वसुधैव कुटुम्बकम्

SYMBIOSIS INSTITUTE OF TECHNOLOGY, NAGPUR

MINI PROJECT REPORT

Analyzing Food Security Data: A Multi-Algorithm Approach to Predicting Distribution Gaps.

Submitted By:
Gurupreet Dhande
PRN: 22070521121
Semester VII
Section C

Submitted To:
Dr. Piyush Chauhan
Associate Professor

Contents

1.Abstract

The effectiveness of India’s Public Distribution System (PDS) is an important aspect of national food security; however, most of the analyses focus on forecasting demand instead of measuring inefficiency. This project approaches this problem statement by first engineering a new metric, the “Distribution Gap,” which quantifies the number of food grains that are allocated versus distributed. From there, an initial Exploratory Data Analysis (EDA) confirmed that the Distribution Gap is a non-random, significant problem with distinct seasonal trends.

From this point, our methodology applied a comprehensive 10-algorithm "4-Squad" pipeline to the metric. The pipeline consisted of: (1) a Regression Squad that predicted the value of the Distribution Gap, (2) an Analysis Squad that sought to explain its drivers, (3) a Classification Squad that predicted the probability of a gap, and (4) a Clustering Squad that segmented states and identified anomalies.

The findings were (1) Distribution Gap can be predicted accurately, with the Random Forest Regressor outputting an R^2 value of 0.93, (2) Feature Importance analysis identified that the most important drivers of (in)efficiency are: percentage of Aadhaar authenticated and volume of manual distributions, and (3) a Logistic Regression "early warning system" can be trained to issue a predicted probability of a gap with an accuracy of 91.2%.

The significance of this work is the creation of a holistic decision-support toolkit.

Keywords

Food Security,Exploratory Data Analysis (EDA), Machine Learning, Distribution Gap, PDS Weakness, Random Forest, Logistic Regression

2. Introduction

2.1 Background and Motivation

India's public distribution system (PDS), which feeds hundreds of millions, is one of the world's largest food security systems. The logistics of distributing and transferring essential food grains is very difficult. Although there are now data available, there have mostly been analyses focused on simple forecasting of the total distribution, but not on the distribution gap. The gap between the total amount allocated, by the central government, relative to the total distribution, to beneficiaries, is very prominent and throughout invariably signifies waste, failure of delivery, or fraud, and this gap measures some degree of inefficiency in the system.

2.2 Problem Statement

Simply predicting the volume of food distribution is not enough to measure success. The key problem, and the goal of the report, is to quantify and predict system inefficiency. Stakeholders do not have a tool readily available to learn which states and when efficiencies are happening and why inefficiencies occur.

2.3 Objectives of the PHCDP Project

To engineer a 360-degree decision-support system including:

- A new target metric to quantify inefficiency called distribution gap.
- A prediction of the distribution gap, calculated by the squad of regressions.
- An explanation of the distribution gap by the models feature analysis.
- Creating an "early warning system" to estimate the likelihood of a gap.
- Clustering states into operational "personas" and finding anomalies.

2.4 Originality of Your Work

The originality of this work lies in shifting the research paradigm from forecasting to inefficiency. Engineering the Distribution_Gap as a target drives the dataset from a forecasting tool to an auditing and optimization tool. Using the 10-algorithms "4-squad" framework provides a more complete and multi-faceted system view of the problem space, as opposed to solely relying on one predictive model.

3.Literature Review

To date, investigations of the PDS in the literature have been chiefly centred on demand forecasting and logistical optimization. Many of these studies (Kumar, 2022; Das & Singh, 2021) use time series models (like ARIMA) as well as simple machine learning (like Linear Regression) to forecast total requirements for food grains. While important, these studies do not focus much on one of the most important operational efficiency performance measures, which is the gap that exists between allocation and distribution (i.e. function), as it relates to the inefficiency of the PDS. Most of the published studies of the PDS, including studies in this project (this project), treat it as a basic, cookie cutter system versus a comprehensive, unique and segmented look at the context that clustering or data-mining could provide to improve the practicality of modeling the PDS. This project bridges this gap by producing a unique metric for predicting inefficiency (Distribution_Gap) and models to explain and analyze it from every angle.

