KisanSetu AI: A Review of AI-Based Farmer Assistance Methods and Design of an Integrated Decision-Support Platform

Badrinath Balasaheb Dube[1], Chaitanya Maruti Sanap[2], Aniket Nanaji Bachhav[3], Ms. Vrushali D. Mane[4]

Department of AI & Data Science Engineering
Matoshri College of Engineering and Research Centre, Nashik

Abstract
Farmers have to decide what to sow, how to protect the crop, when the weather will allow field work and where to sell, and the facts behind each choice usually sit on different portals. This paper reviews six recent studies on crop recommendation, leaf-disease detection, rainfall forecasting and commodity-price forecasting, and then describes KisanSetu AI, a web platform that brings these functions together for one farmer. The farmer enters a location, farm details and, if needed, a photo of a leaf. The platform returns a crop suggestion, a probable disease class with guidance, local weather information and market prices on a single dashboard. It is planned with Python, Django, machine-learning models and public weather and price sources. The review shows that each earlier study solves a single piece of the problem, mostly on clean benchmark data, and that no study links the pieces into something a farmer can actually use. Later sections cover the architecture, how a request flows through it, how it will be tested and where it falls short. No field results are claimed at this stage.

Keywords: Smart farming, crop recommendation, plant disease detection, weather advisory, price forecasting, machine learning, Django, decision support

I. INTRODUCTION

Farming in India still runs on judgement calls made under uncertainty. A farmer must pick a crop before the season starts, watch the weather, spot disease early and then find a fair price at the end. Each of these questions has an answer somewhere, but the answers are spread over weather bulletins, market portals, extension officers and word of mouth. A farmer with a small holding and a basic phone rarely has the time to collect them all.

Machine learning has been applied to every one of these questions. Models can suggest a crop from soil and climate readings [1], [4], [5], recognise a diseased leaf from a photograph [2], [3] and forecast the price of a vegetable in the coming weeks [6]. Official services also exist, for example the agro-meteorological advisories of the India Meteorological Department [7] and the e-NAM market platform [8]. What is missing is a single place where a farmer can use these functions together.

Our work has two parts. We first go through six studies [1]–[6], noting what each leaves open. We then lay out the design of KisanSetu AI, a web-based farmer assistance platform with four modules: crop recommendation, crop disease assistance, weather assistance and market information. The web layer is built with the Django framework [9]. The aim is to support a farmer's own decision, not to replace an agricultural expert.

Section II covers the literature and the gaps in it. Section III explains how the platform is put together. Section IV sets out how it will be tested, what we expect from it and where it is weak, and Section V closes with a short summary and the next steps.

II. LITERATURE REVIEW

The six studies fall into four groups, which are crop recommendation, leaf-disease detection, forecasting of rainfall and prices, and the gaps that remain across all of them.

A. Crop Recommendation from Soil and Climate Data

Dey et al. [1] tested five models, namely support vector machine, XGBoost, random forest, k-nearest neighbours and decision tree, on a public Kaggle dataset. The inputs were the three soil nutrients N, P and K, soil pH, and three weather readings. They trained separate models for eleven field crops and ten horticultural crops and then one for both groups together. XGBoost came out best, with a precision of 99.09% for field crops, 99.3% for horticultural crops and 98.51% for the joint set. The authors conclude that a model for each crop group works better than one model for everything.

Dhingra et al. [4] built KRISHI.AI on the same seven inputs. A LightGBM classifier handles 22 crop classes and reaches 99.39% test accuracy. The new part is explanation: SHAP charts show why a crop was chosen, and a "what-if" module shows how a small change in the inputs would change the advice. Across 100 test cases about 94.6% of the what-if scenarios were valid, and the whole pipeline answers in under 200 ms inside a Streamlit dashboard. The authors themselves warn that the dataset is balanced and clean, so the score is a benchmark figure and not proof of field reliability.

Jithendra et al. [5] used two stages. A neuro-fuzzy network (ANFIS) tuned with the Puma optimizer first predicts rainfall, and an SVM then picks the crop from soil values and the predicted rainfall. Rainfall prediction reached 99.15% accuracy, and crop classification reached 90%, but only four crops (rice, maize, watermelon, coffee) and just 48 samples were used. The authors admit that the data is small and that the method is hard to interpret.

