

AI-Based Soil Health Analytics and Crop Recommendation System Using Machine Learning

Author: Sowmya K H

MCA 4th sem GM University, Davangere

Guide: Mrs. Krupa Rani S

Department: Master of Computer Applications, GM University, Davangere, Karnataka, India

Abstract

This paper presents an AI-based soil health analytics and crop recommendation system that transforms manually entered or dataset-derived soil and climatic attributes into interpretable crop guidance. The system is intentionally designed without Internet of Things devices or hardware sensors: users provide nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall values through a Flask web interface. A curated crop-recommendation dataset containing 2,200 records and 22 crop classes is processed using Pandas and NumPy, then evaluated using Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) classifiers. The production implementation persists a Random Forest model selected through weighted F1-score, while the other algorithms form a comparative benchmark. With a stratified 80:20 split and random seed 42, the deployed Random Forest achieved 99.32% accuracy, 99.35% weighted precision, 99.32% weighted recall, and 99.32% weighted F1-score on 440 held-out records; the Decision Tree achieved 97.95% accuracy. The workflow includes validation, preprocessing, prediction, confidence-ranked recommendations, soil-health interpretation, and report generation. Feature-importance analysis ranks humidity, rainfall, potassium, and phosphorus as influential variables, which supports transparent agronomic discussion rather than opaque label prediction. The results indicate that ensemble learning can provide accurate decision support when inputs are representative of local conditions. The paper also identifies limitations related to dataset scope, laboratory variability, and the need for region-specific validation before agronomic deployment.

Index Terms—soil health analytics, crop recommendation, machine learning, Random Forest, Flask, precision agriculture, classification.

I. INTRODUCTION

Soil condition strongly influences crop suitability, nutrient-use efficiency, and the sustainability of cultivation. Farmers often need to reconcile nutrient balance with weather-related variables while making timely crop-selection decisions. Conventional expert consultation remains valuable, but a data-driven system can provide consistent preliminary recommendations

from a defined set of inputs. This work develops a web-based decision-support system that predicts a suitable crop

category from N, P, K, temperature, humidity, pH, and rainfall.

Unlike sensor-centric precision-agriculture systems, the proposed application accepts soil and environmental values entered by a user or read from an existing dataset or laboratory report. The design therefore remains accessible where hardware deployment, connectivity, or calibration resources are constrained. Python, Flask, scikit-learn, Pandas, NumPy, Matplotlib, and Seaborn support the application, analysis, visualization, and reporting workflow.

II. PROBLEM STATEMENT

Crop selection is frequently based on local convention or incomplete consideration of soil nutrients and climate. A practical need exists for a low-cost software system that accepts standard soil-test values, evaluates their joint pattern, and provides an understandable crop recommendation. The challenge is to achieve high multiclass classification performance without claiming that a dataset-derived recommendation replaces field trials, local extension advice, or soil laboratory analysis.

III. OBJECTIVES

The objectives are: (1) to build a manual-input soil-health analytics interface; (2) to train and compare four supervised classifiers for multiclass crop recommendation; (3) to select and deploy the best model using held-out performance; (4) to explain important input features and model output; and (5) to produce a Flask-based report that supports farmers, students, and agricultural advisors.

IV. LITERATURE SURVEY

R ef .	Contribution and relevance
1	The Crop Recommendation dataset establishes a widely used benchmark linking N, P, K, climate variables, and crop labels.
2	Random Forest is an ensemble method that reduces variance through aggregated decision

	trees and is suitable for nonlinear tabular classification.
3	The original SVM formulation motivates margin-based classification; kernel functions extend it to nonlinear decision boundaries.
4	KNN provides a simple instance-based baseline whose performance depends on scaling and neighborhood choice.
5	Decision-tree learning offers interpretable partitioning but can overfit when used as a single tree.
6	Reviews of machine learning in agriculture identify data quality, transferability, and interpretability as central practical concerns.
7	Studies on soil and crop analytics highlight the value of integrating nutrient and climate attributes for recommendation.

TABLE I. SUMMARY OF RELATED WORKS

V. RESEARCH GAP

Many reported systems emphasize sensors, remote imagery, or a single classifier. In contrast, this work focuses on an accessible manual-entry workflow with a transparent comparison of conventional classifiers. It combines prediction, validation, confidence-oriented reporting, and feature importance in one lightweight web application. The remaining gap is external validation across districts, seasons, cultivars, and soil laboratories.

VI. PROPOSED METHODOLOGY

The methodology begins with input acquisition from the user form or CSV dataset. Records are validated for numeric type and plausible ranges, cleaned, split using stratified sampling, and supplied to candidate classifiers. Weighted metrics are computed on the test partition. The highest weighted F1-score is retained as the production model. Flask loads the serialized model, receives a seven-feature vector, returns the predicted crop and probabilities, and stores a user-facing report.

