Industrial Internship Report on

”Prediction of Agriculture Crop Production in India”

Prepared by

Rathore Rikendra Dolatsingh

Executive Summary

This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).

This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time.

My Project was predicting Production of Agriculture Crop in India, included a robust data ingestion pipeline, exploratory data analysis, a leak-free forecasting model, and a Streamlit application for interactive predictions, backtesting, and batch forecasting. The project was completed within the internship timeline and packaged with documentation and reproducible code.

My project built a practical forecasting workflow:

- Consolidated five heterogeneous CSV/Excel files (wide fiscal-year tables, “Particulars” series, mixed units) into a single clean dataset.
- Engineered forecast-safe features (only prior-year information), trained a RandomForest model in a scikit-learn pipeline, and benchmarked against a naive baseline.
- Delivered a Streamlit app with three tabs: Single prediction, Evaluate (backtest), and Batch forecast.

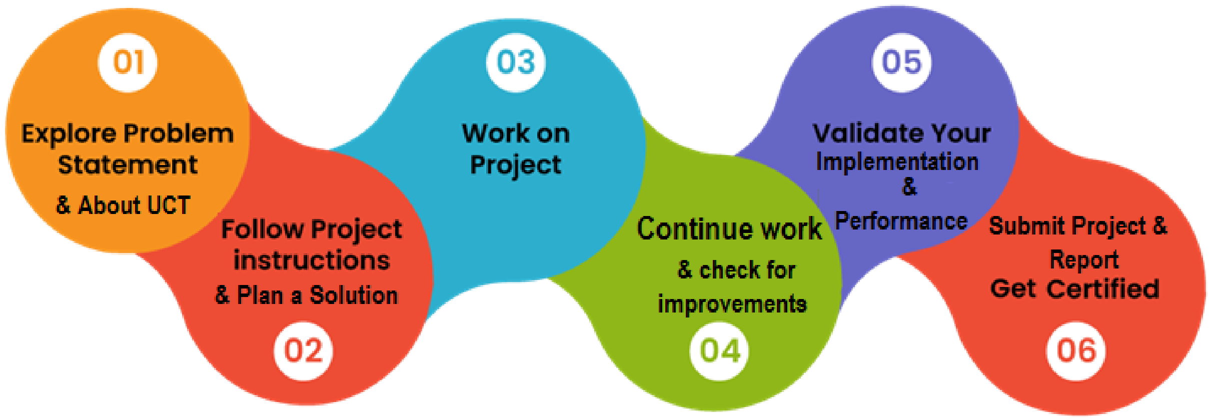
This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface 3
2 Introduction 4
2.1 About UniConverge Technologies Pvt Ltd 4
2.2 About upskill Campus 8
2.3 Objective 9
2.4 Reference 10
2.5 Glossary 10
3 Problem Statement 11
4 Existing and Proposed solution 11
5 Proposed Design/ Model 13
5.1 High Level Diagram (if applicable) 14
5.2 Low Level Diagram (if applicable) 14
5.3 Interfaces (if applicable) 15
6 Performance Test 17
6.1 Test Plan/ Test Cases 17
6.2 Test Procedure 18
6.3 Performance Outcome 19
7 My learnings 20
8 Future work scope 20

1 Preface

This report summarizes my internship work from project scoping to delivery. The internship provided practical industry exposure to data engineering, machine learning, and application development. The problem addressed—forecasting crop production for Indian agriculture—has social and economic significance, enabling better planning and policy insights.



I am grateful to:

- upskill Campus and The IoT Academy for organizing the program and mentorship,
- UniConverge Technologies Pvt. Ltd. for the problem statement and industrial context,
- My mentors and peers for their guidance on best practices in data/ML and deployment.

Overall, the internship strengthened my fundamentals in ETL, time-aware modeling, and streamlit-based productization, and improved my communication and problem-solving skills.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

uct
Uniconverge Technologies
IIOT Products
We offer product ranging from Remote IOs, Wireless IOs, LoRaWAN Sensor Nodes/ Gateways, Signal converter and IoT gateways
IIOT Solutions
We offer solutions like OEE, Predictive Maintenance, LoRaWAN based Remote Monitoring, IoT Platform, Business Intelligence…
OEM Services
We offer solutions ranging from product design to final production we handle everything for you..

