GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

DEEP LEARNING TECHNIQUES FOR SMART FARMING BASED ON SOIL AND ENVIRONMENTAL FACTORS

A project report submitted in partial fulfilment of the requirement for the award of

degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

Submitted by

Routhu Sathish	**19341A1296**
Thulluru Prem Chand	**19341A12B7**
Sai Ram Prudhvi Gummaluri	**19341A1297**
Marripudi Vyas Kanmani	**19341A1268**

Under the esteemed guidance of

Mrs. T. Daniya

Sr. Asst. Professor, Dept of Information Technology

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2015 Certified Institution)

GMR Nagar, Rajam – 532127,

Andhra Pradesh, India

2022 – 2023

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

Department of Information Technology

CERTIFICATE

This is to certify that the thesis entitled **DEEP LEARNING TECHNIQUES FOR SMART FARMING BASED ON SOIL AND ENVIRONMENTAL FACTORS** submitted by **ROUTHU SATHISH (19341A1296), THULLURU PREM CHAND (19341A12B7), SAI RAM PRUDHVI GUMMALURI (19341A1297), MARRIPUDI VYAS KANMANI (19341A1268)** has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology** in **Information Technology** of **GMRIT, Rajam** affiliated to **JNTUK** is a record of bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Signature of Supervisor
Mrs. T. Daniya
Sr. Asst. Professor
Department of IT
GMRIT, Rajam

Signature of HOD
Dr. Ajit Kumar Rout
Professor & HOD
Department of IT
GMRIT, Rajam

The report is submitted for the viva-voice examination held on ……..

Signature of External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to our guide **Mrs. T. Daniya,** Sr. Asst. Professor, Department of Information Technology of whole hearted and invaluable guidance throughout the project work. Without her sustained and sincere effort, this project work would not have taken this shape. He encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the Department **Dr. Ajit Kumar Rout**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr. C.L.V.R.S.V. Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank the project coordinator **Mr. Ch. Anil Kumar**, all the faculty members and the non-teaching staff of the Department of Information Technology for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

ROUTHU SATHISH	**19341A1296**
THULLURU PREM CHAND	**19341A12B7**
SAI RAM PRUDHVI GUMMALURI	**19341A1297**
MARRIPUDI VYAS KANMANI	**19341A1268**

ABSTRACT

Agriculture plays a critical role to Indian global economy and contributes a major part to GDP. The yield of a crop is mainly determined by the climatic conditions like temperature, rainfall, soil conditions, and fertilizers. In this project a new technique for smart farming where different machine learning algorithms such as KNN Classifier, SVM, Ensembled learning techniques are applied to predict the more profitable crop and suitable fertilizers. Ensemble learning techniques such as Random Forest, XG Boost works by building a strong learner on many weak learners and they will perform far more better in this prediction problems. Even though we cultivate a best crop and provide a suitable fertilizer some times crops may get damaged by their diseases. It is very difficult to identify the diseases in the paddy field. But it can be achieved by applying deep learning techniques with CNN and Transfer learning. If we can detect the disease we can suggest solution according severity of disease by a manual data dictionary. Convolutional Networks performs information extraction from images by applying several filters through convolutions and Transfer learning provides pre-trained models built upon large datasets and applies the knowledge acquired from those datasets to solve some other problem. In simple words it is the transfer of knowledge from one experience to solve other problem. Several transfer learning architectures include Efficient Net, ResNet, VGG16 Etc. A web application is made up by using Html, CSS, JavaScript and Flask to deploy in a cloud server for all these models to be used in general use case.

Keywords: Soil, Fertilizer, Paddy Fields, Machine Learning Algorithms, Deep Learning, Transfer Learning.

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVATIONS

AI	**:**	ARTIFICIAL INTELLIGENCE
API	**:**	APPLICATION USER INTERFACE
CIFAR	**:**	CANADIAN INSTITUTE FOR ADVANCE RESEARCH
CNN	**:**	CONVOLUTION NEURAL NETWORK
CPU	**:**	CENTRAL PROCESSING UNIT
DAG	**:**	DIRECTED ACYCLIC GRAPH
DCNN	**:**	DEEP CONVOLUTION NEURAL NETWORK
DL	**:**	DEEP LEARNING
DFD	**:**	DATA FLOW DIAGRAM
DNFN	**:**	DEEP NEURO-FUZZY NETWORK
DNN	**:**	DEEP NEURAL NETWORK
DRNN	**:**	DEEP RECURRENT NEURAL NETWORK
EDA	**:**	EXPLORATORY DATA ANALYTICS
EWMA	**:**	EXPONENTIALLY WEIGHTED MOVING AVERAGE
FC	**:**	FULLY CONNECTED
GBM	**:**	GRADIENT BOOSTING MACHINE
GLCM	**:**	GRAY LEVEL CO-OCCURANCE MATRIX
GPU	**:**	GRAPHICAL PROCESSING UNIT
HCL	**:**	HARDWARE COMPATABILITY LIST
HTML	**:**	HYPER TEXT MARKUP LANGUAGE
JSON	**:**	JAVASCRIPT OBJECT NOTATION
KNN	**:**	K – NEAREST NEIGHBOUR
MAE	**:**	MEAN ABSOLUTE ERROR
MCC	**:**	MATHHEWS CORRELATION COEFFICENT
ML	**:**	MACHINE LEARNING
MSE	**:**	MEAN SQUARED ERROR
OCT	**:**	OVERALL COMMUNICATION THROUGHPUT
OS	**:**	OPERATING SYSTEM
PSR	**:**	POWER SPLITTING RELAY
RMSE	**:**	ROOT MEAN SQAURE ERROR
RCNN	**:**	RECURRENT CONVOLUTIONAL NEURAL NETWORK
ROI	**:**	REGION OF INTEREST

RNN : RECURRENT NEURAL NETWORK

RELU : RECTIFIED LINEAR ACTIVATION UNIT

ROA : RIDER OPTIMISATION ALGORITHM

RSW : RECURSIVE SMITH WATERMAN ALGORITHM

SDK : SOFTWARE DEVELOPMENT KIT

SIR : SIGNAL TO INTERFERENCE RATIO

SMO : SEQUENTIAL MINIMAL OPTIMISATION

SMOTE : SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE

SNR : SIGNAL TO NOISE RATIO

SVM : SUPPORT VECTOR MACHINE

TNR : TRUE NEGATIVE RATE

TPR : TRUE POSITIVE RATE

TPU : TENSOR PROCESSING UNIT

TRS : TRANSMISSION SUCCESS RATE

TSR : TIME SWITCHING RELAY

USDA : UNITED STATES DEPARTMENT OF AGRICULTURE

WWO : WATER WAVE OPTIMISATION

WSGI : WEB SERVER GATEWAY INTERFACE

YOLO : YOU ONLY LOOK ONCE

Chapter – 1

INTRODUCTION

1. INTRODUCTION

Smart farming is a new concept that makes agriculture more efficient and effective by using advanced information technologies. The latest advancements in connectivity, automation, and artificial intelligence enable farmers better to monitor all procedures and apply precise treatments determined by machines with superhuman accuracy. Farmers, data scientists and, engineers continue to work on techniques that allow optimizing the human labour required in farming. With valuable information resources improving day by day, smart farming turns into a learning system and becomes even smarter. Deep learning is a type of machine learning method, using artificial neural network principles. The main feature by which deep learning networks are distinguished from neural networks is their depth and that feature makes them capable of discovering latent structures within unlabelled, unstructured data.

Deep learning networks that do not need human intervention while performing automatic feature extraction have a significant advantage over previous algorithms. However, real-time monitoring of agricultural activities is not enough to make agriculture smart. Smart agriculture should follow the cycle of observation, diagnosis, decision, and action. In this continuously repeating cycle, data should be collected and used quickly to make changes that optimize the farming process. During the observation phase, data can be obtained and recorded using sensors capturing features from natural resources like crops, livestock, atmosphere, soils, water, and biodiversity. During the diagnostic phase, the soil and environmental conditions are transferred to platform based on predefined decision models that determine the state of the object under investigation.

During the decision phase, the components based on machine learning techniques determine whether an action is required. During the action phase, the end-user evaluates the situation and applies the action. Then the cycle starts all over again. In the early days of artificial intelligence, it was discovered that mentally challenging problems for humans were simple for computers as long as they could be described as a list of mathematical and logical rules. As the field of artificial intelligence expands and evolves, to benefit from the experience, to recognize sound and image, and to make intuitive decisions became the focuses of research.

Machine learning, which is a sub-branch of artificial intelligence, uses a self-learning approach to derive meaning from presented data. Instead of manually creating rules by analysing large amounts of data, machine learning gradually improves prediction performance by capturing information in the data. This approach provides a more effective solution that can make evidence-based decisions. Machine learning, to extract meaningful relationships from data, uses learning rules such as supervised learning, unsupervised learning, reinforced learning, and hybrid learning.

Deep learning is a type of machine learning that uses artificial neural network principles. Deep networks are distinguished from neural networks by their depth. Before the big-data age, most machine learning techniques have been used in shallow architecture. These architectures generally consist of up to one or two layers containing nonlinear transformations. Shallow architectures are effective in solving well-structured problems, but they are inadequate for more complex real-world data applications such as images, human speech, natural voice, and language. With deep learning, it became possible to process these data.

1.1 Smart Farming Techniques

1.1.1 Plant classification

Some techniques have been used to classify plants for some given conditions and they have applied them to farm the fields with those plants to get better results.

1.1.2 Disease detection

This technique involves the identification of disease that affected the crop by seeing an image of the plant or their leaves. However it mainly consists of CNN and deep learning architectures to classify the diseased images. It give the type of disease that occurred among the available classes of diseases.

1.1.3 Yield estimation

To know the estimated production rate of crop that is being farmed this regression technique will helps to predict the approximate value by using machine learning and deep learning technique of regression.

1.1.4 Pesticides recommendation

Pesticides help to eliminate the pest that damages the crops in agriculture. We have to know the appropriate pesticide for the pest that observed in the crop. Recommendation of the pesticides will help the farmers to make decision on usage of pesticides.

1.1.5 Weed detection

Weeds compete with crops for sunlight, water, nutrients, and space. In addition, they harbour insects and pathogens, which attack crop plants. We need to detect and identify these weeds to use suitable solution to prevent weed growth in the fields. Weeds can be detected by their images using deep learning techniques. The weed detection process has the three major steps which are segmentation, feature extraction and classification. The classification techniques can classify the weed and non weed detection portion from the image.

1.2 Machine Learning

Machine learning is an application of AI that enables systems to learn and improve from experience without being explicitly programmed. Machine learning focuses on developing computer programs that can access data and use it to learn for themselves. Generally Machine Learning process begins with observations or data, such as examples, direct experience or instruction. It looks for patterns in data so it can later make inferences based on the examples provided. The primary aim of ML is to allow computers to learn autonomously without human intervention or assistance and adjust actions accordingly.

The term "Machine Learning" was coined by Arthur Samuel, a computer scientist at IBM and a pioneer in AI and computer gaming. Samuel designed a computer program for playing checkers. The more the program played, the more it learned from experience, using algorithms to make predictions. ML has proven valuable because it can solve problems at a speed and scale that cannot be duplicated by the human mind alone. With massive amounts of computational ability behind a single task or multiple specific tasks, machines can be trained to identify patterns in and relationships between input data and automate routine processes.

Machine learning is not science fiction. It is already widely used by businesses across all sectors to advance innovation and increase process efficiency. In 2021, 41% of companies accelerated their rollout of AI as a result of the pandemic. These newcomers are joining the 31% of companies that already have AI in production or are actively piloting AI technologies. Some of the sectors making use of ML are

- Data security
- Finance
- Healthcare
- Fraud detection
- Retail

Machine learning offers clear benefits for AI technologies. But which machine learning approach is right for your organization? There are many to ML training methods to choose from including:

- Supervised learning
- Unsupervised learning
- Reinforcement Learning

Supervised machine learning algorithms apply what has been learned in the past to new data using labelled examples to predict future events. By analysing a known training dataset, the learning algorithm produces an inferred function to predict output values. Unsupervised machine learning algorithms are used

when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. Reinforcement machine learning algorithms are a learning method that interacts with its environment by producing actions and discovering errors or rewards. The most relevant characteristics of reinforcement learning are trial and error search and delayed reward. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context to maximize its performance.

1.3 KNN Classifiers

K-nearest neighbours (KNN) is a supervised machine learning algorithm that can be used to solve both classification and regression tasks. We can see KNN as an algorithm that comes from real life. People tend to be effected by the people around them. Our behaviour is guided by the friends we grew up with. Our parents also shape our personality in some ways. If you grow up with people who love sports, it is highly likely that you will end up loving sports. There are ofcourse exceptions. KNN works similarly. The value or category of a new data point is determined by the data points around it.

For Example if you have one very close friend and spend most of your time with him/her, you will end up sharing similar interests and enjoying same things. That is KNN with k=1. Else If you always hang out with a group of 5, each one in the group has an effect on your behaviour and you will end up being the average of 5. That is KNN with k=5. KNN classifier determines the class of a data point by majority voting principle. If k is set to 5, the classes of 5 closest points are checked. Prediction is done according to the majority class. Similarly, KNN regression takes the mean value of 5 closest points.

Let us assume a case where we observe people who are close but how data points are determined to be close? The distance between data points is measured. There are many methods to measure the distance. Euclidean distance (Minkowski distance) is one of most commonly used distance measurement. It is calculated using the square of the difference between x and y coordinates of the two points A(X1,Y1) & B(X2,Y2) in 2 dimensional space.

$$\textbf{Euclidean Distance of points A,B} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

1.4 Ensemble Learning

Ensemble models in machine learning operate on a similar idea. They combine the decisions from multiple models to improve the overall performance. Ensemble Learning Techniques in Machine Learning, Machine learning models suffer bias and/or variance. Bias is the difference between the predicted value and actual value by the model. Bias is introduced when the model doesn't consider the variation of data and creates a simple model.

The simple model doesn't follow the patterns of data, and hence the model gives errors in predicting training as well as testing data i.e. the model with high bias and high variance. When the model follows even random quirks of data, as pattern of data, then the model might do very well on training dataset i.e. it gives low bias, but it fails on test data and gives high variance. Therefore, to improve the accuracy (estimate) of the model, ensemble learning methods are developed. Ensemble is a machine learning concept, in which several models are trained using machine learning algorithms.

It combines low performing classifiers (also called as weak learners or base learner) and combine individual model prediction for the final prediction. On the basis of type of base learners, ensemble methods can be categorized as homogeneous and heterogeneous ensemble methods. If base learners are same, then it is a homogeneous ensemble method. If base learners are different then it is a heterogeneous ensemble method.

1.4.1 Ensemble Learning Methods

Ensemble techniques are classified into three types:

1. Bagging
2. Boosting
3. Stacking

1.4.2 Bagging

Bagging is a parallel method, which means several weak learners learn the data pattern independently and simultaneously from the divided modules of whole Training data. The output of each weak learner is averaged to generate final output of the model. So each weak learner has equal weight in the final output. It is also known as Bootstrapped Aggregation technique which is used to reduce the variance or variability in the predictions. However, it does not help to reduce bias of the model.

Examples of Bagging algorithm are Random forest, Bagging meta-estimator etc.

1.4.3 Boosting

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and then trained sequentially—that is, each model tries to compensate for the weaknesses of its predecessor. With each iteration, the weak rules from each individual classifier are combined to form one, strong prediction rule. Simply we can tell Boosting is an algorithm that helps in reducing variance and bias in a machine learning and can improve model predictions for learning algorithms.

Examples of Boosting algorithms are AdaBoost, XGBoost, CatBoost etc.

1.5 Random Forest

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

1.5.1 Working of Random Forest Algorithm

- In Random forest n number of random records are taken from the data set having k number of records.
- Individual decision trees are constructed for each sample.
- Each decision tree will generate an output.
- Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

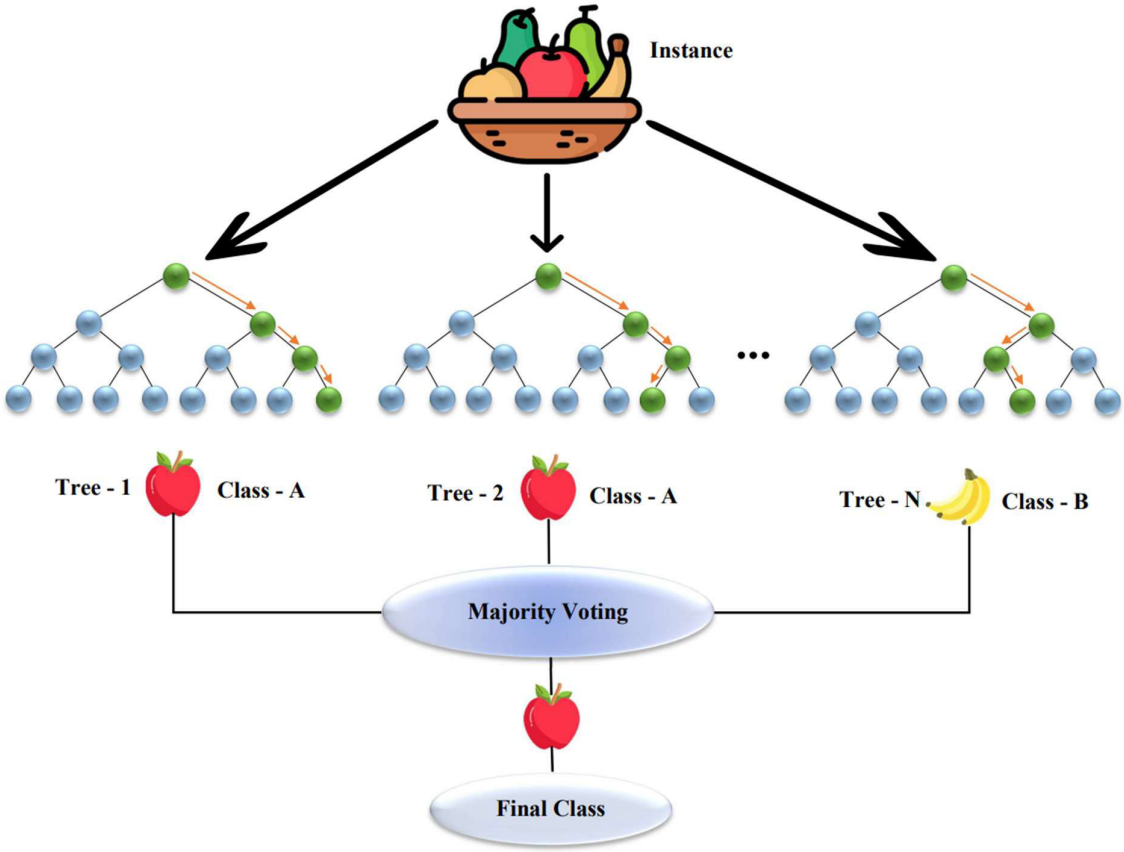


Figure 1.1 : Random Forest Workflow

For example: Consider the fruit basket as the data as shown in the Figure 1.1 above. Now n number of samples are taken from the fruit basket and an individual decision tree is constructed for each sample. Each

decision tree will generate an output as shown in the figure. The final output is considered based on majority voting. In the above figure you can see that the majority decision tree gives output as an apple when compared to a banana, so the final output is taken as an apple.

1.5.2 Important Features of Random Forest

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

1.6 XGBoost

XGBoost stands for "Extreme Gradient Boosting". XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. XGBoost is an extension to gradient boosted decision trees (GBM) and specially designed to improve speed and performance. It implements Machine Learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting to solve many data science problems in a fast and accurate way.

1.6.1 Features of XGBoost

- **Regularization -** helps to reduce overfitting.
- **Parallel Processing -** implements parallel processing and is blazingly faster as compared to GBM.
- **High Flexibility** - allows users to define custom optimization objectives and evaluation criteria.
- **Handling Missing Values -** has an in-built routine to handle missing values.
- **Tree Pruning -** A GBM would stop splitting a node when it encounters a negative loss in the split. XGBoost make splits upto the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.
- **Built-in Cross-Validation -** XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run.

- **Continue on Existing Model -** User can start training an XGBoost model from its last iteration of previous run.

1.6.2 XGBoost System Features

- **Parallelization** of tree construction using all of your CPU cores during training.
- **Distributed Computing** for training very large models using a cluster of machines.
- **Out-of-Core Computing** for very large datasets that don't fit into memory.
- **Cache Optimization** of data structures and algorithm to make best use of hardware.

1.7 Deep Learning for Image Classification

Image classification is the task of categorizing and assigning labels to groups of pixels or vectors within an image dependent on particular rules. The categorization law can be applied through one or multiple spectral or textural characterizations. Image classification techniques are mainly divided into two categories: Supervised and unsupervised image classification techniques.

Unsupervised classification technique is a fully automated method that does not leverage training data. This means machine learning algorithms are used to analyze and cluster unlabeled datasets by discovering hidden patterns or data groups without the need for human intervention. K-means is one of the unsupervised image classification algorithm that groups objects into k groups based on their characteristics. K-means clustering is one of the simplest and very popular unsupervised machine learning algorithms.

Supervised image classification methods use previously classified reference samples (the ground truth) in order to train the classifier and subsequently classify new, unknown data. Supervised classification technique is the process of visually choosing samples of training data within the image and allocating them to pre-chosen categories including vegetation, roads, water resources, and buildings. This is done to create statistical measures to be applied to the overall image.

1.8 CNN

The convolutional neural network (CNN) is a class of deep learning neural networks. CNNs represent a huge breakthrough in image recognition. They're most commonly used to analyse visual imagery and are frequently working behind the scenes in image classification. They can be found at the core of everything from Facebook's photo tagging to self-driving cars. Image classification is the process of taking an input (like a picture) and outputting a class (like "cat") or a probability that the input is a particular class ("there's a 90% probability that this input is a cat").

1.8.1 Layers of CNN

CNN has following layers

- Convolutional layers
- ReLU layers
- Pooling layers
- a Fully connected layer

A classic CNN architecture would look something like this

Input -> Convolution -> ReLU -> Convolution -> ReLU -> Pooling -> ReLU -> Convolution -> ReLU -> Pooling -> Fully Connected

CNNs have an input layer, and output layer, and hidden layers. The hidden layers usually consist of convolutional layers, ReLU layers, pooling layers, and fully connected layers. Convolutional layers apply a convolution operation to the input. This passes the information on to the next layer.

Pooling layer combines the outputs of clusters of neurons into a single neuron in the next layer. Fully connected layers connect every neuron in one layer to every neuron in the next layer. In a convolutional layer, neurons only receive input from a subarea of the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer.

A CNN works by extracting features from images. This eliminates the need for manual feature extraction. The features are not trained! They're learned while the network trains on a set of images. This makes deep learning models extremely accurate for computer vision tasks. CNNs learn feature detection through tens or hundreds of hidden layers. Each layer increases the complexity of the learned features.

1.8.2 General WorkFlow of CNN

- starts with an input image
- applies many different filters to it to create a feature map
- applies a ReLU function to increase non-linearity
- applies a pooling layer to each feature map
- flattens the pooled images into one long vector.
- inputs the vector into a fully connected artificial neural network.
- processes the features through the network. The final fully connected layer provides the "voting" of the classes that we're after.

- trains through forward propagation and backpropagation for many, many epochs. This repeats until we have a well-defined neural network with trained weights and feature detectors.

1.9 Fully Connected Networks

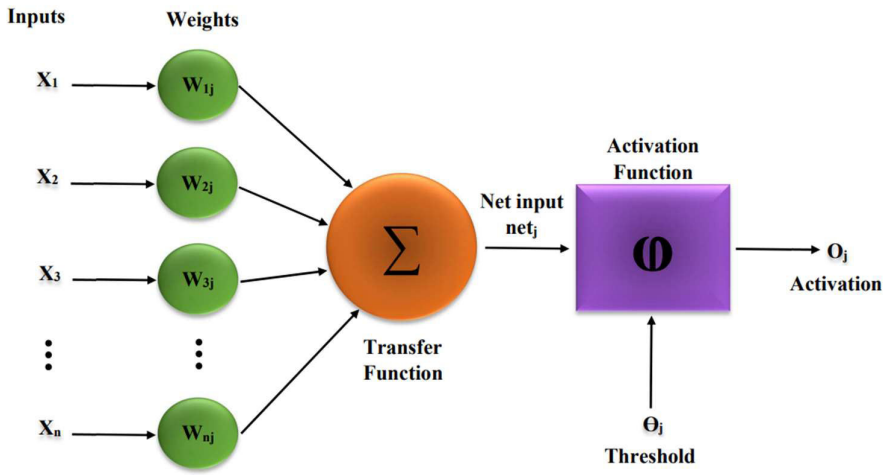


Figure 1.2 : Fully Connected Layer Neural Network

Fully Connected layers in a neural networks are those layers where all the inputs from one layer are connected to every activation unit of the next layer. In most popular machine learning models, the last few layers are full connected layers which compiles the data extracted by previous layers to form the final output. It is the second most time consuming layer second to Convolution Layer.

The left most layer is called the input layer, and the neurons within the layer are called input neurons. A neuron of this layer is of a special kind since it has no input and it only outputs an x_j value of the j^{th} features. The right most or output layer contains the output neurons (just one here). The middle layer is called a hidden layer, since the neurons in this layer are neither inputs nor outputs.

1.9.1 Need of Fully Connected Layers

- **Feature extraction** - In the conventional classification algorithms, like SVMs, we used to extract features from the data to make the classification work. The convolutional layers are serving the same purpose of feature extraction. CNNs capture better representation of data and hence we don't need to do feature engineering.

- **Classification** - After feature extraction we need to classify the data into various classes, this can be done using a fully connected (FC) neural network. In place of fully connected layers, we can also use a conventional classifier like SVM. But we generally end up adding FC layers to make the model end-to-end trainable. The fully connected layers learn a (possibly non-linear) function between the high-level features given as an output from the convolutional layers.

1.10 Transfer Learning :

We, humans, are very perfect in applying the transfer of knowledge between tasks. This means that whenever we encounter a new problem or a task, we recognize it and apply our relevant knowledge from our previous learning experiences. This makes our work easy and fast to finish. For instance, if you know how to ride a bicycle and if you are asked to ride a motorbike which you have never done before. In such a case, our experience with a bicycle will come into play and handle tasks like balancing the bike, steering, etc. This will make things easier compared to a complete beginner. Such leanings are very useful in real life as it makes us more perfect and allows us to earn more experience. Following the same approach, a term was introduced Transfer Learning in the field of machine learning.

