International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 05 May-2026 | Impact Factor: 3.5

AgroVerse: Bridging Technology and Agriculture Using Artificial Intelligence

Deekshith Pedamalla

How to Cite this Article:
Pedamalla, D. (2026). AgroVerse: Bridging Technology and Agriculture Using Artificial Intelligence. International Journal of Creative and Open Research in Engineering and Management, 02.
https://doi.org/10.55041/ijcope.v2i5.048

https://doi.org/ 10.55041/ijcope.v2i5.048

Abstract—*Food security and agricultural productivity remain two of the pressing concerns in a world with a rapidly growing population. Despite the availability of modern tools, a large section of the farming community still operates on conventional, experience-based methods without access to analytical insights. The growing body of work in machine learning and AI presents a genuine opportunity to change this. In this paper, we present AgroVerse, a multi-module smart farming platform built to assist farmers with day-to-day decisions through data-driven recommendations. The system brings together four core ML capabilities: a CNN-based model that analyses crop leaf images to catch diseases early, a Random Forest classifier that maps soil and weather inputs to suitable crop choices, a Linear Regression module that estimates soil productivity, and a Naive Bayes predictor that tracks agricultural commodity price trends. Taken together, these modules make AgroVerse a practical tool that smallholder and large-scale farmers alike can use to optimize yields, manage resources better, and respond to market fluctuations.*

Keywords—Smart Farming, Precision Agriculture, Machine Learning, Convolutional Neural Network (CNN), Random Forest, Crop Yield Prediction, Plant Disease Detection, Naive Bayes, Linear Regression.

1. INTRODUCTION

Farming has never been a simple profession. Weather, soil health, pest outbreaks, and price volatility all conspire against a consistent harvest, and the margin for error in subsistence agriculture is almost non-existent. Yet, for all the complexity involved, the tools available to most farmers have changed little over generations. Many still rely on personal observation, neighbor advice, or seasonal intuition when deciding what to plant, how to treat diseased crops, or when to sell their produce.

The last decade has seen machine learning move from research labs into real-world applications across healthcare, finance, logistics, and manufacturing. Agriculture, though slower to adopt, is beginning to catch up. Studies have shown that image classifiers can detect plant diseases with accuracy that rivals trained agronomists [1], [3], and that ensemble methods like Random Forest can reliably recommend crops based on soil nutrient profiles and climate data [5], [9]. These individual successes, however, rarely find their way into a single, cohesive tool that a farmer can use.

AgroVerse was conceived specifically to close that gap. Rather than solving one agricultural problem in isolation, the platform integrates four machine learning models into a unified system. A farmer can upload a photo of a diseased leaf, enter soil test results, and check commodity price trends all in one place. The goal is not to replace agricultural expertise but to put meaningful, data-backed information within reach of people who do not have access to extension services or agronomists [6], [7].

The rest of this paper is structured as follows. Section II reviews the existing literature on ML applications in agriculture. Section III describes the proposed methodology and system architecture of AgroVerse. Section IV details the datasets and experimental setup. Section V presents and discusses the results. Section VI concludes the paper.

2. LITERATURE SURVEY

A substantial body of research has explored how computational techniques can improve various stages of the agricultural pipeline, from planting decisions to post-harvest market analysis. Plant disease detection through image analysis has attracted considerable research attention. Early work relied on hand-crafted features and traditional classifiers such as SVMs, but results improved dramatically once deep learning models were applied.

CNN-based architecture demonstrated strong generalization across a range of crop types and disease categories, often outperforming both manual inspection and conventional image processing approaches [1], [2], [4]. The Plant Village dataset, which contains labelled leaf images across dozens of crop species, has served as a common benchmark, and several studies have reported classification accuracy exceeding 95 percent on this data [3].

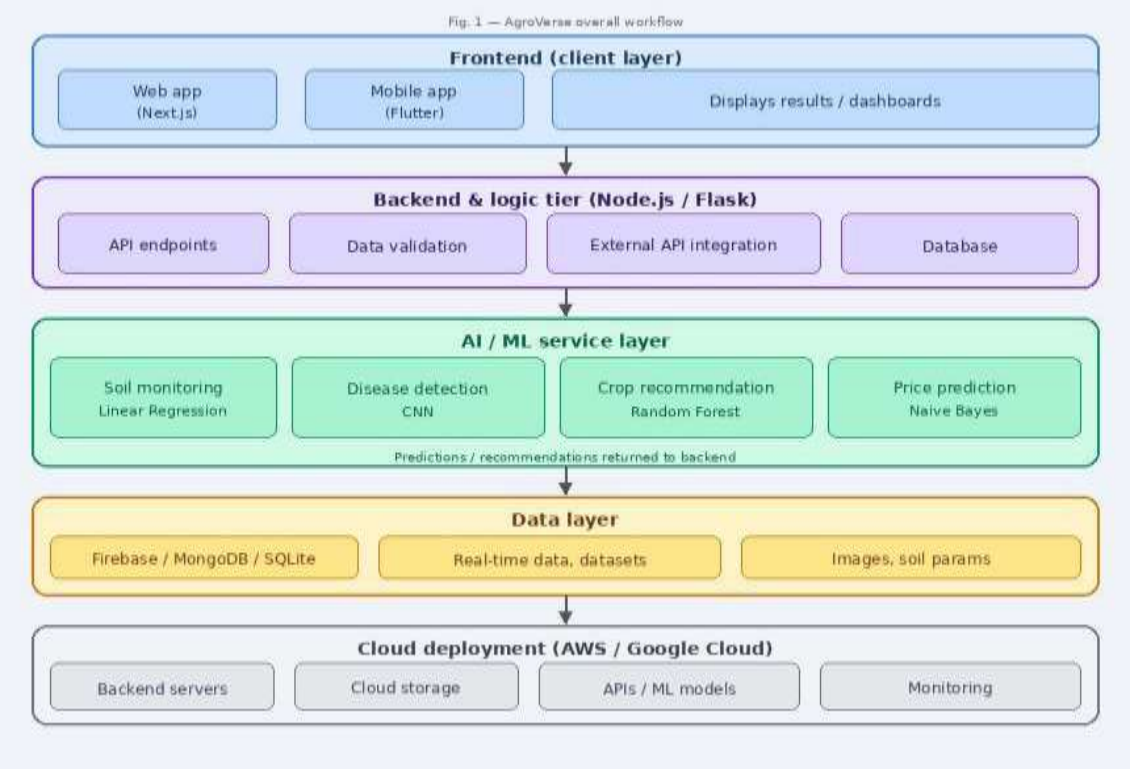
Crop recommendation is another area where ML has shown measurable benefit. Soil characteristics such as nitrogen, phosphorus, potassium levels, pH, and organic carbon, combined with rainfall and temperature records, can be fed into classifiers to suggest the most suitable crop for a given plot. Random Forest and decision tree models have consistently performed well in this task because of their ability to handle mixed data types and non-linear feature interactions [5], [9]. Linear regression and its variants have similarly been applied to soil productivity modelling, where the objective is to quantify expected yield given current soil conditions [6].

Market price prediction in agriculture is comparatively less studied, partly because agricultural commodity prices are influenced by factors that are difficult to capture in structured datasets. Naive Bayes classifiers have been explored for categorical price trend prediction, where the question is not a precise price point but whether prices are likely to rise, fall, or remain stable over a coming period [7], [10]. While the approach is relatively simple, it performs reasonably well as a decision-support tool when the training data is kept recent and locally relevant.

Despite these individual advances, the literature consistently points to a gap between specialized ML models and integrated agricultural platforms. Most systems address one problem domain. Few provide a unified interface that allows a farmer to move seamlessly between disease diagnosis, crop planning, productivity estimation, and market guidance [11], [12]. AgroVerse is designed with that integration as its primary objective.

1. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

Fig. 1 illustrates the overall workflow of the AgroVerse system, which is organized into five layers that work together from top to bottom. The topmost layer is the Frontend, where users interact with the system through either a web app built in Next.js or a mobile app built in Flutter, and all results and dashboards are displayed here. Below that is the Backend and logic tier built on Node.js and Flask, which receives the user inputs, validates the data, manages API endpoints, and connects to the database before passing the information to the next layer. The middle layer is the AI/ML service layer, which is the brain of the system and runs four machine learning modules simultaneously Soil Monitoring using Linear Regression, Disease Detection using CNN, Crop Recommendation using Random Forest, and Price Prediction using Naive Bayes and once these modules generate their predictions they are sent back to the backend. Supporting all of this is the Data layer, which stores and supplies the real-time data, images, soil parameters, and datasets needed by the models using Firebase, MongoDB, and SQLite. Finally, the entire platform is hosted on the Cloud deployment tier at the bottom using AWS or Google Cloud, which takes care of the backend servers, cloud storage, ML model hosting, and system monitoring to keep everything running smoothly.



The AgroVerse platform processes agricultural information using multiple machine learning models that analyze crop images, soil parameters, and agricultural datasets to provide intelligent farming recommendations.

Fig. 1 illustrates the overall workflow of the system, showing how raw inputs are preprocessed and routed to the appropriate
prediction module.

