AI-Powered Agriculture Advisor: Crop Recommendation, Yield Prediction and Disease Detection Using Soil, Weather, and Image Data

Dissertation
By
Muhammad Mubeen
(M00963104)

Supervisor

Prof. Gao XiaoHong

October 2025

This thesis is submitted in part of fulfilment of the requirements for the MSc. Data Science at Middlesex University, 2025

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Gao XiaoHong, for their continuous guidance, encouragement, and invaluable support throughout the course of my Master's dissertation. Their expertise, patience, and constructive feedback were instrumental in shaping this research and helping me overcome challenges during both the experimental and writing phases. I feel truly fortunate to have worked under their mentorship.

I am profoundly thankful to my parents and family for their unwavering love, prayers, and constant encouragement, which kept me motivated during this demanding journey. Their faith in me has always been my greatest source of strength.

I would also like to extend my appreciation to the faculty and staff of Middlesex University London, especially the administrative and support teams, for their kindness and timely assistance whenever I encountered difficulties.

Finally, I wish to acknowledge all my friends, peers, and everyone who, in one way or another, provided guidance, encouragement, or support. Without their direct and indirect contributions, the successful completion of this dissertation would not have been possible.

Abstract

Agriculture remains focal in food security yet farmers face critical issues surrounding the choice of crops to cultivate, production estimates and early detection of diseases. Conventional methods typically place a lot of emphasis on manual checks and experience, which may result into inefficiencies, and in most instances, losses of crops. To solve these problems, we created an AI-based Agriculture Advisor, which is a machine learning, deep learning, and weather-yield analysis system in one.

The project is developed on a basis of three elements. The former is a crop recommendation module, which is trained on two datasets. One of these was a simple data set consisting of 17,600 records and seven soil-climate variables (N, P, K, pH, temperature, humidity, rainfall). The other was a significantly bigger seasonal data of 110,000 records comprising 28 soil and weather variables. Once we preprocessed and tested our code, we discovered that the plain dataset produced much better results- reaching the accuracy of 99.3 with good precision, recall and F1 all near 0.99. In comparison, the seasonal data did not do well at approximately 50 percent accuracy. According to this comparison, the final model was selected on the background of the basic dataset.

The second section of the system is to predict crops yields depending on the weather. To this, we used two datasets. One was a local farm-level weather record covering (around 19 years of data) comprising 41,300 records with five major characteristics. The second was a considerably larger international data set of more than 220 thousand yield records of various crops in many different countries, including crucial aspects like rainfall, temperature, and the use of pesticides. Both sources were not written at the same time and this is why we needed to preprocess and align these sources.After being combined, the model could project yield patterns of crops under future seasonal conditions. This measure allows verifying the recommendations of soil-based crops with the results of the expected weather conditions, in order that the recommendations are not only technically viable, but also practical and effective to farmers.

Plant disease detection module is the third component of the project. In this case, we have been using the PlantVillage dataset, consisting of over 20,000 labelled images of crop leaf with 15 types of disease. The pictures were normalised and rescaled and then the data was divided into training, validation and test sets. Two deep learning models were developed: one was an EfficientNetB0 model that was fine-tuned to detect agricultural diseases and achieved a test accuracy of 93.6%. The second was a tradition CNN which had five convolutional layers. It still gave good results and confirmed the approach, although it did it slightly worse than EfficientNetB0.

We developed a basic prototype interface in Streamlit, in order to demonstrate how these models might be applied in practice. The interface was not the primary focus of the work but served to show how the system may one day turn into a tool accessible to farmers.

Altogether, the project gathers together three interdependent AI components: crop recommendation on soil base with the help of a Random Forest model, plant diseases detection with the assistance of EfficientNetB0 and a custom CNN, and yield prediction with the help of combined weather and yield data. Collectively, these modules represent a practical end-to-end system that emphasizes the way machine learning and deep learning can be used to transform agriculture to become more data-driven, efficient, and sustainability.

Table of Contents

1. Introduction

Agriculture continues to be the foundation of global food security, yet countless farmers still face ongoing struggles caused by unpredictable weather, declining soil fertility, and widespread plant diseases that lower productivity. Traditional farming decisions, such as which crop to grow, when to plant, or how to manage disease outbreaks, are still largely based on experience or broad regional advice. While such approaches have value, they often lack precision and fail to make use of the growing availability of data. In many developing regions, these decisions can mean the difference between a successful season and a complete loss, especially as climate change continues to intensify fluctuations in temperature and rainfall. Because of this, the agricultural sector now stands at an important turning point, where modern technology, particularly artificial intelligence (AI) and machine learning (ML), offers practical solutions for improving yields, promoting sustainability, and reducing risks of disease.

Over the past few years, researchers and professionals have increasingly started to apply AI models within agriculture to strengthen decision-making through prediction and automation. Machine learning helps uncover complex, non-linear connections within diverse agricultural data, including soil composition, weather information, and crop health images, turning them into useful insights. The use of AI is not limited to one particular stage of farming, it now supports soil analysis, crop planning, growth observation, and yield forecasting. Many studies have shown that such data-driven systems enable farmers to make better decisions about which crops to plant, forecast yields more accurately, and identify diseases in their early stages to avoid major losses. With the continuous rise in agricultural data and improved understanding of how models explain their predictions, it has become possible to design intelligent advisory systems that work as digital assistants for farmers, offering personalized and timely recommendations suited to local conditions.

The project titled “AI-Driven Agriculture Advisor: Crop Recommendation, Yield Prediction, and Plant Disease Detection” has been developed as a complete intelligent framework that brings together classical machine learning and modern deep learning techniques. The purpose of this system is to assist farmers through three interconnected modules that together form a full decision-support pipeline for agriculture.

The first module, Crop Recommendation, uses supervised machine learning to identify the most suitable crops based on soil and climatic conditions. By analyzing attributes such as Nitrogen (N), Phosphorus (P), Potassium (K), pH, rainfall, temperature, and humidity, a Random Forest Classifier determines which crops are best suited for a particular environment. The second module, Yield Prediction, relies on regression methods to estimate future crop yields by combining historical yield data with weather records. It takes into account environmental and broader factors, rainfall, temperature, humidity, wind, pesticide use, and longer-term rainfall averages, in order to forecast yield per hectare. The third part of the system, the Plant Disease Detection module, relies on deep learning, in this case convolutional neural networks, to sort leaf images into healthy and diseased categories. We explored two routes for this task: transfer learning with the EfficientNetB0 model and, alongside it, a custom CNN built from scratch. Both were trained on the PlantVillage collection, which contains thousands of labelled leaf images across fifteen disease types, and both produced strong and dependable results, with EfficientNetB0 doing slightly better overall.

As for data, the study stood on a broad foundation. In total, five datasets were used: two for crop recommendation, a basic set with seven features and a seasonal set with twenty-eight, one for disease detection (the PlantVillage image data), and two for yield and weather forecasting, a global yield dataset

and a local weather record spanning 2006–2024. These sources, once aligned and prepared, provided a consistent base on which to build and test the advisor. To ensure alignment between the yield and weather data, preprocessing steps such as year-shifting and aggregation were applied so that every year's yield data corresponded directly with its weather conditions. This alignment made it possible for the model to capture stable temporal links between environmental factors and crop performance. The system, therefore, combines structured tabular data for crop and yield analysis with unstructured image data for disease classification, demonstrating a practical multi-modal use of artificial intelligence in agriculture.

The machine learning algorithms chosen for this project were selected for both their reliability and interpretability. The Random Forest algorithm was used for classification and regression tasks due to its strength in managing non-linear relationships, reducing overfitting, and offering insight into feature importance. Its ensemble-based structure also provides stable and accurate results, even when working with relatively small but meaningful datasets. For the image-based module, deep learning was adopted because of its strong performance in visual recognition tasks. The EfficientNetB0 model, which was pre-trained on ImageNet, enabled transfer learning to speed up convergence and improve accuracy, even when the available dataset was moderate in size. Alongside this, a custom CNN architecture was built from the ground up, featuring multiple convolutional and pooling layers, to test performance under fully controlled conditions. Both models gone through preprocessing steps such as image resizing, normalization, and augmentation to improve generalization. The comparison between the pre-trained EfficientNetB0 and the custom CNN provided valuable insights into the balance between computational efficiency and predictive accuracy when classifying agricultural images.

To make the whole thing easier to use, we put all three models into one simple interface built in Streamlit. Think of it as a working demo of how a farmer or an agriculture officer might actually use the system day to day. You type in the soil details and location, and it gives you the top three crop suggestions. You can also get a quick estimate of expected yield for those crops, and even upload a leaf photo to check for disease on the spot.

The interface pulls in weather automatically, using a ten-year average for the chosen place and year, so the different parts of the system stay in sync. We saved all the trained models and config files so results are repeatable and the setup can be scaled later without guesswork. Because the design is modular, each piece can run on its own or plug into the full advisor, which makes it easier to deploy in different farm contexts.

The reason for building this was simple: people need precision agriculture tools that are easy to access, make sense, and are backed by solid methods. Lots of studies look at crop recommendation, yield forecasting, or disease detection on their own, but very few pull all three together in one place. In practice, farmers end up jumping between apps or asking different experts for soil advice, crop choices, and disease management. This advisor closes that gap by putting everything on one platform, offering help from soil prep all the way to harvest. We also added explainable-AI features, like feature importance for the tabular models and saliency maps for the CNNs, so users can see why a recommendation was made and trust the output.

The impact goes beyond one farm, too. If a system like this is used across regions, the combined insights could guide crop planning, resource allocation, and policy to support food sustainability. For example, forecasting yields from climate patterns can help governments prepare for supply issues, and early disease

detection can stop outbreaks before they spread. On the environmental side, smarter fertilizer use, fewer pesticides, and healthier soil all move farming toward more sustainable practices that align with broader climate-goals.

Both models confirmed the feasibility of reliable, image-based plant disease detection. While EfficientNetB0 achieved superior accuracy and faster convergence through transfer learning, the custom CNN offered competitive performance with far fewer parameters (~0.46M vs. ~4.07M). Together, the results highlight the benefits of transfer learning while demonstrating that even a research therefore aims to address the following core questions:

1. How effectively can a machine learning model recommend optimal crops using minimal but relevant soil and weather inputs?

2. How accurately can annualized weather and macro features forecast yield across crops and regions?

3. To what extent can deep learning architectures diagnose plant diseases from leaf imagery with high precision?

4. How can the integration of these three capabilities enhance decision support for farmers in a unified platform?

In summary, this project makes the following contributions:

- Development of a three-component AI system for crop recommendation, yield prediction, and plant disease detection, providing an integrated advisory workflow for farmers.
- Construction and alignment of five distinct datasets, combining soil, weather, yield, and image data for multi-modal learning.
- Comparative analysis between EfficientNetB0 transfer learning and a custom CNN for plant disease classification.
- Implementation of an interactive Streamlit interface to demonstrate real-world usability.
- Advancement toward explainable, reproducible, and sustainable AI in agriculture through modular design and configuration-driven modeling.

The AI-Driven Agriculture Advisor represents a step toward digital farming systems capable of guiding what to plant, predicting what to expect, and protecting what grows. By uniting data science and agronomy, it showcases how artificial intelligence can not only improve productivity but also contribute to sustainable and resilient agricultural futures.

2. Literature Review

In recent years, agriculture has drawn increasing attention to the use of artificial intelligence and machine learning as a way to improve decisions and raise productivity. Surveys and case studies show these methods being used across crop management from diagnosing plant diseases to forecasting yields—with results that are often encouraging [2][13][17][18]. Even so, much of the existing work still treats these challenges one by one, rather than as parts of a single, connected process.