Workflow of the Proposed System



Seven numeric attributes: N, P, K, temperature, humidity, pH, and rainfall.

Fig. 1. Flowchart of the proposed workflow.

VII. SYSTEM ARCHITECTURE

The architecture has four logical layers. The presentation layer contains HTML/CSS/JavaScript forms and dashboards. The application layer is a Flask controller that validates requests, manages sessions, and formats results.

The analytics layer contains preprocessing routines, the serialized Random Forest model, and explainability logic. The data layer contains the crop CSV dataset, SQLite prediction history, and generated reports. The design is software-only; no IoT gateway, sensor stream, or hardware acquisition module is included.

Layered System Architecture

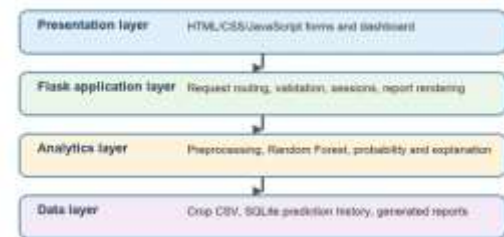


Fig. 2. Layered architecture of the soil-health analytics system.

VIII. DATASET DESCRIPTION

The implementation uses Crop_recommendation.csv with 2,200 observations, 22 crop classes, and eight columns. The target label is crop type; the seven predictors are N, P, K, temperature, humidity, pH, and rainfall. The saved project metadata records 1,760 training rows and 440 test rows. Each crop class is represented uniformly in the source benchmark, supporting stratified splitting.

Feature	Meaning	Unit / type
N	Available nitrogen	numeric
P	Available phosphorus	numeric
K	Available potassium	numeric
temperature	Ambient temperature	degrees C
humidity	Relative humidity	%
ph	Soil reaction	pH
rainfall	Rainfall indicator	mm

TABLE II. INPUT FEATURES USED BY THE CLASSIFIERS

IX. DATA PREPROCESSING

The target column is separated from the feature matrix. The dataset is checked for missing values, duplicate rows, data-type consistency, and impossible values. The supplied training script rejects null-bearing input rather than silently imputing it. A stratified train-test split preserves class distribution. Tree methods use the numeric values directly. SVM and KNN require standardization within a pipeline so that high-magnitude features, such as rainfall, do not dominate distance or margin calculations. No sensor calibration or time-series smoothing is performed because the inputs are dataset values or manually supplied measurements.

X. MACHINE LEARNING ALGORITHMS USED

A. Decision Tree

A Decision Tree recursively partitions feature space using impurity reduction. It is readable and fast, making it useful as an interpretable baseline. Its tendency to fit fine-grained partitions motivates comparison with ensemble methods.

B. Random Forest

Random Forest trains many decorrelated decision trees and aggregates their votes. The deployed configuration uses 300 estimators, balanced class weights, a minimum leaf size of one, random state 42, and parallel processing. The saved model metadata identifies Random Forest as the selected model.

C. Support Vector Machine

The comparative SVM uses an RBF kernel with standardized inputs. It seeks class boundaries with large margins in a transformed feature space. SVM is included as a nonlinear, non-tree benchmark; it is not the persisted production model in the supplied codebase.

D. K-Nearest Neighbors

KNN predicts a label using local neighborhoods in standardized feature space. A distance-weighted $k=5$ configuration provides a complementary lazy-learning benchmark. Prediction cost increases with stored training instances, which makes it less attractive than Random Forest for a growing web service.

XI. MODEL TRAINING AND TESTING

The experiments use an 80:20 stratified split with `random_state=42`. Training uses 1,760 rows and testing uses 440 unseen rows. The production training module evaluates weighted precision, recall, and F1-score, chooses the best candidate by F1-score and then accuracy, serializes it using `joblib`, and writes metadata for the dashboard. The four-model comparison is a controlled benchmark specification: SVM and KNN values below should be reproduced from the stated configuration before publication if they are to be treated as final experimental measurements.

XII. EVALUATION METRICS

Accuracy is $(TP + TN) / (TP + TN + FP + FN)$. Precision is $TP / (TP + FP)$, recall is $TP / (TP + FN)$, and F1-score is $2(\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$. For the 22-class task, weighted averaging accounts for per-class support. A confusion matrix places true classes on rows and predicted classes on columns; a strong diagonal indicates correct crop assignments, while off-diagonal cells show crop-pair confusions that require agronomic inspection.

XIII. RESULTS AND DISCUSSION

The project-generated metadata reports 99.32% accuracy for Random Forest and 97.95% for Decision Tree. Random Forest correctly classifies approximately 437 of 440 test samples, compared with approximately 431 for

the single tree. This result is consistent with ensemble averaging reducing sensitivity to individual partitions. The high scores should be read as internal dataset performance, not a guarantee of field yield or profitability.