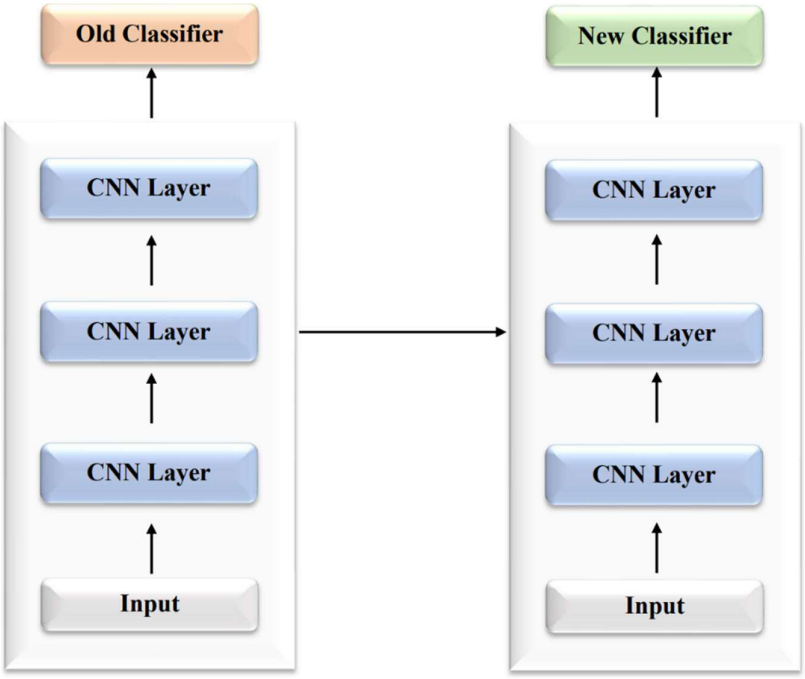


Figure 1.3 : Transfer Learning Workflow

Transfer learning is a technique for predictive modelling on a different yet similar problem that can then be reused partly or wholly to accelerate its training and eventually improve the performance of the model for the problem. With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a network has learned at “task A” to a new “task B.”

The general idea is to use the knowledge a model has learned from a task with a lot of available labelled training data in a new task that doesn't have much data. Instead of starting the learning process from scratch, we start with patterns learned from solving a related task. Transfer learning is mostly used in computer vision and natural language processing tasks like sentiment analysis due to the huge amount of computational power required.

1.10.1 Working of Transfer Learning

In computer vision, for example, neural networks usually try to detect edges in the earlier layers, shapes in the middle layer and some task-specific features in the later layers. In transfer learning, the early and middle layers are used and we only retrain the latter layers. It helps leverage the labeled data of the task it was initially trained on. Let's go back to the example of a model trained for recognizing a backpack on an image, which will be used to identify sunglasses. In the earlier layers, the model has learned to recognize objects, because of that we will only retrain the latter layers so it will learn what separates sunglasses from other objects.

In transfer learning, we try to transfer as much knowledge as possible from the previous task the model was trained on to the new task at hand. This knowledge can be in various forms depending on the problem and the data. For example, it could be how models are composed, which allows us to more easily identify novel objects. Transfer learning has several benefits, but the main advantages are saving training time, better performance of neural networks (in most cases), and not needing a lot of data.

There are some pre-trained machine learning models out there that are quite popular. One of them is the Inception-v3 model, which was trained for the Image Net. Microsoft also offers some pre-trained models through the Microsoft ML python & R Packages. Other quite popular models are ResNet and AlexNet.

1.11 Efficient Net

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients.

There are a variety of convolutional neural networks and all have their own advantage. With the varying architectures, these models have shown an overwhelming performance in a number of computer vision applications. EfficientNet is one of these variants of the Convolutional Neural Network. EfficientNet model was proposed by Mingxing Tan and Quoc V. Le of Google Research, Brain team in their research paper 'EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks'. This paper was presented in the International Conference on Machine Learning, 2019.

These researchers studied the model scaling and identified that carefully balancing the depth, width, and resolution of the network can lead to better performance. Based on this observation, they proposed a new scaling method that uniformly scales all dimensions of depth, width and resolution of the network. They used the neural architecture search to design a new baseline network and scaled it up to obtain a family of deep

learning models, called EfficientNets, which achieve much better accuracy and efficiency as compared to the previous present Convolutional Neural Networks.

For example, if we want to use 2^N times more computational resources, then we can simply increase the network depth by αN, width by β^N, and image size by γ^N, where α,β,γ are constant coefficients determined by a small grid search on the original small model. EfficientNet uses a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way. The compound scaling method is justified by the intuition that if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image.

The base EfficientNet-B0 network is based on the inverted bottleneck residual blocks of MobileNetV2, in addition to squeeze-and-excitation blocks. EfficientNets also transfer well and achieve state-of-the-art accuracy on CIFAR-100 (91.7%), Flowers (98.8%), and 3 other transfer learning datasets, with an order of magnitude fewer parameters. Models such as EfficientNet are particularly useful for using deep learning on the edge, as it reduces compute cost, battery usage, and also training and inference speeds. This kind of model efficiency ultimately enables the use of deep learning on mobile and other edge devices.

1.12 VGG16

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) is an annual event to showcase and challenge computer vision models. In the 2014 ImageNet challenge, Karen Simonyan & Andrew Zisserman from University of Oxford showcased their VGG16 model in the paper titled "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION," which won the 1st and 2nd place in object detection and classification. A convolutional neural network is also known as a ConvNet, which is a kind of artificial neural network. A convolutional neural network has an input layer, an output layer, and various hidden layers.VGG16 is a type of CNN (Convolutional Neural Network) that is considered to be one of the best computer vision models to date.

The creators of this model evaluated the networks and increased the depth using an architecture with very small (3 × 3) convolution filters, which showed a significant improvement on the prior-art configurations. They pushed the depth to 16–19 weight layers making it approx — 138 trainable parameters. VGG16 is object detection and classification algorithm which is able to classify 1000 images of 1000 different categories with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning. The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers.

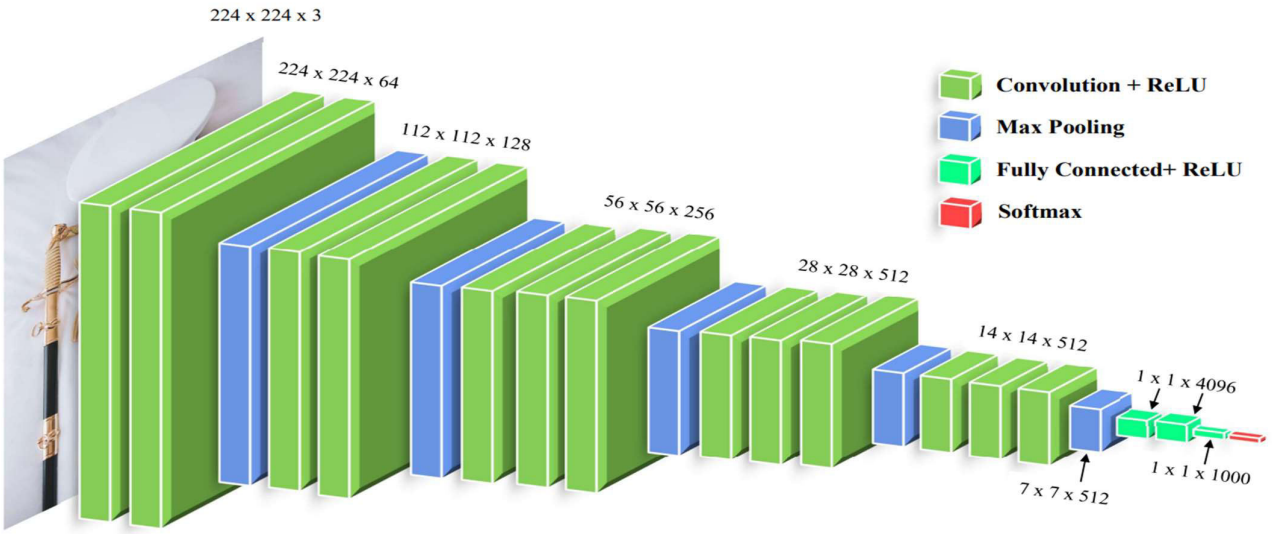


Figure 1.4 : VGG16 Architecture

Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2. VGG16 takes input tensor size as 224, 244 with 3 RGB channel. The convolution and maxpool layers are consistently arranged throughout the whole architecture. Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters. Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

1.12.1 Limitations Of VGG16

- It is very slow to train (the original VGG model was trained on Nvidia Titan GPU for 2-3 weeks).
- The size of VGG-16 trained imageNet weights is *528* MB. So, it takes quite a lot of disk space and bandwidth which makes it inefficient.
- 138 million parameters lead to exploding gradients problem.

1.13 TensorFlow

TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well. sTensorFlow is basically a software library for numerical computation using data flow graphs where nodes in the graph represent mathematical operations

and edges in the graph represent the multidimensional data arrays (called tensors) communicated between them. (Please note that tensor is the central unit of data in TensorFlow).

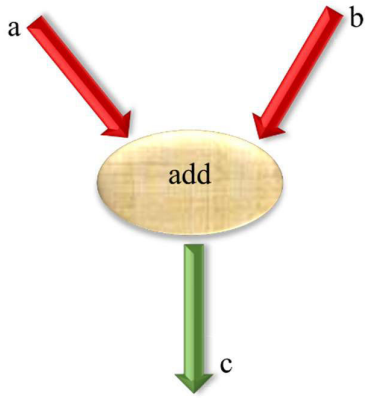


Figure 1.5 : Representation of Tensor

Here, add is a node which represents addition operation. a and b are input tensors and c is the resultant tensor. This flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API!. TensorFlow provides multiple APIs (Application Programming Interfaces). These can be classified into 2 major categories. They are Low level API and High level API. Google's TensorFlow is now the most well-known deep learning package on the planet.

Machine learning is used by Google in all of its products to enhance search, translation, picture captions, and recommendations. Tensorflow enables programmers to construct machine learning applications by utilizing a variety of tools, frameworks, and community resources. Tensorflow works by accepting inputs as a multi-dimensional array called Tensor, TensorFlow allows you to create dataflow graphs and structures to specify how data travels through a graph. Preprocessing the data, building the model, and finally training and estimating the model are the three elements of the Tensorflow structure.

Tensorflow gets its name from the fact that it takes input in the form of a multi-dimensional array, commonly known as tensors. You can create a flowchart of the operations you'd want to run on that input. The input enters at one end, travels through this system of various processes, and emerges as output at the other. The tensor goes in, runs through a set of operations, and then comes out the other side, which is why it's called TensorFlow. TensorFlow is the finest library of them all since it is designed to be user-friendly. The Tensorflow library includes a variety of APIs for creating large-scale deep learning architectures such as CNNs and RNNs.

TensorFlow is a graph-based programming language that allows developers to view the neural network's creation using Tensorboad. This software debugging tool is really useful. Finally, Tensorflow is designed to be used in large-scale deployments. It runs on both the CPU and the GPU. In comparison to the

other deep learning frameworks, Tensorflow also has the most popularity on GitHub. Below are the different Algorithms that can use in Tensorflow. They are

1. Linear regression: tf.estimator.LinearRegressor
2. Classification:tf.estimator.LinearClassifier
3. Deep learning classification: tf.estimator.DNNClassifier
4. Deep learning wipe and deep: tf.estimator.DNNLinearCombinedClassifier
5. Booster tree regression: tf.estimator.BoostedTreesRegressor
6. Boosted tree classification: tf.estimator.BoostedTreesClassifier

1.14 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research. Keras is Simple but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter. Keras is flexible.

It adopts the principle of progressive disclosure of complexity i.e simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned. Keras is also powerful and provides industry-strength performance and scalability. it is used by organizations and companies including NASA, YouTube, or Waymo. Keras is the high-level API of TensorFlow 2. It is an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning.

It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity. Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2 that is we can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

1.15 Flask

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects. It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases. Simply we can tell flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application.

Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages. Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have form a validation support. Instead, Flask supports the extensions to add such functionality to the application. Flask is pretty impressive. Because it has built-in development server and fast debugger, integrated support for unit testing, RESTful request dispatching, Jinja2 templating, support for secure cookies, WSGI 1.0 compliant and Unicode based etc.s

1.16 WEB TECHNOLOGIES

1.16.1 HTML

HTML stands for Hyper Text Markup Language. HTML is the universal markup language for the Web. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between web pages. A markup language is used to define the text document within the tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. HTML was created by Tim Berners-Lee in 1991.

1.16.2 CSS

CSS Stands for Cascading Style Sheets. It is used to apply styles to web pages. Cascading Style Sheets are fondly referred to as CSS. It is used to make web pages presentable. The reason for using this is to simplify the process of making web pages presentable. It allows you to apply styles on web pages. It enables you to do this independently of the HTML that makes up each web page. Styling is an essential property for any website. It increases the standards and overall look of the website that makes it easier for the user to interact with it. So that is why CSS makes a huge important role in webpage creation.

1.16.3 Java Script

JavaScript is the world's most popular lightweight, interpreted compiled programming language. It is also known as scripting language for web pages. It can be used for Client-side as well as Server-side developments. JavaScript can be added to your HTML file in two ways. One way is Internal JavaScript i.e adding JavaScript code directly to our HTML file the <script> tag. Another way is External JavaScript File i.e Create a file with .js extension and paste the JavaScript code inside it. After creating the file, add this file in <script src="file_name.js"> tag inside <head> tag of the HTML file. Java script is used by programmers across the world to create dynamic and interactive web content like applications and browsers.

Chapter – 2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Study of Related Papers

Following research papers are studied in detail to understands the proposed recommendation technique and experimental result for predicting the output.

Here we have gathered several periodicals that have conducted research on our connected work, which is based on the rice plant, and we have separately summarized each work as shown below.

Paper 1 : M. Alencastre-Miranda, R. M. Johnson and H. I. Krebs, "Convolutional Neural Networks and Transfer Learning for Quality Inspection of Different Sugarcane Varieties", in IEEE Transactions on Industrial Informatics, Volume 17, Issue 2, pp. 787–794, February 2021.

In this paper, computer vision and deep learning techniques are used to select and plant healthy billets in order to increase sugarcane crop growth and hectare yield. The data set for the training model was obtained from the USDA Sugarcane Research Farm. The sugarcane dataset is based on three varieties of sugarcane collected from farms and classified by experts as good and damaged billets. When processing large datasets, highly regarded CNN models like AlexNet, VGG-16, GoogLeNet, and ResNet101 are used to achieve better results. Each dataset was randomly divided into three types to compare these models: 60%, 20%, 20% for training, validation, evaluation.

A two-step transfer approach is suggested in this paper to create an automatic system which every farmer can apply to any wide range of sugarcane. In this case, the four architectures selected produced similar results (TNR, TPR, MCC, and proportion of great billets per plant), but ResNet101 and AlexNet produced great outcomes for various range of renewable. ResNet101, on the other hand, demanded an unusually high computational cost. AlexNet appears to be the ideal choice for them, with a two-stage transfer learning process that requires approximately 22 times fewer logs to train up the system for a fresh sugarcane variant. The created model can be applied to any variety of sugar cane.

When it is necessary to harvest and plant a new sugarcane variety, a farmer might swiftly categorise a minimum number of billets of the new variety (about 50 billets) and retrain the system, resulting in improvements of 33 to 80% when compared to without the system. Using the model allows farmers to grow quality billets, thereby increasing sugarcane production rates in the future, but due to the damaged variety of sugarcane billets classified by humans, the initial training dataset may not be accurate. In addition to using high-quality billets, sugarcane quality control also depends on soil and climate.

Paper 2 : Wei Li, Tengfei Zhu, Xiaoyu Li, Jianzhang Dong and Jun Liu, " Recommending Advanced Deep Learning Models for Efficient Insect Pest Detection", in MDPI on Application of Machine Learning in Agriculture , Volume 12, Issue 7, 1065, pp. 1–17, June 2022.

One of the key strategies for increasing crop yield and quality in agriculture is insect pest control, which has a significant impact on crop productivity since it can quickly and precisely identify insect pests. Prior to now, the majority of insect pest identification duties have been dependent on the expertise of agricultural professionals, which is difficult, time-consuming, and unreliable. Various intelligent detecting techniques have appeared in recent years.

For effective insect pest detection, In this paper they have used three cutting-edge Deep Convolutional Neural Network (DCNN) models—Faster-RCNN, Mask-RCNN, and Yolov5. Additionally, they independently created two coco datasets based on the IP102 dataset and the Baidu AI insect detection dataset, and they compared these three with cutting-edge deep learning models on the two coco datasets. The background of the images in the Baidu AI insect detection dataset is simple, in contrast to the background of the IP102 dataset, which is complex. Two different type of algorithms are primarily employed in this paper. These were YoloV5 is a one-stage algorithm, and FasterRCNN and MaskRCNN are two-stage algorithms.

One-stage algorithms extract features from images, locate objects, and categorize them all in one step. These will display in the image (Object coordinates such as bounding boxes) and then categorize that bounding box. Due to the lack of regional design, these are quite fast. In contrast, two-stage algorithms create regions of interest for individual objects in the image (such as masks for the object) and then extract features from those regions to classify the individual objects. The images in the datasets were manually labelled using the image tagging application LabelMe in order to train the Faster-RCNN, Mask-RCNN, and Yolov5 models. LabelMe is extremely lightweight and easy to use, making it a popular choice as an open-source visual annotation tool. You can use it to create annotations for object detection, semantic segmentation, and panoptic segmentation for both images and videos.

On simple datasets with lower complexity, one-stage algorithms worked more quickly and efficiently. On datasets with complex backgrounds, two-stage algorithms outperformed one-stage algorithm. On images from the Baidu AI insect recognition dataset with a plain background, Yolov5 has achieved accuracy of 99%, however on the IP102 dataset, its accuracy is only about 97%. Faster-RCNN and Mask-RCNN achieve 99% accuracy on the IP102 dataset, which has a complicated background and a large number of categories, but only 98% accuracy on the Baidu AI insect detection dataset. Manual labelling takes a lot of time and processing resources to train these sophisticated models.

Paper 3 : D. Elavarasan and P.M.D. Vincent, "Crop Yield Prediction Using Deep Reinforcement Learning Model for Sustainable Agrarian Applications", in IEEE Access, Volume 8, pp. 86886–86901, April 2020.

This paper proposed a Deep Recurrent Q-Network models of Reinforcement learning in order to estimate the yield of a crop with given crop parameters. They collected data from paddy crops in Tamilnadu. Data parameters include cultivable land, paddy growth, and yield obtained and independent features include climatic factors, soil parameters and distinct hydro-chemical characteristics of underground water.

DRQN-based processes provide a satisfactory explanation which incudes a non-linear mapping between agricultural output, Weather conditions, soil, and underground water factors independently. This benefit can reduce the need for previous knowledge and expert dependence when making agricultural yield predictions. Based on the input parameters, the Q-learning network creates a crop yield prediction environment.

The output values of the recurrent neural network are translated into Q-values via a linear layer. The reinforcement learning agent combines a threshold and a set of parametric variables that help forecast crop yield. The agent then obtains an overall score for the actions taken, which is determined by minimizing error and maximizing forecast accuracy. With an accuracy of 93.7%, this model accurately forecasts crop yield while outperforming other models and maintaining the original data distribution.

Paper 4 : K.S Neethu and P. Vijay Ganesh, "Leaf Disease Detection and Selection of Fertilizers using Artificial Neural Network", in IRJET on Computer Science Journal, Volume 4, Issue 6, pp. 1852–1858, June 2017.

This paper included disease identification in lemon and mango plants using image segmentation techniques and the appropriate fertilizer. Several stages are used to identify diseases, such as Image acquisition, Image Pre-processing, Classification by Artificial Neural Network, Fertilizer recommendation by Mapping standard deviation to the fertilizers available.

To figure out the amount of disease infection as in leaf, proposed system divides the input leaf image into diseased and background areas using the K means clustering algorithm, Feature extraction by using Gray Level co-occurrence Matrix (GLCM) along with first order statistical moment's method, Fertilizer recommendation based on disease detected but Segmentation provided some misclassifications. Here, the proposed system simply recommends the chemical fertilizers that were applied. However, the amount of dosage of fertilizer use is not mentioned.

Paper 5 : T. Daniya and Dr. S. Vigneshwari, "Deep Neural Network for Disease Detection in Rice Plant Using the Texture and Deep Features", The Computer Journal, Volume 65, Issue 7, pp. 1812–1825, July 2022.

Images from the input plant are first pre-processed to create them suited about further handling. Pre-processed images have been routed through a Image Segment module, which employs SegNet to segment the pre-processed image. Pre-processing is done to improve the contrast of the image, remove noise and disorders in order to detect plant diseases. The CNN features extract statistical data from each segment. These features are then put together into a feature vector. When using the deep recurrent neural network (Deep RNN), these features are used to detect plant diseases. The pre-processed paddy image is sent to the segment module using SegNet can supply high-dimensional data segmentation and to identify disease regions taking into account each segment.

Deep RNN receives the extracted features for classification and trains on the proposed RSW, which combines ROA, SMO, and WWO. It is proposed to build a classifier known as RSW-based Deep Recurrent Neural Network by adjusting the Deep Recurrent Neural Network training algorithm including the Recursive Smith-Waterman-seq algorithm. The proposed Recursive Smith-Waterman-seq is used to optimize the weight of the classifier. To select the best weights, Recursive Smith-Waterman-seq modifies the Deep Recurrent Neural Network by integrating the Read-While-Write and Sequential Minimal Optimization algorithms.

The goal of the proposed Recursive Smith-Waterman-seq-based Deep Recurrent Neural Network is to identify disorder using features extracted from the source images. The proposed Recursive Smith-Waterman-seq based Deep Recurrent Neural Network produces high results: With Maximum Accuracy, Sensitivity, Specificity (90.5%, 84.9%, 95.2%) respectively. However, the proposed method cannot detect all rice diseases.

Paper 6 : R. Salini, A. Farzana and B. Yamini, "Pesticide Suggestion and Crop Disease Classification using Machine Learning", in IRJET on Computer Science Journal, Volume 11, Issue 4, pp. 27997–27999, April 2021.

In this paper to forecast the diseased area of the leaves, they suggested improving machine learning. The infected area is divided using a colour-based segmentation model defined by them. On sample images, experimental evaluations of the time complexity and the infected area were performed. Images can be processed to identify plant diseases. Image acquisition steps in the disease identification process include image pre-processing, image segmentation, feature extraction, and classification.

Image augmentation s(virtually increase the number of samples in your dataset using data you already have) is possible to achieve by applying geometric transformations, altering the colour, brightness, contrast or

by adding noise. To enable model to adapt to various situations where noisy data could occur, such as during the rainy season. The dataset is gathered from UCI machine learning repository it comprises three leaf diseases. 20% of the images were used for validation and testing, while the remaining 80% were used for training.

Classification Algorithm used : SVM. When model is trained with images that have undergone image augmentation or acquisition, it performs 5% better on the validation set than a model that is trained with images that have not. The model used in this paper is only relevant to three diseases and its usefulness to other diseases is not guaranteed.

Paper 7 : T. Daniya and Dr.S. Vigneshwari, "Exponential Rider-Henry Gas Solubility optimization-based deep learning for rice plant disease detection", in Springer, International Journal Of Information Technology, pp. 1–11, July 2022.

A strong method ExpRHGSO-based Hybrid Deep Learning, a combination of DRN and DFDN is developed for disease identification and leaf classification. DRN stands for Dilated Residual Networks used for image classification and segmentation purposes. Initially, images are retrieved out of a database that combines two data sources to create a single dataset. The ROI extraction is then applied to the input images for pre-processing. Following that, image segmentation is performed using DFC to segment appropriate regions. Statistical, CNN, and texture features are extracted for subsequent processing. To improve the detection and classification processes, data augmentation is used to enhance the dimensionality of the features.

Utilizing rotation, cropping, and zooming, the data is improved. The DNFN classifier is used to carry out the rice leaf detection process. This classifier reduces the problems associated with computational complexity while achieving the ideal result more quickly. The developed ExpRHGSO algorithm, which combines EWMA and RHGSO, is used to train the DNFN model. EWMA stands for Exponentially weighted moving average used to improve simple variance by assigning weights to the periodic returns. The EWMA algorithm detects slight changes in the way the target values are processed since it has the ability to produce accurate result. RHGSO is integrated with the EWMA to produce better results. After the identification of the rice leaf disease, the BLB, Blast, and Brown spot.

DRN is used for rice leaf disease classification. In the DRN training process, the ExpRHGSO method is included. With higher values of 0.916, 0.923, and 0.919, In terms of performance the created ExpRHGSO-based Hybrid Deep Learning technique performed better. On the basis of testing accuracy, sensitivity, and specificity, the model demonstrated excellent performance. When tested against huge datasets with a greater variety of diseases, the model might not keep the same accuracy and other standards.

Paper 8 : F.T. Pinki, N. Khatun and S.M.M. Islam, "Content based paddy leaf disease recognition and remedy prediction using support vector machine," 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–5, December 2017.

In this study, an automated approach is proposed for the detection of 3 main paddy leaf diseases. Depending on the severity of the diseases, pesticides and fertilizers are also suggested. The goal of this study is to provide a reliable and straightforward system for categorizing paddy leaf diseases. To distinguish the damaged part from the paddy leaf image, K-means clustering is used. These disorders are categorised using the visual content's colour, Shape and Texture. SVM classifier recognizes the type of paddy leaf diseases. After identification, a preventative solution is recommended that can assist individuals and organizations involved in agriculture in responding appropriately to these diseases.

Training and testing phases make the experiment's two stages. Different preprocessing processes are used to process the training image during the training phase. The image was then resized to 900 x 700, filtered using the median filter, and the contrast was increased. Kmeans clustering is used for segmentation. The disease-affected section from the three segmented images is picked to extract the feature. The feature vector is constructed and stored in the database after the features have been extracted. Similar to the training phase, the preprocessing, segmentation, and feature extraction of a query image are carried out. A query image's feature vector is created, then submitted to the classifier where SVM categorizes paddy leaf diseases with training dataset.

In this study, it is proposed that the following treatments for Brown spot are effective: Treating diseased plants with pesticides and herbicides such Benzoyl and Iprodione and antibiotics like Validamycin and Polyoxin. Systemic fungicides, such as triazoles and strobilurins, can be used sparingly to reduce leaf blast. To effectively control the illness, apply a fungicide at heading. Foliar spray of Streptocycline and Copper Sulfate for bacterial leaf blight. The overall accuracy is about 92%. Suggested pesticides or fertilizers are divided into 2 categories one is at initial stage and another at severe stage. If we use different techniques then the accuracy may increase.

Paper 9 : Junde Chen, Defu Zhang, Yarser A Nanehkaran and Dele Li, "Detection of rice plant diseases based on deep transfer learning", Journal of the Science of Food and Agriculture, Volume 100, Issue 7, pp. 3246–3256, March 2020.

In the world of agriculture, a quick, automated, cheap, and trustworthy method to identify rice infections is desperately needed. Because severe rice diseases could result in no harvest of grains. Deep learning has proven to perform well in image processing and classification challenges, thus they used this

approach in this research to address the problem. In order to address the vanishing-gradient problem that arises as network depth grows, the deep convolutional neural network architecture known as DenseNet links the result among all layers to the source among all subsequent layers in inside dense block.

