Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning.

B. Tech – CSE, Semester - VIII

Prepared at

ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

Bhaskaracharya National Institute for Space Applications & Geo-informatics
Ministry of Electronics and Information Technology, Govt. of India.

Gandhinagar

Prepared By

Viranshu Paruparla (20162121014)
ID No. 24RN4

Tirth Patel (20162121029)
ID No. 24RN4

Guided By:
Prof. Tejas Kadia
Prof. Sulabh Bhatt
Ganpat University, Kherva.

External Guide:
Dr. Yagnesh Vyas
Project Director,
BISAG-N, Gandhinagar

SUBMITTED TO

Institute of Computer Technology

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology
NAAC A Grade

Ganpat University - Kherva

Bhaskaracharya National Institute for Space Applications and Geo-informatics

MeitY, Government of India

Phone: 079 - 23213081 Fax: 079 - 23213091

E-mail: info@bisag.gujarat.gov.in, website: https://bisag-n.gov.in/

ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

CERTIFICATE

This is to certify that the project report compiled by ***Mr. Viranshu Paruparla and Mr. Tirth Patel*** *students of 8th Semester* ***B.Tech-CSE from Intitute of Computer Technology, Ganpat University, Kherva*** *have completed their final Semester internship project satisfactorily. To the best of our knowledge this is an original and bonafide work done by them. They have worked on application for "****Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning.****", starting from January 01st, 2024 to April 30th, 2024.*

During their tenure at this Institute, they were found to be sincere and meticulous in their work. We appreciate their enthusiasm & dedication towards the work assigned to them.

We wish them every success.

Dr. Yagnesh Vyas
Project Director,
BISAG- N, Gandhinagar

Punit Lalwani
CISO,
BISAG- N, Gandhinagar

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥

Institute of Computer Technology

National Assessment and Accreditation Council
NAAC
A
Grade
Accredited University

CERTIFICATE

This is to certify that the ***Industry*** *Project work entitled* ***"Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning."*** *by Viranshu Paruparla (Enrolment No.20162121014) and Tirth Patel (Enrolment No.20162121029) of Ganpat University, towards the partial fulfilment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering, carried out by them in the CSE (BDA) Department at BISAG-N, Gandhinagar. The results contained in this Project have not been submitted in part or full to any other University / Institute for award of any other Degree.*

Name & Signature of Internal Guide

Prof. Tejas Kadia **Prof. Sulabh Bhatt**

Name & Signature of Head

Prof. Dharmesh Darji

Place: Institute of Computer Technology, GUNI

Date:

About BISAG- N

ABOUT THE INSTITUTE

Modern day planning for inclusive development and growth calls for transparent, efficient, effective, responsive and low cost decision making systems involving multi-disciplinary information such that it not only encourages people's participation, ensuring equitable development but also takes into account the sustainability of natural resources. The applications of space technology and Geo–informatics have contributed significantly towards the socio-economic development. Taking cognizance of the need of geo-spatial information for developmental planning and management of resources, the department of Ministry of Electronics and Information Technology, Government of India, established "Bhaskaracharya National Institute for Space Applications and Geo-informatics" (BISAG- N). BISAG- N is an ISO 9001:2008, ISO 27001:2005 and CMMI: 5 certified institute. BISAG- N which was initially set up to carryout space technology applications, has evolved into a centre of excellence, where research and innovations are combined with the requirements of users and thus acts as a value added service provider, a technology developer and as a facilitator for providing direct benefits of space technologies to the grass root level functions/functionaries.

BISAG- N's Enduring Growth

Since its foundation, the Institute has experienced extensive growth in the sphere of Space technology and Geo-informatics. The objective with which BISAG- N was established is manifested in the extent of services it renders to almost all departments of the State. Year after year the institute has been endeavouring to increase its outreach to disseminate the use of geo-informatics up to grassroots level. In this span of nine years, BISAG- N has assumed multi-dimensional roles and achieved several milestones to become an integral part of the development process of the Gujarat State.

BISAG- N Journey

2003-04

Gujarat SATCOM Network

2007-08

Centre for Geo-informatics Applications

2010-11

Academy of Geo-informatics for Sustainable Developmen t

2012-13

A full-fledged Campus

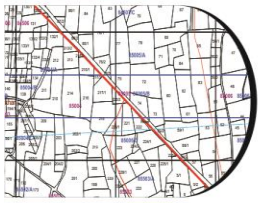
Activities

Satellite Communication..

for promotion and facilitation of the use of broadcast and teleconferencing networks for distant interactive training, education and extension.

Remote Sensing..

for Inventory, Mapping, Developmental planning and Monitoring of natural & man-made resources.

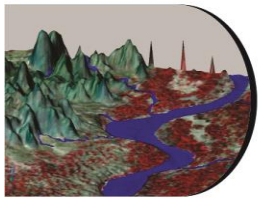


Geographic Information System..

for conceptualization, creation and organization of multi purpose common digital database for sectoral/integrated decision support systems.

Global Navigation Satellite System..

for Location based Services, Geo-referencing, Engineering Applications and Research.



Photogrammetry..

for Creation of Digital Elevation Model, Terrain Characteristic, Resource planning.

Cartography..

for thematic mapping, value added maps.

Software Development..

for wider usage of Geo-spatial applications, Decision Support Systems (desktop as well as web based), ERP solutions.

Education, Research and Training..

for providing Education, Research, Training & Technology Transfer to large number of students, end users & collaborators.

Applications of Geospatial Technology for Good Governance: Institutionalization

Through the geospatial technology, the actual situation on the ground can be accessed. The real life data collected through the technology forms the strong foundation for development of effective social welfare programs benefiting directly the grass root level people. The geospatial data collected by the space borne sensors along with powerful software support through Geographic Information System (GIS), the vital spatio-temporal maps, tables, and various statistics are being generated which feed into Decision Support System (DSS).

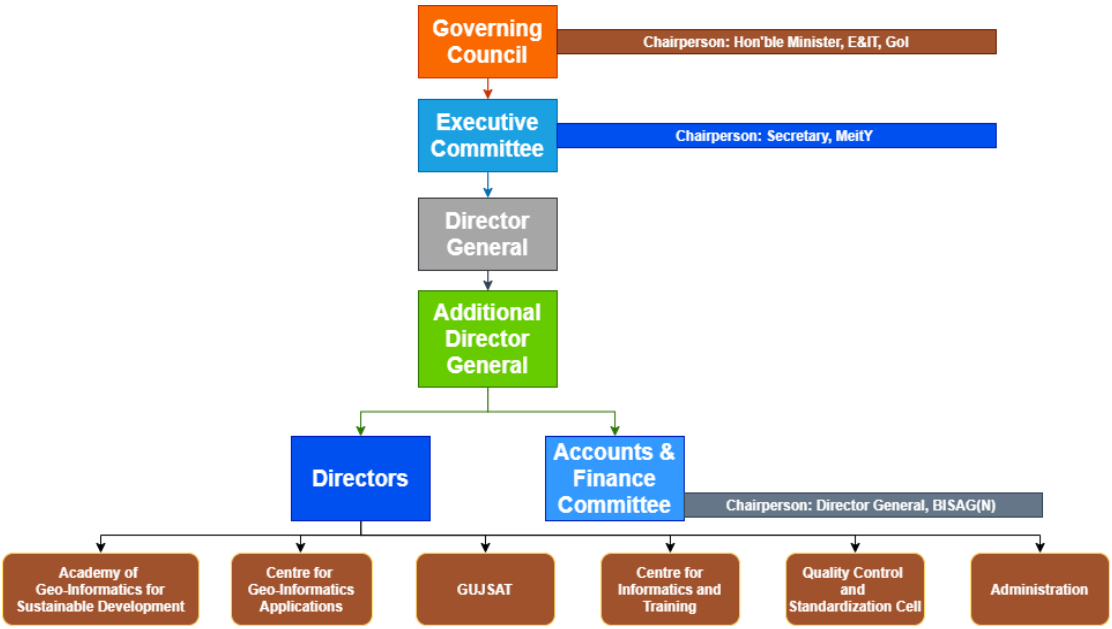
A multi-threaded approach is followed in the process of institutionalization of development of such applications. The 5 common threads which run through all the processes are: *Acceptability, Adaptability, Affordability, Availability and Assimilability*.
These are the "Watch Words" which any application developer has to meet. The "acceptability" addresses the issue that the application developed has met the wide acceptability among the users departments and the ultimate end beneficiary by way of providing all necessary data and statistics required. The "affordability" addresses the issue of the application product being cost effective. The "availability" aspect looks into aspect of easily accessible across any platform, anywhere and anytime. The applications should have inbuilt capability of easy adaptability to the changing spatio- and temporal resolutions of data, new aspects of requirements arising from time to time from users. The assimilability aspect ensures that the data from various sources / resolutions and technologies can be seamlessly integrated.

ACCEPTABILITY	▪ Problem definition by users • Proof of Concept development without financial liability on users ▪ Execution through collaboration under user's ownership
ADOPTABILITY	▪ Applications as per present systems & database ▪ Maximum Automation ▪ Minimum capacity building requirement at the user end
AFFORDABILITY :	▪ Multipurpose geo-spatial database, common, compatible, standardized (100s of layers) ▪ In house developed/open source software ▪ Full Utilization of available assets
AVAILABILITY:	▪ Departmental /Integrated DSS ▪ Desired Product delivery anytime, anywhere in the country
ASSIMILABILITY	▪ Integration of Various technologies like RS, GIS, GPS, Web MIS, Mobile etc.

Organizational Setup

The Institute is responsible for providing information and technical support to different Departments and Organizations. The Governing Body and the Empowered Executive Committee govern the functioning of BISAG- N. The Institute is registered under the Societies Registration Act 1860. Considering the scope and extent of activities of BISAG- N, its organizational structure has been charted out with defined functions.

Organizational Setup of BISAG- N



Governing Body

For smoother, easier and faster institutionalization of Remote Sensing and GIS technology, decision makers of the state were brought together to form the Governing Body. It is the supreme executive authority of the Institute. The Governing Body comprises of ex-officio members from various Government departments and Institutes.

- Hon'ble Minister of Electronics and Information Technology..... Chairperson (Ex-Officio)
- Hon'ble Minister of State Electronics and Information Technology.....……Deputy Chairperson (Ex-Officio)
- Secretary of Government of India: Ministry of Electronics and Information Technology..... .....Executive Vice Chairperson (Ex-Officio)
- Chief Executive Officer, Niti Aayog..... ……Member (Ex-Officio)
- Chairman, Indian Space Research Organization ..... ……Member (Ex-Officio)
- Secretary to Government of India: Department of Science and Technology .. ……Member (Ex-Officio)
- Additional Secretary to Government of India: Ministry of Electronics and Technology ……Member (Ex-Officio)
- Chief Secretary to Government of Gujarat..... ……Member (Ex-Officio)
- President & Chief Executive Officer, National e-Governance Division, Ministry of Electronics
- and Information Technology..... …… Member (Ex-Officio)
- Financial Advisor to Government of India: Ministry of Electronics and Information Technology ..... …Member (Ex-Officio)
- Distinguished Professionals from the GIS field-Three (3) (To be nominated by the Chairperson)

♦ Director-General, Bhaskaracharya National Institute for Space Application and Geo-Informatics {BISAG(N)} ..... Member Secretary (Ex-Officio)

Centre for Geo-informatics Applications

Introduction

The objective of this technology group is to provide decision support to the sectoral stake holders through scientifically organized, comprehensive, multi-purpose, compatible and large scale (village level) geo-spatial databases and supporting analytical tools. These activities of this unit are executed by a well-trained team of multi-disciplinary scientists. The government has provided a modern infrastructure along with the state-of-the-art hardware and software. To study the land transformation and development over the years, a satellite digital data library of multiple sensors of last twenty years has been established and conventional data sets of departments have been co-registered with satellite data. The geo-spatial databases have been created using conventional maps, high resolution satellite 2D and 3D imagery and official datasets (attributes).The geo-spatial databases include terrain characteristics, natural and administrative systems, agriculture, water resources, city survey maps, village maps with survey numbers, water harvesting structures, water supply, irrigation, power, communications, ports, land utilization pattern, infrastructure, urbanization, environment data, forests, sanctuaries, mining areas, industries. They also include social infrastructure like the locations of schools, health centres, institutions, aganwadies, local government infrastructure etc. The geospatial database of nagar-palikas includes properties and amenities captured on city and town planning maps with 1000 GIS layers. Similar work for villages has been initiated as a pilot project.

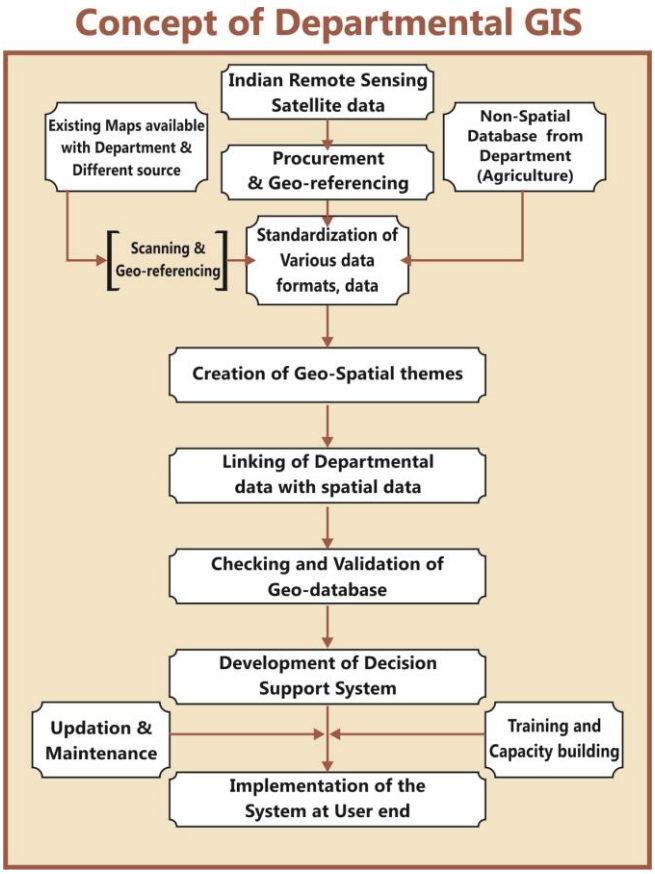
The applications of space technology and geo-informatics have been operational in almost all the development sectors of the state. Remote sensing and GIS applications have provided impetus to planning and developmental activities at grass root level as well as monitoring and management in various disciplines.

The GIS based Applications Development

The GIS software is a powerful tool to handle, manipulate and integrate both the spatial and non-spatial data. The GIS system operates on the powerful backend data base and Sequential Query Language (SQL) to inquiry the data bases. It has the capability to handle large volume of data and process to yield values of parameters which can be input to very important government activity as Decision Support System (DSS). Its mapping capabilities help the users and specialists in generating single and multi-theme wise maps.

The GIS based applications development has been institutionalized in BISAG- N. This process can be listed as (Refer Figure for Details)

- Making the users aware of the GIS capabilities through introductory training programme and by exposing to already developed projects as success stories.
- Helping the users in defining the GIS based projects.
- Digitizing the data available with the users and encouraging them to collect any additional data as may be required.
- Generating the appropriate data bases with the full involvement of the users following the data bases standards



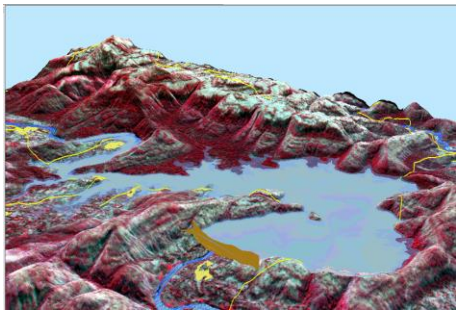
Remote Sensing and GIS Sectoral Applications:

Geo-informatics based Irrigation Management and Monitoring System

- The Geo-spatial information system for Irrigation water Management and Monitoring system for command areas in Sardar Sarovar Narmada Nigam Limited (SSNL) has been developed. Satellite image-based Irrigation monitoring system has been developed in GIS. From the multi-spectral Satellite images of every month, the irrigated areas were extracted.
- The irrigated area were overlaid on the geo-referenced cadastral maps and the statistics of area irrigated has been estimated.

- The user friendly Customized Decision Support System (DSS) has been developed.

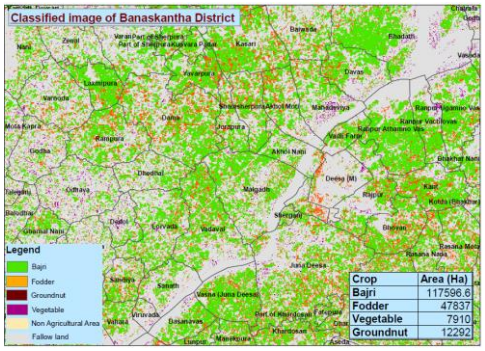
Preparation of DPR of Par–Tapi-Narmada Link using Geo-informatics for National Water development Agency (NWDA)



- The main objective of Par–Tapi-Narmada Link project is to divert surplus water available in west flowing rivers of south Gujarat and Maharashtra for utilization in the drought prone Saurashtra and Kachcha. On the request from NDWA, preparation of various maps for proposed DPR work was undertaken by the BISAG- N. Land use and submergence maps of proposed dams along with its statistics have been prepared by the BISAG- N. The detailed work consisted of generation of Digital Elevation Model (DEM), contour generation, Land use mapping, forest area generation of submergence extent at different levels etc.