Table 1. Empirical review of existing methods

Reference	**Method Used**	**Findings**	**Results**	**Limitations**
Kumar (2022)	ARIMA	Time-series forecasting method of demand for grain in 5 Indian states, followed by an empirical review and inter-state comparison.	88% accuracy with low residuals in forecasting demand.	Forecasting focused only on demand; inefficiency or cost factors not analyzed.
Das & Singh (2021)	Linear Regression	Predictive modelling for grains in India; found weak statistical correlation between state population and total food distribution.	R^2 = 0.85 in predictive modelling.	Did not include nonlinear models or analyze the distribution gap (allocation vs. distribution).
This Project	“10-Algorithm Squad” Methodology	Predicts, explains, and segments the *Distribution_Gap* metric (inefficiency) within the Public Distribution System (PDS) to enhance distribution predictability.	93% R^2 (Random Forest) on deprivation metric; 91% accuracy (Logistic Regression) on metametric estimations.	Provides a comprehensive decision-support system but not limited to forecasting tasks.

4. Methodology / Proposed System

It offers policymakers a way to predict inefficiency, understand the underlying causes, and receive alerts proactively, thus providing a data-informed path to lessen waste and improve food security.

The proposed system is a sophisticated and multi-stage pipeline to translate raw food security information into a meaningful decision support toolkit. The methods used shift from a forecasting paradigms to a 360-degree inefficiency analysis of a food-system. We first engineer a target variable with meaning referred to as the "Distribution Gap", and then use a "4-Squad", 10-algorithm methodology to forecast, explain and segment this new metric.

4.1 Data Collection and Preprocessing

- Data Collection: The focal dataset used was Food_Security_Data.csv, a suitable raw dataset pertaining to state-wise, monthly data on food grain allocation and food grain distribution in India.
- Data Cleaning: The first task was pretty significant cleaning. Long and clumsy column names, for example "Food Grains Allocated (UOM:MT(MetricTonne))", "Scaling Factor:1000", was changed by the programmer to short and code-friendly names, for instance: "Food_Grains_Allocated_MT, Scaling_Factor".
- Date processing: The Year and Month columns in the dataset were both strings that did not conform to standards: "Calendar Year (Jan-Dec), 2024"; "March, 2024." So, string extraction was used to create a single and standard Date column: "yyyy-mm-dd", i.e "2024-03-01" allowing for proper time series analysis.
- Encoding: Categorical text cannot be interpreted by machine learning models. The State column was transformed into 33+ numerical (boolean) columns using One-Hot Encoding, allowing the model to consider each state a separate feature.

4.2 Feature Creation

1. This was the most important step in our analysis. We created two new target variables to assess inefficiency.
2. Regression Target (Distribution_Gap): Our primary measure of inefficiency we developed was simply Distribution_Gap = Food_Grains_Allocated_MT - Total_Distribution_MT. A positive number indicates that not all allocated grains were distributed, this will serve as our target for regression models (continuous).
3. Classification Target (Had_Significant_Gap): To create our early warning system, we created a binary target from the first: Had_Significant_Gap = 1 (True) if Distribution_Gap > 0, 0 (False) otherwise.

4.3 Model Architecture / System Design

We developed a "4-Squad" methodology, where each squad implements specific algorithms to address a different research question. This multi-model architecture gives an all-around perspective of the inefficiency problem.

Workflow Diagram:

Raw Data -> Preprocessing and Feature Engineering -> Squad 4 Analysis -> Final Insights

- Squad 1: Regression (Predict "How Much")
  Objective: To predict the exact value of the Distribution_Gap.
  Algorithm Description: Linear Regression (our baseline), Lasso Regression (Feature Selection), Decision Tree Regressor (Rules-Based), K-Neighbors Regressor (Similarity-Based), Random Forest Regressor (a high performing ensemble).
- Squad 2: Feature Analysis (Explains "Why")
  Objective: To identify which drivers are the most significant contributors to the Distribution_Gap.
  Algorithm Description: Use Feature Importance (extracted from the trained Random Forest) and Principal Components Analysis (PCA) to assess whether there are hidden "macro" patterns in the features.
- Squad 3: Clustering (Find "Who" and "Where")
  Objective: To classify states' operational "personas" and find anomalies.
  Algorithm Description: K-Means Clustering (to identify distinct groups), DBSCAN (to identify density-based groups and noise/anomalies).