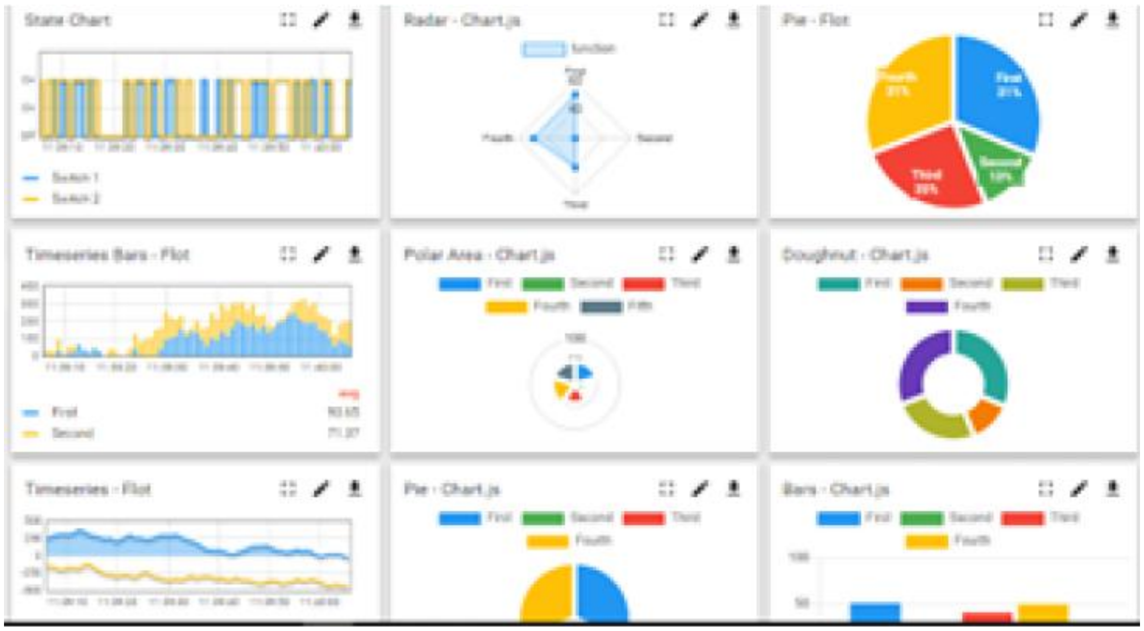
i. UCT IoT Platform (uct Insight)

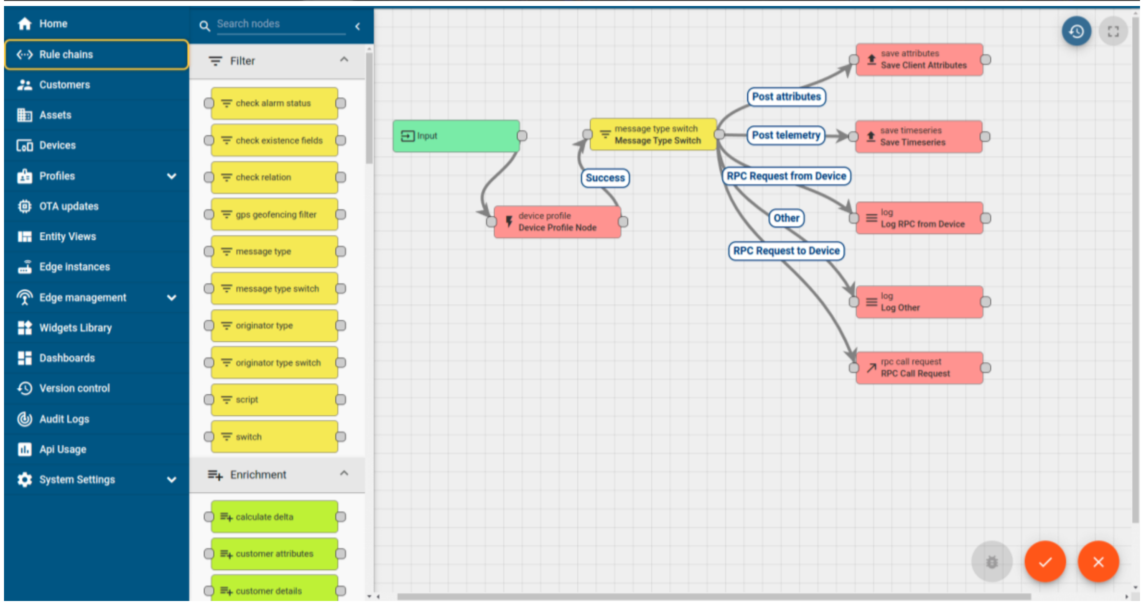
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





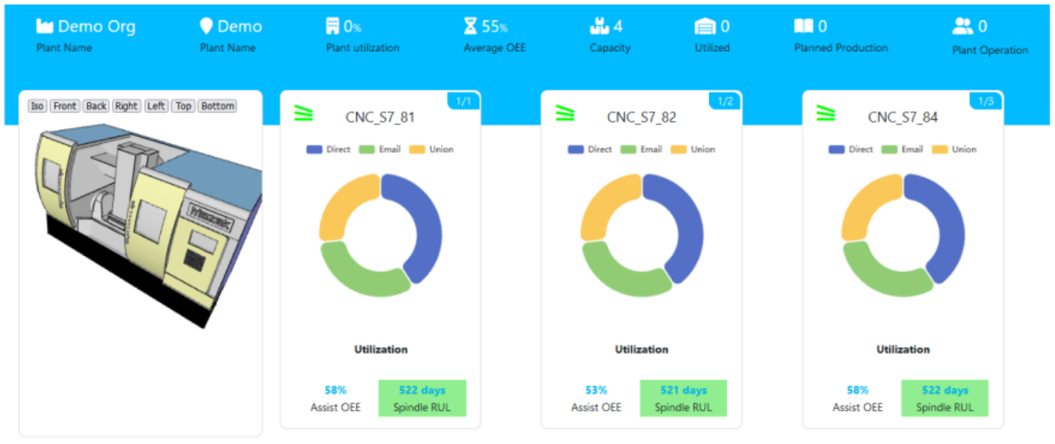
ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

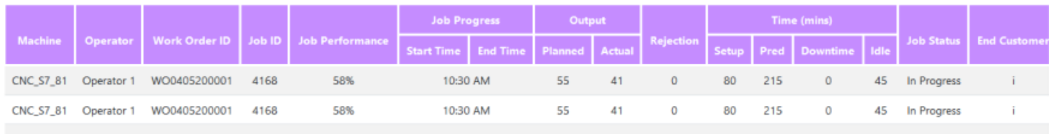
Its unique SaaS model helps users to save time, cost and money.



