INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 7, July 2026 | ISSN: 2456-4184 | IJNRD.ORG

AI-Driven Crop Recommendation System: An Ensemble Machine Learning Framework with Real-Time API Integration and Multi-Lingual Accessibility

Ms. Shivjyoti D. Patil[1], Mr. Abhijit S. Shinde[2]

[1]Student, [2]Assistant Professor

[1,2] Department of Computer Science and Engineering

[1,2] D. Y. Patil Agriculture & Technical University, Talsande, Maharashtra, India

Abstract: Crop selection remains a high-stakes decision for smallholder farmers in India, with poor choices leading to significant income losses annually. This paper presents the design, implementation, and evaluation of an AI-Driven Crop Recommendation System, a production-ready web application that delivers ensemble machine learning recommendations alongside real-time weather data, live market intelligence, ICAR-calibrated fertilizer advice, and regional language support. The system combines a Random Forest (97.05% accuracy), Support Vector Machine with RBF kernel (96.59%), and Gaussian Naive Bayes (90.23%) through majority voting, achieving an ensemble accuracy of 98.18% on a 22-class crop dataset of 2,200 records. A two-tier geolocation pipeline auto-populates temperature and humidity from the OpenWeather API. A fertilizer engine computes exact bag counts for Urea, DAP, and MOP calibrated to ICAR nutrient rates and user-specified farm area. A rebuilt multilingual pipeline supports 20-plus Indian regional languages via a novel Google Translate cookie-injection mechanism. Live commodity prices from the Agmarknet API cover all 36 states and union territories of India. Evaluation on a held-out test set and a 20-participant user acceptance study confirm all stated objectives. User questionnaire scores averaged 4.6/5.0 for perceived usefulness and 4.7/5.0 for output quality.

Keywords: *Crop recommendation, ensemble learning, Random Forest, SVM, precision agriculture, Flask, real-time API, fertilizer recommendation, multilingual web application*

Introduction

Agriculture accounts for approximately 18 percent of India's GDP and employs over half the workforce. Despite this, most smallholder farmers make planting decisions based on tradition and intuition rather than data. A poor crop choice, driven by mismatched soil chemistry, climate conditions, or market saturation, can eliminate an entire season's income for families with little financial resilience. The problem is not a lack of data: soil test results, mandi prices, and weather observations are all publicly available, but the absence of an integrated, accessible tool that converts this data into farm-level action.

Existing digital advisory tools address parts of this problem but not all of it. Most rely on static datasets, cover only a subset of states for market data, operate in English or Hindi only, and produce qualitative nutrient advice without the procurement quantities farmers actually need. This paper describes a system designed to address all of these gaps simultaneously: accurate ensemble ML prediction, automated weather integration, ICAR-calibrated fertilizer bag counts, all-state live market prices, and multilingual support across 20-plus regional languages.

The key contributions of this work are: (1) a three-model ensemble classifier achieving 98.18% accuracy on a 22-class crop recommendation task, outperforming all comparable single-model published systems; (2) a novel hybrid Google Translate cookie-injection mechanism for state-preserving language switching in CSP-enforced hosting environments; (3) a two-tier geolocation pipeline for automated real-time weather retrieval achieving 95%+ auto-fill success; and (4) an ICAR-calibrated fertilizer bag count engine validated against manual calculations with under 0.2-bag maximum error.

Related Work

Pudumalar et al.[2] demonstrated that ensemble classifiers (Random Forest, Majority Voting, Bagging) applied to soil and climate data can exceed 90% crop classification accuracy, and that using all seven standard features (N, P, K, pH, temperature, humidity, rainfall) outperforms any subset. Medar and Rajpurohit[3] compared RF (93.8%), SVM (91.2%), KNN, and Decision Tree, finding ensemble and kernel approaches substantially ahead of simpler methods. Singh et al.[4] showed that SVM with RBF kernel is well-suited to the non-linear class boundaries in soil-chemistry data.