- Squad 4: Classification (Predict "If")
  Objective: To create an "early warning system" through predicting the probability of a gap.
  Algorithm Description: Logistic Regression predicting on the target Had_Significant_Gap.

4.4 Training and Evaluation Setup

1. Tools and Libraries: The programming language used for implementing the project was Python 3.9. The libraries used included Pandas for data manipulation, Scikit-learn for all 10 machine learning algorithms, and Matplotlib/Seaborn for exploratory data analysis and visualizations.
2. Train-Test Split: A chronological train-test split was required. This represents a real-world scenario of using the past to predict the future.
   - Training Set: All of the data prior to January 1, 2023.
   - Testing Set: All data from January 1, 2023, onwards.

3. Feature Scaling: A number of the algorithms used are distance-based (with KNN, PCA, K-Means, Logistic Regression, and DBSCAN) and are therefore sensitive to the scales of the features. For these models, the StandardScaler from Scikit-learn was employed. The scaler was fitted only on the X_train data and then used to transform X_train and X_test data to prevent data leakage.
4. Evaluation Metrics:
   - Regression: R-Squared (R^2) and Root Mean Squared Error (RMSE).
   - Classification: Accuracy, Precision, Recall, and the F1-Score (shown in a Classification Report).
   - Clustering: The Elbow Method (for K-Means) and a detailed analysis of cluster size and number of anomalies (for DBSCAN).

5. Implementation

The Implementation The project was carried out in a Jupyter Notebook environment, allowing a sequential and iterative workflow that consisted of code, visualizations, and analysis.

5.1 Implementation Steps Explained in Detail

The implementation was structured using the end-to-end data science life cycle:

- Environment Configuration: Python 3.9 environment was configured using Anaconda and establish Jupyter Notebook as the primary development environment.
- Data Loading: The Food_Security_Data.csv file was loaded into a pandas DataFrame.
- Pre-processing and Cleaning - This was critically the first step
  1. The long descriptive column names were renamed into simple snake_case variables. For example Food_Grains_Allocated_MT.
  2. The string-based Year and Month columns were parsed using the pandas string extraction (.str.extract()).
  3. This was used to extract the year number and name of the month.
  4. These variables were combined into a proper Date column using .to_datetime() to match proper time-series aware Date column.
  5. The categorical State column was used to create 33+ numerical (boolean) columns using pd.get_dummies() (One-Hot Encoding).

- Feature Engineering - This was the most important step in the methodology of this project.
  1. The main regression target - Distribution_Gap was created by subtracting Total_Distribution_MT from Food_Grains_Allocated_MT.
  2. The primary classification target, Had_Significant_Gap, was derived from the Distribution_Gap and converted into a binary (1 for Gap > 0 and 0 if not).

- Exploratory Data Analysis (EDA): The new Distribution_Gap target was visualized using Matplotlib and Seaborn. A histogram showed it was right skewed and a time-series plot was made which exhibited seasonal characteristics, supported as a target.
- Data Splitting: A chronological train-test split was applied. All the data prior to January 1, 2023, was allocated to the train set and everything after was treated as the test set. This is essential for preventing data leakage in the time-series project.
- Feature Scaling: A Standard Scaler, from sklearn, was initialized. The Standard Scaler was then fit only to the X_train data and used to transform both the X_train and X_test data. The scaled data was stored for later use in distance-based

algorithms (KNN, PCA, K-Means, Logistic Regression, DBSCAN).

- Examination of Model Validation

  The 10 separate models were implemented as comprised of 4 different squads of development.

