A Project Report on

Intelligent Solution for Crops Using Crop Recommendation, Fertilizer Suggestion and Plant Disease Detection

submitted in partial fulfillment for the award of

Bachelor of Technology

in

Computer Science and Engineering

by

A. Santhosh Kumar (Y20ACS403) **G. Harshitha (Y20ACS445)**

B. Bhavana Sai (Y20ACS419) **B. Ajay Kumar (Y20ACS414)**

Under the guidance of
Mr. T. Y. Srinivasa Rao, Assistant Professor

Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
BAPATLA – 522 102, Andhra Pradesh, INDIA
2023-2024

Department of
Computer Science and Engineering

<u>CERTIFICATE</u>

This is to certify that the project report entitled **Intelligent Solution for Crop Recommendation, Fertilizer Suggestion and Plant Disease Detection** is being submitted by A. Santhosh Kumar (Y20ACS403), G. Harshitha (Y20ACS445), B. Bhavana Sai (Y20ACS419), B. Ajay Kumar (Y20ACS414) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Acharya Nagarjuna University is a record of bonafede work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
Mr. T. Y. Srinivasa Rao
Assistant Professor

Signature of the HOD
Dr. M. Rajesh Babu
Associate Professor

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

A. Santhosh Kumar (Y20ACS403)
G. Harshitha (Y20ACS445)
B. Bhavana Sai (Y20ACS419)
B. Ajay Kumar (Y20ACS414)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr. T. Y. Srinivasa Rao,** Assistant Professor, Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu,** Associate Prof. & Head of the Dept. for extending his co operation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our thanks to our project coordinator **Dr. N. Sudhakar,** Professor, Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

A. Santhosh Kumar (Y20ACS403)
G. Harshitha (Y20ACS445)
B. Bhavana Sai (Y20ACS419)
B. Ajay Kumar (Y20ACS414)

Abstract

This project report presents an integrated agricultural decision support system aimed at enhancing crop yield and health through advanced data analytics techniques. The system comprises three distinct modules: crop recommendation, fertilizer suggestion, and plant disease detection.

The first module leverages machine learning approaches, specifically decision trees and random forests, to recommend suitable crops based on various environmental factors including N, P, K nutrient levels, humidity, temperature, pH, and rainfall. By analyzing historical data and current environmental conditions, the system assists farmers in making informed decisions regarding crop selection, optimizing yield, and resource utilization.

The second module focuses on suggesting appropriate fertilizer regimes tailored to specific crop requirements. Utilizing the insights gained from the crop recommendation module, this system employs machine learning algorithms to determine the optimal combination of fertilizers based on soil nutrient levels and crop nutrient demands. By optimizing nutrient supply, the system aims to improve soil fertility and promote healthy crop growth.

The third module employs Convolutional Neural Networks (CNNs) for the early detection of plant diseases. By analyzing images of plant leaves captured through sensors or smartphones, the system identifies signs of disease or pest infestation with high accuracy. Early detection allows farmers to implement timely interventions, such as targeted pesticide application or crop rotation, minimizing yield losses and ensuring crop health. Through the integration of these modules, the proposed agricultural decision support system provides a comprehensive solution for farmers to enhance productivity, optimize resource utilization, and mitigate risks associated with crop management. The report discusses the methodology, implementation details, performance evaluation, and future research directions of each module, highlighting the potential for technology-driven advancements in agriculture.

Table of Contents

List of Figures

List of Tables

1 Introduction

India's agriculture, which employs a major portion of the125-crore people, has a significant impact on the country's economy and daily lives of ordinary people. Crops are needed to fill the food need of the country men. Due to a shortage or complete lack of food caused by an increasing population, many countries still face necessary food requirements. Increasing crop production is an effective way to end this daily rise of famine like conditions due to lack of food. United Nations critical objective by 2030 are Food security development and decline of hunger [1].

Crop prediction has become an integral part of this mission. The main purpose of agriculture planning is to find a better crop yield and there should be profit for the farmer [2]. The crop prediction is made using the data of the soil in the particular area in which the prediction is being made (soil nutrients like potassium, nitrogen etc) and Data on different crops i.e, the ample amount of the nutrients need to grow the plants and using machine learning technique we can then predict the crops which are right for that piece of land. Through this we can provide options to the farmer of different crops which will have better chances of profit and better yields.

Plants play an essential role in climate change, agriculture industry and a country's economy. Thereby taking care of plants is very crucial. Just like humans, plants are affected by several disease caused by bacteria, fungi, and virus. Identification of these disease timely and curing them is essential to prevent whole plant from destruction. [Objective] This system proposes a deep learning-based model named plant disease detector. The model can detect several diseases from plants using pictures of their leaves. [Methodology]

Plant disease detection model is developed using neural network. First, augmentation is applied on dataset to increase the sample size.

Later Convolution Neural Network (CNN) is used with multiple convolution and pooling layers. Plant Village dataset is used to train the model. After training the model, it is tested properly to validate the results. [Results] We have performed different experiments using this model. 15% of data from Plant Village data is used for testing purpose that contains images of healthy as well as diseased plants. Proposed model has achieved 98.3% testing accuracy. [Conclusion] This study is focused on deep learning model to detect disease in plant leave. But, in future model can be integrated with drone or any other system to live detect diseases from plants and report the diseased plant's location to people so that they can be cured accordingly.

1. Enhancing Agricultural Productivity:

Agriculture is a vital sector for global food security and economic development. By developing a crop recommendation system, farmers can receive tailored advice on the types of crops to grow based on various factors such as soil type, climate conditions, and historical data. This personalized guidance can lead to optimized crop selection, improved yields, and increased profitability for farmers.

2. Optimizing Resource Utilization:

Efficient use of fertilizers is essential for maximizing crop yields while minimizing environmental impact. The fertilizer suggestion system aims to provide farmers with precise recommendations for fertilizer application based on soil nutrient levels, crop requirements, and environmental factors. By optimizing fertilizer usage, farmers can achieve better nutrient management, reduce input costs, and mitigate the risk of nutrient runoff and soil degradation.

3. Early Disease Detection and Management:

Plant diseases pose significant threats to crop health and yield. Early detection and prompt management are crucial for minimizing crop losses and ensuring food security. The plant disease detection system leverages deep learning techniques to identify and diagnose diseases in plants based on image analysis. By detecting diseases at early stages, farmers can implement timely interventions such as targeted treatments or crop rotation, reducing the spread of diseases and preserving crop yields.

4. Integration of Advanced Technologies:

The project integrates cutting-edge technologies such as machine learning algorithms (Random Forest, Decision Tree, Gaussian Naive Bayes) and deep learning models (Convolutional Neural Network) to address complex agricultural challenges. By harnessing the power of data analytics and artificial intelligence, the project aims to provide innovative solutions that empower farmers with actionable insights and decision support tools.

5. Sustainability and Environmental Conservation:

Sustainable agriculture practices are essential for preserving natural resources and mitigating the impact of climate change. By promoting precision agriculture techniques through crop recommendation and fertilizer suggestion systems, the project contributes to sustainable farming practices. By optimizing resource utilization and reducing chemical inputs, farmers can minimize environmental pollution, conserve soil fertility, and promote long-term agricultural sustainability.

6. Empowering Farmers with Technology:

The project aims to empower farmers with accessible and user-friendly technology solutions that enhance their decision-making capabilities and improve farm management practices. By providing intuitive interfaces and actionable recommendations, the systems enable farmers to make informed choices, adapt to changing environmental conditions, and optimize their agricultural operations for improved productivity and resilience.

By leveraging these recommendations, farmers can make informed decisions regarding crop selection, leading to improved yields and economic returns. Secondly, the fertilizer suggestion system addresses the critical need for efficient soil health and nutrient management. Through personalized fertilizer recommendations derived from soil nutrient levels, crop requirements, and environmental considerations, the system aims to optimize nutrient utilization and reduce environmental impact.

1.1 Statement and Solution to the Problem

The challenge of providing intellectual solutions for crop recommendation, fertilizer suggestion, and plant disease detection can be addressed through advanced technologies like machine learning and sensor networks. By analyzing various factors such as soil conditions, weather data, and plant health indicators, tailored recommendations for crop selection, optimal fertilizer usage, and early disease detection can be provided to farmers. This approach enables precise and sustainable farming practices, ultimately enhancing agricultural productivity and reducing resource wastage.

1.1.1 Problem Statement

In modern agriculture, farmers face numerous challenges related to crop selection, optimal nutrient management, and early detection of plant diseases. The integrated system aims to seamlessly combine crop recommendation, fertilizer suggestions, and plant disease detection capabilities into a single platform.

1.1.2 Solution to the Problem

The proposed system uses computer techniques to assist farmers in managing their crops better. It provides guidance on which crops to grow, which fertilizers to use, and how to identify plant diseases. This platform helps farmers make smarter decisions and enhance their farming methods. By analyzing large amounts of data, it can offer accurate suggestions for crop management, fertilizer suggestion, and disease detection.

1.2 Scope of the Project

The scope of this project encompasses the development, implementation, and evaluation of an integrated agricultural management system comprising three distinct modules. The first module focuses on crop recommendation, employing machine learning methodologies such as decision trees and random forests to analyze environmental variables including soil nutrient levels (N, P, K), humidity, temperature, pH, and rainfall. The scope encompasses the design and optimization of algorithms for crop recommendation, considering diverse agricultural environments and crop varieties. Furthermore, the scope extends to the integration of real-time data acquisition systems to facilitate continuous monitoring and adaptation of crop recommendations based on evolving environmental conditions.

The second module of the system entails the suggestion of fertilizer regimes tailored to meet the specific nutritional requirements of selected crops. Within this scope, the project encompasses the development of machine learning algorithms to optimize fertilizer composition and application rates based on soil nutrient profiles and crop nutrient demands. Additionally, the scope extends to the integration of decision support tools to facilitate farmers in implementing recommended fertilizer regimes effectively. Evaluation of the fertilizer suggestion system will involve field trials and comparative analyses to assess its efficacy in promoting soil health, reducing nutrient runoff, and enhancing crop productivity.

The third module focuses on the early detection of plant diseases using Convolutional Neural Networks (CNNs) for image-based diagnosis. The scope includes the development of CNN models trained on image datasets comprising plant leaves exhibiting various disease symptoms. Implementation involves the integration of image capture devices, such as sensors or smartphones, for real-time disease detection in agricultural settings. Evaluation of the disease detection system will encompass validation studies to assess the accuracy, sensitivity, and specificity of the CNN models in identifying plant diseases. Moreover, the scope extends to the development of user-friendly interfaces for farmers to interpret and act upon disease detection results effectively.

Overall, the scope of this project report encompasses the design, implementation, and evaluation of an integrated agricultural decision support system aimed at enhancing crop productivity, optimizing resource utilization, and promoting sustainable farming practices. Through the integration of machine learning and neural network approaches, the project endeavors to empower farmers with actionable insights and advanced analytics capabilities to address the challenges of modern agriculture effectively. The scope includes data preprocessing, algorithm development, system implementation, user interface design, and testing/validation phases for each system. Consideration will be given to scalability, usability, reliability, and security aspects to ensure the effectiveness and acceptance of the systems. The project scope encompasses both software development aspects (algorithm implementation, system architecture) and domain-specific knowledge (agricultural practices, crop diseases, soil science).

2 Preliminary Work

In this section the core concepts on which this study is based on are explained. Some of the relevant information is presented to give an understanding of how the models described in the following section are specified.

2.1 Time Series Data

Time series data is like a timeline of information, showing how things change over time, like stock prices or temperature readings. We use it to spot trends, predict future values, and find unusual patterns, helping us make smarter decisions in areas like finance and weather forecasting. We clean up the data and use special techniques to understand it better and make accurate predictions.