The recurrence of a Batch Normalization for a specified number of times is known as a dense block. Each layer's feature-maps are utilized as sources and its own feature-maps are employed as sources for all subsequent layers, minimizing feature loss. The easiest way to enhance the capacity of deep neural networks is to raise their depth or the networks' width. However, as network depth and width have increased, the number of factors has increased as well. This increases the use of computational resources. In order to solve these issues, the Inception used was a module. The module, which consists of a max-pooling layer and convolutional layers of different sizes that are stacked together and utilized to do the dimension reduction, has to be combined using a concatenation filter.

The traditional DenseNet was altered by the addition of an Inception module and a brand-new, fully-connected ReLU layer with 512 neutrons, which was then replaced with a global pooling layer to replace the connection layer entirely. Since it integrated the advantages of both, the DenseNet pre-trained on ImageNet and Inception module was chosen to be utilized in the network. This method outperforms other cutting-edge approaches, and the outermost layer employs the Softmax algorithm to classify images of rice disease. DENSE-INCEP network model is the algorithm we are using to solve the above problem.

The DenseNet pre-trained on ImageNet and Inception module was chosen to be utilized in the network since it combined the benefits of both, and this method outperforms other cutting-edge approaches. In the public dataset, it obtains an average prediction accuracy of at least 94.07%. Even when various diseases were taken into account, the class prediction of photos of rice disease has an average accuracy of 98.63%.

Paper 10 : Senthil Kumar Swami Durai and Mary Divya Shamili, "Smart farming using Machine Learning and Deep Learning techniques", Decision Analytics Journal, Volume 3, 100041, ISSN 2772–6622, pp. 1–30, March 2022.

Every nation is built on agriculture. As a result, it needs to be timely monitored. Farmers have so far used traditional farming methods. These methods took a lot of time and were ineffective because they were not exact. Precision farming includes a variety of techniques such as weather forecasting, soil analysis, crop recommendations, and calculating the necessary dosages of pesticides and fertilizers. Precise Farming gathers data, trains systems, and forecasts outcomes using cutting-edge technologies like Internet Of Things, Data Mining and Analytics, and ML. This research suggests four modules: crop prescription, weed detection, pesticide advice, and farm budget.

In order to find the optimal model for analysis, they used a total of 11 algorithms and In this work, hyperparameter adjustment has been used to maximize the model potential. The model is saved in a pickle file after being determined to be the best crop prediction model so that it may be quickly summoned and utilized to forecast crops. To classify the images in the weed identification module, the Resnet keras model and Deep Residual Networks using skip connection technique were utilized. This allowed DL models to be trained without the problems that disappearing gradients generate. This module had 25 epochs run, with a batch size of 32. It resulted in accuracy of 89%. The pesticide recommendation system is similarly modelled after the herbicide recommendation system based on weeds.

This module had 20 epochs run, with a batch size of 32. The achieved accuracy was 98%. Pesticides cannot be used on any crop in the same way that herbicides are. Both the soil fertility and crop growth could be harmed. Crop names enter by user and predicted insect names are sent to the Random Forest Classifier to ensure that the correct pesticides are recommended. The numerous cost ideas and the total cost of cultivation are estimated in the next module. For this module, eight different datasets that span the years 2010 to 2018 have been loaded. These datasets are combined then loaded. The state and crop columns both lack in numerical values.

Label Encoding is employed 4 to produce numeric data for a given item. The year, crop name encoded, and state name encoded columns are provided to the ML and ensemble regressor models. The R2 score, RMSE, MSE, MAE are calculated for each regressor model. With the highest R2score, the XGBoost regressor is selected as the effective model. The overall purpose of this research is to assist a person in efficiently cultivating crops in order to maximize productivity while minimizing costs. Predicting the overall cost of farming is also aided by this information.

Paper 11 : R. Alfred, J.H. Obit, C.P.Y. Chin, H. Haviluddin and Y. Lim, "Towards Paddy Rice Smart Farming: A Review on Big Data, Machine Learning, and Rice Production Tasks", in IEEE Access, Volume 9, pp. 50358–50380, March 2021.

In this paper, they have conducted a review of the most recent studies on the use of intelligent data processing technologies in agriculture, notably in the production of rice. By examining the applications of machine learning in various scenarios, including smart irrigation for paddy rice, predicting paddy rice yield estimation, monitoring paddy rice growth, monitoring paddy rice disease, assessing the quality of paddy rice, and classifying paddy rice sample, they describe the data captured and elaborate the role of machine learning algorithms in paddy rice smart agriculture. The methodology presented in this study connects each activity outlined in the production and post-production phases of paddy rice production to the data utilized in data modelling and machine learning techniques.

The objective of this study is to support an individual in efficiently cultivating crops in order to attain high yield at a reasonable price. For recommending a crop to the user, ML algorithms like Neural Network, Naive Bayes, Decision Tree, K Nearest Neighbor, regression model, and SVM were employed. Instantly retrieved information on the temperature and humidity is fed into the most basic model, which consists of 10 procedures with hyper parameter adjustment.

To obtain accurate prediction, the Resnet algorithm is used. Since the Resnet method uses layers like the skip layer, identity layer that attempt to make the source image into the output itself, it attempts to achieve more accuracy. Predictions is therefore 100% accurate. Although te CenterNet algorithm is a method for identifying weeds on the ground, Resnet152V2 aims to produce accurate prediction. This study has defined a method for taking into account a picture's features with a relevant insect dataset. Its major objective is to find a key that aids in classifying the classes. This proposed work helps to clarify the significance of a secret in identifying the many insect classes.

Resnet model is thus used in the proposed model to classify data. The suggested model aids in insect detection and also recommends pesticides in this regard. It offers a detailed look at production costs, flat rate, and overall cost. They came to the conclusion, based on the suggested mapping framework, that a successful integration of all three technologies is essential for transforming conventional rice farming techniques into a fresh perspective on intelligence in rice precision agriculture. Finally, this paper also provides a comprehensive overview of all the issues and technological developments related to the use of numerous sources in agriculture.

Paper 12 : S. Mishra, D. Mishra and G.H. Santra, "Applications of Machine Learning Techniques in Agricultural Crop Production: A Review Paper", Indian Journal of Science and Technology, Volume 9, Issue 38, pp. 1–14, October 2016.

The objective of this study is to re-evaluate research regarding the use of ML techniques in crop production. A few techniques, such as artificial neural networks, decision trees, fuzzy information networks, and Bayesian belief networks. Statistical analysis, the k nearest neighbours, Markoff process model, k-means clustering, k nearest neighbours, and SVM were all used in the context of agriculture. To forecast corn yields, nonlinear regression is used. For predicting cotton yields from surveys, the Markoff process approach is used. For estimating the grain yield of maturing rice, regression toward the mean is used. Belief Networks are used to produce future crops.

Crop yield prediction uses neurofuzzy modelling. Second Order Markov Chains is employed for Forecasting of Crop Yields. Factors Influencing the Growth of Cultivated Varieties in Crop Margins Use

Polynomial Regression For seasonal to cross weather forecasting, stochastic and probabilistic forecasting approaches are used. For the purpose of predicting sugar production from populations of samples, k-nearest neighbour algorithm is used. For the purpose of predicting corn and soybean production, ANN are used. Neural Network is employed for Rice Crop Monitoring. Artificial Neural Networks is employed for Forecasting Thailands Rice Export.

Climate-related variables are used in regression to estimate sugarcane production. Decision-tree algorithms are employed for modelling soya productivity. For crop selection, a fuzzy modelling of decision network is used. Crop yield forecasting makes use of statistical techniques. India uses data mining with climate variables to increase jowar crop yield. For the purpose of predicting apple crop yield, fuzzy cognitive map training is used. Models using regression and NN are used to predict agricultural output. To predict crops, KMeans clustering is used. Farm Crop Production Prediction uses Artificial Neural Network Approach. Rotations are used for crop development.

Several techniques, including artificial neural networks, information fuzzy networks, decision trees, regression analysis, and Bayesian belief networks, can be used to improve the system. Agriculture-related applications of time series analysis, Markov chain modelling, k-means clustering, k closest neighbour, and support vector machines were discussed. In order to find a meaningful association between these novel techniques and the numerous factors available in the data base, this study aims to evaluate them.

Paper 13 : J. Sustain, DR. D. Sivaganesan, "Performance Estimation of Sustainable Smart Farming with Blockchain Technology", Wireless Systems, Volume 03, ISSN 2582–3167, Issue 02, pp. 97–106, June 2021.

This paper proposes a blockchain-enabled wireless network-based sustainable smart farming system, whose performance is improved by maximization of SIR or SNR values for selection of an ideal relay on an end-to-end basis. When choosing the optimal relay performance with and without interference, the overall communication throughput (OCT), power splitting relaying (PSR), time switching relaying (TSR), and transmission success rate (TRS) are also determined. TSR control the flow of electrical power and can be used to control power to many different types of electrical loads.

Numerical simulations confirm the theoretical values of accuracy. It is numerical simulation is a calculation that is run on a computer following a program that implements a mathematical model for a physical system. For the purpose of evaluating the performance of the proposed wireless networks for sustainable smart farming equipped with blockchain technology, the OCT, PSR, TSR, and TRS values are estimated. . The performance of OCT, PSR, TSR and TRS increases with the increase in the number of potential relay nodes.

Chapter – 3

REQUIREMENT SPECIFICATION

3 . REQUIREMENT SPECIFICATION

3.1 Software Requirements

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

3.1.1 Operating System

Windows is a graphical operating system developed by Microsoft. It allows users to view and store files, run the software, play games, watch videos, and provides a way to connect to the internet. It was released for both home computing and professional works.

MacOS is the computer operating system (OS) for Apple desktops and laptops. It is a proprietary graphical OS that powers every Mac. OSes interact with a computer's hardware, allocating the resources necessary to complete tasks given to it, for example, running an application. OSes allocate resources including memory, processing power and file storage.

Linux is an operating system. In fact, one of the most popular platforms on the planet, Android, is powered by the Linux operating system. An operating system is software that manages all of the hardware resources associated with your desktop or laptop. To put it simply, the operating system manages the communication between your software and your hardware. Without the operating system (OS), the software wouldn't function.

3.1.2 SDK

SDK stands for software development kit or devkit for short. It's a set of software tools and programs used by developers to create applications for specific platforms. SDK tools will include a range of things, including libraries, documentation, code samples, processes, and guides those developers can use and integrate into their own apps. SDKs are designed to be used for specific platforms or programming languages.

Some of SDK's used in this project are stated below:

1. TensorFlow
2. Flask
3. Numpy
4. Scikit-Learn etc.

3.2 Hardware Requirements :

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements. The Hardware Interfaces Required are:

1. Ram: Minimum 8GB or higher
2. GPU: 4GB dedicated
3. SSD: 128GB
4. Processor : Intel i5 10th Gen or Ryzen 5 with Octa core.

3.3 Non-Functional Requirements:

Non-Functional Requirements are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non- Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more. Non-Functional Requirements address vital issues of quality for software system. It includes below things: Capacity, Availability and Performance etc.

3.4 Python Libraries To be installed:

The following libraries with the specific versions type must be installed for this project to function. These can be installed using command “pip install library_name==version"

- numpy==1.23.3
- Pillow==9.2.0
- scikit-learn==1.1.2
- scipy==1.9.2
- sklearn==0.0
- tensorboard==2.10.0
- tensorflow==2.10.0
- tensorflow-estimator==2.10.0
- tensorflow-gpu==2.10.0
- tensorflow-hub==0.12.0

Chapter – 4

METHODOLOGY

4. METHODOLOGY

4.1 Crop and Fertilizer Recommendation :

4.1.1 Datasets Collection and Partition of dataset

Datasets have been collected from the Kaggle website. The crops dataset consists of totally 8 features in which one is target variable. Remaining 7 are independent variables. There are totally 2200 samples in the dataset. The dataset consists of 7 independent features N (Nitrogen), P (Phosphorous), K (Potassium), Humidity, pH Value and Rainfall the label describes the crop which gives more yield with given features.

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

Figure 4.1 : Crop Recommendation Dataset

Fertilizers dataset consists of 100 samples with 8 independent features and Fertilizer as a target feature. The dataset consists of 8 independent features Temperature, Humidity, Moisture, Soil Type, Crop Type, N (Nitrogen), P (Phosphorous) and K (Potassium), The fertilizer name is the target feature that has to be predicted.

	Temparature	Humidity	Moisture	Soil Type	Crop Type	Nitrogen	Potassium	Phosphorous	Fertilizer Name
0	26	52	38	Sandy	Maize	37	0	0	Urea
1	29	52	45	Loamy	Sugarcane	12	0	36	DAP
2	34	65	62	Black	Cotton	7	9	30	14-35-14
3	32	62	34	Red	Tobacco	22	0	20	28-28
4	28	54	46	Clayey	Paddy	35	0	0	Urea

Figure 4.2 : Fertilizer Recommendation Dataset

These datasets are csv files. They have been loaded by using python pandas library to process them. By using pandas library we can load and manipulate the datasets by using different functions available in the library. datasets will be analysed by visualization process using matplotlib and seaborn. They provide several

methods and pre-built functions to plot different graphs. Datasets are split into train sets and test sets for evaluation of models on unseen data. This includes shuffling of data and random splitting into subsets as train set and test set.

4.1.2 K-Fold Cross Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    a) Take the group as a test data set
    b) Take the remaining groups as a training data set
    c) Fit a model on the training set and evaluate it on the test set
    d) Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores.

4.1.3 Pre-Processing Datasets

Exploratory Data Analysis

It involves initial investigation of the data before creating any kind of model. There are a lot of different techniques that can be employed while doing EDA. It doesn't have a set of rules that needs to be followed. Rather it is more of a philosophy, more of an art than science. The sole intention of carrying out EDA is to gain insight about the data and its underlying structure. It helps in identifying the variables which will have greater impact on the model. It is mostly used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

Univariate Analysis

Univariate data are the ones which consist of only one variable. Univariate data analysis are pretty straightforward as we are dealing with only one variable. The analysis that we do in case of univariate data analysis doesn't have to do anything with the relationships between variables. The main purpose is to describe the data and find patterns that are present. Plots such as distplot, histogram and boxplot are used to know the insights from the features.

Bivariate Analysis

This type of data involves two different variables. The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship among the two variables. Plots such as Scatter plot , line plot, boxplot, bar chart etc.

Multivariate Analysis

Data which involves 3 or more variables are termed as Multivariate data. These are similar to bivariate but contains more than one dependent variable.

Label Encoding

It refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

Min-Max Normalization: This technique re-scales a feature or observation value with distribution value between 0 and 1.

$$X_{new} = (X_i - X_{min}) \ / \max(x) - X_{min}$$

Standardization: It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1.

$$X_{new} = (X_i - X_{mean}) \ / \text{Standard Deviation}$$

4.1.4 Handling Imbalanced Datasets

Synthetic Minority Oversampling Technique (SMOTE)

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short. After over sampling the fertilizers dataset is balanced and all target classes have equal no. of samples in the dataset.

4.1.5 Model Building

Installation

Scikit-learn requires:

NumPy, SciPy as its dependencies.

Before installing scikit-learn, ensure that you have NumPy and SciPy installed. Once we have a working installation of NumPy and SciPy, the easiest way to install scikit-learn is using pip:

pip install -U scikit-learn

TensorFlow installation :

pip install -U tensorflow

Steps to build a machine learning model

1. Import required libraries and classes.
2. Create an object of model class (ex : KNN Classifier)
3. Fix the parameters of the models
4. Fit the training data.
5. Test the model by test data

4.1.6 Hyper Parameter Tuning

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. The two best strategies for Hyperparameter tuning are:

GridSearchCV

In this approach, the machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

RandomizedSearchCV

RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings. It moves within the grid in a random fashion to find the best set of hyperparameters. This approach reduces unnecessary computation.

Hyperparameters for KNN classifier model is the value of K and hyper parameters of the Random forest model is no. of trees to be built, depth of the tree, no. of splits etc. XGBoost model includes hyperparameters such as no. of iterations to build the strong learner.

4.1.7 Model Saving

We have to save the build models in the form of pickle objects to deploy them for real time use. Pickle library provides saving of python objects into a specific format and we load them by using the same pickle library in python.

4.2 Paddy Disease Classification

4.2.1 Collection of Images of paddy plants

Paddy leaf disease dataset consists of totally 10407 images of 10 different disease categories. The dataset consists of totally 10 types of disease images of the paddy field. The disease classes are : Bacterial Leaf Blight , Bacterial Leaf Streak, Bacterial Panicle Blight, Blast, Brown Spot, Dead heart, Downy mildew, Hispa, Normal, Tungro. Paddy leaf disease images are partitioned into 80%, 10% and 10% for training and validation and testing.

4.2.2 Pre-processing Images

Data Augmentation for Paddy Images Dataset

The data augmentation helps in visual transformations. The transformation that is mainly focused on image classification are flipping, colour modifications, cropping, rotation, geometric transformation, padding, re-scaling, zooming, Gray scaling, darkening and brightening, random erasing etc. When you don't have a large image dataset, it's a good practice to artificially introduce sample diversity by applying random, yet realistic, transformations to the training images, such as rotation and horizontal flipping. This helps expose the model to different aspects of the training data and reduce overfitting.

Rescaling and Resizing Images

Paddy images in the dataset are having different dimensions such as (448 x 448) or (600 x 680) to make all the images of same size we have resized the images to (224 x 224) as input data to the Neural Network. The images are rescaled by dividing with 255 to make all values to be lies in the range of 0 and 1.

4.2.3 TensorFlow data input pipeline

The tf.data API enables you to build complex input pipelines from simple, reusable pieces. For example, the pipeline for an image model might aggregate data from files in a distributed file system, apply random perturbations to each image, and merge randomly selected images into a batch for training. The pipeline for a text model might involve extracting symbols from raw text data, converting them to embedding identifiers with a lookup table, and batching together sequences of different lengths.

The tf.data API makes it possible to handle large amounts of data, read from different data formats, and perform complex transformations. The tf.data API introduces a tf.data.Dataset abstraction that represents a sequence of elements, in which each element consists of one or more components. For example, in an image pipeline, an element might be a single training example, with a pair of tensor components representing the image and its label.

There are two distinct ways to create a dataset:

- A data source constructs a Dataset from data stored in memory or in one or more files.
- A data transformation constructs a dataset from one or more tf.data.Dataset objects.

4.2.4 Building a Custom CNN model

Deep Learning model by TensorFlow

TensorFlow library provides different pre-defined models and layers to train neural networks. There are two types of models that we can build with TensorFlow keras layers.

1. **Sequential Models**

   A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

2. **Functional Models**

   The Keras functional API is a way to create models that are more flexible than the tf.keras. Sequential API. The functional API can handle models with non-linear topology, shared layers, and even multiple inputs or outputs. The main idea is that a deep learning model is usually a directed acyclic graph (DAG) of layers. So the functional API is a way to build graphs of layers.

4.2.5 Steps to build a Deep Learning Model

1. Import TensorFlow and the other required Python modules.
2. Load the dataset (ex : TensorFlow dataset)
3. Define a model using Keras Layers (Sequential / Functional)

4. Compile the model (Parameters : Optimizers, No. of epochs etc.)
5. Fit the training data.
6. Save and Test the model.

Compilation of model

When we compile the Keras model, it uses the backend numerical libraries such as TensorFlow or Theano. Whatever backend you are using automatically chooses the best way to represent the network on your hardware such as CPU, GPU, or TPU. When we are compiling the model we must specify some additional parameters to better evaluate the model and to find the best set of weights to map inputs to outputs.

Loss Function

One must specify the loss function to evaluate the set of weights on which model will be mapped. we will use cross-entropy as a loss function which is actually known as binary cross-entropy used for binary classification.

Optimizer

Second is the optimizer to optimize the loss. we will use "adam" which is a popular version of gradient descent and gives the best result in most problems. DL models train on the dataset by a certain no. of epochs such that the error rate will be reduced by end of training. Deep Learning models uses the back propagation technique to adjust the wights of the connected edges between nodes in the neural network. Learning rate is a parameter that specifies the amount of change for each epoch to the model.

Fitting Model

After successful compilation of the model, we are ready to fit data to the model and start training the neural network. Along with providing data to model, we need to define a number of epochs and batch size over which training occurs.

Epoch - one single pass through all the rows in the training dataset

Batch size - number of samples considered by the model before updating the weights. One epoch can be comprised of more than one batch. These parameters are finally decided after the heat and trial method.

4.2.6 Transfer Learning Models

TensorFlow models provide a module called applications which contain different pre-trained models such as VGG16, Mobilenet, ResNet etc. we can import those models and download those trained weights of those corresponding trained networks. To acquire knowledge of those trained models we need to import them

from Tensorflow.keras.applications module. We need to re-build the top layer such that the output layer will be modified according to our problem statement.

4.2.7 TensorFlow.keras.applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning. Weights are downloaded automatically when instantiating a model.

A transfer learning model model can instantiated by following arguments

a. **include_top** - whether to include the 3 fully-connected layers at the top of the network.
b. **Weights** - one of None (random initialization) or "imagenet" (pre-training on ImageNet).
c. **input_tensor** - optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
d. **inputs_shape** - optional shape tuple, only to be specified if include_top is False (otherwise the input shape has to be (224, 224, 3) (with tf dimension ordering).

Then it will return a keras model instance.

Efficient Net

```
Import tensorflow.keras

BaseModel = keras.applications.EfficientNetB0( include_top=True,
                    weights='imagenet', input_tensor=None, input_shape=(224,224,3) )
```

Now this base model acts as a TensorFlow model we can add this entire model as a layer to the neural network. The weights are obtained from the model that is trained on ImageNet dataset. Output layer can be rebuilt for our custom labels available in the training dataset.

VGG16

```
BaseModel = keras.applications.vgg16.VGG16(include_top=False, weights='imagenet', input_tensor=None, input_shape=None)
```

Above code snippet imports the VGG16 model, with weights pre-trained on ImageNet. We can rebuild the top layer according to our problem statement. These pre-trained models will not adjust their weights while training however it will require more time and computational power as they include complex networks with many no. of layers of convolutions and fully connected.

4.2.8 Training Models on our dataset

After importing these pre-trained models we fit our train dataset to obtain results with their previous knowledge on imagenet weights. At the first epoch itself they give an accuracy of more than 60%. Further more because of output layer adjustment we can get more accurate results while training.

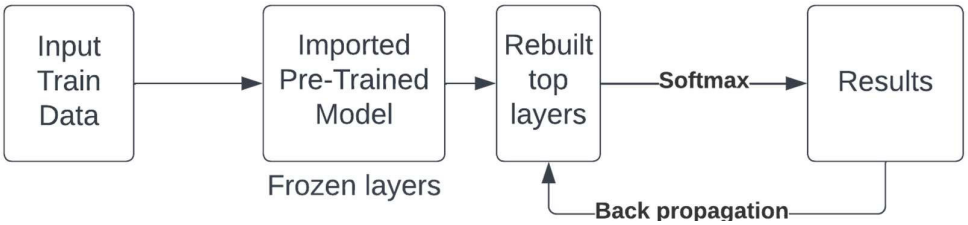


Figure 4.3 : Flow Chart of Training the Model

4.2.9 Fine Tuning Models

The imported pre-trained models contain many layers which makes it more complex to train. But we can unfreeze some layers of that model such that their weights can be adjusted according to our train dataset. This process is called fine-tuning. But this will require more time to train compared to previous unfrozen model.

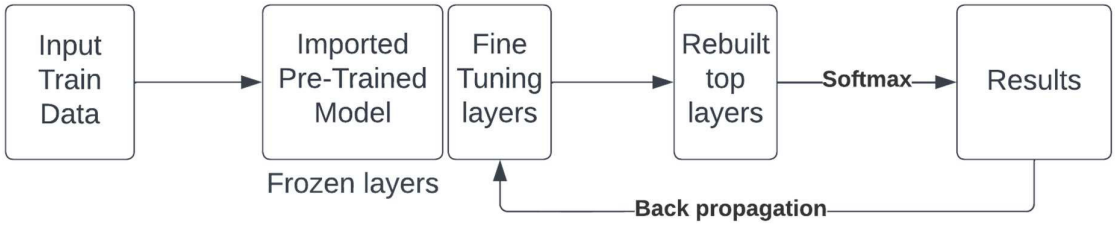


Figure 4.4 : Flow Chart of Fine Tuning Model

The back propagation start from some of the layers that are made trainable in the imported network.

4.2.10 Saving and Testing Models

A Keras model consists of multiple components

- The architecture, or configuration, which specifies what layers the model contain, and how they're connected.
- A set of weights values (the "state of the model").
- An optimizer (defined by compiling the model).
- A set of losses and metrics (defined by compiling the model or calling add_loss() or add_metric()).

The Keras API makes it possible to save all of these pieces to disk at once, or to only selectively save some of them

- Saving everything into a single archive in the TensorFlow Saved Model format (or in the older Keras H5 format). This is the standard practice.

- Saving the architecture / configuration only, typically as a JSON file.
- Saving the weights values only. This is generally used when training the model.

The saved model can be loaded again by using TensorFlow's load_model method. We can use this as a trained model to predict the given input image.

4.3 Deployment of Models

Deployment can be done by saved models loaded into our web server. This project has been deployed by using Python flask micro web framework. After models have been tested they have been saved as pickle objects (Random forest model) and Tensor flow models (VGG16). These models have been loaded into flask server to create an API. This API takes post requests and send the prediction results in json (JavaScript object notation) format.

4.3.1 Creating Flask API

This proposed comprised of mainly 3 end points such each takes a post request from user and serves the prediction results.

/predict_crop - This end point will take post request such that the values required for Random forest model to predict a suitable crop and serves the predicted crop result as a json format.

/predict_fertilizer - This is similar to crop recommendation end point. It will take the input requests for fertilizer prediction.

/predict_disease - This end point takes an input request with image media file and serves the result with predicted disease, confidence level and suggested recovery solution.

4.3.2 User Interface

The user interface is made by using Html, CSS and JavaScript. This project consists of mainly 4 pages such as home page, crop recommendation page, fertilizer recommendation page and paddy disease detection page. Each page consists of navigation bar of entire application and respective input forms to send post requests to the flask server. It will display the results on the web application by java script.

When the sent request receives a response from the server it will update the content on the HTML page without reloading the page. This makes the web interface faster and easily accessible without any intervention. This user-interface page acts as a client to the server. It will send requests to the flask server and receives the json response from the server.

