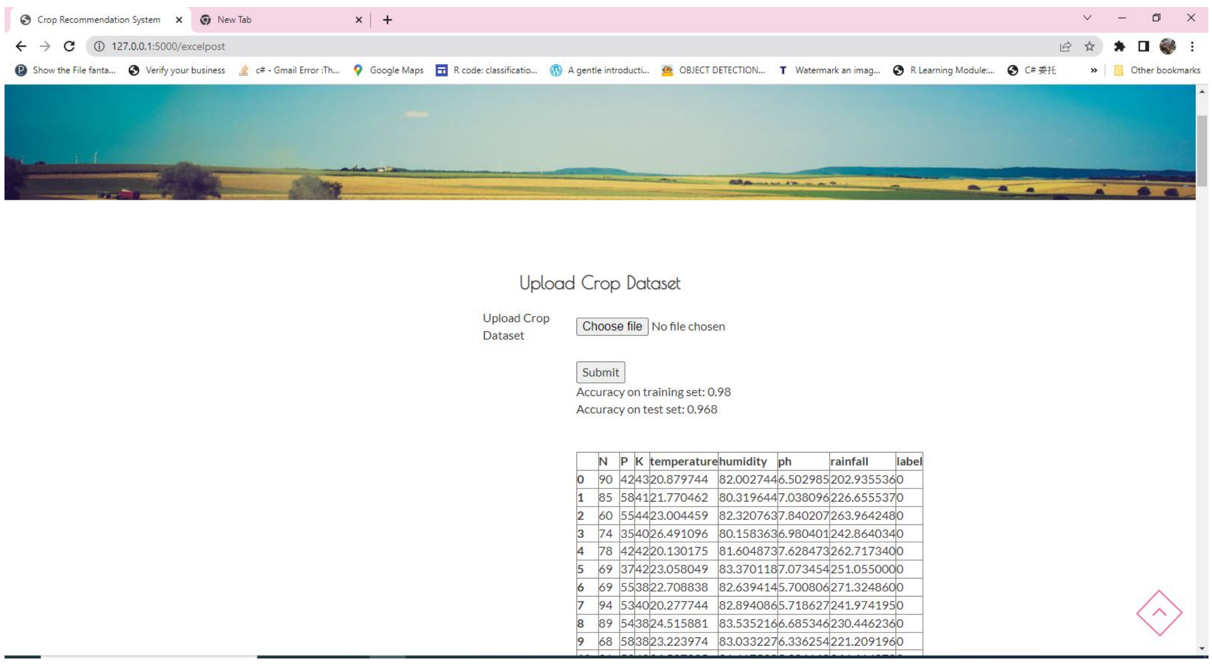
Upload Crop Dataset

Upload Crop Dataset Crop_recommendation.csv

COUNTERPLOT



VIEW THE DATASETS



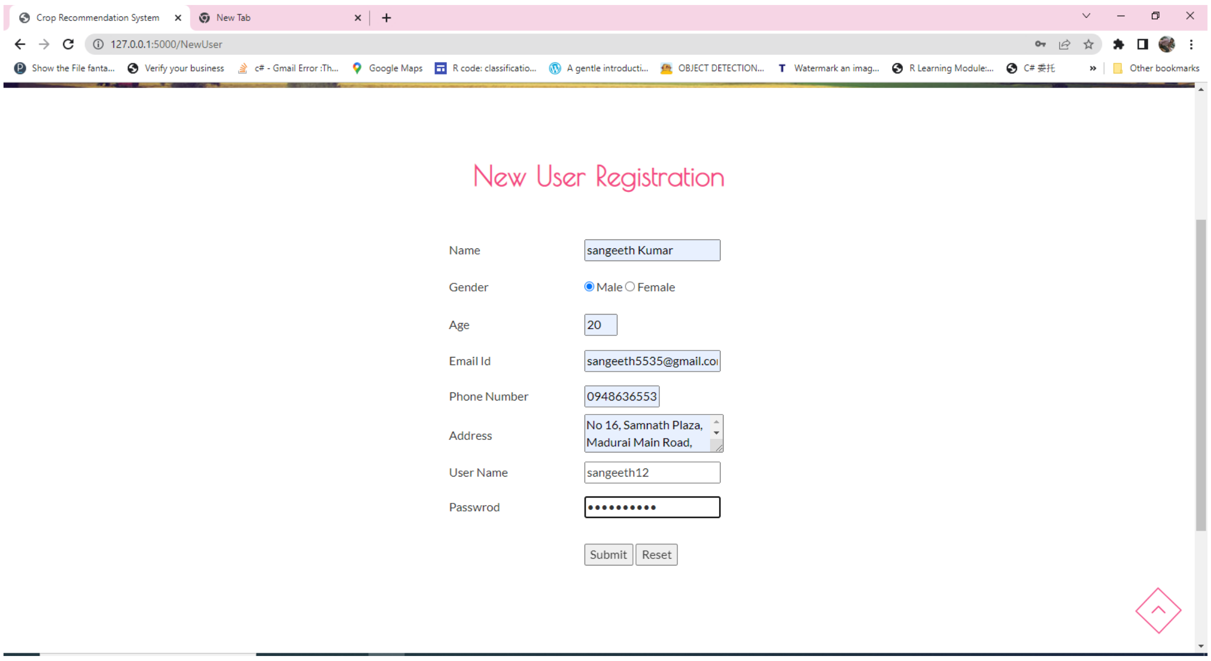
Upload Crop Dataset

Upload Crop Dataset

Accuracy on training set: 0.98
Accuracy on test set: 0.968

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	0
1	85	58	41	21.770462	80.319644	7.038096	226.655537	0
2	60	55	44	23.004459	82.320763	7.840207	263.964248	0
3	74	35	40	26.491096	80.158363	6.980401	242.864034	0
4	78	42	42	20.130175	81.604873	7.628473	262.717340	0