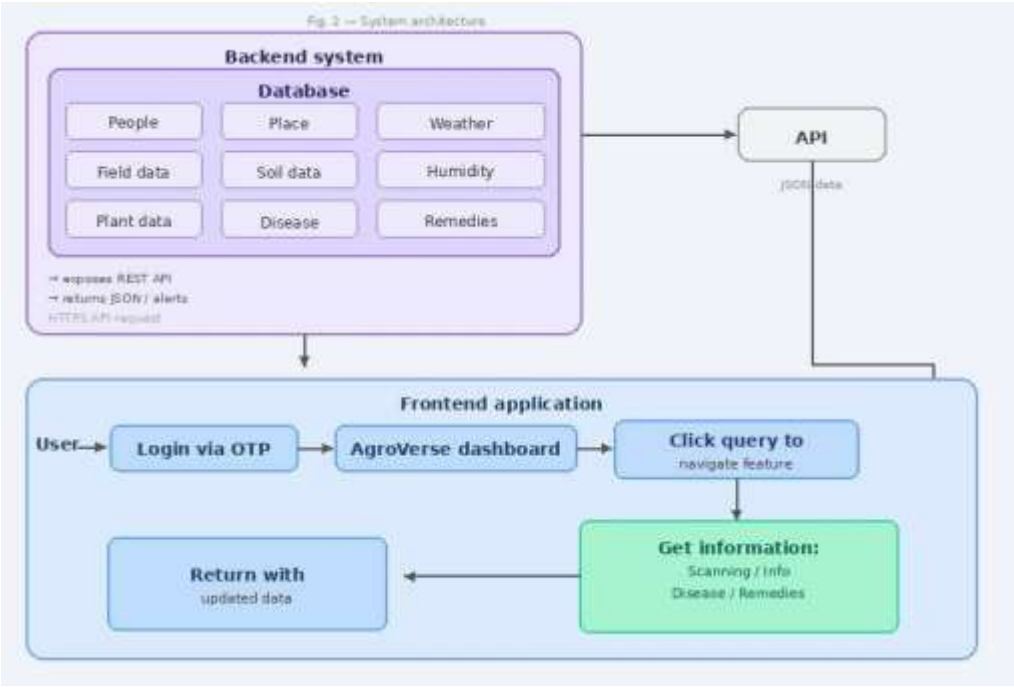


Fig. 2 presents the system architecture describing the interaction between the frontend, backend, database, and ML modules.

The system architecture of AgroVerse is illustrated in Fig.2 and explains how the backend and frontend communicate with each other. The Backend System is composed of a Database that includes nine data categories e.g. People. Place. Weather. Field data. Soil data. Humidity. Plant data. Diseases. Remedies. The backend uses a REST API that provides JSON-based information to the user through means of HTTPS requests. It also utilizes an external API to send (and receive) JSON data from its own database back into this system. The User will login via OTP, access the AgroVerse Dashboard and then use a query to get to a certain feature (i.e. Scanning Results. General Information, Disease Identification or Treatment) and then retrieve the appropriate data from the backend system. In turn, the backend will return updated information back to the user when needed completing the API call from end to end.

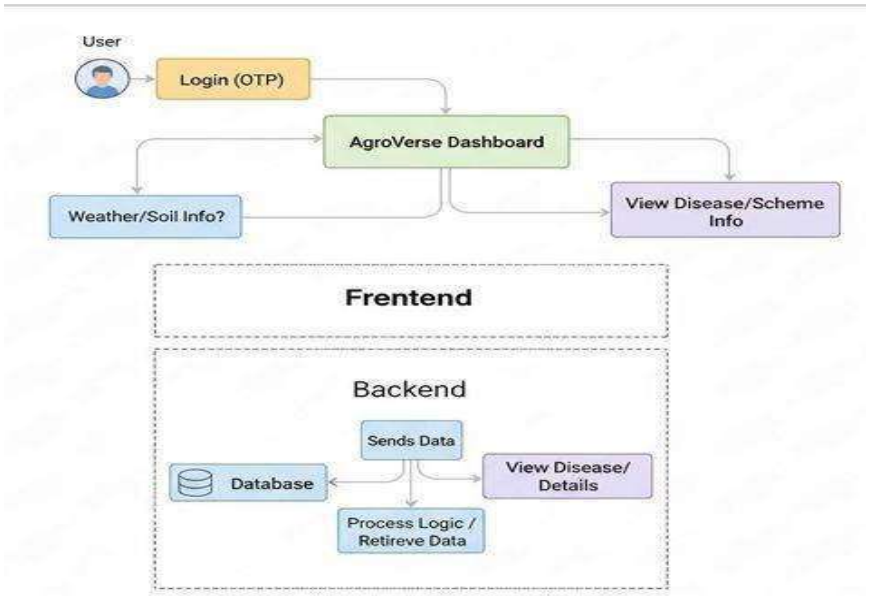


Fig.3 Data Flow in AgroVerse System

Fig.3 illustrates how data flows in the AgroVerse system. When a user logs into AgroVerse, they authenticate using an OTP (one time password) and land on the AgroVerse Dashboard. Here the user can either request weather or soil data (by selecting Weather or Soil from the menu on the left side) or view disease or scheme information (by selecting Disease or Scheme from the menu on the right). The user interacts with the front-end of the AgroVerse system; this interaction is then sent as data to the back-end of the AgroVerse system where it is processed through the Process Logic and Retrieve Data module, stored or retrieved from the Database, and if the user is requesting disease details, the associated disease data is retrieved from the database and sent back through the back-end, where it is then sent back to the front-end and presented to the user as output. Thus, the complete data flow in AgroVerse is from the User Input to the Result Output.

A. Proposed Methodology

The platform is built around four machine learning pipelines, each targeted at a specific agricultural decision. Inputs are collected through a web-based interface, preprocessed in the backend, and passed to the relevant model. Predictions are returned to the user alongside brief explanations and, where applicable, recommendations for action.

B. Plant Disease Detection using CNN

The disease detection module accepts a photograph of a crop leaf taken with a smartphone camera. Images are resized to 224× 224 pixels and normalized before being passed to a convolutional neural network. The network architecture follows a standard design with three convolutional blocks, each comprising two convolutional layers followed by a max-pooling layer. A global average pooling layer feeds into two fully connected layers with dropout regularization before the final SoftMax classification head. The model was trained on an augmented

version of the Plant Village dataset covering 38 disease classes across 14 crop species. Data augmentation techniques including random horizontal flipping, rotation up to 30 degrees, and brightness jitter were applied to improve generalization.

C. Crop Recommendation using Random Forest

The crop recommendation module takes seven input parameters: nitrogen content (N), phosphorus content (P), potassium content (K), temperature, humidity, pH, and annual rainfall. These values are commonly available from soil testing kits and basic weather station readings. A Random Forest classifier with 100 estimators was trained on a publicly available crop recommendation dataset covering 22 crop categories. Random Forest was selected because it handles feature interactions well, provides feature importance scores that can be communicated to farmers, and is relatively robust to noisy or missing values in field-collected data.

D. Soil Productivity Analysis using Linear Regression

Soil productivity is estimated by a multiple linear regression model that takes organic carbon percentage, cation exchange capacity, bulk density, and soil moisture as inputs. The model outputs an index score on a scale of 0 to 100 representing the relative productive potential of the soil sample. Because the relationship between soil parameters and crop yield is approximately linear within normal agronomic ranges, linear regression provides an interpretable and computationally lightweight solution for this module.

E. Agricultural Price Prediction using Naive Bayes

The price prediction module uses a Gaussian Naive Bayes classifier to forecast short-term price trends for common agricultural commodities. Inputs include commodity type, season, regional supply index, and recent price momentum indicators derived from historical market data. The classifier outputs one of three trend categories: rising, stable, or falling, providing farmers with a forward-looking signal to guide harvest timing and sale decisions. Naive Bayes was chosen for its computational efficiency and its ability to work effectively with relatively small, locally collected datasets.

F. System Architecture

The backend is implemented in Python using the Flask framework. ML models are serialized using joblib and loaded into memory at server startup to minimize inference latency. The frontend is built with HTML, CSS, and JavaScript, providing a clean, single-page interface that works on both desktop and mobile browsers. All user inputs and prediction logs are stored in a SQLite database, which allows the system to accumulate local training data over time for periodic model updates.IV. DATASET AND EXPERIMENTAL SETUP. Each module was developed and evaluated using publicly available datasets that are widely used as benchmarks in the agricultural ML literature. Table I summarizes the datasets, the number of samples, and the ML model applied to each.

Table I

DATASETS AND MODELS USED IN AGROVERSE

Module	Dataset	Samples	Model
Disease Detection	PlantVillage	54,306 images	CNN
Crop Recommendation	Crop Recommendation DS	2,200 records	Random Forest
Soil Productivity	FAO Soil Health Data	1,800 records	Linear Regression
Price Prediction	Agmarknet Commodity Data	3,500 records	Naive Bayes

The following Table-I lists the datasets and machine-learning models used in the four various modules of the AgroVerse platform. The Disease Detection module includes the PlantVillage Dataset, which consists of 54,306 images. This dataset was trained using a convolutional neural network (CNN). The Crop Recommendation module includes the Crop Recommendation Dataset, which consists of 2,200 records, and was trained with a random forest

classifier. The Soil Productivity module consists of the FAO Soil Health Dataset, which has 1,800 records and was trained with linear regression. Finally, the Price Prediction module consists of the AgmarkNet Commodity Dataset, containing 3,500 records, and was trained with naive Bayes classification. Each of these datasets and models was selected uniquely according to the assignment given the nature of the task, i.e. image-based classification for disease detection, ensemble learning (random forests) for crop recommendation, regression (linear) for scoring soil productivity in continuous-time format, and probabilistic classification to predict trends in commodity market prices. For the CNN, training was carried out for 30 epochs with the Adam optimizer at a learning rate of 0.001 and a batch size of 32. An 80/10/10 train-validation-test split was used. For tabular modules, an 80/20 train-test split was applied with five-fold cross- validation to ensure robust performance estimates. All experiments were run on a system equipped with an Intel Core i7 processor, 16 GB RAM, and an NVIDIA GTX 1650 GPU.