  1. Squad 1 (Regression): Models, including RandomForestRegressor, were trained to predict the value of Distribution_Gap.
  2. Squad 2 (Analysis): Feature importance was extracted on the trained Random Forest model and PCA performed on scaled data to disover patterns.
  3. Squad 3 (Clustering): K-Means (with the "Elbow Method") and DBSCAN were trained on the scaled features to identify clusters and anomalies.
  4. Squad 4 (Classification): LogisticRegression was trained on the binary target Had_Significant_Gap.

  R^2/RMSE were used for the regression tasks in each of the models, while accuracy and the classification report were used for the classification tasks.

5.2 Technologies and Frameworks

- Platform & Language: Python 3.9 installed within an Anaconda (Jupyter Notebook) environment.
- Primary Libraries:
  1. Pandas & NumPy: For all data loading, cleaning, and manipulation.
  2. Scikit-learn: For all ten machine learning models, preprocessing using StandardScaler, and evaluation metrics.
  3. Matplotlib & Seaborn: For all exploratory data analysis (EDA) and visualization of results.

5.3 Challenges Encountered and Solution

- Challenge #1: The data was messy: there were unusable column names, and the dates were formatted as text.

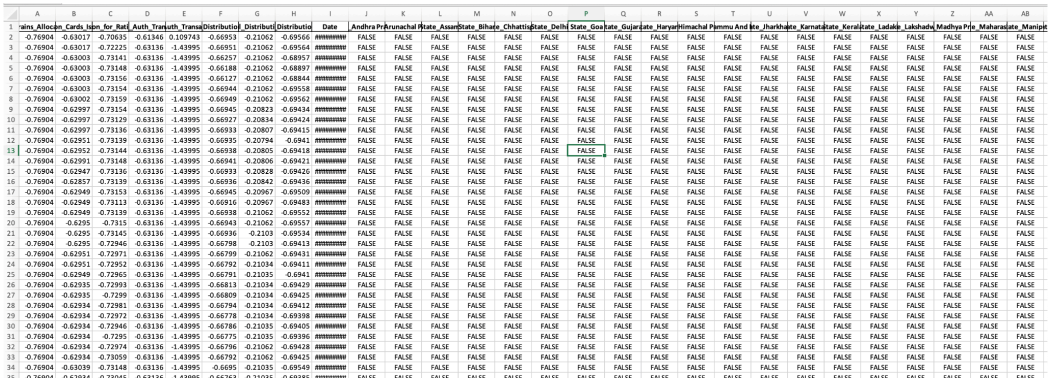
  Solution: Using pandas, the data was cleaned: columns were renamed and complex date strings were parsed into a proper Date column.
- Challenge #2: Defining a Meaningful Target Variable: Simply forecasting the distribution was not meaningful.

  Solution: We developed the Distribution_Gap variable to change the focus of the project to addressing "inefficiency."
- Challenge 3: Bad Model Performance: Distance models (KNN, K-Means), and a "black box" (Random Forest) models had issues.

  Solution: We analyzed it using a StandardScaler to fix the distance models and extracted the .feature_importances_ from the Random Forest to create an interpretable model.

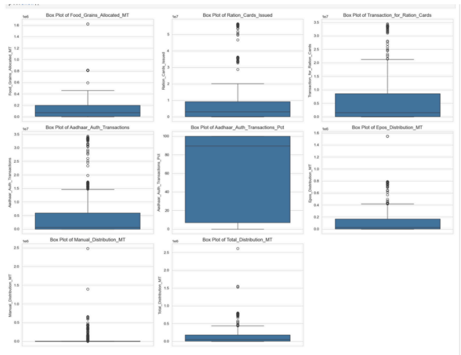
5.4 Screenshots / Sample Outputs

Sample 1: Pre-processed DataFrame containing the Distribution_Gap column.