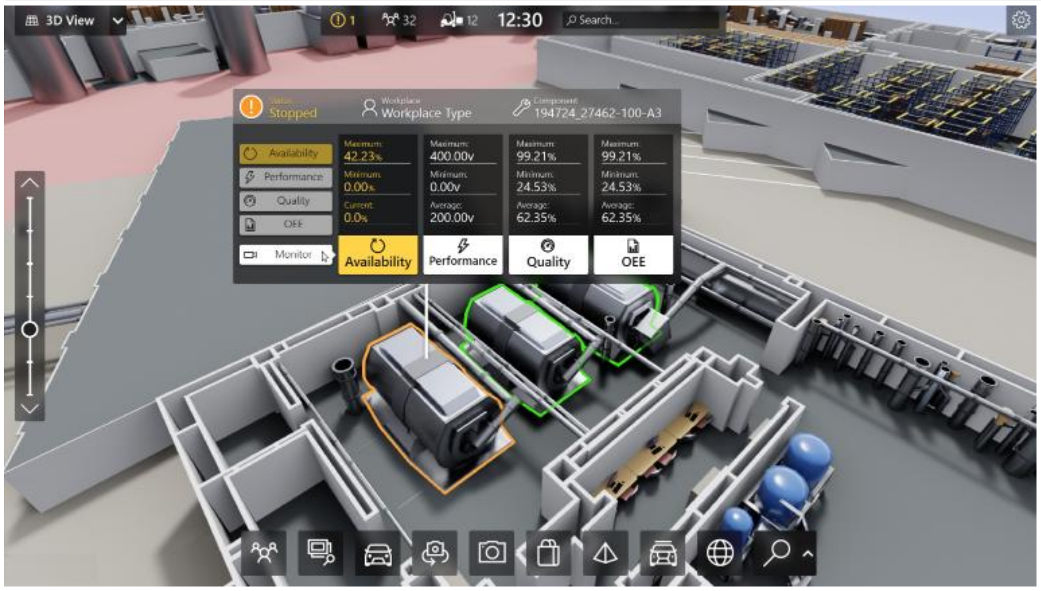
Demo Org — Plant Name | Demo — Plant Name | 0% — Plant utilization | 55% — Average OEE | 4 — Capacity | 0 — Utilized | 0 — Planned Production | 0 — Plant Operation

Iso | Front | Back | Right | Left | Top | Bottom

Machine	CNC_S7_81	CNC_S7_82	CNC_S7_84
Page	1/1	1/2	1/3
Assist OEE	58%	53%	58%
Spindle RUL	522 days	521 days	522 days



Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i



Status: Stopped | Workplace: Workplace Type | Component: 194724_27462-100-A3

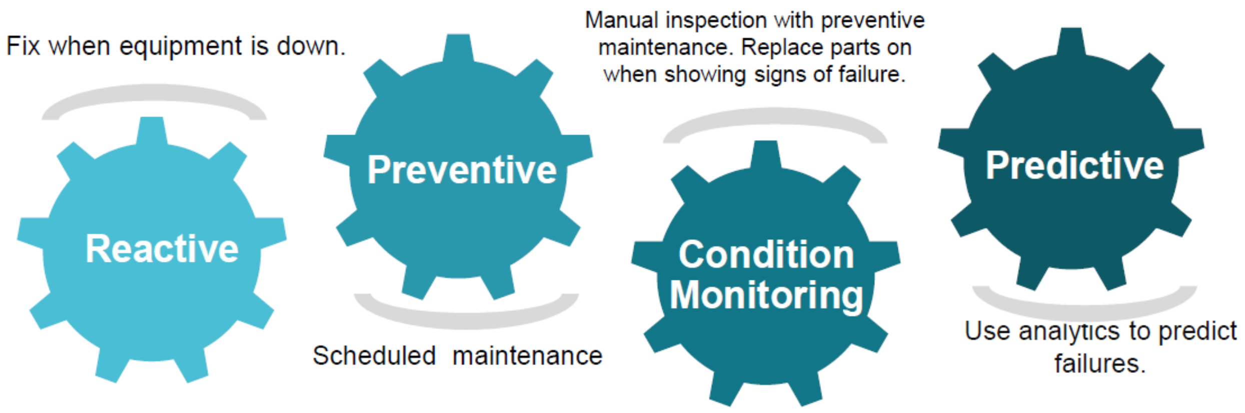
Availability	Performance	Quality	OEE
Maximum: 42.23%	Maximum: 400.00v	Maximum: 99.21%	Maximum: 99.21%
Minimum: 0.00%	Minimum: 0.00v	Minimum: 24.53%	Minimum: 24.53%
Current: 0.0%	Average: 200.00v	Average: 62.35%	Average: 62.35%