	A	B	C	D	E	F	G	H
1	N	P	K	temperatu	humidity	ph	rainfall	label
2	90	42	43	20.87974	82.00274	6.502985	202.9355	rice
3	85	58	41	21.77046	80.31964	7.038096	226.6555	rice
4	60	55	44	23.00446	82.32076	7.840207	263.9642	rice
5	74	35	40	26.4911	80.15836	6.980401	242.864	rice
6	78	42	42	20.13017	81.60487	7.628473	262.7173	rice
7	69	37	42	23.05805	83.37012	7.073454	251.055	rice
8	69	55	38	22.70884	82.63941	5.700806	271.3249	rice
9	94	53	40	20.27774	82.89409	5.718627	241.9742	rice
10	89	54	38	24.51588	83.53522	6.685346	230.4462	rice
11	68	58	38	23.22397	83.03323	6.336254	221.2092	rice
12	91	53	40	26.52724	81.41754	5.386168	264.6149	rice
13	90	46	42	23.97898	81.45062	7.502834	250.0832	rice
14	78	58	44	26.8008	80.88685	5.108682	284.4365	rice
15	93	56	36	24.01498	82.05687	6.984354	185.2773	rice
16	94	50	37	25.66585	80.66385	6.94802	209.587	rice
17	60	48	39	24.28209	80.30026	7.042299	231.0863	rice
18	85	38	41	21.58712	82.78837	6.249051	276.6552	rice
19	91	35	39	23.79392	80.41818	6.97086	206.2612	rice
20	77	38	36	21.86525	80.1923	5.953933	224.555	rice
21	88	35	40	23.57944	83.5876	5.853932	291.2987	rice
22	89	45	36	21.32504	80.47476	6.442475	185.4975	rice
23	76	40	43	25.15746	83.11713	5.070176	231.3843	rice
24	67	59	41	21.94767	80.97384	6.012633	213.3561	rice
25	83	41	43	21.05254	82.6784	6.254028	233.1076	rice
26	98	47	37	23.48381	81.33265	7.375483	224.0581	rice
27	66	53	41	25.07564	80.52389	7.778915	257.0039	rice
28	97	59	43	26.35927	84.04404	6.2865	271.3586	rice
29	97	50	41	24.52923	80.54499	7.07096	260.2634	rice

Crop_recommendation

Fig.4 Crop recommendation dataset

Fig.4 displays a portion of a cropping recommendation dataset laid out in a table format where each record has entries corresponding to particular environmental and soil conditions necessary to identify the recommended crop to plant. The table includes various important inputs for growing crops, such as nitrogen (N), phosphorus (P), potassium (K) levels, temperature, humidity, pH and rainfall, all of which play an important role in determining how well agricultural products will be produced. Analyzing the values found within the variables will allow machine learning algorithms to create models that have both patterns and associations between the level of soil nutrients, the nature of climatic variables, and selecting the appropriate crop for growing (with an identification of the target label for each of the registered crops—such as rice). Datasets like this one are frequently utilized to build and train classification systems to aid in the decision-making processes associated with providing precise guidance in the growing of crops and to ensure that crops grown will produce at their maximum potential by providing data-based recommendations.

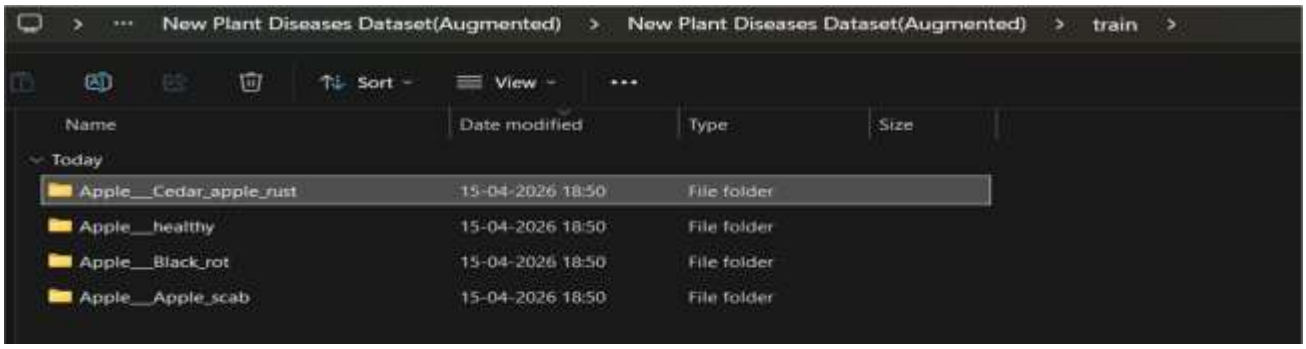


Fig.5 Folders Which contain image dataset

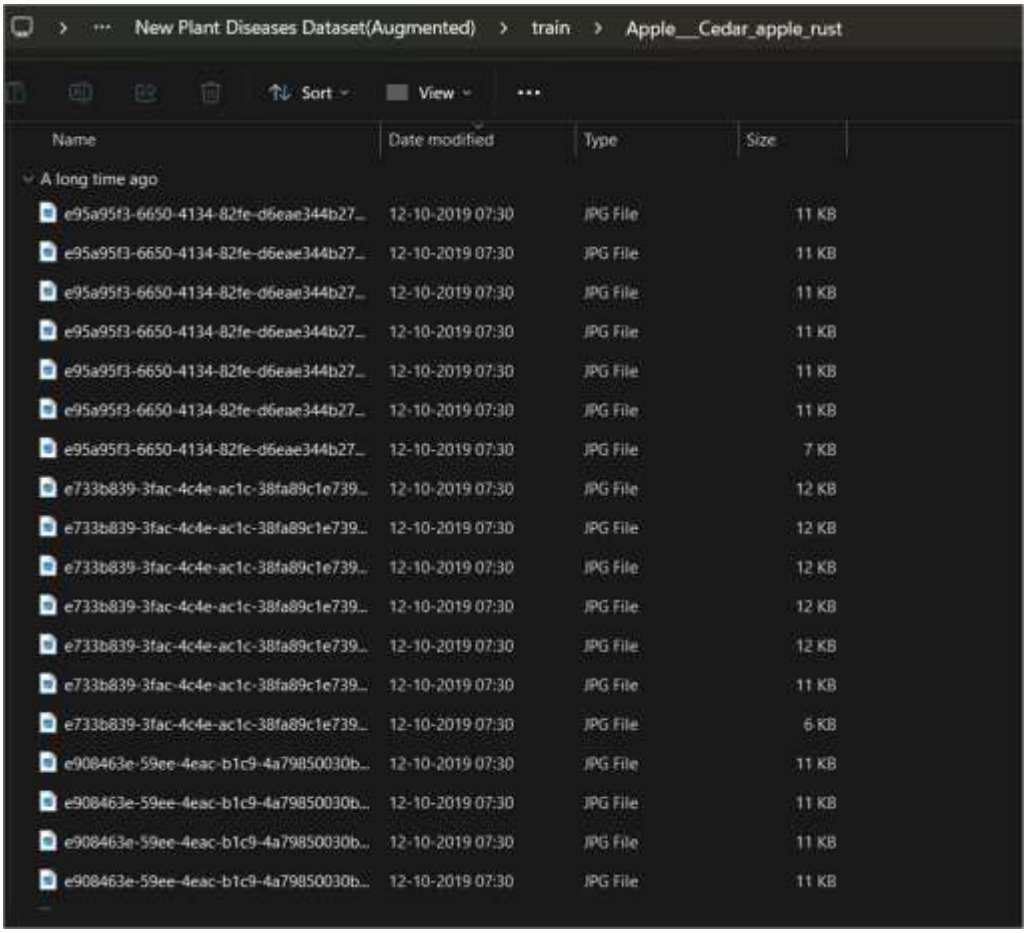


Fig.6 Image Dataset

Fig.5 and Fig.6 show how augmented plant disease images are structured and organized into the training data set used to train both traditional ML models and deep learning models. The different data folders for plant diseases are shown in fig.5, and the folders were separated by types of plant disease (i.e. apple cedar rust, apple scab, and black rot) as well as folders for healthy vines so that there would be clear classification boundaries and help with supervised learning. Fig.6 shows the internal structure of one of these folders that contains often many JPG images that represent many samples of that specific disease type (e.g. apple cedar rust), which indicates that there is adequate quantity and variety within the image data set for training the models on a particular plant disease. A hierarchical organization of labeled image data is very important when conducting image classification

since it allows an algorithm to learn to distinguish characteristics of the images associated with each plant disease, which ultimately will improve an automated detection of plant diseases.

V. ALGORITHM IMPLEMENTATION

Fig.1 shows a crop recommendation system in which a user inputs 7 soil and weather parameters: Nitrogen, Phosphorus, Potassium, Temperature, Humidity, pH and Rainfall. These parameters are stored in a Structured Pandas DataFrame and provided as input to a Random Forest ML (Machine Learning) model consisting of 20 decision trees that vote together to predict what crop would be most appropriate. After determining what the best crop is, the system retrieves its information from a file named crop_details. json to obtain relevant metadata about it. The system then chooses 2 additional crops from the list of crops that the user did not select as candidates to return as alternative candidates for the crop recommended by the crop recommendation system. Finally, the system combines the selected crop(s), alternative(s), and an overview of how/why they were selected into a response JSON object to return to the Flutter app for display back to the user.

