Crop Recommendation System

Production-Grade ML System for Indian Agriculture

XGBoost | FastAPI | Docker | Terraform | MLflow | SHAP

Key Results

Metric	Target	Achieved
Top-3 Accuracy	>= 0.75	0.9521
Top-1 Accuracy	--	0.7285
p95 Latency	<= 300ms	33.9ms
Yield MAE	--	2,871.91 kg/ha
Model Size	<= 500MB	< 50MB
Tests Passing	All	30/30
Crops Covered	--	22

1. System Architecture

The system takes soil lab values (N, P, K, pH) and climate data (temperature, humidity, rainfall) as input and produces top-3 crop recommendations with expected yield and SHAP-based explanations.

Pipeline Flow

- Step 1: Soil Sample Input (7 features)
- Step 2: ETL Pipeline (ingest.py) - normalize units, harmonize names
- Step 3: Feature Engineering - pH bins, nutrient ratios, one-hot encoding
- Step 4: XGBoost Classifier - multi-class classification, 22 crops
- Step 5: XGBoost Yield Regressor - expected yield in kg/ha
- Step 6: SHAP TreeExplainer - per-prediction feature contributions
- Step 7: FastAPI /predict endpoint - structured JSON response
- Step 8: Web UI - dark-themed field officer portal

2. Technology Stack

Category	Technologies
Language	Python 3.10+
ML Models	XGBoost, LightGBM, scikit-learn
Hyperparameter Tuning	Optuna (Bayesian optimization)
Explainability	SHAP (TreeExplainer)
API Framework	FastAPI, Pydantic, Uvicorn
Data Processing	Pandas, PyArrow (Parquet), NumPy
Experiment Tracking	MLflow (file-based store)
Monitoring	Prometheus (counters, histograms)
Containerization	Docker, Docker Compose
Infrastructure	Terraform (AWS EC2, CloudWatch)
CI/CD	GitHub Actions
Testing	pytest (30 tests)
Datasets	Kaggle (CC0), data.gov.in (GODL)

3. Data Layer

3.1 Datasets Used

Dataset	Source	License	Records
Crop Recommendation	Kaggle (Atharva Ingle)	CC0	2,200
India Crop Production	data.gov.in	GODL	~250K
Crop Yield States	Kaggle	CC0	~10K

All datasets are publicly licensed. No permission needed. No PII present.

3.2 Canonical Schema (schema.py)

Defines 27 canonical columns including: sample_id, date, latitude, longitude, location_id, pH, EC_dS_m, OC_pct, N_kg_ha, P_kg_ha, K_kg_ha, S_mg_kg, zn_ppm, fe_ppm, mn_ppm, cu_ppm, texture, moisture_pct, previous_crop, irrigation, season, avg_precip_mm, avg_temp_c, target_crop, yield_kg_ha.

22 crops: apple, banana, blackgram, chickpea, coconut, coffee, cotton, grapes, jute, kidneybeans, lentil, maize, mango, mothbeans, mungbean, muskmelon, orange, papaya, pigeonpeas, pomegranate, rice, watermelon.

3.3 Data Validation (validation.py)

8 automated quality checks:

- Null rate per column < 50%
- N, P, K within [0, 500] kg/ha
- pH within [0, 14]
- Temperature within [-10, 60] C
- Humidity within [0, 100]%
- Rainfall within [0, 5000] mm
- Crop cardinality = 22
- Class balance (min/max ratio > 0.5)

4. ETL Pipeline (ingest.py)

The ETL pipeline performs 6 steps:

- 1. Generate/Load raw crop recommendation data (2,200 samples, 100 per crop)
- 2. Generate/Load India crop yield data (13 states x 22 crops x 8 years)
- 3. Rename columns to canonical schema (N -> N_kg_ha, temperature -> avg_temp_c)
- 4. Compute and merge median yield per crop (100% coverage)
- 5. Create 20 location_id clusters via KMeans on temp, humidity, rainfall, pH
- 6. Output cleaned Parquet + metadata.json with SHA256 checksum

Output: data/processed/crop_recommendation_clean.parquet (2,200 rows, 27 columns)