B. Leaf Disease Detection from Images

Mohanty et al. [2] trained AlexNet and GoogLeNet on 54,306 photographs of healthy and diseased leaves. The pictures came from fourteen crops and twenty-six diseases, which gave 38 classes. The best set-up, a GoogLeNet pre-trained and fine-tuned on colour images, reached 99.35% on a held-out part of the same collection. The more important finding comes next. When the same model was tried on pictures taken under different conditions, its accuracy fell to about 31%. The study shows both the promise of smartphone-based diagnosis and the danger of training only on lab-style pictures.

Begum et al. [3] went for a lighter pipeline. A pre-trained ResNet50 turns each leaf image into features, PCA shrinks the features, and an SVM assigns the class. On the 38-class PlantVillage set the model gave an mean accuracy of 98.63% under five-fold cross-validation. The authors stress that the pictures were taken in controlled conditions and do not claim that the model is ready for real fields. They did add tests with changed lighting and added noise, which is a useful step towards field use.

C. Price Forecasting for Farm Produce

Manogna et al. [6] compared classical, machine-learning and deep-learning forecasters on daily wholesale prices of 23 commodities from the AGMARKNET portal. The records span 165 markets between 2010 and mid-2024. The models were ARIMA, SVR, XGBoost, a multilayer perceptron, RNN, LSTM, GRU and an echo state network. Recurrent models did best. For onion the GRU gave an RMSE of 369.54, against 1564.62 for ARIMA, and for tomato the figures were 210.35 and 1298.60. The paper suggests that weather data should be added in future, and it also notes that deep models need more computing and data preparation.

D. Summary of Gaps

Table I sets the six studies next to the proposed platform. A "No" says the paper does not report that feature, "Part" says it appears only in a limited form, and "Planned" marks something KisanSetu AI aims to deliver but must still prove. The table shows three things. First, every study handles one slice of farm decision-making, and none connects crop advice, disease checks, weather and prices. Second, only [4] gives the farmer a working interface, and even that covers crop advice alone. Third, almost all results come from clean benchmark data. The one study that tried a different kind of picture [2] saw a sharp drop.

TABLE I
REVIEWED STUDIES AGAINST KISANSETU AI

Feature	[1]	[2]	[3]	[4]	[5]	[6]	Proposed
Crop recommendation	Yes	No	No	Yes	Yes	No	Yes
Leaf disease detection	No	Yes	Yes	No	No	No	Yes
Weather / rainfall use	Part	No	No	Part	Yes	No	Yes
Market price data	No	No	No	No	No	Yes	Yes
Explains its result	No	No	No	Yes	Part	No	Part
Farmer-facing interface	No	No	No	Yes	No	No	Yes
All functions in one platform	No	No	No	No	No	No	Yes
Measured results	Yes	Yes	Yes	Yes	Yes	Yes	Planned
Tested on field-style data	No	Part	No	No	No	Part	Planned

III. PROPOSED METHODOLOGY

KisanSetu AI is a web platform that takes a few inputs from the farmer, calls the right model or data service and shows the answers together. The block diagram is in Fig. 1. The farmer works through a browser or a phone. The Django application handles login, forms and the links to the four modules, and a database holds the farmer profile and the history of advice. Results from all modules end up on one dashboard. Fig. 2 traces one request from start to finish.

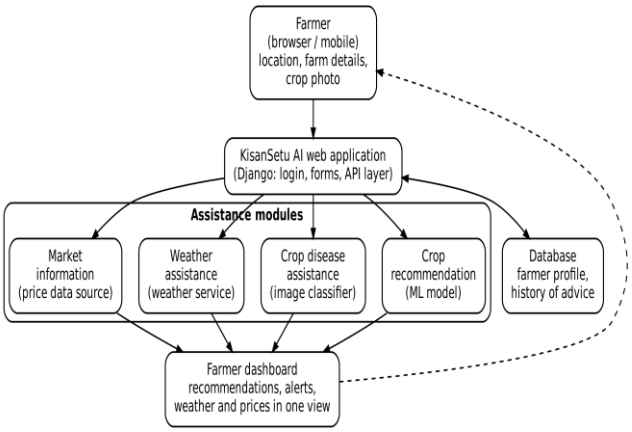


Fig. 1. Architecture of the KisanSetu AI platform.

