ANNAPURTI

Grain Consumption Forecasting System

A Machine Learning Solution for Predicting Monthly Grain Demand at Fair Price Shops in the Public Distribution System

What This System Does:

This system uses historical transaction data to predict how much grain (rice, wheat, sugar) will be needed at each Fair Price Shop for the next month. It helps government agencies and supply chain managers plan inventory, logistics, and budgets effectively.

Document Purpose: Complete Project Guide
Audience: Technical and Non-Technical Stakeholders
System Version: Production-Grade Implementation

Table of Contents

1. Problem Statement

The Challenge

India's Public Distribution System (PDS) provides subsidized food grains to millions of beneficiaries through a network of Fair Price Shops (FPS). The Annapurti system is a smart card-based grain distribution platform that tracks who receives what grain, when, and how much.

The Core Problem:

How much grain should be stocked at each Fair Price Shop next month?

Without accurate predictions, shops may:
- Run out of grain (leaving beneficiaries without food)
- Over-stock grain (leading to wastage and storage costs)
- Misallocate transportation resources

Why This Matters

Reason	Impact
Supply Planning	Know exactly how much grain to stock at each FPS
Logistics	Plan transportation routes and schedules efficiently
Budget Management	Forecast expenses and allocate funds properly
Reduce Wastage	Minimize over-stocking that leads to spoilage
Beneficiary Satisfaction	Ensure grain is available when people need it

The Solution Approach

This system uses **Machine Learning** to analyze historical grain distribution patterns and predict future demand. Instead of relying on manual estimates or simple averages, the system learns complex patterns from data — including seasonal variations, household characteristics, and historical consumption trends.

Key Prediction Levels Supported:

- **Per Smart Card:** Predict consumption for each individual household
- **Per Fair Price Shop:** Total demand for each distribution point
- **Per Commodity:** Separate predictions for rice, wheat, and sugar
- **Combined:** FPS × Commodity for detailed supply planning

2. System Overview

The Annapurti Forecasting System is designed as a complete end-to-end solution that takes raw transaction data and produces actionable predictions. Here's how it works at a high level:

Step	What Happens	Output
1. Data Input	Load transaction history, member details, card info	Clean, structured data
2. Transform	Convert transactions into monthly consumption per card	Monthly panel dataset
3. Features	Create predictive variables from historical patterns	25+ feature columns
4. Train	Machine learning algorithm learns patterns	Trained prediction model
5. Predict	Apply model to generate next month's forecast	Predictions (CSV)

Key Concepts Explained

Smart Card	A unique identifier for each household that receives grain benefits.
Fair Price Shop	Government-authorized shops where beneficiaries collect grain rations.
Allotment Month	The month FOR WHICH the grain is allocated (not when collected).
Commodity Code	Numeric code for grain type: 41=Rice, 43=Wheat, 45=Sugar.
Card Category	Priority level: A (Antyodaya), PH (Priority), S (State).

3. Machine Learning Overview

This section explains the technologies and algorithms used in the system. Each component plays a specific role in transforming raw data into accurate predictions.

Libraries and Their Roles

Library	What It Does	Why We Use It
pandas	Handles tabular data (like spreadsheets)	Read CSV, filter, group, merge data
numpy	Fast mathematical operations	Calculate statistics efficiently
scikit-learn	Machine learning basics	Evaluate model accuracy
LightGBM	Gradient Boosting algorithm	Main prediction engine

The Core Algorithm: Gradient Boosting

The heart of this system is a technique called **Gradient Boosting**. Here's a simple way to understand how it works:

How Gradient Boosting Works: Each tree corrects the errors of previous trees

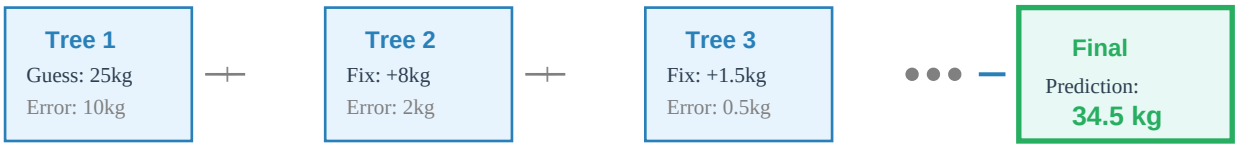

Figure: Gradient Boosting combines multiple simple predictions into one accurate result

Simple Explanation: Instead of one person making a guess, you have a team of 100 experts (called "trees"). Each expert looks at the previous experts' mistakes and tries to fix them. The final prediction combines all opinions for much better accuracy than any single expert could achieve.