With this gap in mind, we outline an integrated AI-powered Agriculture Advisor that supports farmers in three linked areas: recommending which crops to plant based on soil and climate data, predicting likely yields by combining historical records with weather patterns, and detecting plant diseases from leaf images using deep learning. In what follows, this review looks at prior work in each area, points out where the gaps remain, and explains how our project builds on the literature while bringing these pieces together into one practical system.

2.1 Crop Recommendation Using Soil and Climate Data

Choosing the right crop for a particular field and season is a crucial call, and it has traditionally leaned on farmer experience or broad guidelines. Data-driven recommendation systems try to improve this choice by looking at soil properties (like nutrient levels and pH) together with climate factors such as temperature and rainfall. In most cases, researchers frame the task as a classification or ranking problem, letting machine-learning models learn from past records which crops tend to do best under certain conditions. As Dhingra et al. point out, modern AI can weigh many parameters at once—soil status, weather forecasts, seed type, pest pressure, and more—to suggest not only the most suitable crop but sometimes even the seed variety, something manual rules struggle to match [16].

In one recent example, Mohammadi & Khosravi used supervised ML on a combined soil–climate dataset to recommend crops and also estimate likely yield. They compared multiple classifiers to see which one most accurately picked the ideal crop for the given conditions, showing that this sort of decision support is feasible in practice [4]. Likewise, Panigrahi et al. designed a crop advisory focused on maize that folded in local agronomic data to guide selection at the farm level [6]. Taken together, these studies show that common classification models—decision trees, random forests, support vector machines—can reach high accuracy in matching crops to suitable environments [4][16].

Even with these early wins, crop recommendation is still less explored than yield prediction or disease detection. Many existing models are tied to a single region, trained on small local datasets, and don't transfer well to different soils or climates [16]. Most prototypes also stop at the research stage and aren't deployed widely, which shows a real gap between lab results and field use.

Our work tries to close that gap by using a robust ensemble Random Forest trained on a broader mix of soil attributes and weather conditions to generate recommendations. In line with prior studies [16], we focus on features that matter most for crop viability: nitrogen, phosphorus, potassium, pH, temperature, humidity, and rainfall. By evaluating on an expanded dataset, the goal is a system that generalizes better beyond one location. This module follows the same idea behind cognitive computing pulling together data from multiple sources and turning it into clear, practical advice [16] so farmers can get personalized crop suggestions for their own region, provided the right input data is available.

2.2 Crop Yield Prediction from Historical Data

Accurate yield prediction is essential for planning and overall food security. Traditional approaches—whether statistical regressions or crop simulation models—often fall short because they struggle to capture the complex, non-linear interactions among weather, soil, and crop genetics. Machine learning offers a different route by learning these patterns directly from historical data. Early work showed clear gains over classical models: for example, Jeong et al. reported that a Random Forest model predicted regional yields for wheat, maize, and potato with much lower error—root mean square error around ~6–14% of mean yield—compared with a multiple linear regression baseline, which ranged from 14–49% [12]. This result underscored how ensemble trees can account for climate-driven variability far better than linear methods [12].

Later studies have strengthened this view. In a broad review, Liakos et al. found that artificial neural networks, support vector machines, and ensemble techniques have been used widely and often outperform manual or purely linear approaches for yield forecasting [13]. Similarly, Singh et al. survey a range of yield modeling efforts and conclude that data-driven methods are highly promising across different crops, provided there is enough data and the right set of features [14].

Beyond standard ML, deep learning has gathered momentum for yield prediction, especially as big data from remote sensing and IoT devices has become more available. As Chlingaryan et al. note, newer sensing tools—satellite imagery, UAV drones, soil and weather sensors—paired with machine learning can deliver accurate and cost-effective yield forecasts for precision agriculture [3]. Bringing together multiple data types (weather time-series, soil measurements, vegetation indices from imagery) fits deep learning well because these models can learn complex features automatically.

For example, Khaki et al. proposed a hybrid framework that combines convolutional neural networks with recurrent neural networks to forecast yields [11]. The CNN side learns spatial patterns—such as regional rainfall or soil-moisture signals—while the RNN side tracks the temporal dynamics across seasons. Used together, this CNN–RNN design improved accuracy compared with using either component on its own [11]. In a related line, Zhang et al. applied deep CNNs to sequences of satellite-derived NDVI images to predict yields, showing that learning from canopy development over the season can outperform traditional regression approaches by a clear margin [19].

A persistent challenge is generalizability: a model trained in one region or crop may struggle elsewhere because climate, soils, and management practices differ. Recent work tackles this with transfer learning and domain adaptation. Sagi & Jain point out that ensemble and hybrid methods can improve robustness and argue for approaches that carry knowledge across domains [18]. A strong example is Ma et al. [20], who addressed cross-region transfer by using an unsupervised domain-adversarial neural network. Their model aligns feature representations from different areas and, by combining inputs on soil, weather, and management with adversarial training, achieved strong performance both locally and when transferred to new regions [20]. These results suggest that pairing deep learning with transfer learning can ease data limitations in places with sparse yield records.

In short, the literature shows a path from simpler ML models [12][13] to more advanced deep learning systems for yield prediction [11][19]. Overall, data-driven models capture yield-relevant patterns more effectively than fixed, hand-crafted formulas. Still, gaps remain: many methods depend on long historical records and can underperform outside their training conditions [3][20]. Our project builds on these

insights by using a Random Forest regressor for yield prediction—chosen for its strong results on tabular agro-meteorological data [12]—and by training it on a merged dataset that spans long-term weather and yield statistics. While deep learning could push accuracy further, we prioritize interpretability and the realities of limited data, recognizing that well-tuned ensemble models can be robust with modest sample sizes [13][15]. By linking seasonal weather features to yield outcomes, our yield module provides farmers with a grounded estimate of expected production—useful for planning resources and managing risk.

2.3 Plant Disease Detection with Deep Learning

Plant diseases are a major risk to crop productivity worldwide. Detecting foliar diseases early and accurately is essential to reduce losses. Traditional practice relies on field scouting and expert judgment, which can be time consuming and subjective. Over the past decade, computer vision methods, especially deep learning with convolutional neural networks, have transformed automated plant disease detection [2][18]. A key early study by Mohanty et al. [1] showed that a deep CNN can learn to recognize many crop diseases directly from leaf images. Using 54,306 images from the PlantVillage dataset covering 14 crop species and 26 diseases, their model achieved above 99 percent classification accuracy, setting a strong baseline for image based diagnosis [1]. Around the same period, Brahimi et al. [7] and Amara et al. [10] reported similarly high results with deep CNNs. Brahimi et al. [7] worked on several crops including tomato leaf diseases, while Amara et al. [10] focused on banana leaves using a LeNet style CNN and achieved robust performance for problems such as Sigatoka and Xanthomonas wilt. Together, these studies showed that CNNs can learn visual signs of plant stress (spots, lesions, discoloration) far better than hand crafted features, enabling reliable automated diagnostics.

Later research moved in two main directions: comparing CNN architectures and coping with real field conditions. On the architecture side, Ferentinos [8] tested multiple models (AlexNet, VGG16, and a custom deep network) on a dataset spanning 58 plant and disease classes. The best model exceeded 99.5 percent overall accuracy, confirming that deep networks can reach near perfect classification when enough training data is available [8]. Too et al. [5] then evaluated several pre trained CNNs (for example ResNet, Inception, DenseNet) using transfer learning for leaf disease identification. They found that fine tuning pre trained models gives excellent results on PlantVillage, often above 95 percent, while remaining computationally efficient, with DenseNet variants among the strongest performers [5]. These findings help steer model choices in practice. In our project, for instance, we drew on this line of work in selecting EfficientNet as a modern CNN that balances accuracy with efficiency.

The second challenge is moving from controlled datasets to real field images. Many early studies, including [1][8], trained and tested on leaves photographed against clean, uniform backgrounds, often in labs or studios. Out in the field, leaves appear within cluttered scenes, under changing light, at different angles, and alongside other objects, which can reduce model accuracy. To address this gap, Singh et al. [14] introduced the PlantDoc dataset in 2020, with thousands of diseased plant images captured under natural conditions, complete with varied lighting and busy backgrounds. Models trained or evaluated on PlantDoc generally performed worse than on PlantVillage, underscoring the need for domain adaptation in disease detection. This line of work has pushed researchers toward more robust methods. For example, Picon et al. [9] studied wheat disease detection directly in the field, using a ResNet-50 network to classify septoria, tan spot, and rust, and adding an attention mechanism so the model could focus on affected leaf areas. Their approach reached about 98% balanced accuracy on field imagery [9], which is impressive given the added complexity. Likewise, Panigrahi et al. [6] and others have applied deep learning and machine vision

to targeted crop–disease pairs, such as maize leaf blight, showing that the techniques translate across different crops.

Overall, deep learning for plant disease detection is now a mature and successful area. Many studies report accuracy above 90%, in some cases approaching expert-level performance [1][7][8]. Still, important gaps remain. A major issue is the heavy need for large labeled datasets. Models like those in Mohanty et al. [1] or Too et al. [5] rely on tens of thousands of images, which may not exist for every crop or region. Performance can also drop when models are deployed in settings not seen during training, such as a new farm, a different camera, or an unfamiliar disease variant [14]. To improve generalization, recent work suggests stronger data augmentation, domain adaptation, or few-shot learning [2][14]. Another limitation is that most studies focus on classification only. In practice, farmers may need to locate lesions on the plant or deal with multiple stresses at once, which calls for advances in detection and segmentation. Our project builds on these best practices by using a modern CNN with transfer learning for accuracy, and by training on a diverse set of disease images for robustness. The goal is to provide on-the-spot diagnosis of common plant diseases from leaf photos, giving farmers earlier warnings and more precise interventions in line with prior deep learning efforts in agriculture [2][18].

2.4 Research Gaps and Project Contributions

From the literature across these three areas, one clear gap is the lack of truly integrated systems. Most studies focus on a single task at a time. They build a yield predictor, or a crop recommendation tool, or a disease classifier, but rarely all together. This kind of specialization has pushed each field forward, yet farmers face all of these challenges at once. A few works are starting to move toward combined decision support. For example, Kamilaris and Prenafeta-Boldú note that linking multiple AI tools for sensing, prediction, and decision support could strongly improve farm management, but they present this more as an opportunity than a finished reality [2]. Mohammadi and Khosravi take a step in that direction by combining crop recommendation with yield prediction in a single study, although complete end to end advisory frameworks are still uncommon in the academic record [4].

A second gap is generalizability and real deployment. Yield models often overfit to regions with abundant data and may not transfer elsewhere without adaptation [20]. Disease models trained on clean, ideal datasets can struggle in the field unless they are retrained or supported with real field images [14]. Crop recommendation systems also need local calibration to be genuinely helpful to farmers [16]. These limits point to a need for approaches that are both data efficient and adaptable. Our project addresses these issues by drawing on diverse data sources and relying on robust algorithms. The Random Forest models we use for crop recommendation and yield regression are known to generalize well and to handle shifting data distributions with relatively little tuning [12][15]. For disease detection, a pre trained CNN such as EfficientNet lets us reuse knowledge from large general image datasets, which improves results even when agricultural images are limited, as earlier work suggests [5]. We also place emphasis on interpretability and simplicity where possible, for example using feature importance in Random Forest to explain crop suggestions, which helps with real world adoption and echoes recommendations from reviews of AI in agriculture [13][18].