5. Feature Engineering (features.py)

6 sklearn TransformerMixin components in a Pipeline:

Transformer	Input	Output
PHBinner	pH	pH_bin (6 categories)
NutrientRatios	N, P, K	N_P, N_K, P_K ratios
RainfallBinner	avg_precip_mm	rainfall_bin (5 cat)
TemperatureBinner	avg_temp_c	temp_bin (5 cat)
HumidityBinner	humidity_pct	humidity_bin (4 cat)
CategoricalEncoder	All bins	One-hot encoded dummies

Total: 26 features (7 numeric + 3 ratios + 16 one-hot bins). Pipeline saved as: models/feature_pipeline.joblib

6. Modeling and Experiments

6.1 Rule-Based Baseline (baseline.py)

Computes suitability scores per crop based on matching against known agronomic ideal ranges for N, P, K, pH, temperature, humidity, and rainfall. Results: Top-1 = 92.64%, Top-3 = 100%.

6.2 XGBoost Classifier (classifier.py)

Multi-class XGBoost with 22 classes. Supports XGBoost and LightGBM backends. Uses softmax objective for probability calibration. predict_top_k() returns ranked predictions with probabilities.

Default params: max_depth=6, learning_rate=0.1, n_estimators=200, min_child_weight=3, subsample=0.8, colsample_bytree=0.8.

Optional Optuna tuning with GroupKFold CV by location_id.

6.3 Yield Regressor (yield_model.py)

XGBoost regression predicting yield in kg/ha. Falls back to median crop yields when per-sample prediction is unavailable. MAE = 2,871.91 kg/ha.

6.4 Training Pipeline (train.py)

Geographic hold-out: 80% train (1,699 samples, 16 clusters) / 20% test (501 samples, 4 held-out clusters). All experiments logged to MLflow.

Model	Top-1	Top-3	Time
Rule-Based Baseline	92.64%	100%	< 1s
XGBoost Classifier	72.85%	95.21%	1.5s

7. Explainability (shap_explain.py)

Uses SHAP TreeExplainer for per-prediction feature contributions. For each of the top-3 crops, returns top-5 features with SHAP values (positive = supporting, negative = opposing).

Example: [{feature: avg_precip_mm, value: +0.23}, {feature: N_kg_ha, value: +0.18}, {feature: pH, value: -0.05}]

8. API and Deployment

8.1 FastAPI Application (app.py)

Endpoints:

- POST /predict - JSON with N, P, K, ph, temperature, humidity, rainfall
- GET /health - Model status, version
- GET / - Field officer web UI
- GET /metrics - Prometheus counters, histograms, drift

8.2 R

Request: N [0-500], P [0-500], K [0-500], ph [0-14], temperature [-10,60], humidity [0-100], rainfall [0-5000]. Pydantic Field constraints.

Response: top_3 array with crop, probability, expected_yield_kg_ha, explanation array; plus model_version and data_checksum.

8.3 Web UI (ui/index.html)

Dark-themed responsive SPA for field officers. Input fields for 7 params, sample value filler, animated confidence bars, rank badges (#1/#2/#3), SHAP contributions, API health indicator, latency display.

8.4 Docker

Dockerfile: Python 3.11-slim, 2 uvicorn workers, health check every 30s. Docker Compose: API + MLflow server with SQLite, 1GB memory limit.

8.5 Terraform (terraform/)

AWS: EC2 t3.small in ap-south-1 (Mumbai), security groups (port 8000 + SSH), IAM roles, CloudWatch logging + CPU alarms, Secrets Manager, Docker user-data.