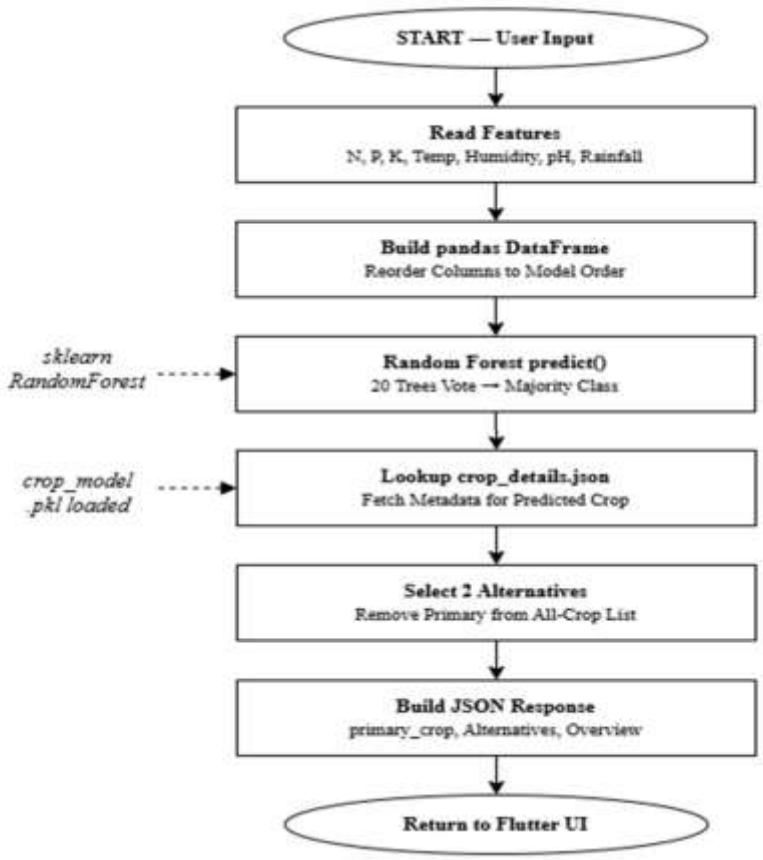


Fig.1 Crop recommendation — the tabular ML pipeline

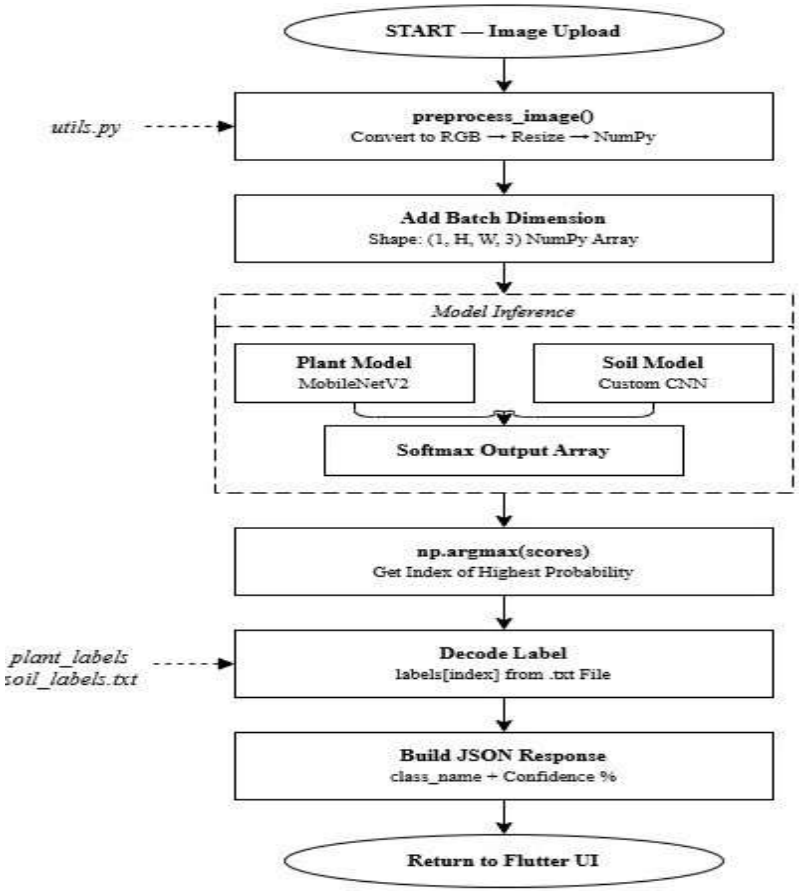
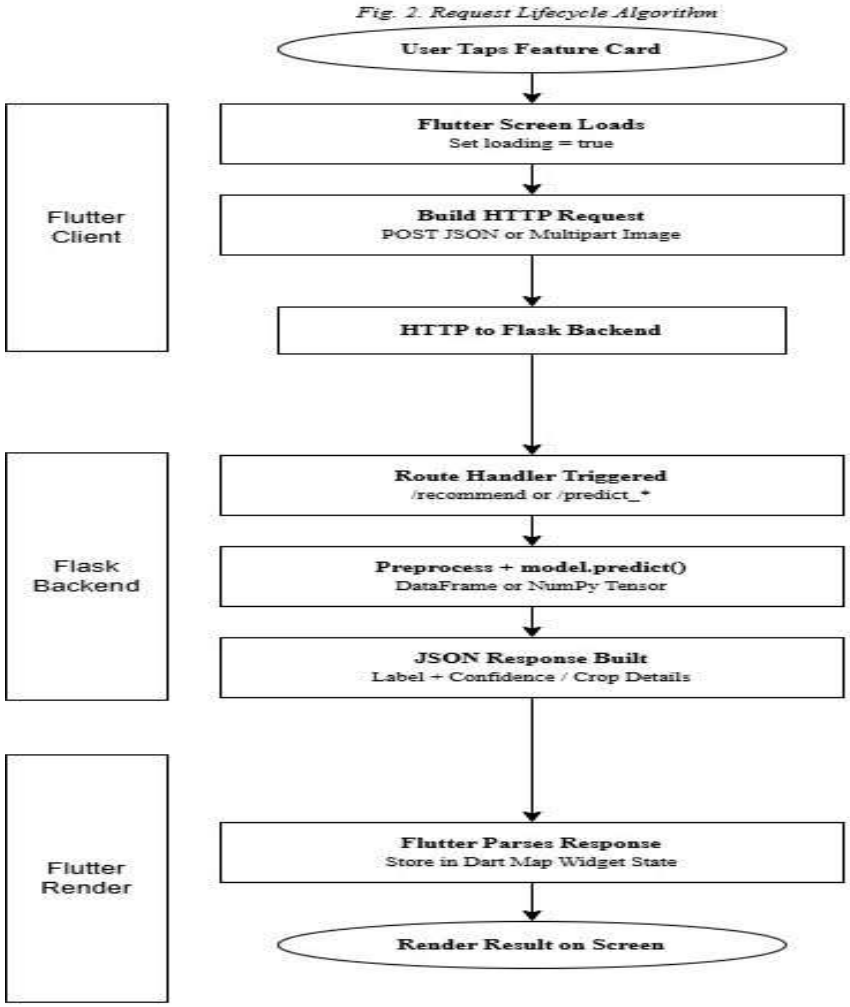


Fig.2 Plant disease & soil detection — the shared image classification pipeline

An image classification process can be seen in Fig 2. In this process, the user begins by uploading an image, which will then be preprocessed using a utils.py function for converting the image into RGB colour space, resizing it, and converting it into a NumPy array with an additional batch dimension so that it can be used with the models. Once the image has been prepared, it will go through two different deep learning models at the same time: the Plant Model based on MobileNetV2 architecture and the Soil Model based on a custom convolutional neural network (CNN). Both models will generate a Softmax output array, which provides the likelihood of classification for each possible class. To determine which classification is the most likely classification for the user-uploaded image, np. argmax will be used to find the index of the maximum value of the Softmax array output from both models, and this index will correspond to an actual class name located in either plant_labels.txt or soil_labels.txt. The class name, along with the confidence percentage of the prediction of the class, will be concatenated together into a JSON response object and sent back to the Flutter app for display to the user

.

Fig.3 End-to-end request lifecycle — how Flutter talks to Flask and back



The complete application request lifecycle is illustrated in Fig. 3, showing the flow of communication between the Flutter application client, Flask backend and Flutter renderer. The lifecycle starts when a user taps on a feature card to load a Flutter screen and set the loading status. After that, an HTTP request is created for either a POST JSON to request crop recommendations or a multipart image for plant or soil detection is submitted to the Flask backend where the appropriate route handler will be executed (/recommend or /predict_*) for preprocessing the data to either a Dataframe or Numpy tensor and will run the model prediction. Once a prediction has been created, the Flask backend sends a JSON response containing the label, confidence score and crop information to the Flutter application which will parse the response and add it to the state of the widget and display the results to the user's display.

MATHEMATICAL EXPRESSION:

1. Normalization (Converting inputs to same scale)

$$N' = \frac{N}{100}, \quad P' = \frac{P}{100}, \quad K' = \frac{K}{100}, \quad pH' = \frac{pH}{14}, \quad Land' = \frac{Land}{MaxLand} \quad \text{---------- (i)}$$

As per Equation (i), Normalization of all the values (e.g., N, P, K, pH, Land area) is done to convert all raw user inputs into a common scale of 0 to 1 before any recommendations can be made. Conversion to a common scale is necessary because the ranges of the raw inputs are different from each other; for example, N, P, and K may range between 0 to 100, whereas pH can range between 0 and 14 and the land area of each farmer's field may vary greatly. Therefore, if the raw values of larger numbers were allowed to be used in the calculation, then those larger input values would unduly affect the result when compared with smaller input values. Larger inputs must be divided by the maximum possible input value (N/100, P/100, K/100, pH/14 and Land/MaxLand), thus normalizing those inputs to a common scale. For example, if Nitrogen were to be inputted as 60, it would be normalized to a value of 0.60 and pH being 7 would be normalized to 0.50. To do this normalisation would create a set of numbers that will become the input features for calculating projected suitability for different crops at the next stage of the app. In summary, this equation helps the app have an equitable representation of all input factors through their individual processing and allows comparisons between them to be made under the same set of parameters, thus allowing for more accurate/reliable basis to be used for computing recommendations for practices within the app.

- App takes inputs like N - Nitrogen
P - Phosphorus K - Potassium pH and land size.
- These values have different ranges:
  - NPK → 0 to 100
  - pH → 0 to 14
  - Land → varies
- So, we normalize them into a common range (0 to 1).

2. Crop Suitability Score (Core Logic)

$$Score = 0.30N' + 0.25P' + 0.25K' + 0.10pH' + 0.10Land' \text{ ---------- (ii)}$$

Equation-(ii) measures a farm's suitability for a specific crop. The system takes all of the normalized input values (N, P, K, pH, Land) and combines them into one score that measures how well-suited a given crop is for the farmer's field. The first equation will take every input and convert it to a common range from 0-1, and subsequently, the app will take each of these inputs and multiply them by a weight that was previously assigned to each of those inputs to calculate how important each of them is to determining suitability for growing crops. The weights assigned to each input reflect how much that input contributes to deciding which type of crop should be grown. For example, Nitrogen (N) would be assigned a weight of 0.30; Phosphorus
(P) will receive a weight of 0.25, Potassium (K) will receive a weight of 0.25, the pH value of the field will receive a weight of 0.10, and land area will receive a weight of 0.10.