4. Data Understanding

The system uses three input files that together provide a complete picture of grain distribution patterns.

File 1: Transaction Data

This is the main data file containing all grain distribution records. Each row represents one transaction — a beneficiary collecting grain for a specific month.

Dataset Statistics: 18,909 transactions | Date Range: Jan 2025 - Mar 2026 | Unique Cards: 4,946 | Commodities: Rice (41), Wheat (43), Sugar (45)

Column	Meaning	Example
card_no	Smart card number	02061012291
ALLOTMENT_MONTH	Month for which grain is allocated	2025-10-01
COMMODITY_CODE	Type of grain	41 (Rice)
qty	Quantity in kilograms	35.00
home_fps	Home Fair Price Shop ID	6257

File 2: Member Details

Contains information about each family member linked to a smart card. This helps understand household composition — number of members, ages, and gender distribution.

Dataset: 15,851 members across 4,932 cards (average: 3.2 members per household)

File 3: Card Details

Card Categories and Entitlements:

- **A (Antyodaya)**: Poorest of the poor — approximately 34 kg/month average
- **PH (Priority Household)**: Below poverty line — approximately 16 kg/month average
- **S (State Priority)**: State-determined criteria — approximately 15 kg/month average

5. How the System Works

This section walks through each step of the prediction process in detail.

Step 1: Data Loading and Cleaning

The system reads the three CSV files and performs initial cleaning: parsing date columns, converting quantities to numeric values, handling missing data, and extracting date components.

Step 2: Create Monthly Panel

Raw transaction data may have multiple rows per card per month. We transform this into one row per card per month:

BEFORE: Transaction Level

Card	Date	Qty
C001	Jan-15	10
C001	Jan-20	25
C001	Feb-10	20

AFTER: Monthly Panel

Card	Month	Total	
C001	Jan-25	35	
C001	Feb-25	20	
C001	Mar-25	0	← *No transaction = 0*

Figure: Data transformation from transaction level to monthly panel format

> **Important:** Months with no transactions are filled with 0 (zero consumption). This is critical because the model needs to learn that some cards are inactive in certain months.

Step 3: Feature Engineering

We create "features" — input variables that help the model make predictions:

Category	What It Captures	Examples
Lag Features	Recent consumption history	qty 1, 2, 3 months ago
Rolling Stats	Consumption trends	avg last 3/6 months
Temporal	Time-related patterns	month, is_festival_season
Demographics	Household characteristics	num_members, avg_age

6. Running Predictions

The forecasting system supports four different prediction levels. Each level answers a different question about grain demand. Here's how to run each one and what they produce:

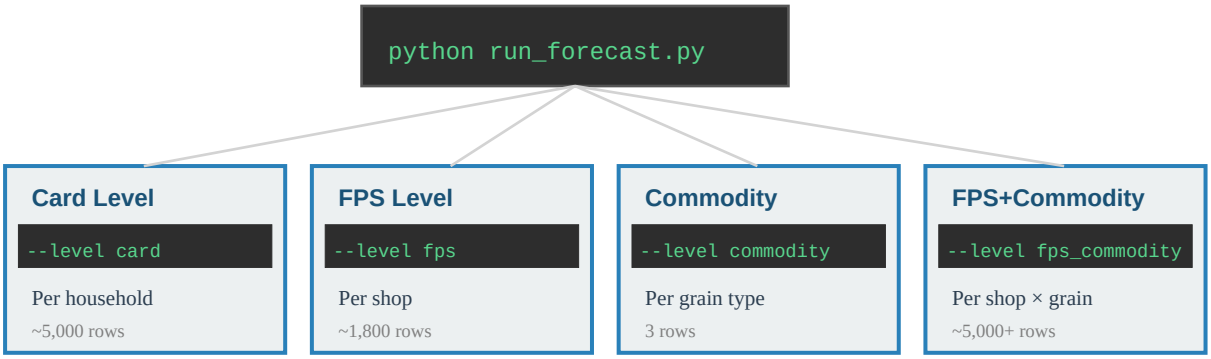

Figure: The four prediction levels supported by the system

Level 1: Card-Level Prediction (Default)

```
> python run_forecast.py --level card
```

Question Answered: "How much grain will each CARD HOLDER need next month?"