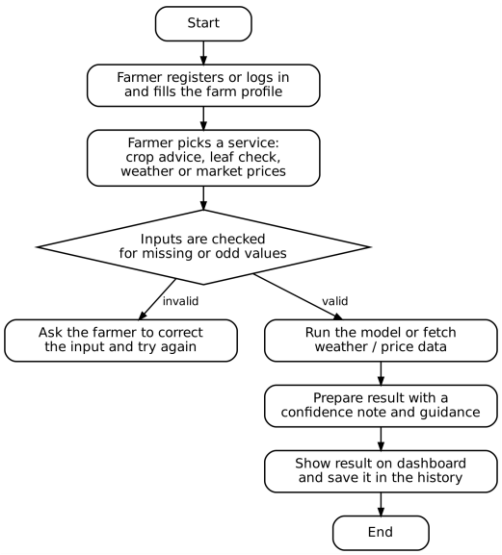


Fig. 2. Flow of one farmer request through the platform.

A. Data Collection and Preparation

Three kinds of data are needed. Crop recommendation needs a table of soil and climate readings labelled with the crop, such as the public dataset used in [1], [4] and [5]. Disease assistance needs labelled leaf photographs, and PlantVillage as used in [2] and [3] is the obvious start. Weather and price information will come from public services and are not used for training. Missing and inconsistent rows are cleaned, categories are encoded and numbers are scaled. Leaf images are resized and augmented by flipping, rotating and changing brightness. To address the weakness seen in [2], some photographs taken in real fields will be collected and kept aside for testing only.

B. Crop Recommendation Module

The module takes the soil nutrient values, pH and weather readings, or fills the weather values from the weather service for the farmer's location. Several classifiers will be compared, among them random forest, XGBoost and LightGBM, as the studies in [1] and [4] suggest. The one with the best validation score and a small size will be kept. Following [1], crops may be grouped into field and horticultural sets with a model for each. The reply lists the top few crops with their probabilities instead of a single answer, so the farmer can see how sure the model is.

C. Crop Disease Assistance Module

The farmer uploads a photo of a leaf. The image is resized and passed to a convolutional network that is pre-trained and fine-tuned by transfer learning, as in [2]. A lighter alternative is ResNet50 features with PCA and an SVM [3], which needs less computing and may suit a modest server. The module returns the most likely disease class, a confidence value and short care guidance. When the confidence is low, it asks for a clearer photo and suggests contacting an agricultural officer, because a wrong diagnosis is worse than no diagnosis.

D. Weather and Market Information

The weather module fetches the forecast for the farmer's location from a public weather service and turns it into plain advice, such as whether the next days are suitable for spraying or irrigation. The market module shows commodity prices from a reliable public source when one is available, with the market name and date next to each price. Price forecasting with recurrent networks [6] is kept as a later extension, because it needs long price histories for each commodity and market.

E. Backend, Database and Security

Django provides user accounts, forms, the dashboard pages and the API through which the models are called [9]. Each farmer has a profile with village, land size and crops grown. Every request and its result are stored,

so that the dashboard can show earlier advice. SQLite is enough while the platform is being built, and a larger database can replace it at deployment. Django hashes stored passwords and ships with defences against forged form submissions, and the deployed site should use HTTPS. Bootstrap keeps the pages readable on a small phone. Table II lists the tools.

TABLE II
TOOLS AND TECHNOLOGIES FOR THE PLATFORM

Component	Function
Python	Main language for data and models
scikit-learn, XGBoost / LightGBM	Crop recommendation models
TensorFlow or PyTorch	Leaf disease image classifier
OpenCV / Pillow	Image resizing and augmentation
Django	Backend, login, forms, APIs
SQLite (development)	Farmer profiles, history of advice
Weather and price services	Live weather and market data
Bootstrap	Web pages for phone and PC

F. Working Sequence

The farmer registers or logs in and fills the farm profile once. After that, he or she chooses a service on the dashboard. The platform checks the inputs for missing or odd values and asks for a correction if any are found. A valid request goes to the model or the data service. The result is prepared together with a confidence note and short guidance, put on the dashboard and written to the history. Every result is worded as an aid to decision-making and never as a guarantee.

IV. RESULTS AND DISCUSSION

What we present here is a design and a test plan. The platform has not been run on real farm data, so we quote no accuracy for it. The numbers quoted in Section II belong to the cited studies and were obtained on their own datasets. The rest of this section says what the design should achieve and how we will judge it.

We first expect a crop model that stays reliable on unseen data. It will be scored with accuracy, precision, recall and F1 on a held-out split, and then on a small set of records from local farms. The second is a disease module that works on ordinary phone photographs and not only on clean laboratory pictures. For this, the field photographs kept aside in Section III will be used, and the drop in accuracy from the clean set to the field set will be reported openly. The third is that the weather and price figures on the dashboard match their sources for the same place and date. The fourth is a platform that answers quickly and does not break when a farmer leaves a field empty or types a wrong value. Table III lists the planned tests and their targets. The targets are engineering goals for a pilot and are not measured results.