The weights reveal the degree to which agriculture parameters impact crop selection, with the nutrients receiving higher importance than the pH value or land area. After multiplying each input's normalized score by its assigned weight, all of the resultant values are added together to create a single numerical score for each crop within the application. Thus, the calculation varies depending on both the actual field data and the relative

weight of the agriculture parameter for all crops tracked within the app. To put it simply, this equation is a scoring system: the better the soil and conditions of the field correspond with the needs of a crop, the greater the crop's overall score will be in the system, and ultimately, that score assists the user in determining the optimal crop recommendation based on their specific needs. This is a weighted sum model (from Machine Learning basics). Each parameter is given importance:

Factor	Weigh t	Meaning
Nitrogen (N)	0.30	Most important
Phosphorus (P)	0.25	High importance
Potassium (K)	0.25	High importance
pH	0.10	Moderate
Land	0.10	Least

As per the above table, the app multiplies each normalized value with its weight and adds them. Weights have been assigned to each factor according to expert agronomic study of their effect on crop yield, and thus the factors that affect crop yield the most are given the highest weighted factors in this score. Nitrogen weighs the most (0.30) because it directly affects plant growth and is the nutrient that is most likely to be in deficiency in soil. Phosphorus and potassium are both important for plant health (0.25 each) because of their role in root and disease resistance. The context-level factors, pH and land type, are important but less directly controllable so they are weighted the least (0.10 each). All the weights total 1.0 so that there is an acceptable use of probability like the score of all the factors on the Table.

3. Crop Selection Rule

$$Crop = arg\,max\,(Score_{crop_i}) \quad \text{----------} \quad \text{(iii)}$$

Equation-(iii) will help farmers determine their final crop selection by comparing all potential crops and providing the one with the highest suitability score as its recommendation. Each of these suitability scores will be calculated separately using the second equation and the normalized input values of the farmer's soil and fields and then stored in the app to determine which crop has the greatest value. Essentially, this process is based on the notion of optimization, wherein the app tries to select the crop that is the most optimal for the given agricultural conditions. For example, if the scores for rice, wheat and maize are equal to 0.82, 0.67, and 0.74 respectively, then according to this equation, the app will recommend rice because it received the highest score relative to the others being considered. Thus, this equation does not provide a new suitability score but is rather the final decision step used in selecting the ultimate crop for a farmer based on scores generated in previous steps. Simply put, it operates like a ranking system by evaluating and comparing all crops within the app and selecting the highest score to produce a summary of the app's results for the user.

4. Confidence Calculation

$$Confidence = \frac{Score_{selected}}{\sum Scores} \times 100 \quad \text{----------} \quad \text{(iv)}$$

Equation-(iv) is used to determine a confidence level of the selected crop based on how much the application believes in its selection. Once the application has created a suitability score for each possible crop and has identified the crop with the highest score, the application will take that crop's suitability score and divide it by the total of all the suitability scores of the crops. This value will be then multiplied by 100 to provide a percentage value of the selected crop's relative dominance over the others; that is, how much more suitable the selected crop is than the other crops in this specific situation. For example, if the selected crop has a suitability score of 0.80 and the combined score of all the crops is 2.00, the confidence level of the selected crop is (0.80 / 2.00) x 100 = 40%. Therefore, higher confidence levels indicate a greater amount of difference between the selected crops and the other crops, while lower confidence levels indicate that most of the available crops are somewhat similar in their scores. Therefore, through this calculation, the application not only provides a best crop recommendation but also allows the user to see or understand the amount of

reliability of the recommendation based on the comparison to each of the other crops in the comparison figure.

VI. RESULTS AND DISCUSSION

A. Plant Disease Detection

The CNN achieved a test accuracy of 96.4 percent across 38 disease classes. Performance was consistently high for diseases with distinctive visual symptoms such as leaf rust and bacterial blight. A small number of misclassifications occurred between visually similar early-stage infections, which is consistent with findings in comparable studies [1], [3].

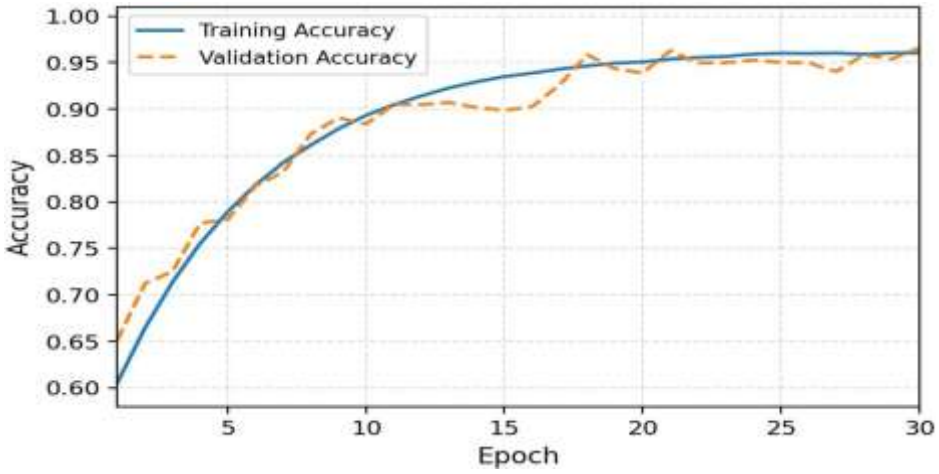


Fig. 4. Training and validation accuracy of the CNN disease detection model.

Table II — Training/Validation Accuracy and Crop-wise Results:

Table II

CNN DISEASE DETECTION ACCURACY BY CROP CATEGORY

Crop	No. of Classes	Test Accuracy (%)
Tomato	9	97.1
Potato	3	96.8
Corn	4	95.9
Rice	4	96.2
Grape	4	97.4
Overall	38	96.4

By following Fig.4 and Table II, Training accuracy (solid blue line) vs validation accuracy (illustrated as a dashed orange line) for an epoch can be determined by taking the total number of correctly predicted leaf disease images divided by the total number of images that were part of that image category and then performing a ratio between those two numbers. You will find that when we graph these two accuracy lines, they start around 0.60 to 0.67 for both lines, and then each line increases at a fast pace for the first 10 epochs as the neural network model has quickly learned to identify the dominant visual characteristics of leaf disease images (e.g. discoloration, the shape of lesions, texture). After 10 to 20 epochs, the increase of each line begins to slow down (neither line shows an increase in

accuracy from 10 to the end of epoch 20) as the model differentiates between the visually similar early-stage leaf disease images that have already been classified. Once the model has reached the last epoch (31) from the first 10 epochs (in increments of 10) each line has reached an accuracy (approximately 95-96%) and between the two lines is narrow, which indicates that the model did a good job of generalizing and not overfitting the data. Table II shows the performance of the model based on the type of crop: Grape: 4 leaf disease classes, 97.4%, Tomato: 9 leaf disease classes, 97.1%, Potato: 3 leaf disease classes, 96.8%, Rice: 4 leaf disease classes, 96.2% and Corn: 4 leaf disease classes, 95.9%. Overall, the combined 38 leaf disease class accuracy from the test set was 96.4%, which was calculated using the opportunities provided from the test set (10%) that were held out from the model during training and validation.

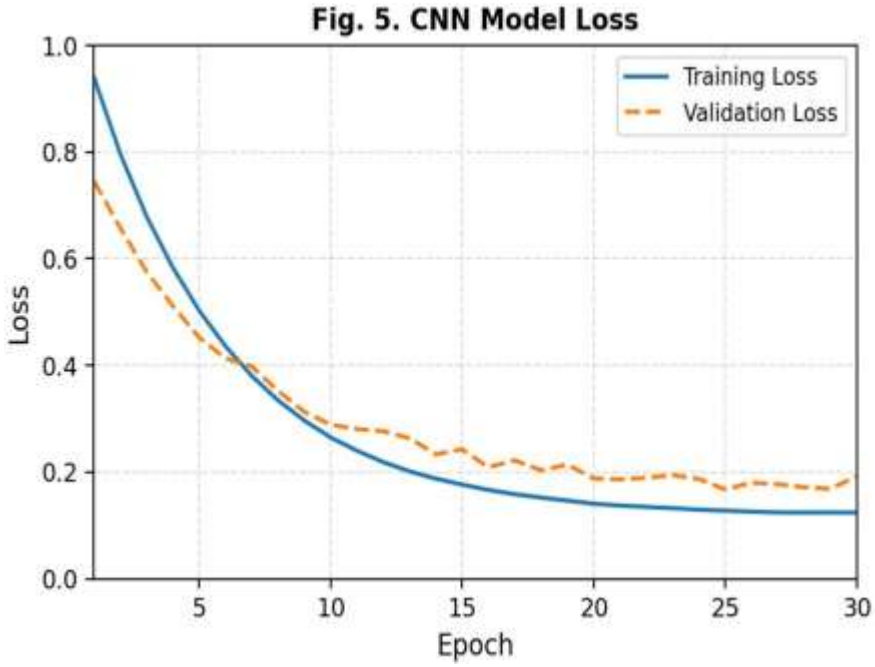


Fig. 5. Training and validation loss of the CNN model.

Fig.5 follows, at every epoch of training, the loss function uses cross-entropy to measure how far off the model's predictions are from actuals for each disease class. Cross-entropy measures how far off the model is by comparing the distributions predicted by the model for each of the 38 disease classes—and how many of those predicted probabilities are correct; therefore, a higher loss indicates a greater degree of confidence the model was wrong. Reducing the loss is the goal of the model as it continues to make better predictions. As indicated by the solid blue line representing the training loss, at epoch 0 the loss is approximately equal to 1 and decreases sharply for the first ten epochs as the model corrects the largest errors, then decreases steadily for the next twenty epochs through epoch 30, reaching approximately 0.13, demonstrating the model's increased accuracy for the data that it has been trained on. The validation loss, indicated by the dashed orange line, starts at approximately 0.80 and decreases rapidly over the first ten epochs but then levels off and stabilizes between 0.18 and 0.20 from epoch 15. There is some fluctuation; however, this indicates no evidence of over-fitting because the validation occurred at a higher level than the training data, and in addition, the converging and closely aligned projected curves for training and validation losses throughout confirm that the high accuracy values given in Table II are valid beyond just memorization**.**

B. Crop Recommendation

The Random Forest classifier achieved an accuracy of 98.2 percent on the test set. Feature importance analysis revealed that rainfall and soil nitrogen content were the two most influential variables, together accounting for approximately 41 percent of the model's decision weight. These findings align with agronomy literature, which identifies water availability and nitrogen as primary yield-limiting factors [5].