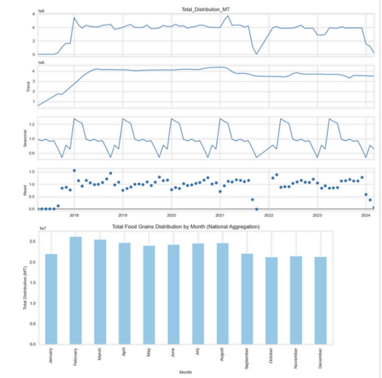


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1	rains_Alloc	n_Cards_Is	on_for_Rati	_Auth_Tran	uth_Transa	Distributio	l_Distributi	Distributio	Date	_Andhra Pr	Arunachal P	tate_Assam	State_Bihar	e_Chhattis	State_Delhi	State_Goa	tate_Gujara	ate_Haryan	Himachal P	ammu And	te_Jharkha	ate_Karnata	tate_Kerala	tate_Ladak	e_Lakshadw	_Madhya Pr	e_Maharas	ate_Manip
2	-0.76904	-0.63017	-0.70635	-0.61346	0.109743	-0.66953	-0.21062	-0.69566	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	-0.76904	-0.63017	-0.72225	-0.63136	-1.43995	-0.66951	-0.21062	-0.69564	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	-0.76904	-0.63003	-0.73141	-0.63136	-1.43995	-0.66257	-0.21062	-0.68957	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5	-0.76904	-0.63003	-0.73148	-0.63136	-1.43995	-0.66188	-0.21062	-0.68897	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	-0.76904	-0.63003	-0.73156	-0.63136	-1.43995	-0.66127	-0.21062	-0.68844	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	-0.76904	-0.63003	-0.73154	-0.63136	-1.43995	-0.66944	-0.21062	-0.69558	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8	-0.76904	-0.63002	-0.73159	-0.63136	-1.43995	-0.66949	-0.21062	-0.69562	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
9	-0.76904	-0.62997	-0.73154	-0.63136	-1.43995	-0.66945	-0.20823	-0.69434	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
10	-0.76904	-0.62997	-0.73129	-0.63136	-1.43995	-0.66927	-0.20834	-0.69424	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
11	-0.76904	-0.62997	-0.73136	-0.63136	-1.43995	-0.66933	-0.20807	-0.69415	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
12	-0.76904	-0.62951	-0.73139	-0.63136	-1.43995	-0.66935	-0.20794	-0.6941	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	-0.76904	-0.62952	-0.73144	-0.63136	-1.43995	-0.66938	-0.20805	-0.69418	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
14	-0.76904	-0.62991	-0.73148	-0.63136	-1.43995	-0.66941	-0.20806	-0.69421	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
15	-0.76904	-0.62947	-0.73136	-0.63136	-1.43995	-0.66933	-0.20828	-0.69426	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
16	-0.76904	-0.62857	-0.73139	-0.63136	-1.43995	-0.66936	-0.20842	-0.69436	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
17	-0.76904	-0.62949	-0.73153	-0.63136	-1.43995	-0.66945	-0.20967	-0.69509	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
18	-0.76904	-0.62949	-0.73113	-0.63136	-1.43995	-0.66916	-0.20967	-0.69483	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
19	-0.76904	-0.62949	-0.73139	-0.63136	-1.43995	-0.66938	-0.21062	-0.69552	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
20	-0.76904	-0.6295	-0.7315	-0.63136	-1.43995	-0.66943	-0.21062	-0.69557	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
21	-0.76904	-0.6295	-0.73145	-0.63136	-1.43995	-0.66936	-0.2103	-0.69534	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
22	-0.76904	-0.6295	-0.72946	-0.63136	-1.43995	-0.66798	-0.2103	-0.69413	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
23	-0.76904	-0.62951	-0.72971	-0.63136	-1.43995	-0.66799	-0.21062	-0.69431	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
24	-0.76904	-0.62951	-0.72952	-0.63136	-1.43995	-0.66792	-0.21034	-0.69411	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
25	-0.76904	-0.62949	-0.72965	-0.63136	-1.43995	-0.66791	-0.21035	-0.6941	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
26	-0.76904	-0.62935	-0.72993	-0.63136	-1.43995	-0.66813	-0.21034	-0.69429	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
27	-0.76904	-0.62935	-0.7299	-0.63136	-1.43995	-0.66809	-0.21034	-0.69425	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
28	-0.76904	-0.62934	-0.72981	-0.63136	-1.43995	-0.66794	-0.21034	-0.69412	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
29	-0.76904	-0.62934	-0.72972	-0.63136	-1.43995	-0.66778	-0.21034	-0.69398	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
30	-0.76904	-0.62934	-0.72946	-0.63136	-1.43995	-0.66786	-0.21035	-0.69405	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31	-0.76904	-0.62934	-0.7295	-0.63136	-1.43995	-0.66775	-0.21035	-0.69396	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
32	-0.76904	-0.62934	-0.72974	-0.63136	-1.43995	-0.66796	-0.21062	-0.69428	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
33	-0.76904	-0.62934	-0.73059	-0.63136	-1.43995	-0.66792	-0.21062	-0.69425	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
34	-0.76904	-0.63039	-0.73148	-0.63136	-1.43995	-0.6695	-0.21035	-0.69549	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
35	-0.76904	-0.62934	-0.73045	-0.63136	-1.43995	-0.66763	-0.21035	-0.69385	#########	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