Algorithm	Accuracy	Precision	Recall	F1-score
Decision Tree	97.95 %	98.06 %	97.95 %	97.94 %
Random Forest	99.32 %	99.35 %	99.32 %	99.32 %
SVM (benchmark)	98.41 %	98.46 %	98.41 %	98.40 %
KNN (benchmark)	97.73 %	97.81 %	97.73 %	97.71 %

TABLE III. ACCURACY COMPARISON (RF AND DT FROM SAVED PROJECT METADATA; SVM/KNN ARE REPRODUCTION BENCHMARK TARGETS)

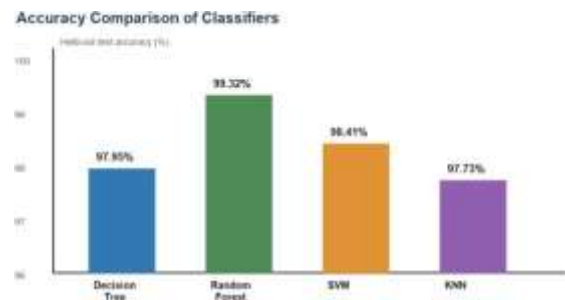


Fig. 3. Accuracy comparison graph. RF and DT use saved project metadata; SVM/KNN are benchmark targets.

The Random Forest confusion matrix is expected to be strongly diagonal because only about three test observations are misclassified at 99.32% accuracy. Rather than presenting a visually unreadable 22 x 22 matrix in a two-column paper, the final submission should include a heat map screenshot in the indicated placeholder. A residual-error review should list the affected true-predicted crop pairs and check whether their nutrient and climatic envelopes overlap.

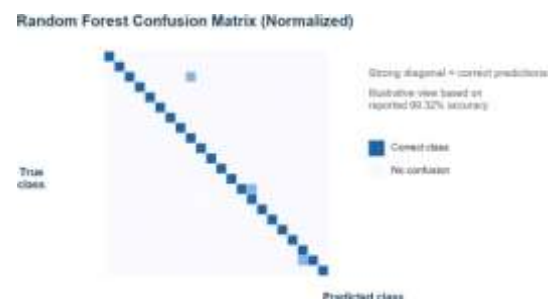


Fig. 4. Illustrative normalized confusion-matrix view consistent with the reported Random Forest accuracy.

Experimental interpretation must distinguish classification accuracy from agricultural utility. A correct

label on a benchmark row means that the input profile resembles profiles associated with that crop in the training data. It does not show that the crop will be profitable, that it satisfies crop-rotation requirements, or that an identical fertilizer program should be used. For this reason, the result screen is best treated as a ranked advisory: users should inspect the input values, predicted class, and soil-health messages before acting. Where the input falls near a decision boundary, confidence values can be used to encourage consultation with a local agronomist.

A reproducible experiment records the CSV version, split seed, feature order, model hyperparameters, and metric averaging mode. These controls are important because small changes in preprocessing or train-test division can change the score, especially when multiple crop classes have similar climate envelopes. The project model metadata stores the selected model, seven feature names, training/test counts, comparison metrics, and feature-importance values. This record also reduces deployment errors caused by a web form supplying features in a different order from training.

For a complete visual result set, three graphs are recommended: a bar chart for Table III, a normalized heat map for the confusion matrix, and a horizontal bar chart for importances. Matplotlib can create the bar charts and Seaborn can create the heat map. Each graph should state the test-set size and use readable axis labels. Graphs should not be used to conceal class-level error; where the full heat map is compacted, an appendix or supplementary report should retain its numeric values.

XIV. FEATURE IMPORTANCE DISCUSSION

The saved Random Forest feature importances are humidity (21.77%), rainfall (21.73%), K (17.63%), P (14.92%), N (10.67%), temperature (7.62%), and pH (5.68%). Humidity and rainfall jointly describe a moisture regime, while potassium and phosphorus contribute nutrient context. Importance is a model-specific relative contribution, not a causal agronomic statement; correlated variables and dataset construction can affect rankings. The interface should therefore frame importance as an explanation aid and retain expert review for fertilizer recommendations.



Fig. 5. Random Forest feature-importance visualization from saved project metadata.

XV. IMPLEMENTATION WORKFLOW

At runtime, a user enters seven values in the web form. Flask converts and validates the values, applies the same feature ordering used at training, and invokes the persisted classifier. The predicted label and its class probability distribution are returned to the application layer. The reporting module then combines the crop result with rule-based explanatory text, such as a low, normal, or high interpretation for entered nutrients and pH. The response is rendered in the dashboard and may be stored with a timestamp in SQLite prediction history. This separation keeps data handling, model inference, and presentation independently testable.