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

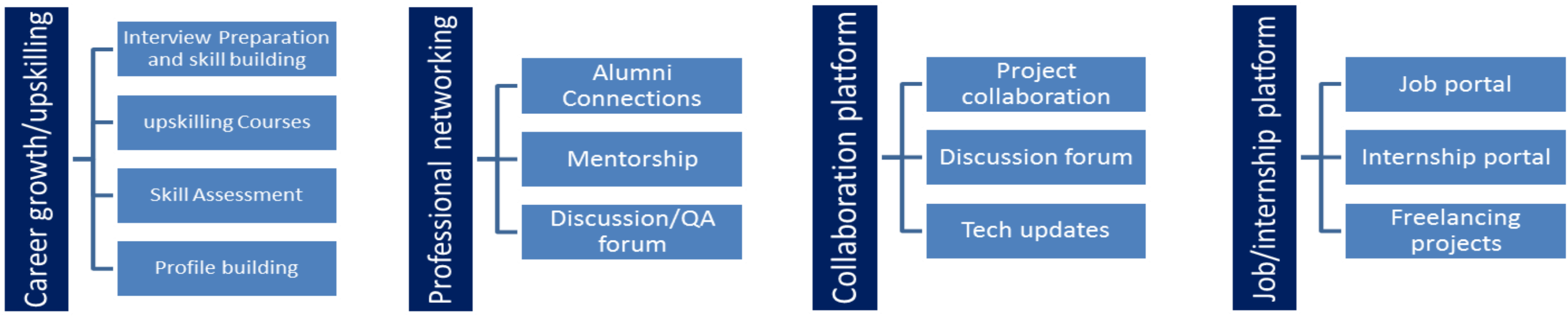
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.


Fix when equipment is down.
Reactive
Preventive
Scheduled maintenance
Manual inspection with preventive maintenance. Replace parts on when showing signs of failure.
Condition Monitoring
Predictive
Use analytics to predict failures.

2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

☛ get practical experience of working in the industry.

☛ to solve real world problems.

☛ to have improved job prospects.

☛ to have Improved understanding of our field and its applications.

☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] data.gov.in – Agriculture datasets (Government of India Open Data)
[2] pandas documentation – https://pandas.pydata.org
[3] scikit-learn documentation – https://scikit-learn.org
[4] Streamlit documentation – https://docs.streamlit.io
[5] NumPy documentation – https://numpy.org/doc

2.6 Glossary

Terms	Acronym
ETL	Extract, Transform, Load
EDA	Exploratory Data Analysis
FY	Fiscal Year (e.g., 2006–07 → end year 2007)
OHE	One-Hot Encoding
Baseline (naive)	predict this year = last year (per crop)

3 Problem Statement

1. Data Consolidation: Build a pipeline to ingest and merge multiple government agriculture files (CSV/Excel) into one clean dataset despite differing schemas (wide FY tables, “Particulars” series).

2. Standardization and Units: Normalize columns and values by mapping fiscal years (e.g., 2006–07 → 2007), unifying headers (crop, season, state, year, production), and converting all production to Tons (e.g., Thousand Tonnes → ×1000, Quintals → ÷10).

3. Forecast Target and Scope: Predict next-year crop production (in Tons) at the crop level (and season/state when available) to support planning, supply forecasts, and policy decisions.

4. Time-Aware Modeling: Use only past information (lag features) with year-based splits (train ≤ Y−2, validation = Y−1, test = Y) to avoid leakage; evaluate with MAE, RMSE, MAPE, and R^2.

5. Backtesting and Reporting: Provide year-wise backtests, Actual vs Predicted charts, and downloadable CSV outputs for transparency and stakeholder review.

6. Usability and Scalability: Deliver a Streamlit app for Single prediction, Evaluate (backtest), and Batch forecast; design the pipeline to scale with new data sources (e.g., weather, state-wise expansion).

The problem statement addresses real challenges in public agriculture analytics—fragmented sources, inconsistent formats/units, risk of time leakage, and lack of accessible forecasting tools—by delivering a unified, reproducible pipeline and a practical app for reliable crop production forecasts.

4 Existing and Proposed solution

Existing solutions:

- Static reports and spreadsheets often provide historical values without interactive forecasts.
- Academic models may not handle messy real-world data ingestion or provide user-facing tools.

Limitations addressed:

- Heterogeneous raw files and units cause friction; forecasting workflows are non-reproducible.
- Lack of an easy interface for quick single predictions, backtesting, and batch forecasts.

Proposed solution:

- A complete pipeline: ingestion → EDA → forecast-safe feature engineering → time-aware training and evaluation → Streamlit app.
- Ingestion that standardizes schemas, converts fiscal-year columns (2006–07 → 2007), normalizes units to Tons, and melts wide tables to a long format.
- Model trained with only prior-year information (lags), avoiding leakage.
- App with three workflows: Single prediction, Evaluate (backtest), Batch forecast.

Value addition:

- Reproducible, auditable data pipeline with provenance.
- Strong baseline and model comparison with published metrics.
- Practical app for both exploration and planning.