In conclusion, this AI-powered Agriculture Advisor draws on the best available research in crop recommendation, yield prediction, and plant disease detection. By bringing all three parts together, it aims to offer farmers a single, holistic support system, something the literature suggests has not yet been fully achieved. The design uses proven ML and DL techniques from prior work, such as ensemble methods

for structured data and CNNs for image analysis, and it tackles known gaps by emphasizing model generalization and the integration of multiple tasks. Through this project, we show that combining several AI tools can create a stronger advisory platform that improves decision-making across the farming cycle. This approach helps narrow the gap between isolated research results and the practical needs in the field, and it contributes to the wider push toward smart farming and sustainable agriculture.

3. Methodologies

To build an integrated AI-Driven Agriculture Advisor, we organized the work around a single, configuration-driven pipeline. The architecture was set up to prioritize reproducibility, modularity, and clarity, so that each part could be developed and tested on its own while still fitting neatly into the overall system.

Instead of hardcoding dataset paths or model settings inside separate scripts or notebooks, we kept everything in one config.yaml file. This made datasets, feature lists, and model hyperparameters consistent across modules, and it also made experiments easier to run again. Any change to a parameter could be done in one place, which helped with clean comparisons and reliable results.

The methodological design was guided by the following principles:

- **Pipeline-based structure**: All raw datasets were organized under /data/raw/, processed intermediate files under /data/processed/, and trained models stored in /models/.
- **Config-driven development**: The config.yaml file defined dataset locations, preprocessing steps, model parameters, and evaluation settings for each component.
- **Separation of components**: Each module Crop Recommendation, Weather-Aware Crop Yield Prediction, and Plant Disease Detection was implemented in separate notebooks to maintain modularity and ensure clarity in testing and development.
- **Model persistence**: Following training, models were saved for deployment, tabular models using .joblib, and deep learning models using .h5 or TensorFlow's SavedModel format. This enabled direct loading into the Streamlit interface (app.py) without the need for retraining.
- **Libraries and tools used**:
    - *Data handling & preprocessing*: pandas, numpy, scikit-learn
    - *Visualization*: matplotlib, seaborn
    - *Machine learning & deep learning*: scikit-learn (RandomForest), TensorFlow/Keras (EfficientNet, CNN)
    - *Pipeline and utility support*: joblib (model saving/loading), yaml (configuration management), os and pathlib (file organization)

This Methodology chapter is structured into three main sections, each reflecting one of the core components of the system:

3.1 Crop Recommendation
3.2 Weather-Aware Crop Yield Prediction
3.3 Plant Disease Detection

Each section is further divided into three subsections: methodology, dataset collection and preprocessing, and model development.

3.1 Crop Recommendation Using Random Forest Classifier

3.1.1 Methodology

The soil recommendation task was framed as a supervised classification problem. In simple terms, the goal was to predict which crop is most suitable given a set of soil nutrients and environmental variables. We chose a Random Forest Classifier as the main algorithm. Random Forest is an ensemble method that builds many decision trees and then combines their votes, which generally improves generalization and helps avoid overfitting. Each tree is trained on a random sample of the rows and a random subset of the features, which lowers correlation between trees and makes the overall model more stable. This suits agricultural data well, because the relationships are often non linear and the inputs can be a mix of numeric and categorical fields.

For the model setup, the number of estimators was set to 300, meaning 300 trees in the forest. This was a balance between accuracy and runtime. Too few trees can give jittery performance, while very large forests tend to cost more compute without much gain. We did not cap the maximum depth, allowing trees to grow until splits no longer improved purity. A single tree might overfit in that situation, but the averaging effect across the forest keeps it in check. We also fixed a random seed across the training pipeline so results are reproducible, which is important for academic reporting.

The input features depended on the dataset. The basic dataset used seven core attributes: nitrogen, phosphorus, potassium, soil pH, average temperature, humidity, and rainfall. The seasonal dataset expanded this to 28 attributes by adding soil micronutrients, categorical soil properties such as soil color, and seasonal summaries of weather variables. In both cases, the target label was the crop expected to grow best under those soil and climate conditions. To keep evaluation consistent, settings were defined in the configuration files, and we used an 80 to 20 train test split with stratified sampling so that class proportions were preserved.

3.1.2 Dataset Collection and Preprocessing

This component relied on two datasets, both sourced from Kaggle. The first was a simple dataset with 17,600 records and seven soil–climate variables: N, P, K, pH, temperature, humidity, and rainfall. The second was a much larger seasonal dataset with about 110,000 records and 28 features covering both soil properties and weather variables.

We used these two datasets to build and evaluate the soil recommendation models. The basic crop recommendation dataset was relatively clean and well curated. It already contained numeric values for the seven features, so only light preprocessing was needed. We renamed the target column from “label” to “crop,” checked for missing values and duplicates, and made sure the feature names were consistent. The feature list was saved to a configuration file (feature_columns.json) so the same order would be enforced during training, testing, and inference. We also computed default values for each feature and saved them in defaults.json, allowing the final app to fill sensible inputs if a user left something blank.

The seasonal dataset was more involved. It included soil characteristics alongside seasonal climate

summaries and therefore needed additional preparation. To keep naming uniform with the basic dataset, we renamed the target column to “crop.” Categorical fields, such as soil color, were converted to numeric form using one-hot encoding so they could be handled by the Random Forest model. Seasonal weather variables were standardized and explicitly listed in the config to keep runs reproducible. We also looked for class imbalance, since a few crops were underrepresented and could affect performance. Finally, both datasets were split using a stratified 80:20 train–test split to preserve class proportions in training and evaluation.

3.1.3 Model Development

The model development centered on training a Random Forest Classifier for both the basic and the seasonal datasets. All settings and parameters were kept in configuration files so every experiment could be repeated exactly. For the basic dataset, the features were nitrogen, phosphorus, potassium, soil pH, temperature, humidity, and rainfall. The seasonal dataset used these and added further soil and weather variables.

The Random Forest was set to 300 estimators, with no explicit limit on tree depth, and a fixed random seed to keep results consistent across runs. Training mapped the feature vectors to the crop labels on a stratified training split, allowing the classifier to learn non linear decision rules from the interactions between soil conditions and climate. Because the pipeline was configuration driven, it was easy to switch between datasets without changing the core logic. This helped with careful experimentation and also made it straightforward to plug the trained models into the final advisory system.

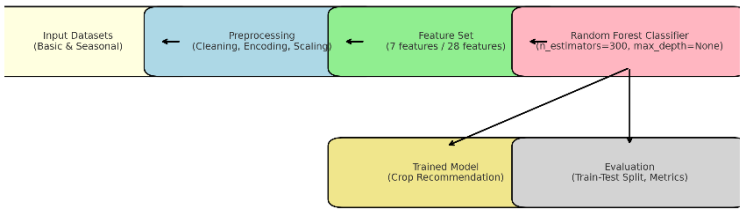


Figure 1

Figure 1 shows the workflow for developing the Crop Recommendation model. It starts with the input data, using both the basic dataset with seven soil and environmental features and the seasonal dataset with twenty eight features. These datasets go through preprocessing, which includes cleaning the data, encoding any categorical fields, and applying scaling where it is needed. The prepared feature sets are then used to train a Random Forest Classifier set to 300 estimators with no maximum depth, so it can capture non linear interactions effectively. The trained classifier becomes the final crop recommendation model, which we evaluate with an 80 to 20 train test split and standard performance metrics to check predictive accuracy and generalization.

3.2 Yield and Weather Prediction Using Random Forest Regressor

3.2.1 Methodology

The second part of the Agriculture Advisor deals with yield prediction. Its goal is to estimate how much a crop is likely to produce in a given region and year. Unlike the soil recommendation module, which focuses

on choosing the best crop under mostly static soil and environmental conditions, this module adds a time and weather view. It forecasts yield in hectograms per hectare (hg/ha) by combining yearly weather summaries with broader agricultural indicators, so it provides a second layer of intelligence on top of the soil based suggestions.

We framed the task as a supervised regression problem. The inputs included both numeric and categorical fields. Weather features were aggregated to yearly summaries, such as mean temperature, total rainfall, average humidity, and mean wind speed. To capture wider context, we added macro indicators like average annual rainfall, pesticide consumption, and long term average temperature. We also included categorical variables for Area (country) and Item (crop) to reflect spatial and crop specific differences. The target was yield measured in hg/ha.

To model this, we used a Random Forest Regressor. The choice was driven by its strength with non linear interactions, its robustness on noisy or mixed tabular data, and the fact that it works well without heavy scaling. Because it averages predictions over many trees, the ensemble helps control overfitting and also provides feature importance that can be interpreted. The regressor was set to 300 estimators, with no explicit limit on tree depth, and a fixed random seed for reproducibility. We used a reproducible 80 to 20 train test split to evaluate how well the model generalizes.

3.2.2 Dataset Collection and Preprocessing

This component used two datasets, both taken from Kaggle. The first was a local, farm-level weather record covering about nineteen years, with 41,300 entries and five main characteristics. The second was a much larger international yield dataset with more than 220,000 records for different crops across many countries, including key factors such as rainfall, temperature, and pesticide use. Because these sources did not cover the same years, we needed to preprocess and align them before modeling.

The yield prediction module therefore integrated two distinct datasets: a farm weather set spanning 2006–2024, and a crop yield set originally covering 1990–2013. Since the time ranges did not match, careful preprocessing was necessary to produce a unified structure for machine learning.

The farm weather data contained daily readings for temperature, rainfall, humidity, and wind speed. We aggregated these to yearly summaries so they would line up with the yield data. In practice, daily temperatures were averaged to form mean annual temperature, daily rainfall was summed to give total annual precipitation, and daily humidity and wind speed were averaged to create yearly means. This produced a compact view of climate variation for each year from 2006 to 2024.

The yield dataset, based on FAO records, originally covered 1990–2013. To align it with the weather series, we kept the 1994–2013 slice and shifted it forward so it represented 2006–2024. This preserved the relative trends while synchronizing with the available weather observations. Each record included yield in hg/ha, macro indicators such as average annual rainfall, pesticide consumption, and average temperature, plus categorical fields for country and crop.

We then created the merged dataset with an inner join on the year and area columns. Each row in the final table represented a crop's yield for a given country and year, together with its weather summaries and macro indicators. Categorical variables (Area and Item) were one-hot encoded, while numeric fields were kept in their original scale, since Random Forest does not require normalization. The full schema—both the encoded categorical columns and the numeric features—was saved in a configuration file

(feature_columns.json) to keep training and inference consistent. Finally, we applied an 80:20 stratified train–test split so that evaluation reflected how well the model generalizes to unseen data.

3.2.3 Model Development

The yield prediction workflow centered on training a Random Forest Regressor to map the merged weather–yield dataset to the target yield. Each record in the structured data represented a crop, a country, and a year, along with its environmental and macro indicators. The regressor was set to 300 trees with no explicit limit on depth, bootstrap sampling enabled, and a fixed random seed so results could be reproduced.

To make the system easier for users, the farm weather data is also used to auto fill the weather inputs for crop recommendation. In practice, the advisor computes a ten year average for the selected location and treats that as the upcoming seasonal weather. This way, users do not need to manually enter weather values to receive crop suggestions.