1. Crop Recommendation System:

a. Time: Monthly or seasonal interval
b. Feature
c. Soil pH levels (Nitrogen, Phosphorus, Potassium)
d. Temperature
e. Humidity
f. Ph
g. Rainfall
h. Target
i. Recommended crop for the upcoming season

2. Fertilizer Suggestion System:

a. Time: Each application or seasonal interval
b. Features
c. Soil nutrient levels (Nitrogen, Phosphorus, Potassium, etc.)
d. Crop type.
e. Target
f. Recommended type and quantity of fertilizer

3. Plant Disease Detection System:

a. Time: Continuous monitoring with periodic inspection
b. Features
c. Images of plants (captured at regular interval
d. Target
e. Presence or absence of disease, identified from the images.

Each system will have its own time series dataset, with timestamps indicating when the data was recorded. For example, for the crop recommendation system, each row of the dataset might represent the conditions like Rainfall, Ph, Temperature, Humidity, N, P and K while for the plant disease detection system, each row might represent an image captured for a particular plant.

The datasets can be generated synthetically by defining relationships between the features and the target variables based on domain knowledge or by using historical data if available. Simulating changes in environmental factors and crop health over time can provide valuable insights into the performance of the systems and help optimize their algorithms and parameters.

2.1.1 Time Series Analysis Types

Time series analysis includes many categories or variations of data, analysts sometimes must make complex models. However, analysts can't account for all variances, and they can't generalize a specific model to every sample. Models that are too complex or that try to do too many things can lead to a lack of fit. Lack of fit or overfitting models lead to those models not distinguishing between random error and true relationships, leaving analysis skewed and forecasts incorrect.

Models of time series analysis include:

Classification: Identifies and assigns categories to the data.

Curve fitting: Plots the data along a curve to study the relationships of variables within the data.

Descriptive analysis: Identifies patterns in time series data, like trends, cycles, or seasonal variation.

Explanative analysis: Attempts to understand the data and the relationships within it, as well as cause and effect.

Exploratory analysis: Highlights the main characteristics of the time series data, usually in a visual format.

Forecasting: Predicts future data. This type is based on historical trends. It uses the historical data as a model for future data, predicting scenarios that could happen along future plot points.

Intervention analysis: Studies how an event can change the data.

Segmentation: Splits the data into segments to show the underlying properties of the source information.

2.2 Pre-processing the Data

Preprocessing the data for the crop recommendation system and the plant disease detection system involves different steps due to the nature of the data. Here's how you can preprocess each type of data:

1. **Crop Recommendation System (Numerical Data):**

   Handling Missing Values: Check for missing values in the dataset and impute them using techniques such as mean imputation, median imputation, or using more advanced methods like K-nearest neighbors (KNN) imputation.

   Normalization: Normalize the numerical features (N, P, K, temperature, humidity, pH, rainfall) to ensure that each feature has a similar scale. This can be done using techniques like min-max scaling or standardization.

   Feature Engineering: Create additional features if necessary, such as derived features from existing ones (e.g., creating a new feature for soil fertility based on N, P, and K levels).

Handling Categorical Variables: If there are any categorical variables (e.g., soil type), encode them using techniques like one-hot encoding or label encoding to represent them numerically.

Feature Selection: Select relevant features that have a significant impact on crop recommendation using techniques like correlation analysis, feature importance ranking, or domain knowledge.

Splitting the Data: Split the preprocessed data into training and testing sets for model development and evaluation.

1. **Plant Disease Detection System (Image Data):**

Image Preprocessing: Resize all images to a consistent size to ensure uniformity across the dataset. Normalize pixel values to a range between 0 and 1 to facilitate model training. Augment the images with techniques like rotation, flipping, and scaling to increase the diversity of the training data and improve model generalization.

Data Augmentation: Generate augmented images by applying random transformations to the original images, such as rotation, translation, shearing, and zooming. This helps increase the variability of the dataset and improve model robustness.

Label Encoding: Encode the labels for plant diseases using techniques like one-hot encoding or label encoding to represent them numerically.

Splitting the Data: Split the preprocessed image data into training, validation, and testing sets, ensuring that each set contains a balanced distribution of images across different classes (healthy plants vs. diseased plants).

Data Loader Creation: Create data loaders or generators to efficiently load and preprocess batches of images during model training, validation, and testing. By following these preprocessing steps, you can effectively prepare the data for both the crop recommendation system and the plant disease detection system, enabling seamless model development and training.

2.3 Segmentation of the Data

Segmentation of data involves dividing the dataset into subsets based on certain criteria. For the crop recommendation system and the plant disease detection system, segmentation can be approached differently due to the nature of the data.

Here's how you can segment the data for each system:

1. Crop Recommendation System (Numerical Data):

Temporal Segmentation: Divide the dataset into temporal segments based on the time period (e.g., years, seasons) during which the data was collected. This segmentation helps capture seasonal variations and trends in agricultural factors.

Geographical Segmentation: Segment the dataset based on geographical regions or locations where the data was collected. This segmentation accounts for variations in soil types, climate conditions, and agricultural practices across different regions.

Crop Type Segmentation: Segment the dataset based on different crop types or categories. This segmentation allows for the development of crop-specific recommendation models tailored to the unique requirements of each crop.

Feature-Based Segmentation: Segment the dataset based on specific features or combinations of features (e.g., soil pH, temperature, rainfall) to create subsets with similar characteristics. This segmentation helps identify relationships and patterns within subsets of the data.

Stratified Sampling: Stratify the dataset based on key factors such as soil fertility levels, climate zones, or crop growth stages to ensure representative sampling across different conditions.

2. Plant Disease Detection System (Image Data):

A Plant Disease Detection System using image data employs machine learning to analyze images of plants and identify signs of diseases or pests. By training algorithms on labeled images, the system learns to distinguish between healthy and diseased plants, aiding in early detection and intervention. This technology helps farmers monitor crops efficiently, enabling timely treatment and minimizing yield loss, contributing to sustainable agriculture**.**

Disease Class Segmentation:

Segment the dataset based on different disease classes or categories (e.g., fungal diseases, bacterial diseases, viral diseases). This segmentation allows for the development of disease-specific detection models these enhances the accuracy and effectiveness of the disease.

Severity Level Segmentation:

Segment the dataset based on the severity levels of plant diseases (e.g., mild, moderate, severe). This segmentation helps prioritize disease management strategies based on the severity of the infection.

Crop Type Segmentation:

Segment the dataset based on different crop types or plant species. This segmentation accounts for variations in disease susceptibility and symptoms across different crops.

Image Quality Segmentation:

Segment the dataset based on the quality of images (e.g., high-resolution images, low-resolution images, images with artifacts). This segmentation helps identify and filter out poor-quality images that may affect model performance.

Anatomical Segmentation:

Segment the dataset based on different plant parts (e.g., leaves, stems, fruits) to train specialized models for detecting diseases in specific plant structures. By segmenting the data using these approaches, you can create subsets that are more homogeneous and representative of the underlying patterns and characteristics within the datasets, facilitating the development of effective models for crop recommendation and plant disease detection.

2.4 Neural Networks

Neural networks are artificial systems that were inspired by biological neural networks. These systems learn to perform tasks by being exposed to various datasets and examples without any task-specific rules. The idea is that the system generates identifying characteristics from the data they have been passed without being programmed with a pre-programmed understanding of these datasets. Neural networks are based on computational models for threshold logic. Threshold logic is a combination of algorithms and mathematics. Neural networks are based either on the study of the brain or on the application of neural networks to artificial intelligence.

The learning of neural network basically refers to the adjustment in the free parameters i.e, weights and bias. The learning rule modifies the weights and thresholds of the variables in the network. There are basically three sequences of events of learning process. These includes:

The neural network is simulated by a new environment.

Then the free parameters of the neural network are changed because of this simulation.

The neural network then responds in a new way to the environment because of the changes in its free parameters.

Supervised vs Unsupervised Learning:

Neural networks learn via supervised learning; Supervised machine learning involves an input variable *x* and corresponding desired output variable *y*. Here we introduce the concept of teacher who has knowledge about the environment. Thus, we can say that the teacher has both input-output set. The neural network is unaware of the environment. The input is exposed to both teacher and neural network, the neural network generates an output based on the input. This output is then compared with the desired output that teacher has and simultaneously an error signal is produced. The free parameters of the network are then step by step adjusted so that error is minimum.

The learning stops when the algorithm reaches an acceptable level of performance. Unsupervised machine learning has input data X and no corresponding output variables. The goal is to model the underlying structure of the data for understanding more about the data. The keywords for supervised machine learning are classification and regression. For unsupervised machine learning, the keywords are clustering and association.

Evolution of Neural Networks:

Hebbian learning deals with neural plasticity. Hebbian learning is unsupervised and deals with long-term potentiation. Backpropagation solved the exclusive-or issue that Hebbian learning could not handle. This also allowed for multi-layer networks to be feasible and efficient. If an error was found, the error was solved at each layer by modifying the weights at each node. This led to the development of support vector machines, linear classifiers, and max pooling. The vanishing gradient problem affects feedforward networks that use back propagation and recurrent neural network. This is known as deep learning.

Hardware-based designs are used for biophysical simulation and neurotrophic computing. They have large scale component analysis and convolution creates new class of neural computing with analogue. This also solved back-propagation for many-layered feedforward neural networks. Convolutional networks are used for alternating between convolutional layers and max-pooling layers with connected layers (fully or sparsely connected) with a final classification layer. Convolutional networks employ a structure that alternates between convolutional and max-pooling layers, combined with fully or sparsely connected layers, culminating in a classification layer, facilitating effective feature extraction and pattern recognition in tasks like image classification.

Types of Neural Networks

There are many types of neural networks that are used. Some of them are:

Multilayer perceptron which has three or more layers and uses a nonlinear activation function.

Convolutional neural network that uses a variation of the multilayer perceptron.

Recursive neural network that uses weights to make structured predictions.

Recurrent neural network that makes connections between the neurons in a directed cycle. The long short-term memory neural network uses the recurrent neural network architecture and does not use an activation function.

Sequence-to-sequence modules which use two recurrent networks.

Shallow neural networks which produce a vector space from an amount of text.

Deep Convolutional Neural Network

Deep convolutional neural networks (CNN or DCNN) are the type most used to identify patterns in images and videos. DCNNs have evolved from traditional artificial neural networks, using a three-dimensional neural pattern inspired by the visual cortex of animals.

Deep convolutional neural networks are mainly focused on applications like object detection, image classification, recommendation systems, and are also sometimes used for natural language processing.

A DCNN uses a three-dimensional neural network to process the red, green, and blue elements of the image at the same time. This considerably reduces the number of artificial neurons required to process an image, compared to traditional feed forward neural networks. Deep convolutional neural networks receive images as an input and use them to train a classifier. The network employs a special mathematical operation called a "convolution" instead of matrix multiplication. The architecture of a convolutional network typically consists of four types of layers: convolution, pooling, activation, and fully connected.

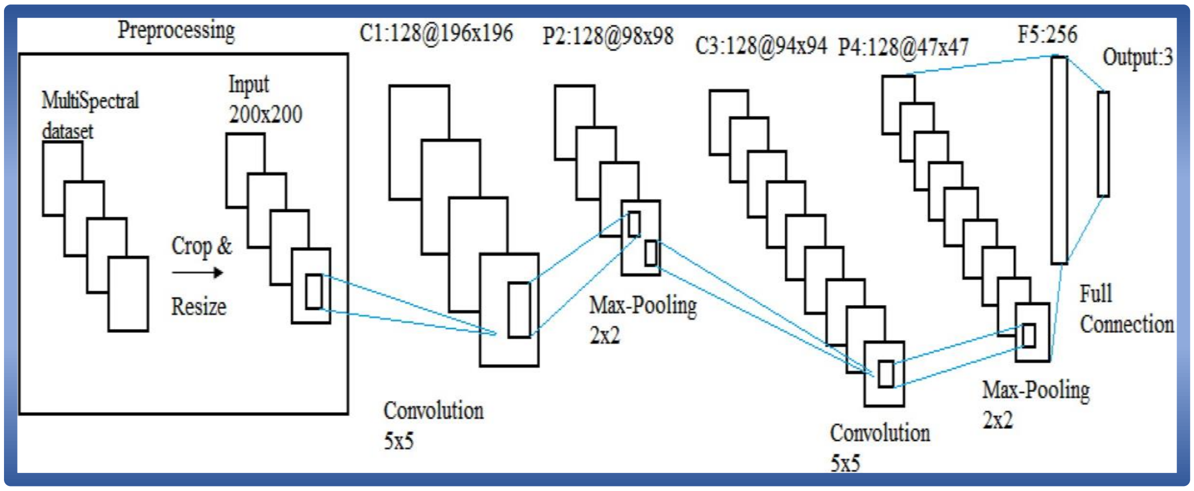


Figure 2.1 CNN Model

2.4.1 Convolutional Layer

Applies a convolution filter to the image to detect features of the image. Here is how this process works:

A **convolution**—takes a set of weights and multiplies them with inputs from the neural network.