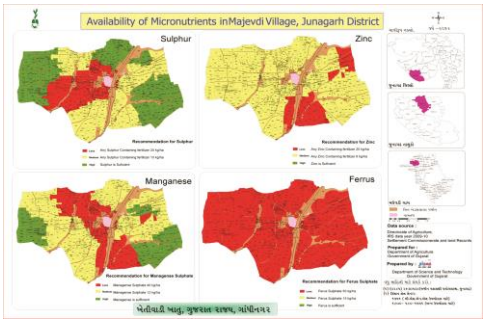
Agriculture

District and Village-level Crop Inventory



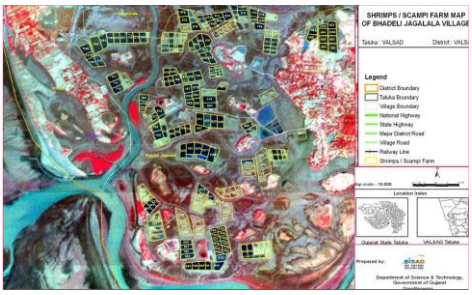
- Remote Sensing (RS) based Village-level Crop Acreage Estimation at was taken up in two villages of Anand and Mehsana districts of Gujarat state. The major objective of this study was to attempt village-level crop inventory during two crop seasons of Kharif (monsoon season) and Rabi (winter season) using single-date Indian Remote Sensing (IRS) LISS-III and LISS-IV digital data of maximum vegetative growth stage of major crops during each season.
- District-level crop acreage estimation during three cropping seasons namely Kharif, Rabi and Zaid (summer) seasons was also carried out in all the 26-districts of Gujarat State. Summer crop acreage estimation Gujarat State was carried out during 2012.

Spatial Variability Mapping of Soil Micro-Nutrients



- The spatial variability of soil micro-nutrients like Fe, Mn, Zn and Cu in various villages of different districts, Gujarat state was mapped using geo-informatics technology. The major objectives of this study were i) to quantify the variability of Mn, Fe, Cu and Zn concentration in soil; ii) to map the pattern of micro-nutrient variability in cadastral maps, iii) suggest proper application of micro-nutrients based on status of deficiency for proper crop management and iv) preparation of village-level atlases showing spatial variability of micro-nutrients.

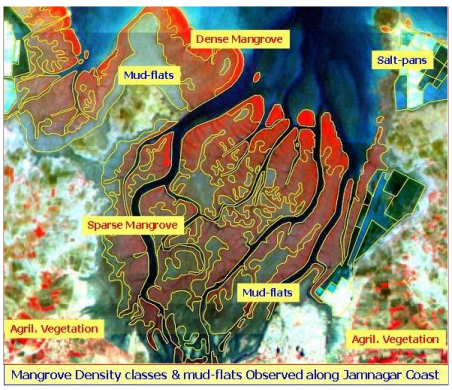
Geo-spatial Information System for Coastal Districts of Gujarat



- The project on development of Village-level Geo-spatial Information System for Shrimp Farms in Coastal Districts of Gujarat, was taken with major objective of development of Village-level Geo-spatial Information System for Shrimp/Scampi areas using Remote Sensing (RS) and GIS. This project was sponsored by the Marine Products Export Development Authority (MPEDA), Ministry of Commerce & Industry, Government of India for scientific management of Scampi farms in the coastal districts which can help fishermen to better their livelihood and increase the economic condition on sustainable basis. The customized query shell was developed using the open source software for sharing the information amongst the officers from MPEDA and potential users. This has helped the farmers to plan their processing and marketing operations so as to achieve better remunerations.

Environment and Forest

Mapping and Monitoring of Mangroves in the Coastal Districts of Gujarat State



Mangrove Density classes & mud-flats Observed along Jamnagar Coast

- Gujarat Ecology Commission, with technical inputs from the Bhaskaracharya National Institute for Space Applications and Geo-informatics - N (BISAG- N) made an attempt to publish Mangrove Atlas of the Gujarat state. Mangrove atlas for 13-coastal districts with 35-coastal talukas in Gujarat, have been prepared using Indian Remote sensing satellite images. The comparison of mangrove area estimates carried out by BISAG- N and Forest Survey of India (FSI) indicates a net increase in the area under mangrove cover. The present assessment by BISAG- N, has recorded 996.3 sq. km under mangrove cover, showing a steep rise to the tune of 88.03 sq. km. In addition to the existing Mangrove cover, the present assessment also gives the availability of potential area of 1153 sq. km, where mangrove regeneration program can be taken up.

Academy of Geo-informatics for Sustainable Development

Introduction

- Considering the requirement of high end research and development in the areas having relevance of geo-informatics technology for sustainable development, a separate infrastructure has been established. In collaboration with different institutes in the state as well as in the country, R&D activities are being carried out in the areas of climate change, environment, disaster management, natural resources management, infrastructure development, resources planning, coastal hazard and coastal zone management studies, etc. under the guidance of eminent scientists.

- Various innovative methodologies/models developed in this academy through the research process have helped in development of various applications. There are plans to enhance R&D activities manifold during coming years.
- This unit also provides training to more than 600 students every year in the field of Geo-informatics to the students from various backgrounds like water resources, urban planning, computer Engineering, IT, Agriculture in the areas of Remote sensing, GIS and their applications.
- This Academy has been established as a separate infrastructure for advanced research and development through following schools:
  - School of Geo-informatics
  - School of Climate & Environment
  - School of Integrated Coastal Zone Management

- School of Sustainable Development Studies
- School of Natural Resources and Bio-diversity
- School of Information Management of Disasters
- School of Communication and Society

During XIIth Five year Plan advance applied research through above schools shall be the main thrust area. Already M. Tech and Ph.D. students of other Universities/ Institutes are doing research in this academy in applied sciences under various collaborative programmes.

M. Tech. Students' Research Programme

The academy started M. Tech. students' research programme in a systematic way. It admitted 11 students from various colleges and universities in Gujarat, Rajasthan and Madhya Pradesh for period of 10 months from August 2011 to May 2012. All the students were paid stipend of Rs. 6000 per month during the tenure. The research covered the following areas:

- Cloud computing techniques
- Mobile communication
- Design of embedded systems
- Aquifer modelling
- Agricultural and Soils Remote Sensing
- Digital Image processing Techniques (Data Fusion and Image Classification).

The research resulted in various dissertations and publications in national and international journals.

• Now nine students, one from IIT, Kharagpur, three from GTU, one from M. S University, Vadodara and four from GU, are undergoing their Ph. D programme. Out of nine, two thesis have been submitted. Two students are from abroad. One each from Vietnam and Yemen. Since then (after approval of research programme from the Governing Body), 200+ papers have been published by the Academy.

CANDIDATE'S DECLARATION

We declare that 8th semester internship project report entitled "**Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning.**" is our own work conducted under the supervision of the external guide **Dr. Yagnesh Vyas** from BISAG-N (Bhaskaracharya National Institute for Space Applications & Geo-informatics). We further declare that to the best of our knowledge the report for this project does not contain any part of the work which has been submitted previously for such project either in this or any other institutions without proper citation.

Candidate 1's Signature

Viranshu Paruparla

Student ID: 24RN4

Candidate 2's Signature

Tirth Patel

Student ID: 24RN4

Submitted To:

Institute of Computer Technology

Ganpat University.

ACKNOWLEDGMENT

We are grateful to **Shri. T.P. Singh,** Director General (BISAG-N) for giving us this opportunity to work the guidance of renowned people of the field of Data Science and AI/ML also providing us with the required resources in the company.

We would like to express our endless thanks to our external guide **Dr. Yagnesh Vyas** and to Training Cell **Mr. Punit Lalwani & Mr. Sidhdharth Patel** at Bhaskaracharya National Institute of Space Application and Geo-informatics for their sincere and dedicated guidance throughout the project development.

Also, our hearty gratitude to our Head of Department, **Prof. Dharmesh Darji** and our internal guide **Prof. Tejas Kadia and Prof. Sulabh Bhatt** for giving us encouragement and technical support on the project.

Viranshu Paruparla.
Student ID: 24RN4

Tirth Patel.
Student ID: 24RN4

ABSTRACT

"Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning." focuses on utilizing artificial intelligence (AI) and machine learning (ML) techniques to assist farmers in making informed decisions for crop cultivation. The system primarily consists of two main modules: Crop Recommendation and Disease Detection using Deep Learning. The Crop Recommendation module analyses environmental factors and location data input by the user to recommend the most suitable crops for cultivation, considering both favourable and unfavourable conditions. This assists farmers in optimizing yield and reducing risks associated with unsuitable environmental factors. Disease Detection module employs Deep Learning models to predict crop diseases by analysing images of crop leaves. By identifying disease symptoms at an early stage through image analysis, farmers can take timely action to mitigate the spread and minimize crop damage. Together, these modules provide comprehensive support to farmers in crop planning, environmental management, and disease prevention, thereby promoting sustainable and efficient agricultural practices.

TABLE OF CONTENT

LIST OF IMAGES

LIST OF TABLES

1. Introduction

In the dynamic landscape of modern agriculture, the confluence of technology and farming practices has emerged as a transformative force, reshaping traditional methodologies and paving the way for innovative solutions to address the multifaceted challenges faced by farmers globally. The project titled "Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning." stands at the forefront of this technological revolution, harnessing the power of artificial intelligence (AI) and machine learning (ML) to offer comprehensive support to farmers.

This groundbreaking initiative is anchored by two core modules: Crop Recommendation and Disease Detection using Deep Learning, each designed to provide farmers with actionable insights and decision-making tools to optimize agricultural productivity, enhance crop resilience, and foster sustainable farming practices. The Crop Recommendation module serves as an intelligent decision-support system, leveraging a sophisticated algorithm to analyse a diverse array of environmental factors, soil conditions, and location-specific data input by the user. By synthesizing this information, the module generates tailored recommendations on the most suitable crops for cultivation, taking into account both the favourable and unfavourable conditions prevalent in the specified geographic region.

This proactive approach enables farmers to make informed decisions that align with the natural characteristics of their land, thereby maximizing yield potential, minimizing risks associated with environmental variability, and promoting resource-efficient farming practices. Furthermore, the module facilitates adaptive farming strategies, allowing farmers to respond dynamically to changing environmental conditions, such as fluctuations in temperature, precipitation, and soil quality, ensuring optimal crop performance and sustainability.

In parallel, the Disease Detection module harnesses the capabilities of deep learning algorithms and advanced image processing techniques to detect and identify various crop diseases through the analysis of images captured from crop leaves. Early detection of diseases is paramount in agriculture, as it enables timely intervention and implementation of effective control measures, thereby reducing crop losses, preserving crop health, and safeguarding yield potential. By providing farmers with early warnings and predictive insights, this module empowers them to implement targeted disease management strategies, ranging from precision application of pesticides to crop rotation and integrated pest management practices, fostering resilience against disease outbreaks and enhancing overall crop health.

Moreover, the seamless integration of these modules into a unified platform offers a user-friendly interface with real-time data analysis capabilities, facilitating intuitive interaction between farmers and technology. This integration not only streamlines the decision-making process but also bridges the gap between data analytics and practical farming applications, enabling farmers to leverage the benefits of AI and ML without requiring specialized technical expertise. By democratizing access to advanced agricultural technologies, the system empowers farmers of all scales to adapt to the complexities of modern agriculture, optimize resource utilization, and implement sustainable farming practices effectively.

The innovative design and functionality of the "Farmer Support System using AI/ML" project signify a significant milestone in the evolution of agricultural technology, demonstrating the transformative potential of AI and ML in reimagining farming practices and supporting farmers in navigating the challenges of contemporary agriculture. By fostering innovation, embracing technological advancements, and promoting collaborative learning, this project endeavours to create a synergistic ecosystem that harmonizes technological innovation with agricultural expertise, paving the way for a resilient, productive, and sustainable agricultural future.

1.1 Scope

"Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning." project involves developing an integrated platform utilizing AI and ML technologies to assist farmers. The project focuses on collecting and analysing environmental data to build a database, implementing machine learning algorithms for crop recommendations and environmental adaptability predictions, and utilizing deep learning for early disease detection from crop leaf images. The platform will offer a user-friendly interface for real-time data analysis and personalized recommendations, aiming to empower farmers with actionable insights to optimize crop cultivation, enhance agricultural productivity, and promote sustainable farming practices.

1.2 Objective

The primary objective of the "Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction using machine learning and disease detection using deep learning." project is to harness the capabilities of artificial intelligence (AI) and machine learning (ML) to provide comprehensive support to farmers. The project aims to develop an intelligent decision-support

system that analyses environmental factors, soil conditions, and location-specific data to recommend the most suitable crops for cultivation. Additionally, the project seeks to implement a predictive analytics module that forecasts the performance of selected crops in specific environmental conditions and a robust disease detection module that utilizes deep learning algorithms to identify crop diseases through image analysis. By achieving this, the project aspires to empower farmers with actionable insights, enabling them to optimize crop yields, minimize risks, and foster sustainable farming practices effectively.

1.3 Literature Review

1.3.1 Literature Review for Crop Recommendation and Favourability

Crop recommendation systems are revolutionizing agriculture by empowering farmers with data-driven decision-making for optimal crop selection. These systems leverage machine learning algorithms to analyse a variety of factors, with a particular emphasis on environmental conditions. Research by Anitha et al. (2020) and Singh et al. (2021) highlight the importance of environmental factors like soil properties, weather patterns (temperature, rainfall), and historical data in informing crop recommendations [1, 3]. Li et al. (2023) explores the application of Graph Convolutional Neural Networks (GCNs) for crop recommendation, demonstrating their effectiveness in modelling relationships between these environmental factors [2]. Recent advancements in explainable AI (XAI) are also shaping the field, with Shams et al. (2024) introducing XAI-CROP, a system emphasizing user-friendliness and clear explanations for farmers [4]. While environmental factors are crucial, Patil et al. (2022) advocate for a more holistic approach, considering aspects like water availability and pest/disease risks for sustainable agriculture [5]. Feature selection of environmental data points is critical for model accuracy, as highlighted by Alemán et al. (2020) and Singh et al. (2019) in their reviews of machine learning models for crop yield prediction [6, 7].

1.3.2 Literature Review for Crop Disease Detection

A survey by Singh et al. (2022) provides a comprehensive overview of deep learning applications in plant disease detection [8]. Specifically for wheat, studies by Li et al. (2020) demonstrate the effectiveness of CNNs in identifying fungal diseases [9], while Jha et al. (2020) explore leveraging image segmentation for early detection [10]. Ma et al. (2018) investigates the use of deep learning for wheat yellow rust disease detection using hyperspectral imaging, a technique that captures information beyond the visible spectrum,

offering potential for early disease identification [11]. Similar success with rice disease detection is documented by Lee et al. (2017) [12] and Wang et al. (2020), who propose a deep learning model for classifying both diseases and insect pests [13]. Nadal et al. (2018) compare deep learning with another machine learning technique, support vector machines (SVMs), for rice disease diagnosis, highlighting potential advantages of deep learning for specific disease classes [14]. Coffee is another crop benefiting from deep learning. Souza et al. (2019) compared different architectures, finding CNNs most effective for coffee leaf disease classification [15]. Girão et al. (2ia8) built a deep learning model achieving high accuracy in real-time coffee leaf disease detection [16]. Ustaoglu et al. (2020) propose a deep convolutional neural network (DCNN) architecture specifically designed for coffee leaf disease detection and classification [17]. For tea, Palanisamy et al. (2018) proposed a CNN-based framework for effective tea leaf disease classification [18].