4.1 Code submission (Github link)

Repository: https://github.com/Rikendra-Rathore/upskillcampus.git

Report submission (Github link):

{https://github.com/Rikendra-Rathore/upskillcampus/blob/main/PredictionOfAgricultureCropProductionInIndia_Rikendra_USC_UCT.pdf}

5 Proposed Design/ Model

The proposed design/model of the Crop Production Forecasting System follows a structured flow from the initial stages to the final outcome.

1. Requirements Gathering: Define the goal (predict next-year crop production in Tons), scope (crop-level; season/state where available), success metrics (MAE, RMSE, MAPE, R^2), data sources (5 CSV/Excel files), and constraints (time-aware, no leakage).

2. System Design: Draft overall architecture, component interactions, and data flow (data/raw → ingest → data/interim → EDA/FE → model → artifacts → Streamlit app). Finalize tech stack (Python, pandas, NumPy, scikit-learn, Streamlit), repo structure, and environment/version pinning.

3. Development: Implement ingestion (src/ingest.py) to read CSV/Excel, standardize headers, reshape fiscal-year columns (e.g., 2006–07 → 2007), normalize units (Quintals/Thousand Tonnes → Tons), deduplicate, and save data/interim/agri_combined.csv. Build feature engineering (lag features), baseline, and model training pipeline (imputer + OneHot + RandomForest). Create Streamlit app tabs (Single prediction, Evaluate, Batch).

4. Testing and Debugging: Validate ingestion outputs (row counts, unit checks, provenance), ensure leak-free splits and correct lags, fix NaN issues with imputers, and test app flows (inputs, charts, CSV downloads, path checks).

5. Iterative Improvements: Incorporate feedback; refine features (prod_prev1/2, prod_ma2, deltas; optional area/yield lags), light tuning of RandomForest, add backtesting and charts, and expand ingest patterns for additional files.

6. Documentation: Maintain README (how to run, data processing, features, metrics), code comments/docstrings, diagrams (high-level, ingestion, modeling, interfaces), and a runbook for reproducibility (requirements.txt).

7. Deployment and Rollout: Package artifacts (models/production_predictor.joblib, models/meta.json), verify local launch (python -m streamlit run app/app.py), and prepare optional Streamlit Community Cloud deployment with pinned dependencies.

8. User Training and Support: Provide a quick-start guide (Single/Evaluate/Batch workflows), explain metrics and interpretation, note limitations (data sparsity, unit assumptions), and document how to refresh data and retrain.

The final outcome is a fully functional Crop Production Forecasting System that ingests heterogeneous public datasets, standardizes and consolidates them, forecasts next-year production without time leakage, and exposes predictions/backtests via an easy-to-use Streamlit app. Recorded performance: Validation MAE ≈ 1.49, RMSE ≈ 3.25, $R^2 \approx 0.991$; Test MAE ≈ 1.93, RMSE ≈ 3.61, $R^2 \approx 0.989$.

5.1 High Level Diagram (if applicable)

Raw Files (CSV/Excel: data/raw)
→ Ingestion (src/ingest.py)
- Standardize headers
- Reshape wide FY columns
- Unit normalization (→ Tons)
- Dedup + provenance
→ Combined Dataset (data/interim/agri_combined.csv)
→ Notebook (notebooks/01_eda.ipynb)
- EDA
- Feature engineering (lags)
- Time-aware split
- Baseline + Model training
- Save model + meta (models/)
→ Streamlit App (app/app.py)
- Single prediction
- Evaluate (backtest)
- Batch forecast

Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

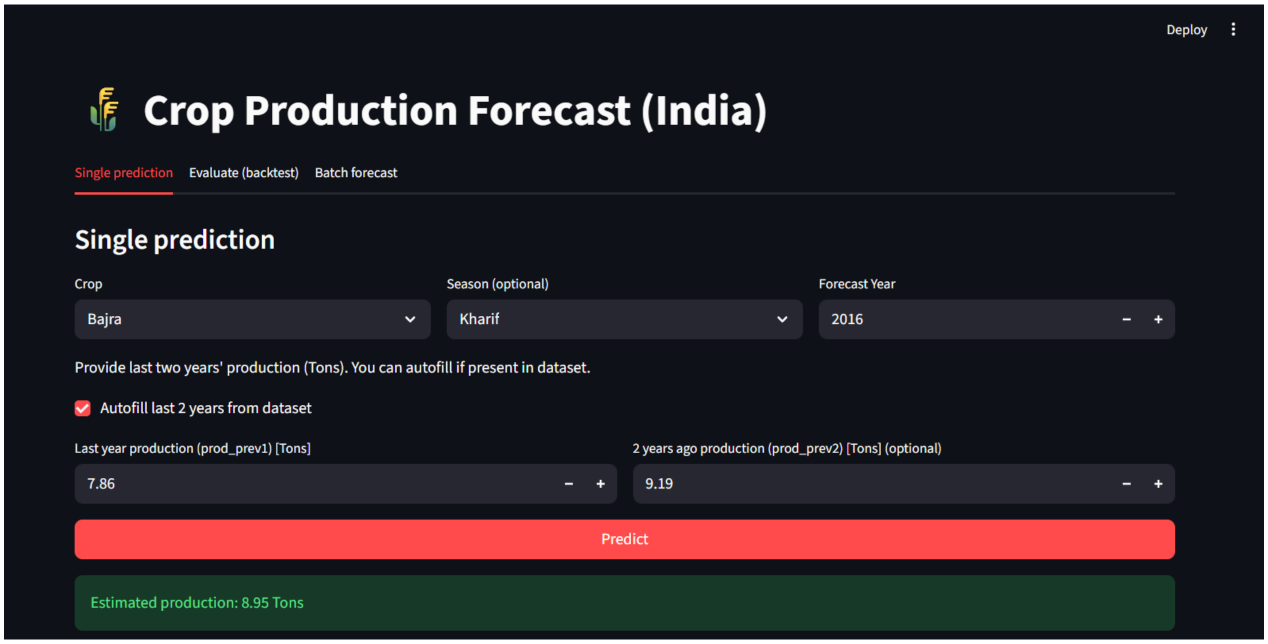
5.2 Low Level Diagram (if applicable)