Kernels or filters—during the multiplication process, a kernel (applied for 2D arrays of weights) or a filter (applied for 3D structures) passes over an image multiple times. To cover the entire image, the filter is applied from right to left and from top to bottom.

Dot or scalar product—a mathematical process performed during the convolution. Each filter multiplies the weights with different input values. The total inputs are summed, providing a unique value for each filter position.

ReLU Activation Layer

The convolution maps are passed through a nonlinear activation layer, such as Rectified Linear Unit (ReLU), which replaces negative numbers of the filtered images with zeros.

Pooling Layer

The pooling layers gradually reduce the size of the image, keeping only the most important information. For example, for each group of 4 pixels, the pixel having the maximum value is retained (this is called max pooling), or only the average is retained (average pooling). Pooling layers help control overfitting by reducing the number of calculations and parameters in the network.

After several iterations of convolution and pooling layers (in some deep convolutional neural network architectures this may happen thousands of times), at the end of the network there is a traditional multi-layer perceptron or “fully connected” neural network.

Fully Connected Layer

In many CNN architectures, there are multiple fully connected layers, with activation and pooling layers in between them. Fully connected layers receive an input vector containing the flattened pixels of the image, which have been filtered, corrected, and reduced by convolution and pooling layers. The SoftMax function is applied at the end to the outputs of the fully connected layers, giving the probability of a class the image belongs to – for example, is it a car, a boat, or an airplane.

Types of Deep Convolutional Neural Networks

There are five deep convolutional neural network architectures commonly used to perform object detection and image classification.

1. R-CNN
2. Fast R-CNN
3. Google Net (2014)
4. VGGNet (2014)
5. ResNet (2015)

R-CNN

Region-based Convolutional Neural Network (R-CNN), is a network capable of accurately extracting objects to be identified in the image. However, it is very slow in the scanning phase and in the identification of regions.

The poor performance of this architecture is due to its use of the selective search algorithm, which extracts approximately 2000 regions of the starting image. Afterwards it executes N CNNs on top of each region, whose outputs are fed to a support vector machine (SVM) to classify the region.

Fast R-CNN

Fast R-CNN is a simplified R-CNN architecture, which can also identify regions of interest in an image but runs a lot faster. It improves performance by extracting features before it identifies regions of interest. It uses only one CNN for the entire image, instead of 2000 CNN networks on each superimposed region. Instead of the SVM which is computationally intensive, a SoftMax function returns the identification probability. The downside is that Fast R-CNN has lower accuracy than R-CNN in terms of recognition of the bounding boxes of objects in the image.

Google Net (2014)

Google Net, also called Inception v1, is a large-scale CNN architecture which won the ImageNet Challenge in 2014. It achieved an error rate of less than 7%, close to the level of human performance. The architecture consists of a 22-layer deep CNN based on small convolutions, called “inceptions”, batch normalization, and other techniques to decrease the number of parameters from tens of millions in previous architectures to four million.

VGGNet (2014)

VGGNet is a deep convolutional neural network architecture with 16 convolutional layers. It uses 3x3 convolutions and trained on 4 GPUs for more than two weeks to achieve its performance. The downside of VGGNet is that unlike Google Net, it has 138 million parameters, making it difficult to run in the inference stage.

ResNet (2015)

The Residual Neural Network (ResNet) is a CNN with up to 152 layers. ResNet uses “gated units”, to skip some convolutional layers. Like Google Net, it uses heavy batch normalization. ResNet uses an innovative design which lets it run many more convolutional layers without increasing complexity. It participated in the ImageNet Challenge 2015, achieving an impressive error rate of 3.57%, while beating human-level performance on the trained dataset.

ResNet is a deep convolutional neural network architecture that has made significant contributions to the field of computer vision. It was introduced by Kaiming He, Xiangyi Zhang, Shaoqing Ren, and Jian Sun in their paper "Deep Residual Learning for Image Recognition" in 2015.

Here are some key points about ResNet:

1. Residual Learning: The fundamental innovation of ResNet lies in its use of residual learning. Traditional neural networks learn to approximate a mapping from input to output. In contrast, ResNet aims to learn residual functions, i.e., the difference between the input and the desired output. This is done by introducing shortcut connections, also known as skip connections, that bypass one or more layers.

2. Deeper Architectures: ResNet allows the training of extremely deep neural networks, surpassing the limitations faced by earlier architectures. The shortcut connections mitigate the vanishing gradient problem, which occurs when gradients diminish as they propagate through many layers during training. As a result, ResNet architectures can be significantly deeper, reaching over a hundred layers, while still being trainable effectively.

3. Architecture: ResNet architectures are typically composed of several blocks, each containing multiple convolutional layers. The key element in these blocks is the residual unit, which consists of two or more convolutional layers with shortcut connections. These shortcut connections add the original input to the output of the stacked layers. The most common type of residual unit used in ResNet is the bottleneck unit, which consists of three convolutional layers: 1x1, 3x3, and 1x1 convolutions.

4. Versions: ResNet comes in several versions, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The numbers in their names indicate the total number of layers, including both convolutional and fully connected layers. For example, ResNet-50 has 50 layers.

5. Applications: ResNet architectures have been widely adopted in various computer vision tasks, including image classification, object detection, and image segmentation. Pre-trained versions of ResNet on large datasets like ImageNet are often used for transfer learning, where the learned features from these networks are fine-tuned for specific tasks with smaller datasets.

6. Performance: ResNet achieved state-of-the-art performance on various benchmark datasets at the time of its introduction and has become a cornerstone in the development of deep learning models for computer vision tasks.

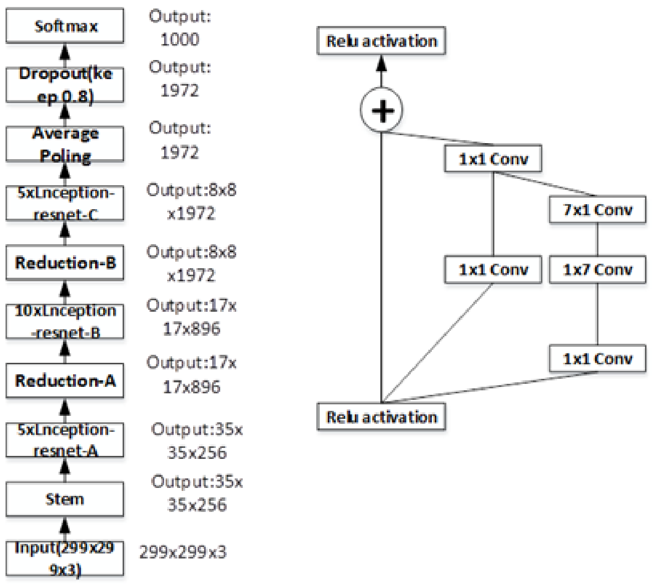


Figure 2.2 Residual Component of ResNet-20

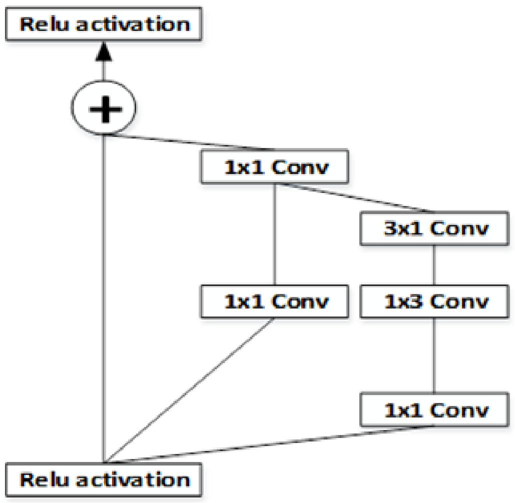


Figure 2.3 Inception Resnet_Modules

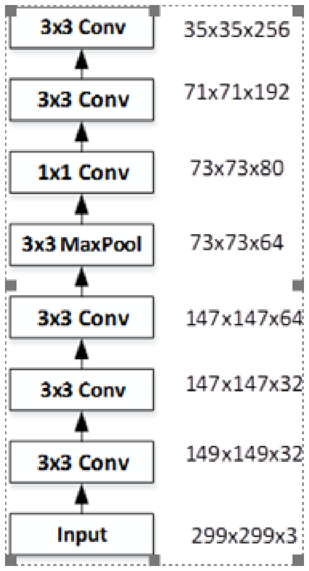


Figure 2.4 Resnet_C Module

ResNet, or Residual Network, is a groundbreaking deep neural network architecture designed to address the challenge of training very deep networks. Introduced in 2015 by Kaiming He and his team, ResNet revolutionized deep learning by introducing skip connections, allowing gradients to flow more directly during training. This innovation alleviates the vanishing gradient problem encountered in deep networks and enables the successful training of extremely deep models. ResNet's residual blocks consist of convolutional layers with shortcut connections, where the original input is added to the output before passing through an activation function. This approach allows the network to learn residual mappings, making it easier to optimize and resulting in superior performance on various computer vision tasks, including image classification, object detection, and image segmentation.

The performance of ResNet has been nothing short of remarkable across a wide array of tasks in computer vision. In image classification tasks, ResNet models have consistently achieved state-of-the-art accuracy on benchmark datasets such as ImageNet. Their ability to learn hierarchical features at various levels of abstraction has made them incredibly effective feature extractors.

Furthermore, ResNet's impact extends beyond image classification. Its architecture has been adapted and applied to tasks such as object detection, semantic segmentation, and even natural language processing, where its ability to capture intricate patterns in data has proven invaluable.

The success of ResNet has sparked a surge of research into understanding the principles behind its effectiveness and exploring further improvements. Variants such as Wide Residual Networks (WRN), densely connected networks (DenseNet), and more recently, EfficientNet, have built upon ResNet's foundations, pushing the boundaries of what is achievable in terms of model accuracy and efficiency.

3 Requirement Analysis

An integrated agricultural system leveraging soil analysis, crop data, and climate information can suggest fertilizers based on soil conditions and crop type while considering the farmer's budget and preferences. By incorporating machine learning algorithms, it can detect a wide range of plant diseases early and accurately, providing farmers with tailored recommendations for disease management, including cultural practices, chemical controls, and biological interventions, thus ensuring optimized crop health and yield.

3.1 Functional Requirements

The Plant Disease Detection System's functional requirements outline its key features, like image upload, disease detection, and treatment recommendations. It aims for user-friendliness, scalability, and accuracy across diverse plant types and conditions, facilitating effective crop management for farmers.

Crop Recommendation System:

1. Input Data:

Historical crop data including soil type, climate conditions, previous crop yields, and geographical location. User input for specific preferences or constraints (e.g., preferred crop type, available resources).

2. Preprocessing:

Data cleaning and normalization are essential preprocessing steps, ensuring accuracy by addressing missing values, outliers, and standardizing data formats.

3. Algorithm Selection:

Implementation of Random Forest, Decision Tree, and Gaussian Naive Bayes supervised learning algorithms. Each algorithm should be trained on the input data to create models for crop recommendation.

4. Recommendation Generation:

Utilize the trained models to recommend suitable crop types based on input parameters.

5. Output:

Display recommended crop types along with confidence scores. Provide explanations for the recommendations, highlighting key factors influencing the decision.