5	69	37	42	23.058049	83.370118	7.073454	251.055000	0
6	69	55	38	22.708838	82.639414	5.700806	271.324860	0
7	94	53	40	20.277744	82.894086	5.718627	241.974195	0
8	89	54	38	24.515881	83.535216	6.685346	230.446236	0
9	68	58	38	23.223974	83.033227	6.336254	221.209196	0

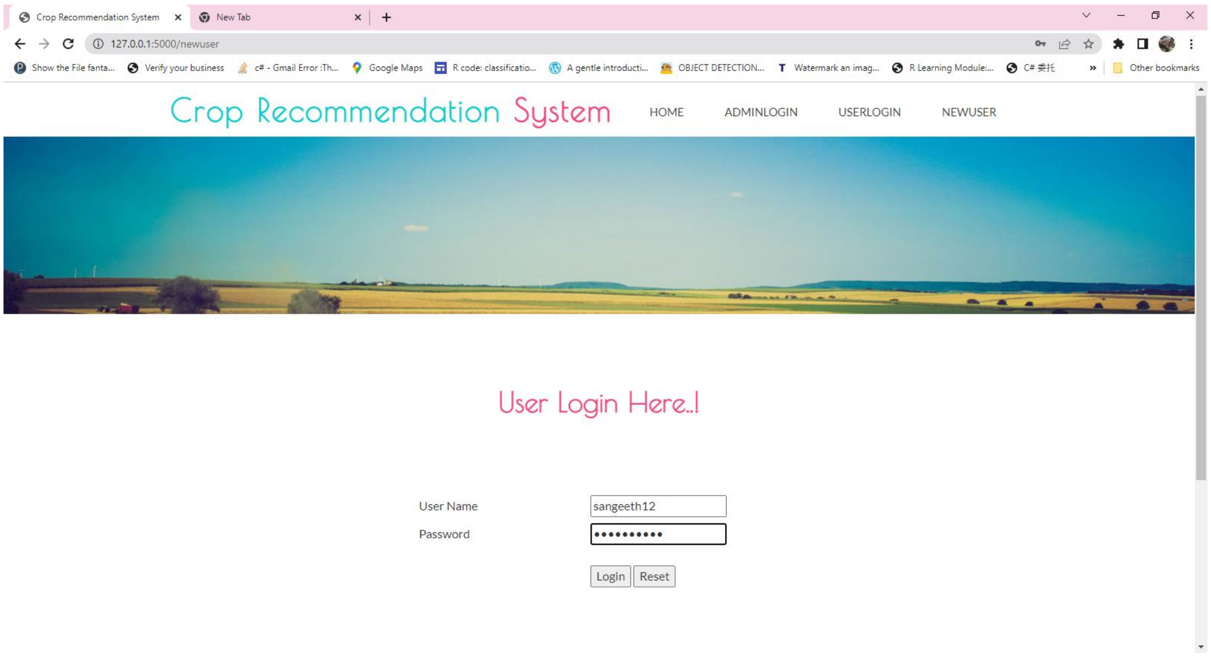
NEW USER REGISTRATION



New User Registration

Name	sangeeth Kumar
Gender	[✓ Male] Female
Age	20
Email Id	sangeeth5535@gmail.co
Phone Number	0948636553
Address	No 16, Samnath Plaza, Madurai Main Road,
User Name	sangeeth12
Passwrod	••••••••••