Chapter – 5

SYSTEM DESIGN

5. SYSTEM DESIGN

5.1 System Overview

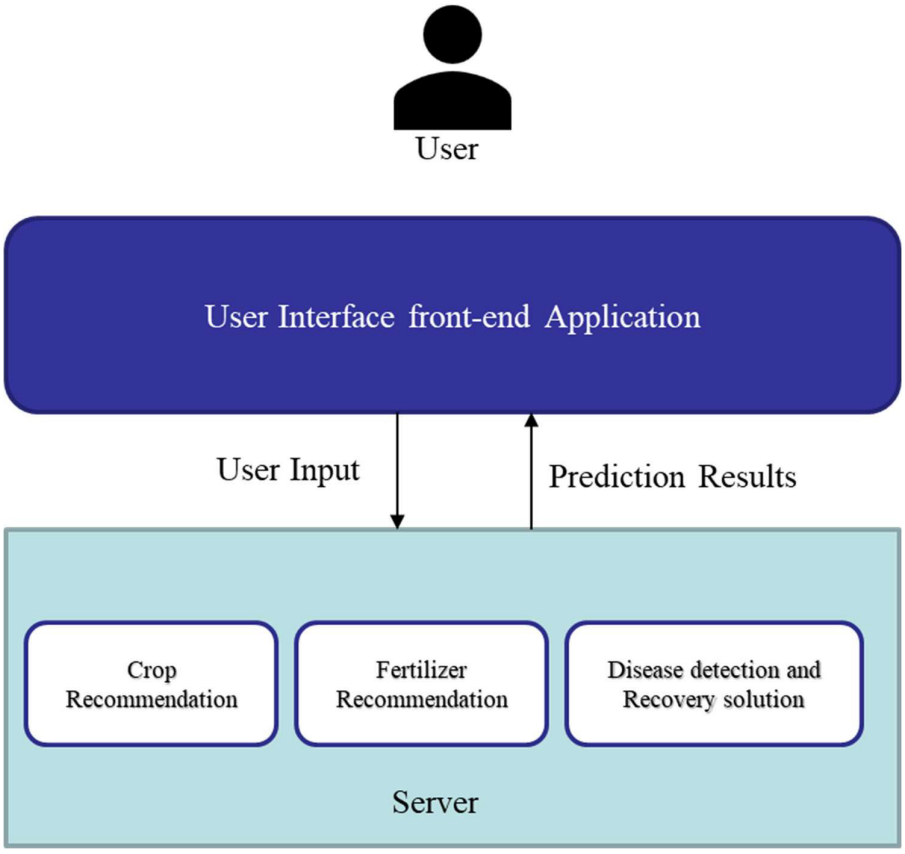


Figure 5.1 : Overview of System Architecture

5.2 Data Flow Diagram:

The DFD is also known as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system. It maps out the flow of information for any process or system, how data is processed in terms of inputs and outputs. It uses defined symbols like rectangles, circles and arrows to show data inputs, outputs, storage points and the routes between each destination. They can be used to analyses an existing system or model of a new one. A DFD can often visually "say" things that would be hard to explain in words and they work for both technical and non- technical. There are four components in DFD:

1. External Entity
2. Process
3. Data Flow
4. Data Store

External Entity

It is an outside system that sends or receives data, communicating with the system. They are the sources and destinations of information entering and leaving the system. They might be an outside organization or person, a computer system or a business system. They are known as terminators, sources and sinks or actors. They can be used to analyses an existing system or model of a new one. These are sources and destinations of the system's input and output.

Data Store

A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. They can be used to analyses an existing system or model of a new one. These are sources and destinations of the system's input and output.

Circle

A circle (bubble) shows a process that transforms data inputs into data outputs. It is just like a function that changes the data, producing an output. It might perform computations for sort data based on logic or direct the dataflow based on business rules.

Data Flow

A curved line shows the flow of data into or out of a process or data store. A dataflow represents a package of information flowing between two objects in the data-flow diagram, Data flows are used to model the flow of information into the system, out of the system and between the elements within the system.

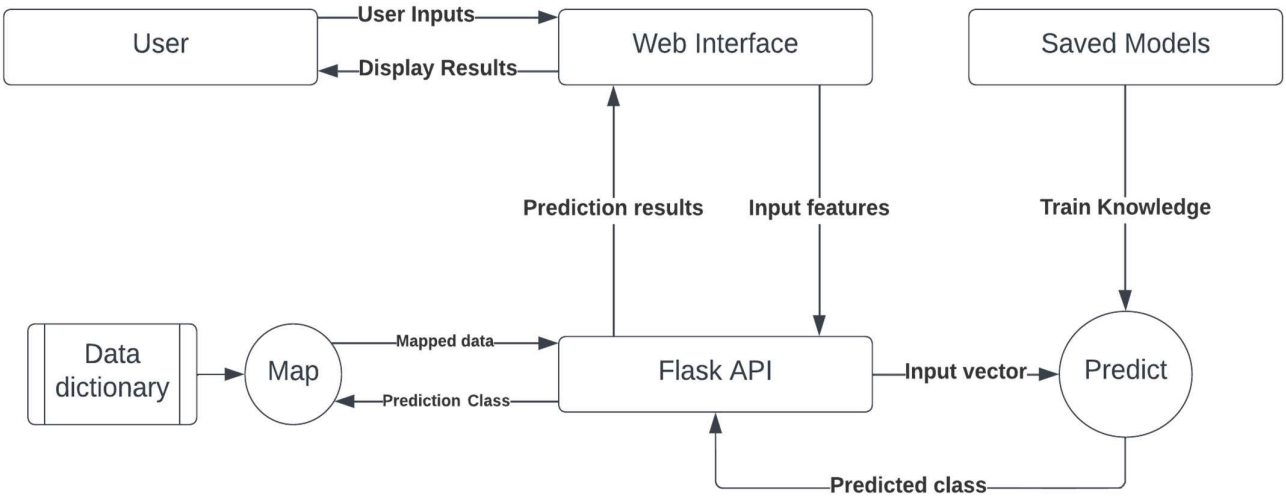


Figure 5.2 : Data Flow Diagram

5.3 Use Case Diagram

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system. Use case describes the behavior of the system as seen from the actor's point of view.

The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment. It describes the function provided by the system as a set of events that yield a visible result for the actor.

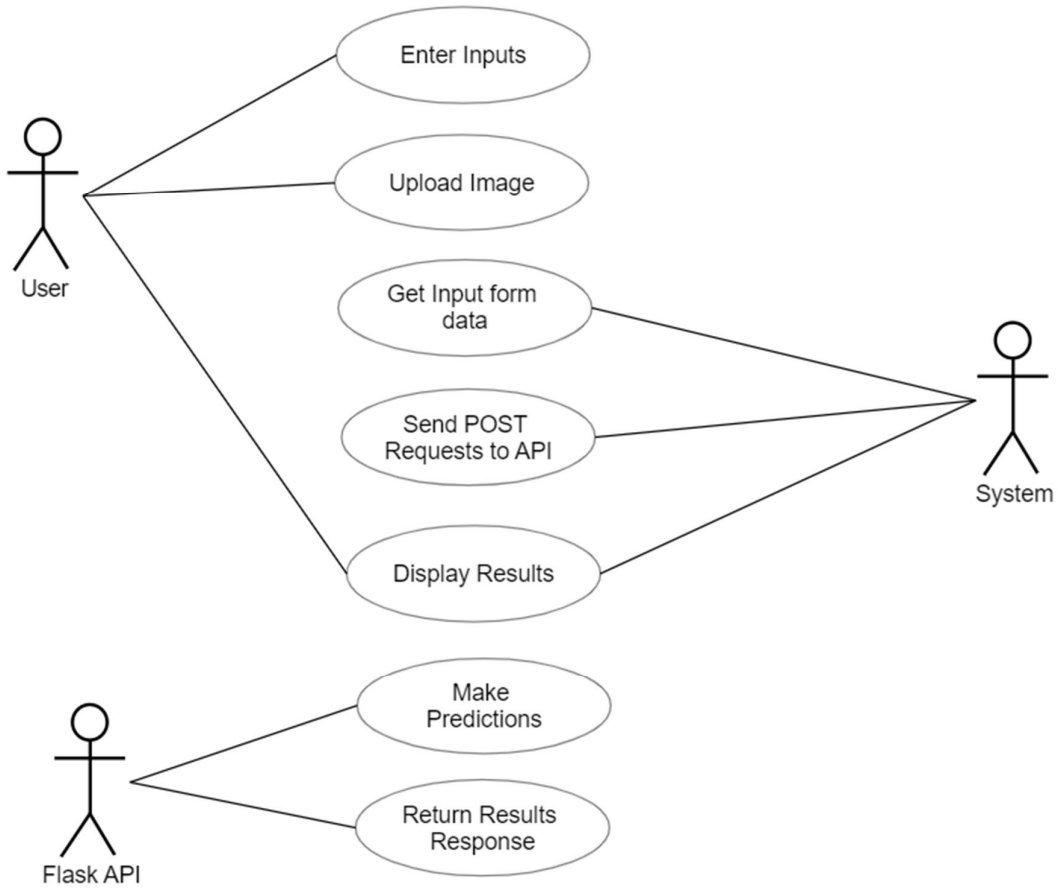


Figure 5.3 : Use Case Diagram of Web UI

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements.

5.4 System Architecture

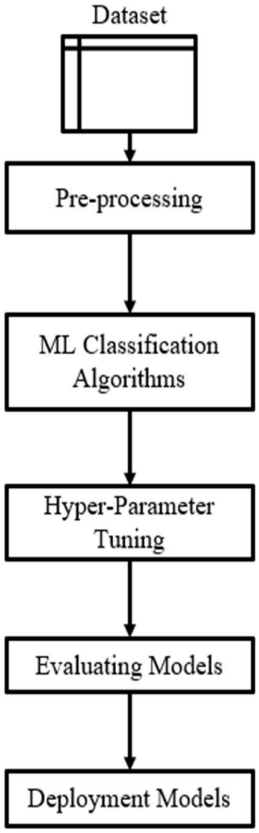


Figure 5.4 : Pipeline for crop and fertilizer recommendation

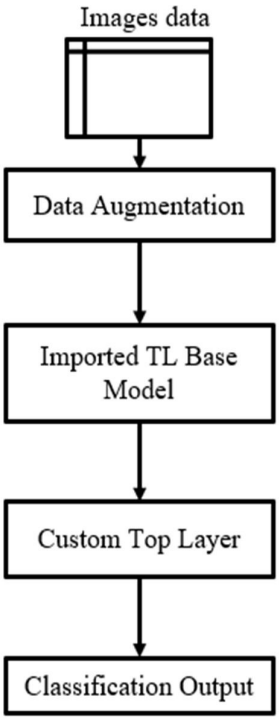


Figure 5.5 : Pipeline for Image Classification

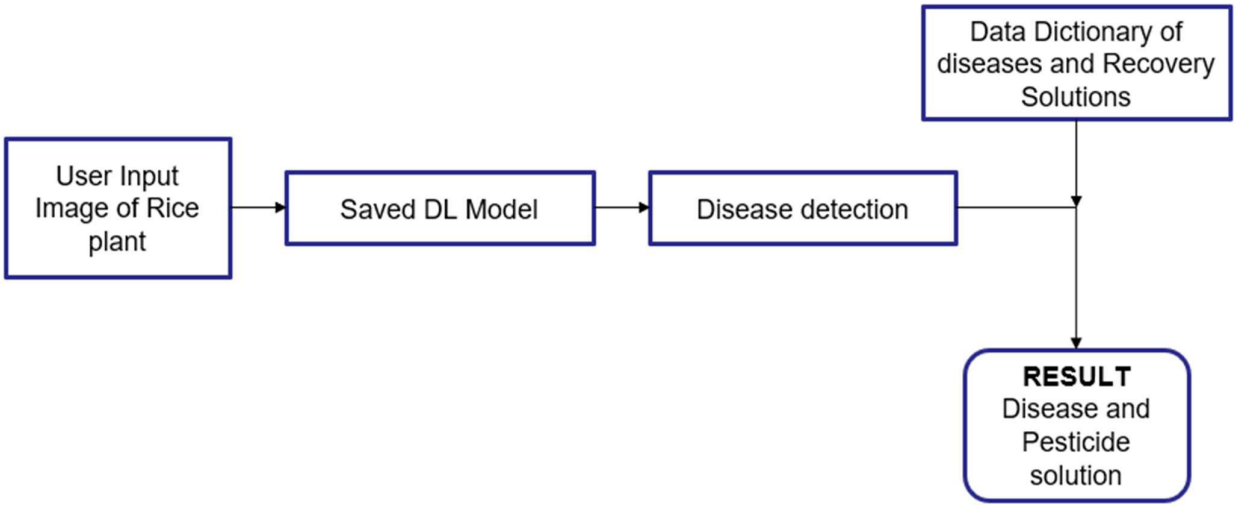


Figure 5.6 : Paddy disease recovery solution suggestion mapping

Chapter – 6

RESULTS & DISCUSSION

6. RESULTS & DISCUSSION

In this project we have proposed a system for helping farmers in making decisions regarding the cultivation of crops. This project mainly implemented a web application for recommending crops and fertilizers along with that it provides a facility to predict the disease of a paddy field with a suitable recovery solution.

However while coming to the models build on crops and fertilizer datasets we got good result by ensemble learning models. VGG16 architecture with "imagenet" weights gave the best results on the paddy leaf images dataset. These results are shown in the following tables.

6.1 Crops and fertilizers dataset results

Table 6.1 : Accuracy scores on Crop & Fertilizers dataset.

Task	KNN Classifier		Random Forest		XGBoost	
	Train	Test	Train	Test	Train	Test
Crop Recommendation	97%	95%	99%	99%	99%	98%
Fertilizer recommendation	100%	94%	100%	100%	100%	100%

The KNN classifier has given best accuracy of 97% with K value = 4. The ensemble learning models outperformed and gave best result with 99% accuracy score. Random forest given best accuracy score with no. of estimators = 100.

6.2 Paddy disease dataset results

Table 6.2 : Accuracy scores on paddy leaf images dataset

Architecture	Training accuracy	Validation Accuracy	Testing Accuracy
Custom Model	73%	62%	60%
EfficientNet B0	94%	90%	88%
VGG16	98%	95%	93%

Each model has been trained up to 50 epochs. On the paddy leaf images dataset Efficient Net B0 has given a validation accuracy of 90% after fine tuning the model. VGG16 after fine tuning of last 20 layers in the network gave a validation accuracy of 95%. Both models taken very less time compared to the custom CNN model.

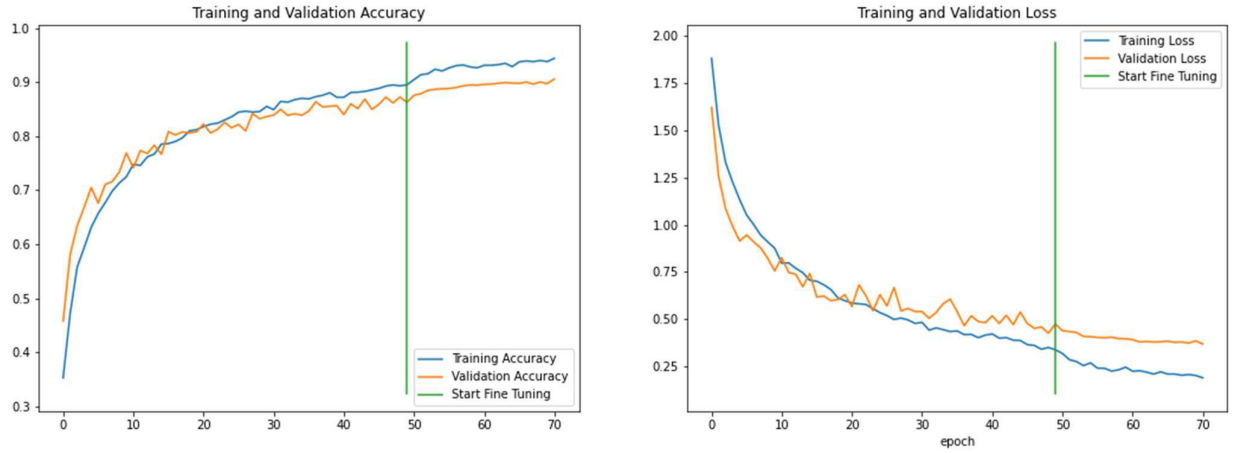


Figure 6.1 : Efficient Net B0 results with fine tuning.

Figure 6.1 describes the training results of the EfficientNet B0 model at different epochs. It is observed that the training and validation accuracy has increased up to 89% and thereafter there is less improvement in the model. Here it will call early stopping. After the fine tuning (green line in the plot) the accuracy has been improved to more than 90%.

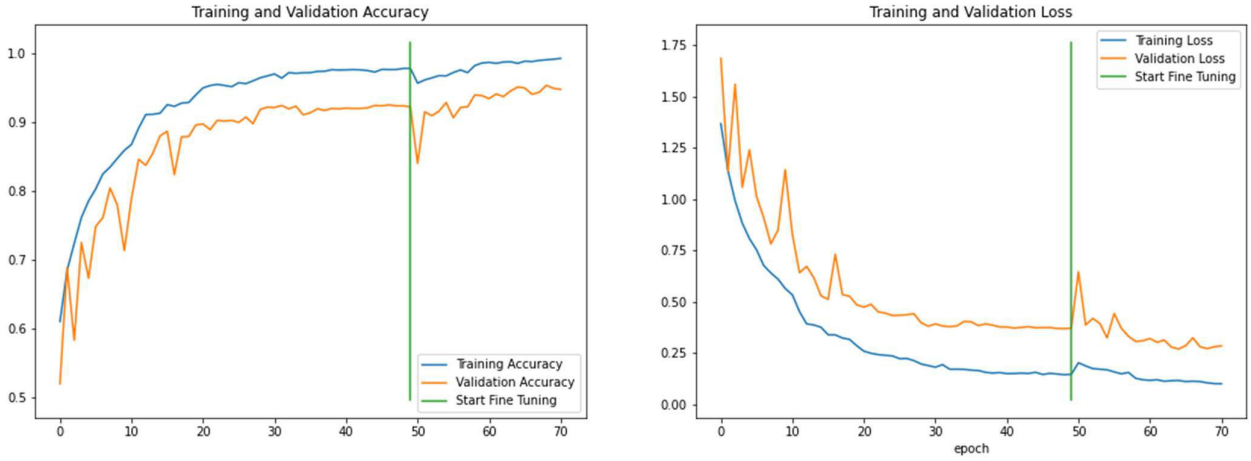


Figure 6.2 : VGG16 results with fine tuning.

Figure 6.2 explains the training and validation results of the VGG16. It has given significant results with less training time. With in 10 epochs it has given 80% training accuracy. After fine tuning the results are very less beneficial but we can observe some improvement in the accuracy scores after 20 more epochs.

However both of these two transfer models have been trained within short time than actual models but still we have obtained accurate results in our problem statement.