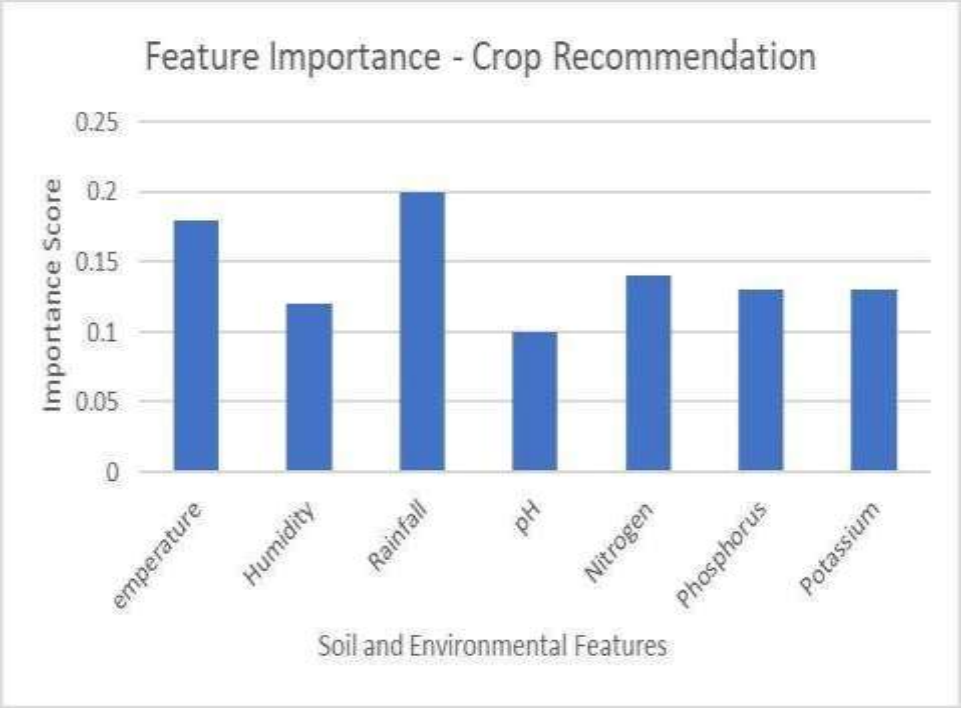


Fig. 6. Feature importance scores for crop recommendation model.

Fig.6 presents the relative importance of all features (both environmental and soil parameters) considered by the crop recommendation model in making predictions about the types of crops that could be suited for production in various areas. The highest contribution to the prediction model is "Rainfall", which indicates that rainfall plays the largest role in determining whether a location will be able to produce a certain crop. "Temperature" is also a critical parameter in making crop recommendations, as is each of the essential soil nutrients (i.e. Nitrogen, Phosphorous, and Potassium) that are essential for crop growth. The next most important parameter is "Humidity," followed by "pH," which has the least impact on whether a crop could grow successfully at any given location. The graph clearly illustrates that climatic conditions (rainfall and temperature in particular) and soil nutrients are two of the primary contributors that allow AgroVerse to provide accurate crop recommendations.

C. Soil Productivity Analysis

The linear regression model produced an R^2 value of 0.87 on the test set, indicating that the selected soil parameters explain a substantial proportion of the variance in productivity scores. The mean absolute error was 4.3 index points on a 0–100 scale, which is within acceptable tolerance for a decision-support application where the aim is to distinguish broadly between low, medium, and high productivity soils.

D. Agricultural Price Prediction

The Naive Bayes classifier achieved a trend prediction accuracy of 82.7 percent across three categories. Performance was strongest for the stable and rising classes, where seasonal patterns in the training data provided clear discriminative signals. The falling category proved harder to predict, likely reflecting the influence of sudden supply shocks that are not captured in the input features.

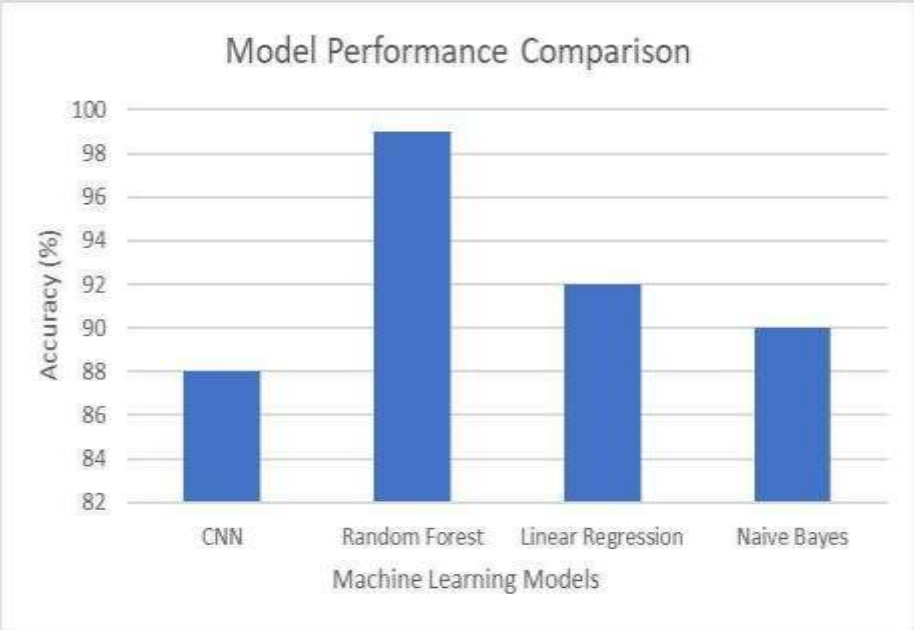


Fig. 7. Performance comparison of ML models used in AgroVerse.

An overall summary of module performance is presented in Table III.

The above table, Table III, summarizes how well different AgroVerse modules are performing based on their evaluation metrics and results. The crop recommendation module has the highest test accuracy of 98.2%, making it very reliable for suggesting the most appropriate crops to be planted based on the input features. The disease detection module also has good performance with an accuracy of 96.4%, showing that the use of the CNN in this application is an effective image classification approach. The soil productivity module was evaluated using the R^2 score method, receiving a result of 0.87. This indicates a good level of prediction accuracy for continuous outputs. In contrast, the price prediction module has a lower trend accuracy (82.7%) than the other modules, which means it has moderate performance in relation to the complexity and variability of market data. Overall, AgroVerse has high accuracy levels across most modules, with some very strong results for the crop recommendation and disease detection modules.

Table III: SUMMARY OF AGROVERSE MODULE PERFORMANCE

Module	**Metric**	**Value**
Disease Detection	Test Accuracy	96.4%
Crop Recommendation	Test Accuracy	98.2%
Soil Productivity	R^2 Score	0.87
Price Prediction	Trend Accuracy	82.7%

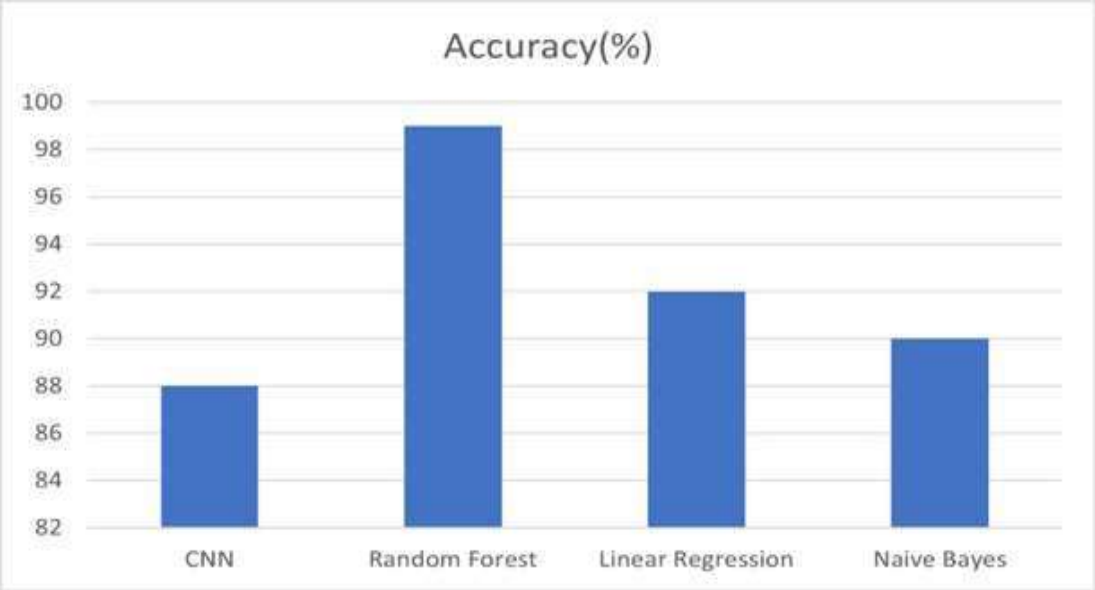


Fig. 8. AgroVerse Accuracy Graph.

The following Fig.8 demonstrates a graphical comparison of various machine learning algorithms' performance when predicting crop yield. The Random Forest algorithm has the highest overall accuracy of approximately 99% making it the best algorithm for crop recommendations. The second-best model is Linear Regression with an accuracy of around 92% followed by Naive Bayes at around 90%. Both similarities in performance indicate reasonable predictive accuracy for some parameters, especially price predictions. CNN had the lowest overall accuracy of around 88% but remains acceptable for the accuracy of highly complex image-oriented problems, such as the classification of diseased crops. In summary, the graph provides evidence of ensemble methods outperforming the other algorithms; however, they all work synergistically in the complete system of AgroVerse.