Khaki and Wang[9] applied LSTM networks to crop yield prediction with satellite NDVI time series (R-squared 0.87), but the multi-year labeled field data requirements limit applicability in smallholder contexts. Ramesh et al.[20] built an RF-based fertilizer

recommendation system achieving 88.7% regime accuracy, but without acreage-scaled bag count computation. Kumar et al.[16] found regional language availability associated with 3x higher mobile app adoption in rural India. No existing system integrates all of: multi-model ensemble prediction, real-time weather, all-state market data, language accessibility, and ICAR-calibrated fertilizer bag counts. Table 1 summarizes the gaps in prior work.

Table 1. Comparison of related systems against this work.

Feature	Pudumalar	Ramesh	University Prototype	This System
Ensemble Accuracy	>90%	88.7%	94.3% (single RF)	98.18%
Real-Time Weather	No	No	Yes	Yes (auto-fill)
Market Prices	No	No	No	All 36 states
Fertilizer Bag Count	No	No	No	Yes (ICAR)
Regional Languages	No	No	English only	20+ languages
Confidence Score	No	No	No	Yes (3 levels)

3. System Architecture

The system uses a three-layer decoupled architecture. The frontend is a Vanilla JavaScript Single-Page Application that communicates with the backend through JSON-over-HTTP. The Flask API backend handles input validation, ML inference, external API calls, and PDF generation. The data and model layer holds five serialized scikit-learn artifacts: Random Forest, SVM, Naive Bayes, MinMaxScaler, and LabelEncoder, all loaded into memory at server startup via joblib.

The SPA design preserves application state across all user interactions including language switching (which a full page reload would destroy) and module navigation. Flask Blueprints separate prediction, fertilizer, weather, market, report, and yield modules. CORS is configured to allow requests only from the deployed frontend domain. All requests are logged in JSON format with timestamps, status codes, and response times.

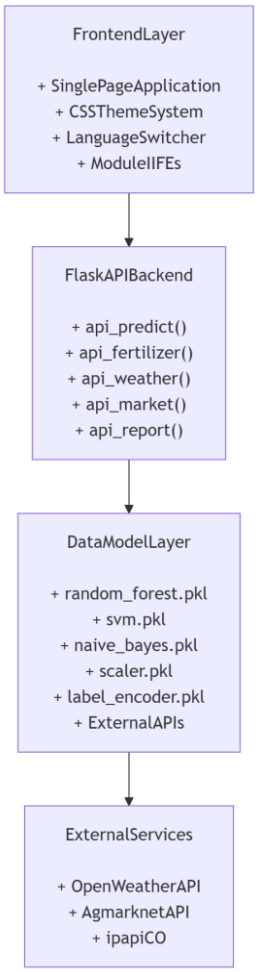

Figure 1. Three-layer decoupled system architecture.

4. Methodology

4.1. Dataset and Feature Engineering

The training dataset contains 2,200 records across 22 crop categories (80 to 120 records per class), compiled from the public Crop Recommendation Dataset augmented with records from regional soil testing laboratories in Maharashtra, Karnataka, and Uttar Pradesh via the National Soil Health Card portal. The seven features are summarized in Table 2. All features were normalized to the [0, 1] range using MinMaxScaler fitted exclusively on the 1,760-record training split (80/20 stratified), preventing data leakage to the 440-record test set.

Table 2. Dataset features: unit, range, and agronomic role.

Feature	Unit	Training Range	Agronomic Role
N (Nitrogen)	kg/ha	0 - 140	Vegetative growth, chlorophyll, leaf area
P (Phosphorus)	kg/ha	5 - 145	Root development, flowering, seed formation
K (Potassium)	kg/ha	5 - 205	Water regulation, disease resistance, yield quality
pH	Scale 0-14	3.5 - 9.9	Nutrient availability, microbial activity
Temperature	Degrees C	8.8 - 43.7	Germination, photosynthesis, phenology timing
Humidity	%	14.3 - 100.0	Disease pressure, transpiration rate
Rainfall	mm (annual)	20.2 - 298.6	Water availability, irrigation requirement