6.3 Web Application User Interface Results :

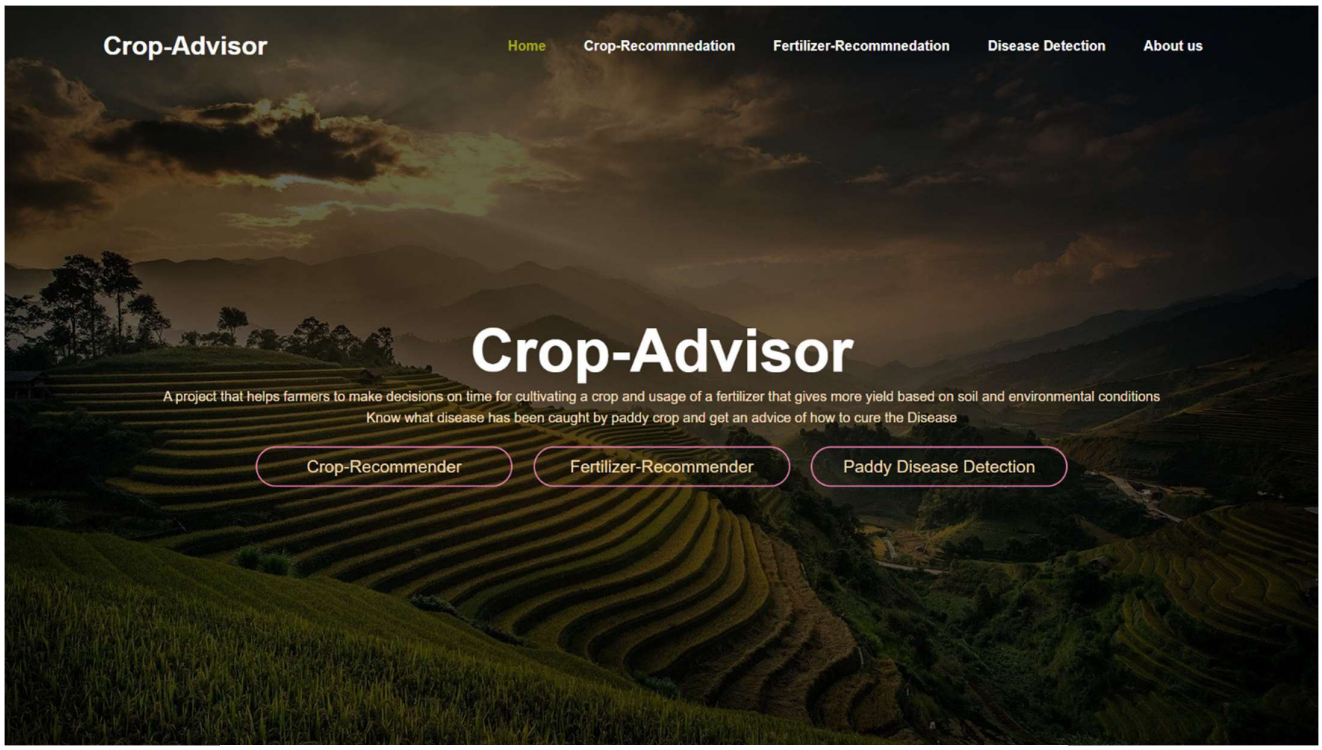


Figure 6.3 : Home page

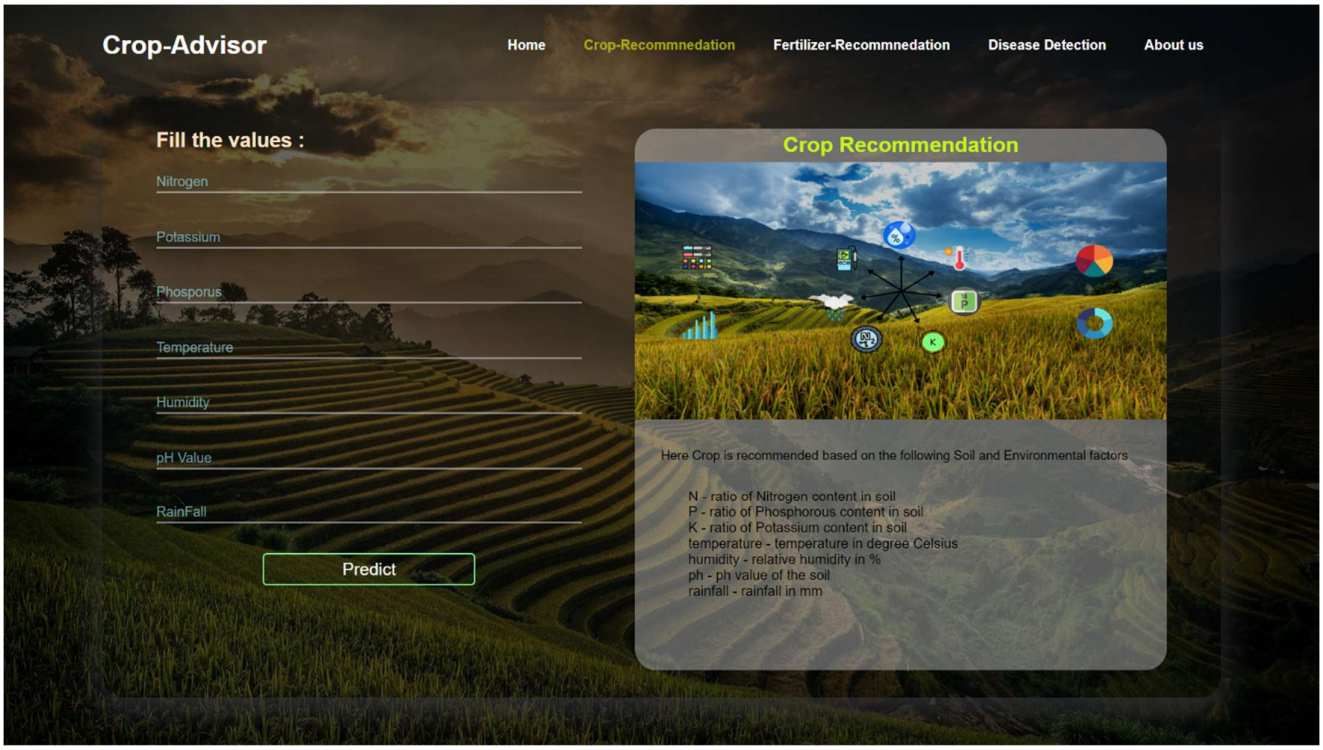


Figure 6.4 : Crop recommendation Page

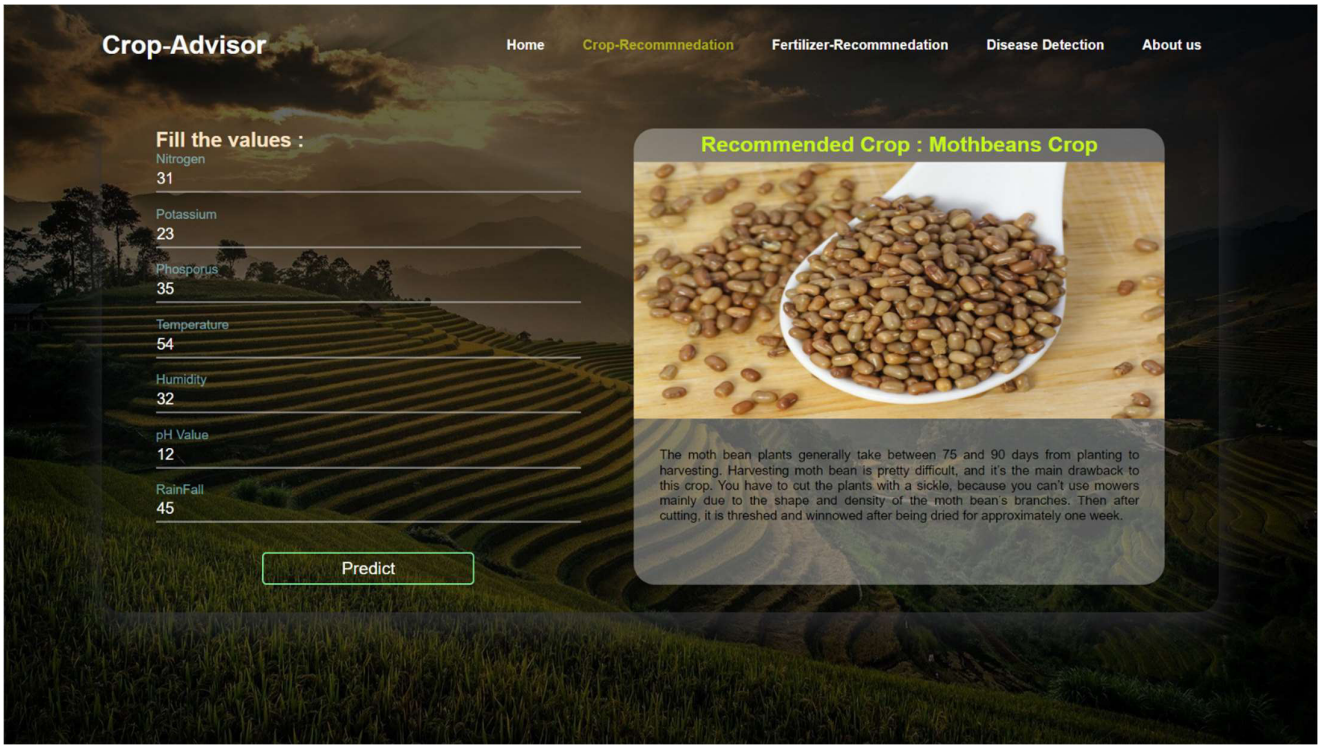


Figure 6.5 : Crop recommendation Predicted result for given Inputs

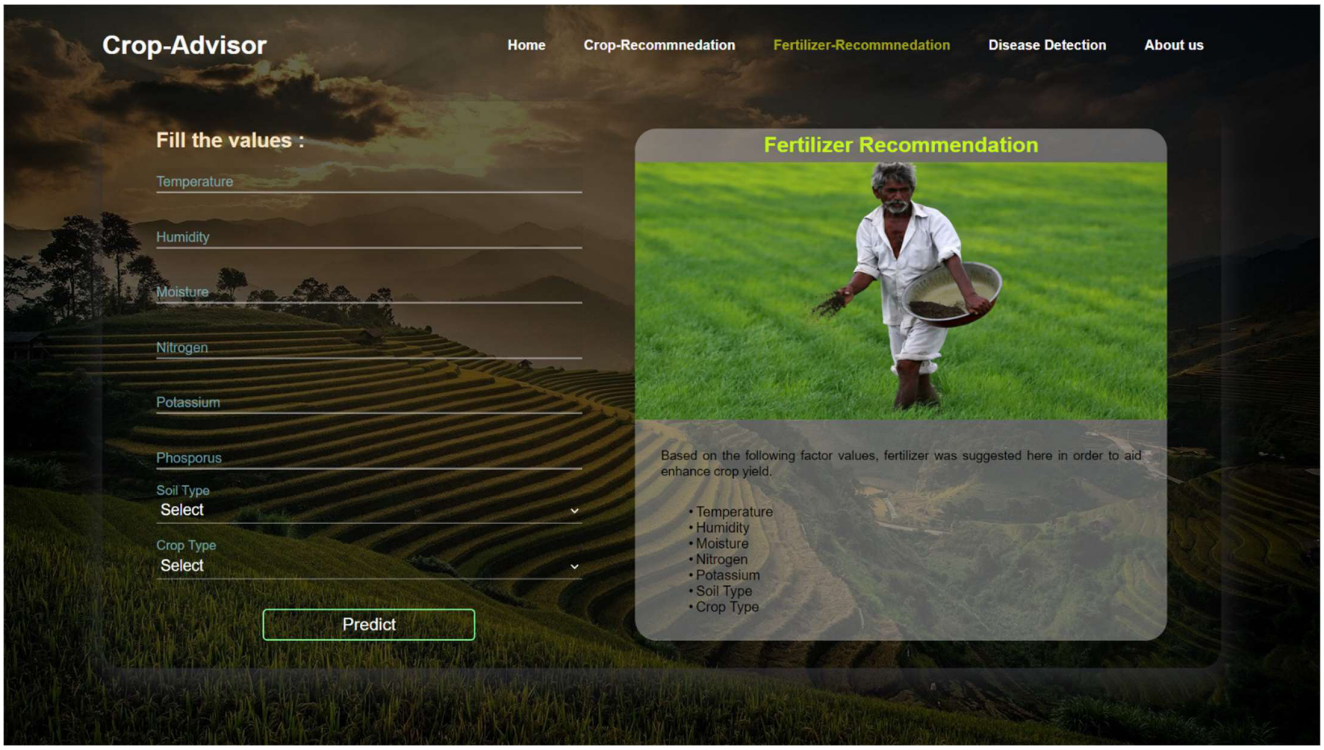


Figure 6.6 : Fertilizer Recommendation Page

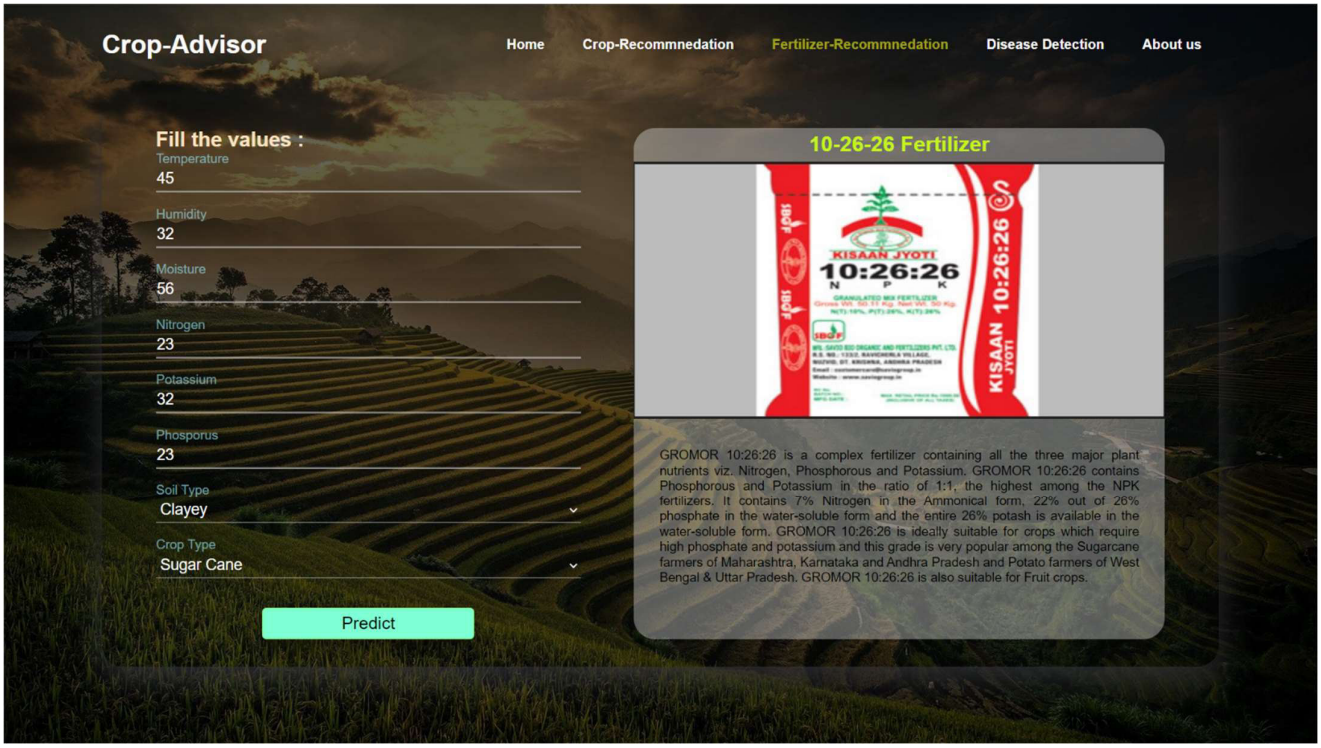


Figure 6.7 : Fertilizer Recommendation Predicted result for given Inputs

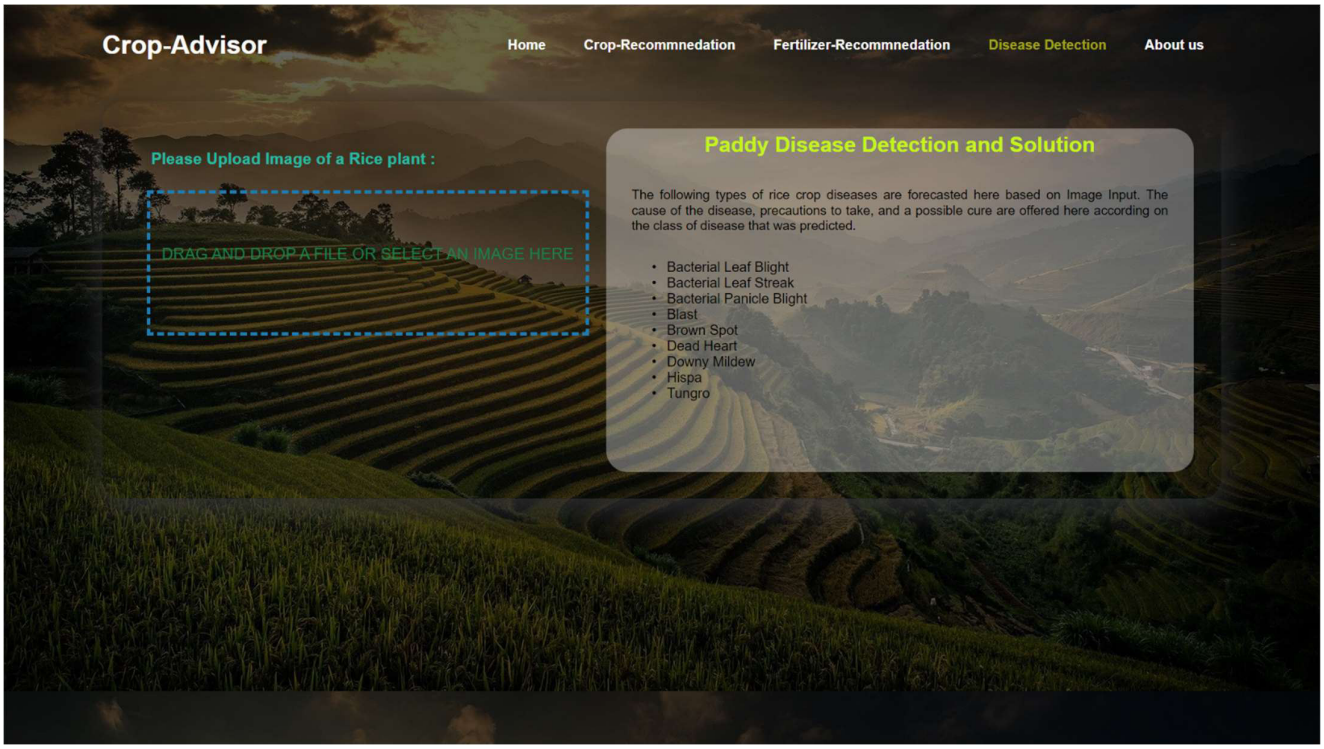


Figure 6.8 : Paddy Disease Identification Page

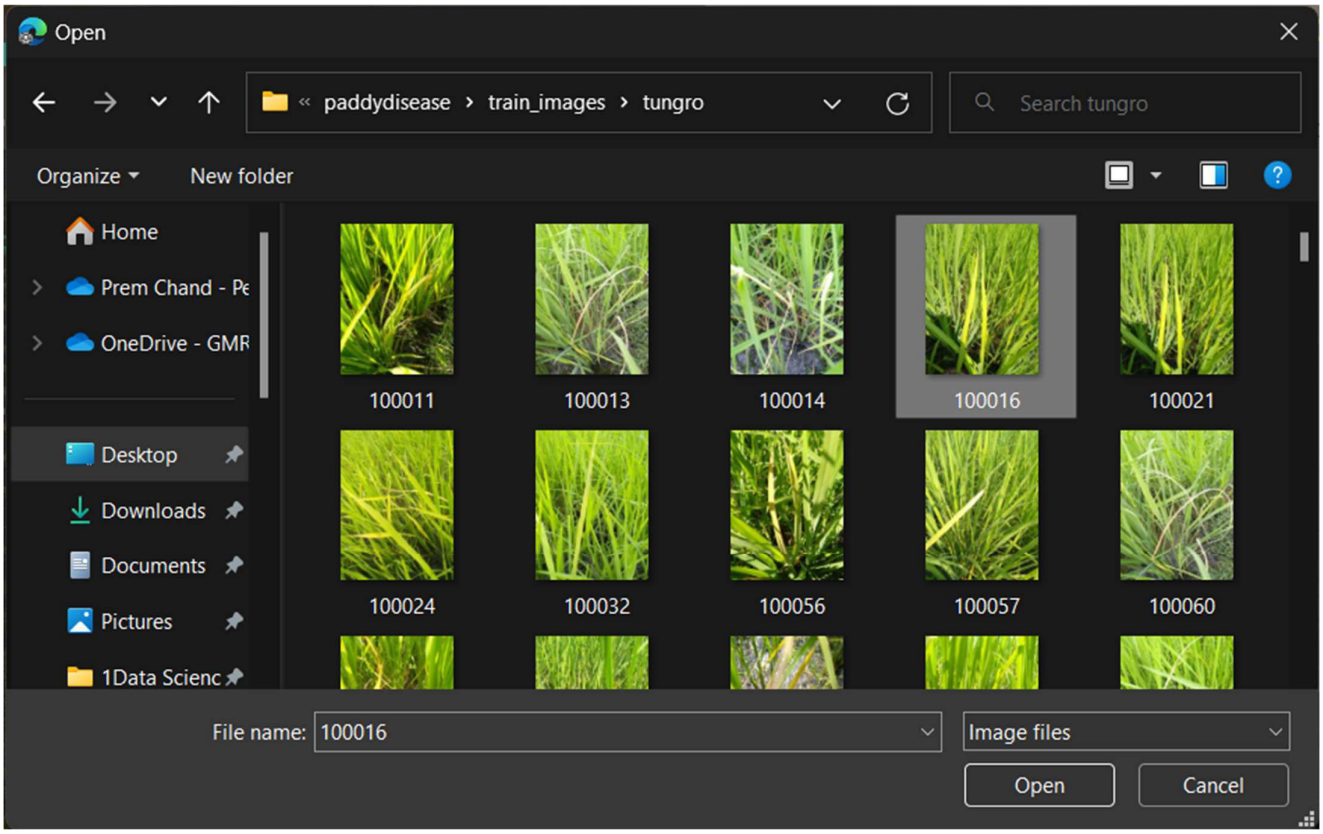


Figure 6.9 : Input image uploading from local system

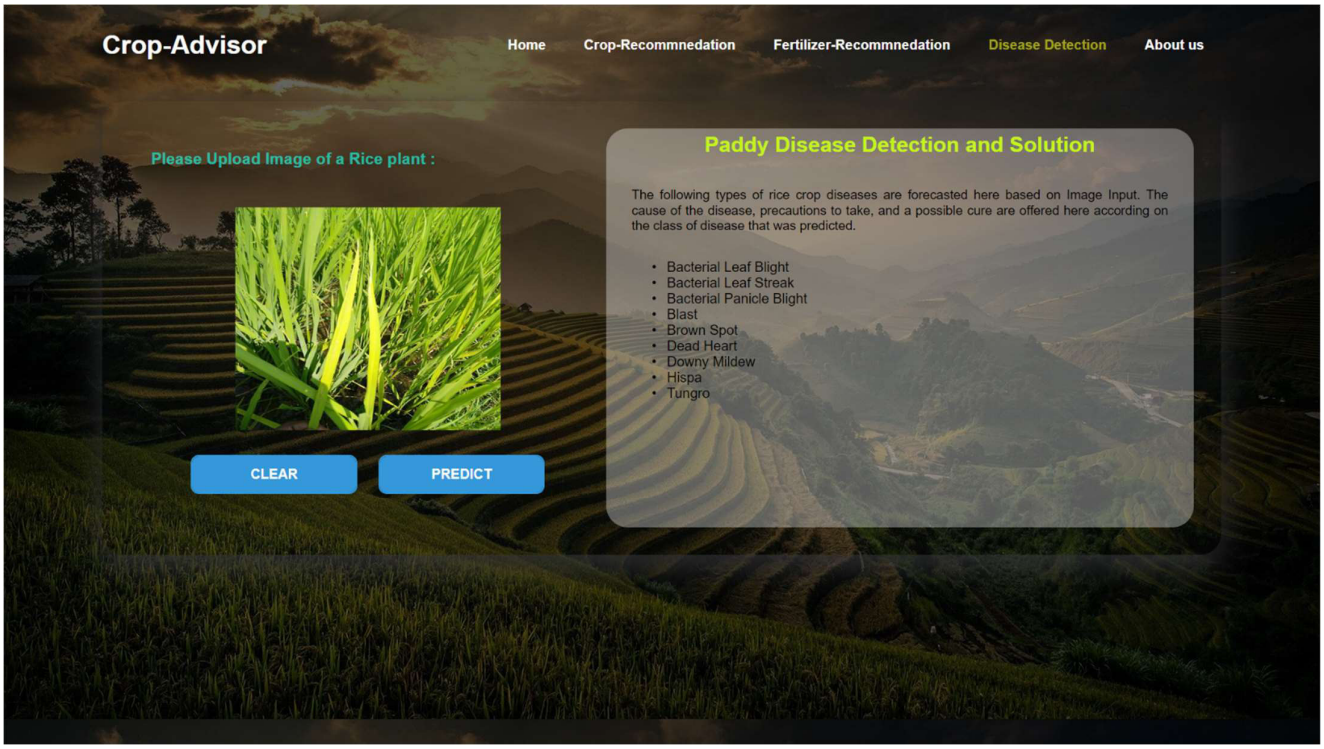


Figure 6.10 : Paddy Disease Identification Page with loaded Input

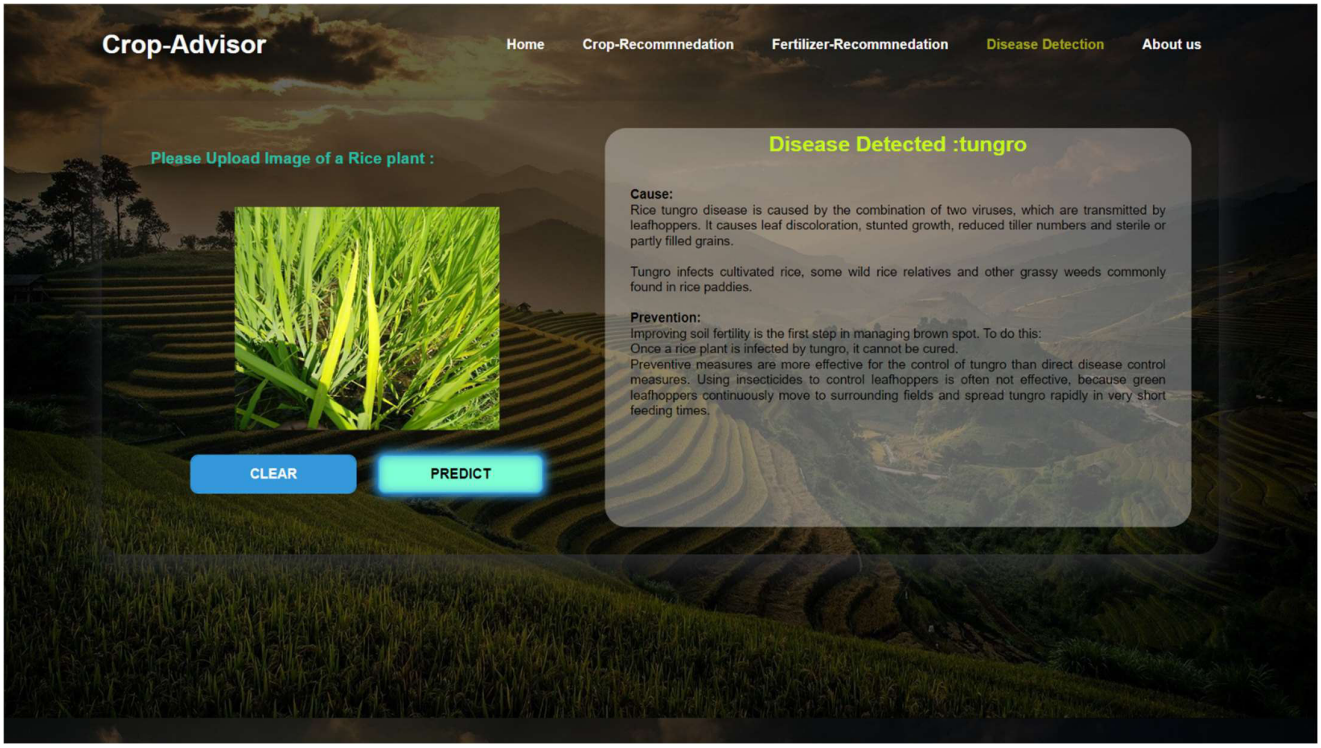


Figure 6.11 : Paddy Disease Predicted result and Preventive measures for given input image

Chapter – 7

CONCLUSION & FUTURESCOPE

7. CONCLUSION & FUTURESCOPE

Main aim of the project is to develop a system that can help farmers to make decision while cultivating several crops. We have developed 3 features within this project Crop recommendation and fertilizer recommendation to get more yield out of the crop. Paddy disease classification to know what kind of disease that has been occurred in the rice field out of 10 types of diseases. We have used KNN classifier , Random forest and XGBboost models for the classification purpose. Among those Random forest performed very well.

While coming to paddy image dataset we have used transfer learning approach to build the efficient image classification models. VGG16 model that was built with imagenet weights has given better result on both training and testing images. After finetuning the transfer learning models has increased their learning capacity. After model building phase a web application with an user friendly interface has been made by using HTML, CSS, JavaScript as frontend of the system and Flask API as the backend of the system.

Agriculture field always need to be developed according to the new technologies in the era. By integrating these technologies into this farming field will be highly beneficial to farmers. However more research need to be done in this sector to build more feasible solutions to the problems faced by many farmers. Solving the real time problems frequently encountered by farmers need to be solved by using the advanced techniques. The yield of a crop is mainly determined by the climatic conditions like temperature, rainfall, soil conditions and fertilizers. Such Yield estimation, weed detection, pesticide recommendation etc. problems can be solved to make more advancement in the agriculture sector.

Chapter – 8

REFERENCES

[1] M. Alencastre-Miranda, R. M. Johnson and H. I. Krebs, "Convolutional Neural Networks and Transfer Learning for Quality Inspection of Different Sugarcane Varieties", in IEEE Transactions on Industrial Informatics, Volume 17, Issue 2, pp. 787–794, February 2021.

[2] Wei Li, Tengfei Zhu, Xiaoyu Li, Jianzhang Dong and Jun Liu, " Recommending Advanced Deep Learning Models for Efficient Insect Pest Detection", in MDPI on Application of Machine Learning in Agriculture , Volume 12, Issue 7, 1065, pp. 1–17, June 2022.

[3] D. Elavarasan and P.M.D. Vincent, "Crop Yield Prediction Using Deep Reinforcement Learning Model for Sustainable Agrarian Applications", in IEEE Access, Volume 8, pp. 86886–86901, April 2020.

[4] K.S Neethu and P. Vijay Ganesh, "Leaf Disease Detection and Selection of Fertilizers using Artificial Neural Network", in IRJET on Computer Science Journal, Volume 4, Issue 6, pp. 1852–1858, June 2017.

[5] Daniya and Dr. S. Vigneshwari, "Deep Neural Network for Disease Detection in Rice Plant Using the Texture and Deep Features", The Computer Journal, Volume 65, Issue 7, pp. 1812–1825, July 2022.

[6] R. Salini, A. Farzana and B. Yamini, "Pesticide Suggestion and Crop Disease Classification using Machine Learning", in IRJET on Computer Science Journal, Volume 11, Issue 4, pp. 27997–27999, April 2021.

[7] T. Daniya and Dr.S. Vigneshwari, "Exponential Rider-Henry Gas Solubility optimization-based deep learning for rice plant disease detection", in Springer, International Journal Of Information Technology, pp. 1–11, July 2022.

[8] F.T. Pinki, N. Khatun and S.M.M. Islam, "Content based paddy leaf disease recognition and remedy prediction using support vector machine," 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–5, December 2017.

[9] Junde Chen, Defu Zhang, Yarser A Nanehkaran and Dele Li, "Detection of rice plant diseases based on deep transfer learning", Journal of the Science of Food and Agriculture, Volume 100, Issue 7, pp. 3246–3256, March 2020.

[10] Senthil Kumar Swami Durai and Mary Divya Shamili, "Smart farming using Machine Learning and Deep Learning techniques", Decision Analytics Journal, Volume 3, ISSN 2772–6622, pp. 1–30, March 2022.

[11] R. Alfred, J.H. Obit, C.P.Y. Chin, H. Haviluddin and Y. Lim, "Towards Paddy Rice Smart Farming: A Review on Big Data, Machine Learning, and Rice Production Tasks", in IEEE Access, Volume 9, pp. 50358–50380, March 2021.

[12] S. Mishra, D. Mishra and G.H. Santra, "Applications of Machine Learning Techniques in Agricultural Crop Production: A Review Paper", Indian Journal of Science and Technology, Volume 9, Issue 38, pp. 1–14, October 2016.

[13] J. Sustain, DR. D. Sivaganesan, "Performance Estimation of Sustainable Smart Farming with Blockchain Technology", Wireless Systems, Volume 03, ISSN 2582–3167, Issue 02, pp. 97–106, June 2021.