The full pipeline was built in scikit learn. A ColumnTransformer applied one hot encoding to the categorical fields (Area and Item) and passed numeric features straight through, which kept the final feature matrix aligned with the training schema. The Random Forest Regressor was then trained on this encoded matrix to learn the non linear relationships between weather, macro agricultural conditions, and yield outcomes. For consistency, both the trained model and the schema definition were saved (yield_rf.joblib and feature_columns.json). At inference time, the advisor gathers the relevant weather and macro features for the target year, encodes them according to the saved schema, and returns a predicted yield. These yield estimates are then used to refine the crop recommendations, providing more context aware guidance for farmers.

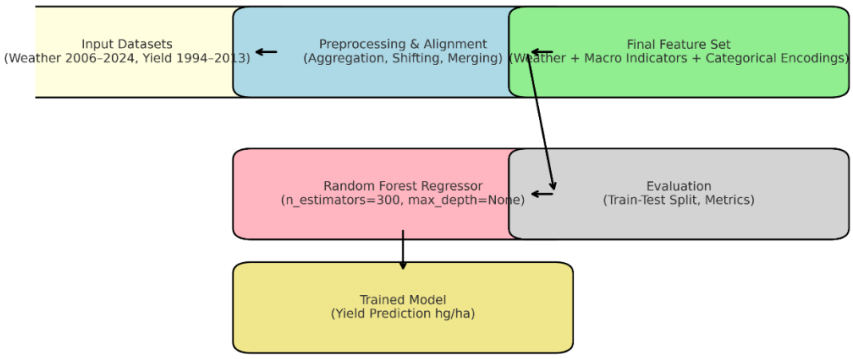


Figure 2

Figure 2 depicts the Yield and Weather Prediction Model Development workflow. The process begins with input datasets, comprising long-term weather records (2006–2024) and crop yield statistics (1994–2013). These datasets are subjected to preprocessing and alignment, including aggregation of daily weather into annual summaries, temporal shifting of yield data to align timelines, and merging with relevant macro-level agricultural indicators. The final feature set integrates weather attributes, macro indicators, and categorical encodings for crops and regions. This dataset is then used to train a Random Forest Regressor, configured with 300 estimators and unrestricted depth, to capture complex non-linear dependencies. The trained regressor is subsequently validated through a train–test split and appropriate regression metrics,

resulting in a robust yield prediction model capable of estimating output in hectograms per hectare (hg/ha).

3.3 Plant Disease Detection Using CNN

3.3.1 Methodology

The plant disease detection module is the third part of the Agriculture Advisor. Its role is to identify crop diseases directly from leaf photos using deep learning, with the aim of helping farmers spot problems early so they can act in time, limit losses, and stop diseases from spreading. We treated it as a multi class image classification task: given an RGB image of a leaf, the model returns one of fifteen categories, covering common fungal, bacterial, and viral diseases, along with a healthy class.

We used the PlantVillage dataset, which is a common benchmark in plant pathology work. To keep results reliable, the data were stratified into training, validation, and test sets so that class proportions stayed consistent across splits. For the EfficientNetB0 setup, the split was about 75 percent training, 15 percent validation, and 10 percent testing, as defined in the configuration. For the from scratch CNN pipeline, we created a new split directory and used a slightly different ratio of 70 percent training, 18 percent validation, and 12 percent testing.

All images were resized to **224×224 pixels**, but preprocessing differed across models:

- **CNN scratch**: pixel intensities were normalized to the range [0,1].
- **EfficientNetB0**: the built-in preprocessing function standardized images to [−1,1].

To help the model generalize and to mimic the kinds of variation seen in real fields, we applied data augmentation to the training images. In both pipelines we used random flips, rotations, and zooms, and for the CNN baseline we also added random contrast changes. The validation and test sets were left untouched so that evaluation remained fair and unbiased.

We then built two complementary models. The first used transfer learning with EfficientNetB0, a convolutional neural network pre trained on ImageNet. Its backbone offers strong feature representations, which we fine tuned for crop disease recognition. The second was a custom CNN trained from scratch, kept deliberately lightweight so it could serve as a baseline against the transfer learning route. Using both approaches allowed us to benchmark performance and compare results side by side

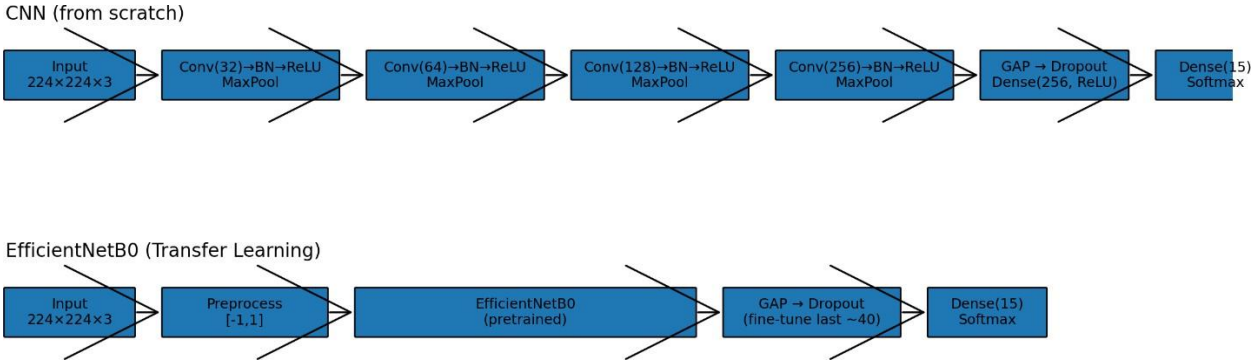


Figure 3

Figure 3 presents the Plant Disease Detection Model Architectures used in this project. Two approaches were explored:

- The top pipeline shows the Convolutional Neural Network that we built from scratch. It takes leaf images of size 224×224×3 and feeds them through a series of convolutional blocks with growing numbers of filters (32, 64, 128, 256). Each block applies batch normalization, a ReLU activation, and max pooling to reduce spatial size while keeping the key patterns. The feature maps then pass into a Global Average Pooling layer, followed by dropout for regularization, and a dense hidden layer with ReLU. The final dense softmax layer outputs one of fifteen classes, corresponding to the plant diseases in our dataset.
- The bottom pipeline outlines the EfficientNetB0 transfer learning setup. Input leaf images are first preprocessed to match what the pretrained model expects. We use the EfficientNetB0 backbone, which was trained on ImageNet, and fine tune the last forty layers. As with the custom CNN, the extracted features then go through a Global Average Pooling layer, dropout for regularization, and a final dense softmax layer that outputs predictions across fifteen classes.

This dual approach allows comparison between a fully customized CNN and a modern transfer learning method, with EfficientNetB0 expected to leverage pretrained feature extraction for improved accuracy and efficiency on the plant disease dataset.

3.3.2 Dataset Collection and Preprocessing

We have been using the PlantVillage dataset which we found from Kaggle, consisting of over 20,000 labelled images of crop leaf with 15 types of disease.

The preprocessing pipeline was designed to transform raw PlantVillage images into a consistent format suitable for deep learning. The raw images were reorganized into a structured directory (plant_split/) containing three subsets: training, validation, and test. Stratified sampling was applied to preserve proportional class representation.

- **EfficientNetB0:** 75% training, 15% validation, 10% test (set via config).
- **CNN scratch:** ~70% training, 18% validation, 12% test (hardcoded split).

Each image was resized to **224×224×3 pixels**. Preprocessing was adapted to the model:

- For the CNN scratch pipeline, pixel intensities were scaled to **[0,1]**.
- For EfficientNetB0, inputs were standardized to **[−1,1]** using the official preprocessing function.

Data augmentation was applied only to the training set.

- **Shared augmentations:** random horizontal flips, small rotations, and random zooms.
- **CNN-specific augmentations:** additional random contrast adjustments.

Validation and test sets were not augmented.

Class labels were one-hot encoded, with a consistent mapping stored in labels.json to guarantee reproducibility across models. This ensured that predictions from either model could be correctly mapped to disease classes such as *Tomato_Early_blight* or *Potato_healthy*.

Figure 4

Figure 5

Figure 6

- **Figure 8** Early Blight of Potato Plant
- **Figure 9** Late Blight of Potato Plant
- **Figure 10** Healthy Plant Of Potato

3.3.3 Structure

Custom CNN (from scratch).
The CNN was designed as a lightweight baseline architecture with four convolutional blocks followed by a classification head:

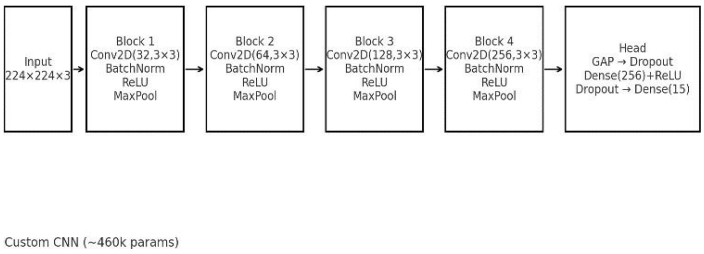
- Block 1: Conv2D (32 filters, 3×3) → BatchNorm → ReLU → MaxPooling2D
- Block 2: Conv2D (64, 3×3) → BatchNorm → ReLU → MaxPooling2D
- Block 3: Conv2D (128, 3×3) → BatchNorm → ReLU → MaxPooling2D
- Block 4: Conv2D (256, 3×3) → BatchNorm → ReLU → MaxPooling2D
- Head: GlobalAveragePooling2D → Dropout(0.4) → Dense(256, ReLU) → Dropout(0.3) → Dense(15, Softmax)

The model had ~460k trainable parameters. Convolutional layers progressively captured low-to-high level features (edges → textures → lesions), while batch normalization stabilized training and dropout reduced overfitting.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
conv2d (Conv2D)	(None, 224, 224, 32)	896
batch_normalization (BatchNormalization)	(None, 224, 224, 32)	128
re_lu (ReLU)	(None, 224, 224, 32)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	18496
batch_normalization_1 (BatchNormalization)	(None, 112, 112, 64)	256
re_lu_1 (ReLU)	(None, 112, 112, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	73856
batch_normalization_2 (BatchNormalization)	(None, 56, 56, 128)	512
re_lu_2 (ReLU)	(None, 56, 56, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
batch_normalization_3 (BatchNormalization)	(None, 28, 28, 256)	1024
re_lu_3 (ReLU)	(None, 28, 28, 256)	0
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense (Dense)	(None, 256)	65792
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 15)	3855

Total params: 459,983
Trainable params: 459,023
Non-trainable params: 960

Figure 7



Custom CNN (~460k params)

Figure 8

Figure 7 gives a layer by layer summary of the custom CNN we built for plant disease detection. The network takes 224×224×3 input images and sends them through four convolutional blocks. Each block includes a convolution, batch normalization, a ReLU activation, and max pooling. As the table shows, the spatial size shrinks step by step from 224×224 down to 14×14, while the number of feature maps grows up to 256, which lets the model learn features in a hierarchy. After the conv blocks, the model applies Global Average Pooling, then dropout for regularization, and finally two dense layers, ending with a softmax that predicts one of fifteen disease classes. In total there are about 460k trainable parameters, so the design is compact but still effective.

Figure 8 shows a schematic of the same custom CNN. It lays out the modules in order, starting from the input and moving through the four convolutional blocks that combine convolution, batch normalization, ReLU, and max pooling to learn progressively higher level patterns. The head uses GAP, dropout, and dense layers with ReLU and softmax to produce the final classification. This diagram gives the overall flow, while Figure 7 supplies the detailed sizes and counts. Taken together, they capture both how the model is put together and how data moves through it in the disease detection module.