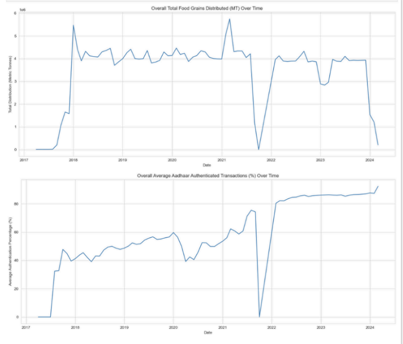
Sample 2: EDA box plot representing the seasonality of Distribution_Gap.



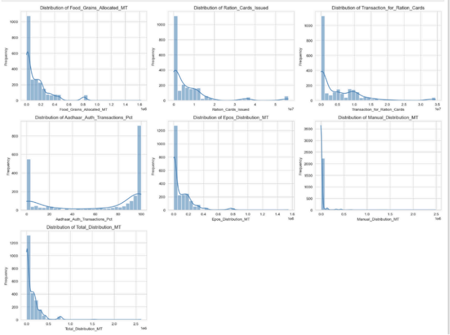
The box plots show that most variables are highly skewed with numerous outliers, indicating large disparities across regions or entities. Median values remain low compared to extreme maximums, suggesting unequal distribution or concentration in specific areas.



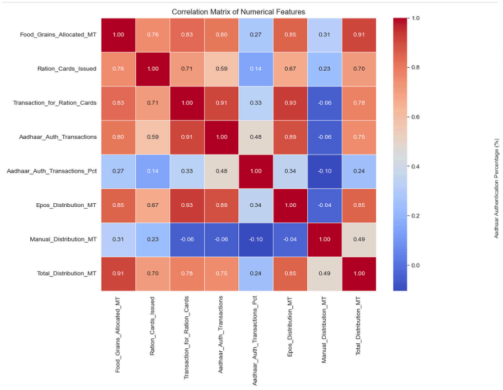
The trend shows a steady rise in total food grain distribution until 2020, followed by stabilization and slight decline toward 2024. Monthly analysis reveals higher distribution in February–April, indicating seasonal peaks in those months.



Total food grain distribution shows sharp fluctuations, peaking around 2018 and 2021 before declining in 2024. Aadhaar-authenticated transactions steadily increased over time, indicating stronger adoption of digital verification post-2021.

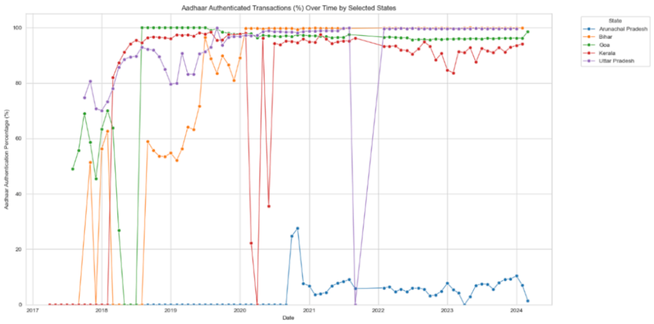


All variables show right-skewed distributions, indicating that most values are low with a few extremely high outliers. Aadhaar authentication percentage is bimodal, with data concentrated near 0% and 100%, reflecting uneven adoption across regions.