APPENDIX - A

APPENDIX – A

Crop Recommendation

```
// Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.neighbors import  KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, plot_confusion_matrix, confusion_matrix
import pickle
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline

//Loading data
data = pd.read_csv('./Data/Crop_recommendation.csv')

//check size of data
print(data.shape)

//check for null values
data.isna().sum()

//plotting the data histograms
features = data.columns[:-1]
plt.figure(figsize=(14,12))
for i,col in enumerate(features):
        plt.subplot(3,3,i+1)
        sns.histplot(data[col])
plt.show()
plt.tight_layout()
```

```
//Bi variate analysis by using barplot
for i,col in enumerate(features):
        plt.figure(figsize=(14,5))
        sns.barplot(x='label',y=col,data=data)
        plt.xticks(rotation=90)
plt.title(f"{col} vs Crop type")
plt.show()

//Pairplot Multivariate analysis
plt.figure(figsize=(19,17))
sns.pairplot(data, hue = "label")
plt.show()
```

Pre-Processing

```
//Label encoding
label_encoder = LabelEncoder()
X = data[features]
y = label_encoder.fit_transform(data["label"])

//Train Test split
X_train, X_test, y_train, y_test = train_test_split(X.values, y, test_size = 0.2, random_state = 0)
print(f"Train Data: {X_train.shape}, {y_train.shape}")
print(f"Train Data: {X_test.shape}, {y_test.shape}")
```

KNN Classifier :

```
error_rate = []
for i in range(1, 50):
        knn = KNeighborsClassifier(n_neighbors = i)
        pipeline = make_pipeline(StandardScaler(), knn)
        pipeline.fit(X_train, y_train)
        predictions = pipeline.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        print(f"Accuracy at k = {i} is {accuracy}")
        error_rate.append(np.mean(predictions != y_test))
plt.plot(range(1,50),error_rate,color='blue', linestyle='dashed', marker='o',markerfacecolor='red',
markersize=10)
plt.title('Error Rate vs. K Value')
```

Random Forest :

```
rf_pipeline = make_pipeline(StandardScaler(), RandomForestClassifier(random_state = 18))
rf_pipeline.fit(X_train, y_train)

# Accuray On Test Data
predictions = rf_pipeline.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy on Test Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y_test, predictions), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

XG Boost :

```
xgb_pipeline = make_pipeline(StandardScaler(),XGBClassifier())
xgb_pipeline.fit(X_train, y_train)

# Accuray On Test Data
predictions = xgb_pipeline.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy on Test Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y_test, predictions), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

Saving Models :

```
pickle.dump(knn_pipeline, open("./models/knn_pipeline.pkl", "wb"))
pickle.dump(rf_pipeline, open("./models/rf_pipeline.pkl", "wb"))
pickle.dump(xgb_pipeline, open("./models/xgb_pipeline.pkl", "wb"))
pickle.dump(label_dict, open("./models/label_dictionary.pkl", "wb"))
```

Fertilizer Recommendation

```
data = pd.read_csv('./Data/Fertilizer.csv')
data.shape
```

Visualization and Analysis

```
sns.countplot(data['Fertilizer Name'])
continuous_features = ["Temparature", "Humidity ", "Moisture", "Nitrogen", "Phosphorous"]
categorical_features = ["Soil Type", "Crop Type"]
plt.figure(figsize=(12,8))
for i,col in enumerate(continuous_features):
        plt.subplot(2,3,i+1)
        sns.histplot(data[col])
        plt.xlabel(col)
        plt.tight_layout()
plt.show()


plt.figure(figsize=(16,6))
for i,col in enumerate(categorical_features):
        plt.subplot(1,2,i+1)
        sns.countplot(data[col])
        plt.xlabel(col)
        plt.xticks(rotation=45)
plt.show()
```

Encoding Categorical data :

```
soil_type_encoder = LabelEncoder()
data['Soil Type'] = soil_type_encoder.fit_transform(data['Soil Type'])
crop_encoder = LabelEncoder()
data['Crop Type'] = crop_encoder.fit_transform(data['Crop Type'])
fertilizer_encoder = LabelEncoder()
data['Fertilizer Name']=fertilizer_encoder.fit_transform(data['Fertilizer Name'])
```

X and y splitting data :

```
X = data.drop("Fertilizer Name",axis=1)
y = data['Fertilizer Name']
```

UpSampling :

```
upsampler = SMOTE()
X,y = upsampler.fit_resample(X,y)
```

Train Test split :

```
X_train, X_test, y_train, y_test = train_test_split(X.values, y, test_size = 0.2, random_state = 0)
```

KNN Classifier :

```
error_rate = []
for i in range(1, 30):
        pipeline = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors = i))
        pipeline.fit(X_train, y_train)
        predictions = pipeline.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        print(f"Accuracy at k = {i} is {accuracy}")
        error_rate.append(np.mean(predictions != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,30),error_rate,color='blue', linestyle='dashed', marker='o',markerfacecolor='red',
markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
print("Minimum error:-",min(error_rate),"at K =",error_rate.index(min(error_rate))+1)
```

SVM

```
svm_pipeline = make_pipeline(StandardScaler(), SVC(probability=True))
svm_pipeline.fit(X_train, y_train)

# Accuray On Test Data
predictions = svm_pipeline.predict(X_test)
```

XGBoost

```
xgb_pipeline = make_pipeline(StandardScaler(), XGBClassifier(random_state = 96))
xgb_pipeline.fit(X_train, y_train)

# Accuray On Test Data
predictions = xgb_pipeline.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy on Test Data: {accuracy*100}%")
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(y_test, predictions), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

Saving Models and dictionaries

```
pickle.dump(svm_pipeline, open("./models/Fertilizer/svm_pipeline.pkl", "wb"))
pickle.dump(rf_pipeline, open("./models/Fertilizer/rf_pipeline.pkl", "wb"))
pickle.dump(xgb_pipeline, open("./models/Fertilizer/xgb_pipeline.pkl", "wb"))
pickle.dump(fertilizer_dict, open("./models/Fertilizer/fertilizer_dict.pkl", "wb"))
pickle.dump(crop_type_dict, open("./models/Fertilizer/crop_type_dict.pkl", "wb"))
pickle.dump(soil_type_dict, open("./models/Fertilizer/soil_type_dict.pkl", "wb"))
```

Paddy disease classification:

Efficient Net B0:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Set seeds to make the experiment more reproducible.
import random
def seed_everything(seed = 0):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    tf.random.set_seed(seed)
seed = 0
seed_everything(seed)
BATCH_SIZE = 32
IMG_SIZE = 224
```

Loading Images data :

```
input_path = '../input/paddy-disease-classification/'
train_data_dir = input_path + 'train_images/'
test_data_dir = input_path + 'test_images/'
train_ds = tf.keras.utils.image_dataset_from_directory(
                                                    train_data_dir,
                                                    validation_split = 0.2,
                                                    subset="training",
                                                    seed = 123,
                                                    image_size = (IMG_SIZE, IMG_SIZE),
                                                    batch_size = BATCH_SIZE)

val_ds = tf.keras.utils.image_dataset_from_directory(
```

```
                                        train_data_dir,
                                        validation_split = 0.2,
                                        subset="validation",
                                        seed = 123,
                                        image_size = (IMG_SIZE, IMG_SIZE),
                                        batch_size = BATCH_SIZE)

# We can find the class names in the class_names attribute on these datasets.
# These correspond to the directory names in alphabetical order.
class_names = train_ds.class_names
n_classes = len(class_names)
plt.figure(figsize = (10, 10))

for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(class_names[labels[i]]+str(labels.numpy()[i]))
        plt.axis('off')
```

Data Augmentation :

```
data_augmentation = keras.Sequential([
                        layers.RandomRotation(factor = 0.1),
                        layers.RandomFlip('horizontal'),
                        layers.RandomTranslation(height_factor = 0.1, width_factor = 0.1),
                        layers.RandomContrast(factor = 0.1),
                        ])
```

One hot encoding :

```
AUTOTUNE = tf.data.AUTOTUNE
def prepare_dataset(ds):
    def input_preprocess(image, label):
        label = tf.one_hot(label, n_classes)
        return image, label
    ds = ds.map(input_preprocess, num_parallel_calls = AUTOTUNE)
    ds = ds.cache().prefetch(buffer_size = AUTOTUNE)
    return ds
train_ds = prepare_dataset(train_ds)
val_ds = prepare_dataset(val_ds)
```

Model creation :

```
def create_model(n_classes):
        inputs = layers.Input(shape = (IMG_SIZE, IMG_SIZE, 3))
        x = data_augmentation(inputs)
        conv_base = tf.keras.applications.EfficientNetB0(include_top = False,input_tensor = x,
                                                          weights = 'imagenet')

        # Freeze the pretrained weights
        conv_base.trainable = False

        # Rebuild top
        x = layers.GlobalAveragePooling2D(name = 'gap')(conv_base.output)
        x = layers.BatchNormalization()(x)
        x = layers.Dense(1024, activation = 'relu')(x)
        x = layers.Dropout(0.25)(x)
        x = layers.Dense(512, activation = 'relu')(x)
        x = layers.Dropout(0.25)(x)
        x = layers.Dense(128, activation = 'relu')(x)
        x = layers.Dropout(0.25)(x)
        x = layers.Dense(64, activation = 'relu')(x)
        x = layers.Dropout(0.25)(x)
        outputs = layers.Dense(n_classes, activation = 'softmax')(x)

        # Compile
        model = tf.keras.Model(inputs, outputs, name = 'EfficientNetB0')
        model.compile(
                        optimizer = tf.keras.optimizers.Adam(learning_rate = 1e-3),
                        loss = 'categorical_crossentropy',
                        metrics = ['accuracy'])
        return model
model = create_model(n_classes)
```

Model Training:

```
initial_epochs = 50
history = model.fit(train_ds, epochs = initial_epochs, validation_data = val_ds )
```

VGG16 Model Creation:

```
def create_model(n_classes, fine_tune = 0):
        inputs = layers.Input(shape = (IMG_SIZE, IMG_SIZE, 3))
        x = layers.Rescaling(1.0/255)(inputs)
```

```
    x = data_augmentation(x)
    conv_base = tf.keras.applications.VGG16(include_top = False,input_tensor = x,
                                            weights = 'imagenet')

    # Freeze the pretrained weights
    if fine_tune > 0:
        for layer in conv_base.layers[:-fine_tune]:
            layer.trainable = False
    else:
        for layer in conv_base.layers:
            layer.trainable = False

    # Rebuild top
    x = layers.GlobalAveragePooling2D(name = 'gap')(conv_base.output)
    x = layers.BatchNormalization()(x)
    initializer = tf.keras.initializers.GlorotUniform()
    x = tf.keras.layers.Dense(256,
                              kernel_regularizer = tf.keras.regularizers.l2(0.001),
                              kernel_initializer = initializer, activation = 'relu')(x)
    x = tf.keras.layers.Dropout(0.5)(x)
    outputs = layers.Dense(n_classes, kernel_initializer = initializer,activation = 'softmax',
                           name = 'pred')(x)

    # Compile
    model = tf.keras.Model(inputs, outputs, name = 'VGG16')
    model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = 1e-3),
                  loss = 'categorical_crossentropy',
                  metrics = ['accuracy'])
    return model
model = create_model(n_classes, 2)


# ModelCheckpoint callback - save best weights
tl_checkpoint_1 = ModelCheckpoint(filepath = 'vgg16_best_weights.hdf5',
                                  save_best_only = True,
                                  verbose = 0)

# EarlyStopping
early_stop = EarlyStopping(monitor = 'val_loss',
                           patience = 10,
```

```
                                    restore_best_weights = True,
                                    mode = 'min')
rlrop = ReduceLROnPlateau(monitor = 'val_loss',
                                     factor = 0.5,
                                     patience = 3,
                                     verbose = 1)
epochs = 50
history = model.fit(train_ds,
                            epochs = epochs,
                            validation_data = val_ds,
                            callbacks = [tl_checkpoint_1, early_stop, rlrop] )
```

Fine Tuning Model:

```
def unfreeze_model(model):
        for layer in model.layers[-20:]:
                if not isinstance(layer, layers.BatchNormalization):
                        layer.trainable = True
        model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = 1e-5),
                            loss = 'categorical_crossentropy',
                            metrics = ['accuracy'])

unfreeze_model(model)
fine_tune_epochs = 20
initial_epochs=50
total_epochs =  initial_epochs + fine_tune_epoch
history_fine = model.fit(train_ds,
                                epochs = total_epochs,
                                initial_epoch = history.epoch[-1],
                                validation_data = val_ds, verbose = 1,
                                callbacks = [tl_checkpoint_1, early_stop, rlrop] )
```

Deployment

Flask Server :

```
from pydoc import render_doc
from urllib import response
from flask import Flask,redirect,jsonify,request,render_template
import util
import os
app = Flask(__name__)
```

```
app.config['UPLOAD_FOLDER']="./"

@app.route('/')
def home():
	return render_template("index.html")

@app.route('/predict_crop',methods=["POST"])
def predict_crop():
	input_values = list(request.form.to_dict().values())
	input_values = list(map(float,input_values))
	X = [input_values]
	result = util.predict_crop(X)
	response = jsonify({ 'predicted_crop':result })
	response.headers.add("Access-Control-Allow-Origin",'*')
	return response

@app.route('/predict_fert',methods=["POST"])
def predict_fert():
	print("Request recieved Predicting Ferlizier..")
	input_values = list(request.form.to_dict().values())
	input_values = list(map(float,input_values))
	X = [input_values]
	result = util.predict_fert(X)
	response = jsonify({
			'predicted_fert' : result,
			})
	response.headers.add("Access-Control-Allow-Origin",'*')
	return response

@app.route("/predict_disease",methods=["GET","POST"])
def predict_disease():
	if request.method=='POST':
		img = request.files['input_image']
		img.save(os.path.join(app.config['UPLOAD_FOLDER'], "input_image.jpg"))
		output,confidence = util.predict_disease()
		print(output,confidence)
		response = jsonify({ 'output_disease':output,
					'confidence':confidence
				})
```

```
            response.headers.add("Access-Control-Allow-Origin",'*')
            return response
        else:
            return "GET"

if __name__=="__main__":
        util.load_saved_artifacts()
        print("Starting Flask server....")
        app.run()
```

Models Serving :

```
import pickle as pkl
import json
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array

#load_saved_artifacts
def load_saved_artifacts():
        print("Loading... saved artifacts.....")
        class_names = ['bacterial_leaf_blight', 'bacterial_leaf_streak',
                       'bacterial_panicle_blight', 'blast', 'brown_spot',
                       'dead_heart', 'downy_mildew', 'hispa', 'normal',
                       'tungro']
        crop_rf_model  = pkl.load(open('models/Crop/rf_pipeline.pkl',"rb"))
        crop_labels= pkl.load(open('models/Crop/label_dictionary.pkl',"rb"))
        fert_rf_model = pkl.load(open('models/Fertilizer/rf_pipeline.pkl',"rb"))
        fertilizer_dict = pkl.load(open('models/Fertilizer/fertilizer_dict.pkl',"rb"))
        soil_type_dict = pkl.load(open('models/Fertilizer/soil_type_dict.pkl',"rb"))
        crop_type_dict = pkl.load(open('models/Fertilizer/crop_type_dict.pkl',"rb"))
        MODEL = tf.keras.models.load_model("./EfficientNetB0")
        print("Loaded Artifacts done...")

#predict_crop
def predict_crop(X):
        rf_prediction = crop_labels[crop_rf_model.predict(X)[0]]
        return rf_prediction
```

```
def predict_fert(X):
    rf_prediction = fertilizer_dict[fert_rf_model.predict(X)[0]]
    return rf_prediction

def predict_disease():
    img_file = tf.keras.utils.load_img("./input_image.jpg",target_size=(224, 224))
    img_arr = img_to_array(img_file)
    img_f = tf.expand_dims(img_arr,0)
    preds = MODEL.predict(img_f)
    output = class_names[np.argmax(preds)]
    confidence = np.round(preds.max(), 3) * 100
    return output,confidence
```

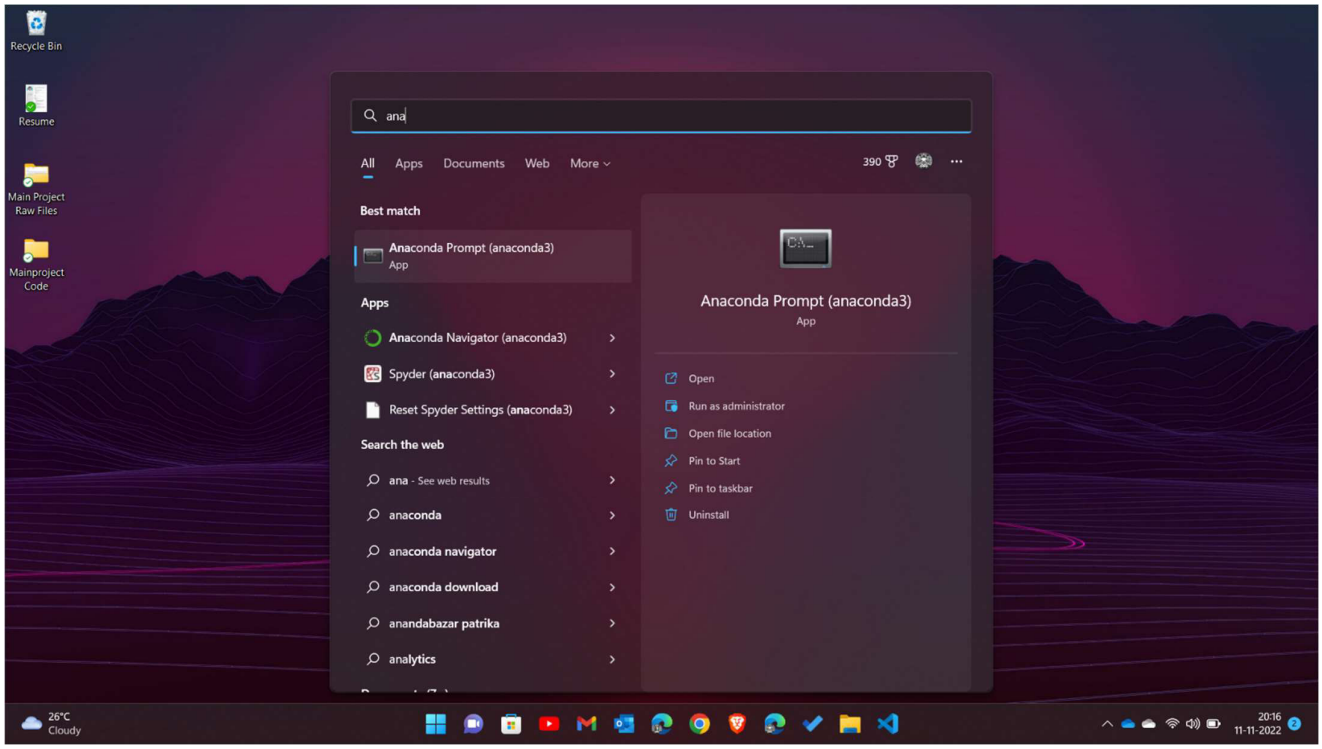
JavaScript API Calls

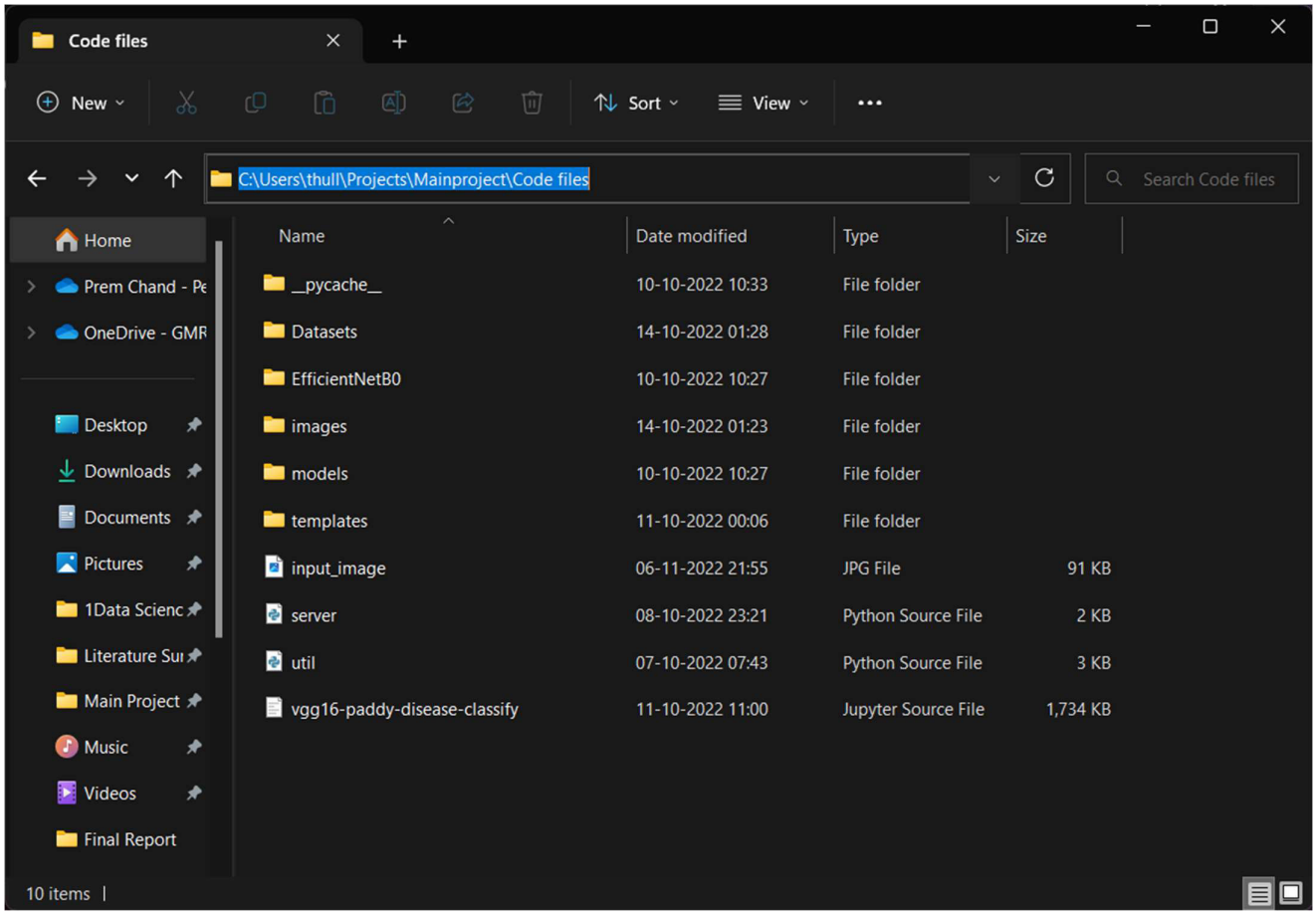
```
function onClickedPredictCrop() {
    console.log("Predict crop clicked");
    var N = document.getElementById("uiN").value;
    var P = document.getElementById("uiP").value;
    var K = document.getElementById("uiK").value;
    var Temp = document.getElementById("uiTemperature").value;
    var Humidity = document.getElementById("uiHumidity").value;
    var ph = document.getElementById("uipH").value;
    var rainfall = document.getElementById("uiRainfall").value;
    var predCrop = document.getElementById("uiPredictedCrop");
    var cropImage = document.getElementById("uiCropImage");
    var cropDesc = document.getElementById("uiCropDesc");
    var url = "http://127.0.0.1:5000/predict_crop";

    $.post(url,

        {Nitrogen : N,
        Phosporus: P,
        Potassium: K,
        Temperature : Temp,
        Humidityy : Humidity,
        pH : ph,
        rainFall : rainfall
        },
        function(data, status) {
```

```
                    console.log(data.predicted_crop);
                    const predictedCrop = cropData[data.predicted_crop];
                    predCrop.innerHTML = "Recommended Crop : " + predictedCrop.title;
                    cropImage.src=predictedCrop.imageUrl;
                    cropDesc.innerHTML = "<p>" +  predictedCrop.description + "</p>";
                    console.log(status);
                }
        );
}

function onClickPredictDisease(){
        console.log("Predict Disease clicked");
        const formData = new FormData();
        const fileField = document.querySelector('input[type="file"]');
        formData.append('input_image', fileField.files[0]);
        fetch('http://127.0.0.1:5000/predict_disease', {method: 'POST', body: formData })
        .then(response => response.json())
        .then(result => { console.log('Success:', result);
        if(result.output_disease == "normal"){
                document.getElementById("uiPredictedDisease").innerHTML= " No Disease
                                                        Detected.. The Plant is Healthy. "
        }
        else{
                document.getElementById("uiPredictedDisease").innerHTML= " Disease Detected
                                                        :" + result.output_disease ;
                document.getElementById("uiSolutionContent").innerHTML = " <p> <strong>Suggested
                Solution : </strong><br>  " + "Solution "  + "</p>";
        }
        .catch(error => { console.error('Error:', error);});
}
```

APPENDIX - B

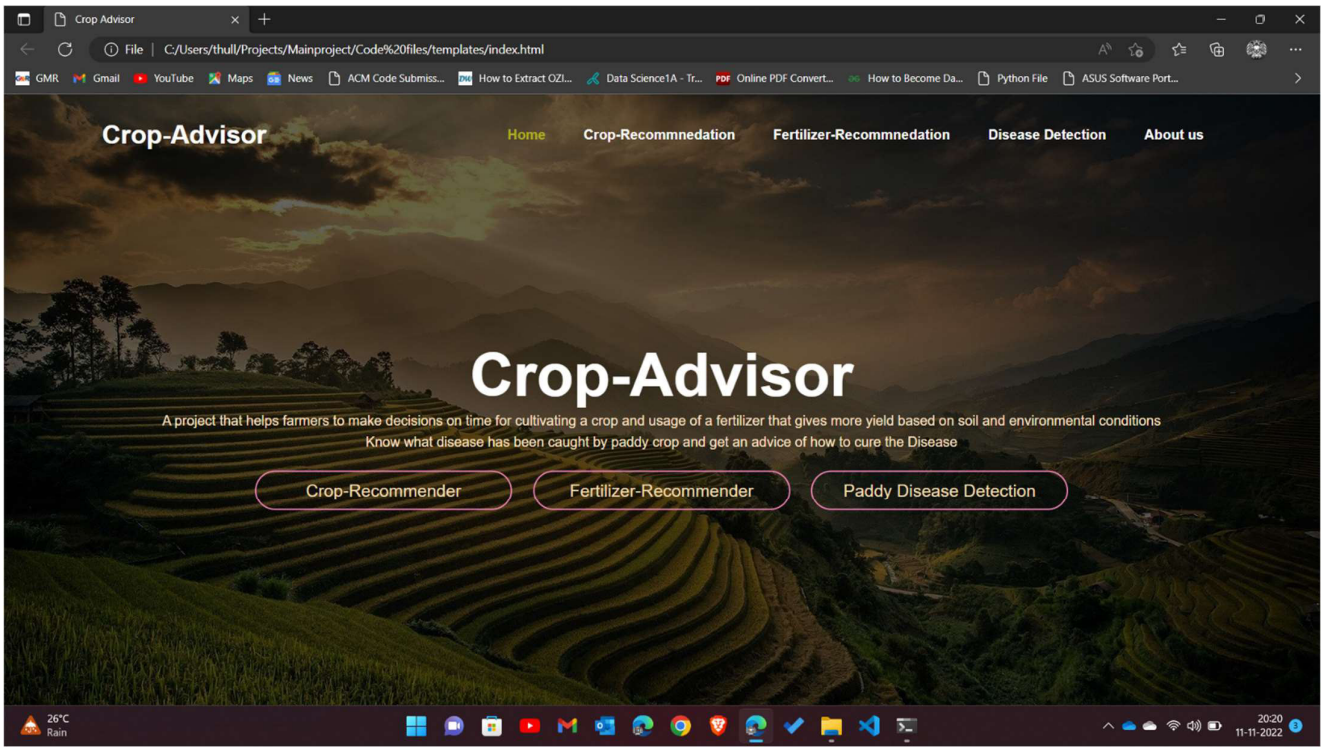
APPENDIX – B

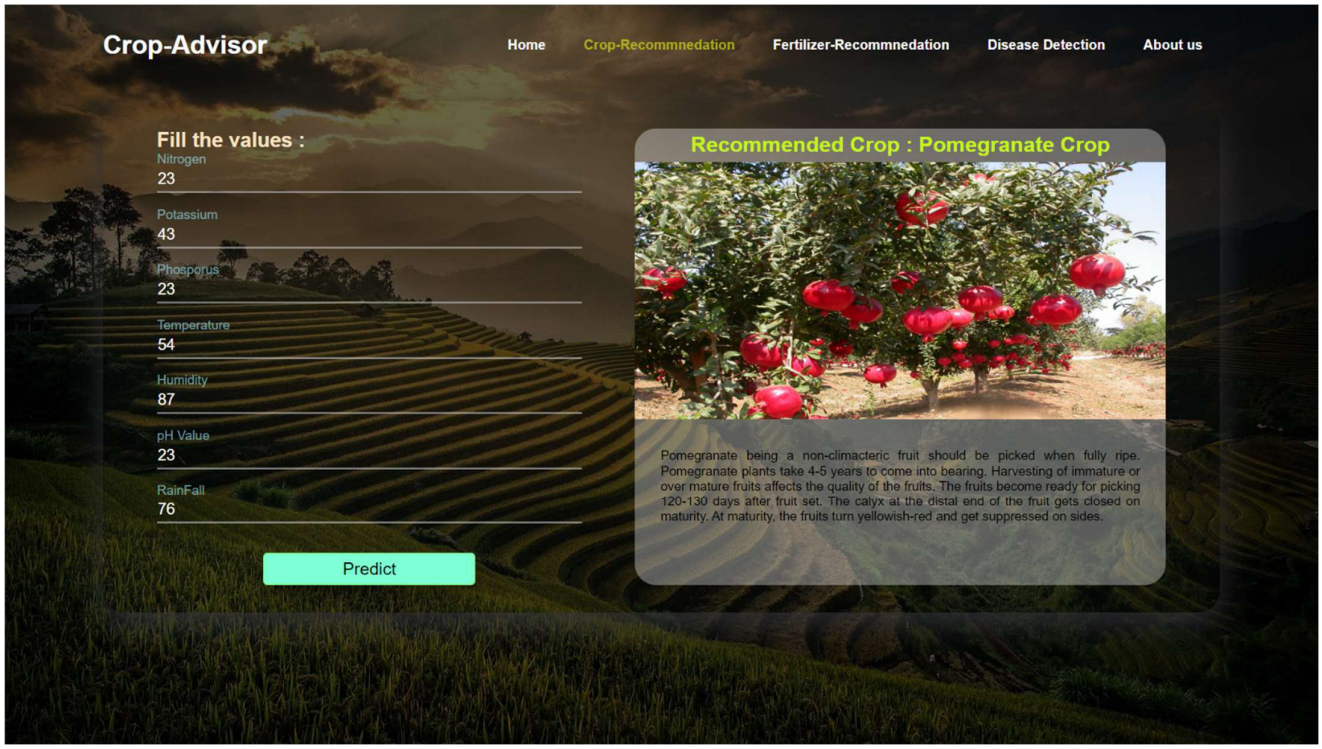

Recycle Bin
Resume
Main Project Raw Files
Mainproject Code
ana
All
Apps
Documents
Web
More
Best match
Anaconda Prompt (anaconda3)
App
Apps
Anaconda Navigator (anaconda3)
Spyder (anaconda3)
Reset Spyder Settings (anaconda3)
Search the web
ana - See web results
anaconda
anaconda navigator
anaconda download
anandabazar patrika
analytics
Anaconda Prompt (anaconda3)
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall
26°C
Cloudy
20:16
11-11-2022