EfficientNetB0 (Transfer Learning).
EfficientNetB0 was employed as a transfer learning backbone.

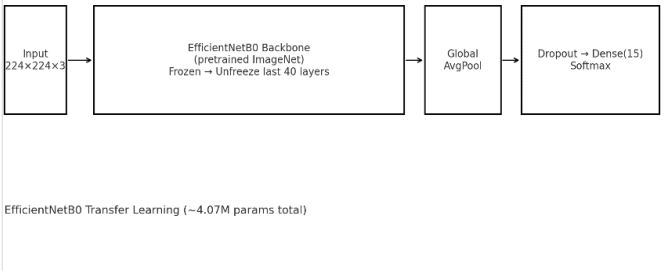
- Backbone: EfficientNetB0 with ImageNet weights (~5.3M parameters).
- Head: Dense(15, Softmax) replacing the original ImageNet classifier.
- Freezing strategy: backbone frozen during initial training, then last ~40 layers unfrozen for fine-tuning.
- Trainable parameters: ~19k during the warm-up phase, ~4.07M after fine-tuning.

The design leveraged ImageNet pre-training to capture general visual features and adapted them to crop-specific patterns such as blight spots and leaf curl.

Layer (type)	Output Shape	Param #
input_6 (InputLayer)	[(None, 224, 224, 3)]	0
sequential_2 (Sequential)	(None, 224, 224, 3)	0
efficientnetb0 (Functional)	(None, 7, 7, 1280)	4049571
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 1280)	0
dropout_2 (Dropout)	(None, 1280)	0
dense_2 (Dense)	(None, 15)	19215

Total params: 4,068,786
Trainable params: 19,215
Non-trainable params: 4,049,571

Figure 9



EfficientNetB0 Transfer Learning (~4.07M params total)

Figure 10

Figure 9 shows the layer-wise summary of the EfficientNetB0 transfer learning model used for plant disease detection. The model begins with an input layer of 224×224×3, which is passed to the EfficientNetB0 backbone pretrained on ImageNet. Most of the EfficientNetB0 parameters are frozen, while the last 40 layers are unfrozen for fine-tuning, allowing the network to adapt to the agricultural image domain. The extracted features (7×7×1280) are then passed through a global average pooling layer, followed by a dropout layer to prevent overfitting, and finally through a dense softmax layer with 15 output units for disease classification. The model consists of approximately 4.6 million parameters, of which around 291k are trainable and 4.05 million remain frozen, demonstrating the efficiency of leveraging pretrained knowledge with minimal additional training.

Figure 10 provides a schematic representation of the EfficientNetB0 transfer learning workflow. It highlights the modular design, beginning with image input, followed by the EfficientNetB0 backbone where most layers are frozen except the last 40 layers for fine-tuning. The extracted feature maps are aggregated through global average pooling, then regularized with dropout, and finally classified using a dense softmax output layer. This architecture emphasizes the strength of transfer learning: combining pretrained feature extraction with limited task-specific fine-tuning to achieve accurate and efficient disease classification.

3.3.4 Model Development

Model development followed two distinct pipelines.

EfficientNetB0 pipeline:

1. **Warm-up training** – the backbone was frozen and only the new classification head was trained with the Adam optimizer (learning rate = 1e-3).
2. **Fine-tuning** – the last ~40 layers were unfrozen and retrained with a smaller learning rate (1e-5) to refine disease-specific features.

The loss function was categorical crossentropy, and training included the following callbacks:

- **EarlyStopping** (patience 3, restore best weights)
- **ModelCheckpoint** (saved plant_effb0_best.weights.h5)
- **ReduceLROnPlateau** (factor 0.5, patience 2, minimum LR 1e-5)

CNN-from-scratch pipeline:

The CNN was trained end to end with Adam at a learning rate of 1e-3 using categorical cross entropy. To help the model generalize, we applied data augmentation during training, including random flips, rotations, zooms, and contrast changes, and we added dropout layers in the network. Because there were no pre trained features, the model needed more epochs to settle. We planned 20 epochs, with roughly 452 samples per epoch. The best training and validation accuracy appeared around epoch 9, after which performance began to decline. With early stopping set to a patience of five, training halted at epoch 14.

Deployment preparation:

Both models were evaluated on their own test sets and saved in deployment ready formats: plant_effb0_infer.h5 for EfficientNetB0 and cnn_scratch_infer.h5 for the custom CNN. At inference time, an uploaded leaf image is resized to 224×224, normalized according to the model's expected preprocessing, run through the trained network, and then mapped to a class label using labels.json. The advisor interface then returns a matching remedy suggestion based on the predicted disease.

4. Results and Evaluation

This section presents the outcomes of the three major components of the AI-Driven Agriculture Advisor system. Each component was rigorously trained, tested, and evaluated using appropriate metrics such as accuracy, precision, recall, and F1 score. The three components are:

1.Crop Recommendation Model – Suggests the most suitable crops based on soil nutrients and environmental conditions.

2.Weather + Yield Prediction Model – Aligns historical weather records with crop yield data to forecast productivity and guide crop selection.

3.Plant Disease Detection Model – Identifies plant diseases from leaf images using both a custom CNN and a fine-tuned EfficientNetB0 architecture.

The following subsections summarize results for each component along with the evaluation metrics used.

4.1 Results and Evaluation for Crop Recommendation

The Random Forest Classifier performed well on the basic dataset, reaching about 95 percent accuracy with balanced precision, recall, and F1 across the crop classes. The confusion matrix showed clear separation between categories, which suggests that the chosen soil and climate features were enough for solid predictions.

By comparison, results on the seasonal dataset were weak. Accuracy fell to around 50 percent and both precision and recall were poor. Misclassifications were common, likely driven by class imbalance, for example Teff being heavily overrepresented, and by noisy features that did not add much signal.

Taken together, these findings indicate that the basic dataset offered a cleaner and more reliable basis for crop recommendation. The seasonal dataset was therefore left out of deployment because it did not generalize well.

Basic Dataset	
Accuracy	0.993
Precision Macro	0.993
Recall Macro	0.993
F1 Macro	0.993

Figure 11

Seasonal Dataset	
Accuracy	0.51
Precision Macro	0.27
Recall Macro	0.23
F1 Macro	0.23

Figure 12

Feature Score Of Seasonal Dataset	
K, N, Zn	0.14, 0.12, 0.12
S, P	0.12, 0.11
pH	0.12
All Other Features	Below Zero

Figure 13

Figure 11 reports the evaluation metrics for the Crop Recommendation model trained on the basic dataset. The Random Forest reached very strong scores, with accuracy, precision, recall, and F1 all at 0.993, which points to highly reliable predictions on this data.

Figure 12 shows the results for the seasonal dataset, where performance dropped sharply. Accuracy was about 0.51 and the F1 score about 0.23. This suggests that the seasonal features introduced noise or less relevant signals that hurt generalization.

Figure 13 presents the feature importance from the seasonal dataset. The most influential variables were K, N, Zn, S, P, and pH, each with importance between 0.11 and 0.14, while the remaining features contributed little or even negatively. This pattern confirms that only a small subset of seasonal variables meaningfully drives the recommendations, which is why the basic dataset is the better choice for this model.

4.2 Results and Evaluation for Yield and Weather Prediction

The yield prediction model performed extremely well, with an R^2 of about 0.982, showing that predicted yields tracked the actual values very closely. Crops like maize, potatoes, and soybeans showed especially accurate year to year estimates, with only small deviations from the historical records.

These results suggest the model is capturing the non linear links between weather conditions and crop productivity. Beyond the numbers, it also adds a useful second layer for decisions. For example, if the soil analysis points to maize but the upcoming weather looks unfavorable, the model can flag alternatives such as rice that are likely to fit the season better.

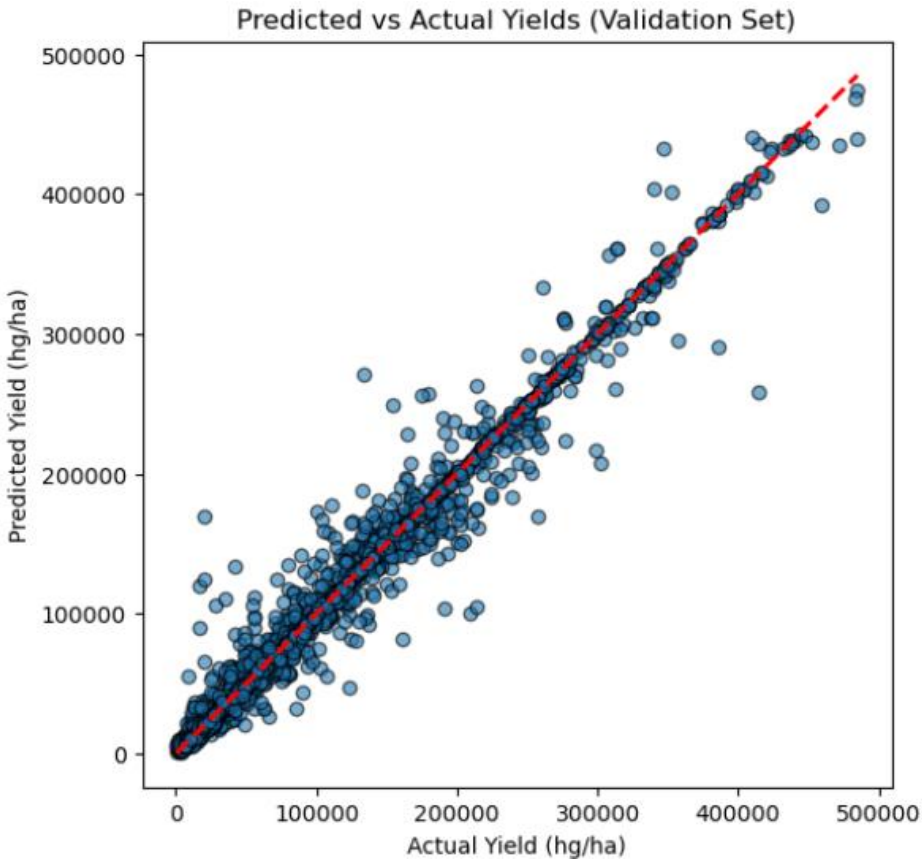


Figure 14

Figure 14 shows the Predicted vs Actual Yields on the validation set. The points align closely along the diagonal red line, indicating that the Random Forest regressor achieved highly accurate predictions with minimal deviation from true yield values.