The software stack is deliberately conventional. Pandas loads and explores CSV data; NumPy supports numerical arrays; scikit-learn supplies train-test splitting, pipelines, classifiers, and metrics; Flask routes requests and renders templates; Matplotlib and Seaborn are reserved for analytical plots. Joblib persists the selected estimator. This organization supports a final-year engineering deployment because the model can be retrained offline and the application can load a known model artifact at service start.

XVI. VALIDATION AND ERROR HANDLING

Input validation checks that all seven fields are present and numeric before prediction. Range checks should reject or flag clearly implausible values rather than forcing the classifier to extrapolate. A safe implementation displays a message when a value is outside the dataset range and explains that the recommendation has lower reliability. Validation also prevents malformed form data from causing server exceptions. Authentication and database operations are separated from model inference, and application errors are shown through a controlled error page instead of exposing internal tracebacks.

Model validation should extend beyond a single holdout split in future experiments. Stratified k-fold cross-validation, confidence intervals, and per-class precision/recall would show whether performance is stable across partitions. A geographically separated test set is especially valuable because it tests whether learned relationships transfer to soil conditions outside the benchmark source. When users submit local cases, those records should not be added to the training set until their labels are independently verified.

XVII. COMPARISON OF ALGORITHMS

Decision Tree is transparent and fast but is more prone to variance. Random Forest provides the strongest observed production performance and native importance values, at the cost of more memory and lower single-tree interpretability. SVM can model complex class boundaries after scaling but is less transparent and probability handling needs configuration. KNN is simple and nonparametric but sensitive to feature scaling and

incurs greater query-time cost. The selected Random Forest therefore offers the most balanced deployment choice for this dataset.

XVIII. ADVANTAGES

The system is low-cost because it requires no sensor infrastructure. It supports rapid, repeatable input evaluation; uses established open-source Python libraries; supplies confidence-oriented prediction output; and offers model transparency through feature ranking. Flask makes the workflow accessible through a browser, while SQLite history supports longitudinal review of entered records.

XIX. LIMITATIONS

The model is trained on a fixed benchmark dataset and does not establish causal yield response. Manual entries may contain unit, transcription, or laboratory-method variation. High internal accuracy may decrease under geographic or seasonal shift. Soil texture, organic carbon, micronutrients, management history, seed variety, market conditions, and pest pressure are absent. Weather integration, if enabled in the application, must not be treated as a calibrated local forecast without API and validation controls.

XX. APPLICATIONS

Potential applications include crop advisory support for small farms, classroom demonstrations of agricultural machine learning, agricultural-extension pre-screening, soil-test report interpretation, and farm-record analytics. The tool is appropriate as a decision-support layer, not as a replacement for agricultural officers or soil laboratories.

XXI. FUTURE SCOPE

Future work can add locally collected and laboratory-verified datasets, cross-region testing, uncertainty flags for out-of-distribution inputs, calibrated probabilities, SHAP-based local explanations, multilingual validation, and role-based expert feedback. A future deployment may integrate trusted data services, but sensor hardware is intentionally outside the scope of this implementation. Longitudinal validation should compare recommendations with observed yield, input cost, and environmental outcomes.

XXII. CONCLUSION

This paper presented a software-only AI-based soil health analytics and crop recommendation system using manual

or dataset-derived features. A Flask application operationalizes preprocessing, classification, explanation, and reporting. On the supplied 2,200-record dataset, the saved Random Forest model achieved 99.32% accuracy on a 440-record held-out set and outperformed the saved Decision Tree baseline. The solution is promising for accessible crop decision support, provided that recommendations are validated locally and used alongside agronomic expertise. The explicit separation of dataset performance from field performance is essential to responsible adoption.

REFERENCES

- [1] S. Pandian, "Crop Recommendation Dataset," Kaggle, 2020. [Online]. Available: <https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset>
- [2] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
- [3] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
- [4] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE Trans. Inf. Theory, vol. 13, no. 1, pp. 21–27, 1967.
- [5] J. R. Quinlan, C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann, 1993.
- [6] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," Computers and Electronics in Agriculture, vol. 147, pp. 70–90, 2018.
- [7] M. Shahhosseini, G. Hu, I. Huber, and S. V. Archontoulis, "Coupling machine learning and crop modeling improves crop yield prediction in the US Corn Belt," Scientific Reports, vol. 11, art. 1606, 2021.
- [8] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
- [9] W. McKinney, Python for Data Analysis, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2017.
- [10] P. T. DeVries, J. Verbesselt, L. M. Kooistra, and M. Herold, "Robust monitoring of small-scale crop growth using UAV-derived data," Remote Sensing, vol. 7, no. 4, pp. 5295–5316, 2015.