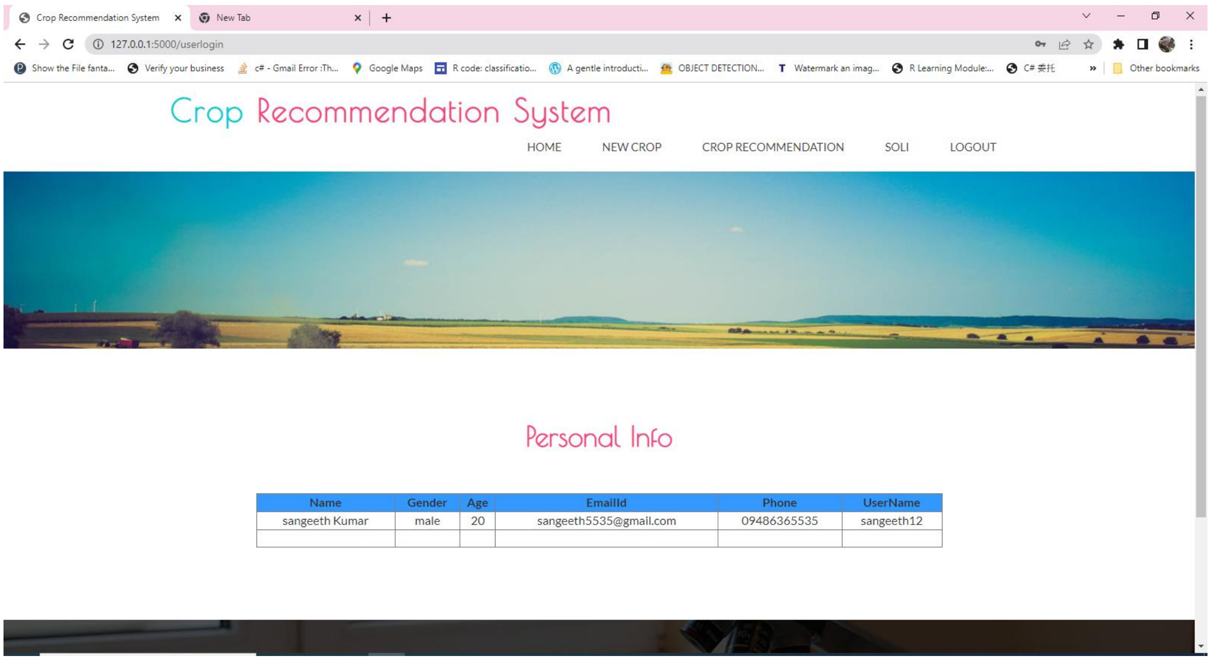
USER LOGIN



User Login Here..!