1.4 Work Flow

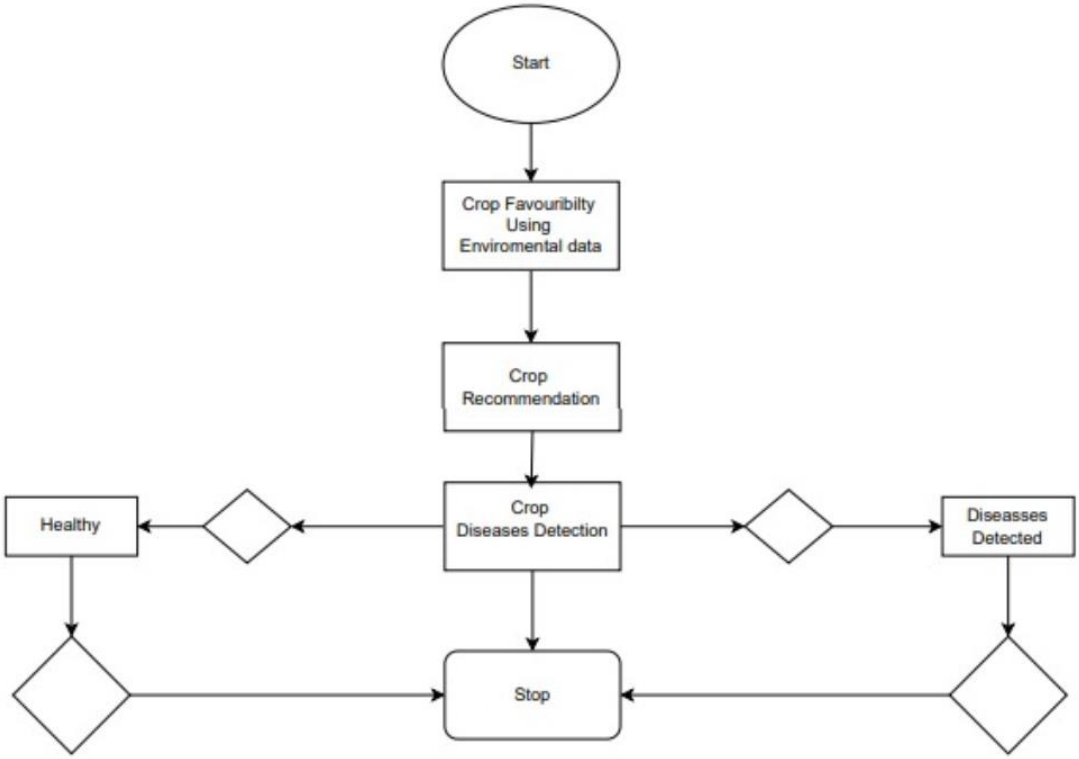


Fig 1 Flow Chart

2. Data Description

2.1 Dataset

2.1.1 Dataset For Crop Recommendation and Favourability

We used Farm Futro [19] Dataset of crop recommendation system which consist of 2212 rows and 19 columns.

2.1.2 Dataset For Crop Disease Detection

We used the Crop Disease Improved Version Dataset [20] for diseases in wheat, apple, and tea leaves, providing a comprehensive image collection for training. The MangoLeafBD Dataset [21] was employed for mango leaf diseases, while JMuBEN [22] and JMuBEN2 [23] datasets were used for coffee leaf diseases, offering diverse images to enhance our model's recognition capabilities. Additionally, the Rice_disease_model [24] dataset aided in identifying diseases in rice leaves. By combining these datasets, we developed comprehensive models proficient in identifying diseases across wheat, apple, tea, mango, coffee, and rice crops, advancing agricultural disease detection for sustainable farming practices.

2.2 Data Description

2.2.1 Data Description for Crop Recommendation and Favourability

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2212 entries, 0 to 2211
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   temperature       2200 non-null   float64
 1   humidity          2200 non-null   float64
 2   ph                2200 non-null   float64
 3   rainfall          2200 non-null   float64
 4   ph after harvest  2200 non-null   float64
 5   Season            2200 non-null   object
 6   DIVISIONS         2200 non-null   object
 7   States            2200 non-null   object
 8   label             2200 non-null   object
 9   Unnamed: 9        0 non-null      float64
 10  Unnamed: 10       0 non-null      float64
 11  Unnamed: 11       0 non-null      float64
 12  Unnamed: 12       0 non-null      float64
 13  Unnamed: 13       0 non-null      float64
 14  Unnamed: 14       0 non-null      float64
 15  Unnamed: 15       0 non-null      float64
 16  Unnamed: 16       0 non-null      float64
 17  Unnamed: 17       0 non-null      float64
 18  Unnamed: 18       0 non-null      float64
dtypes: float64(15), object(4)
memory usage: 328.5+ KB
```

Fig 2 Data info

No.	Column name	Column description
1	Temperature	This column likely represents the temperature of the environment or soil, measured in a specific unit such as Celsius or Fahrenheit.
2	Humidity	This column likely represents the humidity level, which is the amount of moisture present in the air or soil, usually measured in percentage.
3	ph	This column likely represents the humidity level, which is the amount of moisture present in the air or soil, usually measured in percentage.

4	rainfall	This column likely represents the amount of rainfall in a specific area or region, typically measured in millimetres or inches.
5	Ph after harvest	This column could represent the pH level of the soil after harvesting crops, which may be different from the initial pH level.
6	Seasons	This column may indicate the season during which the data was collected, such as summer, winter, etc.
8	States	This column likely indicates the states or regions where the data was collected.
9	Label	This column might contain labels or categories assigned to the data, such as crop types, crop health status, etc.
10	Unnamed 9 to 18	These columns appear to be unnamed and might be additional data columns or possibly formatting artifacts from the dataset. They seem to be empty or contain irrelevant information.

Table 1 Data Description for Crop Recommendation

2.2.2 Data Description for Crop Disease Detection

No.	Crop Name	Disease Name (Class)	Total Photos
1	Wheat	Wheat__brown_rust	3354

		Wheat__healthy	
		Wheat__septoria	
		Wheat__yellow_rust	
2	Tea	Tea__algal_leaf	1,926
		Tea__anthracnose	
		Tea__bird_eye_spot	
		Tea__brown_blight	
		Tea__healthy	
		Tea__red_leaf_spot	
3	Apple	Apple__black_rot	3,299
		Apple__healthy	
		Apple__rust	
		Apple__scab	
4	Mango	Anthracnose	4,000
		Bacterial Canker	
		Cutting Weevil	
		Die Back	
		Gall Midge	
		Healthy	
		Powdery Mildew	

		Sooty Mould	
5	Coffee	Cerscospora	58,549
		Healthy	
		Leaf rust	
		Miner	
		Phoma	
6	Rice	Bacterialblight	7,420
		Blast	
		Brownspot	
		Tungro	
		Rice__healthy	

Table 2 Data Description Crop Disease Detection

2.3 Data Preprocessing

2.3.1 Data Preprocessing for Crop Recommendation and Favourability

In our data preparation process, we meticulously curated the raw CSV data to ensure its suitability for analysis and modelling. Initially, we imported the dataset into a pandas, Data Frame, laying the groundwork for subsequent processing. As we delved into the dataset, we identified columns labelled 'Unnamed' that lacked meaningful information for our analysis. Recognizing their redundancy, we swiftly discarded these columns to streamline our dataset and focus solely on pertinent variables.

Subsequently, our attention turned to handling missing values, a critical step in ensuring the integrity of our data. We prioritized variables essential for agricultural analysis, such as temperature, humidity, pH, and rainfall. Rows

containing missing values in these key parameters were systematically removed, preserving the quality and reliability of our dataset. By prioritizing completeness and accuracy, we were able to retain a clean Data Frame comprising 2200 entries, each brimming with vital details including seasonal information, geographic divisions, states, and crop types.

This meticulously prepared dataset serves as a solid foundation for our forthcoming analysis and modelling endeavours. Its cleanliness and coherence empower us to delve deep into agricultural practices, uncovering meaningful patterns and insights. With this refined dataset at our disposal, we are poised to embark on comprehensive analyses, leveraging advanced techniques to derive actionable insights that can inform agricultural decision-making and contribute to the advancement of sustainable farming practices.

	temperature	humidity	ph	rainfall	ph after harvest
0	20.879744	82.002744	6.502985	202.935536	5.5
1	21.770462	80.319644	7.038096	226.655537	5.6
2	23.004459	82.320763	7.840207	263.964248	5.7
3	26.491096	80.158363	6.980401	242.864034	5.8
4	20.130175	81.604873	7.628473	262.717340	5.9
...	...	...	...	...	...
2207	NaN	NaN	NaN	NaN	NaN
2208	NaN	NaN	NaN	NaN	NaN
2209	NaN	NaN	NaN	NaN	NaN
2210	NaN	NaN	NaN	NaN	NaN
2211	NaN	NaN	NaN	NaN	NaN

	Season	DIVISIONS	States	label	Unnamed: 9	Unnamed: 10
0	Kharif	cereals	UttarPradesh	rice	NaN	NaN
1	Kharif	cereals	Maharashtra	rice	NaN	NaN
2	Kharif	cereals	Punjab	rice	NaN	NaN
3	Kharif	cereals	HimachalPradesh	rice	NaN	NaN
4	Kharif	cereals	WestBengal	rice	NaN	NaN
...	...	...	...	...	...	...
2207	NaN	NaN	NaN	NaN	NaN	NaN
2208	NaN	NaN	NaN	NaN	NaN	NaN
2209	NaN	NaN	NaN	NaN	NaN	NaN
2210	NaN	NaN	NaN	NaN	NaN	NaN
2211	NaN	NaN	NaN	NaN	NaN	NaN

	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...
2207	NaN	NaN	NaN	NaN	NaN
2208	NaN	NaN	NaN	NaN	NaN
2209	NaN	NaN	NaN	NaN	NaN
2210	NaN	NaN	NaN	NaN	NaN
2211	NaN	NaN	NaN	NaN	NaN

	Unnamed: 16	Unnamed: 17	Unnamed: 18

Fig 3 Data before preprocessing

```
Cleaned DataFrame (NaN values and 'Unnamed' columns removed):
      temperature   humidity        ph    rainfall  ph after harvest  \
0       20.879744  82.002744  6.502985  202.935536               5.5
1       21.770462  80.319644  7.038096  226.655537               5.6
2       23.004459  82.320763  7.840207  263.964248               5.7
3       26.491096  80.158363  6.980401  242.864034               5.8
4       20.130175  81.604873  7.628473  262.717340               5.9
...           ...        ...       ...         ...               ...
2195    26.774637  66.413269  6.780064  177.774507               5.6
2196    27.417112  56.636362  6.086922  127.924610               4.9
2197    24.131797  67.225123  6.362608  173.322839               5.0
2198    26.272418  52.127394  6.758793  127.175293               5.1
2199    23.603016  60.396475  6.779833  140.937041               5.2