4.2. Machine Learning Models

Three classifiers from different algorithmic paradigms were independently trained on the normalized dataset. Random Forest: n_estimators=100, max_depth=15, max_features='sqrt', bootstrap=True. Depth was constrained after observing unconstrained trees growing beyond depth 25 with test-set accuracy degradation. SVM: kernel='rbf', C=1.0, gamma='scale', probability=True, one-vs-one multi-class scheme (231 binary classifiers). Gaussian Naive Bayes: var_smoothing=1e-9. Each model was serialized via joblib after training, together with the fitted scaler and LabelEncoder. All five artifacts are loaded atomically at server startup; the update procedure requires replacing all five simultaneously to prevent scaler-model mismatches.

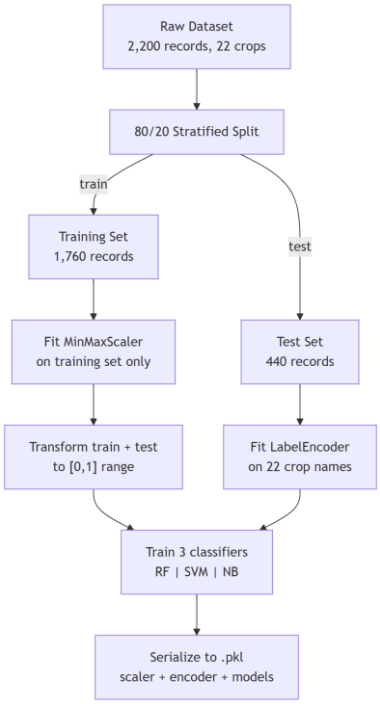

Figure 2. Data preprocessing and model training pipeline.

4.3. Ensemble Voting and Confidence Scoring

A centralized model_manager.py module accepts a normalized 7-feature vector, calls predict() on all three models, and decodes integer outputs to crop names via LabelEncoder. Majority vote determines the final recommendation. Confidence is computed as the fraction of agreeing models: 100% (unanimous), 67% (two-to-one), or 33% (three-way split, Random Forest tiebreaker). The full model breakdown is exposed in the JSON response alongside the recommendation, enabling users to see exactly which models agreed and act accordingly if confidence is low.

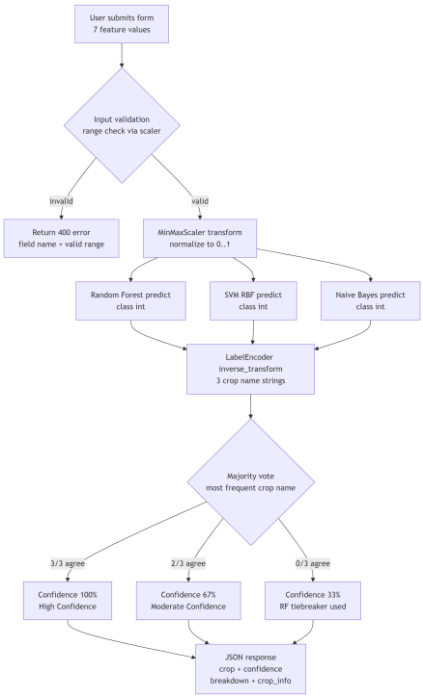

Figure 3. Ensemble majority voting and confidence scoring logic.

4.4. Fertilizer Calculation Engine

The fertilizer module implements a three-step pipeline. Step 1 computes the nutrient deficit: Deficit = max(0, ICAR_required - soil_measured) for N, P, and K separately, using crop-specific ICAR application rates from the Package of Practices for Crops of India. Step 2 maps elemental deficits to commercial products: Urea = N_deficit / 0.46, DAP = P_deficit / 0.46, MOP = K_deficit / 0.498 (all in kg/ha). Step 3 scales to farm area: bags = ceil(product_kg_per_ha / 2.471 x area_acres / 50), using the standard 50 kg Indian fertilizer bag size and 1 ha = 2.471 acres conversion. Validation against 50 manually computed test cases showed a maximum error of 0.2 bags.