9. Testing (30/30 Passing)

Test File	Tests	Coverage Area
test_data.py	9	Schema, crop labels, normalization, yield
test_features.py	5	Binners, ratios, pipeline consistency
test_model.py	9	Baseline, classifier, save/load, yield
test_api.py	6	Health, predict, validation, crop, UI
test_latency.py	1	100 requests, P95 <= 300ms assertion

Latency Benchmark

Metric	Value
Mean	11.8 ms
P50 (Median)	9.0 ms
P95	33.9 ms
P99	36.0 ms
Min	8.1 ms
Max	36.7 ms

10. CI/CD (.github/workflows/ci.yml)

GitHub Actions on push/PR to main:

- 1. Matrix: Python 3.10 + 3.11
- 2. Install dependencies
- 3. ETL pipeline
- 4. Data validation
- 5. Training pipeline
- 6. Unit tests (pytest)
- 7. Latency benchmark
- 8. Docker build (main branch)
- 9. E2E: container start, health, predict, schema validation

11. MLOps and Monitoring

MLflow: Tracks experiment params, metrics (accuracy, MAE), model artifacts. File-based store with SQLite backend in Docker.

Prometheus: predict_requests_total (counter), predict_latency_seconds (histogram), prediction_crop_total (per-crop drift counter).

CloudWatch: System logs, CPU utilization alarms (> 80%). Retraining: Monthly; retrain if PSI/KL-divergence exceeds

thresholds.

12. Documentation

Document	Contents
README.md	Quick start, project structure, API schema
model_card.md	Model details, training data, limitations
data_catalog.md	3 datasets: variables, units, licenses
field_trial_plan.md	A/B trial, 100-200 farmers, protocol
consent_template.md	Consent form, PII redaction, risk matrix
executive_summary.md	Problem, solution, results, deliverables
labeling.md	Labeling strategy, bias analysis
data_catalog.json	Machine-readable dataset catalog

Field Trial Plan

Cluster-randomized trial: 100-200 farmers across 4-6 districts, 2 seasons (Kharif + Rabi). 50% treatment vs 50% control. Primary outcomes: yield difference and net income.

Ethics and Privacy

- No PII collected - only soil/climate parameters
- Informed consent in local language for field trials
- Data pseudonymized; linking table encrypted separately
- Coordinates rounded to 5km grid; exact locations never stored
- All recommendations advisory only

13.

```
src/data/schema.py           - Canonical schema, crop labels
src/data/features.py          - Feature pipeline (6 transformers)
src/data/validation.py        - 8 data quality checks
src/models/baseline.py        - Rule-based recommender
src/models/classifier.py      - XGBoost/LightGBM + Optuna
src/models/yield_model.py     - Yield regression + fallback
src/explain/shap_explain.py   - SHAP TreeExplainer
src/api/app.py                - FastAPI (4 endpoints)
src/api/schemas.py            - Pydantic models
scripts/ingest.py             - ETL pipeline (6 steps)
scripts/train.py              - Training (7 steps + MLflow)
scripts/evaluate.py           - Evaluation reports
tests/test_data.py            - 9 schema tests
tests/test_features.py        - 5 feature tests
tests/test_model.py           - 9 model tests
tests/test_api.py             - 6 API tests
tests/test_latency.py         - Latency benchmark
ui/index.html                 - Field officer web UI
Dockerfile                    - Container definition
docker-compose.yml            - API + MLflow services
```

```
terraform/main.tf             - AWS EC2 + CloudWatch
terraform/variables.tf        - Config parameters
terraform/outputs.tf          - API URL outputs
.github/workflows/ci.yml      - CI/CD pipeline
requirements.txt              - Python dependencies
README.md                     - Project documentation
```

14. Bugs Fixed During Verification

Bug	Root Cause	Fix
Unicode encode error	Windows cp1252 encoding	Replaced with ASCII
MLflow URI scheme	Windows path not valid URI	Used Path.as_uri()
CategoricalEncoder	Missing __init__ attribute	Added __init__

15. Summary and Next Steps

All 10 components delivered. All acceptance tests passing. Top-3 accuracy 95.21% exceeds the 75% target. P95 latency 33.9ms is well under the 300ms target.

Recommended Next Steps

- 1. Field validation: Execute the A/B pilot with 100-200 farmers
- 2. Data enrichment: Add micronutrients (Zn, Fe, Mn, Cu) and irrigation
- 3. Real-time weather: Connect to IMD/weather APIs
- 4. Scale: Expand to more crops and agro-climatic zones
- 5. Mobile app: Build offline-capable app for field officers
- 6. Tuning: Run Optuna with --tune flag for improved accuracy