         Season   DIVISIONS          States   label
0       Kharif      cereals     UttarPradesh    rice
1       Kharif      cereals      Maharashtra    rice
2       Kharif      cereals           Punjab    rice
3       Kharif      cereals  HimachalPradesh    rice
4       Kharif      cereals       WestBengal    rice
...        ...         ...              ...     ...
2195  Perennial   cashcrops        NorthEast  coffee
2196  Perennial   cashcrops        Karnataka  coffee
2197  Perennial   cashcrops        TamilNadu  coffee
2198  Perennial   cashcrops           Kerela  coffee
2199  Perennial   cashcrops        NorthEast  coffee
```

Fig 4 Data After Preprocessing

2.3.2 Data Preprocessing for Crop Disease Detection

In our pursuit of creating a high-quality dataset for machine learning, we undertook a rigorous image cleaning process to ensure the utmost precision and reliability. This meticulous phase involved scrutinizing each image carefully, eliminating any that appeared blurry, unclear, or compromised in quality. By upholding such rigorous standards, we aimed to train our machine learning model exclusively on pristine and informative examples, setting the stage for building a robust and accurate model.

After completing the exhaustive cleaning process, we proceeded to strategically partition the remaining data into three distinct subsets. The first and most substantial subset, constituting 70% of the data, was designated as the training set. This set served as the primary learning ground for our model, exposing it to a vast array of images to identify intricate patterns and relationships within the data. As the model engaged with this set, it gradually honed its ability to

make precise predictions, even when encountering entirely new and unseen images.

The second subset, accounting for 20% of the data, was allocated as the testing set. Importantly, this set remained entirely independent of the training data to ensure unbiased evaluation of the model's performance. By presenting the model with novel and unseen examples from the testing set, we were able to gauge its generalizability and assess how effectively it could perform on data it hadn't previously encountered.

Lastly, we incorporated an optional but crucial subset, constituting 10% of the data, known as the validation set. This set played a pivotal role in fine-tuning the model during its training phase. Its primary function was to act as a safeguard against overfitting, a phenomenon where the model becomes excessively attuned to the training data, compromising its ability to generalize to new data. By monitoring the model's performance on the validation set, we could identify and rectify potential overfitting issues, ensuring the model's reliability and adaptability.

By meticulously segmenting the data into these specialized training, testing, and validation subsets, we achieved several essential objectives. First, the model was trained on a diverse and representative sample of the data, enhancing its learning capability. Second, the independent testing set enabled us to assess the model's performance on new and unseen examples, ensuring its generalizability. Lastly, the validation set acted as a critical safeguard, helping to prevent overfitting and ensuring the model's reliability across various datasets. This comprehensive and multi-faceted approach culminated in the development of an accurate, reliable, and robust machine learning model capable of effectively identifying crop diseases across different datasets and agricultural contexts.

Before Preprocessing	After Preprocessing
	

Table 3 Image before and after preprocessing

Data Augmentation	Rotation Range = 20
	Wide Shift Range = 0.2
	Hight Shift Range = 0.2
	Shear Range = 0.2
	Zoom Range = 0.2
	Image size = 224x224
	Batch Size = 32

Table 4 Data Augmentation

3. Model Architectures

3.1 Machine Learning Models for Crop recommendation and Favourability

In our methodology, the primary goal is to maximize the predictive performance of our classifiers, which are essential tools in making accurate predictions based on our agricultural dataset. To accomplish this, we employ a diverse set of machine learning techniques, each chosen for its unique strengths in extracting meaningful insights from the data.

3.1.1 XGBoost (eXtreme Gradient Boosting)

XGBoost is a standout algorithm known for its efficiency and accuracy in predictive tasks. Its iterative approach of adding decision trees gradually enhances model performance while avoiding overfitting. Notably, XGBoost excels in handling various data types and missing values seamlessly, reducing the need for extensive preprocessing. Its customizable hyperparameters offer practitioners the flexibility to fine-tune the model for optimal performance across different datasets and tasks.

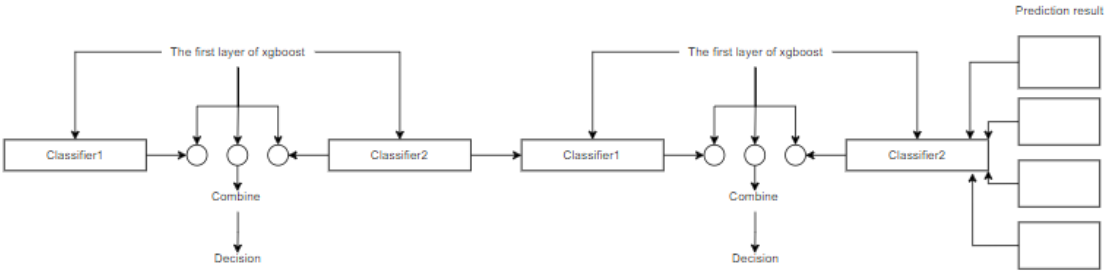


Fig 5 XGBoost Architecture

3.1.2 Support Vector Machines (SVM)

SVMs are versatile tools well-suited for classification and regression tasks. They excel in identifying optimal hyperplanes that separate data points in high-dimensional spaces, maximizing the margin between different classes. SVMs are particularly adept at handling high-dimensional datasets while remaining resilient against overfitting, owing to the concept of support vectors in defining decision boundaries.

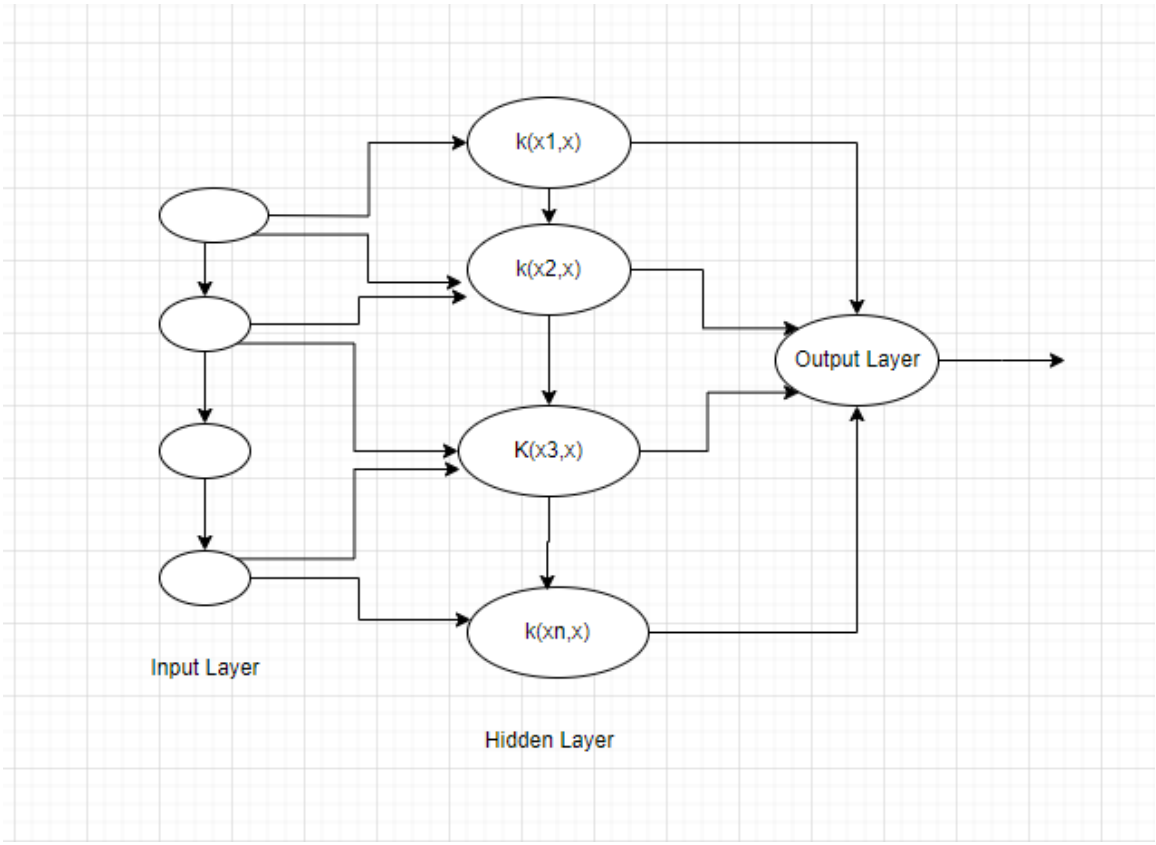


Fig 6 SVM Architecture

3.1.3 Logistic Regression

Logistic Regression serves as a simple yet powerful tool for binary classification tasks. By fitting a sigmoidal curve to the data, it calculates the likelihood of an event occurrence, providing interpretable results. Robust to noise and compatible with diverse feature types, Logistic Regression is a reliable choice for straightforward classification tasks. However, its limitation lies in its reliance on linear relationships, which may not capture nonlinear data patterns effectively.

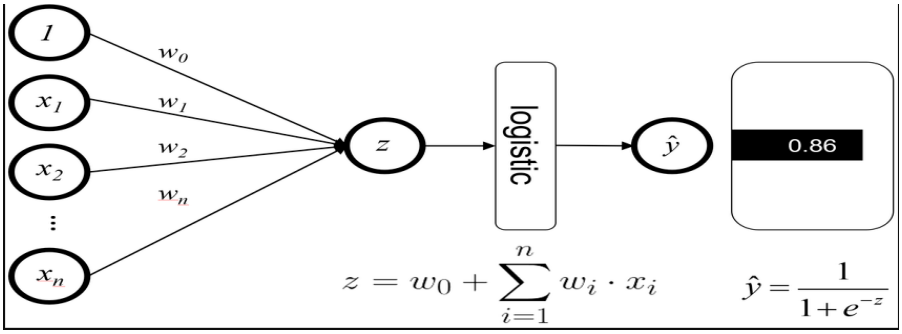


Fig 7 Logistic Regression Architecture

3.1.4 Random Forest

Random Forest is a potent ensemble learning method for both classification and regression tasks. By constructing multiple decision trees and aggregating their outputs, it enhances accuracy and mitigates overfitting. Random Forest excels in managing complex datasets and is resilient against noise without demanding extensive parameter adjustments. However, its interpretability may be limited compared to some other methods.

In summary, our methodology involves carefully selecting and leveraging these machine learning techniques to extract the most salient information from our agricultural dataset. By enhancing the predictive capabilities of our models, we aim to make accurate predictions that can inform decision-making in agricultural contexts, ultimately contributing to improved crop yield and sustainable farming practices.

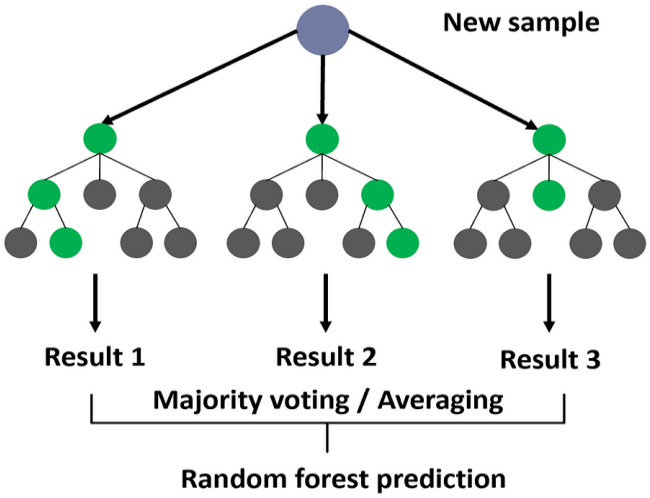


Fig 8 Random Forest Architecture

3.2 Deep Learning Models for Crop Disease Detection

Beyond basic object recognition, certain deep learning models excel in image classification and understanding tasks. This section delves into four popular choices: MobileNet, ResNet-50, VGG16, and AlexNet. While initially designed for object recognition, their versatility through transfer learning enables them to address various image-related challenges, such as video classification, pinpointing specific image regions, and efficiently organizing and searching image databases. MobileNet prioritizes efficiency on devices with limited resources. It achieves this by breaking down traditional convolutions into smaller, more manageable parts, reducing computational costs without compromising accuracy. ResNet-50 addresses a common issue in deep learning - the vanishing gradient problem. It incorporates "skip connections" that allow the network to learn more effectively by adding outputs from previous layers directly to later ones.

VGG16, on the other hand, focuses on achieving high accuracy. It achieves this by stacking multiple small convolutional filters, resulting in a deep architecture capable of capturing complex image features. Lastly, AlexNet, a groundbreaking architecture, paved the way for future advancements. It utilizes stacked convolutional and pooling layers followed by fully-connected layers, serving as a blueprint for efficient CNN designs.

Each architecture offers unique strengths and weaknesses, making them suitable for different applications depending on resource constraints, desired accuracy, and

task complexity. The following section will explore how these specific models perform in the context of Crop Disease Classification, assessing their suitability and effectiveness in identifying and classifying diseases across various crop types.

3.2.1 MobileNet Architecture

MobileNets are a specialized class of convolutional neural network (CNN) architectures designed for mobile and embedded vision applications, emphasizing efficiency and adaptability [25]. They employ depth-wise separable convolutions to significantly reduce computational costs compared to traditional convolutional layers. Additionally, MobileNets incorporate two essential hyperparameters: the width multiplier and the depth multiplier. These parameters enable developers to fine-tune the model's size and accuracy trade-off, making it versatile and adaptable to various resource constraints. The width multiplier adjusts the number of channels in each layer, impacting computational cost, while the depth multiplier regulates the network's layer depth and learning capacity. This flexibility allows MobileNets to be optimized for a wide range of applications, from low-power mobile devices to high-performance embedded systems, without compromising performance, thereby expanding the possibilities for efficient image processing in resource-constrained environments.

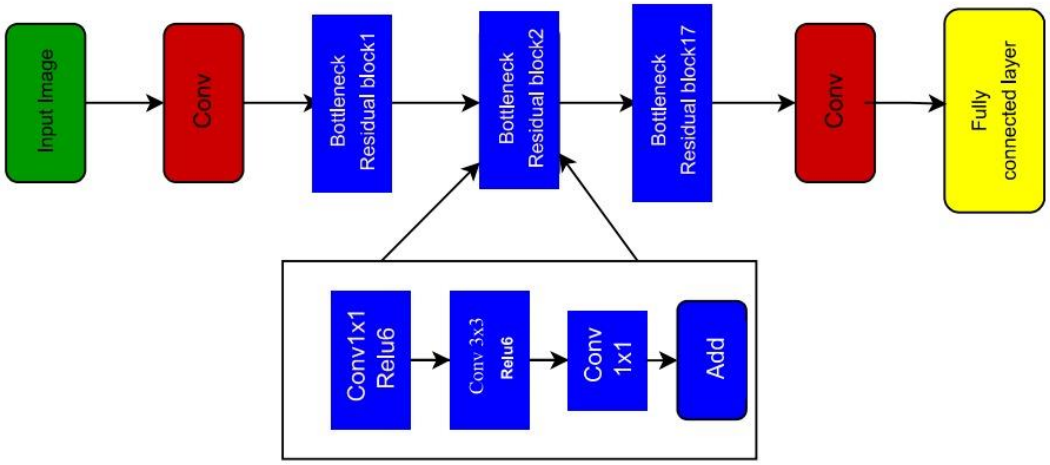


Fig 9 MobileNet Architecture

3.2.2 VGG16 Architecture

The VGG16 architecture, introduced by Simonyan and Zisserman in 2014, features a deep structure with 16 convolutional layers, followed by three fully-connected layers and a SoftMax classification layer. Utilizing 3x3 convolutional filters with a stride of 1 and same padding, each convolutional layer is paired with a Rectified Linear Unit (ReLU) activation function to introduce non-linearity and facilitate feature learning. Interposed between the convolutional layers are max-pooling layers with 2x2 filters and a stride of 2, which down sample the feature maps to capture spatial hierarchies at varying scales. The fully-connected layers consist of three dense layers with 4096, 4096, and 1000 neurons, respectively, serving as classifiers. The final SoftMax layer normalizes the class scores into probability distributions for classification. While VGG16 excels in accuracy for image classification, its computational intensity, stemming from a large parameter count, poses challenges for deployment on resource-limited devices, contrasting with more lightweight architectures. Nonetheless, its architectural simplicity and performance have solidified its position as a benchmark in deep learning, influencing subsequent convolutional neural network designs.

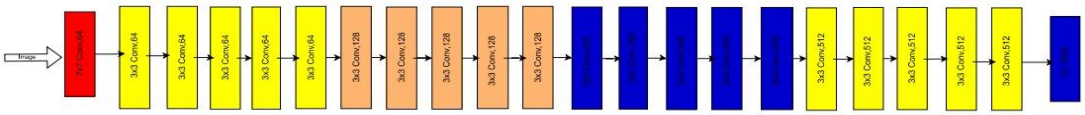


Fig 10 VGG16 Architecture

3.2.3 AlexNet Architecture

AlexNet, introduced by Krizhevsky et al. in 2012[27], stands as a groundbreaking convolutional neural network (CNN) architecture that marked a significant milestone in image classification accuracy, catalyzing the resurgence of deep learning in computer vision. The architecture comprises eight layers in total, beginning with five convolutional layers, each paired with ReLU activation functions to introduce non-linearity and facilitate feature extraction. Following the convolutional layers, AlexNet incorporates three fully-connected layers, serving as classifiers to transform the extracted features into class scores. This innovative design, coupled with the utilization of techniques such as dropout for

regularization and data augmentation, enabled AlexNet to achieve remarkable performance on the ImageNet dataset, setting new benchmarks in image classification tasks and sparking renewed interest and advancements in deep learning for computer vision.

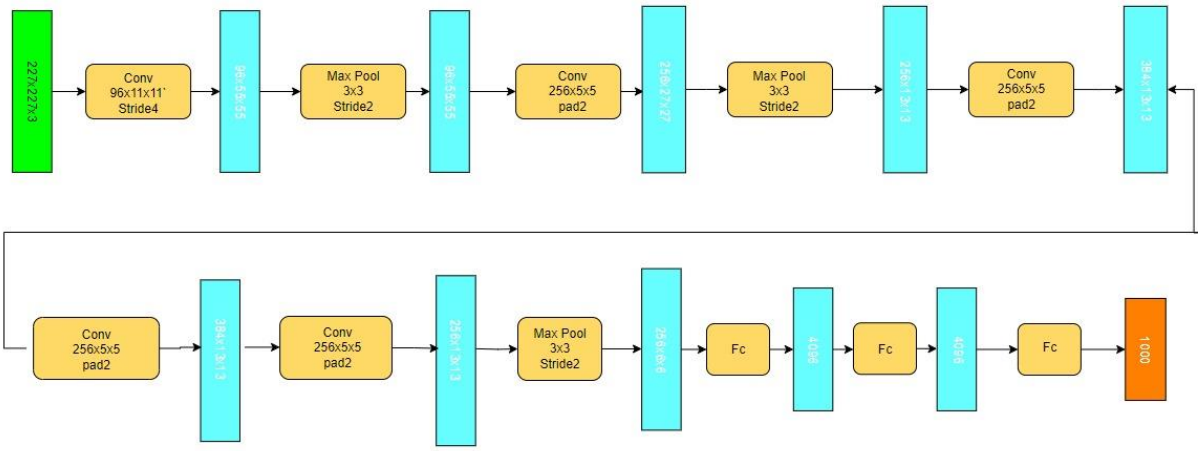


Fig 11 AlexNet Architecture

3.2.4 ResNet-50 Architecture

ResNet-50, introduced by He et al. in 2015 [28], is a deep convolutional neural network (CNN) architecture that revolutionized the field by addressing the challenges of training very deep networks through its innovative use of residual learning. With a total of 50 layers, ResNet-50 incorporates residual connections, or "skip connections," which allow the original input to be added to the output of deeper layers, facilitating the learning of residual representations and mitigating the vanishing gradient problem. This design not only enables the effective training of deeper networks but also enhances the model's capacity to learn complex features and patterns in the data. Leveraging batch normalization and ReLU activation functions further stabilizes and accelerates the training process, contributing to improved performance. As a result, ResNet-50 has become a cornerstone in deep learning for computer vision, demonstrating exceptional accuracy in image classification and serving as a foundation for various computer vision applications due to its ability to capture intricate feature representations.

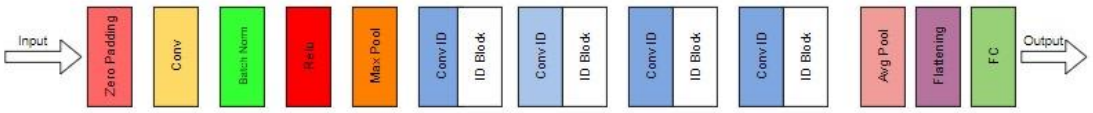
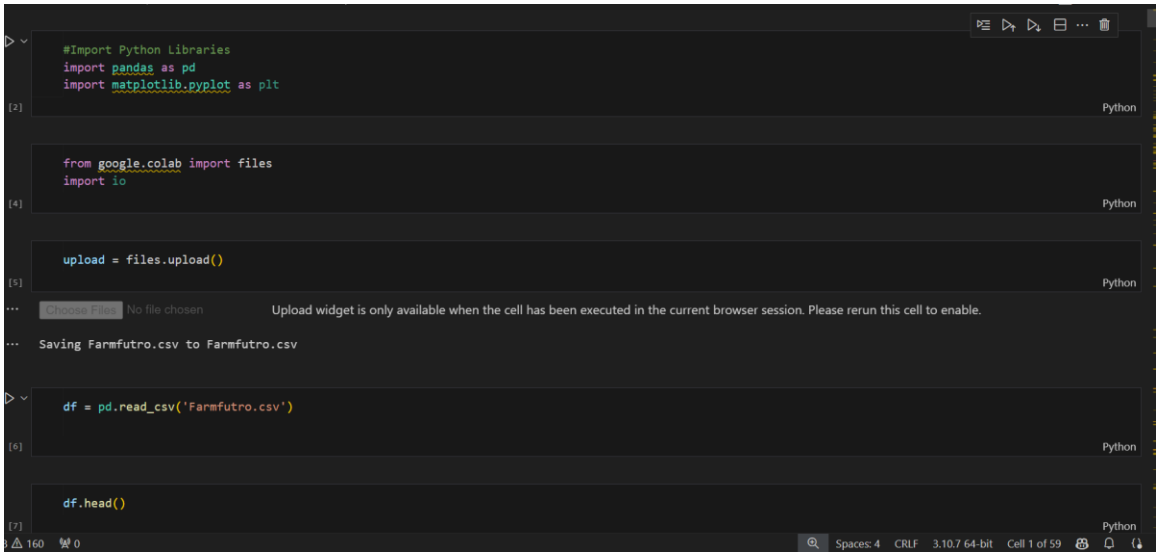


Fig 12 ResNet-50 Architecture

4. Implementation

4.1 Implementation of Crop recommendation and Favourability



	temperature	humidity	ph	rainfall	ph after harvest	Season	DIVISIONS	States	label	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnam
0	20.879744	82.002744	6.502985	202.935536	5.5	Kharif	cereals	UttarPradesh	rice	NaN	NaN	NaN	NaN	NaN	N
1	21.770462	80.319644	7.038096	226.655537	5.6	Kharif	cereals	Maharashtra	rice	NaN	NaN	NaN	NaN	NaN	N
2	23.004459	82.320763	7.840207	263.964248	5.7	Kharif	cereals	Punjab	rice	NaN	NaN	NaN	NaN	NaN	N
3	26.491096	80.158363	6.980401	242.864034	5.8	Kharif	cereals	HimachalPradesh	rice	NaN	NaN	NaN	NaN	NaN	N
4	20.130175	81.604873	7.628473	262.717340	5.9	Kharif	cereals	WestBengal	rice	NaN	NaN	NaN	NaN	NaN	N