Output: One row per smart card (~5,000 rows)
Use Case: Individual beneficiary tracking, identifying high-demand households

Level 2: FPS-Level Prediction

```
> python run_forecast.py --level fps
```

Question Answered: "How much grain will each FAIR PRICE SHOP need next month?"

Output: One row per shop (~1,800 rows)
Use Case: Shop-level stock planning, logistics, supply distribution

Level 3: Commodity-Level Prediction

```
> python run_forecast.py --level commodity
```

Question Answered: "How much of each GRAIN TYPE will be needed next month (total)?"

Output: One row per commodity (3 rows: Rice, Wheat, Sugar)
Use Case: Procurement planning, warehouse stocking, budget forecasting

Level 4: FPS + Commodity Prediction

```
> python run_forecast.py --level fps_commodity
```

Question Answered: "How much of each GRAIN TYPE will each SHOP need?"

Output: One row per shop × commodity combination (~5,000+ rows)
Use Case: Detailed distribution planning — exactly what to send to each shop

Quick Comparison

Level	Command	Rows	Best For
Card	`--level card`	~5,000	Individual tracking
FPS	`--level fps`	~1,800	Shop stock planning
Commodity	`--level commodity`	3	Total procurement
FPS+Commodity	`--level fps_commodity`	~5,000+	Detailed distribution

Quick Start Commands

```
# Navigate to project folder
cd d:\Projects\Annapurti_Prediction_ALG

# Run default (card-level)
python run_forecast.py

# Run all 4 levels
python run_forecast.py --level card
python run_forecast.py --level fps
python run_forecast.py --level commodity
python run_forecast.py --level fps_commodity
```

7. Examples & Scenarios

Let's walk through some real-world scenarios to see how the system works in practice.

Scenario 1: Regular Household

Card: 02061012291 (Antyodaya category)
Household: 4 members, average age 45
History: Consistently collects 32-35 kg of rice monthly

What the model sees:
• lag_1 = 35 kg, lag_2 = 34 kg, lag_3 = 32 kg
• rolling_mean_3m = 33.7 kg | activity_rate = 100%

Prediction: 34.2 kg for next month
Interpretation: Stable household, prediction matches historical average.

Scenario 2: Irregular Collector

Card: 1041610527 (Priority Household)
Household: 2 members
History: Collects grain only every 2-3 months (in bulk)

What the model sees:
• lag_1 = 0 kg, lag_2 = 0 kg, lag_3 = 45 kg (bulk)
• activity_rate = 40%

Prediction: 0.2 kg (essentially zero)
Interpretation: Model learned irregular pattern; recent bulk collection means unlikely to collect again soon.

Scenario 3: FPS-Level Planning

Fair Price Shop: FPS #1120 | **Registered Cards:** 85 households

What the system does: Predicts consumption for each card, then sums them.

Prediction: 793.9 kg total for April 2026

Action Items:
• Stock approximately 800 kg of grain
• Plan for ~85 beneficiary visits
• Add 10-15% buffer (~80 kg) for safety stock

8. Model Performance

Understanding accuracy helps you know how much to trust predictions and account for uncertainty in planning.

Accuracy Metrics Explained

Metric	Value	What It Means
R^2 (R-squared)	0.854 (Card) 0.930 (FPS)	85-93% of variation is explained by the model. Higher is better.
MAE	1.98 kg (Card) 4.95 kg (FPS)	On average, predictions are off by about 2 kg per card.
MAPE	20.3% (Card) 21.0% (FPS)	Predictions are typically within 20% of actual values.

Is 85% R^2 Good?

Yes! For demand forecasting, anything above 70% is considered good. The 85% at card level and 93% at FPS level indicate a very reliable model.

Why FPS-level is more accurate: Individual behavior can be unpredictable, but when aggregating many households, random variations cancel out.