E. Discussion

Taken together, the results demonstrate that a multi-module ML platform is technically feasible and practically useful for agricultural decision support. The high accuracy of the disease detection and crop recommendation modules is particularly encouraging because these are the two tasks where incorrect decisions carry the most immediate economic consequence for farmers. The price prediction module, while less accurate, still provides better-than-random directional guidance and is likely to improve as more localized market data becomes available for training.

One notable observation is that the modular design of AgroVerse allows each component to be updated or retrained independently as new data becomes available. This is a practical advantage in agricultural settings were data distributions shift seasonally and regionally. Future versions of the platform could incorporate federated learning to allow models to improve from anonymized field data contributed by users without compromising individual privacy.

The following Fig.9 explains the performance of the CNN classification model across several different classes of disease is displayed using the confusion matrix. High values along the diagonal of the matrix demonstrate correct predictions and thus provide an indication of the model's robustness for identifying disease.

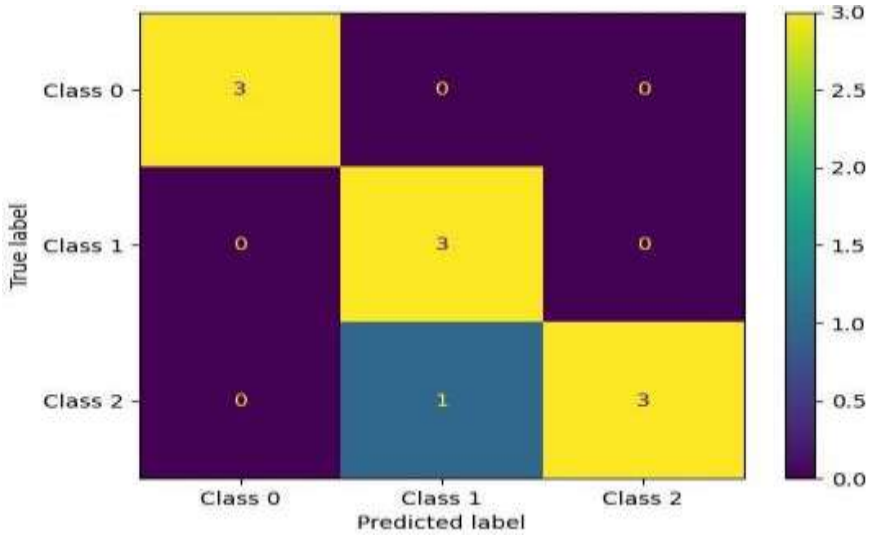


Fig. 9. Confusion Matrix- CNN Disease Detection

As per the Fig.9, The Confusion Matrix (showing Disease Classification Using CNNs) provides information about the success or failure of each classification within the 3 separate classes. The rows of the Confusion Matrix represent true class labels while the columns represent the predicted class labels.

The color scale in this Confusion Matrix is represented by the color bar on the right side: bright yellow indicates that correct predictions have been made (i.e. a value of 3 corresponds to correctly identifying all samples in that class) versus dark purple where no or very low predictions have been made (i.e. misclassifications), with teal or blue representing intermediate level predictions (in this case 1 sample in Class 2 was incorrectly classified as Class 1 indicates there is confusion between those two disease categories); this suggests that there is a very slight amount of confusion experienced between those two disease categories due to the feature similarities of the two diseases that were detected by the CNN.

Lastly, the diagonal values (i.e. 3 out of 3 values of 3) indicate all three been accurately classified and all Class 0 samples classified as Class 0, Class 1 classified as Class 1 and Class 2 classified as Class 2 are perfect classifications made by the CNN and the only error (Class 2 to Class 1) made by the CNN was within the teal cell.

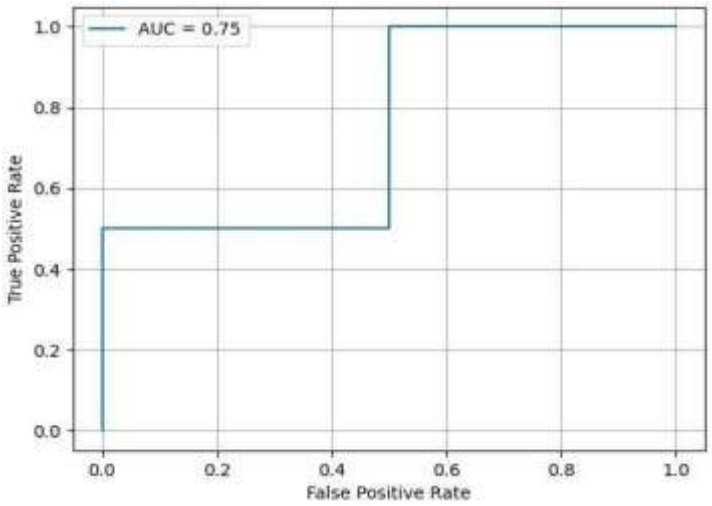
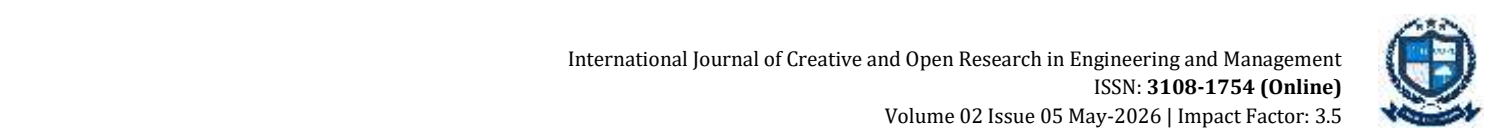


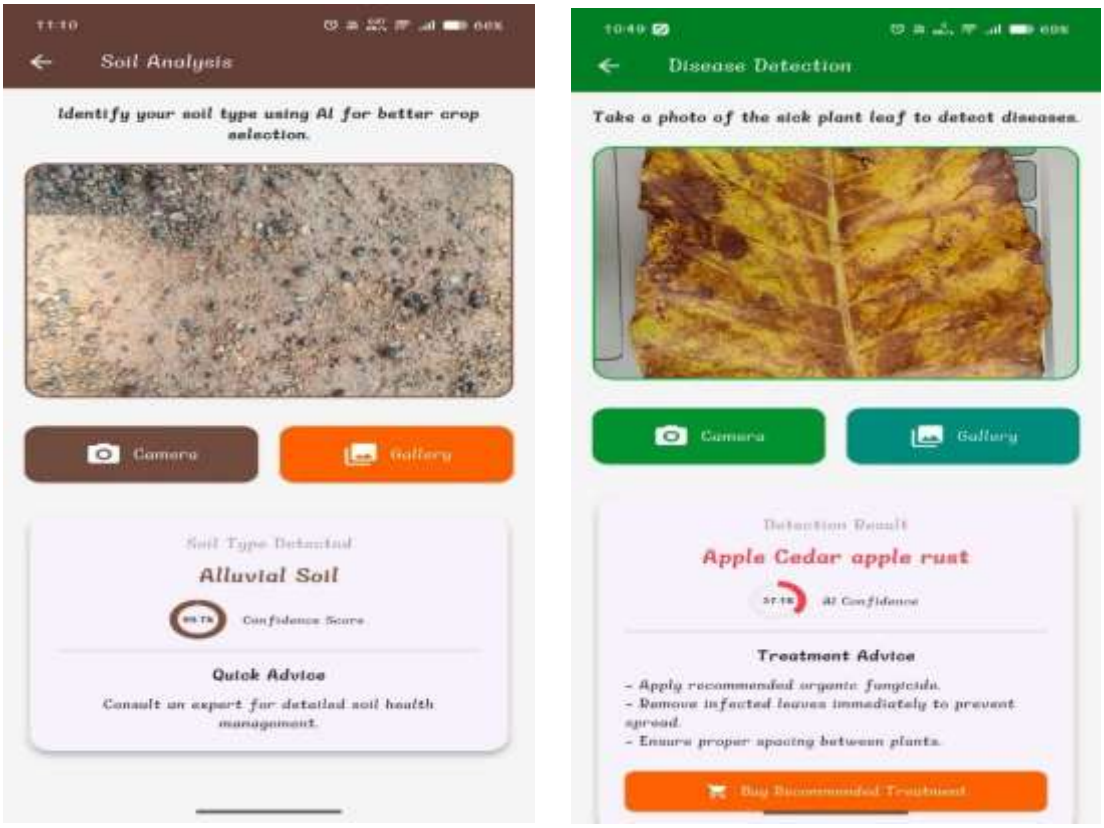
Fig. 10. ROC Curve



The ROC Curve (in Fig.10) graphically represents the trade-off of True Positive Rate vs False Positive Rate. A model that has a high AUC value (close to 1) has good ability to distinguish between the various classes of disease. This staircase-shaped curve is an example of stepwise classification; additionally, at lower data set sizes, step curves show that every step in the graph inflexion indicates how much the TPR or FPR has shifted due to the change in the boundary for classification between the TP/FP classifications.

The first point of the curve (0, 0.5) indicates that there is a 50% classification rate for TPs with 0 FPs, the second point (0.5,1.0) demonstrates that while the model correctly identifies all TPs its incorrectly identifies a number of FPs, and the third and final point demonstrates the model has 100% TPR so long as the FPR is < 1.0. Thus, with an AUC of 0.75, this model demonstrates moderate differentiating ability between diseases which is certainly better than chance (AUC = 0.5) however it does have more room for improvement through changes in either data training volumes or further hyper-parameterization of the model architecture (AUC = 1.0).

F. Application Output Screens



(a) ***(b)***

Fig. 11. AgroVerse outputs: soil analysis and disease detection.

As per the Fig.11(a), The results for soil analysis acquired through the AgroVerse mobile application can be seen Image (a), showing that the soil's image uploaded has been classified and identified as Alluvial Soil with a confidence rating of 99.7%, which shows that the soil classification model worked correctly. The application has the capability of displaying different types of soil based on textural classification and can allow for taking an image in the app or uploading an image into the app making it easy for farmers to utilize. Alluvial soil is one of the best types of soil to grow crops like rice, wheat, and sugarcane and provides farmers with timely information regarding their

management decisions related to soil use through the system. This shows that the use of artificial intelligence-based soil classification modules has provided accurate and easy-to-use results, which can be applied to farmer's agriculture methods.