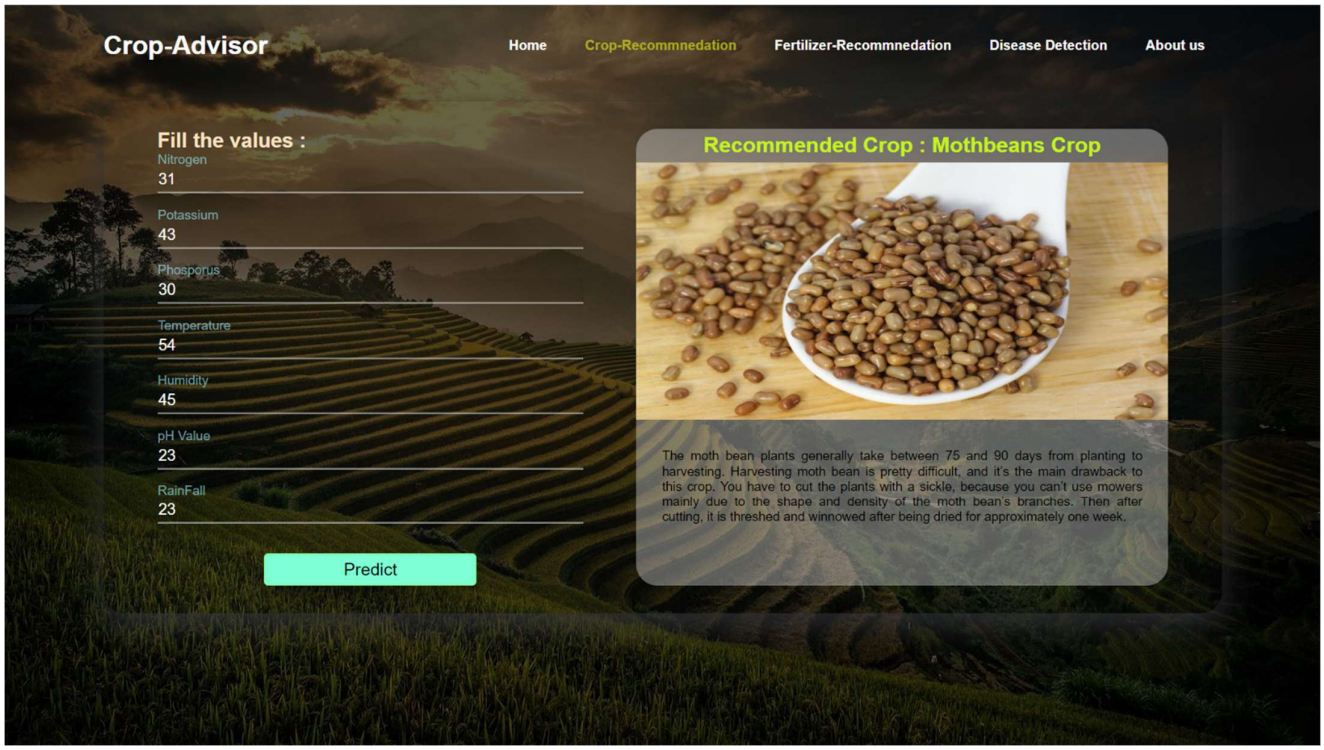

Code files
New
Sort
View
C:\Users\thull\Projects\Mainproject\Code files
Search Code files
Home
Prem Chand - Pe
OneDrive - GMR
Desktop
Downloads
Documents
Pictures
1Data Scienc
Literature Sur
Main Project
Music
Videos
Final Report
Name
Date modified
Type
Size
__pycache__ 10-10-2022 10:33 File folder
Datasets 14-10-2022 01:28 File folder
EfficientNetB0 10-10-2022 10:27 File folder
images 14-10-2022 01:23 File folder
models 10-10-2022 10:27 File folder
templates 11-10-2022 00:06 File folder
input_image 06-11-2022 21:55 JPG File 91 KB
server 08-10-2022 23:21 Python Source File 2 KB
util 07-10-2022 07:43 Python Source File 3 KB
vgg16-paddy-disease-classify 11-10-2022 11:00 Jupyter Source File 1,734 KB
10 items

```
Anaconda Prompt (anaconda:

(base) C:\Users\thull>conda activate gpu

(gpu) C:\Users\thull>cd C:\Users\thull\Projects\Mainproject\Code files

(gpu) C:\Users\thull\Projects\Mainproject\Code files>python server.py
```

```
Anaconda Prompt (anaconda:

WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullIntV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformFullIntV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomGetKeyCounter cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting AdjustContrastv2 cause Input "contrast_factor" of op 'AdjustContrastv2' expected to be loop invariant.
WARNING:tensorflow:Using a while_loop for converting AdjustContrastv2 cause Input "contrast_factor" of op 'AdjustContrastv2' expected to be loop invariant.
Loaded Artifacts done...
Starting Flask server....
 * Serving Flask app 'server'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
```




Crop-Advisor
Home
Crop-Recommnedation
Fertilizer-Recommnedation
Disease Detection
About us
Fill the values :
Nitrogen
23
Potassium
43
Phosporus
23
Temperature
54
Humidity
87
pH Value
23
RainFall
76
Predict
Recommended Crop : Pomegranate Crop
Pomegranate being a non-climacteric fruit should be picked when fully ripe. Pomegranate plants take 4-5 years to come into bearing. Harvesting of immature or over mature fruits affects the quality of the fruits. The fruits become ready for picking 120-130 days after fruit set. The calyx at the distal end of the fruit gets closed on maturity. At maturity, the fruits turn yellowish-red and get suppressed on sides.


Crop-Advisor
Home
Crop-Recommnedation
Fertilizer-Recommnedation
Disease Detection
About us
Fill the values :
Nitrogen
31
Potassium
43
Phosporus
30
Temperature
54
Humidity
45
pH Value
23
RainFall
23
Predict
Recommended Crop : Mothbeans Crop
The moth bean plants generally take between 75 and 90 days from planting to harvesting. Harvesting moth bean is pretty difficult, and it's the main drawback to this crop. You have to cut the plants with a sickle, because you can't use mowers mainly due to the shape and density of the moth bean's branches. Then after cutting, it is threshed and winnowed after being dried for approximately one week.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

Fill the values :

Nitrogen
80

Potassium
43

Phosporus
30

Temperature
24

Humidity
45

pH Value
17

RainFall
52

Predict

Recommended Crop : Watermelon Crop

Watermelons will be ready for harvest 65 to 90 days after sowing depending on the variety. When watermelons are ready to harvest vine tendrils will begin to turn brown and die off. If the tendrils are green the melon is not ripe. A ripe watermelon will make a dull hollow sound when thumped. The soil-side of a watermelon will turn from white to pale yellow when the fruit is ready for harvest. Ripe melons will have a sweet aroma at the stem end. Limit water for a week in advance of the harvest to concentrate sweetness. Watermelons on a single plant will all be ready for harvest over a two week period.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

Fill the values :

Temperature
43

Humidity
23

Moisture
45

Nitrogen
65

Potassium
23

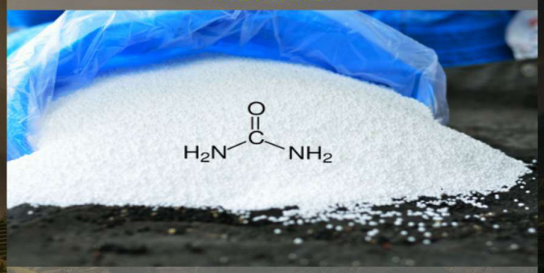
Phosporus
12

Soil Type
Black

Crop Type
Cotton

Predict

Urea Fertilizer



The agricultural industry widely uses urea, a white crystalline solid containing 46 percent nitrogen as an animal feed additive and fertilizer. Here, we'll focus on its role as a nitrogen fertilizer. In the past decade, urea has surpassed and nearly replaced ammonium nitrate as a fertilizer.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

Fill the values :

Temperature
51

Humidity
23

Moisture
45

Nitrogen
12

Potassium
23

Phosporus
78

Soil Type
Loamy

Crop Type
Oil Seeds

Predict

10-26-26 Fertilizer

GROMOR 10:26:26 is a complex fertilizer containing all the three major plant nutrients viz. Nitrogen, Phosphorous and Potassium. GROMOR 10:26:26 contains Phosphorous and Potassium in the ratio of 1:1, the highest among the NPK fertilizers. It contains 7% Nitrogen in the Ammonical form, 22% out of 26% phosphate in the water-soluble form and the entire 26% potash is available in the water-soluble form. GROMOR 10:26:26 is ideally suitable for crops which require high phosphate and potassium and this grade is very popular among the Sugarcane farmers of Maharashtra, Karnataka and Andhra Pradesh and Potato farmers of West Bengal & Uttar Pradesh. GROMOR 10:26:26 is also suitable for Fruit crops.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

Fill the values :

Temperature
10

Humidity
11

Moisture
51

Nitrogen
13

Potassium
13

Phosporus
11

Soil Type
Clayey

Crop Type
Sugar Cane

Predict

17:17:17 Fertilizer

N-P-K 17-17-17 Fertilizer is a complex fertilizer containing all three major plant nutrient ,Nitrogen.Phosphorous.Potassium in equal proportion .contain nitrogen ,phosphorous.potassium in 17:17:17 combination.It also helps regulate metabolic activities n-p-k 17-17-17 fertilizers are water soluble and can be taken up by the plant almost immediately an all purpose 17-17-17 fertilizer will provide the nutrients all plants need for healthy growth.All parts of plants need nitrogen for growth -the roots leaves,stems, flowers and fruits.Nitrogen gives plants their green color and is needed to form protein. A lack of nitrogen causes the lower leaves to turn yellow and the whole plant to turn pale green.Phosphorous is needed for cell division and help form roots flowers and fruits.Phosphorous deficiency causes stunted growth and poor flowering and fruiting.Potassium is important for regulation of water and nutrient movement in plant cells, there by promoting fruiting ,flowering, hardiness and promote the overall health of the plant by providing resistance to diseases. A Potassium shortage show up in various ways but stun growth and yellowish lower leaves.Usage Instruction:seeds and seedling stage use 1/4 tsp to 1 liter water.After 3-4 true leaves stage use 1/2 tsp to 1 liter water . blooming stage use 2/3 tsp n-p-k to 1 liter water.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

Please Upload Image of a Rice plant :

CLEAR | PREDICT

Disease Detected :bacterial_leaf_streak

Cause:
Bacterial leaf streak is caused by Xanthomonas oryzae pv. oryzicola.
Infected plants show browning and drying of leaves. Under severe conditions, this could lead to reduced grain weight due to loss of photosynthetic area.

Prevention:
To prevent and effectively manage bacterial leaf streak:
• Plant resistant varieties.
• Treat seeds with hot water.
• Keep fields clean—remove weed hosts and plow under rice stubble, straw, rice ratoons, and volunteer seedlings, which may be infected by the bacteria.
• Use balanced amounts of plant nutrients, especially nitrogen.
• Ensure good drainage of fields (in conventionally flooded crops) and nurseries.
• Drain the field during severe flood.
• Dry the field during the fallow period to kill the bacteria in the soil and in plant residues.
• In cases of severe infection, when yield may be affected, a copper-based fungicide applied at heading can be effective in controlling the disease.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

Please Upload Image of a Rice plant :

CLEAR | PREDICT

Disease Detected :brown_spot

Cause:
Brown spot is a fungal disease that infects the coleoptile, leaves, leaf sheath, panicle branches, glumes, and spikelets.

Its most observable damage is the numerous big spots on the leaves which can kill the whole leaf. When infection occurs in the seed, unfilled grains or spotted or discolored seeds are formed.

Prevention:
Improving soil fertility is the first step in managing brown spot. To do this:
• monitor soil nutrients regularly.
• apply required fertilizers.
• for soils that are low in silicon, apply calcium silicate slag before planting.

Fertilizers, however, can be costly and may take many cropping seasons before becoming effective. More economical management options include:

• Use resistant varieties.Contact your local agriculture office for up-to-date lists of varieties available.
• Use fungicides (e.g., iprodione, propiconazole, azoxystrobin, trifloxystrobin, and carbendazim) as seed treatments.
• Treat seeds with hot water (53–54°C) for 10–12 minutes before planting, to control primary infection at the seedling stage. To increase effectiveness of treatment, pre-soak seeds in cold water for eight hours.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

Please Upload Image of a Rice plant :

CLEAR PREDICT

Disease Detected :dead_heart

Cause:
Stem borers can destroy rice at any stage of the plant from seedling to maturity.
They feed upon tillers and causes deadhearts or drying of the central tiller, during vegetative stage; and causes whiteheads at reproductive stage.

Prevention:
Improving soil fertility is the first step in managing brown spot. To do this:
• Use resistant varieties.
• At seedbed and transplanting, handpick and destroy egg masses.
• Raise level of irrigation water periodically to submerge the eggs deposited on the lower parts of the plant.
• Before transplanting, cut the leaf-top to reduce carry-over of eggs from the seedbed to the field.
• Ensure proper timing of planting and synchronous planting, harvest crops at ground level to remove the larvae in stubble, remove stubble and volunteer rice, plow and flood the field.
• Encourage biological control agents: braconid, eulophid, mymarid, scelionid, chalcid, pteromalid and trichogrammatid wasps, ants, lady beetles, staphylinid beetles, gryllid, green meadow grasshopper, and mirid, phorid and platystomatid flies, bethylid, braconid, elasmid, eulophid, eurytomid and ichneumonid wasps, carabid and lady bird beetles, chloropid fly, gerrid and pentatomid bugs, ants, and mites, earwigs, bird, asilid fly, vespid wasp, dragonflies, damselflies, and spiders.
• Bacteria and fungi also infect the larvae; mermithid nematode, chalcid, elasmid and eulophid.
• Apply nitrogen fertilizer in split following the recommended rate and time of application.

Crop-Advisor

Home | Crop-Recommnedation | Fertilizer-Recommnedation | Disease Detection | About us

SATHISH ROUTHU
19341A1296
Department of Information Technology
GMR Institute of Technology
Rajam, Srikakulam, AP

T PREM CHAND
19341A12B7
Department of Information Technology
GMR Institute of Technology
Rajam, Srikakulam, AP

G S RAM PRUDHVI
19341A1297
Department of Information Technology
GMR Institute of Technology
Rajam, Srikakulam, AP

M VYAS KANMANI
19341A1268
Department of Information Technology
GMR Institute of Technology
Rajam, Srikakulam, AP

T DANIYA
PROJECT SUPERVISOR
Department of Information Technology
GMR Institute of Technology
Rajam, Srikakulam, AP

APPENDIX - C

APPENDIX – C

ICAISS

International Conference on
Augmented Intelligence and Sustainable Systems

ICAISS-2022 ★ 24-26, November 2022 ★ Trichy, India

http://icaiss.in/ | icaiss.confdesk@gmail.com

Letter of Acceptance

TO

Routhu Sathish, Thulluru Prem Chand, Sai Ram Prudhvi Gummaluri, Marripudi Vyas Kanmani, T. Daniya
GMR Institute Of Technology Rajam, Andhra Pradesh

Herewith, the conference committee of the International Conference on Augmented Intelligence and Sustainable Systems ICAISS 2022 is pleased to inform you that the peer reviewed research paper **"Paper ID: ICAISS138"** entitled "**A Review on Smart Farming Based on Soil and Environmental Factors with Deep Learning Techniques**" has been accepted for oral presentation as well as it will be recommended in ICAISS Conference Proceedings.

ICAISS will be held on 24-26, November 2022, in CARE College of Engineering, Trichy, India. ICAISS encourages only the active participation of highly qualified delegates to bring you various innovative research ideas.

We congratulate you on being successfully selected for the presentation of your research work in our esteemed conference.

Yours' Sincerely

Dr. A. Pasumpon Pandian
Conference Chair

A Review On Smart Farming Based On Soil And Environmental Factors With Deep Learning Techniques

Routhu Sathish
Information Technology
GMR Institute Of Technology
Rajam, Andhra Pradesh, India
sathishrouthu296@gmail.com

Thulluru Prem Chand
Information Technology
GMR Institute Of Technology
Rajam, Andhra Pradesh, India
thulluru.premchand@gmail.com

Sai Ram Prudhvi Gummaluri
Information Technology
GMR Institute Of Technology
Rajam, Andhra Pradesh, India
ramprudhvi1409@gmail.com

Marripudi Vyas Kanmani
Information Technology
GMR Institute Of Technology
Rajam, Andhra Pradesh, India
vyaskanmani00777@gmail.com

T. Daniya
Information Technology
GMR Institute Of Technology
Rajam, Andhra Pradesh, India
daniya.t@gmrit.edu.in

***Abstract*— India's global economy is critically dependent on agriculture, which also accounts for a sizable portion of GDP. This paper provides a review of the use of machine learning and deep learning techniques in agriculture and elaborates different systems developed for smart agriculture including crop recommendations, fertilizer suggestions, categorization procedures and diseases identification by images. Climate factors like temperature, rainfall, soil quality, and fertilizers are the key determinants of agricultural productivity. Decision making such as determining the diseases and applying solutions to them is an important step in agricultural activities. With advanced technologies evolved in this digital world we can build more sustainable systems to make these farming procedures automated. Deep learning techniques are giving very prominent results in solving problems of predictions and decision making. The application of machine learning algorithms that are used in solving different problems that are arising in farming process are discussed. This paper contributes to know the available techniques for the automated farming procedures.**

***Keywords*—Agriculture, Climate factors, Deep Learning, Farming, Image Processing**

I. INTRODUCTION

Artificial intelligence is a fast growing technology that tries to eliminate manual operations and Artificial intelligence is a fast growing technology that tries to eliminate manual operations and replace them by machines in many fields. This technology has evolved in many sectors such as medical, defence, Agricultural, services and many more. In India many workers depend on agriculture as their main profession. There are many factors affecting the production in farming of crops in India. These factors can be of environment or soil. To make certain decisions in farming we need to automate the process in such a way that the production rate in the field is increased. There are many available techniques for smart farming such as crop yield predictions, crop selections, fertilizer recommendations, weed detection in the field, diseases identification of crop leaves, pesticide suggestion, seed selection and many more techniques helps famers to get a better production results. The primary goal of this paper is to examine the available smart farming techniques in terms of feasibility, efficiency, and effectiveness.

This paper studies more frequently faced problems in farming and deep learning techniques used to solve them. The relevant study describes the regular workflow of the machine learning procedures to build an efficient model.

II. LITERATURE SURVEY

Here we have gathered several periodicals that have conducted research on our connected work, which is based on the rice plant, and we have separately summarized each work as shown below.

In this paper, computer vision and deep learning techniques are used to select and plant healthy billets in order to increase sugarcane crop growth and hectare yield. The data set for the training model was obtained from the USDA Sugarcane Research Farm. The sugarcane dataset is based on three varieties of sugarcane collected from farms and classified by experts as good and damaged billets. When processing large datasets, highly regarded CNN models like AlexNet, VGG-16, GoogLeNet, and ResNet101 are used to achieve better results. Each dataset was randomly divided into three types to compare these models: 60%, 20%, 20% for training, validation, evaluation. A two-step transfer approach is suggested in this study to create an automatic system which every farmer can apply to any wide range of sugarcane. In this case, the four architectures selected produced similar results (TNR, TPR, MCC, and proportion of great billets per plant), but ResNet101 and AlexNet produced great outcomes for various range of renewable. ResNet101, on the other hand, demanded an unusually high computational cost. AlexNet appears to be the ideal choice for them, with a two-stage transfer learning process that requires approximately 22 times fewer logs to train up the system for

a fresh sugarcane variant. The created model can be applied to any variety of sugar cane. Using the model allows farmers to grow quality billets, thereby increasing sugarcane production rates in the future, but due to the damaged variety of sugarcane billets classified by humans, the initial training dataset may not be accurate. In addition to using high-quality billets, sugarcane quality control also depends on soil and climate. [1]

This paper describes deep learning techniques for detecting insect pests in fields. Two types of algorithms are mainly used: One-stage algorithm: YoloV5, two-stage algorithm: FasterRCNN & MaskRCNN. One-step algorithms directly extract features from images and detect object locations and classify them. These will print (Object coordinates like bounding boxes) in the image and then classify that bounding box. these are relatively fast as there is no need for regional design. However, two-phase algorithms generate a region of interest for objects in the image (such as masks for the object) and then extract features from them to classify them. These models are relatively fast as an intermediate stage of searching for included regions. However, one-step algorithms performed faster and better on simple datasets with less complexity. Two-phase algorithms performed better on datasets with complex backgrounds. Yolov5 can more accurately identify insect pests on images with simple backgrounds, such as the Baidu AI insect detection dataset, but manual labeling is time-consuming and requires a lot of computing power to train these complex models. Accuracy is on average 99%. [2]

This paper proposed a Deep Recurrent Q-Network models of Reinforcement learning in order to estimate the yield of a crop with given crop parameters. They collected data from paddy crops in Tamilnadu. Data parameters include: (1) The production of paddy is estimated on the basis of cultivable land, paddy growth, and yield obtained. (2) Several independent features include climatic factors, soil parameters and distinct hydro-chemical characteristics of underground water. DQRN model, Reinforcement Learning, and Q-Learning are the techniques used. DRQN-based processes provide a satisfactory explanation which incudes a non-linear mapping between agricultural output, Weather conditions, soil, and underground water factors independently. This benefit can reduce the need for previous knowledge and expert dependence when making agricultural yield predictions but Gradient Exploding in RNN model for long time series data and Statistical uncertainty with non probabilistic model.[3]

This paper included disease identification in lemon and mango plants using image segmentation techniques and the appropriate fertilizer. Several stages are used to identify diseases, such as Image acquisition, Image Pre-processing, Classification by Artificial Neural Network, Fertilizer recommendation by Mapping standard deviation to the fertilizers available. To figure out the amount of disease infection as in leaf, proposed system divides the input leaf image into diseased and background areas using the K means clustering algorithm, Feature extraction by using Gray Level co-occurrence Matrix (GLCM) along with first order statistical moment's method, Fertilizer recommendation based on disease detected but Segmentation provided some misclassifications. Here, the proposed system simply recommends the chemical fertilizers that were applied. However, the amount of dosage of fertilizer use is not mentioned.[4]

Images from the input plant are first pre-processed to create them suited about further handling. Pre-processed images have been routed through a Image Segment module, which employs SegNet to segment the pre-processed image. Pre-processing is done to improve the contrast of the image, remove noise and disorders in order to detect plant diseases. The CNN features extract statistical data from each segment. These features are then put together into a feature vector. The pre-processed paddy image is sent to the segment module using SegNet can supply high-dimensional data segmentation and to identify disease regions taking into account each segment. Deep RNN receives the extracted features for classification and trains on the proposed RSW, which combines ROA, SMO, and WWO. It is proposed to build a classifier known as RSW-based Deep Recurrent Neural Network by adjusting the Deep Recurrent Neural Network training algorithm including the Recursive Smith-Waterman-seq algorithm. The proposed Recursive Smith-Waterman-seq is used to optimize the weight of the classifier. To select the best weights, Recursive Smith-Waterman-seq modifies the Deep Recurrent Neural Network by integrating the Read-While-Write and Sequential Minimal Optimization algorithms. The goal of the proposed Recursive Smith-Waterman-seq-based Deep Recurrent Neural Network is to identify disorder using features extracted from the source images. The proposed Recursive Smith-Waterman-seq based Deep Recurrent Neural Network produces high results: With Maximum Accuracy, Sensitivity, Specificity (90.5%, 84.9%, 95.2%) respectively. However, the proposed method cannot detect all rice diseases. [5]