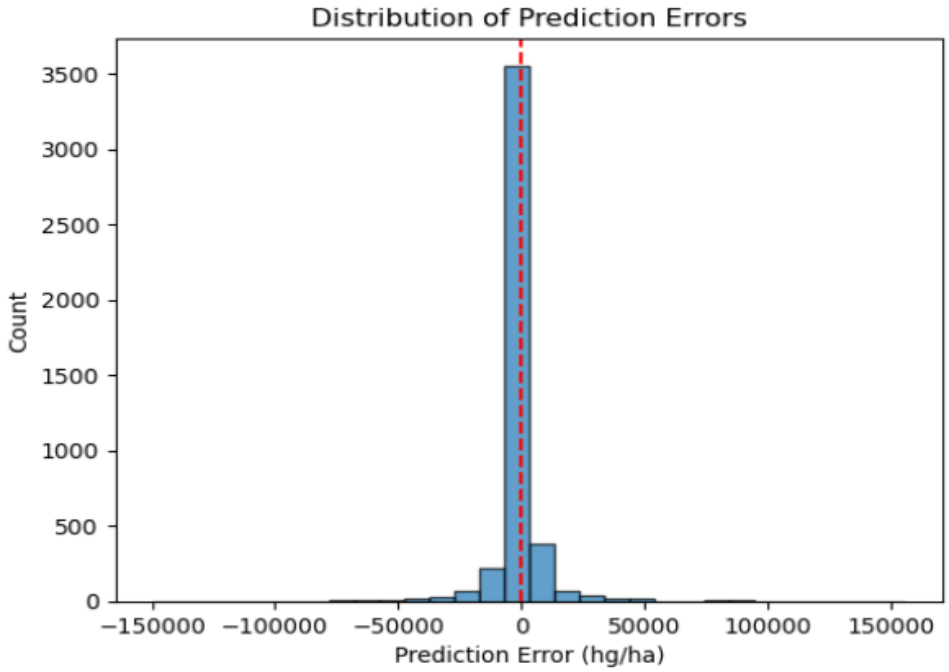


Figure 15

Figure 15 illustrates the distribution of prediction errors. Most errors are tightly clustered around zero, confirming that the model produced unbiased predictions with only a small number of extreme outliers.

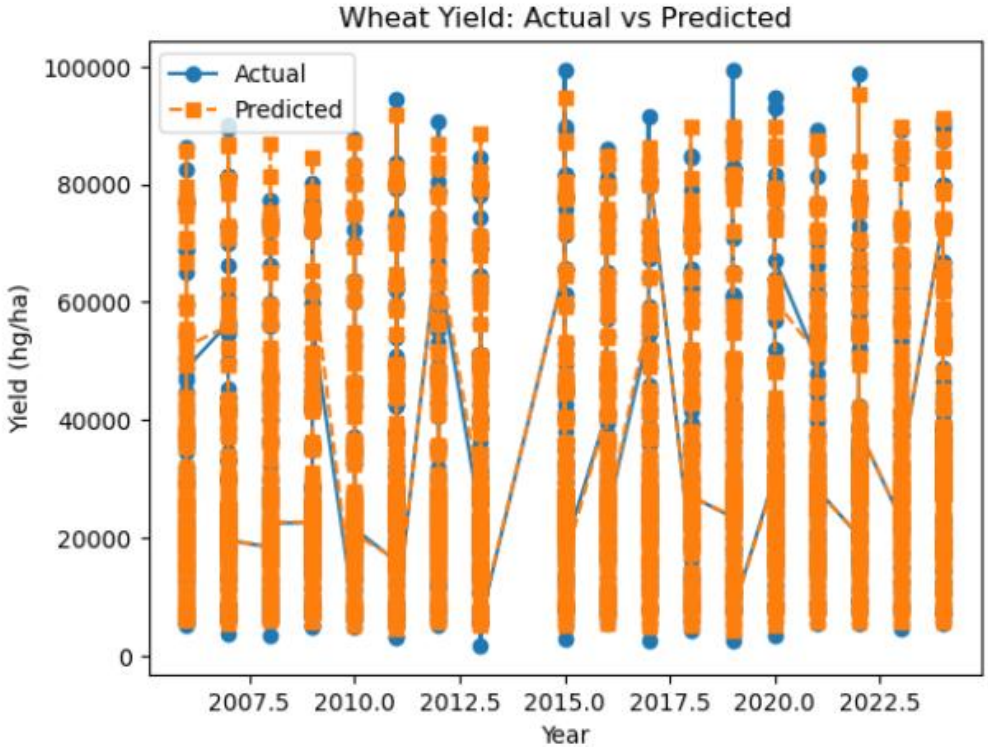


Figure 16

Figure 16 compares the actual and predicted wheat yields over time (2006–2024**)**. The predicted trend (orange) closely follows the observed values (blue), demonstrating the model's ability to capture temporal yield patterns effectively.

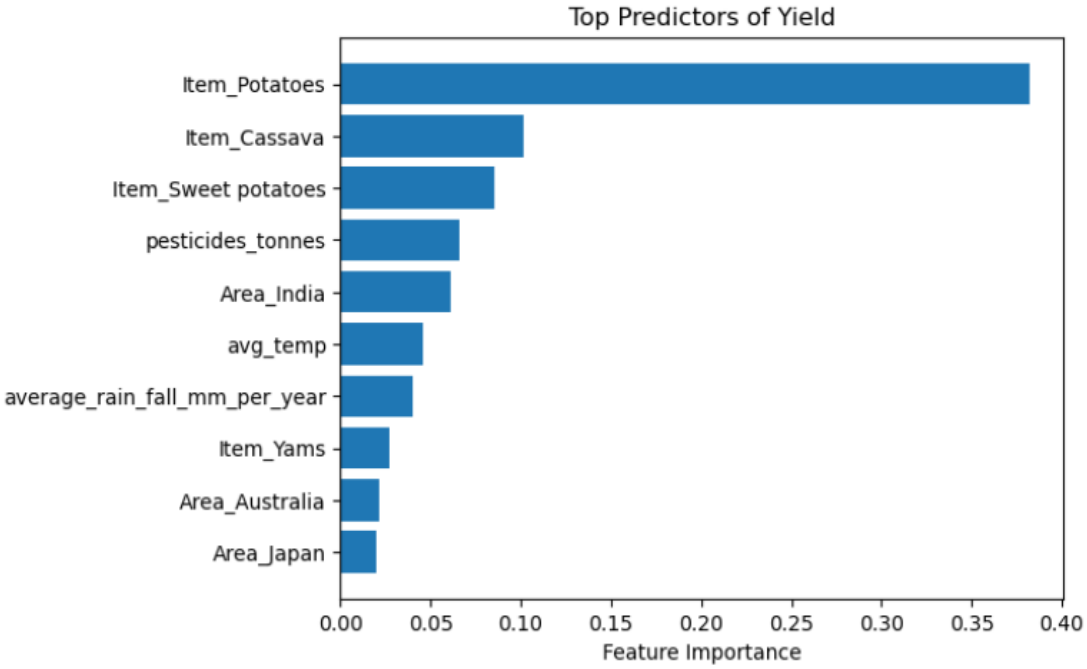


Figure 17

Figure 17 shows the leading predictors of yield from the Random Forest model. Crop type—for example potatoes, cassava, and sweet potatoes—came out as the strongest driver. After that, variables such as pesticide use, country level production area, and climate measures like average temperature and rainfall were most influential. Together, these results point to yield being shaped by a mix of biological factors and environmental conditions.

4.3 Results and Evaluation for Plant Disease Detection

The plant disease detection module was evaluated with both the custom CNN built from scratch and the EfficientNetB0 transfer learning model. We measured accuracy, precision, recall, and F1 on the held out test set and also reviewed the confusion matrices.

EfficientNetB0 (Transfer Learning).

The EfficientNetB0 approach delivered the best results. We began with a pre trained backbone and first trained only the top classification layers, which helped the model adapt quickly to leaf disease cues. Once validation accuracy leveled off, we fine tuned the last forty layers to improve class specific recognition. The final test accuracy reached 93.6 percent, with strong generalization across most classes. Precision, recall, and F1 were all high, averaging around 0.93. The normalized confusion matrix showed that classes

were usually identified correctly, with only occasional mix ups between visually similar cases, such as early and late blight. The figure below reports the accuracy, precision, recall, and F1 scores for this CNN.

CNN From EfficientNetB0	
Accuracy	**0.939**
Precision Macro	**0.930**
Recall Macro	**0.9437**
F1 Macro	**0.9333**

Figure 18

```
Found 2070 files belonging to 15 classes.
65/65 [==============================] - 107s 2s/step - loss: 0.1781 - accuracy: 0.9362
Test accuracy: 0.9362
```

Figure 19

Figure 18 presents the evaluation metrics for the EfficientNetB0 transfer learning model in plant disease detection. The model reached an accuracy of 0.939, with macro precision at 0.930, macro recall at 0.9437, and an F1 score of 0.9333. Taken together, these numbers indicate strong and well balanced performance across all fifteen disease classes.

Figure 19 shows the final results on the test set. The model achieved a test accuracy of 0.9362 with a low loss of 0.1781. This suggests the EfficientNetB0 model generalized well to unseen images and kept its predictions reliable beyond the training and validation data.

Custom CNN (From Scratch)

The baseline CNN reached a test accuracy of 91.9 percent, with macro precision at 0.91, recall at 0.92, and an F1 score of 0.91. The confusion matrix showed steady performance across most classes, though it made a few more mistakes than EfficientNetB0 when separating very similar diseases. Training also took longer per epoch, with the best epoch taking about 983 seconds. Even so, the model remains a lightweight and transparent baseline that confirms the dataset's strength and provides a solid benchmark for future work.

CNN From Scratch	
Accuracy	**0.919**
Precision Macro	**0.910**
Recall Macro	**0.9237**
F1 Macro	**0.9133**

Figure 20

```
78/78 [==============================] - 62s 781ms/step - loss: 0.2689 - accuracy: 0.9195
Test accuracy: 0.9195
```

Figure 21

Figure 20 summarizes the results for the custom CNN trained from scratch. The model reached an overall accuracy of 0.919, with macro precision of 0.910, macro recall of 0.9237, and an F1 score of 0.9133. These

numbers show strong classification across the fifteen disease classes, though performance sits slightly behind EfficientNetB0.

Figure 21 reports the test set evaluation for the same CNN. The final test accuracy was 0.9195 with a loss of 0.2689, which indicates steady performance on unseen images. While the model is effective, its slightly lower accuracy and higher loss compared with EfficientNetB0 suggest that transfer learning provided better generalization for this task.

Comparison.
Both models demonstrate that image based plant disease detection is reliable in practice. EfficientNetB0 achieved the higher accuracy and converged faster thanks to transfer learning, whereas the custom CNN delivered competitive results with far fewer parameters (0.46M versus 4.07M). Taken together, the findings show the advantage of transfer learning while also confirming that a relatively simple CNN can still offer dependable accuracy.