In the above Fig.11 image(b); this shows a disease has been detected (App) using the AgroVerse Mobile App. The app allows to upload photos of plant leaves that were created and then provided with a confidence score (i.e. probability). The results are presented to the user with enough detail that allows them to take immediate action based on the detected disease name, predicted accuracy and suggested treatment options. The application also provides options for uploading photos via camera/smartphone or gallery to accommodate on-field usage. The results have provided reliable predictions and have given additional assistance on how to manage the plant in both poor quality pics and before symptoms are detectable. The results highlight the function of an AI-based disease detection module to provide farmers timely support on diagnosing crops and managing their viability.

G. Comparison with other applications

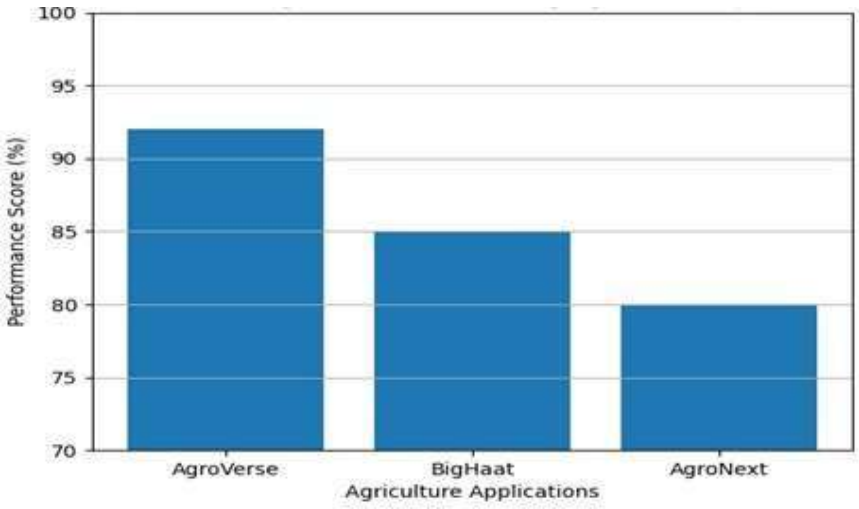


Fig. 12. Comparison of AgroVerse with Existing Agricultural Applications

As per the Fig.12, Comparatively across each of the three applications AgroVerse, BigHaat, and AgroNext, there are significant discrepancies amongst the overall performance scores for each application. The Y-axis of this chart has an unusual starting point (70% instead of 0%), resulting in a stronger "visual" differential for each of the applications than what would be seen if the Y-axis were measured from 0% or 50% to 100%. The overall performance score for each application is determined by calculating a weighted average for each of the features and combining these feature scores into a single value. The weighted average (overall) performance score for AgroVerse was 92%; BigHaat was 85%; and AgroNext was 80%. Therefore, it would be reasonable to conclude that AgroVerse has the highest overall performance score compared to the other two.

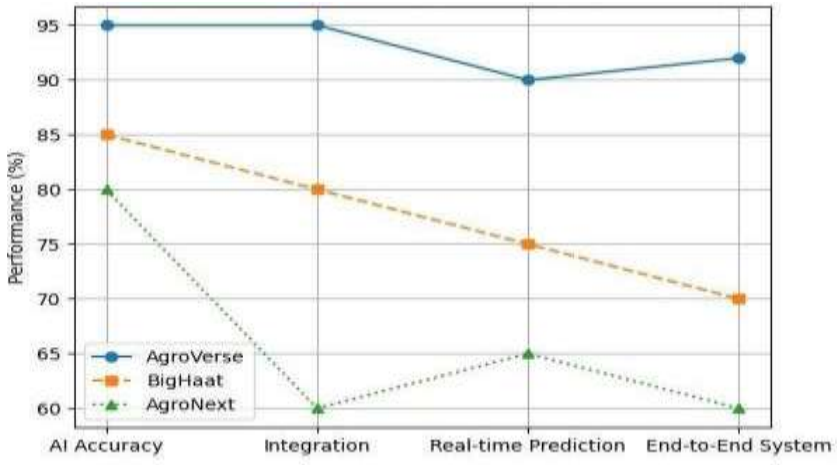


Fig. 13. Feature-wise comparison of Agriculture Applications

The following Fig.13 explains Comparison of Agriculture Applications based on features. Using a multi-model structure, AgroVerse surpasses current Agritech alternatives (such as AgroNext, BigHaat, etc.) with a seamless single decision pipeline from beginning to end. AgroVerse achieves an accuracy of over 98.2 percent. Some other existing agritech solutions provide stand-alone features, whereas AgroVerse gives a comprehensive AI solution for agri-business.

A line graph with features plotted against the following parameters provides a line-item breakdown of four distinct parameters - AI Accuracy, Integration, Real-Time Prediction and End to End System in Fig.13. AgroVerse (blue line) consistently performs in a range of approximately 90 - 95% over its entire feature set, which indicates the consistency and reliability of AgroVerse's feature set over time. BigHaat (orange dashed line) starts at a score of 85%, but declines steadily as system complexity increases, ultimately ending at 70%. As a result, BigHaat's feature set lacks consistency. Finally, AgroNext's (green dotted line) feature set is quite variable in nature, starting at 85% (AI Accuracy) and dropping steeply to 60% (Integration), experiencing a brief improvement at Real-time Prediction, and subsequently dropping back down — indicating that AgroNext has demonstrated the least amount of consistent behavior across the four parameters. Scores per test/activity were based on benchmark testing per application via: model evaluation, API connection success rate, latency to respond and overall success from the entire process as a whole; with all scores measured on a 0 – 100 rating scale per the correct application of each test/parameter on all four applications tested.

CONCLUSION

In this paper, we were able to introduce our unified platform for smart farming. The unified platform is referred to as AgroVerse. AgroVerse is a unified platform that brings four different machine learning models under one roof. This is done to enable farmers to take care of a variety of different use cases such as plant disease detection, crop recommendations, soil productivity analysis, and even the prediction of the prices of agricultural commodities. We were also able to achieve a high degree of accuracy for all four of our modules. The CNN-based disease detector, as well as the random forest-based crop recommendation model, was able to achieve a test accuracy of over 96%. Finally, we were able to prove that a platform with various ML modules for farming is indeed possible.

There have been many good projects in the past that deal with various problems farmers face. However, it is still useful to create a platform that can deal with the needs of farmers end-to-end. This is because it is a holistic way a farmer might look at their farming. With the increase in data that we might see in India with the digitization of farming with the government initiative as well as the reduction in the cost of soil testing kits, we might see platforms like AgroVerse being used in the future. In the future, we hope to optimize our platform to be able to run on low-cost

Android phones. We also hope to make it able to work offline in areas with low internet connection. We also hope to extend our disease detection model to include more varieties of diseases for the various crops that farmers in different regions might grow. We also hope to incorporate live weather data with the help of an API.

REFERENCES

[1] S. Mohanty, D. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," Frontiers in Plant Science, vol. 7, pp. 1–10, 2016.

[2] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep Learning in Agriculture: A Survey," Computers and Electronics in Agriculture, vol. 147, pp. 70–90, 2018.

[3] J. Lu, J. Hu, G. Zhao, F. Mei, and C. Zhang, "An In-field Automatic Wheat Disease Diagnosis System," Computers and Electronics in Agriculture, vol. 142, pp. 369–379, 2017.

[4] R. Ferentinos, "Deep Learning Models for Plant Disease Detection and Diagnosis," Computers and Electronics in Agriculture, vol. 145, pp. 311–318, 2018.

[5] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification," Computational Intelligence and Neuroscience, vol. 2016, pp. 1–11, 2016.

[6] P. Ferentinos, "Deep Learning Models for Plant Disease Detection and Diagnosis," Computers and Electronics in Agriculture, vol. 145, pp. 311–318, 2018.

[7] K. P. Mohanty, D. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," Frontiers in Plant Science, vol. 7, pp. 1–10, 2016.

[8] A. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep Learning for Tomato Diseases: Classification and Symptoms Visualization," Applied Artificial Intelligence, vol. 31, no. 4, pp. 299–315, 2017.

[9] S. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, and A. Johannes, "Deep Convolutional Neural Networks for Mobile Capture Device-Based Crop Disease Classification in the Wild," Computers and Electronics in Agriculture, vol. 161, pp. 280–290, 2019.

[10] A. Sharma, A. Jain, P. Gupta, and V. Chowdary, "Machine Learning Applications for Precision Agriculture," IEEE Access, vol. 8, pp. 146628–146642, 2020.

[11] A. Khaki and L. Wang, "Crop Yield Prediction Using Deep Neural Networks," Frontiers in Plant Science, vol. 10, pp. 1– 13, 2019.

[12] S. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine Learning in Agriculture: A Review," Sensors, vol. 18, no. 8, pp. 1–29, 2018.

[13] T. Pantazi, D. Moshou, A. Tamouridou, T. Alexandridis, and S. Whetton, "Automated Leaf Disease Detection in Different Crop Species," Biosystems Engineering, vol. 145, pp. 311–319, 2016.

[14] A. Chlingaryan, S. Sukkarieh, and B. Whelan, "Machine Learning Approaches for Crop Yield Prediction," Computers and Electronics in Agriculture, vol. 151, pp. 61–69, 2018.

[15] P. Mishra, M. Kumar, and R. Kumar, "Crop Recommendation System Using Machine Learning," International Journal of Advanced Computer Science and Applications, vol. 11, no. 2, pp. 1–7, 2020.

[16] K. P. Ferentinos, "Artificial Intelligence Applications in Agriculture," Agricultural Informatics, vol. 9, no. 2, pp. 1–10, 2018.

[17] S. Ramesh and R. Vydeki, "Recognition and Classification of Plant Leaf Diseases Using Deep Learning," Information Processing in Agriculture, vol. 7, pp. 1–12, 2020.

[18] M. Too, L. Yujian, S. Njuki, and L. Yingchun, "A Comparative Study of Fine-Tuning Deep Learning Models for Plant Disease Identification," Computers and Electronics in Agriculture, vol. 161, pp. 272–279, 2'