```python
df.shape
```

```
(2212, 19)
```

+ Code + Markdown

```python
df.columns
```

```
Index(['temperature', 'humidity', 'ph', 'rainfall', 'ph after harvest',
       'Season ', 'DIVISIONS', 'States', 'label', 'Unnamed: 9', 'Unnamed: 10',
       'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14',
       'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18'],
      dtype='object')
```

```python
print(df)
```

```
      temperature   humidity        ph    rainfall  ph after harvest  \
0       20.879744  82.002744  6.502985  202.935536               5.5
1       21.770462  80.319644  7.038096  226.655537               5.6
2       23.004459  82.320763  7.840207  263.964248               5.7
3       26.491096  80.158363  6.980401  242.864034               5.8
4       20.130175  81.604873  7.628473  262.717340               5.9
...           ...        ...       ...         ...               ...
2207          NaN        NaN       NaN         NaN               NaN
2208          NaN        NaN       NaN         NaN               NaN
2209          NaN        NaN       NaN         NaN               NaN
2210          NaN        NaN       NaN         NaN               NaN
2211          NaN        NaN       NaN         NaN               NaN

       Season  DIVISIONS          States label  Unnamed: 9  Unnamed: 10  \
0     Kharif     cereals     UttarPradesh  rice         NaN          NaN
1     Kharif     cereals      Maharashtra  rice         NaN          NaN
2     Kharif     cereals           Punjab  rice         NaN          NaN
3     Kharif     cereals  HimachalPradesh  rice         NaN          NaN
4     Kharif     cereals       WestBengal  rice         NaN          NaN
...      ...         ...              ...   ...         ...          ...
2207      NaN         NaN              NaN   NaN         NaN          NaN
2208      NaN         NaN              NaN   NaN         NaN          NaN
2209      NaN         NaN              NaN   NaN         NaN          NaN
2210      NaN         NaN              NaN   NaN         NaN          NaN
2211      NaN         NaN              NaN   NaN         NaN          NaN
...
2210          NaN          NaN          NaN
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2212 entries, 0 to 2211
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   temperature       2200 non-null   float64
 1   humidity          2200 non-null   float64
 2   ph                2200 non-null   float64
 3   rainfall          2200 non-null   float64
 4   ph after harvest  2200 non-null   float64
 5   Season            2200 non-null   object
 6   DIVISIONS         2200 non-null   object
 7   States            2200 non-null   object
 8   label             2200 non-null   object
 9   Unnamed: 9        0 non-null      float64
 10  Unnamed: 10       0 non-null      float64
 11  Unnamed: 11       0 non-null      float64
 12  Unnamed: 12       0 non-null      float64
 13  Unnamed: 13       0 non-null      float64
 14  Unnamed: 14       0 non-null      float64
 15  Unnamed: 15       0 non-null      float64
 16  Unnamed: 16       0 non-null      float64
 17  Unnamed: 17       0 non-null      float64
 18  Unnamed: 18       0 non-null      float64
dtypes: float64(15), object(4)
```

```python
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('Farmfutro.csv')

# Check for null values
null_values = df.isnull()

# Count null values in each column
null_count_per_column = df.isnull().sum()

# Count total null values in the DataFrame
total_null_count = df.isnull().sum().sum()

# Display the results
print("Null values in each column:")
print(null_count_per_column)

print("\nTotal null values in the DataFrame:", total_null_count)
```

Python

```
Null values in each column:
temperature          12
humidity             12
ph                   12
rainfall             12
ph after harvest     12
Season               12
DIVISIONS            12
States               12
label                12
Unnamed: 9         2212
Unnamed: 10        2212
Unnamed: 11        2212
Unnamed: 12        2212
Unnamed: 13        2212
Unnamed: 14        2212
Unnamed: 15        2212
Unnamed: 16        2212
Unnamed: 17        2212
Unnamed: 18        2212
dtype: int64

Total null values in the DataFrame: 22228
```


```python
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('Farmfutro.csv')

# Check for null values
null_values = df.isnull()

# Count null values in each column
null_count_per_column = df.isnull().sum()

# Count total null values in the DataFrame
total_null_count = df.isnull().sum().sum()

# Display the results
print("Null values in each column:")
print(null_count_per_column)

print("\nTotal null values in the DataFrame:", total_null_count)
```

Python

```
Null values in each column:
temperature          12
humidity             12
ph                   12
rainfall             12
ph after harvest     12
Season               12
```

```python
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('Farmfutro.csv')

# Remove columns with the name "Unnamed"
df_cleaned = df.loc[:, ~df.columns.str.contains('^Unnamed')]

# Remove rows with NaN values
df_cleaned = df_cleaned.dropna()

# Display the cleaned DataFrame
print("\nCleaned DataFrame (NaN values and 'Unnamed' columns removed):")
print(df_cleaned)

# Save the cleaned DataFrame to a new CSV file
df_cleaned.to_csv('Farmfutro_cleaned.csv', index=False)
```

Python

```
Cleaned DataFrame (NaN values and 'Unnamed' columns removed):
      temperature   humidity        ph    rainfall  ph after harvest  \
0       20.879744  82.002744  6.502985  202.935536               5.5
1       21.770462  80.319644  7.038096  226.655537               5.6
2       23.004459  82.320763  7.840207  263.964248               5.7
3       26.491096  80.158363  6.980401  242.864034               5.8
4       20.130175  81.604873  7.628473  262.717340               5.9
```

```python
import pandas as pd
import ipywidgets as widgets
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import OneHotEncoder

# Read the cleaned CSV file into a DataFrame
dataset = pd.read_csv('Farmfutro_cleaned.csv')

# Display column names
print("Column Names:", dataset.columns)
```

Python

```
Column Names: Index(['temperature', 'humidity', 'ph', 'rainfall', 'ph after harvest',
       'Season ', 'DIVISIONS', 'States', 'label'],
      dtype='object')
```

```python
import ipywidgets as widgetsa
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Python

```python
import pandas as pd

# Assuming your DataFrame is named df1
print(df1.columns)

# Check for any leading or trailing whitespaces in column names
df1.columns = df1.columns.str.strip()

# Select features for the model
features = df1[['temperature', 'humidity', 'ph', 'rainfall', 'ph after harvest', 'Season']]
```

Python

```
Index(['temperature', 'humidity', 'ph', 'rainfall', 'ph after harvest',
       'Season ', 'DIVISIONS', 'States', 'label'],
      dtype='object')
```

+ Code + Markdown

```python
from sklearn.preprocessing import LabelEncoder

# Encode 'Season' column
le = LabelEncoder()
features['Season'] = le.fit_transform(features['Season'])
```

Python

```
<ipython-input-19-db518cdc2714>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```python
from sklearn.model_selection import train_test_split

# Split the dataset into features (X) and labels (y)
X = features
y = df1['label']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Python

```python
from sklearn.ensemble import RandomForestClassifier

# Create a Random Forest Classifier
model = RandomForestClassifier()

# Train the model
model.fit(X_train, y_train)
```

Python

```python
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Python

```
Accuracy: 0.990909090909091
Classification Report:
              precision    recall  f1-score   support

       apple       0.96      0.96      0.96        23
      banana       1.00      1.00      1.00        21
   blackgram       0.95      1.00      0.98        20
    chickpea       1.00      1.00      1.00        26
     coconut       1.00      1.00      1.00        27
      coffee       1.00      1.00      1.00        17
      cotton       1.00      1.00      1.00        17
      grapes       1.00      1.00      1.00        14
        jute       1.00      1.00      1.00        23
 kidneybeans       1.00      1.00      1.00        20
      lentil       1.00      1.00      1.00        11
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        19
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        19
   mothbeans       1.00      0.96      0.98        24
    mungbean       1.00      0.95      0.97        19
   muskmelon       1.00      1.00      1.00        17
      orange       0.93      0.93      0.93        14
      papaya       1.00      1.00      1.00        23
  pigeonpeas       1.00      1.00      1.00        23
 pomegranate       1.00      1.00      1.00        23
        rice       1.00      1.00      1.00        19
...
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19]]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
import joblib

# Save the model to a file
joblib.dump(model, 'crop_recommendation_model.joblib')
```

Python

```
['crop_recommendation_model.joblib']
```

```
def train_model(change):
    state = states_widget.value
    season = season_widget.value

    # Check if the widgets have values
    if all(widget.value and widget.value.strip() for widget in widgets_dict.values()) and state and season:
        try:
            temperature = float(temperature_widget.value)
            humidity = float(humidity_widget.value)
            ph = float(ph_widget.value)
            rainfall = float(rainfall_widget.value)
            ph_after_harvest = float(ph_after_harvest_widget.value)
            division = division_widget.value

            input_data = pd.DataFrame(
                [[temperature, humidity, ph, rainfall, ph_after_harvest, state, season, division]],
                columns=['temperature', 'humidity', 'ph', 'rainfall', 'ph after harvest', 'States', 'Season ', 'DIVISIONS']
            )

            # One-hot encoding
            encoder = OneHotEncoder(handle_unknown='ignore')
            X_encoded = encoder.fit_transform(df1[['States', 'Season ', 'DIVISIONS']])
            input_data_encoded = encoder.transform(input_data[['States', 'Season ', 'DIVISIONS']])

            # Training and test split
            X_train, X_test, y_train, y_test = train_test_split(X_encoded, df1['label'], test_size=0.2, random_state=42)

            # Train the random forest model
```

```
            encoder = OneHotEncoder(handle_unknown='ignore')
            X_encoded = encoder.fit_transform(df1[['States', 'Season ', 'DIVISIONS']])
            input_data_encoded = encoder.transform(input_data[['States', 'Season ', 'DIVISIONS']])

            # Training and test split
            X_train, X_test, y_train, y_test = train_test_split(X_encoded, df1['label'], test_size=0.2, random_state=42)

            # Train the random forest model
            model_crop = RandomForestClassifier(n_estimators=100, random_state=42)
            model_crop.fit(X_train, y_train)

            # Evaluate the model on the test set
            y_pred = model_crop.predict(X_test)

            # Calculate accuracy
            accuracy = accuracy_score(y_test, y_pred)

            with recommend_widget:
                recommend_widget.clear_output()
                print("Model Accuracy:", accuracy)

            # Print classification report
            print("Classification Report:")
            print(classification_report(y_test, y_pred))

            # Print confusion matrix
            print("Confusion Matrix:")
            print(confusion_matrix(y_test, y_pred))
```

```
            # Fertilizer associated with the recommended crop
            recommended_fertilizer = df1[df1['label'] == predicted_crop[0]]['Fertilizer'].values[0]
            with recommend_widget:
                recommend_widget.clear_output()
                print("Recommended Crop:", predicted_crop[0])
                print("Recommended Fertilizer:", recommended_fertilizer)
        except ValueError as e:
            with recommend_widget:
                recommend_widget.clear_output()
                print(f"Error: {e}")
    else:
        with recommend_widget:
            recommend_widget.clear_output()
            print("Please select values for all features, state, and season.")
```

```
import pandas as pd
import ipywidgets as widgets
from IPython.display import display
import joblib
from sklearn.preprocessing import LabelEncoder

# Load the saved model
loaded_model = joblib.load('crop_recommendation_model.joblib')

# Create input widgets for each feature
temperature_widget = widgets.FloatText(description='Temperature:')
humidity_widget = widgets.FloatText(description='Humidity:')
ph_widget = widgets.FloatText(description='pH:')
rainfall_widget = widgets.FloatText(description='Rainfall:')
ph_after_harvest_widget = widgets.FloatText(description='pH after harvest:')
season_widget = widgets.Dropdown(options=['Rabi', 'Kharif', 'Summer'], description='Season:')
predict_button = widgets.Button(description='Predict')
output_widget = widgets.Output()

# Define a LabelEncoder for the 'Season' column
le = LabelEncoder()
```

```python
# Define a LabelEncoder for the 'Season' column
le = LabelEncoder()

# Function to make a prediction
def make_prediction(_):
    # Get user input values
    user_input = {
        'temperature': temperature_widget.value,
        'humidity': humidity_widget.value,
        'ph': ph_widget.value,
        'rainfall': rainfall_widget.value,
        'ph after harvest': ph_after_harvest_widget.value,
        'Season': season_widget.value
    }

    # Create a DataFrame from user input
    new_data = pd.DataFrame(user_input, index=[0])

    # Encode 'Season' column
    new_data['Season'] = le.fit_transform(new_data['Season'])

    # Make a prediction
    predicted_crop = loaded_model.predict(new_data)
```

```python
    # Display the prediction
    with output_widget:
        output_widget.clear_output()
        print("Recommended Crop:", predicted_crop[0])

# Attach the function to the button's click event
predict_button.on_click(make_prediction)

# Display the input widgets, button, and output widget
display(temperature_widget, humidity_widget, ph_widget, rainfall_widget, ph_after_harvest_widget, season_widget, predict_button, output_widget)

# Function to recommend environmental factors for a given crop
def recommend_environment(crop):
    # Assuming you have a function or dataset to retrieve environmental factors for a given crop
    # Replace the following line with actual logic to get environmental factors for the crop
    recommended_environment = {'Temperature': 25.0, 'Humidity': 75.0, 'pH': 6.0, 'Rainfall': 150.0, 'pH after harvest': 6.0, 'Season': 'Kharif'}

    # Update the input widgets with recommended environmental factors
    temperature_widget.value = recommended_environment['Temperature']
    humidity_widget.value = recommended_environment['Humidity']
    ph_widget.value = recommended_environment['pH']
    rainfall_widget.value = recommended_environment['Rainfall']
    ph_after_harvest_widget.value = recommended_environment['pH after harvest']
    season_widget.value = recommended_environment['Season']

# Example: Recommend environmental factors for 'jute'
recommend_environment('jute')
```

```python
import pandas as pd
import ipywidgets as widgets
from IPython.display import display
import joblib

# Load the saved model
loaded_model = joblib.load('crop_recommendation_model.joblib')

# Assuming your DataFrame is named df
# Create a DataFrame with unique labels
unique_labels = df['label'].unique()
unique_labels = [label for label in unique_labels if pd.notna(label)]  # Remove nan
crop_selector_widget = widgets.Dropdown(options=['Select Crop'] + list(unique_labels), description='Select Crop:')

# Create a button to trigger the recommendation
recommend_button = widgets.Button(description='Recommend Factors')

# Create an output widget to display the recommended factors
output_widget = widgets.Output()

# Dictionary of recommended factors for each crop
recommended_factors_mapping = {
    'rice': {
        'temperature': '25-30 °C',
        'humidity': '60-80%',
        'ph': '6.0-7.5',
        'rainfall': '100-150 mm',
        'ph_after_harvest': '5.5-6.5',
        'season': 'Kharif'
    },
    'maize': {
        'temperature': '20-30 °C',
        'humidity': '50-70%',
        'ph': '5.8-7.0',
        'rainfall': '50-100 mm',
        'ph_after_harvest': '5.5-6.5',
        'season': 'Kharif'
    },
    # Add entries for other crops
}

# Define a function to recommend factors based on selected crop
def recommend_factors(_):
    # Get selected crop
    selected_crop = crop_selector_widget.value

    # Display the selected crop
    with output_widget:
        output_widget.clear_output()
        print("Selected Crop:", selected_crop)

        # Recommend factors for the selected crop
        if selected_crop in recommended_factors_mapping:
            recommended_factors = recommended_factors_mapping[selected_crop]
            print("\nRecommended Environmental Factors for", selected_crop, ":")
            print(recommended_factors)
        else:
            print("No recommendations available for", selected_crop)

# Attach the function to the button's click event
recommend_button.on_click(recommend_factors)

# Display the input widgets, button, and output widget
display(crop_selector_widget, recommend_button, output_widget)
```

```
Select Crop: rice
Recommend Factors
Selected Crop: rice

Recommended Environmental Factors for rice :
{'temperature': '25-30 °C', 'humidity': '60-80%', 'ph': '6.0-7.5', 'rainfall': '100-150 mm', 'ph_after_harvest': '5.5-6.5', 'season': 'Kharif'}
```

```python
import pandas as pd
import ipywidgets as widgets
from IPython.display import display

# Load the cleaned DataFrame
df1 = pd.read_csv('Farmfutro_cleaned (1).csv')

# Create a DataFrame with unique labels
unique_labels = df1['label'].unique()
unique_labels = [label for label in unique_labels if pd.notna(label)]  # Remove nan
crop_selector_widget = widgets.Dropdown(options=['Select Crop'] + list(unique_labels), description='Select Crop:')

# Create a button to trigger the recommendation
recommend_button = widgets.Button(description='Recommend Factors')

# Create an output widget to display the recommended factors
output_widget = widgets.Output()

# Define a function to recommend factors and location based on selected crop
def get_recommendations_for_crop(crop):
    # Extract data for the selected crop
    crop_data = df1[df1['label'] == crop].iloc[0]

    # Define recommended factors
    recommended_factors = {
        'temperature': f"{crop_data['temperature']} °C",
        'humidity': f"{crop_data['humidity']}%",
        'ph': f"{crop_data['ph']}",
        'rainfall': f"{crop_data['rainfall']} mm",
        'ph_after_harvest': f"{crop_data['ph_after_harvest']}",
        'season': f"{crop_data['season']}"
    }
    # Recommended locations
    recommended_locations = df1[df1['label'] == crop]['States'].unique()
    return recommended_factors, recommended_locations

# Define a function to recommend factors based on selected crop
def recommend_factors(_):
    # Get selected crop
    selected_crop = crop_selector_widget.value

    # Display the selected crop and recommended factors
    with output_widget:
        output_widget.clear_output()
        print("Selected Crop:", selected_crop)

        # Recommend factors and locations for the selected crop
        if selected_crop != 'Select Crop':
            recommended_factors, recommended_locations = get_recommendations_for_crop(selected_crop)

            print("\nRecommended Environmental Factors for", selected_crop, ":")
            print(recommended_factors)

            print("\nRecommended Locations for", selected_crop, ":")
            print(', '.join(recommended_locations))

# Attach the function to the button's click event
recommend_button.on_click(recommend_factors)

# Display the input widgets, button, and output widget
display(crop_selector_widget, recommend_button, output_widget)
```