Fertilizer Suggestion System:

1. Input Data:

Soil nutrient levels obtained through soil testing. Crop type and growth stage. Environmental factors such as temperature and precipitation.

2. Fertilizer Database:

Maintain a database of various types of fertilizers along with their nutrient compositions.

3. Algorithm Implementation:

Develop algorithms to analyze soil nutrient levels and crop requirements to suggest appropriate fertilizers. Algorithms can provide farmers with personalized fertilizer recommendations tailored to their specific soil and crop conditions, ultimately leading to improved productivity, sustainability, and profitability in agriculture.

4. Recommendation Generation:

Generate fertilizer recommendations based on the analysis, considering nutrient deficiencies and crop nutrient needs.

5. Output:

Present recommended fertilizers along with application instructions (dosage, frequency, application method). Provide explanations for the recommendations, including the rationale behind each suggestion.

Plant Disease Detection System:

1. Input Data: Images of plant leaves or affected areas.

2. Preprocessing: Image preprocessing techniques such as resizing, normalization, and noise reduction.

3. Convolutional Neural Network (CNN): Implement a CNN architecture for image classification and disease detection. Train the CNN model on a dataset of labeled plant disease images.

4. Detection and Classification: Utilize the trained CNN model to detect and classify diseases in the input images.

5. Output: Display the detected diseases along with confidence scores. Provide recommendations for treatment or management strategies specific to each identified disease. Include visual indicators highlighting the affected areas on the input images.

General Requirements:

1. User Interface:

Develop a user-friendly interface for easy input of data and viewing of recommendations.

2. Performance:

Aim for fast and accurate processing of input data. Implement optimizations to minimize processing time and resource usage.

3. Scalability:

Design the systems to handle varying scales of data and user interactions.

4. Security:

Implement appropriate security measures to protect sensitive user data and system integrity.

5. Maintenance and Updates:

Include provisions for system updates, bug fixes, and maintenance to ensure long-term usability and reliability.

6. Documentation:

Provide comprehensive documentation covering system functionality, usage instructions, and technical details for future reference and troubleshooting. These functional requirements outline the key features and capabilities of each system within the project.

3.2 Interface Model

The present model only accepts an image of plant. It accepts image of any format such as JPEG, PNG, JFIF, TIFF etc.

3.2.1 Regulatory/Compliance Requirements

The database will implement a functional audit trail, tracking all changes and activities performed within the system. Each modification, access, or operation will be logged with relevant details such as timestamp, user, and action taken. This audit trail ensures accountability, transparency, and security, enabling administrators to monitor and review database activity effectively. It serves as a crucial tool for compliance, forensic analysis, and identifying unauthorized or suspicious behavior.

3.3 Non-Functional Requirements

Non-functional requirements encompass aspects like performance, security, and usability. They define the system's characteristics rather than specific behaviors, ensuring reliability, scalability, and compliance with industry standards.

By addressing non-functional requirements alongside functional requirements, organizations can deliver systems that not only meet users' immediate needs but also exhibit high performance, security, usability, and reliability, ultimately enhancing overall user satisfaction and achieving business objectives.

1. Performance:

Crop Recommendation System: Ensure that the system can handle large datasets efficiently and provide crop recommendations in a timely manner, with response times not exceeding a specified threshold (e.g., 5 seconds).

Fertilizer Suggestion System: The system should deliver fertilizer recommendations promptly, considering factors such as soil nutrient levels and crop requirements, with minimal latency.

Plant Disease Detection System: Aim for fast and accurate disease detection, minimizing processing time for image analysis and classification to provide timely feedback to users.

2. Scalability:

Crop Recommendation System: Design the system to scale seamlessly as the dataset grows, accommodating an increasing number of users and expanding agricultural data sources.

Fertilizer Suggestion System: Ensure that the system can handle varying levels of user demand and fertilizer database size, scaling resources as needed to maintain performance.

Plant Disease Detection System: Implement scalability measures to support the analysis of large volumes of plant images and accommodate future growth in data and user base.

3. Reliability:

Crop Recommendation System: The system should be highly reliable, minimizing the risk of errors in crop recommendations and ensuring consistent performance under different environmental conditions.

Fertilizer Suggestion System: Ensure the accuracy and reliability of fertilizer recommendations, reducing the likelihood of incorrect suggestions that could negatively impact crop health and yield.

Plant Disease Detection System: Aim for high reliability in disease detection, minimizing false positives and false negatives to provide trustworthy results to users.

4. Security:

Crop Recommendation System: Implement robust security measures to protect sensitive agricultural data and user information, ensuring confidentiality and integrity throughout the recommendation process.

Fertilizer Suggestion System: Safeguard user data and fertilizer database from unauthorized access or manipulation, employing encryption and access controls where necessary.

Plant Disease Detection System: Ensure the security of plant image data and user interactions, preventing unauthorized access or tampering with sensitive information related to disease detection.

5. Usability:

Crop Recommendation System: Design an intuitive user interface that simplifies the input of agricultural data and facilitates the understanding of crop recommendations, catering to users with varying levels of technical expertise.

Fertilizer Suggestion System: Create a user-friendly interface for fertilizer selection, providing clear explanations for recommendations and guidance on application methods to support farmers' decision-making processes.

Plant Disease Detection System: Ensure ease of use for users interacting with the disease detection system, offering intuitive image upload and result interpretation features to aid in plant health management.

6. Maintainability:

Crop Recommendation System: Develop the system using modular and well-documented code, facilitating future updates, enhancements, and maintenance tasks by the development team.

Fertilizer Suggestion System: Ensure that the system architecture is designed for easy maintenance and scalability, allowing for seamless integration of new fertilizer products and updates to recommendation algorithms.

Plant Disease Detection System: Implement coding best practices and version control systems to streamline maintenance and code management efforts, ensuring the longevity and reliability of the disease detection system.

3.4 Feasibility Study

A feasibility analysis determines the ability to succeed in projecting their idea, i.e., it evaluates whether the project's potential for success such as ensuring whether a project is legally, technically and as well as economically justifiable. This study specifies whether investing in it is useful or not and whether the project is doable or not. There are different types of feasibility study and the suitable one for this study is given below:

3.4.1 Technical Feasibility

Technical feasibility specifies whether the technical resources are available and that can convert the ideas into working system. the technical feasibility evaluates the hardware, software and other technical requirements of our proposed system that is LSTM

1. Crop Recommendation System:

Data Availability: Availability of historical crop data, soil information, and climate data is essential for training the recommendation algorithms. Ensuring access to relevant datasets may require collaboration with agricultural research institutions or data providers.

Algorithm Implementation: Implementation of Random Forest, Decision Tree, and Gaussian Naive Bayes algorithms requires expertise in machine learning and data analysis. The technical capability to develop and train these algorithms must be assessed.

2. Fertilizer Suggestion System:

Fertilizer Database: Building and maintaining a comprehensive fertilizer database with accurate nutrient compositions and application guidelines requires technical expertise in agronomy and data management.

Algorithm Complexity: Developing algorithms to analyze soil nutrient levels and crop requirements for fertilizer recommendation may require advanced data analysis techniques and computational resources.

3. Plant Disease Detection System:

Image Data Availability: Availability of labeled image datasets containing examples of plant diseases is crucial for training the CNN model. Data acquisition efforts may be required to collect and annotate relevant images.

CNN Model Complexity: Developing and training a CNN model for disease detection requires expertise in deep learning and image processing. Assessing the technical feasibility of implementing and optimizing the model is essential.

3.4.2 Operational Feasibility

Operational feasibility is a type of feasibility that studies to analyze and determine whether the user needs can be met by the completing the project [3]. Operational feasibility studies also examine how a project plan satisfies the requirements analysis phase of system development. In our study, the forecasting of drought help us to know how many possible cases can be come in certain time and this help the government to take the measures.

1. Crop Recommendation System:

User Acceptance: Farmers and agricultural professionals must be willing to adopt and use the crop recommendation system. Conducting surveys or interviews to gauge user interest and requirements can help assess operational feasibility.

Integration with Existing Systems: Compatibility with existing agricultural management systems and tools is important for seamless integration and user adoption.

2. Fertilizer Suggestion System:

User Acceptance: Farmers and agricultural professionals must be willing to adopt and use the Fertilizer recommendation system. Conducting surveys or interviews to gauge user interest and requirements can help assess operational feasibility.

Integration with Existing Systems: Compatibility with existing agricultural management systems and tools is important for seamless integration and user adoption. prioritizing compatibility with existing systems promotes innovation while minimizing disruption, ultimately driving greater adoption and impact within the agricultural industry.

3. Plant Disease Detection System:

User Training: Farmers and agricultural extension workers may require training on how to use the plant disease detection system effectively. Providing user-friendly interfaces and instructional materials can facilitate adoption.

Field Deployment: Ensuring that the system can be deployed and used in field settings where access to internet connectivity and computing resources may be limited is important for operational feasibility.

Combined System: The effectiveness of agricultural technology systems like crop recommendation, fertilizer suggestion, and plant disease detection hinges on user acceptance and integration with existing agricultural management tools. Engaging farmers and agricultural professionals through surveys or interviews can assess operational feasibility and user needs, facilitating adoption. Seamless integration with existing systems ensures compatibility and streamlines processes, promoting widespread adoption. For plant disease detection, user training and field deployment considerations are vital for effective usage in environments with limited connectivity. Prioritizing these factors is crucial for successful deployment, driving innovation and efficiency in agricultural practices.

4 System Design

4.1 Existing System

Plantix is a mobile application that offers plant disease detection and pest recognition services. It uses image recognition technology.

AgroCares Scanner is a handheld device that measures nutrient levels in soil samples and provides instant analysis through a mobile application.

An AI-powered platform that offers personalized recommendations for plant care, including watering schedules, fertilizer application, and pest management strategies.

4.1.1 Naive Bayes:

The Bayes' theorem serves as the foundation of the Naive Bayes classifier logic to predict across predictors. Simply put, we collect data for well-known features rather than unverified features. It is a relatively straightforward and effective classification technique that presumes independence between predictors.

The prior knowledge is multiplied by the probability of occurrence and then divided by the evidence. The name of the category class is changed to Yes and No, which are encoded as 1 and 0 when it is implemented. Naive Bayes can be used for several tasks,such as creating real-time predictions, determining the likelihood of numerous target attribute classes, spam filtering, and constructing recommendation systems when paired with collaborative filtering. Based on the highest probability, the prediction is made. The accuracy rating is then calculated.

4.1.2 Random Forest:

Random Forest is a supervised learning algorithm. We improve model efficiency by incorporating the ability to connect numerous classifiers to tackle complicated issues, which is based on ensemble learning techniques. Decision trees are trained on portions of the available dataset to multiple classifiers, which help increase the model's accuracy. The random forest algorithm has three parameters: Node size: Terminal nodes offer suggestions for the recommended number of observations, n no of tree that need to grow, m no of variable to split the node.

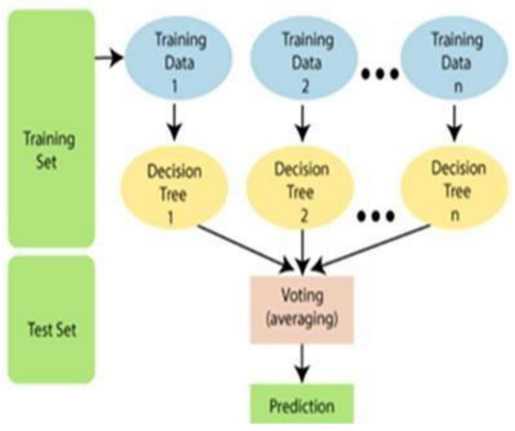


Figure 4.1 Random Forest

4.1.3 Decision Tree:

One of the most effective and well-liked tools for classification and prediction is the decision tree. It is a greedy approach. It is a supervised learning algorithm that uses trees to represent attributes and class labels. Decision trees are frequently used to create training prototypes that is formed by the learning of decision rules obtained from previous training data to predict the class or value of a target variable. Decision nodes and decision leaves are the two categories into which decision trees can be divided. The outcome, or ultimate consequence, is leaves. Every node in the tree is a test case fora certain property, and every edge descending from that node represents a potential response to the test case. Recursive in nature, this procedure repeats for every subtree rooted at the new node.