User Name	sangeeth12
Password	••••••••••

PERSONAL INFORMATION

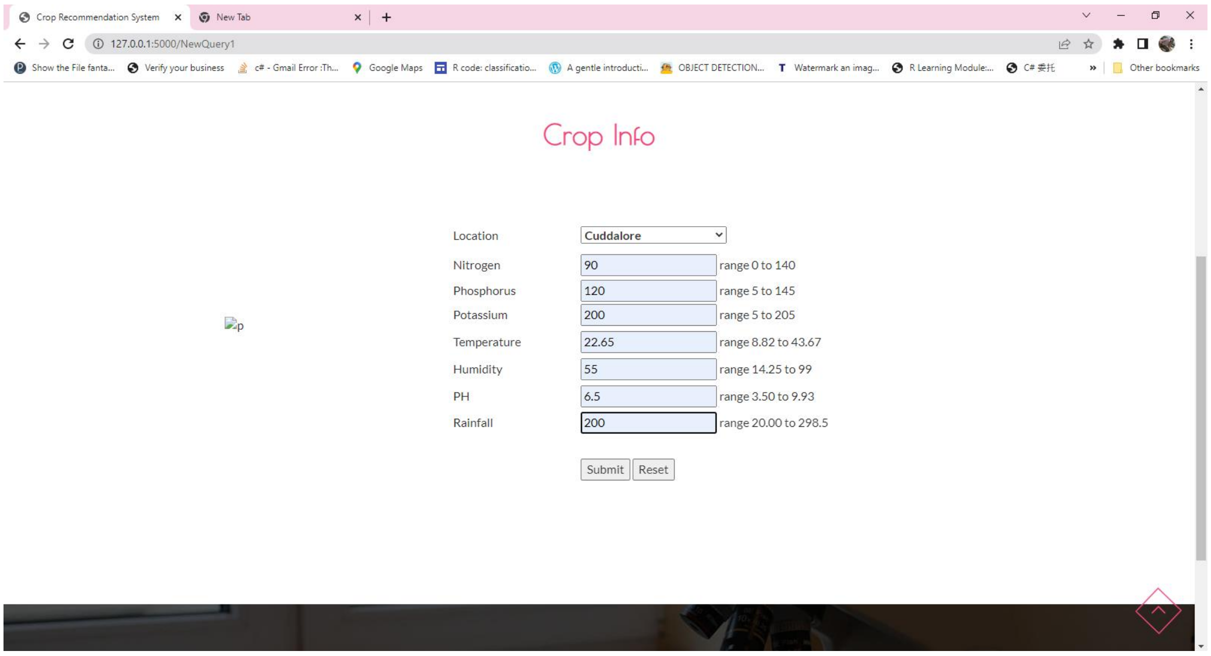


Crop Recommendation System

Personal Info

Name	Gender	Age	EmailId	Phone	UserName
sangeeth Kumar	male	20	sangeeth5535@gmail.com	09486365535	sangeeth12

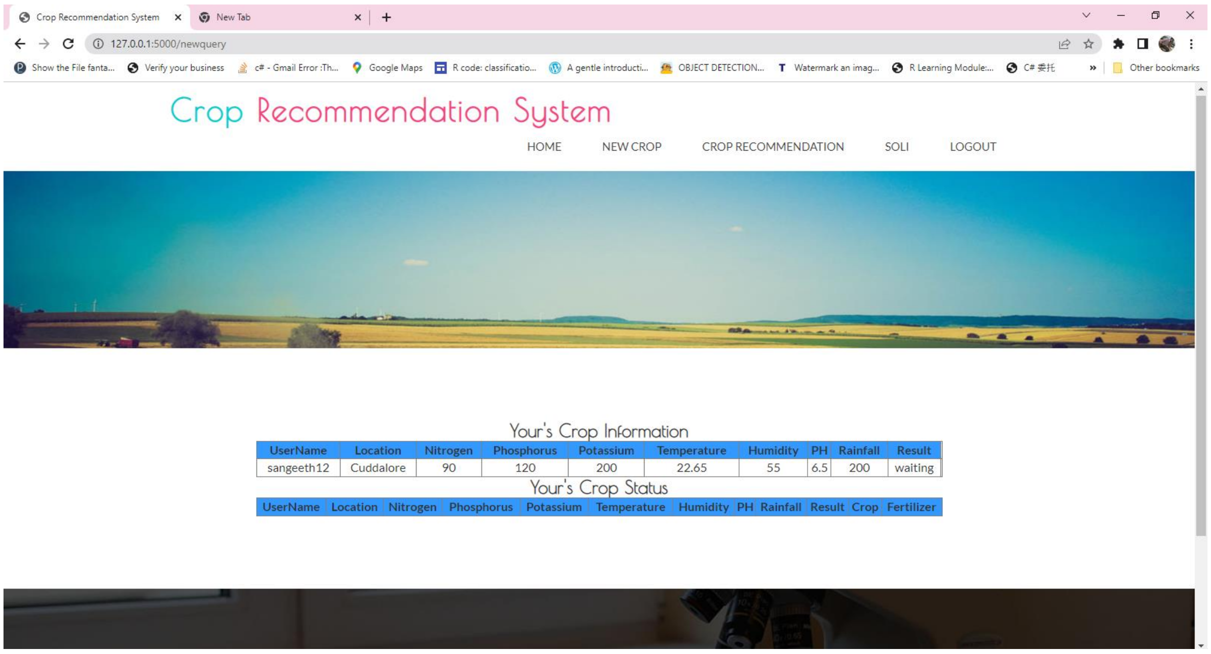
CROP INFORMATION



Crop Info

Location	Cuddalore	
Nitrogen	90	range 0 to 140
Phosphorus	120	range 5 to 145
Potassium	200	range 5 to 205
Temperature	22.65	range 8.82 to 43.67
Humidity	55	range 14.25 to 99
PH	6.5	range 3.50 to 9.93
Rainfall	200	range 20.00 to 298.5