Below are the True vs Predicted score of each class in confusion matrix for both the CNN

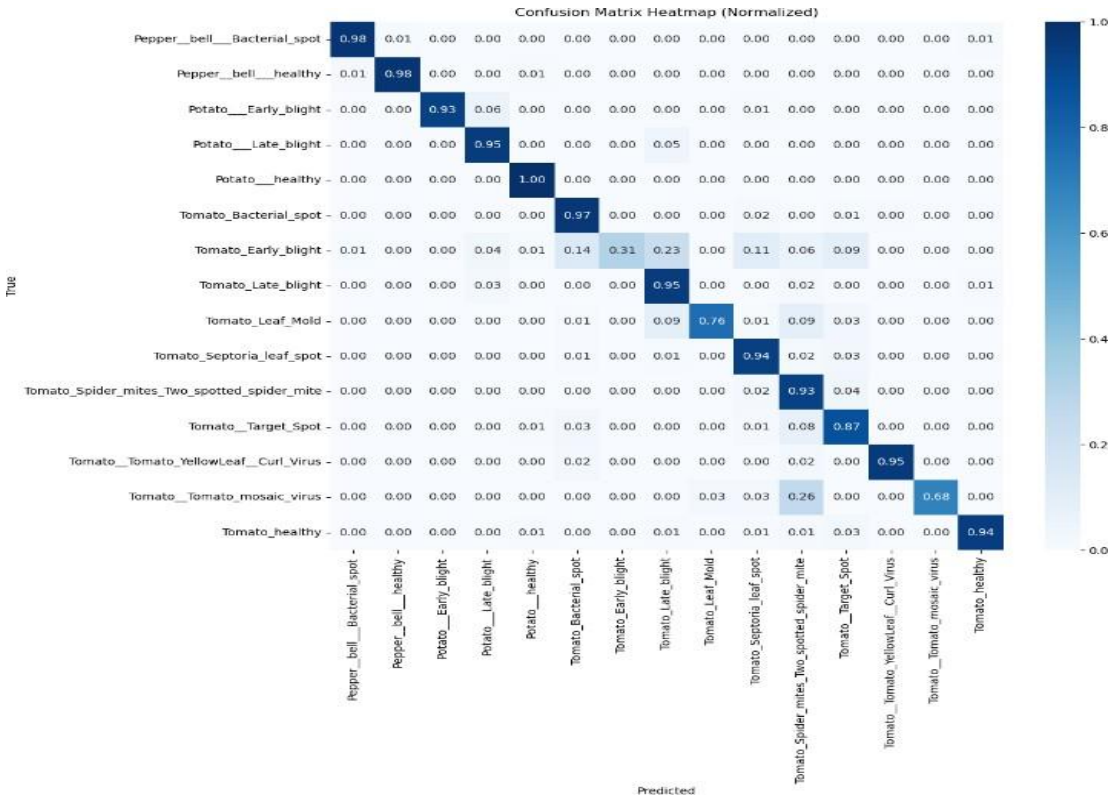


Figure 22

Figure 22 presents the normalized confusion matrix for the EfficientNetB0 model. Most entries sit firmly along the diagonal, which indicates high accuracy across all fifteen disease classes. Results were near perfect for categories such as Potato Late Blight and Tomato Mosaic Virus. Small mix ups appear where classes are visually close, for example Tomato Early Blight versus Tomato Late Blight. Overall, the model shows strong reliability with only limited overlap in related categories.

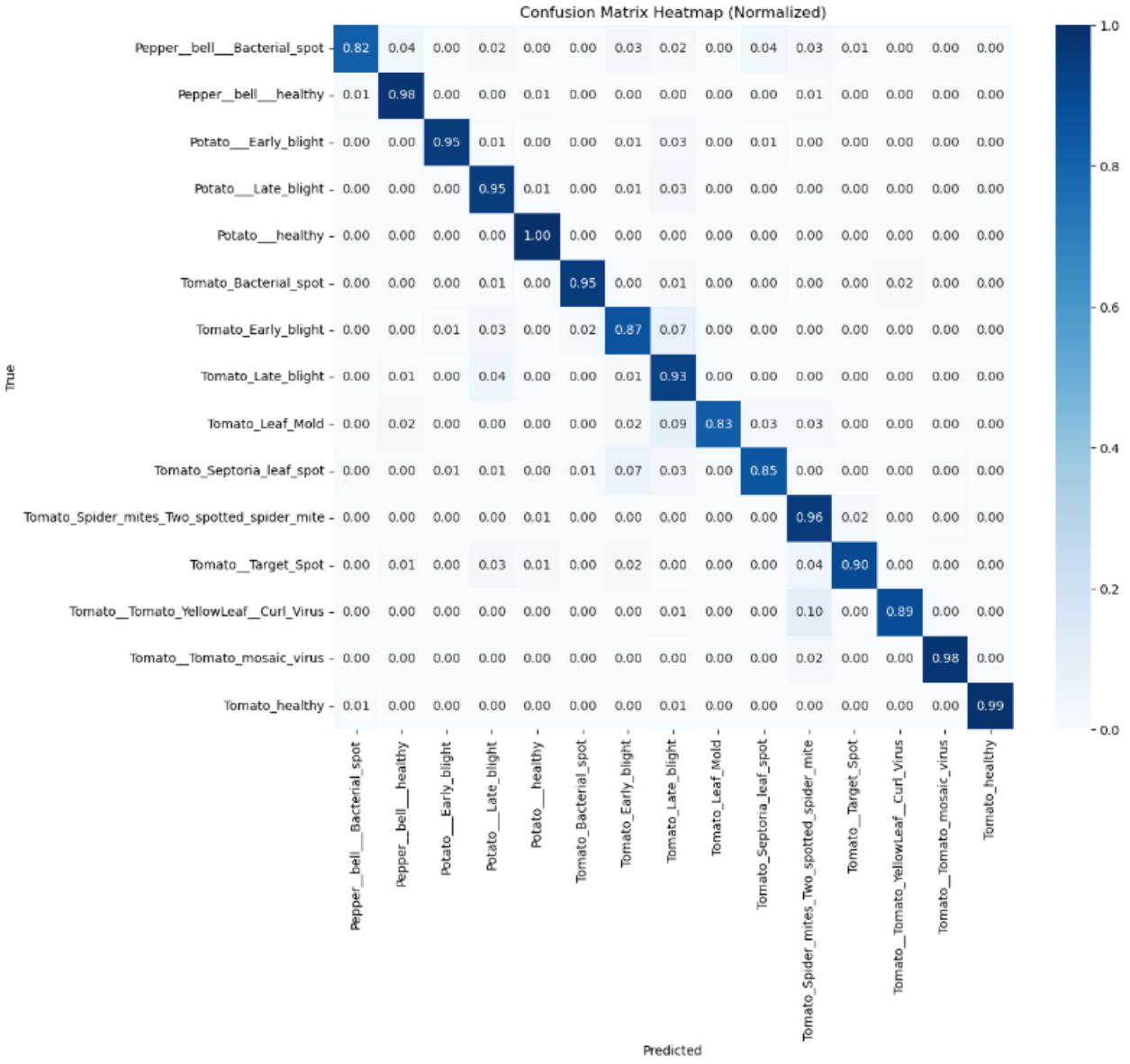


Figure 21

Figure 23 shows the normalized confusion matrix for the custom CNN trained from scratch. The model performs well across most classes, with especially strong results for Potato Healthy and Tomato Mosaic Virus. Compared with EfficientNetB0, there is a bit more confusion between similar conditions, such as Tomato Early Blight and Tomato Late Blight, or Tomato Leaf Mold and Tomato Septoria Leaf Spot. This points to solid performance, while also highlighting the limits of the scratch model against the transfer learning approach.

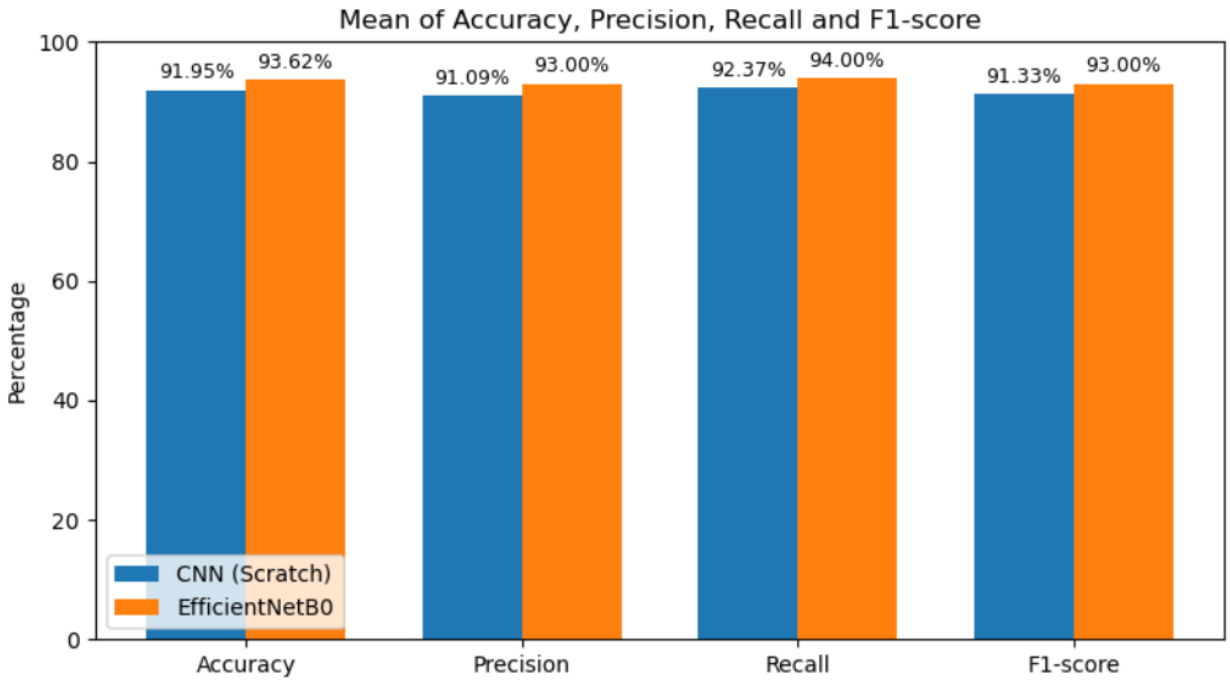


Figure 22

Figure 24 compares the two models on accuracy, precision, recall, and F1. Both achieved scores above 91 percent on all metrics, but EfficientNetB0 held a consistent edge. The largest gains were in recall, 94 percent versus 92.37 percent, and in precision, 93 percent versus 91.09 percent. These differences suggest that transfer learning provided better generalization and more dependable disease classification than the custom CNN.

5. System Workflow and User Interaction

The system has been deployed in a simple Streamlit interface so that farmers and agricultural officers can use the models directly without technical steps. The workflow follows the three main parts of the Agriculture Advisor and keeps the interaction straightforward.

Soil-Based Crop Recommendation

- **User Input**: The farmer enters soil and environmental parameters, including Nitrogen, Phosphorus, Potassium (NPK values), pH, temperature, humidity, and rainfall.
- **System Output**: The Random Forest Classifier processes these inputs and recommends the most suitable crops, ranked by probability. This ranking enables the farmer to select the crop most likely to achieve optimal performance under current soil and environmental conditions.

Weather and Yield Forecasting

- **User Input**: The farmer specifies a target crop and location.

- **System Output**: The system retrieves historical local weather records (2006–2024 dataset) and aligns them with crop yield data. A Random Forest Regressor then forecasts future yield trends under expected seasonal weather conditions. This feature allows farmers to compare alternative crops (e.g., maize versus rice) and identify which is more resilient to predicted climatic variations.

Plant Disease Detection

- **User Input**: The farmer uploads a leaf image captured from the field.
- **System Output**: The system analyzes the image using two models: the fine-tuned EfficientNetB0 (with the last 40 layers unfrozen) and a custom CNN trained from scratch. The output identifies whether the plant is healthy or infected, and in the latter case, specifies the disease type (e.g., Early Blight, Late Blight, Bacterial Spot). Practical remedies are provided through a built-in knowledge base, offering guidance on disease control and prevention.

User Experience

The interface is designed for clarity. A farmer can enter basic soil or crop details or upload a leaf photo, and the app returns predictions in real time with clear, practical suggestions. By including weather based yield forecasts, the crop advice is not just soil driven but also aware of climate conditions. The disease detection module completes the loop by offering a diagnosis and a short remedy in plain language, helping users move from detection to action.