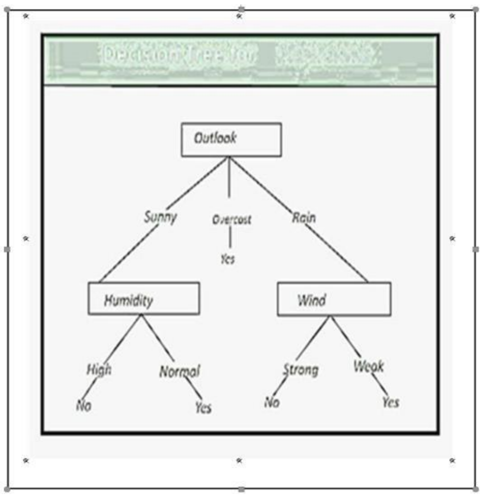


Figure 4.2 Decision Tree

4.1.4 Support Vector Machine:

Support Vector Machine known as SVM is a supervised learning technique. It can be applied to problems involving classification and regression. To increase the effectiveness of outlier detection, it is applied to labelled datasets. Both linear and nonlinear learning problems can be resolved using this approach. The usage of the acquired vectors to categorize the dataset and is a crucial component of this supervised learning process. Compared to most algorithms, the SVM (Support Vector Machine) algorithm is more effective and memory- friendly.

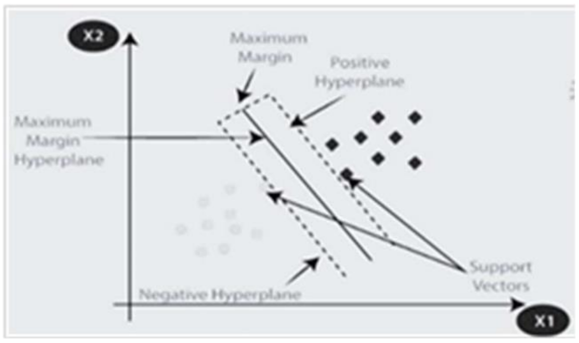


Figure 4.3 Support Vector Machine

KNN (K-Nearest Neighbor):

We can use the K-Nearest Neighbors approach to solve classification and regression prediction issues. It falls under supervised learning. This method is useful for interpreting outputs, calculating time and predictive power. KNN also belongs to machine learning techniques i.e, also known for the sample-based learning method. It can be used to forecast fresh datasets while containing information from earlier datasets. This uses distance functions, like the Manhattan distance or the Euclidean distance. This can be used to determine how far your sample is from every other training sample. Determine the goals for new samples.

The weighted sum of the K nearest target values determines the goal value. K's value might be inversely related to the outcome. High variance and low bias are always indicated by small values of K. A large value of K indicates small variance and large deviation. The advantage of KNN is that it requires no training or tuning. This KNN uses data sampling for prediction on new datasets.

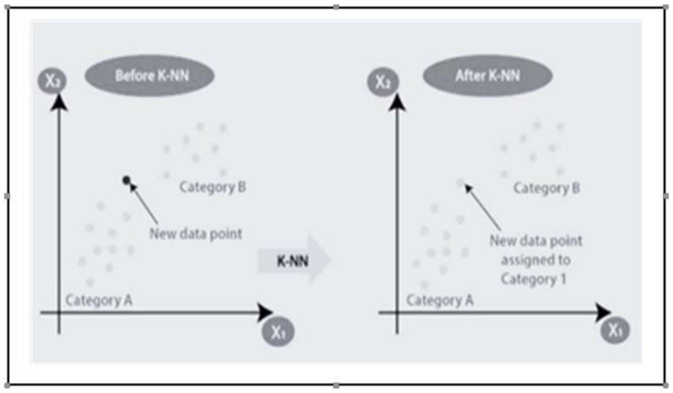


Figure 4.2 K-Nearest Neighbor

4.2 Proposed System

The proposed system uses computer techniques to assist farmers in managing their crops better. It provides guidance on which crops to grow, which fertilizers to use, and how to identify plant diseases. This platform helps farmers make smarter decisions and enhance their farming methods. By analyzing large amounts of data, it can offer accurate suggestions for crop management, fertilizer suggestion, and disease detection.

4.2.1 Decision Tree:

One of the most effective and well-liked tools for classification and prediction is the decision tree. It is a greedyapproach. It is a supervised learning algorithm that uses treesto represent attributes and class labels. Decision trees are frequently used to create training prototypes that is formed bythe learning of decision rules obtained from previous training data to predict the class or value of a target variable. Decision nodes and decision leaves are the two categories into which decision trees can be divided. The outcome, or ultimate consequence, is leaves. Every node in the tree is a test case fora certain property, and every edge descending from that noderepresents a potential response to the test case. Recursive in nature, this procedure repeats for every subtree rooted at thenew node.

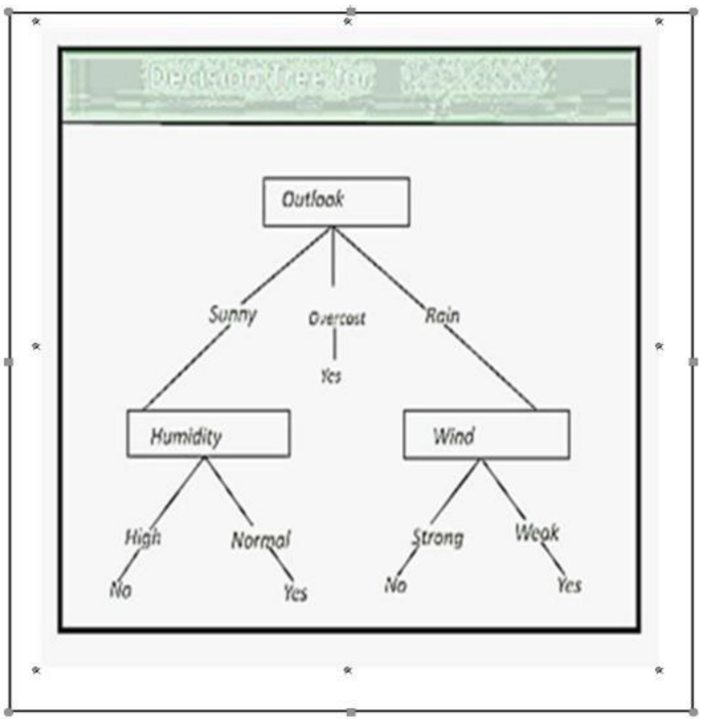


Figure 4.3 Decision Tree

4.2.2 Naive Bayes:

The Bayes' theorem serves as the foundation of the Naive Bayes classifier logic to predict across predictors. Simply put, we collect data for well-known features rather than unverified features. It is a relatively straightforward and effective classification technique that presumes independence between predictors.

The prior knowledge is multiplied by the probability of occurrence and then divided by the evidence. The name of the category class is changed to Yes and No, which are encoded as 1 and 0[4] when it is implemented. Naive Bayes can be used for several tasks,such as creating real-time predictions, determining the likelihood of numerous target attribute classes, spam filtering, and constructing recommendation systems when paired with collaborative filtering. Based on the highest probability, the prediction is made. The accuracy rating is then calculated.

4.2.3 Random Forest:

Random Forest is a supervised learning algorithm. We improve model efficiency by incorporating the ability to connect numerous classifiers to tackle complicated issues, which is based on ensemble learning techniques. Decision trees are trained on portions of the available dataset to providemultiple classifiers, which help increase the model's accuracy. The random forest algorithm has three parameters: Node size: Terminal nodes offer suggestions for the recommended number of observations, n no of tree that need to grow, m no of variable to split the node.

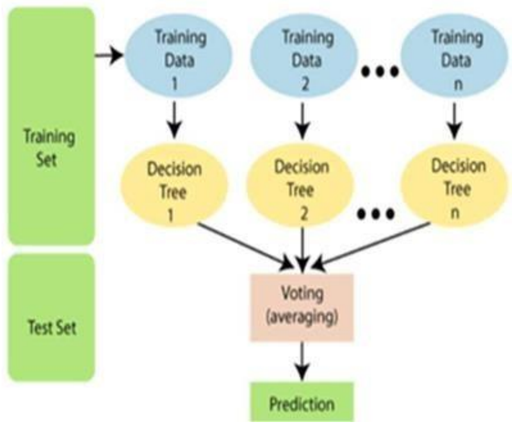
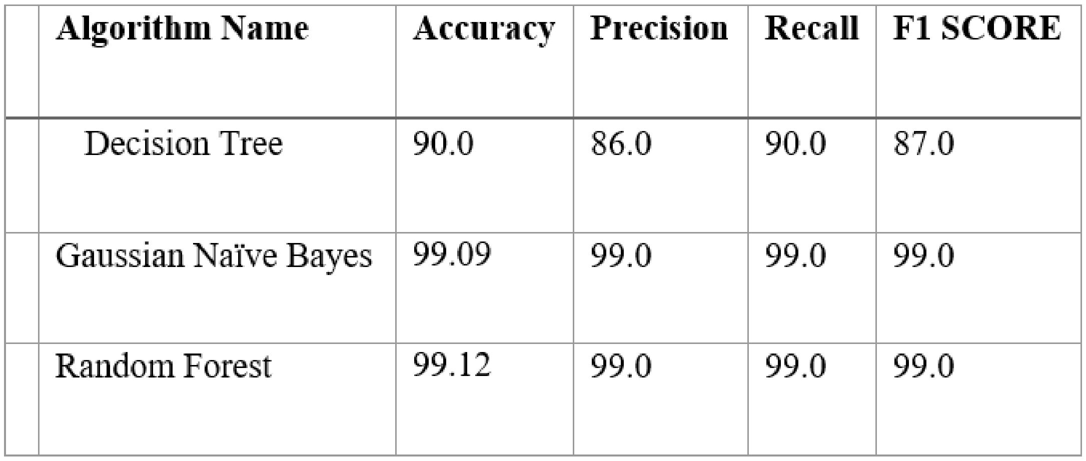


Figure 4.4 Random Forest

Table 1 Comparison Table of Decision Tree, Gaussian Naive Bayes Random Forest



	Algorithm Name	Accuracy	Precision	Recall	F1 SCORE
	Decision Tree	90.0	86.0	90.0	87.0
	Gaussian Naïve Bayes	99.09	99.0	99.0	99.0
	Random Forest	99.12	99.0	99.0	99.0

4.2.4 Deep Convolution Neural Network (DCNN)

The overall network design is given in **Error! Reference source not found.** The input to the convolution layer is a 2D matrix. The output of the convolution layer is given by **Error! Reference source not found.** DCNN

$$\boldsymbol{C_j = f(\textstyle\sum_{i=1}^{N} M_i * L_{i,j} + \rho_j)} \qquad \text{-----------} \qquad \textbf{4-1 DCNN}$$

where M_i is input matrix, C_js output matrix, $L_{i,j}$ is kernel matrices and ρ_j is bias.

The information matrix convolves with the bit frameworks and that bias esteems is added to every component of the yield after arranging the whole of every twisted.

5 Methodology

In our project, we aim to address key challenges in agriculture by developing three distinct systems. Firstly, our focus lies on the Crop Recommendation System, where we plan to implement sophisticated algorithms such as Random Forest, Decision Tree, and Gaussian Naive Bayes. Our goal is to provide farmers with personalized crop recommendations based on a thorough analysis of diverse datasets including historical crop data, soil information, climate data, and geographical factors.