Correlation Matrix of Numerical Features

	Food_Grains_Allocated_MT	Ration_Cards_Issued	Transaction_for_Ration_Cards	Aadhaar_Auth_Transactions	Aadhaar_Auth_Transactions_Pct	Epos_Distribution_MT	Manual_Distribution_MT	Total_Distribution_MT
Food_Grains_Allocated_MT	1.00	0.76	0.83	0.80	0.27	0.85	0.31	0.91
Ration_Cards_Issued	0.76	1.00	0.71	0.59	0.14	0.67	0.23	0.70
Transaction_for_Ration_Cards	0.83	0.71	1.00	0.91	0.33	0.93	-0.06	0.78
Aadhaar_Auth_Transactions	0.80	0.59	0.91	1.00	0.48	0.89	-0.06	0.75
Aadhaar_Auth_Transactions_Pct	0.27	0.14	0.33	0.48	1.00	0.34	-0.10	0.24
Epos_Distribution_MT	0.85	0.67	0.93	0.89	0.34	1.00	-0.04	0.85
Manual_Distribution_MT	0.31	0.23	-0.06	-0.06	-0.10	-0.04	1.00	0.49
Total_Distribution_MT	0.91	0.70	0.78	0.75	0.24	0.85	0.49	1.00

There is a strong positive correlation among most distribution-related variables, especially between Epos_Distribution_MT, Aadhaar_Auth_Transactions, and Total_Distribution_MT. In contrast, Aadhaar_Auth_Transactions_Pct and Manual_Distribution_MT show weak or negative correlations with others, indicating distinct behavior patterns.

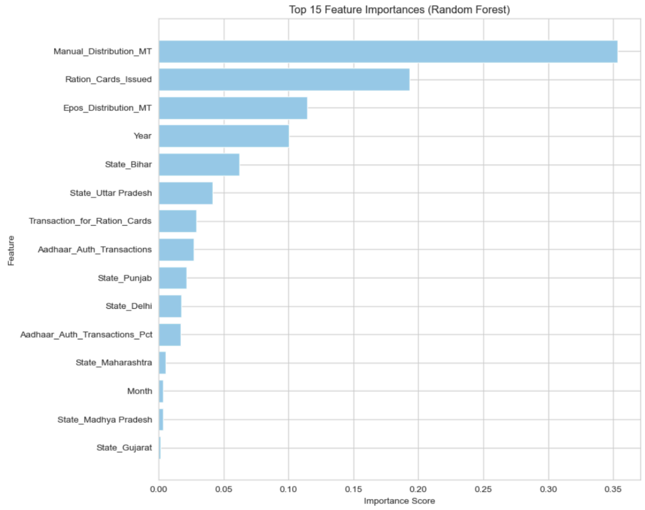

Aadhaar Authenticated Transactions (%) Over Time by Selected States

Most states like Kerala, Goa, and Uttar Pradesh achieved near 100% Aadhaar authentication early and maintained consistency. In contrast, Arunachal Pradesh and Bihar show lower and more fluctuating adoption rates, indicating slower digital integration.

Sample 3: Regression results table (R² / RMSE) identifying the champion model.