Select Crop: rice

Recommend Factors

```
Selected Crop: rice

Recommended Environmental Factors for rice :
{'temperature': '20.87974371 °C', 'humidity': '82.00274423%', 'ph': '6.502985292', 'rainfall': '202.9355362 mm', 'ph_after_harvest': '5.5', 'season': 'Kharif  '}

Recommended Locations for rice :
UttarPradesh, Maharashtra, Punjab, HimachalPradesh, WestBengal, Orissa, NorthEast, TamilNadu, Kerela, AndhraPradesh, Telengana, Bihar, Chattisgarh, Haryana, UttaraKhand,
JammuKashmir, Karnataka, Goa, Jharkhand
```

```python
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.svm import SVC

# Assuming you have features (X) and labels (y) ready
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an SVM model
svm_model = SVC()

# Train the SVM model
svm_model.fit(X_train, y_train)

# Make predictions for the SVM model on the validation set
y_pred_svm = svm_model.predict(X_test)

# Evaluate the SVM model
print("SVM Model:")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
```

```
SVM Model:
Accuracy: 0.6477272727272727
```

```
[ ] from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split

    # Assuming you have features (X) and labels (y) ready
    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Create a Logistic Regression model
    logreg_model = LogisticRegression()

    # Train the Logistic Regression model
    logreg_model.fit(X_train, y_train)

    # Make predictions for the Logistic Regression model on the validation set
    y_pred_logreg = logreg_model.predict(X_test)

    # Evaluate the Logistic Regression model
    print("Logistic Regression Model:")
    print("Accuracy:", accuracy_score(y_test, y_pred_logreg))
    print("Classification Report:\n", classification_report(y_test, y_pred_logreg))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_logreg))
```

```
Logistic Regression Model:
Accuracy: 0.7613636363636364
```

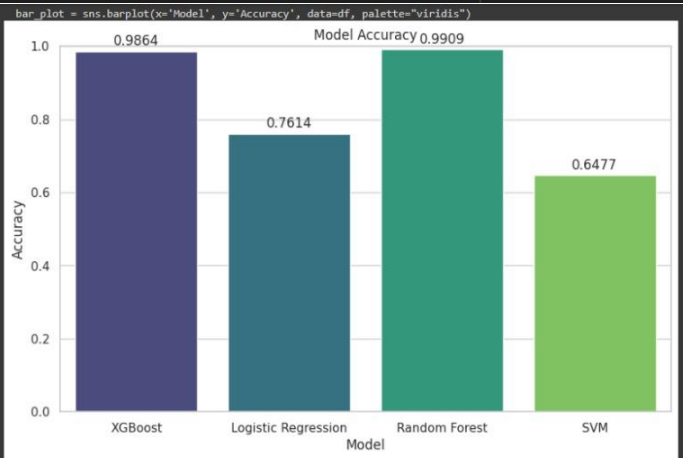
```
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split

# Assuming you have features (X) and labels (y) ready
# Use LabelEncoder to convert string labels to integers
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Create an XGBoost Classifier
xgb_model = XGBClassifier()

# Train the XGBoost model
xgb_model.fit(X_train, y_train)

# Make predictions for the XGBoost model on the validation set
y_pred_xgb = xgb_model.predict(X_test)

# Reverse transformation to get the original labels
y_pred_original_labels = label_encoder.inverse_transform(y_pred_xgb)
```

```
# Make predictions for the XGBoost model on the validation set
y_pred_xgb = xgb_model.predict(X_test)

# Reverse transformation to get the original labels
y_pred_original_labels = label_encoder.inverse_transform(y_pred_xgb)

# Evaluate the XGBoost model
print("XGBoost Model:")
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_xgb))

# Optionally, print the predicted labels in their original form
print("\nPredicted Labels (Original Form):", y_pred_original_labels)
```

```
XGBoost Model:
Accuracy: 0.9863636363636363
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style for Seaborn
sns.set(style="whitegrid")

# Create a bar plot for model accuracies
plt.figure(figsize=(10, 6))
bar_plot = sns.barplot(x='Model', y='Accuracy', data=df, palette="viridis")
plt.title('Model Accuracy')
plt.ylim(0, 1)  # Set the y-axis limit to represent accuracy percentage

# Display the accuracy values on top of the bars
for p in bar_plot.patches:
    bar_plot.annotate(format(p.get_height(), '.4f'),
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
                      xytext = (0, 9),
                      textcoords = 'offset points')

plt.show()
```

```
bar_plot = sns.barplot(x='Model', y='Accuracy', data=df, palette="viridis")
```



4.2 Implementation of Crop Disease Detection

```python
source_dir = '/kaggle/input/dieseas/Dieseas'

# Define destination directory to move folders containing 'Apple' images
destination_dir = '/kaggle/working/Wheat_folders/'
```

[1] Python

```python
import os
import shutil
```

[2] Python

```python
os.makedirs(destination_dir, exist_ok=True)

# Iterate through each folder in the source directory
for folder_name in os.listdir(source_dir):
    folder_path = os.path.join(source_dir, folder_name)
    if os.path.isdir(folder_path):
        # Check if the folder name contains 'Apple'
        if 'Wheat' in folder_name:
            # Copy the folder to the destination directory
            shutil.copytree(folder_path, os.path.join(destination_dir, folder_name))
```

[3] Python

```python
import random
data_dir = '/kaggle/working/Wheat_folders'
train_dir = '/kaggle/working/train/'
test_dir = '/kaggle/working/test/'
val_dir = '/kaggle/working/validation/'

# Create directories if they don't exist
for directory in [train_dir, test_dir, val_dir]:
    if not os.path.exists(directory):
        os.makedirs(directory)

# Define ratio for train, test, and validation sets
train_ratio = 0.7
test_ratio = 0.2
val_ratio = 0.1

# Function to split data
def split_data(source, train, test, val):
    files = os.listdir(source)
    random.shuffle(files)
    train_size = int(train_ratio * len(files))
    test_size = int(test_ratio * len(files))
    val_size = int(val_ratio * len(files))

    train_files = files[:train_size]
    test_files = files[train_size:train_size + test_size]
    val_files = files[train_size + test_size:]

    for file in train_files:
        dest_path = os.path.join(train, os.path.basename(file))
        os.makedirs(os.path.dirname(dest_path), exist_ok=True)
        shutil.copy(os.path.join(source, file), dest_path)

    for file in test_files:
        dest_path = os.path.join(test, os.path.basename(file))
        os.makedirs(os.path.dirname(dest_path), exist_ok=True)
        shutil.copy(os.path.join(source, file), dest_path)

    for file in val_files:
        dest_path = os.path.join(val, os.path.basename(file))
        os.makedirs(os.path.dirname(dest_path), exist_ok=True)
        shutil.copy(os.path.join(source, file), dest_path)
# Split data for each labeled folder
for label_folder in os.listdir(data_dir):
    label_path = os.path.join(data_dir, label_folder)
    if os.path.isdir(label_path):
        split_data(label_path,
                   os.path.join(train_dir, label_folder),
                   os.path.join(test_dir, label_folder),
                   os.path.join(val_dir, label_folder))
```

[4]

```python
# Count folders in each directory
train_folder_count = len(os.listdir(train_dir))
test_folder_count = len(os.listdir(test_dir))
val_folder_count = len(os.listdir(val_dir))

# Display the counts
print("Number of folders in train directory:", train_folder_count)
print("Number of folders in test directory:", test_folder_count)
print("Number of folders in validation directory:", val_folder_count)
```

[5]

```
Number of folders in train directory: 4
Number of folders in test directory: 4
Number of folders in validation directory: 4
```

```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.mobilenet import preprocess_input
from tensorflow.keras.applications.mobilenet import MobileNet
from tensorflow.keras import layers, models
```

[6]

```python
# Data directories
train_dir = '/kaggle/working/train/'
test_dir = '/kaggle/working/test/'
val_dir = '/kaggle/working/validation/'

# Image size and batch size
img_size = (224, 224)
batch_size = 32

# Data augmentation for training data
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Data generators
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)
```

```python
# Data generators
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

valid_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
valid_data = valid_datagen.flow_from_directory(
    val_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)
```

[7]

```
Found 2345 images belonging to 4 classes.
Found 669 images belonging to 4 classes.
Found 340 images belonging to 4 classes.
```

```python
import os
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

# Define class names
class_names = sorted(os.listdir(train_dir))

# Plot one image from each class
plt.figure(figsize=(10, 6))

for i, class_name in enumerate(class_names):
    # Find a random image from the class directory
    class_dir = os.path.join(train_dir, class_name)
    image_files = os.listdir(class_dir)
    image_path = os.path.join(class_dir, image_files[0])  # Take the first image from the class
    image = Image.open(image_path)

    # Plot the image
    plt.subplot(1, len(class_names), i + 1)
    plt.imshow(image)
    plt.title(class_name)
    plt.axis('off')

plt.show()
```

[14] Python

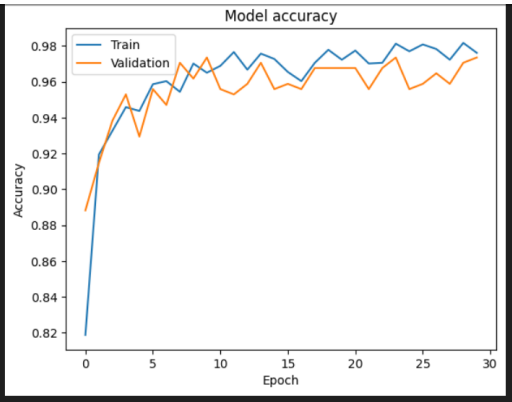

```python
Weights = '/kaggle/input/mobile-net/mobilenet_1_0_224_tf_no_top.h5'

base_model = MobileNet(weights=Weights, include_top=False, input_shape=(img_size[0], img_size[1], 3))
base_model.trainable = False

model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(1024, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(4, activation='softmax')
    ])

    # Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[15] Python

```python
import matplotlib.pyplot as plt

history = model.fit(train_data, epochs=30, validation_data=valid_data)

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

```
Epoch 1/30
74/74 [==============================] - 86s 1s/step - loss: 0.6054 - accuracy: 0.8188 - val_loss: 0.2748 - val_accuracy: 0.8882
Epoch 2/30
74/74 [==============================] - 81s 1s/step - loss: 0.2115 - accuracy: 0.9194 - val_loss: 0.2217 - val_accuracy: 0.9147
Epoch 3/30
74/74 [==============================] - 80s 1s/step - loss: 0.1842 - accuracy: 0.9326 - val_loss: 0.1550 - val_accuracy: 0.9382
Epoch 4/30
74/74 [==============================] - 83s 1s/step - loss: 0.1528 - accuracy: 0.9458 - val_loss: 0.1263 - val_accuracy: 0.9529
Epoch 5/30
74/74 [==============================] - 80s 1s/step - loss: 0.1491 - accuracy: 0.9437 - val_loss: 0.1705 - val_accuracy: 0.9294
Epoch 6/30
74/74 [==============================] - 81s 1s/step - loss: 0.1061 - accuracy: 0.9586 - val_loss: 0.1070 - val_accuracy: 0.9559
Epoch 7/30
74/74 [==============================] - 81s 1s/step - loss: 0.1177 - accuracy: 0.9603 - val_loss: 0.1536 - val_accuracy: 0.9471
Epoch 8/30
74/74 [==============================] - 80s 1s/step - loss: 0.1195 - accuracy: 0.9544 - val_loss: 0.1033 - val_accuracy: 0.9706
Epoch 9/30
74/74 [==============================] - 80s 1s/step - loss: 0.0862 - accuracy: 0.9701 - val_loss: 0.1229 - val_accuracy: 0.9618
Epoch 10/30
74/74 [==============================] - 80s 1s/step - loss: 0.0972 - accuracy: 0.9650 - val_loss: 0.0887 - val_accuracy: 0.9735
Epoch 11/30
74/74 [==============================] - 80s 1s/step - loss: 0.0931 - accuracy: 0.9689 - val_loss: 0.1276 - val_accuracy: 0.9559
Epoch 12/30
74/74 [==============================] - 82s 1s/step - loss: 0.0706 - accuracy: 0.9765 - val_loss: 0.1256 - val_accuracy: 0.9529
Epoch 13/30
...
Epoch 29/30
74/74 [==============================] - 80s 1s/step - loss: 0.0551 - accuracy: 0.9817 - val_loss: 0.0826 - val_accuracy: 0.9706
Epoch 30/30
74/74 [==============================] - 81s 1s/step - loss: 0.0703 - accuracy: 0.9761 - val_loss: 0.0884 - val_accuracy: 0.9735
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...




```
# Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(test_data)

# Print the testing accuracy
print("Testing Accuracy:", test_accuracy)
```

[18]

```
21/21 [==============================] - 13s 631ms/step - loss: 0.0799 - accuracy: 0.9656
Testing Accuracy: 0.9656203389167786
```

```
import tensorflow as tf


model.save('mobilenet_Wheat1.h5')
```

[20]

```
/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`.
  saving_api.save_model(
```

```
from keras.applications.resnet import ResNet50

# Specify the path to the downloaded weights file
weights_path = '/kaggle/input/resnet50/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5'

# Create ResNet50 model with local weights
base_model = ResNet50(weights=weights_path, include_top=False, input_shape=(img_size[0], img_size[1], 3))
base_model.trainable = False
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(4, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])


history = model.fit(train_data, epochs=30, validation_data=valid_data)
```

```python
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

[21] Python

```
Epoch 1/30
74/74 [==============================] - 89s 1s/step - loss: 0.9592 - accuracy: 0.6090 - val_loss: 0.7159 - val_accuracy: 0.7118
Epoch 2/30
74/74 [==============================] - 84s 1s/step - loss: 0.7605 - accuracy: 0.6981 - val_loss: 0.6819 - val_accuracy: 0.7382
Epoch 3/30
74/74 [==============================] - 80s 1s/step - loss: 0.7231 - accuracy: 0.7122 - val_loss: 0.6477 - val_accuracy: 0.7382
Epoch 4/30
74/74 [==============================] - 81s 1s/step - loss: 0.7229 - accuracy: 0.7232 - val_loss: 0.6434 - val_accuracy: 0.7471
Epoch 5/30
74/74 [==============================] - 80s 1s/step - loss: 0.6974 - accuracy: 0.7258 - val_loss: 0.6323 - val_accuracy: 0.7529
Epoch 6/30
74/74 [==============================] - 80s 1s/step - loss: 0.6948 - accuracy: 0.7241 - val_loss: 0.5864 - val_accuracy: 0.7471
Epoch 7/30
74/74 [==============================] - 81s 1s/step - loss: 0.6611 - accuracy: 0.7399 - val_loss: 0.6040 - val_accuracy: 0.7618
Epoch 8/30
74/74 [==============================] - 80s 1s/step - loss: 0.6704 - accuracy: 0.7446 - val_loss: 0.6425 - val_accuracy: 0.7588
Epoch 9/30
74/74 [==============================] - 81s 1s/step - loss: 0.6476 - accuracy: 0.7407 - val_loss: 0.5884 - val_accuracy: 0.7706
Epoch 10/30
74/74 [==============================] - 81s 1s/step - loss: 0.6568 - accuracy: 0.7518 - val_loss: 0.5668 - val_accuracy: 0.7647
Epoch 11/30
74/74 [==============================] - 82s 1s/step - loss: 0.6476 - accuracy: 0.7467 - val_loss: 0.6047 - val_accuracy: 0.7441
Epoch 12/30
74/74 [==============================] - 80s 1s/step - loss: 0.6412 - accuracy: 0.7595 - val_loss: 0.5690 - val_accuracy: 0.7618
Epoch 13/30
```

```
74/74 [==============================] - 80s 1s/step - loss: 0.6412 - accuracy: 0.7595 - val_loss: 0.5690 - val_accuracy: 0.7618
Epoch 13/30
...
Epoch 29/30
74/74 [==============================] - 80s 1s/step - loss: 0.5816 - accuracy: 0.7706 - val_loss: 0.5281 - val_accuracy: 0.7735
Epoch 30/30
74/74 [==============================] - 80s 1s/step - loss: 0.5646 - accuracy: 0.7765 - val_loss: 0.5034 - val_accuracy: 0.7824
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...