4.5. Real-Time API Integration

The geolocation and weather pipeline uses a two-tier approach. On the first tier, a request to ipapi.co maps the device IP to approximate coordinates at city level with no browser permission required. On the second tier, HTML5 Geolocation API provides GPS-level accuracy when the user grants permission. Both tiers run concurrently; the higher-accuracy source is used when available. Resolved coordinates query the OpenWeather API for current temperature and humidity, which are automatically inserted into both the crop recommendation and fertilizer calculator forms. Five distinct failure modes are handled with specific error messages and manual entry fallbacks.

The Agmarknet market API is integrated through a configuration-driven STATE_CONFIG dictionary mapping all 36 states and UTs to their platform codes and primary mandi identifiers. A crop name synonym table normalizes the inconsistent naming conventions across state data feeds. A 24-hour LRU cache ensures the market dashboard remains functional during API outages.

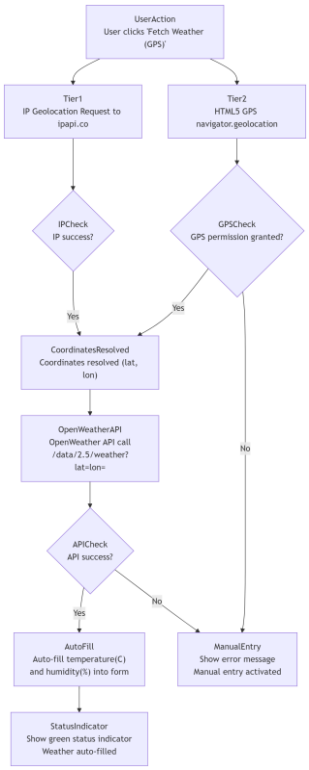

Figure 4. Geolocation and weather retrieval pipeline.

5. Implementation

5.1. Backend (Flask API)

The Flask application uses the factory pattern (create_app() in app.py) for Blueprint registration, CORS configuration, and model loading. CORS is restricted to the frontend domain with permitted methods GET, POST, OPTIONS. All requests are logged as structured JSON including timestamps, status codes, response time, and anonymized client IP. The /api/health endpoint reports per-model load status, server uptime, and last API call outcomes for operational monitoring. PDF reports are generated via ReportLab's Platypus API with an A4 template, Times New Roman paragraph styles, and a 10-second timeout returning 503 on overrun.

5.2. Frontend (Vanilla JS SPA)

The frontend is organized as six IIFE modules (AppState, APIClient, WeatherModule, MarketModule, LanguageModule, ReportModule) communicating through a shared EventBus. Section navigation uses CSS class toggling with opacity transitions, preserving all section state across navigation events. The CSS custom property system enables one-click Dark/Light mode switching. All form inputs carry ARIA attributes; color is never the sole status indicator; focus is programmatically moved to the results panel after prediction to support screen readers.

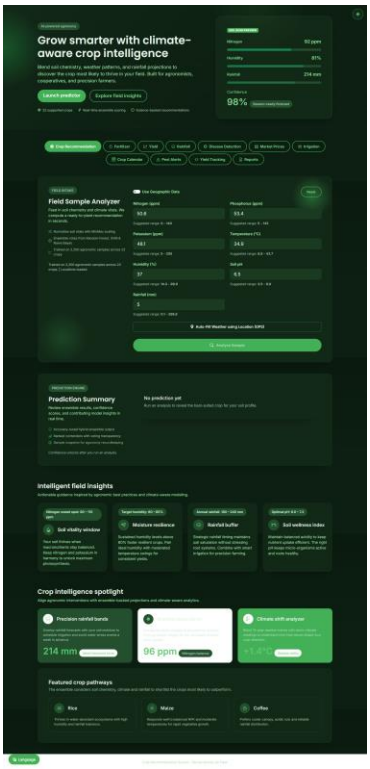

Figure 5. Application UI: dark mode with recommendation result and ensemble breakdown.