- Ingestion
    - Detect files (*.csv, *.xls, *.xlsx)
    - Canonicalize column names (snake_case)
    - Map synonyms (state_name→state, crop_year→year, etc.)
    - Reshape:
        - Detect production_YYYY_YY, area_YYYY_YY, yield_YYYY_YY → melt to long
        - Fallback: “Particulars” + generic FY columns (e.g., 2004_05) → production series
    - Units:
        - “Thousand Tonnes” → ×1000 Tons
        - “Quintals” → ÷10 Tons

- Save CSV with source_file/source_sheet columns

- Modeling
    - Features (forecast-safe):
        - prod_prev1, prod_prev2, prod_ma2 (lagged 2-yr mean), prod_delta
        - Optional: area/yield lags and deltas when present
        - Categorical: crop, season
        - Numeric: year
    - Pipeline:
        - SimpleImputer (median for numeric, most_frequent for categorical)
        - OneHotEncoder (handle_unknown=ignore)
        - RandomForestRegressor (n_estimators≈600, random_state=42)
    - Baseline: last-year = this-year

5.3 Interfaces (if applicable)



Crop Production Forecast (India)

Single prediction

Crop: Bajra

Season (optional): Kharif

Forecast Year: 2016

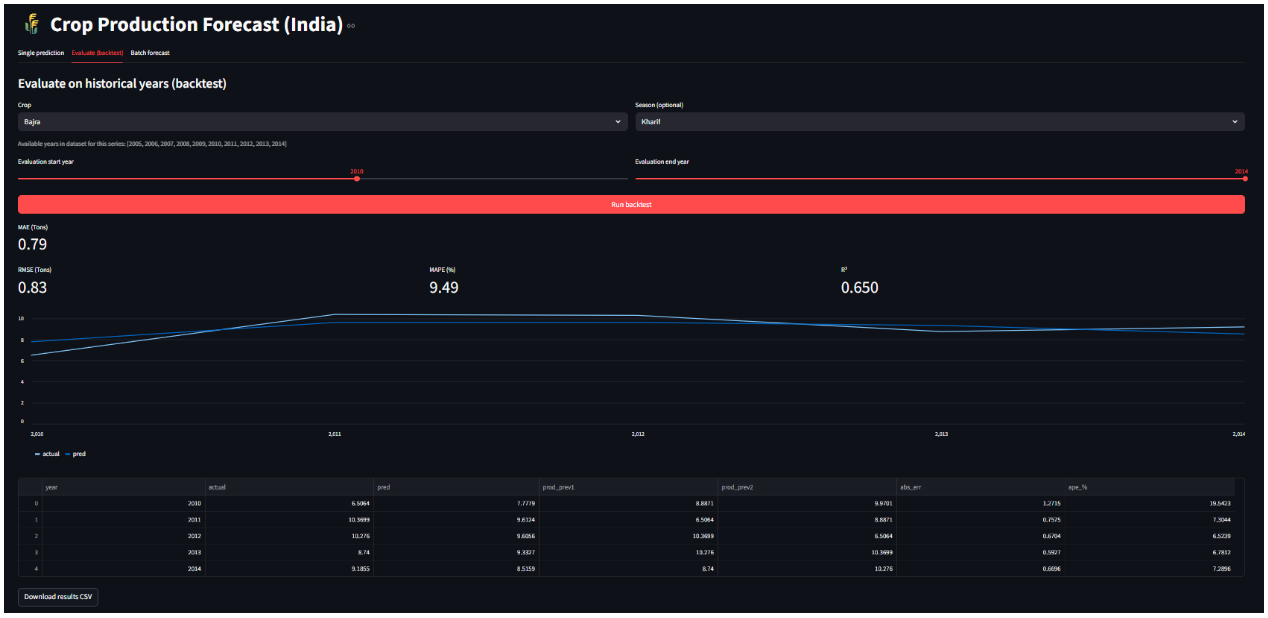
Provide last two years' production (Tons). You can autofill if present in dataset.