We intend to design an intuitive interface that enables farmers to input relevant data easily and receive prompt recommendations. Secondly, we aim to develop a Fertilizer Suggestion System by building a comprehensive database containing nutrient compositions and application guidelines for various crops and soil types. Algorithms will analyze soil nutrient levels, crop requirements, and environmental factors to generate personalized fertilizer recommendations. Our third focus area is the Plant Disease Detection System, where we plan to train a Convolutional Neural Network (CNN) model to accurately identify and classify plant diseases based on visual symptoms captured through image analysis.

We aim to create a user-friendly interface for farmers to upload plant images and receive real-time disease detection results along with recommended management strategies. Throughout our work, we prioritize collaboration with agricultural experts and stakeholders to ensure the effectiveness, scalability, reliability, and usability of our systems in meeting the needs of the farming community.

5.1 Dataset

The dataset for this study is obtained from Kaggle, a popular platform for data resources. Kaggle offers a wide variety of datasets for analysis and research purposes. Using Kaggle's dataset provides researchers with access to quality data curated by the platform's community. This choice highlights the study's reliance on accessible and trusted data sources.

Crop Recommendation System

The proposed system in which ML is used for crop recommendation is based on previously recorded measurements of soil parameters. This technique lessens the possibility of soil degradation and aids in crop health maintenance. Many factors which include rainfall. Temperature, pH, and N, P, K, humidity are analyzed using machine learning algorithms such as Random Forest, Naive Bayes, KNN, Decision Tree, Logistic Regression on which suggestions are made for growing a suitable crop.

Plant Disease Detection

This dataset is recreated using offline augmentation from the original dataset. This dataset consists of about 87K RGB images of healthy and diseased crop leaves which is categorized into 38 different classes. The total dataset is divided into 80/20 ratio of training and validation set preserving the directory structure. A new directory containing 33 test images is created later for prediction purpose.

5.2 Performance Evaluation

To be able to draw conclusions about which model makes better forecasts, the forecasts must be evaluated. This is done using two different error measures which are described below.

The Root Mean Square Error (RMSE) is a common measure used for determining the accuracy and rate of error for different models. However, critics have suggested that the Mean Absolute Error (MAE) is a superior measure when evaluating a model. Even though the MAE might be superior, given that the RMSE is widely used, both are used to evaluate the DCNN & SVM models.

5.2.1 Root Mean Square Error

The RMSE computes the mean of the squared differences between the observed and forecasted values and takes the square root of that value. This can also be written as:

$$\boldsymbol{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y)^2}{n}} \qquad \textbf{------------- \quad 5-1 RMSE}$$

Where, $\hat{y}_i$, are the forecasted values, y, the observed values, and n is the number of forecasts.

5.2.2 Mean Absolute Error

The MAE has similar properties to the RMSE but instead of comparing the squared difference, it uses the absolute values. The MAE is the mean of the absolute values in the differences between forecasted and observed values.

$$\boldsymbol{MAE} = \sqrt{\sum_{i=1}^{n} \frac{|\hat{y}_i - y|}{n}} \qquad \text{-------------} \quad \textbf{5-2 MAE}$$

Where, $\hat{y}_i$, are the forecasted values, y, the observed values, and n is the number of forecasts.

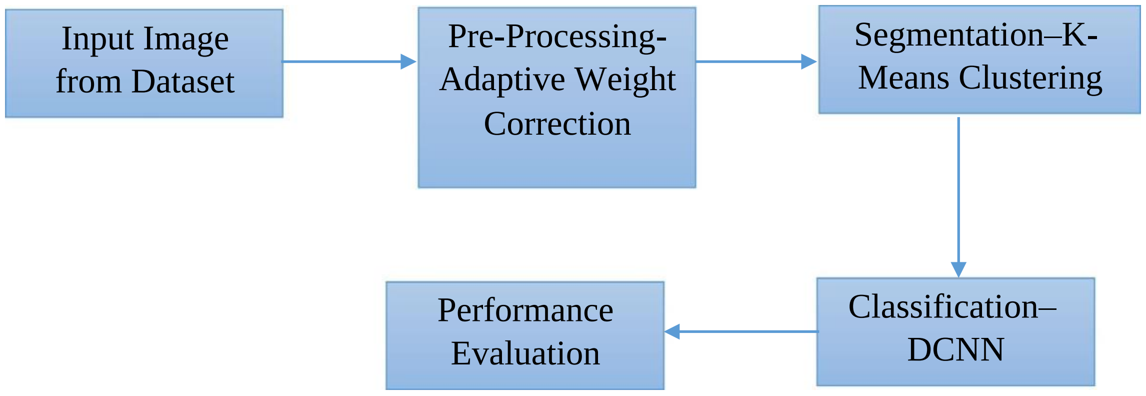

Input Image from Dataset
Pre-Processing- Adaptive Weight Correction
Segmentation–K-Means Clustering
Classification–DCNN
Performance Evaluation

Figure 5.1 Methodology of Proposed Work

6 Implementation

To access the source code and project files, please visit our GitHub repository at https://github.com/Santhosh7386/Harvestify1.git.This repository contains all the necessary files, including scripts, configuration files, and documentation, to reproduce and explore the project implementation. Feel free to clone the repository and contribute to the project.

6.1 Software Requirements

In this study we are using Python as the coding language and the implementation can be done in any of its environments such as Jupyter Notebook, Google Colab…etc. We don't even require any kind of download & installation of tool that are available online. We can access The Google Colab through our Google accounts. For faster execution prefer Jupyter Notebook as it doesn't require uploading of files all the time.

Python:

Python was conceived in the late 1980s by Guido Van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker.

In January 2019, active Python core developers elected a five-member Steering Council to lead the project. As of November 2022, Python 3.11.0 is the current stable release. Notable changes from 3.10 include increased program execution speed and improved error reporting.

6.2 Libraries and APIs used.

For our current working/developing/proposed system we require some libraries as well as API's (Application Programming Interface) such as:

a. NumPy
b. Pandas
c. Matplotlib
d. Scikitlearn
e. Keras
f. TensorFlow
g. Open-cv Python

6.2.1 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy [6] by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.

6.2.2 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

6.2.3 Matplotlib

Matplotlib is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc.

6.2.4 Scikit-learn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. This software library features various classification, regression and clustering algorithms including SVM, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project

6.2.5 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet is also the author of the exception deep neural network model.

6.2.6 TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production.

6.2.7 Open-cv Python

OpenCV is a library of programming functions mainly for real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage, then Itseez (which was later acquired by Intel). The library is cross-platform and licensed as free and open-source software under Apache License, OpenCV features GPU acceleration for real-time operations.

6.3 Code

Before implementation of our code, we may come to know that our current working flow/ proposed system must be implemented in 3 modules via.

1. Pre-processing
2. Segmentation
3. Classification

The first step in the implementation is importing of the libraries. As we listed above, we are using several libraries to implement the proposed system. So, import all the libraries NumPy, Pandas, Matplotlib, Scikitlearn, Keras, TensorFlow and Open-cv python.

Loading the Dataset

The above lines of code help in loading the dataset into our working model.

Reading a Sample Image

The above lines of code help in reshaping of data & reading a sample image.

The output of the above code is as follows.

The params can be calculated by using the formula:

$$\boldsymbol{params = (kernel\_size \times input\_channels + 1) \times numberof\ filters} \textbf{ ----- 6-1}$$

Params

6.4 Metrics Comparison

We will develop a model to predict the future values, compare them with the original labels (test dataset labels) and find the accuracy and other metrics such as recall, precision & f1-score.

6.4.1 Validation Accuracy

The code likely contains a custom function called `plot_accuracies(history)` that takes the training history of a machine learning model (`history`) as input. This function is designed to plot the accuracies of the model over different epochs during training.

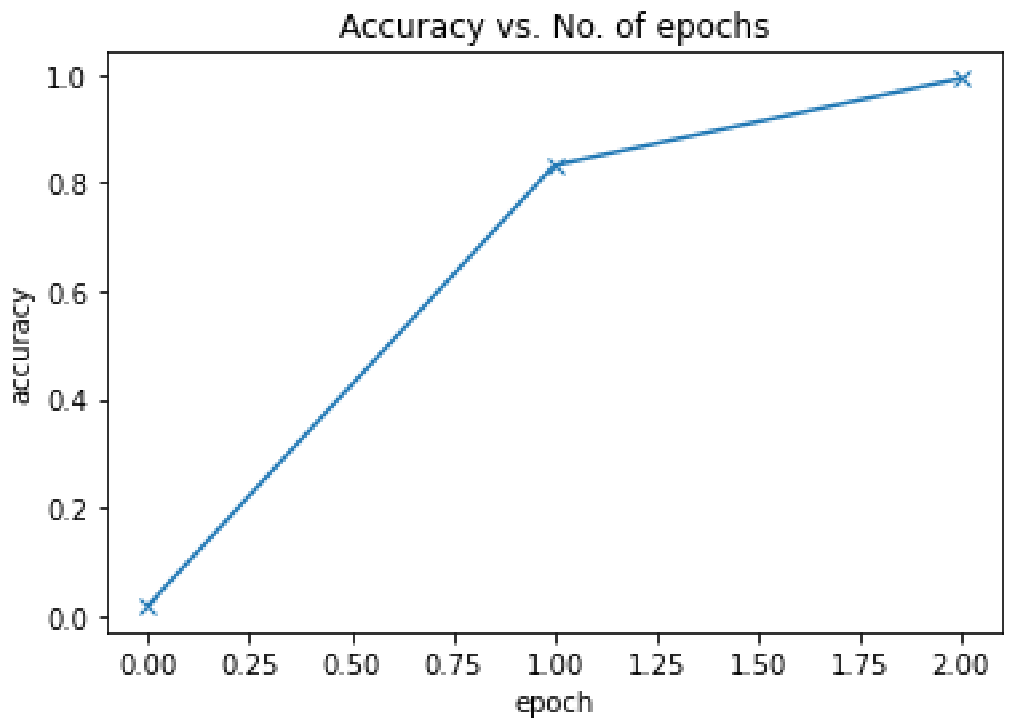


Figure 6.1 Validation Accuracy

6.4.2 Validation Loss

The code likely defines a function `plot_losses(history)` to visualize the loss values of a machine learning model over different training epochs. It takes the training history (`history`) as input, extracts the loss values, creates a plot (using a plotting library like Matplotlib), and customizes it for clarity. Finally, it displays the plot or saves it to a file.

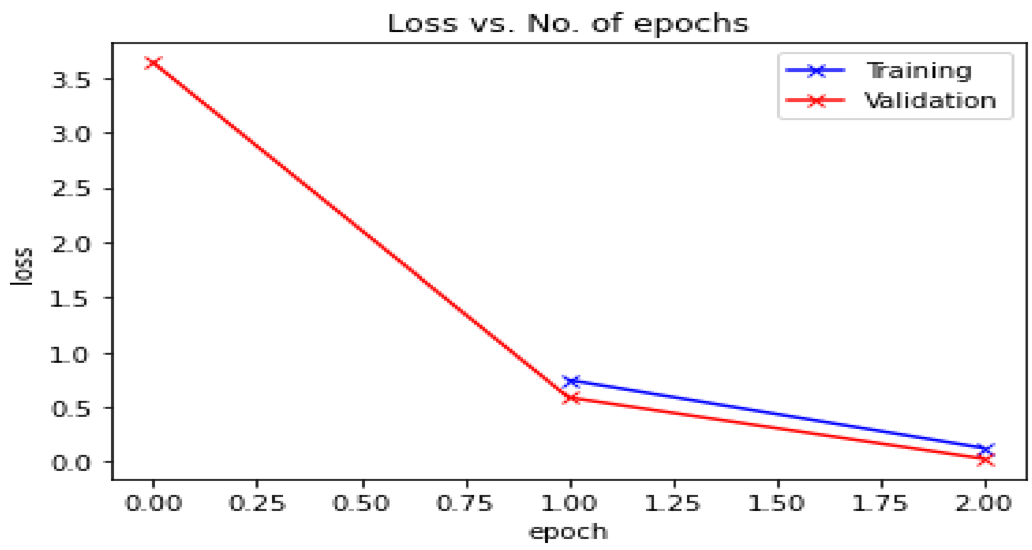


Figure 6.2 Loss vs No. of epochs

6.4.3 Learning Rate overtime

The code probably defines a function called `plot_lrs(history)` to visualize the learning rates used during training epochs. It likely takes the training history (`history`) as input, extracts the learning rates, creates a plot (using a plotting library like Matplotlib), and customizes it for clarity. Finally, it displays the plot or saves it to a file. This kind of plot can be useful for understanding how learning rates change over the course of training and can help in optimizing model training.