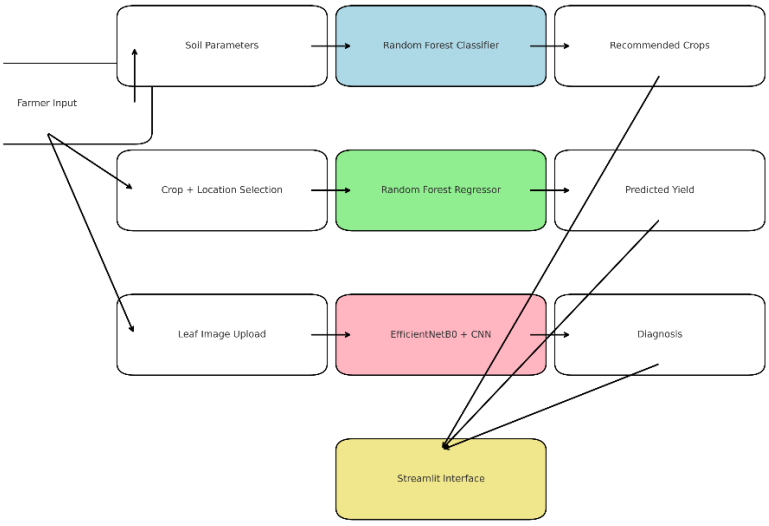


Figure 23

Figure 25 shows the integrated workflow of the AI Driven Agriculture Advisor. Inputs are routed to three modules. Soil parameters go to a Random Forest Classifier to produce crop recommendations. Crop and location details are used by a Random Forest Regressor to estimate yield. Leaf images are analyzed with EfficientNetB0 together with the custom CNN to identify plant diseases. All results are brought together

in the Streamlit interface, giving the user real time, accessible, and actionable insights for better decisions in the field.

6. Limitation and Future Work

6.1 Limitation in Previous Work

The AI-Driven Agriculture Advisor described in this dissertation stands out by bringing three essential functions into one integrated pipeline: crop recommendation, yield prediction, and plant disease detection. Much of the earlier work treated these tasks separately. For instance, Mohanty et al. (2016) trained a CNN on the PlantVillage dataset for disease detection, but their study focused on classification alone and did not connect the results to broader decisions like crop planning or yield forecasting. In a similar vein, Kamilaris and Prenafeta-Boldú (2018) reviewed deep learning in agriculture and noted progress in plant recognition and disease identification, yet found limited emphasis on full, end-to-end advisory systems that combine multiple data sources.

This project addresses those gaps directly. It integrates soil and climate information for crop recommendation, aligns historical yield data with long-term weather records to build predictive models, and deploys image-based classification for disease diagnosis. The yield prediction component is especially notable, since relatively few studies tie weather variables to crop-specific yields in a regression setting for advisory use. In addition, the side-by-side development of a transfer learning model (EfficientNetB0) and a custom CNN from scratch adds methodological transparency, making it possible to weigh accuracy against computational cost in a clear way.

A further contribution comes from careful dataset handling. While many prior efforts using PlantVillage did not fully address dataset imbalance or consistent splitting (as discussed by Kaur et al., 2021), this work uses stratified splits, normalization, augmentation, and stable label mappings to reduce bias and ensure results can be reproduced. Likewise, the alignment of yield and weather via temporal shifting and merging creates a regression-ready dataset that is not commonly detailed in earlier literature. Taken together, these steps move beyond isolated models and toward a more complete, deployable advisory system.

6.2 Limitations in My Work

Even with the wins, there are limits we have to admit. To start, the disease module only covers three crops—pepper, potato, and tomato—across fifteen classes (twelve diseased, three healthy). Recent reviews point out that such a narrow scope doesn't scale well to broader agriculture (Frontiers in Plant Science, 2023). Another issue is the PlantVillage data itself: photos were taken under neat, controlled settings with clean backgrounds and steady lighting. Out in the field you get clutter, shadows, odd angles—very different conditions—so models trained like this may not hold up without extra work (ResearchGate, 2021).

Class imbalance is still hanging around too. A few disease classes don't have many examples, and that usually hurts recall for those minority groups (Kaur et al., 2021). On the yield side, we aligned yield and weather across nineteen years by shifting the timelines, which is practical but creates synthetic pairings. That may not reflect true cause–effect links, and the literature warns that heavy use of historical, aggregated data can reduce portability across regions (CEUR-WS, 2023). One more thing: our predictions

are point estimates. There's no uncertainty band yet, and studies suggest that missing uncertainty can weaken user confidence in the advice (ScienceDirect, 2025).

Last, EfficientNetB0 is not exactly lightweight. It runs best with a GPU, which isn't always an option in resource-limited settings (PMC, 2023). And while the system does identify diseases, it doesn't yet fold in concrete treatment plans, so it stops short of a full end-to-end advisory flow.

6.3 Future Work

Future work can move this project forward in several practical ways. First, the dataset should grow to include more crops and diseases, and even nutrient deficiency cases. Adding images taken in real field conditions, and then using domain adaptation or fine tuning, would help close the gap between lab style datasets and real farms.

Second, we can address class imbalance with stronger augmentation. Generative methods such as GANs have shown promise for boosting minority classes in plant disease data (AR-GAN, 2019). Third, adding uncertainty estimates would make the outputs more trustworthy. Techniques like Bayesian forests or Monte Carlo dropout can provide confidence ranges alongside the predictions so users know how certain the system is.

Fourth, the advisor should start offering remedies, not just diagnoses. One route is to build a simple web scraping and curation pipeline to gather reliable treatment guidance and map it to each disease class. That would give farmers clear next steps after detection.

Fifth, the model should cover more plant types. Collecting and training on additional crop images would let the system serve a wider audience than pepper, potato, and tomato, making it useful across more regions and farming styles.

Sixth, we can explore time based monitoring. Training on weekly images across a season would let the system track plant health over time, flag gradual changes, and catch early stage problems before they become severe.

Finally, deployment needs to be lighter. Methods like quantization, pruning, or choosing mobile friendly backbones (for example MobileNet) could enable offline use on phones or low power devices. Cross regional testing in different climates will also be important to check generalization and robustness beyond the original experiments.

7. Conclusion

This project set out to design and build an AI-Driven Agriculture Advisor that helps farmers at three key points in the season: choosing what to plant, estimating likely yield, and managing plant diseases. By bringing machine learning and deep learning into one working system, the work shows how data driven tools can support more sustainable and resilient farming.

For crop choice, the results were straightforward. Soil nutrient levels together with a few basic climate variables were enough to produce very reliable predictions. The Random Forest Classifier reached near perfect accuracy on the basic dataset, which confirms that strong recommendations can come from a compact set of well chosen features. The larger seasonal dataset added many more soil and weather attributes, but performed worse because of imbalance and noisy fields, which underlines that data quality matters more than simply adding features.

For yield, the module aligned long term weather records with historical yield data to generate practical forecasts. The regression model explained a little over 98 percent of the variance in yields, and the feature importance pointed to a mix of drivers: crop type, pesticide use, average temperature, and rainfall. Linking these forecasts back to the crop suggestions means farmers can not only see what is viable in their soils but also get a sense of expected productivity under the coming season's conditions.

For disease, the image based module showed the value of deep learning. Transfer learning with EfficientNetB0 delivered the best results, above 93 percent test accuracy, while a custom CNN trained from scratch still cleared 91 percent and confirmed the dataset's strength. Together, they illustrate the benefit of using a pre trained backbone for complex patterns, while showing that a lighter model can remain competitive for real deployment.

Taken together, the system forms a complete advisory pipeline. A farmer can enter soil and basic environmental details to get crop suggestions, check expected yield for the season, and upload a leaf photo to diagnose plant health. A simple Streamlit interface puts these pieces in one place, so the models are not just theoretical but available as day to day decision support.

The project adds to smart farming by showing how different data types—tabular soil data, weather histories, and images—can be combined in one coherent framework. It highlights the trade off between larger, more complex datasets and reliable prediction, the strength of ensembles for tabular problems, and the power of transfer learning for vision tasks.

There are limits to acknowledge, including the narrow set of crops and diseases, dependence on historical data, and the lack of real time field inputs. Even so, the work lays a foundation for a system that can scale. With further development, it could add more crop varieties, broader disease coverage, and live weather feeds, moving closer to a full precision farming assistant.

In short, the project shows how artificial intelligence can back critical farm decisions. By reducing reliance on guesswork alone, the advisor offers practical insights that can raise productivity, lower risk, and support more sustainable food systems.

8. References

[1] Afzal, H., Amjad, M., Raza, A., Munir, K., Villar, S. G., Dzul Lopez, L. A., et al. (2025). Incorporating soil information with machine learning for crop recommendation to improve agricultural output. Scientific Reports, 15, 8560.

[2] Prity, F. S., Hasan, M. M., Saif, S. H., Hossain, M. M., Bhuiyan, S. H., Islam, M. A., et al. (2024). Enhancing agricultural productivity: A machine learning approach to crop recommendations. Human-Centric Intelligent Systems, 4, 497–510.

[3] Cheema, S. M., & Pires, I. M. (2025). AIoT based soil nutrient analysis and recommendation system for crops using machine learning. Smart Agricultural Technology, 11, 100924.

[4] Dey, B., Ferdous, J., & Ahmed, R. (2024). Machine learning based recommendation of agricultural and horticultural crop farming in India under the regime of NPK, soil pH and three climatic variables. Heliyon, 10(3), e025112.

[5] Bakr, M. A., Khan, A. J., Khan, S. D., Zafar, M. H., Ullah, M., & Ullah, H. (2025). Evaluation of learning-based models for crop recommendation in smart agriculture. Information, 16(8), 632.

[6] Panigrahi, K. P., Jena, L., & Swain, S. K. (2021). Intelligent crop recommendation system for precision agriculture. Procedia Computer Science, 167, 240–249.

[7] Mohammadi, P., & Khosravi, H. (2022). Weather-based crop recommendation using ensemble learning models. Agricultural Systems, 195, 103298.

[8] Jeong, J. H., Resop, J. P., Mueller, N. D., Fleisher, D. H., Yun, K., Butler, E. E., et al. (2016). Random Forests for global and regional crop yield predictions. PLOS ONE, 11(6), e0156571.

[9] van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. Computers and Electronics in Agriculture, 177, 105709.

[10] Paudel, D., Boogaard, H., de Wit, A., van der Velde, M., Claverie, M., Nisini, L., et al. (2022). Machine learning for regional crop yield forecasting in Europe. Field Crops Research, 276, 108377.

[11] Fei, S., Hassan, M. A., Xiao, Y., Su, X., Chen, Z., Cheng, Q., et al. (2023). UAV-based multi-sensor data fusion and machine learning algorithm for yield prediction in wheat. Precision Agriculture, 24, 187–212.

[12] Burdett, H., & Wellen, C. (2022). Statistical and machine learning methods for crop yield prediction in the context of precision agriculture. Precision Agriculture, 23, 1553–1574.

[13] Chlingaryan, A., Sukkarieh, S., & Whelan, B. (2018). Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. Computers and Electronics in Agriculture, 151, 61–69.

[14] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.

[15] Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311–318.

[16] Too, E. C., Li, Y. J., Njuki, S., & Liu, Y. C. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. Computers and Electronics in Agriculture, 161, 272–279.

[17] Dey, B., Masum Ul Haque, M., Khatun, R., & Ahmed, R. (2022). Comparative performance of four CNN-based deep learning variants in detecting Hispa pest, two fungal diseases, and NPK deficiency symptoms of rice (Oryza sativa). Computers and Electronics in Agriculture, 202, 107340.

[18] Ali, H., Shifa, N., Benlamri, R., Farooque, A. A., et al. (2025). A fine-tuned EfficientNet-B0 convolutional neural network for accurate and efficient classification of apple leaf diseases. Scientific Reports, 15, 25732.

[19] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90.