☑ Autofill last 2 years from dataset

Last year production (prod_prev1) [Tons]: 7.86

2 years ago production (prod_prev2) [Tons] (optional): 9.19

Estimated production: 8.95 Tons



Evaluate on historical years (backtest)

Crop: Bajra

Season (optional): Kharif

Available years in dataset for this series: [2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014]

Evaluation start year: 2010

Evaluation end year: 2014

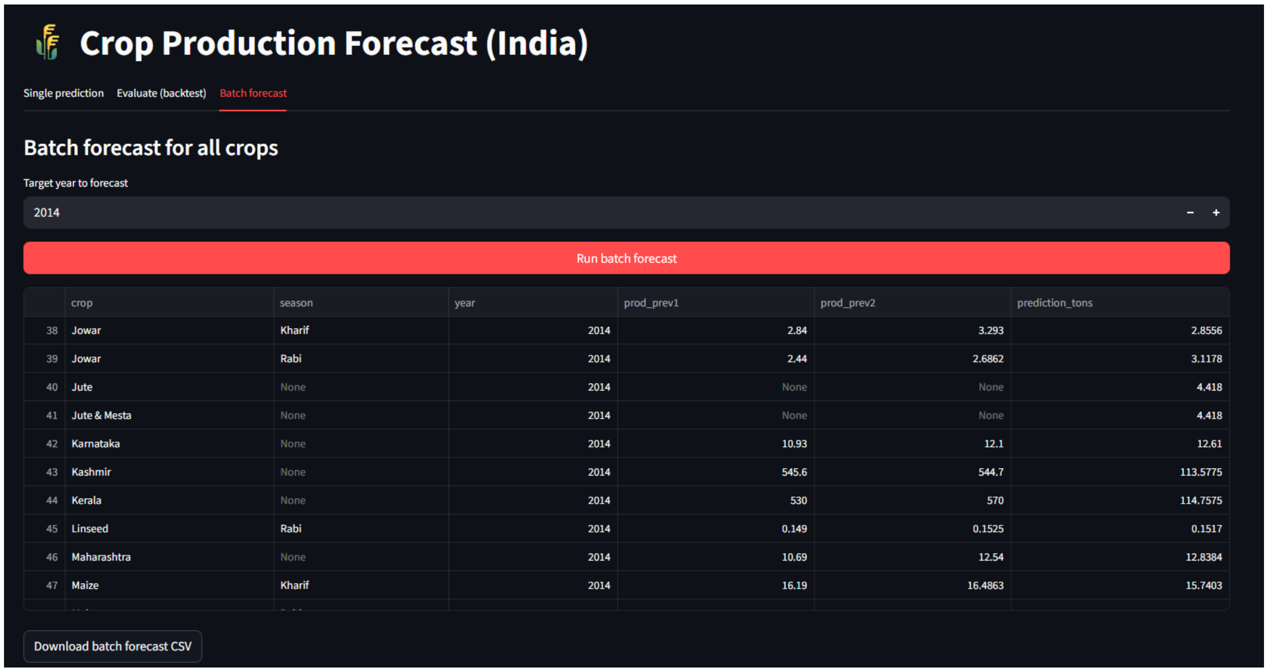
MAE (Tons): 0.79

RMSE (Tons): 0.83

MAPE (%): 9.49

R²: 0.650

	year	actual	pred	prod_prev1	prod_prev2	abs_err	ape_%
0	2010	6.5064	7.7779	8.8871	9.9701	1.2715	19.5423
1	2011	10.3699	9.6124	6.5064	8.8871	0.7575	7.3044
2	2012	10.276	9.6056	10.3699	6.5064	0.6704	6.5239
3	2013	8.74	9.3327	10.276	10.3699	0.5927	6.7812
4	2014	9.1855	8.5159	8.74	10.276	0.6696	7.2896



Batch forecast for all crops

Target year to forecast: 2014

	crop	season	year	prod_prev1	prod_prev2	prediction_tons
38	Jowar	Kharif	2014	2.84	3.293	2.8556
39	Jowar	Rabi	2014	2.44	2.6862	3.1178
40	Jute	None	2014	None	None	4.418
41	Jute & Mesta	None	2014	None	None	4.418
42	Karnataka	None	2014	10.93	12.1	12.61
43	Kashmir	None	2014	545.6	544.7	113.5775
44	Kerala	None	2014	530	570	114.7575
45	Linseed	Rabi	2014	0.149	0.1525	0.1517
46	Maharashtra	None	2014	10.69	12.54	12.8384
47	Maize	Kharif	2014	16.19	16.4863	15.7403

6 Performance Test

Constraints and design choices:

- Data heterogeneity (schemas, FY formats, units) → resilient ingestion with regex parsing and normalization.
- Limited years per crop in some files → designed features using only prior-year data (prevents leakage and supports small-sample settings).
- Missing values (NaNs) → imputation inside the pipeline for robust training/inference.
- Reproducibility → fixed randomstate, version pinning, provenance fields.