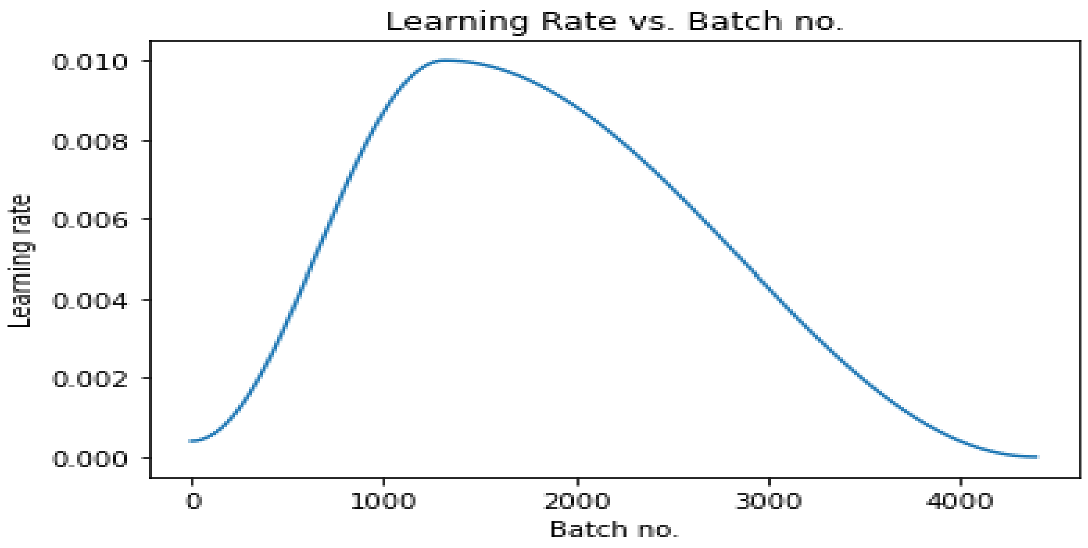


Figure 6.3 Learning Rate vs Batch No.

7 Results

Following are the steps involved in obtaining the results.

User Input: The interface provides a form where users can input soil and weather characteristics such as nitrogen, phosphorous, potassium, temperature, humidity, pH, and rainfall.

Data Transmission: Once the user submits the form, the interface sends this input data to the backend, where the deployed machine learning models are hosted.

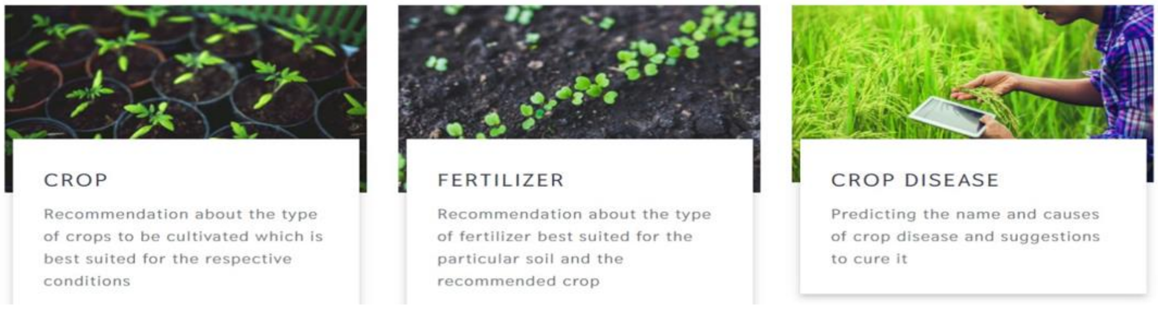
Model Prediction: The backend receives the input data and passes it through the trained machine learning models, which predict the most suitable crop based on the provided characteristics.

Result Display: The interface receives the prediction from the backend and displays it to the user, indicating the recommended crop for cultivation.

Additional Features: The interface may also provide links or options for users to explore additional features like fertilizer recommendation and disease detection, redirecting them to other sections of the application or external resources.

User Interaction: Users can interact with the interface to make multiple predictions, compare results, and explore different features offered by the application.

Home page of the website.


CROP

Recommendation about the type of crops to be cultivated which is best suited for the respective conditions

FERTILIZER

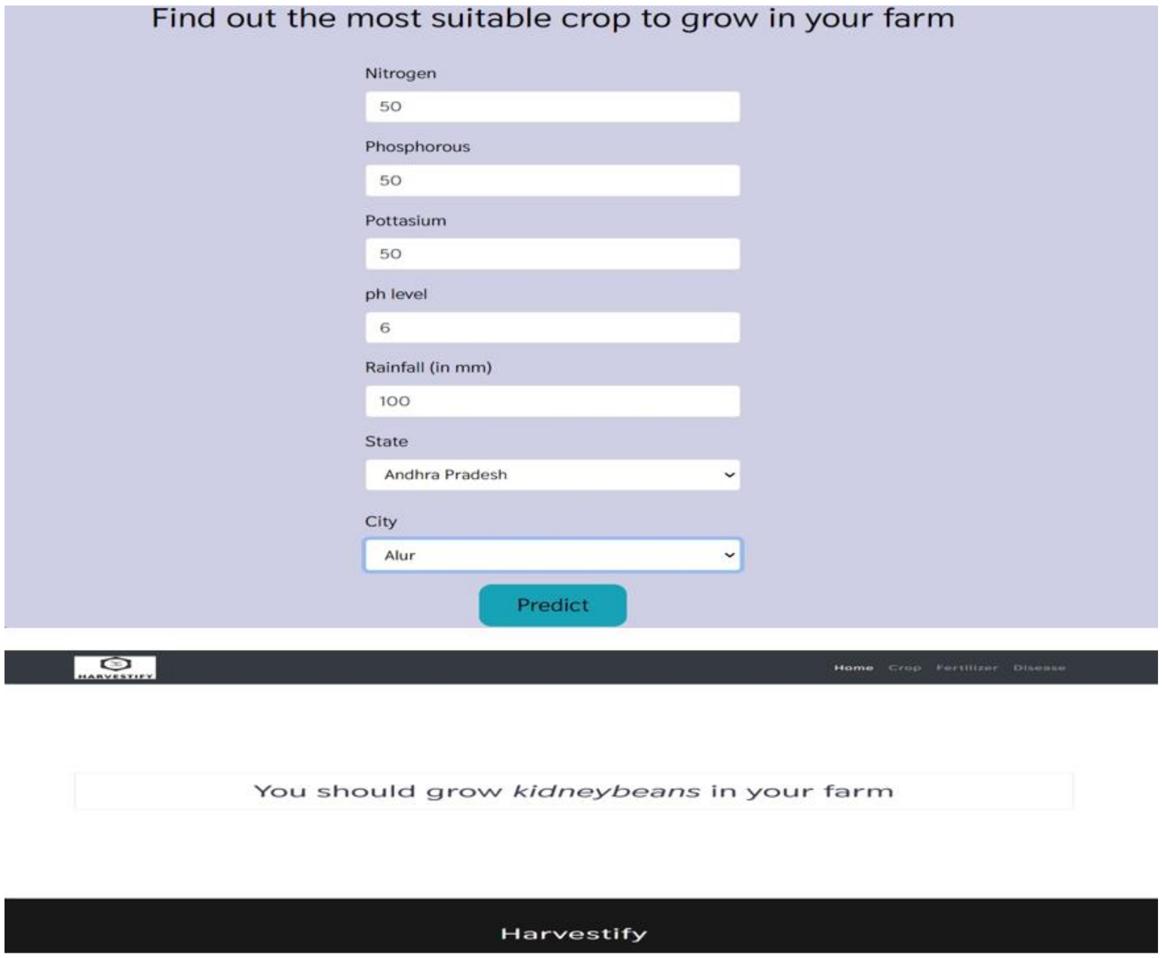
Recommendation about the type of fertilizer best suited for the particular soil and the recommended crop

CROP DISEASE

Predicting the name and causes of crop disease and suggestions to cure it

Figure 7.1 Home Page

After clicking the crop recommendation and submission result


Find out the most suitable crop to grow in your farm

Nitrogen: 50

Phosphorous: 50

Pottasium: 50

ph level: 6

Rainfall (in mm): 100

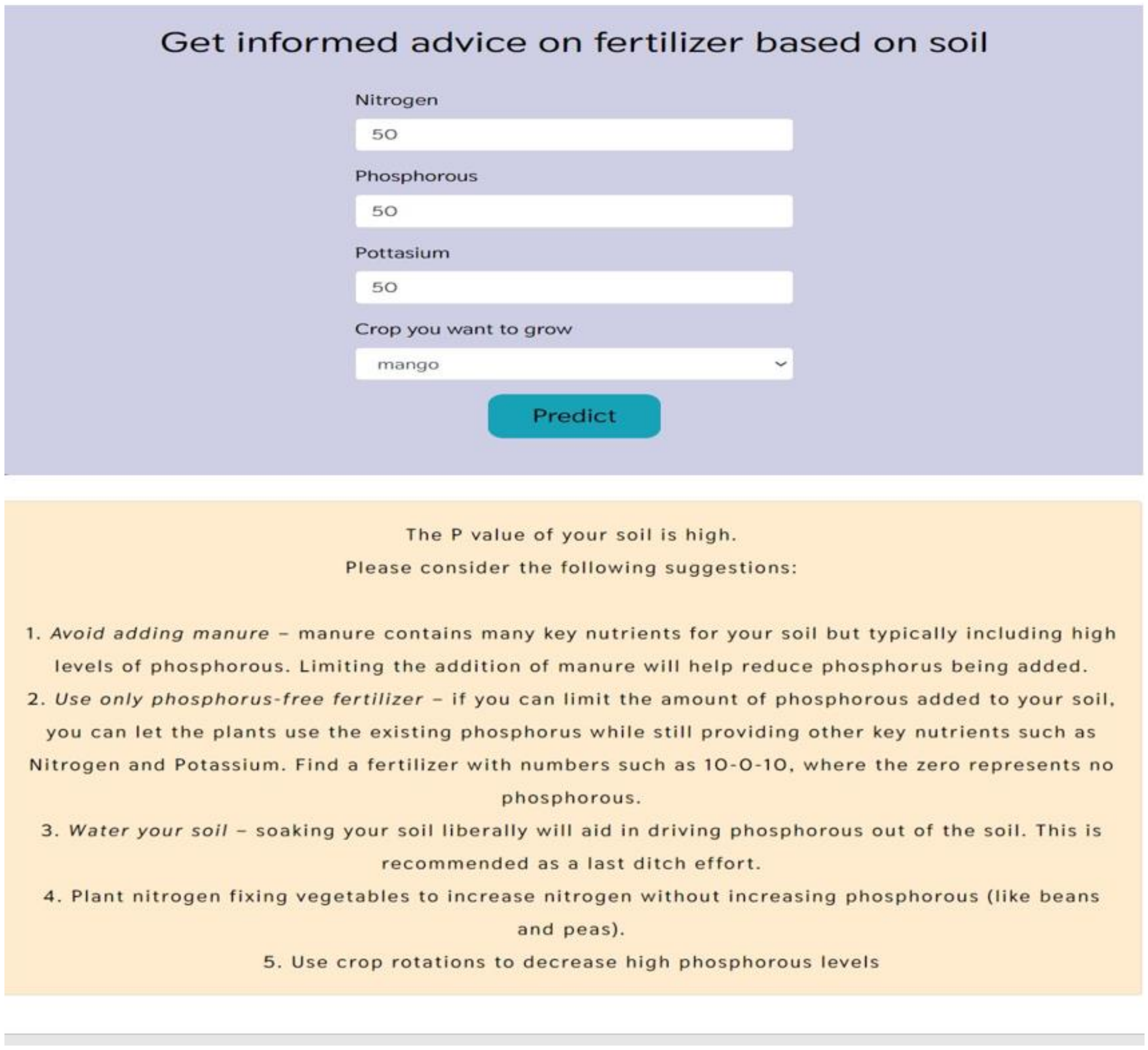
State: Andhra Pradesh

City: Alur

You should grow *kidneybeans* in your farm

Figure 7.2 Crop Recommendation

After clicking the Fertilizer suggestion and submission.