Most Important Features

#	Feature	Why It Matters
1	avg_age	Household age composition strongly affects consumption
2	month_sin/cos	Seasonal patterns (festivals have higher demand)
3	cumulative_avg	Historical average is a strong baseline
4	lag_1	Last month's consumption predicts this month's
5	activity_rate	Regular vs irregular collectors differ

9. Output Explanation

The system produces CSV files containing predictions that can be imported into spreadsheets, databases, or other systems.

Sample Output: Card Level

card_no	predicted_qty_kg	prediction_month
02061012291	35.5	2026-04
07020211490	22.3	2026-04
12061410125	18.7	2026-04

How to read: Card 02061012291 will need approximately 35.5 kg of grain in April 2026.

Sample Output: FPS Level

home_fps	predicted_qty_kg	prediction_month
1120	793.8	2026-04
6257	650.2	2026-04
39394	765.1	2026-04

How to read: Shop 1120 will need approximately 794 kg total grain in April 2026.

Sample Output: FPS + Commodity Level

home_fps	COMMODITY_CODE	predicted_qty_kg	prediction_month
1120	41 (Rice)	450.5	2026-04
1120	43 (Wheat)	220.3	2026-04
1120	45 (Sugar)	123.0	2026-04

How to read: Shop 1120 will need 450 kg Rice, 220 kg Wheat, 123 kg Sugar in April.

10. System Flow Diagram

The following diagram shows how data flows through the system from input to output:

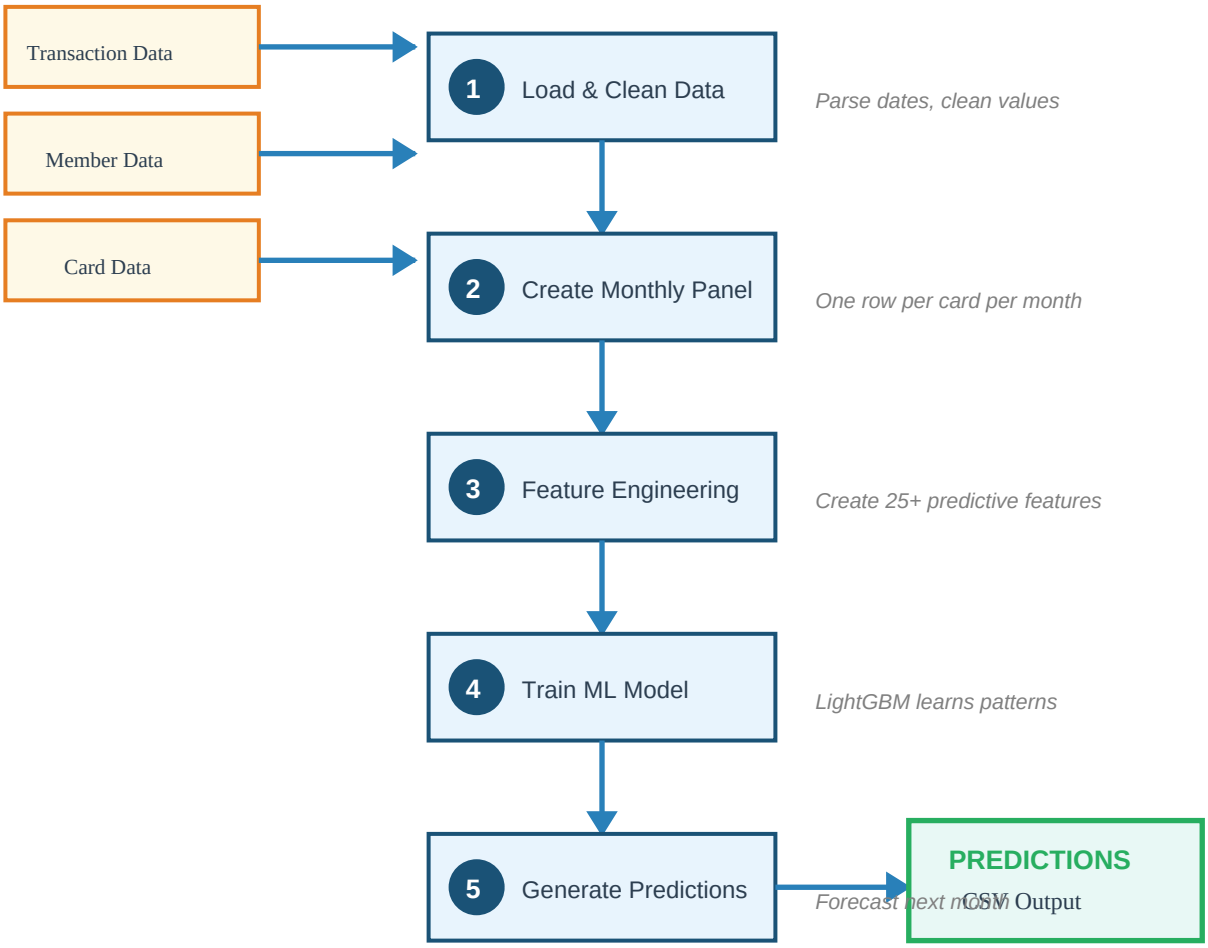

Figure: End-to-end data flow through the Annapurti Forecasting System

Feature Engineering Overview

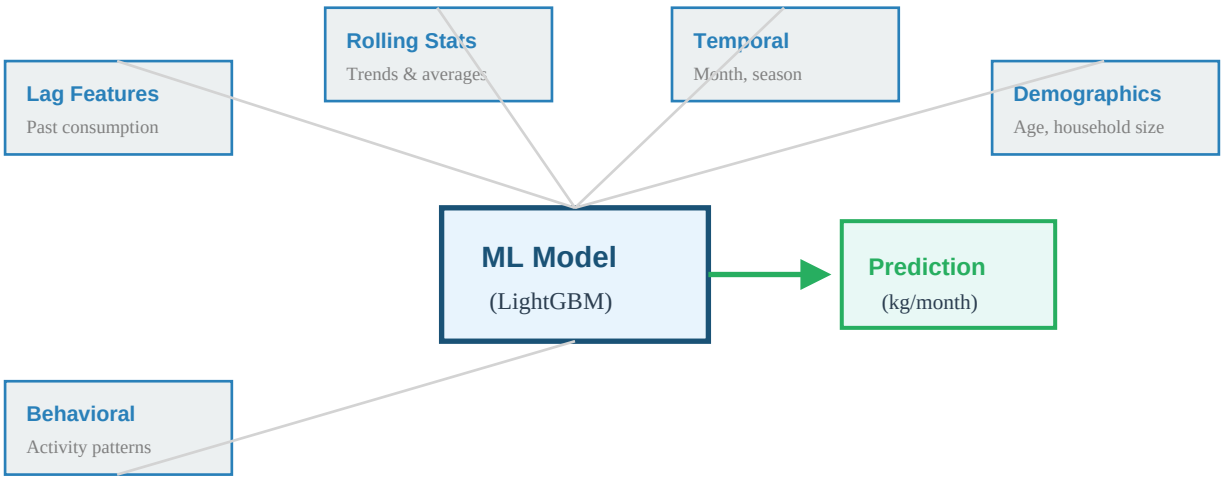

Figure: Multiple feature types feed into the ML model to generate predictions

11. Conclusion

The Annapurti Grain Consumption Forecasting System provides a powerful, data-driven approach to demand prediction in the Public Distribution System.

Key Benefits

- **Accuracy:** 85-93% prediction accuracy for reliable supply planning
- **Flexibility:** Four prediction levels (Card, FPS, Commodity, FPS+Commodity)
- **Automation:** Reduces manual estimation effort and human error
- **Scalability:** Handles thousands of cards and hundreds of FPS locations
- **Transparency:** Feature importance shows which factors drive predictions

Practical Recommendations

For Supply Chain Managers:
- Use FPS-level predictions with a 10-15% buffer for safety stock
- Monitor actual vs predicted values monthly to track performance
- Update the model quarterly with new transaction data

For Technical Teams:
- Run the forecaster at the start of each month
- Archive predictions for historical comparison
- Flag FPS locations where predictions deviate significantly

In Summary:

This system transforms historical grain distribution data into actionable predictions, enabling better resource allocation, reduced wastage, and improved service delivery. With 85%+ accuracy and support for multiple aggregation levels, it provides a solid foundation for data-driven decision making in the Public Distribution System.

For technical implementation details, code documentation, and API reference, please refer to the grain_forecaster.py source code and the accompanying guide.md file.