In this paper to forecast the diseased area of the leaves, they suggested improving machine learning. The infected area is divided using a colour-based segmentation model defined by them. On sample images, experimental evaluations of the time complexity and the infected area were performed. Images can be processed to identify plant diseases. Image acquisition (virtually increase the number of samples in your dataset using data you already have), steps in the disease identification process include image pre-processing, image segmentation, feature extraction, and classification. Image augmentation is possible to achieve by applying geometric transformations, altering the colour, brightness, contrast or by adding noise. To enable model to adapt to various situations where noisy data could occur, such as during the rainy season. The dataset is gathered from UCI machine learning repository it comprises three leaf diseases. 20% of the images were used for validation and testing, while the remaining 80% were used for training. Classification Algorithm used : SVM. When model is trained with images that have undergone image augmentation or acquisition, it performs 5% better on the validation set than a model that is trained with images that have not. The model used in this paper is only relevant to three diseases and its usefulness to other diseases is not guaranteed.[6]

A strong method ExpRHGSO-based Hybrid Deep Learning, a combination of DRN and DFDN is developed for disease identification and leaf classification. Initially, images are retrieved out of a database that combines two data sources to create a single dataset. The ROI extraction is then applied to the input images for pre-processing. Following that, image segmentation is performed using DFC to segment appropriate regions. Statistical, CNN, and texture features are extracted for subsequent processing. To improve the detection and classification processes, data augmentation is used to

enhance the dimensionality of the features. Utilizing rotation, cropping, and zooming, the data is improved. The DNFN classifier is used to carry out the rice leaf detection process. This classifier reduces the problems associated with computational complexity while achieving the ideal result more quickly. The developed ExpRHGSO algorithm, which combines EWMA and RHGSO, is used to train the DNFN model. The EWMA algorithm detects slight changes in the way the target values are processed since it has the ability to produce accurate result. RHGSO is integrated with the EWMA to produce better results. After the identification of the rice leaf disease, the BLB, Blast, and Brown spot. DRN is used for rice leaf disease classification. In the DRN training process, the ExpRHGSO method is included. With higher values of 0.916, 0.923, and 0.919, In terms of testing accuracy, sensitivity, and specificity, the created ExpRHGSO-based Hybrid Deep Learning technique performed better. On the basis of testing accuracy, sensitivity, and specificity, the model demonstrated excellent performance. When tested against huge datasets with a greater variety of diseases, the model might not keep the same accuracy and other standards. [8]

In this study, an automated approach is proposed for the detection of 3 main paddy leaf diseases. Depending on the severity of the diseases, pesticides and fertilizers are also suggested. The goal of this study is to provide a reliable and straightforward system for categorizing paddy leaf diseases. To distinguish the damaged part from the paddy leaf image, K-means clustering is used. These disorders are categorised using the visual content's colour, Shape and Texture. SVM classifier recognizes the type of paddy leaf diseases. After identification, a preventative solution is recommended that can assist individuals and organizations involved in agriculture in responding appropriately to these diseases. In this study, it is proposed that the following treatments for Brown spot are effective: Treating diseased plants with pesticides and herbicides such Benzoyl and Iprodione and antibiotics like Validamycin and Polyoxin. Systemic fungicides, such as triazoles and strobilurins, can be used sparingly to reduce leaf blast. To effectively control the illness, apply a fungicide at heading. Foliar spray of Streptocycline and Copper Sulfate for bacterial leaf blight. The overall accuracy is about 92%. Suggested pesticides or fertilizers are divided into 2 categories one is at initial stage and another at severe stage. If we use different techniques then the accuracy may increase. [9]

For years, rice diseases have been one of the numerous enduring problems that farmers and planting specialists have had to deal with. In the world of agriculture, a quick, automated, cheap, and trustworthy method to identify rice infections is desperately needed. Because severe rice diseases could result in no harvest of grains. Deep learning has proven to perform well in image processing and classification challenges, thus they used this approach in this research to address the problem. In order to address the vanishing-gradient problem that arises as network depth grows, the deep convolutional neural network architecture known as DenseNet links the result among all layers to the source among all subsequent layers in inside dense block. The recurrence of a Batch Normalization for a specified number of times is known as a dense block. Each layer's feature-maps are utilized as sources and its own feature-maps are employed as sources for all subsequent layers, minimizing feature loss. The easiest way to enhance the capacity of deep neural networks is to raise their depth or the networks' width. However, as network depth and width have increased, the number of factors has increased as well. This increases the use of computational resources. In order to solve these issues, the Inception used was a module. The module, which consists of a max-pooling layer and convolutional layers of different sizes that are stacked together and utilized to do the dimension reduction, has to be combined using a concatenation filter. The traditional DenseNet was altered by the addition of an Inception module and a brand-new, fully-connected ReLU layer with 512 neutrons, which was then replaced with a global pooling layer to replace the connection layer entirely. Since it integrated the advantages of both, the DenseNet pre-trained on ImageNet and Inception module was chosen to be utilized in the network. This method outperforms other cutting-edge approaches, and the outermost layer employs the Softmax algorithm to classify images of rice disease. DENSE-INCEP network model is the algorithm we are using to solve the above problem. [10]

Every nation is built on agriculture. As a result, it needs to be timely monitored. Farmers have so far used traditional farming methods. These methods took a lot of time and were ineffective because they were not exact. Precision farming includes a variety of techniques such as weather forecasting, soil analysis, crop recommendations, and calculating the necessary dosages of pesticides and fertilizers. Precise Farming gathers data, trains systems, and forecasts outcomes using cutting-edge technologies like Internet Of Things, Data Mining and Analytics, and ML. This research suggests four modules: crop prescription, weed detection, pesticide advice, and farm budget. In order to find the optimal model for analysis, they used a total of 11 algorithms and In this work, hyperparameter adjustment has been used to maximize the model potential. The model is saved in a pickle file after being determined to be the best crop prediction model so that it may be quickly summoned and utilized to forecast crops. To classify the images in the weed identification module, the Resnet keras model and Deep Residual Networks using skip connection technique were utilized. This allowed DL models to be trained without the problems that disappearing gradients generate. This module had 25 epochs run, with a batch size of 32. It resulted in accuracy of 89%. The pesticide recommendation system is similarly modelled after the herbicide recommendation system based on weeds. This module had 20 epochs run, with a batch size of 32. The achieved accuracy was 98%. Pesticides cannot be used on any crop in the same way that herbicides are. Both the soil fertility and crop growth could be harmed. Crop names enter by user and predicted insect names are sent to the Random Forest Classifier to ensure that the correct pesticides are recommended. The numerous cost ideas and the total cost of cultivation are estimated in the next module of the cost estimating procedure. For this module, eight different datasets that span the years 2010 to 2018 have been loaded. These datasets are combined then loaded. The state and crop columns both lack in numerical values. Label Encoding is employed to produce numeric data for a given item. The year, crop name encoded, and state name encoded columns are provided to the ML and ensemble regressor models. The R2 score, RMSE, MSE, MAE are calculated for each regressor model. With the highest R2score, the XGBoost regressor is selected as the effective model. [11]

The objective of this study is to support an individual in efficiently cultivating crops in order to attain high yield at a reasonable price. For recommending a crop to the user, ML algorithms like Neural Network,

Naive Bayes, Decision Tree, K Nearest Neighbor, regression model, and SVM were employed. Instantly retrieved information on the temperature and humidity is fed into the most basic model, which consists of 10 procedures with hyper parameter adjustment. To obtain accurate prediction, the Resnet algorithm is used. Since the Resnet method uses layers like the skip layer, identity layer that attempt to make the source image into the output itself, it attempts to achieve more accuracy. Predictions is therefore 100% accurate. Although te CenterNet algorithm is a method for identifying weeds on the ground, Resnet152V2 aims to produce accurate prediction. This study has defined a method for taking into account a picture's features with a relevant insect dataset. Its major objective is to find a key that aids in classifying the classes. This proposed work helps to clarify the significance of a secret in identifying the many insect classes. Resnet model is thus used in the proposed model to classify data. The suggested model aids in insect detection and also recommends pesticides in this regard. The suggested method employs ensemble regression techniques to predict prices through 2028. It offers a comparison analysis of a crop for a chosen state from 2010 to 2028. It offers a detailed look at production costs, flat rate, and overall cost. [12]

The objective of this study is to reevaluate research regarding the use of ML techniques in crop production. A few techniques, such as artificial neural networks, decision trees, fuzzy information networks, and Bayesian belief networks. Statistical analysis, the k nearest neighbours, Markoff process model, k-means clustering, k nearest neighbours, and SVM were all used in the context of agriculture. To forecast corn yields, nonlinear regression is used. For predicting cotton yields from surveys, the Markoff process approach is used. For estimating the grain yield of maturing rice, regression toward the mean is used. Belief Networks are used to produce future crops. Crop yield prediction uses neurofuzzy modelling. Second Order Markov Chains is employed for Forecasting of Crop Yields. Factors Influencing the Growth of Cultivated Varieties in Crop Margins Use Polynomial Regression For seasonal to cross weather forecasting, stochastic and probabilistic forecasting approaches are used. For the purpose of predicting sugar production from populations of samples, k-nearest neighbour algorithm is used. For the purpose of predicting corn and soybean production, ANN are used. Neural Network is employed for Rice Crop Monitoring. Artificial Neural Networks is employed for Forecasting Thailands Rice Export. In Andalusia, olive farming is carried out using an advisory and a network of mathematically huge information. Climate-related variables are used in regression to estimate sugarcane production. Decision-tree algorithms are employed for modelling soya productivity. For crop selection, a fuzzy modelling of decision network is used. Crop yield forecasting makes use of statistical techniques. India uses data mining with climate variables to increase jowar crop yield. For the purpose of predicting apple crop yield, fuzzy cognitive map training is used. Models using regression and NN are used to predict agricultural output. To predict crops, KMeans clustering is used. Farm Crop Production Prediction uses Artificial Neural Network Approach. Rotations are used for crop development [13]

Explain how a soil nutrient test has been carried out to ascertain any plant's productivity levels in this paper. Variables related to water and climate were promptly discussed, and their correlations with soil quality are plotted appropriately. All of these elements have an impact on crop productivity. Strategies for the consequences of a maize-chickpea nutritional system were also established. For increased system accuracy, the soil categorization is classified using a knowledge base algorithm. [15].

This paper proposes a blockchain-enabled wireless network-based sustainable smart farming system, whose performance is improved by maximization of SIR or SNR values for selection of an ideal relay on an end-to-end basis. When choosing the optimal relay performance with and without interference, the overall communication throughput (OCT), power splitting relaying (PSR), time switching relaying (TSR), and transmission success rate (TRS) are also determined. Numerical simulations confirm the theoretical values of accuracy. For the purpose of evaluating the performance of the proposed wireless networks for sustainable smart farming equipped with blockchain technology, the OCT, PSR, TSR, and TRS values are estimated. [16]

III. RELEVANT STUDY

Analysis of different techniques to build a system for following smart farming techniques :

A. Plant classification : Some techniques have been used to classify plants for some given conditions and they have applied them to farm the fields with those plants to get better results.

B. Disease detection : This technique involves the identification of disease that affected the crop by seeing an image of the plant or their leaves. However it mainly consists of CNN and deep learning architectures to classify the diseased images. It give the type of disease that occurred among the available classes of diseases.

C. Yield estimation : To know the estimated production rate of crop that is being farmed this regression technique will helps to predict the approximate value by using machine learning and deep learning technique of regression.

D. Pesticides recommendation : Pesticides help to eliminate the pest that damages the crops in agriculture. We have to know the appropriate pesticide for the pest that observed in the crop. Recommendation of the pesticides will help the farmers to make decision on usage of pesticides.

E. Weed detection : Weeds compete with crops for sunlight, water, nutrients, and space. In addition, they harbor insects and pathogens, which attack crop plants. We need to detect and identify these weeds to use suitable solution to prevent weed growth in the fields. Weeds can be detected by their images using deep learning techniques.

DEEP LEARNING :

Artificial intelligence is evolving and expanding it’s art of learning to all branches of the current technical world. The latest developed computational resources are helping to achieve more efficient results in this AI Technology. Machine learning is a sub-branch of AI, it uses self-learning approach to derive meaningful insights from presented data without manual rules. Deep learning is a type of machine learning that uses neural networks to build more complex models. At the earlier stages of AI they have been limited in terms of computational resources. But with latest advancements we can build more complex

and large computational models by using deep learning to process large data effectively. As the name suggests AI Neural networks imitates the human brain nerve mechanism to learn from presented data. Deep learning has many variations based on the given input data such as images, text, numerical etc. Convolutional Neural Networks are used for processing image data and their usage has been applied to solve many computer vision problems. CNN process the input data by applying many convolution functions and filters to extract features from the given input images.

Data Acquisition **:** As the machine learning or deep learning models require a lot of data to extract information and learn from them to solve problems we have to collect the proper data from different sources.

Data Pre-Processing **:** This step is very important in the process of model building to solve a problem. The acquired raw data may consists of noise, missing values, outliers or any other irrelevant data. We need to process this raw data in such a way that the model can learn effectively. This pre-processing step involves several techniques:
For a textual or numerical data :

- We need to replace the missing values with proper corresponding measures of central tendency or other ways of imputations.
- The outliers which will results in wrong predictions needs to be handled.
- Scaling of data needs to be done as the continuous values maybe measured at different scales.
- Encoding of categorical values or string type variables.

And there can be several other pre-processing steps such as data deduplication, data distribution transformations etc. needs to be done as per the observed data.
For an image data :

- ***Data Augmentation :*** The data augmentation helps in visual transformations. The transformation that is mainly focused on image classification are flipping, color modifications, cropping, rotation, geometric transformation, padding, re-scaling, zooming, gray scaling, darkening and brightening, random erasing etc. When we don't have a large image dataset, it's a good practice to artificially introduce sample diversity by applying random, yet realistic, transformations to the training images, such as rotation and horizontal flipping. This helps expose the model to different aspects of the training data and reduce overfitting.
- ***Image Resizing and Rescaling :*** It is better to resize all the training images to the same size as the model learns in more efficient way. Rescaling makes all the numerical values in a standard scale of measure.
- ***Image Segmentation :*** it is a process of partitioning the image into multiple segments knowns as regions (Collection of pixels) to simplify the representation of an image and extract information easily. It extracts the boundaries, edges, regions, contours and such type of information from the image. It will group the pixels such that the same group pixels has certain common characteristics.

Feature Extraction : Feature extraction is a dimensionality reduction technique by which a raw data is reduced to some groups known as features for processing. Large datasets need more computational power to process them and very difficult to extract accurate information from them. Feature extractions is a collection of methods those tries to reduce the amount of data without loosing much information from the raw data. It has many benefits like removal of redundant data, reduction computational resources and it facilitates the speed of training, generalization. Principal Component Analysis and Linear discriminant analysis are linear dimensionality reduction techniques frequently used in many machine learning procedures. Fig1 explains the general workflow of Machine Learning models development.

Model Evaluation **:** Models should be evaluated in such a way that the they learned more generalized and give test results also accurate. Accuracy score is the ratio of total no. of correct predictions to total no. of samples.

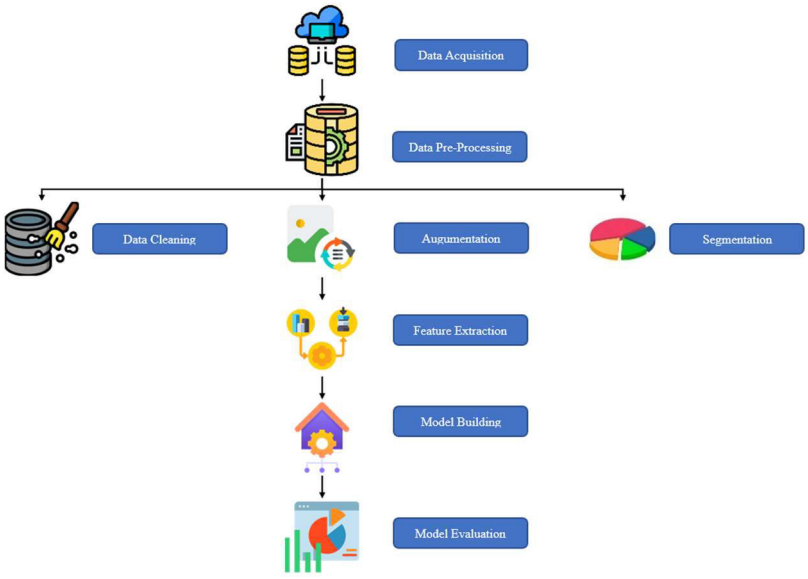


Fig1 : Regular workflow of ML Modelling

Table 1 : Comparison of different Techniques used in Smart farming

Ref Author's Name and Year	Dataset Used	Crops	Model / Techniques Used	Estimated Variables / Research Approaches	Advantages
[1] Moises Alencastre-Miranda, Richard M Johnson and Hermano Igo Krebs (2020)	USDA Sugarcane Dataset	Sugarcane with varieties HoCP 09-804, HoCP 96-540, L 01-299	Transfer Learning Process CNN Architecture model	Healthy Sugarcane billets Classification	Valid to any Sugarcane variety and enables farmers to plant quality billets
[2] W. Li, T. Zhu, X. Li, J. Dong and J. Liu (2022)	Baidu AI, IP 102 Datasets		Faster RCNN, Mask RCNN, YOLO v5	Pests Identification	Yolov5 can more accurately identify insect pests in images with a simple background
[3] Dhivya Elavarasan and P.M. Durairaj Vincent (2020)	Custom dataset is used	Paddy	DRQN Model	Using climatic, land, hydrological, and agricultural characteristics to predict agricultural yield	When predicting agricultural productivity, the use of DRQN-based procedures eliminates the requirement for prior information and expert dependence.
[4] K.S Neethu and P. Vijay Ganesh (2017)	Custom dataset is used	Lemon, Mango	K means clustering algorithm	Disease detection	Based on the disease detected, fertilizer also recommended.
[5] T. Daniya and Dr.S. Vigneshwari (2021)	Rice Grain disease Dataset	Rice	RideSpider Water Wave Based Deep Recurrent Neural Network	Disease detection	The model was able to achieve its highest levels of precision, sensitivities, and selectivity.
[6] R. Salini, A. Farzana and B. Yamini (2021)	Dataset that contain classes of leaf smut, bacterial leaf blight and brown spot		SVM Classifier, custom colour-based segmentation model	Disease detection & Fertilizer recommendation	
[7] Syeda I Hassan, Muhammad M Alam, Usman Illah, Mohammed A Al Ghamdi, Sulthan H Almotri and Mazliham M Suud (2021)			Approaches utilizing the Internet of Things, spatial information, NIR, RGB, multispectral and ML	crop illnesses, weed control, pesticides management, drainage management	Give a succinct overview of the advances in smarter agricultural on the efforts of many researchers.
[8] T. Daniya and Dr.S. Vigneshwari (2022)	Rice Plant Dataset, Rice Plant Disease Dataset	Rice	ExpRHGSO-based Hybrid Deep Learning	Disease identification	In terms of precision sensitivities, and selectivity, great results were achieved.

[9] Farhana T Pinki, Nipa Khatun and S.M. Mohidul Islam (2017)	Paddy Leaf Disease Dataset	Paddy	K-means clustering, SVM Classifier	Disease Detection & Fertilizer Recommendation	Achieved overall accuracy about 92%
[10] J. Chen, D. Zhang, Y.A. Nanehkaran and D. Li (2020)	Custom Dataset prepared based on Rice Plant Diseases	Rice	DENS-INCEP network model	Disease Detection	With usage of Image acquisition technique model achieved average accuracy of 98.63%
[11] S.K.S Durai and M.D Shamili (2022)	Crop recommender, soil, scientific_names datasets from Kaggle	Different Corps	XGBoost regressor, Random forest classifier, Randomized CV, and RESNET152V2 pre-trained algorithm are used to optimise the Random forest classifier.	Weed detection, pesticide prescription, agricultural cost planning, and crop suggestions	Not just predicting weeds and pests, but also recommending the pesticides and herbicides
[12] Rayner Alfred, Joe H Obit, Christie P Chin, Haviluddin and Yuto Lim (2021)	Datasets from Lens website	Different Corps	Linear Regression Model, Decision-tree, KNN, NN and SVM	Crop Recommendation, Weed Identification, Pests Identification	With the usage of Hyperparameter tuning model received accuracy of 95.45%
[13] S. Mishra, D. Mishra and G.H. Santra (2016)		Soya bean, Rice, Sugarcane	Decision-tree, Markov-Chain Model, k-means clustering, KNN and SVM etc.	Crop Yield Prediction based on Climate forecasting	
[14] T. Daniya and Dr.S. Vigneshwari (2019)	Custom Dataset prepared based on Rice Plant Diseases	Rice	ML, image processing & segmentation techniques	Disease recognition	Provides a comparison of the many techniques used to identify rice diseases
[15] H. James Deva Koresh (2021)		Maize, Chickpea	Soil Evaluation and testing techniques like Machine driven analysis, Mineralogical testing, Agrochemical analysis etc.	Soil Quality, Soil Helath Metrics, pH values of Soil	Water and climate variables were quickly discussed, and their relationships with soil quality are correctly plotted. These factors all affect how productive a crop is.
[16] J. Sustain, DR. D. Sivaganesan (2021)			Wireless blockchain based network	OCT, PSR, TSR and TRS	With more relay nodes, OCT and TRS work better, and numerical simulations are used to confirm the theoretical concept's accuracy.

IV. COMPARATIVE STUDY

In order to identify pests, recommend crops based on soil and climate conditions, and recommend fertilizers and herbicides based on disease and weed, the table above i.e Table 1 provides a brief comparison of the various Deep Learning and machine learning approaches utilized in the agriculture area. This is a summary analysis that was created from research published in a variety of periodicals, focusing on the different problems solved in agricultural farming activities by using machine learning and deep learning .

V. CONCLUSION AND FUTURE SCOPE

Agriculture field always need to be developed according to the new technologies in the era. By integrating these technologies into this farming field will be highly beneficial to farmers. However more research need to be done in this sector to build more feasible solutions to the problems faced by many farmers. Solving the real time problems frequently encountered by farmers need to be solved by using the advanced techniques, will be the future scope of this paper. This paper

makes two significant contributions. The first step is to provide a thorough study of research publications that make use of the Deep Learning technique. Each paper's originality, models, and experiments are elaborated. The second step is to know what are the general machine learning procedures to build efficient systems for smart farming. The most common smart farming techniques used are plants classification, yield estimation, disease detection and many other. Deep learning helped to build more efficient systems for such class of problems. These DL Techniques were used in almost all relevant problems and gave better results in real time.

VI REFERENCES

[1] M. Alencastre-Miranda, R. M. Johnson and H. I. Krebs, “Convolutional Neural Networks and Transfer Learning for Quality Inspection of Different Sugarcane Varieties”, in IEEE Transactions on Industrial Informatics, Volume 17, Issue 2, pp. 787–794, February 2021.

[2] Wei Li, Tengfei Zhu, Xiaoyu Li, Jianzhang Dong and Jun Liu, “ Recommending Advanced Deep Learning Models for Efficient Insect Pest Detection”, in MDPI on Application of Machine Learning in Agriculture , Volume 12, Issue 7, 1065, pp. 1–17, June 2022.

[3] D. Elavarasan and P.M.D. Vincent, “Crop Yield Prediction Using Deep Reinforcement Learning Model for Sustainable Agrarian Applications”, in *IEEE Access*, Volume 8, pp. 86886–86901, April 2020.

[4] K.S Neethu and P. Vijay Ganesh, “Leaf Disease Detection and Selection of Fertilizers using Artificial Neural Network”, in IRJET on Computer Science Journal, Volume 4, Issue 6, pp. 1852–1858, June 2017.

[5] T. Daniya and Dr.S. Vigneshwari, “Deep Neural Network for Disease Detection in Rice Plant Using the Texture and Deep Features”, The Computer Journal, Volume 65, Issue 7, pp. 1812–1825, July 2022.

[6] R. Salini, A. Farzana and B. Yamini, “Pesticide Suggestion and Crop Disease Classification using Machine Learning”, in IRJET on Computer Science Journal, Volume 11, Issue 4, pp. 27997–27999, April 2021.

[7] S.I. Hassan, M.M. Alam, U. Illahi, M.A. Al Ghamdi, S.H. Almotiri and M.M. Su’ud, “A Systematic Review on Monitoring and Advanced Control Strategies in Smart Agriculture”, in *IEEE Access*, Volume 9, pp. 32517–32548, January 2021.

[8] T. Daniya and Dr.S. Vigneshwari, “Exponential Rider-Henry Gas Solubility optimization-based deep learning for rice plant disease detection”, in Springer, International Journal Of Information Technology, pp. 1–11, July 2022.

[9] F.T. Pinki, N. Khatun and S.M.M. Islam, “Content based paddy leaf disease recognition and remedy prediction using support vector machine,” 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–5, December 2017.

[10] Junde Chen, Defu Zhang, Yarser A Nanehkaran and Dele Li, “Detection of rice plant diseases based on deep transfer learning”, Journal of the Science of Food and Agriculture, Volume 100, Issue 7, pp. 3246–3256, March 2020.

[11] Senthil Kumar Swami Durai and Mary Divya Shamili, “Smart farming using Machine Learning and Deep Learning techniques”, Decision Analytics Journal, Volume 3, 100041, ISSN 2772–6622, pp. 1–30, March 2022.

[12] R. Alfred, J.H. Obit, C.P.Y. Chin, H. Haviluddin and Y. Lim, “Towards Paddy Rice Smart Farming: A Review on Big Data, Machine Learning, and Rice Production Tasks”, in *IEEE Access*, Volume 9, pp. 50358–50380, March 2021.

[13] S. Mishra, D. Mishra and G.H. Santra, “Applications of Machine Learning Techniques in Agricultural Crop Production: A Review Paper”, Indian Journal of Science and Technology, Volume 9, Issue 38, pp. 1–14, October 2016.

[14] T. Daniya and Dr.S. Vigneshwari, “A Review on Machine Learning Techniques for Rice Plant Disease Detection in Agricultural Research”, International Journal of Advanced Science and Technology, Volume 28, Issue 13, pp. 49–62, October 2019.

[15] H. James Deva Koresh, “Analysis of Soil Nutrients based on Potential Productivity Tests with Balanced Minerals for Maize-Chickpea Crop”, Journal of Electronics and Informatics, Volume 3, ISSN 2582–3825, Issue 01, pp. 23–35, March 2021.

[16] J. Sustain, DR. D. Sivaganesan, “Performance Estimation of Sustainable Smart Farming with Blockchain Technology”, Wireless Systems, Volume 03, ISSN 2582–3167, Issue 02, pp. 97–106, June 2021.

19IT701 Project Work

0 0 16 8

At the end of the project work the students will be able to

1. Identify a contemporary engineering application to serve the society at large
2. Use engineering concepts and computational tools to get the desired solution
3. Justify the assembled/fabricated/developed products intended.
4. Organize documents and present the project report articulating the applications of the concepts and ideas coherently
5. Demonstrate ethical and professional attributes during the project implementation.
6. Execute the project in a collaborative environment.

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	2				3	2						3	3
CO2	3	3			3								3	3
CO3	3	3	3	2							2		3	3
CO4										3		2	3	3
CO5								3					3	3
CO6									3				3	3