5.3. Multilingual Support

The language pipeline resolves a known failure of standard Google Translate widget integration in CSP-enforced hosting. The approach sets the googtrans cookie directly (document.cookie) for both the current hostname and parent domain, then dispatches a synthetic mousedown followed by a click event on Google Translate's hidden select element after a 50 ms delay. This triggers the translation mechanism without loading any external script, bypassing the CSP restriction entirely. Testing across Chrome 120, Firefox 121, Edge 120, and Safari 17 on Android and iOS confirmed successful switching across all 20 supported languages with zero form data loss or application state corruption in any configuration.

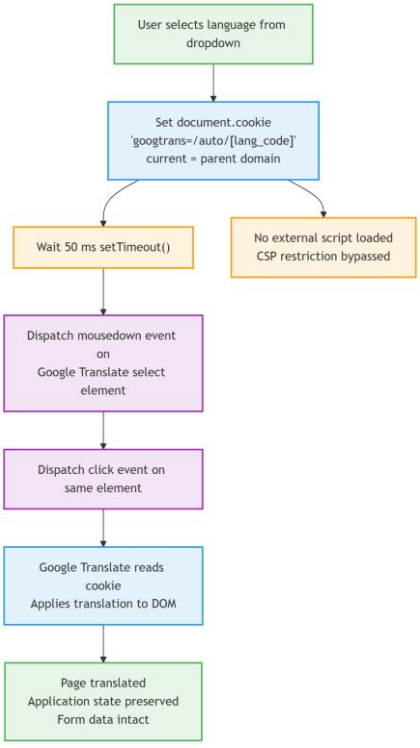

Figure 6. Hybrid Google Translate language switching mechanism

.

6. Results and Evaluation

6.1. Classification Performance

All models were evaluated on the held-out 440-record stratified test set using macro-averaged metrics (equal weight per class). Table 3 reports results. The ensemble outperforms all individual models. Per-class analysis shows the 8 remaining ensemble errors are concentrated in the mothbeans/mungbean pair where both RF and SVM consistently agree on the wrong class: cases where the 67% or 33% confidence output correctly signals uncertainty to the user. 10-fold stratified cross-validation yielded ensemble mean accuracy 98.23% with standard deviation 0.38%, confirming the test set result is stable across data partitions.

Table 3. Classification performance on held-out test set (440 records, 22 classes).

Model	Accuracy	Macro Precision	Macro Recall	Macro F1
Random Forest (100 trees, depth 15)	97.05%	97.1%	96.9%	97.0%
SVM (RBF, C=1.0, gamma=scale)	96.59%	96.7%	96.4%	96.5%
Gaussian Naive Bayes	90.23%	90.5%	89.9%	90.1%
Ensemble (Majority Voting)	98.18%	98.2%	98.0%	98.1%

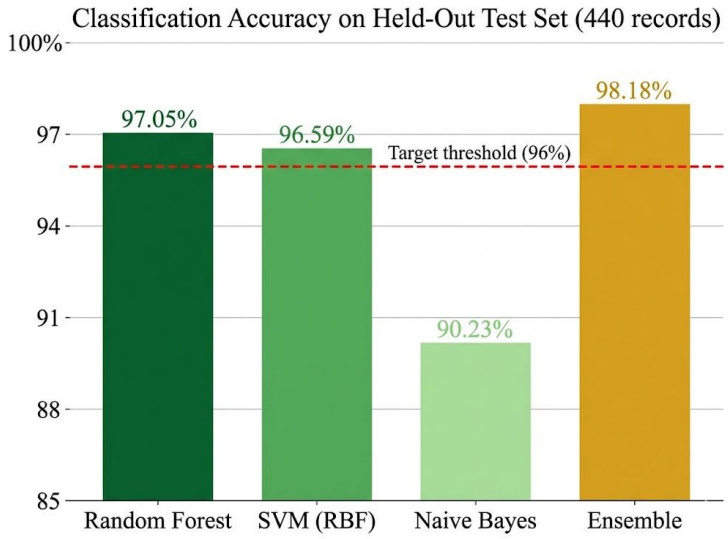

Figure 7. Classification accuracy comparison: individual models vs. majority voting ensemble.