```python
test_loss, test_accuracy = model.evaluate(test_data)

# Print the testing accuracy
print("Testing Accuracy:", test_accuracy)
```

[22] Python

```
21/21 [==============================] - 15s 727ms/step - loss: 0.5170 - accuracy: 0.8087
Testing Accuracy: 0.8086696267127991
```

```python
model = models.Sequential([
        layers.Conv2D(96, kernel_size=(11, 11), strides=(4, 4), activation='relu', input_shape=(img_size[0], img_size[1], 3)),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
        layers.Conv2D(256, kernel_size=(5, 5), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
        layers.Conv2D(384, kernel_size=(3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.Conv2D(384, kernel_size=(3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
        layers.Flatten(),
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(4, activation='softmax')
    ])
```

wheat-disease (3).ipynb

E: > BISAG > CODE > wheat-disease (3).ipynb > ...

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· Select Kernel

```
    # Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

import matplotlib.pyplot as plt

history = model.fit(train_data, epochs=30, validation_data=valid_data)

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

[23] Python

```
Epoch 1/30
74/74 [==============================] - 90s 1s/step - loss: 8.6213 - accuracy: 0.6111 - val_loss: 5.8045 - val_accuracy: 0.4794
Epoch 2/30
74/74 [==============================] - 82s 1s/step - loss: 2.1562 - accuracy: 0.6904 - val_loss: 2.1894 - val_accuracy: 0.4588
Epoch 3/30
74/74 [==============================] - 80s 1s/step - loss: 0.8892 - accuracy: 0.7262 - val_loss: 1.2746 - val_accuracy: 0.5206
Epoch 4/30
74/74 [==============================] - 79s 1s/step - loss: 0.8982 - accuracy: 0.7505 - val_loss: 1.6251 - val_accuracy: 0.4853
Epoch 5/30
74/74 [==============================] - 82s 1s/step - loss: 0.5785 - accuracy: 0.8000 - val_loss: 1.5501 - val_accuracy: 0.5853
Epoch 6/30
74/74 [==============================] - 80s 1s/step - loss: 0.6367 - accuracy: 0.7876 - val_loss: 0.7592 - val_accuracy: 0.7471
Epoch 7/30
74/74 [==============================] - 79s 1s/step - loss: 0.5463 - accuracy: 0.8098 - val_loss: 0.6239 - val_accuracy: 0.7765
Epoch 8/30
74/74 [==============================] - 80s 1s/step - loss: 0.5040 - accuracy: 0.8277 - val_loss: 0.5559 - val_accuracy: 0.7853
Epoch 9/30
```

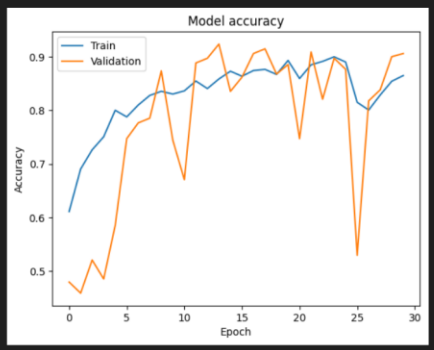
wheat-disease (3).ipynb

E: > BISAG > CODE > wheat-disease (3).ipynb > ...

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· Select Kernel

```
Epoch 12/30
74/74 [==============================] - 77s 1s/step - loss: 0.4272 - accuracy: 0.8546 - val_loss: 0.3417 - val_accuracy: 0.8882
Epoch 13/30
...
Epoch 29/30
74/74 [==============================] - 76s 1s/step - loss: 0.5307 - accuracy: 0.8542 - val_loss: 0.3978 - val_accuracy: 0.9000
Epoch 30/30
74/74 [==============================] - 81s 1s/step - loss: 0.4422 - accuracy: 0.8648 - val_loss: 0.2883 - val_accuracy: 0.9059
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...




```
test_loss, test_accuracy = model.evaluate(test_data)

# Print the testing accuracy
print("Testing Accuracy:", test_accuracy)
```

[24] Python

```
21/21 [==============================] - 13s 623ms/step - loss: 0.2160 - accuracy: 0.9238
Testing Accuracy: 0.9237667918205261
```

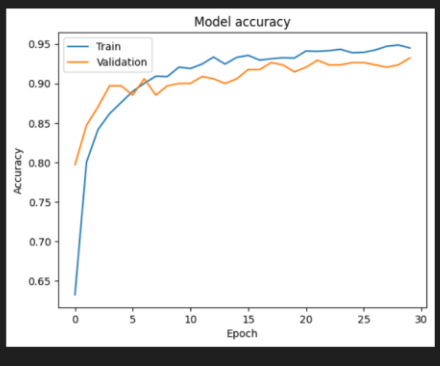
```
from tensorflow.keras.applications.vgg16 import VGG16
# Specify the path to the downloaded weights file
weights_p = '/kaggle/input/vgg-16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5'


base_model = VGG16(weights=weights_p, include_top=False, input_shape=(img_size[0], img_size[1], 3))
base_model.trainable = False
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(4, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

import matplotlib.pyplot as plt

history = model.fit(train_data, epochs=30, validation_data=valid_data)
```

```python
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

[25] Python

```
Epoch 1/30
74/74 [==============================] - 94s 1s/step - loss: 0.9055 - accuracy: 0.6324 - val_loss: 0.5893 - val_accuracy: 0.7971
Epoch 2/30
74/74 [==============================] - 77s 1s/step - loss: 0.5548 - accuracy: 0.8004 - val_loss: 0.4217 - val_accuracy: 0.8471
Epoch 3/30
74/74 [==============================] - 80s 1s/step - loss: 0.4443 - accuracy: 0.8418 - val_loss: 0.3750 - val_accuracy: 0.8706
Epoch 4/30
74/74 [==============================] - 78s 1s/step - loss: 0.3863 - accuracy: 0.8618 - val_loss: 0.3128 - val_accuracy: 0.8971
Epoch 5/30
74/74 [==============================] - 77s 1s/step - loss: 0.3398 - accuracy: 0.8759 - val_loss: 0.2997 - val_accuracy: 0.8971
Epoch 6/30
74/74 [==============================] - 77s 1s/step - loss: 0.3069 - accuracy: 0.8900 - val_loss: 0.2895 - val_accuracy: 0.8853
Epoch 7/30
74/74 [==============================] - 76s 1s/step - loss: 0.2843 - accuracy: 0.9002 - val_loss: 0.2517 - val_accuracy: 0.9059
Epoch 8/30
74/74 [==============================] - 80s 1s/step - loss: 0.2640 - accuracy: 0.9092 - val_loss: 0.2740 - val_accuracy: 0.8853
Epoch 9/30
74/74 [==============================] - 77s 1s/step - loss: 0.2594 - accuracy: 0.9087 - val_loss: 0.2431 - val_accuracy: 0.8971
Epoch 10/30
74/74 [==============================] - 82s 1s/step - loss: 0.2264 - accuracy: 0.9207 - val_loss: 0.2379 - val_accuracy: 0.9000
Epoch 11/30
74/74 [==============================] - 78s 1s/step - loss: 0.2223 - accuracy: 0.9190 - val_loss: 0.2072 - val_accuracy: 0.9000
Epoch 12/30
74/74 [==============================] - 82s 1s/step - loss: 0.2129 - accuracy: 0.9245 - val_loss: 0.2169 - val_accuracy: 0.9088
Epoch 13/30
...
Epoch 29/30
74/74 [==============================] - 77s 1s/step - loss: 0.1417 - accuracy: 0.9488 - val_loss: 0.1801 - val_accuracy: 0.9235
Epoch 30/30
74/74 [==============================] - 78s 1s/step - loss: 0.1505 - accuracy: 0.9450 - val_loss: 0.1702 - val_accuracy: 0.9324
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



```
# Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(test_data)

# Print the testing accuracy
print("Testing Accuracy:", test_accuracy)
```

[26]

```
21/21 [==============================] - 21s 1s/step - loss: 0.1254 - accuracy: 0.9477
Testing Accuracy: 0.9476830959320068
```

```
import tensorflow as tf


model.save('vgg16_rice_dis.h5')
```

[31]

```
import os
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras import layers, models
```

[28]

```
saved_model = load_model('/kaggle/working/vgg16_rice_dis.h5')
```

[38]

```
img_path = '/kaggle/working/Wheat_folders/Wheat__yellow_rust/Yellow_rust523.jpg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
predictions = saved_model.predict(img_array)
predicted_class = np.argmax(predictions[0])
class_labels = list(train_data.class_indices.keys())
print(f'Predicted Crop Type: {class_labels[predicted_class]}')
```

[44]