TABLE III
PLANNED TESTS AND CRITERIA

Test	What it checks	Criterion
Crop model	Accuracy and F1 on held-out records	F1 of at least 0.90
Disease model	Accuracy on field-style photos	At least 80% correct
Weather module	Shown values against the source	Match in all trials
Market module	Shown prices against the source	Match in all trials
Response time	Request sent to result shown	Under 5 s
Invalid input	Empty or wrong values	No crash in any trial

Set against the reviewed work, the proposed platform adds integration. The studies in [1], [4] and [5] stop at crop advice, [2] and [3] stop at leaf classification and [6] stops at price forecasting. KisanSetu AI joins these functions behind one login and one dashboard, using open-source tools on ordinary hardware. It does not try to beat the accuracy reported by any single study, and it does not offer the explanation tools of [4] in its first version. A simple note on which inputs mattered most may be added later.

Several limits should be stated plainly. First, advice depends on the quality and freshness of the training data and of the outside services, and a dataset that is clean and balanced, as in [1] and [4], does not show how the model will behave on noisy farm records. Second, leaf models trained on laboratory images may fail on field photos [2], [3], so the first version covers only a limited set of crops and diseases. Third, weather and price services may be slow, incomplete or unavailable for some places. Fourth, a farmer with poor connectivity or limited literacy may find a web page hard to use, which is why language support and voice help are left for later. Fifth, the output is an aid and not a substitute for an agricultural expert, and the screens say so clearly.

V. CONCLUSION

This work reviewed six studies on crop recommendation, leaf-disease detection, rainfall forecasting and price forecasting, and used them to shape KisanSetu AI, a web-based assistance platform for farmers. The review found that machine-learning models do well in each of these areas on benchmark data [1]–[6], that explanation [4] and field-style testing [2], [3] are still thin, and that no study brings the functions together for the farmer. The proposed platform does that with Python, Django, machine-learning models and public weather and price sources. Future work will build the working prototype, carry out the Table III tests on real and field-collected data, add Marathi and Hindi screens and voice help, widen the list of crops and diseases, add price forecasting and sensor or satellite inputs, and test the platform with a small group of farmers.

REFERENCES

[1] B. Dey, J. Ferdous and R. Ahmed, “Machine learning based recommendation of agricultural and horticultural crop farming in India under the regime of NPK, soil pH and three climatic variables,” Heliyon, vol. 10, Art. no. e25112, 2024, doi: 10.1016/j.heliyon.2024.e25112.

[2] S. P. Mohanty, D. P. Hughes and M. Salathé, “Using deep learning for image-based plant disease detection,” Front. Plant Sci., vol. 7, Art. no. 1419, Sep. 2016, doi: 10.3389/fpls.2016.01419.

[3] S. Begum, E. Naresh and N. N. Srinidhi, “A hybrid deep learning model for robust and efficient plant leaf disease detection using ResNet50, PCA, and SVM,” Sci. Rep., vol. 16, Art. no. 15805, 2026, doi: 10.1038/s41598-026-46085-w.

[4] B. Dhingra, S. Agarwal and R. Gupta, “KRISHI.AI: An explainable machine learning framework for data-driven crop recommendation,” Sci. Rep., 2026, doi: 10.1038/s41598-026-68815-w.

[5] T. Jithendra, A. Divya, I. Aljubayri, T. Haritha and M. Z. Khan, “Two-stage rainfall forecasting and crop classification using puma-optimized ANFIS–SVM,” Smart Agric. Technol., vol. 13, Art. no. 101719, 2026, doi: 10.1016/j.atech.2025.101719.

[6] R. L. Manogna, V. Dharmaji and S. Sarang, "Enhancing agricultural commodity price forecasting with deep learning," Sci. Rep., vol. 15, Art. no. 20903, 2025, doi: 10.1038/s41598-025-05103-z.
[7] India Meteorological Department, "Agrometeorological Advisory Services," Ministry of Earth Sciences, Government of India. [Online]. Available: https://mausam.imd.gov.in
[8] National Agriculture Market (e-NAM), Department of Agriculture and Farmers' Welfare, Government of India. [Online]. Available: https://enam.gov.in
[9] Django Software Foundation, "Django documentation." [Online]. Available: https://docs.djangoproject.com/