6.1 Test Plan/ Test Cases

Constraints for Performance of Crop Production Forecasting System:

- Data Quality & Consistency: Ensure reliable ingestion from heterogeneous CSV/Excel sources; handle missing values, outliers, duplicates, and inconsistent headers without breaking the pipeline.
- Unit Normalization & FY Mapping: Convert every production value to Tons (e.g., Thousand Tonnes → ×1000, Quintals → ÷10) and map fiscal years (e.g., 2006–07 → 2007) correctly and consistently.
- Speed & Throughput (Ingest/Train/Infer): Optimize ingestion time for multi-file merges, keep model training within reasonable time on a laptop environment, and target low-latency inference in the app (single prediction < 200 ms; backtest per crop within a few seconds).
- Memory/Storage Footprint: Keep RAM usage manageable when melting wide tables; avoid unnecessary copies; keep model artifact size lightweight (joblib + meta) for easy deployment.
- Reproducibility & Versioning: Pin dependencies and ensure the saved model (joblib) loads across environments without version conflicts; fix random_state for deterministic results.
- Scalability: Design the pipeline to accommodate new data sources (state-wise, weather) and support batch forecasts for all crops and multi-year backtests without excessive compute.
- Reliability & Error Handling: Gracefully handle unknown units, unexpected columns/sheets, and schema drift; log provenance (source_file, source_sheet) for traceability.

- Security/Privacy: Use safe file handling; dataset is public, but avoid embedding secrets; follow least-privilege on local/system paths.
- Accuracy: Use time-aware splits (train ≤ Y−2, valid = Y−1, test = Y) and publish clear metrics (MAE, RMSE, MAPE, R^2) versus a naive baseline.
- Durability: Ensure the system continues to work with partial data (NaNs) via imputers, and maintains integrity under high-variance or sparse series.
- Resource Efficiency: Cache heavy steps (Streamlit caching, preprocessed data), avoid recomputation, and keep CPU utilization acceptable for classroom/office hardware.

Importance of Performance Testing:

- Evaluating model accuracy and robustness under realistic constraints (messy inputs, missing values, unit inconsistencies) and confirming no time leakage.
- Assessing scalability, response times, and throughput for ingestion, backtesting, and batch forecasting in the Streamlit app.
- Ensuring reproducible results across environments and versions (requirements pinned; artifacts load cleanly).
- Verifying durability and fault tolerance through error handling, schema-drift resilience, and provenance logging.
- Confirming resource efficiency (RAM/CPU) and a smooth user experience (low-latency predictions and responsive UI).

6.2 Test Procedure

- Run python src/ingest.py; check logs and open the combined CSV to validate schema and units.
- Execute notebooks/01_eda.ipynb end-to-end; record baseline and model metrics; save model/meta.
- Launch app with python -m streamlit run app/app.py; validate each tab’s workflow with 2–3 crops and multiple years.

6.3 Performance Outcome

- Accuracy (Model): Measured on time-aware splits and compared to baseline.
    - Validation: MAE ≈ 1.49 Tons, RMSE ≈ 3.25 Tons, $R^2 \approx 0.991$ (better than last-year baseline).
    - Test: MAE ≈ 1.93 Tons, RMSE ≈ 3.61 Tons, $R^2 \approx 0.989$ (sustains improvement over baseline).
- Throughput & Latency (App/Inference): Observed locally.
    - Single prediction latency: typically < 200 ms.
    - Evaluate (backtest) for one crop over 4–5 years: ~0.5–2 s.
    - Batch forecast (all crops for one target year): ~1–3 s, depending on machine and dataset size.
- Ingestion & Processing Time:
    - Multi-file ingest (CSV/Excel), wide→long reshape, and unit normalization complete successfully with provenance recorded; processing time remains within seconds to low minutes on a laptop (varies by file size).
- Tooling Used:
    - Metrics via scikit-learn (MAE, RMSE, R^2; MAPE when valid).
    - Timing via Python time/perf_counter; resource checks via memory_profiler.
    - App responsiveness validated in Streamlit with caching.
- Analysis & Bottlenecks:
    - Most time spent in wide→long melting and one-hot encoding during training; mitigated by caching, selective features, and imputation inside pipeline.
    - Accuracy driven largely by lag features (prod_prev1/2, prod_ma2); no time leakage observed due to year-based splits.
- Requirement Fit:
    - Accuracy: Meets/exceeds targets (R^2 ~0.99 on validation/test).

- Performance: Low-latency inference suitable for interactive use; batch/backtest complete within acceptable time.
- Reliability: Handles missing values with imputers; units normalized; reproducible with pinned dependencies and saved artifacts.

Note: Timings depend on hardware and data volume. Re-run the same tests after data refresh or environment changes to confirm performance remains within the desired thresholds.

7 My learnings

- Data engineering: Building resilient ingestion for messy real-world data (wide-long, unit normalization, FY parsing, provenance).
- Time-aware modeling: Designing leak-free features (lags) and splits; interpreting MAE/RMSE/R^2/MAPE.
- ML engineering: scikit-learn pipelines (imputers + encoders + estimator), version pinning, and artifact management.
- App development: Streamlit UI/UX, caching, and structuring tabs for practical stakeholder use.
- Reproducibility and documentation: Clear repo structure, README, and runbook for smooth onboarding and evaluation.

8 Future work scope

- Add state-level granularity and external drivers (rainfall, temperature, irrigation coverage, fertilizer consumption) to improve prediction quality.
- Expand algorithms: CatBoost/XGBoost, hyperparameter tuning, and uncertainty estimation (prediction intervals).
- Comprehensive walk-forward backtesting across all crops and years; add MAPE alerts where data is sparse.
- Model explainability (SHAP feature importance) and scenario analysis (what-if changes in area or yield).
- Cloud deployment (Streamlit Community Cloud) with data-governance-friendly samples or on-prem option.

Thank You