```
1/1 [==============================] - 0s 23ms/step
Predicted Crop Type: Wheat__yellow_rust
```

5. Results

5.1 Results of Crop recommendation and Favourability

Model	Accuracy
Random Forest	0.9761
XGBoost	0.9450
Logistic Regression	0.8648
Support Vector machine	0.7765

Table 5 Results for Crop recommendation and Favourability

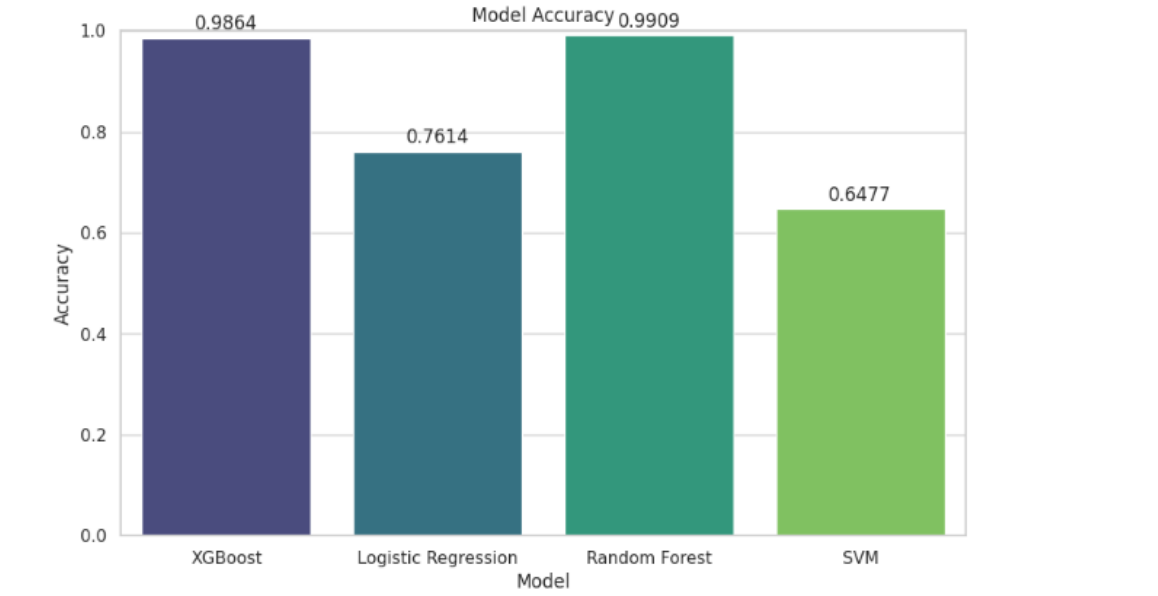


Fig 13 Chart of the results of crop recommendation and Favourability model

5.1.1 Result of Random Forest Model

Random Forest, an ensemble learning technique rooted in decision trees, demonstrates an impressive accuracy score of 0.9761. This figure reflects the model's proficiency in accurately predicting the target variable or class, achieving an accuracy rate of approximately 97.61% on unseen data. The strength of the Random Forest algorithm lies in its utilization of multiple decision trees, each trained on different subsets of the dataset and employing a voting mechanism to determine final predictions. This ensemble approach effectively mitigates overfitting and enhances the model's ability to generalize well to new data instances. Consequently, Random Forest emerges as a robust and reliable tool applicable across diverse domains requiring predictive analytics.

5.1.2 Result of XGBoost Model

XGBoost, an optimized gradient boosting algorithm, achieves a commendable accuracy of 0.9450, signifying its proficiency in predictive tasks. This accuracy score indicates that the XGBoost model accurately predicts the target variable approximately 94.50% of the time on unseen data. Although slightly lower than the accuracy of Random Forest, which stands at 97.61%, an accuracy rate of

94.50% underscores the robustness and reliability of the XGBoost algorithm in predictive modeling. Leveraging a boosting technique that sequentially builds an ensemble of weak learners, XGBoost iteratively improves prediction performance by minimizing errors in previous iterations. Consequently, XGBoost remains a powerful and widely adopted tool in machine learning, particularly suited for tasks demanding high predictive accuracy and scalability.

5.1.3 Result of Logistic Regression Model

Logistic Regression, a linear classification algorithm, achieves an accuracy of 0.8648, indicating its capability to predict the target variable with approximately 86.48% accuracy on unseen data. While this accuracy falls short of Random Forest and XGBoost, which attain accuracies of 97.61% and 94.50% respectively, the performance of Logistic Regression remains noteworthy. Despite its simplicity compared to more complex models, Logistic Regression offers a reliable approach for classification tasks, particularly in scenarios where interpretability and computational efficiency are valued. Moreover, an accuracy rate of 86.48% may still be acceptable depending on the specific context and requirements of the problem at hand. Consequently, Logistic Regression continues to be a widely utilized algorithm in various fields, contributing to the arsenal of tools available for predictive modelling and decision-making.

5.1.4 Result of Support Vector Machine (SVM) Model

Support Vector Machine (SVM), renowned for its efficacy in supervised learning tasks like classification and regression, achieves an accuracy of 0.7765. This accuracy score indicates that the SVM model accurately predicts the target variable approximately 77.65% of the time on unseen data. While SVM is recognized for its robustness and versatility, the attained accuracy of 77.65% suggests that it may not be performing as competitively as the other models in this particular scenario. However, it's important to note that SVM's performance can vary depending on factors such as the dataset characteristics, feature representation, and parameter settings. Despite the relatively lower accuracy compared to Random Forest, XGBoost, and Logistic Regression, SVM remains a valuable tool in machine learning, particularly for scenarios involving non-linear data and complex decision boundaries. Thus, while its performance in this context may be comparatively modest, SVM continues to offer a potent approach for various classification and regression tasks.

5.2 Results of Plant Disease Detection

No.	Crop	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
1	Wheat	MobileNet	0.9761	0.9735	0.9656
		VGG16	0.9450	0.9324	0.9477
		AlexNet	0.8648	0.9059	0.9238
		ResNet50	0.7765	0.7824	0.8087
2	Tea	MobileNet	0.9651	0.9594	0.9601
		VGG16	0.8753	0.8650	0.8883
		ResNet50	0.5731	0.5523	0.5381
3	Apple	MobileNet	0.9779	0.9820	0.9812
		VGG16	0.9549	0.9521	0.9530
		AlexNet	0.8449	0.6599	0.7830
		ResNet50	0.5655	0.6377	0.5960
4	Mango	MobileNet	0.9889	0.9975	0.9925
		VGG16	0.9771	0.9937	0.9800
		AlexNet	0.9275	0.9775	0.9413
		ResNet50	0.5850	0.6800	0.4663
5	Coffee	MobileNet	0.9965	0.9960	0.9998
		VGG16	0.9928	0.9990	0.9990
		AlexNet	0.9817	0.9985	0.9988
		ResNet50	0.7936	0.8670	0.8653

6	Rice	MobileNet	0.9875	0.9960	0.9890
		VGG16	0.9719	0.9853	0.9780
		AlexNet	0.8808	0.9304	0.9028
		ResNet50	0.7299	0.7550	0.7342

Table 6 Results for Plant Disease Detection

5.2.1 Results of Wheat Leaf Disease Detection

The evaluation of deep learning models for wheat disease classification highlights MobileNet as the leading performer, achieving remarkable training and testing accuracies of 97.61% and 96.56%, respectively. Its exceptional generalization capability suggests its potential for accurate disease classification in new wheat images, making it highly suitable for real-world applications, especially given its efficiency for mobile deployment. Following closely, VGG16 showcased robust performance with a training accuracy of 94.50% and a testing accuracy of 94.77%, establishing it as a reliable and competitive model. Meanwhile, AlexNet's results indicate promise, with a training accuracy of 86.48% and a testing accuracy of 92.38%, suggesting its ability to generalize knowledge to unseen data, albeit with room for improvement. In contrast, ResNet-50's performance suggests areas for enhancement, with training and testing accuracies of 77.65% and 80.87%, respectively. Overall, while MobileNet and VGG16 demonstrate strong capabilities, AlexNet and ResNet-50 offer valuable insights for potential refinement and optimization, highlighting the importance of selecting appropriate model architectures for specific tasks.

5.2.2 Results of Tea Leaf Disease Detection

The evaluation of deep learning models for wheat disease classification reveals distinct performance differences among them. MobileNet emerged as the top performer, achieving high accuracy scores across the board with a training accuracy of 96.51%, a validation accuracy of 95.94%, and a testing accuracy of 96.01%. Its strong generalization capabilities and efficiency for mobile deployment position it as a leading candidate for real-world applications. VGG16 followed closely with a consistent performance, recording a training accuracy of 87.53%, a validation accuracy of 86.50%, and a testing accuracy of 88.83%, establishing it as a reliable and competitive model for the task. In

contrast, ResNet50 exhibited lower accuracy levels, with a training accuracy of 57.31%, a validation accuracy of 55.23%, and a testing accuracy of 53.81%, indicating its limitations in capturing and generalizing the wheat disease patterns effectively. These findings emphasize the importance of selecting the right model architecture for specific tasks and highlight potential areas for further research and optimization in wheat disease classification.

5.2.3 Results of Apple Leaf Disease Detection

The evaluation of deep learning models for apple disease classification reveals varying levels of performance and effectiveness. MobileNet emerged as the leading performer with exceptional accuracy scores across all metrics, recording a training accuracy of 97.79%, a validation accuracy of 98.20%, and a testing accuracy of 98.12%. Its strong generalization capabilities and efficiency for mobile deployment make it a standout candidate for real-world applications. VGG16 followed closely with consistent and robust performance, achieving a training accuracy of 95.49%, a validation accuracy of 95.21%, and a testing accuracy of 95.30%, establishing its reliability in capturing the complexities of the dataset. In contrast, AlexNet exhibited mixed results, with a training accuracy of 84.49%, a validation accuracy of 65.99%, and a testing accuracy of 78.30%, indicating challenges in generalization. Similarly, ResNet50 lagged behind with a training accuracy of 56.55%, a validation accuracy of 63.77%, and a testing accuracy of 59.60%, suggesting limitations in learning and generalizing the apple disease patterns effectively. Overall, these findings underscore the importance of selecting the right model architecture tailored to specific tasks and highlight opportunities for further research and optimization in apple disease classification.

5.2.4 Results of Mango Leaf Disease Detection

The evaluation of deep learning models for mango disease classification reveals notable performance differences across the tested architectures. MobileNet emerged as the top performer with exceptional accuracy metrics, achieving a training accuracy of 98.89%, a validation accuracy of 99.75%, and a testing accuracy of 99.25%. Its outstanding generalization capabilities and efficiency for mobile deployment position it as a leading candidate for real-world applications. VGG16 also demonstrated strong and consistent performance, recording a training accuracy of 97.71%, a validation accuracy of 99.37%, and a testing accuracy of 98.00%, establishing its reliability and competitiveness for

mango disease classification. AlexNet exhibited commendable results with a training accuracy of 92.75%, a validation accuracy of 97.75%, and a testing accuracy of 94.13%, indicating its effectiveness in capturing and generalizing the mango disease patterns. In contrast, ResNet50 lagged behind with a training accuracy of 58.50%, a validation accuracy of 68.00%, and a testing accuracy of 46.63%, highlighting its limitations in learning and generalizing the mango disease data. Overall, these findings underscore the importance of selecting the right model architecture tailored to specific tasks and emphasize opportunities for further research and optimization in mango disease classification.

5.2.5 Results of Coffee Leaf Disease Detection

The evaluation of deep learning models for coffee disease classification reveals remarkable performance variations across the tested architectures. MobileNet emerged as the top performer, demonstrating exceptional accuracy metrics with a training accuracy of 99.65%, a validation accuracy of 99.60%, and a testing accuracy of 99.98%. Its outstanding generalization capabilities and efficiency for mobile deployment make it a standout candidate for real-world applications. VGG16 also exhibited strong and consistent performance, achieving a training accuracy of 99.28%, a validation accuracy of 99.90%, and a testing accuracy of 99.90%, establishing its reliability and competitiveness in capturing the complexities of the dataset. AlexNet followed closely with commendable results, recording a training accuracy of 98.17%, a validation accuracy of 99.85%, and a testing accuracy of 99.88%, indicating its effectiveness in learning and generalizing the coffee disease patterns. In contrast, ResNet50 lagged behind with a training accuracy of 79.36%, a validation accuracy of 86.70%, and a testing accuracy of 86.53%, highlighting its limitations in effectively learning and generalizing the coffee disease data. Overall, these findings underscore the importance of selecting the right model architecture tailored to specific tasks and emphasize opportunities for further research and optimization in coffee disease classification.

5.2.6 Results of Rice Leaf Disease Detection

The evaluation of deep learning models for rice disease classification reveals varying levels of performance across the tested architectures. MobileNet emerged as the top performer, showcasing strong accuracy metrics with a training accuracy of 98.75%, a validation accuracy of 99.60%, and a testing accuracy of 98.90%. Its robust generalization capabilities and efficiency for mobile deployment position it as a leading candidate for real-world applications. VGG16 also demonstrated solid and consistent performance,

recording a training accuracy of 97.19%, a validation accuracy of 98.53%, and a testing accuracy of 97.80%, establishing its reliability and competitiveness in capturing the complexities of the rice disease dataset. AlexNet followed with commendable results, achieving a training accuracy of 88.08%, a validation accuracy of 93.04%, and a testing accuracy of 90.28%, indicating its effectiveness in learning and generalizing the rice disease patterns. In contrast, ResNet50 lagged behind with a training accuracy of 72.99%, a validation accuracy of 75.50%, and a testing accuracy of 73.42%, highlighting its limitations in effectively capturing and generalizing the rice disease data. Overall, these findings emphasize the importance of selecting the appropriate model architecture tailored to specific requirements and highlight the potential for ongoing research and development to further improve the performance of deep learning models in rice disease classification.

6. Conclusion

6.1 Conclusion for Crop Recommendation and Favourability

The evaluation of machine learning algorithms for crop recommendation and favourability prediction highlights the strengths and weaknesses of different approaches. Random Forest emerges as the top-performing model, showcasing impressive accuracy and robustness in predicting crop favourability. XGBoost follows closely, demonstrating commendable accuracy and scalability. Logistic Regression offers a reliable and interpretable alternative, while Support Vector Machine (SVM) presents a valuable option for scenarios involving non-linear data. In conclusion, the choice of algorithm should consider factors such as accuracy, interpretability, and computational efficiency, with Random Forest and XGBoost standing out as particularly effective options for crop recommendation and favourability prediction tasks.

6.2 Conclusion for Crop Disease Detection

The evaluation of deep learning models for crop disease detection across various crops underscores the importance of selecting the right model architecture tailored to specific tasks. MobileNet consistently emerges as a top performer, demonstrating exceptional accuracy and generalization capabilities across multiple crops. VGG16 also showcases robust performance, while models like AlexNet and ResNet50 exhibit varying levels of effectiveness. Overall, these findings emphasize the potential of deep learning models to revolutionize crop disease management and contribute to sustainable agricultural practices, with MobileNet standing out as a leading candidate for real-world applications in crop disease detection.

7. Future Work

In envisioning the future of agricultural technology, our aspiration extends far beyond merely providing farmers with a tool; we aim to create an ecosystem that fundamentally transforms the way agriculture is practiced. Our vision encompasses the development of a comprehensive, integrated platform that seamlessly combines cutting-edge technologies, advanced data analytics, and farmer-centric design principles to empower agricultural communities worldwide.

At the core of this transformative ecosystem lies a sophisticated crop recommendation system, underpinned by state-of-the-art machine learning algorithms and vast repositories of agronomic knowledge. This system will not only consider traditional factors like soil type, climate, and historical crop performance but will also leverage emerging data sources such as satellite imagery, remote sensing data, and IoT-enabled sensors to provide farmers with unparalleled insights into their fields. By harnessing the power of predictive analytics and artificial intelligence, the system will generate highly personalized recommendations tailored to each farmer's specific circumstances, enabling them to optimize crop selection, planting schedules, and agronomic practices for maximum yield and profitability.

Complementing the crop recommendation system is an advanced disease detection and management platform, equipped with cutting-edge image recognition technology and real-time disease monitoring capabilities. Farmers will be able to capture images of their crops using their smartphones and upload them to the platform for instant analysis. Leveraging deep learning algorithms trained on vast datasets of plant pathology images, the platform will rapidly identify signs of disease, pest infestations, and nutrient deficiencies, providing farmers with timely diagnoses and actionable recommendations for mitigation. Furthermore, through geolocation-enabled features, the platform will facilitate the creation of dynamic disease heatmaps, allowing farmers to track the spread of diseases in real-time and coordinate targeted interventions to prevent outbreaks and minimize losses.

In addition to its predictive and diagnostic capabilities, our agricultural ecosystem will serve as a hub for knowledge exchange, collaboration, and community engagement. Farmers will have access to a wealth of educational resources, expert insights, and best practices curated specifically for their needs. Through interactive forums, webinars, and peer-to-peer networks, farmers will be able to share their experiences, learn from one another, and collectively tackle common challenges. Furthermore, by integrating social features and gamification elements, the platform

will foster a sense of community and camaraderie among farmers, driving engagement and participation in sustainable farming practices.

Looking ahead, our vision extends beyond the confines of the digital realm to encompass the physical landscape of agriculture. We envision the deployment of a network of IoT-enabled sensors, drones, and autonomous vehicles that will continuously monitor and manage crop health, soil conditions, and environmental factors in real-time. By leveraging emerging technologies such as blockchain and smart contracts, we aim to create transparent, traceable supply chains that ensure fair compensation for farmers and promote sustainable agricultural practices from farm to fork. Moreover, by partnering with governments, NGOs, and international organizations, we seek to democratize access to our platform, ensuring that farmers of all backgrounds and geographies can benefit from its transformative capabilities.

In summary, our vision for the future of agriculture is one of empowerment, innovation, and sustainability. By harnessing the power of technology, data, and community, we aim to create a world where every farmer has the tools, knowledge, and support they need to thrive in an ever-changing agricultural landscape. Together, we can build a brighter future for agriculture—one that nourishes both people and the planet for generations to come.

8. REFERENCES

[1] R. Anitha, E. Thanusree, and M. Abirami, "A Machine Learning Based Crop Recommendation System: A Survey," International Journal of Innovative Technology and Exploring Engineering, vol. 9, no. 3, pp. 881-884, 2020.

[2] J. Li, X. Li, X. Luo, S. Liu, and P. Nie, "Crop Recommendation Systems Based on Soil and Environmental Factors Using Graph Convolution Neural Network: A Systematic Literature Review," Sustainability, vol. 15, no. 1, p. 232, 2023.

[3] A. Singh, A. Singh, and R. P. Singh, "AI-Enabled Crop Recommendation System Based on Soil and Weather Patterns," International Journal of Computer Applications, vol. 182, no. 10, pp. 25-30, 2021.

[4] M.Y. Shams, S.A. Gamel, and F.M. Talaat, "Enhancing crop recommendation systems with explainable artificial intelligence: a study on agricultural decision-making," Neural Computing & Applications, pp. 1-20, 2024.

[5] S. R. Patil, S. S. Deshmukh, A. A. Chaudhari, and S. V. Deshmukh, "A Novel Crop Recommendation System for Sustainable Agriculture," International Journal of Current Microbiology and Applied Science, vol. 11, no. 4, pp. 7288-7293, 2022.

[6] H. P. Alemán, J. C. Villamil, J. C. López, R. H. Salazar, and A. I. Camacho, "Precision Agriculture: A Review of Feature Selection and Machine Learning Models for Crop Yield Prediction," Agriculture, vol. 10, no. 6, p. 182, 2020.

[7] P.K. Singh, S.K. Srivastava, D.S. Verma, and L.N. Tripathi, "Machine Learning for Crop Yield Prediction: A Review," Sustainable Agriculture Research & Extension, vol. 28, no. 1, pp. 1-23, 2019.

[8] A. Kumar Singh, P. Kaur, and P. Singh, "Deep Learning for Plant Disease Detection: A Survey," Agriculture, vol. 12, no. 12, p. 1858, Dec. 2022, doi: 10.3390/agriculture12121858

[9] X. Li et al., "Automated Detection of Fungal Diseases in Wheat Using Deep Learning," Sensors (Basel), vol. 20, no. 11, p. 3306, Jun. 2020, doi: 10.3390/s20113306

[10] Y. Jha, P. Kavuri, P. N. Desai, and M. S. Sasikala, "Early Detection of Wheat Diseases Using Deep Learning and Image Segmentation Techniques," Multimedia Tools and Applications, vol. 79, no. 21-22, pp. 14299-14333, Nov. 2020, doi: 10.1007/s11042-020-09733-7

[11] J. Ma et al., "Wheat Yellow Rust Disease Detection from Hyperspectral Images Using Deep Convolutional Neural Network," IEEE Geoscience and Remote Sensing Letters, vol. 15, no. 1, pp. 211-215, Jan. 2018, doi: 10.1109/LGRS.2017.2790234

[12] S. Y. Lee, H. S. Kwon, and J. H. Lee, "Deep Learning-Based Image Recognition for Rice Disease Detection," Computers and Electronics in Agriculture, vol. 141, pp. 81-88, Dec. 2017, doi: 10.1016/j.compag.2017.08.011

[13] G. Wang et al., "A Novel Deep Learning Model for Rice Diseases and Insect Pest Classification," Agriculture, vol. 10, no. 1, p. 16, Dec. 2019, doi: 10.3390/agriculture10010016

[14] I. Nadal, M. Vilaplana, V. Beuces-Veja, J. Marques-Donate, and J. Simó, "Deep Learning vs. Support Vector Machines for Rice Leaf Disease Classification," in 2018 International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2018, doi: 10.1109/ISCAS.2018.8351713

[15] A. L. R. Souza, G. R. A. A. Silva, S. R. S. A. Junior, M. P. Albuquerque, and J. M. R. S. Nogueira, "Deep Learning for Coffee Leaf Disease Classification," in 2019 IX Brazilian Symposium on Computing Systems (SBCS), pp. 1-6, Nov. 2019, doi: 10.1109/SBCS.2019.00082

[16] A. S. Girão et al., "Automatic Coffee Leaf Disease Detection Using Deep Learning Techniques," in 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2018, doi: 10.1109/ISCAS.2018.8351699

[17] B. Ustaoglu, C. Ozkan, and S. Oncus, "Coffee Leaf Disease Detection and Classification Using Deep Convolutional Neural Network," Ecological Informatics, vol. 57, p. 101071, Feb. 2020, doi: 10.1016/j.ecoinf.2020.101071

[18] S. Palanisamy, A. Krishnamoorthy, and E. S. Kumar, "A Deep Learning Framework for Tea Leaf Disease Detection," Computers and Electronics in Agriculture, vol. 151, pp. 100-108, Feb. 2018, doi: 10.1016/j.compag

[19] Choudhury, G. "Farm Futro" A Machine Learning-Based Crop Recommendation System for Precision Agriculture". Retrieved from Kaggle: https://www.kaggle.com/datasets/gourab8889/farm-futro. (Accessed on 25/4/2024.)

[20] VIRANSHU PARUPARLA & DrYVyas , "Crop Disease Improved version Dataset" Kaggle dataset: https://www.kaggle.com/datasets/viranshuparuparla/dieseas (accessed on 11.04.2024)

[21] Ali, Sawkat; Ibrahim, Muhammad ; Ahmed, Sarder Iftekhar ; Nadim, Md. ; Mizanur, Mizanur Rahman; Shejunti, Maria Mehjabin ; Jabid, Taskeed (2022), "MangoLeafBD Dataset", Mendeley Data, V1, doi: 10.17632/hxsnvwty3r.1

[22] Jepkoech, jennifer; Kenduiywo, Benson; Mugo, David; Chebet, Edna (2021), "JMuBEN", Mendeley Data, V1, doi: 10.17632/t2r6rszp5c.1

[23] Jepkoech, Jennifer; Mugo, David; Kenduiywo, Benson; Chebet, Edna (2021), "JMuBEN2", Mendeley Data, V1, doi: 10.17632/tgv3zb82nd.1

[24]VIRANSHU PARUPARLA "Rice_disease_model" Kaggle dataset: https://www.kaggle.com/datasets/viranshuparuparla/rice-disease-model/data (accessed on 22.04.2024)

[25] Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. https://www.researchgate.net/publication/316184205_MobileNets_Efficient_Convolutional_Neural_Networks_for_Mobile_Vision_Applications .

[26] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations (ICLR 2015), 1–14

[27] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

[28] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

Bhaskaracharya National Institute for Space Applications and Geo-informatics
MeitY, Government of India
Phone: 079 - 23213081 Fax: 079 - 23213091
E-mail: info@bisag.gujarat.gov.in, website: https://bisag-n.gov.in/

ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

Report Verification Procedure

Date:

Project Name: Farmer support system using AI/ML: Crop recommendation, environmental favourability prediction and disease detection using deep learning.

Student Name & ID: 1. Viranshu Paruparla - 24RN4

2. Tirth Patel - 24RN4

	Soft Copy	Hard Copy
Report Format:	☐	☐
Project Index:	☐	☐

Sign by Training Coordinator **Sign by Project Guide**