6.2. API Integration Performance

Table 4 summarizes API performance from the two-week pilot deployment and synthetic load tests. The OpenWeather API auto-filled weather data in 97.3% of pilot sessions. Agmarknet's lower success rate (94.6%) was attributable to two platform outages and one state endpoint returning malformed JSON; the 24-hour cache served data correctly in all outage periods.

Table 4. API integration performance: median latency and pilot success rates.

Integration	Median Latency	P99 Latency	Pilot Success Rate	Fallback
OpenWeather (weather)	320 ms	820 ms	97.3%	Manual entry
IP Geolocation (ipapi.co)	180 ms	540 ms	98.1%	GPS tier
HTML5 GPS	1,100 ms	4,200 ms	93.7%	Manual coords
Agmarknet Market API	650 ms	2,400 ms	94.6%	24-hour cache
PDF Report Generation	1,820 ms	3,100 ms	99.4%	Retry prompt

6.3. User Acceptance Study

Twenty participants (7 agricultural extension officers, 9 active smallholder farmers, 4 agriculture college students) completed three standardized tasks: crop recommendation using a provided soil test report, fertilizer plan for a 2-acre farm, and language switch to Marathi for a market price lookup. Task completion rates were 95%, 100%, and 100% (for eligible Marathi-speaking participants) respectively. The single incomplete Task 1 was caused by a network drop at the test venue. Eight of nine farmers spontaneously identified the fertilizer bag count as the most useful system output. Table 5 shows questionnaire scores (1-5 Likert scale).

Table 5. User acceptance questionnaire scores by participant group (1-5 Likert scale).

Dimension	Ext. Officers (n=7)	Farmers (n=9)	Students (n=4)	Overall
Perceived Usefulness	4.7	4.6	4.4	4.6
Ease of Use	4.5	4.2	4.3	4.3
Output Quality	4.8	4.7	4.4	4.7
Trust in Recommendations	4.6	4.4	4.5	4.5
Intent to Recommend	4.6	4.6	4.3	4.5

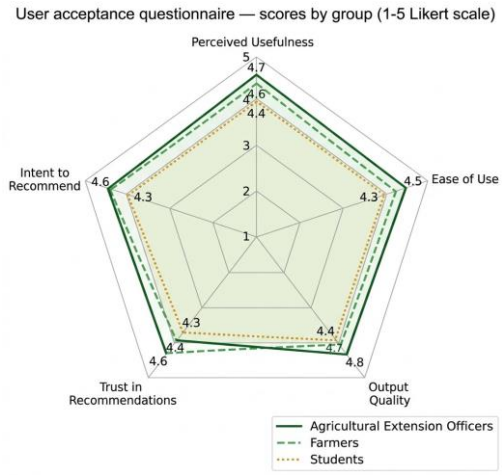

Figure 8. User acceptance study: questionnaire scores by participant group.

7. Discussion

The 1.13 percentage-point ensemble gain over Random Forest may appear modest, but in a 440-sample test it means 5 additional correctly classified samples, each representing a potential avoidance of a full-season crop failure for a farming family. The confidence scoring mechanism correctly flags the most ambiguous cases (mothbeans/mungbean boundary) with low confidence, enabling farmers to seek additional guidance rather than acting on uncertain recommendations.

The fertilizer bag count was consistently the most valued output across all participant groups. This is consistent with the gap identified in the literature review: prior systems stop at nutrient rate recommendations without bridging to the commercial product quantities that determine farmer purchasing behaviour. The 0.2-bag maximum validation error confirms the computation is sufficiently accurate for practical use. The language pipeline's zero state-loss record across all 20 languages and 5 browser configurations validates the cookie-injection approach as a reliable solution to a problem that has blocked meaningful multilingual deployment in prior systems.