PREDICTION RESULT



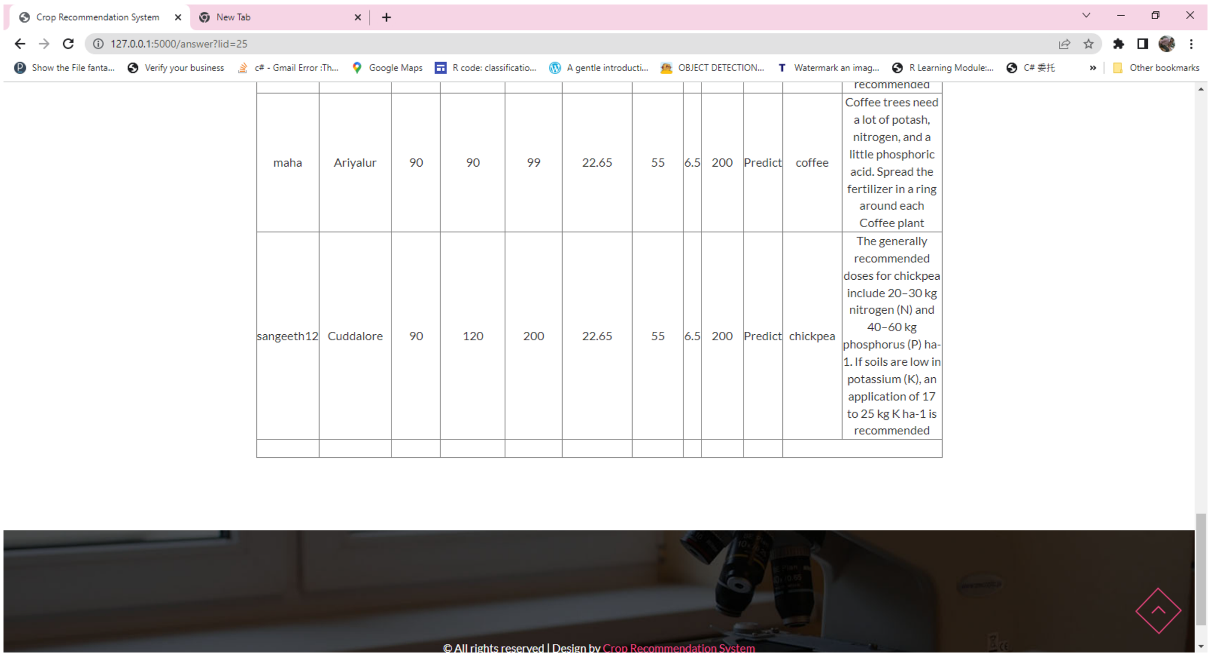
Your's Crop Information

UserName	Location	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	PH	Rainfall	Result
sangeeth12	Cuddalore	90	120	200	22.65	55	6.5	200	waiting

Your's Crop Status

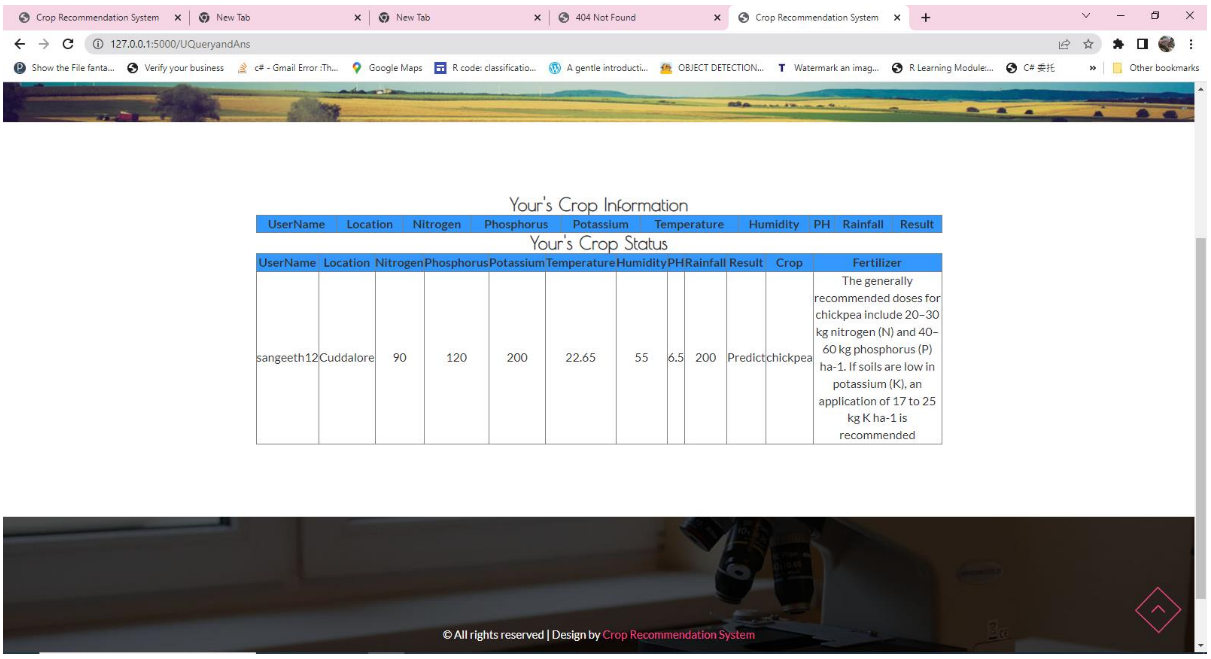
UserName	Location	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	PH	Rainfall	Result	Crop	Fertilizer

