■ Market Price Realtime Analysis & Prediction Dashboard App

> **Empowering Farmers with Realtime Market Insights, AI-Driven Forecasting, and Q&A Bot**

■ Project Overview

This ongoing project aims to provide a centralized, easy-to-use web platform for farmers, delivering **live agricultural market prices** (via API), advanced **AI-powered price predictions**, and a helpful AI chatbot for instant answers on market trends—all powered by robust, scalable modern tech.

■ About the Dataset

- **Scope:** Daily prices of agricultural commodities in India, 2001–2025
- **Records:** 75,000,000+ rows | **Commodities:** 374 | **Varieties:** 1,504 | **Markets:** 1,500+
- **Included:** Vegetables, fruits, grains, spices, & more—across every Indian state
- **Source:** Govt. of India Open Data Platform (GODL-India License)

■ Column Schema

Column Name	Description	Type
State	Indian state of the market	String
District	District of the market	String
Market	Name of the mandi (wholesale market)	String
Commodity	Name of the commodity	String
Variety	Specific type/variety	String
Grade	Quality grade (e.g., FAQ, Medium, Good)	String
Arrival_Date	Date (YYYY-MM-DD, ISO 8601)	Date
Min_Price	Minimum price (INR/quintal)	Decimal
Max_Price	Maximum price (INR/quintal)	Decimal
Modal_Price	Most frequent price (INR/quintal)	Decimal
Commodity_Code	Unique commodity code	Numeric

- **Usage:** Time-series analysis, forecasting, supply chain studies, market visualization, policy research

---

■■ Real-time Data API

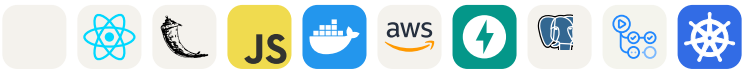
Along with historical data, the dashboard will use the Government of India's official API for continuous real-time price feeds—ensuring the dashboard is always up-to-date for farmers and stakeholders.

■■ Project Structure

```
market-price-realtime-analysis-prediction-dashboard-app/
■■■ src/                        # Main application code
■■■ test_code/                  # Test scripts & validation
■■■ data_processing_pipeline.py # RAM-friendly batch ETL pipeline
■■■ inverse_data_processing_pipeline.py
■■■ row_data_conversion.py
```

```
■■■ solid_model.ipynb          # ML experiment notebook
■■■ requirements.txt
■■■ config.json
■■■ README.md
```

■■ Technologies & Skills



Python | React/JS | Flask | FastAPI | Docker | AWS EC2/S3 | Kubeflow | GitHub Actions | PostgreSQL

AI / ML: - Pandas, NumPy, Scikit-learn, TensorFlow - Model for price prediction per commodity (feature input: State, District, Market, Commodity, Variety, Grade, Arrival_Date, Min_Price, Max_Price, Modal_Price, Commodity_Code)

Web & API: - Flask + FastAPI backend, React .js SPA frontend - Hosted on AWS EC2 (scalable, cloud-ready, secure)

Deployment / DevOps: - Docker containerization for portability - Kubeflow for full ML/data pipelines orchestration - **GitHub Actions** automates testing & pipeline checks on every push

AI Assistant Bot: - Built on Langchain, prompt engineering for robust natural language Q&A

■ Key Challenge & Smart Solution

> **Problem:**
> **75M+ rows (7–8 GB!!)**: Traditional approaches caused **RAM** crashes and slowdowns — especially when cleaning, converting, and prepping data for ML.

My Solution:
- **Chunk-based ETL:** Used S3 for cloud storage and streamed data in manageable batches to avoid "out-of-memory" errors. - Processed only the most recent 15 years (~2.5 GB, row-sampled). - Wrote cleaned batches back to S3, keeping processing fast, modular, and reproducible—no matter the hardware.

> *"Large data? No sweat—custom batch pipelines + cloud streaming for scalable, failure-proof analytics."* ■

■ Pipeline & Modules

1. **Data Processing Pipeline** (`data_processing_pipeline.py`)
2. Batch reads/processes huge rows in RAM-safe chunks—S3 in, S3 out!
3. **Inverse Data Processing** (`inverse_data_processing_pipeline.py`)
4. Restores original structure from processed data, supporting validations & explainability.
5. **Raw Data Conversion** (`row_data_conversion.py`)
6. Preps data for ML by transforming, cleaning, and encoding core columns.
7. **ML Model Pipeline** (`solid_model.ipynb`)
8. Full EDA, feature engineering, training, validation, and export for price prediction.
9. **AI Query Bot**
10. Rapid, language-based Q&A for anything a farmer wants to ask.
11. **Web Dashboard**
12. **Frontend:** React SPA for intuitive access
13. **Backend:** Flask APIs for both live and historic data
14. **Automation & MLOps**

15. **GitHub Actions:** Runs CI and automated pipeline testing on every code push
16. **Docker/Kubeflow:** For pipelined, reproducible deployment and scaling on AWS EC2

■ Coming Soon

- Live streaming of prices from official API
- Advanced, commodity-specific forecasting models
- Full-featured AI bot
- Improved maps/visual charts for better farmer experience
- User portals for feedback/community input

Many new features and improvements are still in development!

❤■ Contribution

Love agri-tech or have feedback?
Star, follow, and contribute!
We welcome contributors in ML, backend/frontend, and data engineering alike.

Made with ❤■ by Vijay Takbhate | Building for impact & community