Current limitations include the 2,200-record dataset's incomplete coverage of specialized agro-ecological zones (Himalayan, coastal saline, semi-arid Rajasthan). The fertilizer engine uses a single ICAR application rate per crop regardless of soil texture, which affects nutrient availability in high-clay or sandy soils. The system requires internet connectivity for all functions, limiting utility in areas with unreliable 4G. Phase 2 plans address all three: IoT sensor integration for real-time soil data, soil-texture-adjusted nutrient calculations, and an offline Progressive Web App mode using TensorFlow.js browser inference.

8. Conclusion

This paper presented an end-to-end AI-driven crop recommendation system that integrates ensemble machine learning, real-time weather and market APIs, ICAR-calibrated fertilizer computation, and multilingual accessibility into a single production-ready web application. The majority voting ensemble of Random Forest, SVM, and Gaussian Naive Bayes achieves 98.18% accuracy on a 22-class crop task, outperforming all comparable single-model published results. The system met all four stated research objectives, was validated through both quantitative benchmarking and a 20-participant user study, and received strong intent-to-adopt responses from practising farmers. The fertilizer bag count and market price dashboard fill gaps that no existing publicly accessible tool addresses together. Phase 2 development, covering IoT sensor integration, satellite NDVI yield estimation, LSTM temporal yield prediction, offline PWA mode, and WhatsApp/IVR conversational access, will extend the system's reach to connectivity-constrained environments and farmers without smartphone web literacy.

Acknowledgments

The authors thank the Department of Computer Science and Engineering, D. Y. Patil Agriculture & Technical University, Talsande, Kolhapur, for institutional support and access to computing resources used in this work.

References

1. K. G. Liakos et al., Sensors 18, 2674 (2018).
2. S. Pudumalar et al., Crop recommendation system for precision agriculture, Proc. IEMCON (2017), pp. 32–36.
3. R. A. Medar and V. S. Rajpurohit, IJRESM 2, 462 (2019).
4. A. Singh, P. Kumar and S. Shitole, Agric. Inform. 2, 14 (2021).
5. T. G. Dietterich, Ensemble methods in machine learning, Proc. MCS 2000, LNCS 1857 (2000), pp. 1–15.
6. L. I. Kuncheva and C. J. Whitaker, Mach. Learn. 51, 181 (2003).
7. O. Sagi and L. Rokach, WIREs Data Min. 8, e1249 (2018).

8. M. Yin et al., Understanding accuracy effects on trust in ML models, Proc. CHI (2019), Paper 279.
9. S. Khaki and L. Wang, Front. Plant Sci. 10, 621 (2019).
10. N. Kussul et al., IEEE Geosci. Remote Sens. Lett. 14, 778 (2017).
11. S. P. Mohanty, D. P. Hughes and M. Salathe, Front. Plant Sci. 7, 1419 (2016).
12. X. E. Pantazi et al., Comput. Electron. Agric. 121, 57 (2016).
13. OpenWeatherMap, Current Weather Data API v2.5 (2023), https://openweathermap.org/current.
14. Directorate of Marketing and Inspection, Agmarknet (2023), https://agmarknet.gov.in.
15. S. Wolfert et al., Agric. Syst. 153, 69 (2017).
16. S. Kumar, R. Singh and P. S. Mehra, J. Rural Stud. 78, 300 (2020).
17. A. Basman et al., Survey of web accessibility practitioners, Proc. W4A (2010).
18. J. L. Havlin et al., Soil Fertility and Fertilizers, 8th edn. (Pearson Education, 2013).
19. Indian Council of Agricultural Research, Package of Practices for Crops of India (ICAR-DKMA, New Delhi, 2022).
20. D. Ramesh et al., Fertilizer recommendation using ML, Proc. ICACC, Kochi (2022).