RESULT (CROP WITH FERTILIZER)



											recommended
maha	Ariyalur	90	90	99	22.65	55	6.5	200	Predict	coffee	Coffee trees need a lot of potash, nitrogen, and a little phosphoric acid. Spread the fertilizer in a ring around each Coffee plant
sangeeth12	Cuddalore	90	120	200	22.65	55	6.5	200	Predict	chickpea	The generally recommended doses for chickpea include 20–30 kg nitrogen (N) and 40–60 kg phosphorus (P) ha-1. If soils are low in potassium (K), an application of 17 to 25 kg K ha-1 is recommended

© All rights reserved | Design by Crop Recommendation System


127.0.0.1:5000/UQueryandAns

Your's Crop Information

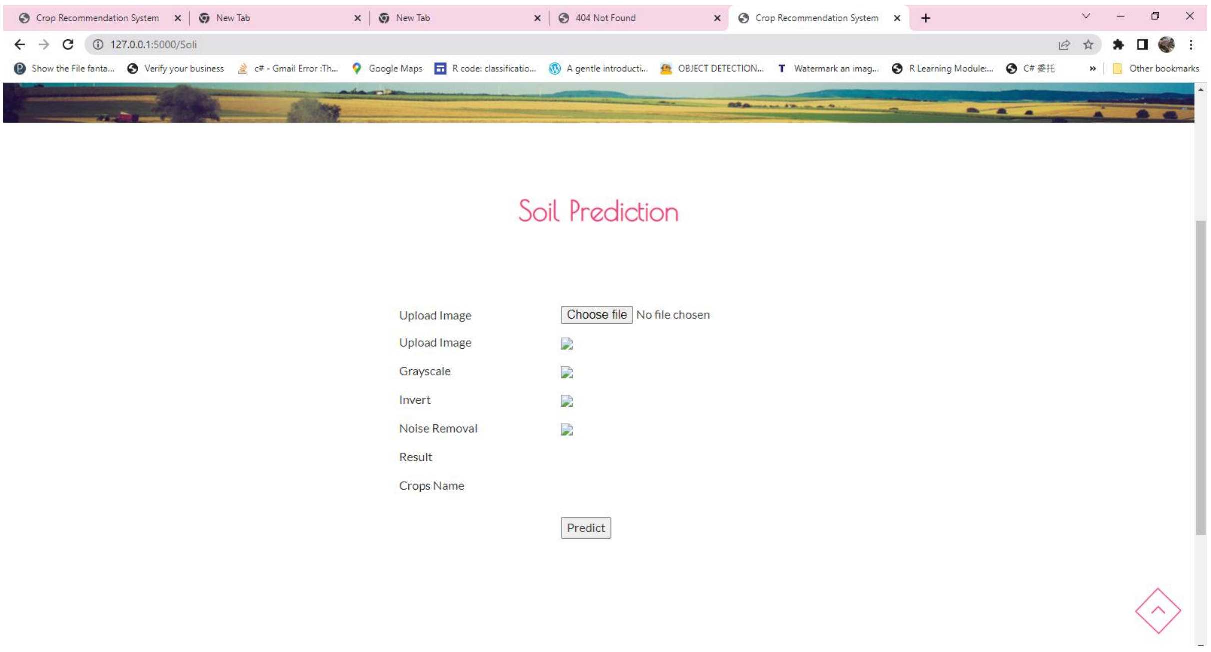
UserName	Location	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	PH	Rainfall	Result