Get informed advice on fertilizer based on soil

Nitrogen: 50

Phosphorous: 50

Pottasium: 50

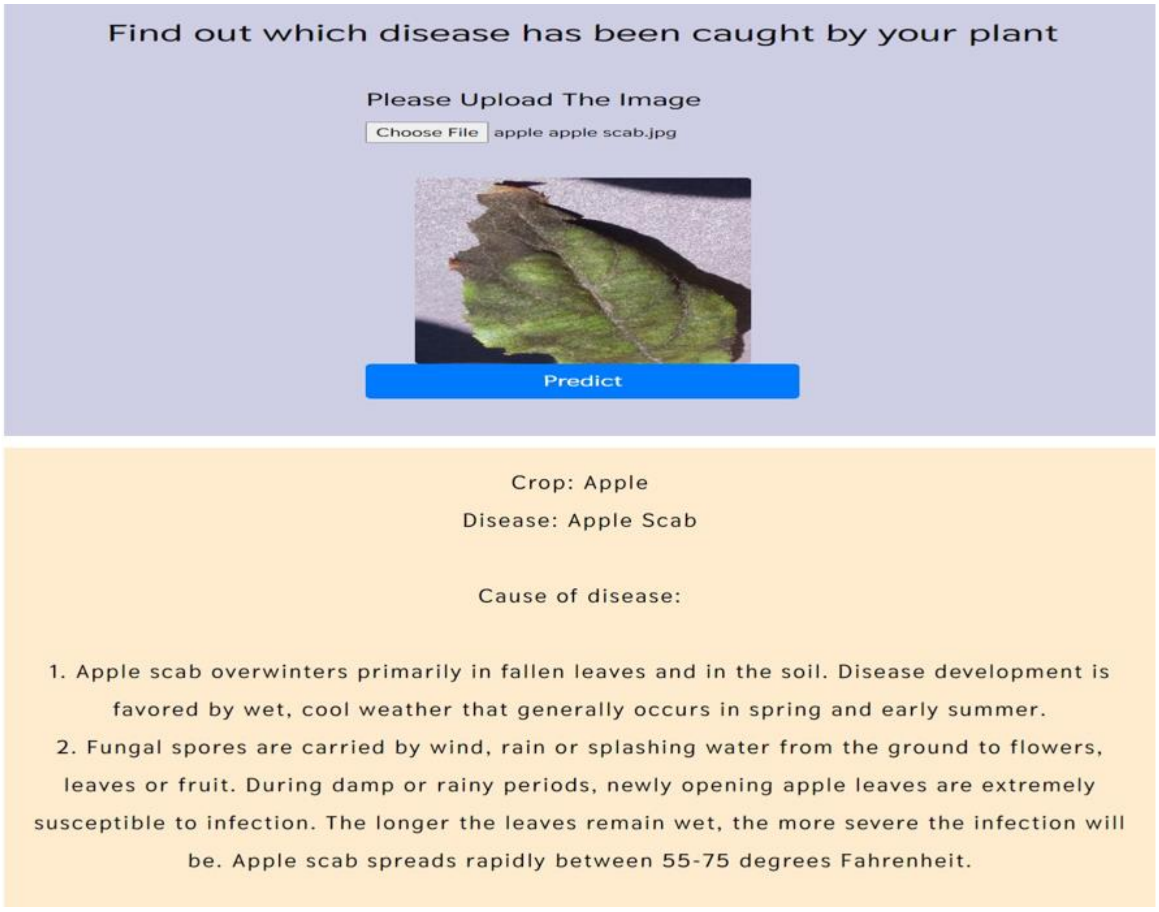
Crop you want to grow: mango

The P value of your soil is high.
Please consider the following suggestions:

1. *Avoid adding manure* – manure contains many key nutrients for your soil but typically including high levels of phosphorous. Limiting the addition of manure will help reduce phosphorus being added.
2. *Use only phosphorus-free fertilizer* – if you can limit the amount of phosphorous added to your soil, you can let the plants use the existing phosphorus while still providing other key nutrients such as Nitrogen and Potassium. Find a fertilizer with numbers such as 10-0-10, where the zero represents no phosphorous.
3. *Water your soil* – soaking your soil liberally will aid in driving phosphorous out of the soil. This is recommended as a last ditch effort.
4. Plant nitrogen fixing vegetables to increase nitrogen without increasing phosphorous (like beans and peas).
5. Use crop rotations to decrease high phosphorous levels

Figure 7.3 Fertilizer Suggestion

After clicking plant disease detection.


Find out which disease has been caught by your plant

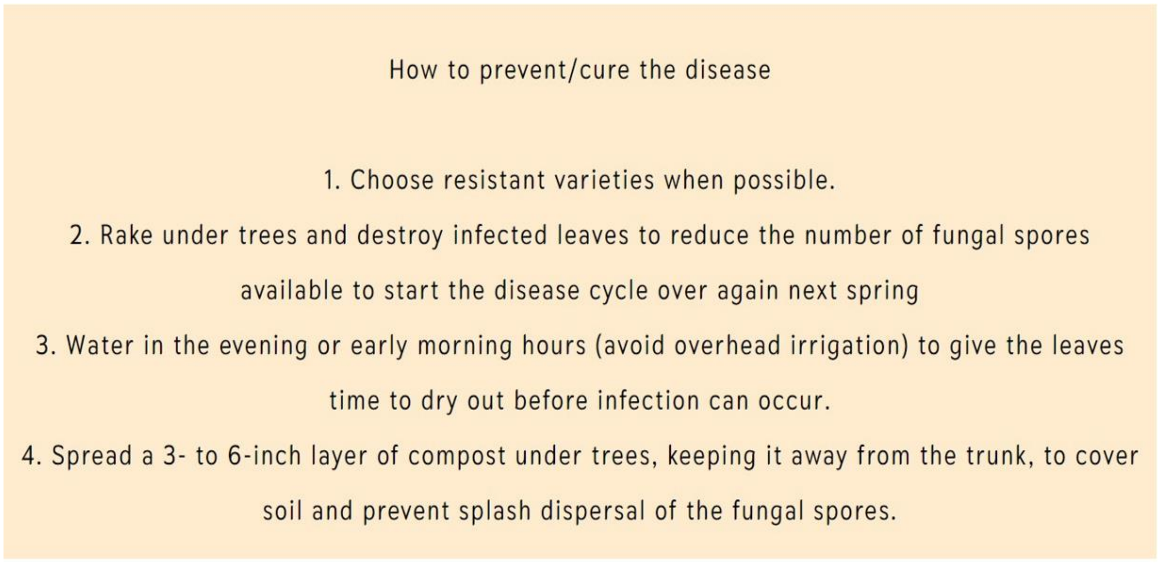
Please Upload The Image

Crop: Apple

Disease: Apple Scab

Cause of disease:

1. Apple scab overwinters primarily in fallen leaves and in the soil. Disease development is favored by wet, cool weather that generally occurs in spring and early summer.
2. Fungal spores are carried by wind, rain or splashing water from the ground to flowers, leaves or fruit. During damp or rainy periods, newly opening apple leaves are extremely susceptible to infection. The longer the leaves remain wet, the more severe the infection will be. Apple scab spreads rapidly between 55-75 degrees Fahrenheit.


How to prevent/cure the disease

1. Choose resistant varieties when possible.
2. Rake under trees and destroy infected leaves to reduce the number of fungal spores available to start the disease cycle over again next spring
3. Water in the evening or early morning hours (avoid overhead irrigation) to give the leaves time to dry out before infection can occur.
4. Spread a 3- to 6-inch layer of compost under trees, keeping it away from the trunk, to cover soil and prevent splash dispersal of the fungal spores.

Figure 7.4 Plant Disease Detection

8 Conclusion

The development of the integrated agricultural management system comprising three distinct subsystems - Crop Recommendation, Fertilizer Suggestion, and Plant Disease Detection - has significantly contributed to advancing agricultural practices and enhancing crop yield and quality.

The Crop Recommendation System leverages the power of machine learning, employing three supervised learning algorithms - Random Forest, Decision Tree, and Gaussian Naive Bayes - to provide farmers with tailored recommendations on the most suitable crops to grow based on environmental factors and historical data. By analyzing a plethora of variables including soil type, climate conditions, and geographical location, this system assists farmers in making informed decisions to optimize their agricultural productivity and profitability. The Fertilizer Suggestion System complements the Crop Recommendation System by offering personalized fertilizer recommendations tailored to the specific needs of each crop and soil condition.

By integrating soil nutrient data and crop requirements, this system enables farmers to optimize fertilizer usage, thereby promoting sustainable agricultural practices and minimizing environmental impact. Furthermore, the Plant Disease Detection System harnesses the capabilities of deep learning, particularly the CNN ResNet architecture, to accurately identify and classify plant diseases based on images of affected areas. By detecting diseases early and accurately, farmers can take timely measures to mitigate the spread of diseases and protect their crops, ultimately safeguarding their livelihoods and ensuring food security.Together, these three systems form a cohesive framework that empowers farmers with actionable insights and recommendations to optimize crop production, minimize resource wastage, and mitigate risks associated with pests, diseases, and environmental factors. By leveraging cutting-edge technologies and advanced analytical techniques, this integrated agricultural management system represents a significant step towards sustainable and efficient farming practices, contributing to the overall advancement of the agricultural sector and ensuring a more resilient and food-secure future.

9 Bibliography

[1] C. Chen and J. Fu, "Construction & application of knowledge decision tree after a disaster for water body information extraction from remote sensing images," *Journal of Remote Sensing* , vol. 2, no. 2, pp. 792-801, 2018.

[2] W. Lui and Z. Wang, "A survey of deep neural networks architecture," *Neurocomputing,* vol. 23, no. 2, pp. 25-37, 2019.

[3] B. Mamandipoor and M. Maid, "Monitoring & detecting faults in wastewater treatment plants using deep learning," *Environmental Monitoring and Assessment,* vol. 192, no. 2, p. 148, 2020.

[4] G. Zhao, Z. Xu and B. Pang, "An enhanced inundation method for urban flood hazard mapping at the large catchment scale," *Journal of Hydrology,* vol. 571, no. 2, pp. 873-882, 2019.

[5] S. Minaee and Y. Wang, "An ADMM approach to masked signal decoposition using subspace representation," *IEEE transactions on Image Preprocessing,* vol. 28, no. 28, pp. 3192-3204, 2019.

[6] Y. Lia and G. Juan, "Flood Monitoring System using DCNN," *NeuroComputing,* vol. 2, no. 4, pp. 236-248, 2019.

[7] A. Davrancvhe, G. Lefebvre and B. Poulin, "Wetland monitoring using classification trees and SPOT-5 seasonal time series," *Remote Sensing of Environment,* vol. 114, no. 2, pp. 552-562, 2012.

[8] L. Y, H. Qin and Z. Zhang, "A Survey of deep neural network architectures and their applications," *Neurocomputing,* vol. 234, no. 2, pp. 11-26, 2017.

[9] T. A and L. Dong, "Identification of water bodies in Landsat8," *Sensors,* vol. 1075, no. 2, p. 16, 2016.

[10] J. Marcais, d. Dreuzy and J. R, "Prospective Interest of Deep learning for hydrological inference," *Groundwater,* vol. 55, no. 5, pp. 688-692, 2017.