Your's Crop Status

UserName	Location	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	PH	Rainfall	Result	Crop	Fertilizer
sangeeth12	Cuddalore	90	120	200	22.65	55	6.5	200	Predict	chickpea	The generally recommended doses for chickpea include 20–30 kg nitrogen (N) and 40–60 kg phosphorus (P) ha-1. If soils are low in potassium (K), an application of 17 to 25 kg K ha-1 is recommended

© All rights reserved | Design by Crop Recommendation System

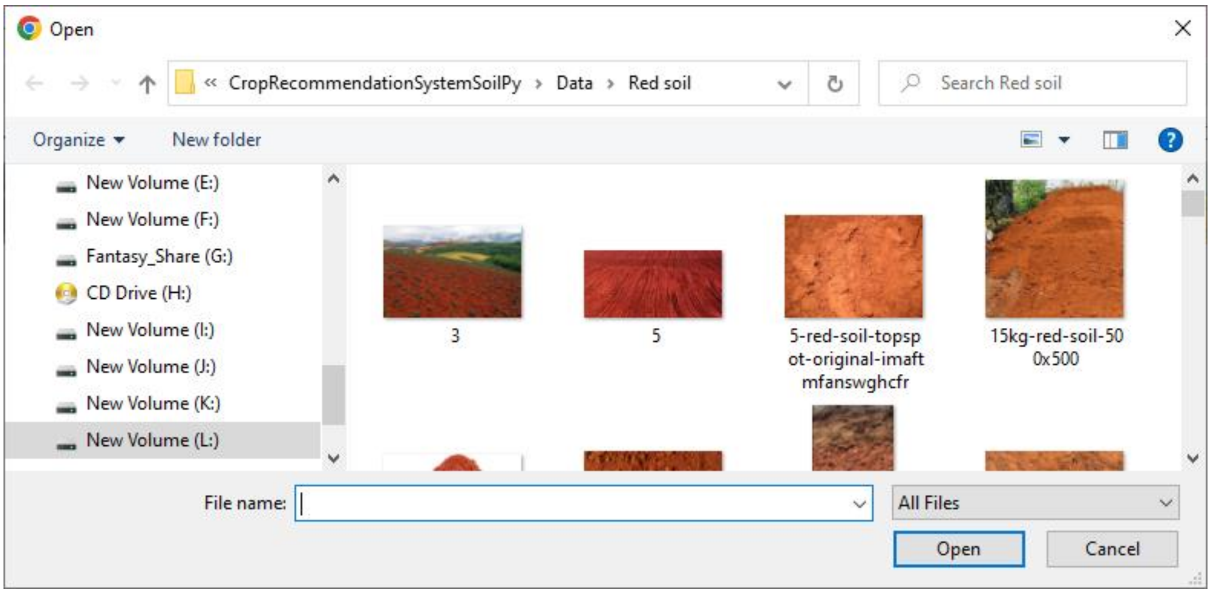
SOIL CLASSIFICATION


127.0.0.1:5000/Soli

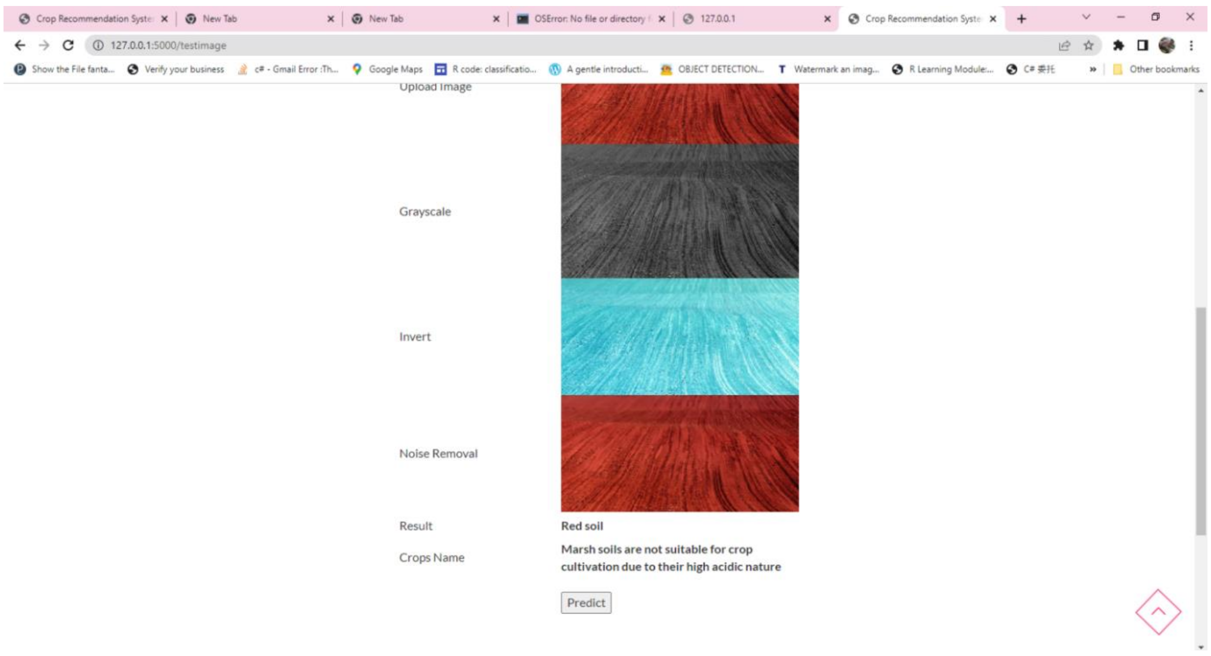
Soil Prediction

Upload Image	Choose file No file chosen
Upload Image	
Grayscale	
Invert	
Noise Removal	
Result	
Crops Name	

UPLOAD SOIL IMAGE



CLASSIFICATION RESULT



Result Red soil

Crops Name Marsh soils are not suitable for crop cultivation due to their high acidic nature

CHAPTER 9

9.CONCLUSION

9.1 CONCLUSION

We presented a deep learning approach for crop prediction, which demonstrated superior performance in Crop Challenge using large datasets of products. The approach used deep neural networks to make crop datasets such as soil and textual datasets. In conclusion, deep learning models offer a promising solution for predicting crop yields based on environmental variables such as temperature, pH, rainfall, and soil data. By using neural networks to analyse large and complex datasets, these models can identify patterns and relationships that would be difficult or impossible for humans to discern. By training the model on historical data and then using it to make predictions on new data, farmers and researchers can gain valuable insights into which crops are most likely to thrive under certain environmental conditions. However, there are still some challenges to overcome, such as the need for high-quality and diverse data, the difficulty of interpreting complex neural networks, and the potential for bias and errors in the training data. Overall, deep learning holds great promise for revolutionizing the field of crop prediction and helping to feed a growing global population

9.2 FUTURE ENHANCEMENT

This project describes crop yield prediction ability of the algorithm. In future we can determine the efficient algorithm based on their accuracy metrics that will helps to choose an efficient algorithm for crop prediction based on soil images.

REFERENCES

1 Paudel, Dilli, et al.: Machine learning for large-scalem crop yield forecasting. Agricultural Systems 187:103016. (2021)

2 Abbas, Farhat, et al.: Crop yield prediction through proximal sensing and machine learning algorithms.Agronomy 10.7:1046. (2020)

3 Anantha, K. H., et al.: Impact of best management practices on sustainable crop production and climate resilience in smallholder farming systems of South Asia.Agricultural Systems 194: 103276. (2021)

4 Palanivel, Kodimalar, and Chellammal Surianarayanan.: An approach for prediction of crop yield using machine learning and big data techniques.International Journal of Computer Engineering and Technology 10.3: 110-118. (2019)

5 Bian, Chaofa, et al.: Prediction of Field-Scale Wheat Yield Using Machine Learning Method and Multi-Spectral UAV Data.Remote Sensing 14.6 (2022): 1474. (2019)

6 Nishant, Potnuru Sai, et al.: Crop yield prediction based on indian agriculture using machine learning. 2020 International Conference for Emerging Technology (INCET),IEEE. (2020)

7 Mishra, Subhadra, Debahuti Mishra, and Gour Hari Santra.: Adaptive boosting of weak regressors for forecasting of crop production considering climatic variability: An empirical assessment.Journal of King Saud University-Computer and Information Sciences 32.8: 949-964. (2020)

8 Talaviya, Tanha, et al.: Implementation of artificial intelligence in agriculture for optimisation of irrigation and application of pesticides and herbicides.Artificial Intelligence in Agriculture 4: 58-73. (2020)

9 Van Klompenburg, Thomas, Ayalew Kassahun, and Cagatay Catal.: Crop yield prediction using machine learning: A systematic literature review.Computers and Electronics in Agriculture 177: 105709. (2020)

10 Pandith, Vaishali, et al.: Performance evaluation of machine learning techniques for mustard crop yield prediction from soil analysis.Journal of Scientific Research 64.2: 394-398. (2020)