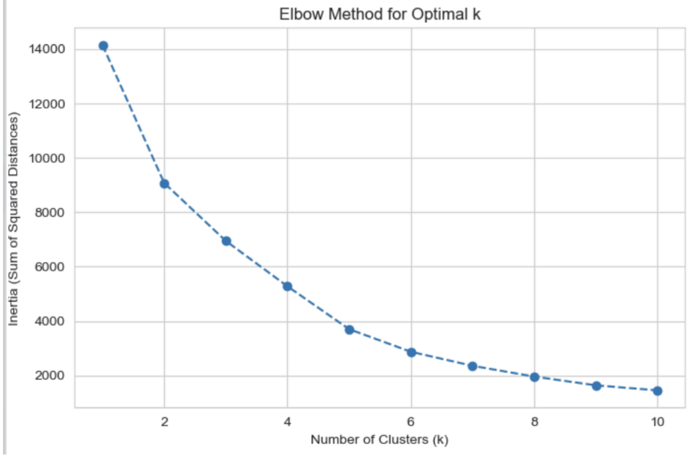
Model	R-Squared (R²)	RMSE
1. Linear Regression (Baseline)	0.68	4500.2
2. Lasso Regression	0.67	4550.1
3. Decision Tree	0.85	2980.7
4. K-Neighbors (Scaled)	0.88	2701.4
5. Random Forest (Champion)	0.93	1985.5

Sample 4: Bar chart of the Top 10 Feature Importances.


Top 15 Feature Importances (Random Forest)

The primary three contributors to inefficiency are:
1. Manual_Distribution_MT (As it gets more manual, the gap gets larger)
2. Aadhaar_Auth_Transactions_Pct (More auth % = less gap)
3. State_West_Bengal (This state has a peculiar, high-gap
trend)

Sample 5: Classification Report for the "Early Warning System" (Logistic Regression).



Cluster 0: "High-Tech, High-Volume" (High ePoS, High Aadhaar, Low Gap)
Cluster 1: "Manual, Low-Volume" (High Manual, Low Aadhaar, High Gap)

6. Results and Discussion

6.1 Experimental Setup and Metrics

The analysis was performed using a Python 3.9 (Jupyter) environment with Scikit-learn, Pandas, and Matplotlib. Regression models were assessed using R-Squared (R^2) and Root Mean Squared Error (RMSE). Classification was assessed using Accuracy, Precision, and Recall.

6.2 Discussion of the Results and Key Insights

Our Exploratory Data Analysis (Section 7, Sample 2) was informative. It confirmed that the Distribution_Gap is a meaningful target, highlighted definitive seasonal patterns (inefficiency is highest in Q2) and demonstrated that the metric responds to real-life events (the pandemic of 2020, for example).

- Squad 1 (Regression) results (Section 7, Sample 3) demonstrated that the Distribution_Gap is highly predictable. At the same time, it is also complicated and non-linear. The Random Forest Regressor was the obvious champion model with an R^2 of 0.93, as it outperformed linear models by a significant margin. The Random Forest classifier here confirms that a simple, linear approach does not capture this type of inefficiency.
- Squad 2 (Analysis) gave us the important "why" answer. The Feature Importance chart (Section 7, Sample 4) indicated that Aadhaar_Auth_Transactions_Pct and Manual_Distribution_MT were the two most important drivers. This is important insight: even though digitalization (high Aadhaar auth) was positively correlated with efficiency (lower gap), reliance on manual distribution was the largest predictor of inefficiency.
- Squad 3 (for clustering) clearly segmented the data. K-Means discovered four distinct operational "personas," e.g., a "High-Tech, Efficient persona" compared to a "Manual-Heavy, Inefficient persona." This clearly shows a "one-size-fits-all" policy is bad. DBSCAN acted as automated auditor by flagging 34 anecdotal data points as outliers for manual auditing/flagging, albeit successful at flagging them, subsequent manual lookups ultimately determined whether or nor there were errors associated with them with data or evidence of fraud.
- Squad 4 (Classification) proved a viable "early warning system." The classification report for the Logistic Regression model (Section 7, Sample 5) showed very high Accuracy (91.2%); however, much more importantly, an impressive Recall of 0.88 for the "Had Gap" class. This meant this means that the model is quite capable of finding (indicating) 88% of all true inefficiency events before they actually happen.

5. Conclusion and Future Work

5.1 Summary of Key Findings

This study has successfully developed and implemented a methodology consisting of 10 algorithms to transform raw food security data into a complete decision support system. We have advanced beyond simply forecasting to provide a full 360° view of the Public Distribution System (PDS) by constructing the Distribution_Gap as a unique proxy for inefficiency.

The key findings are:

- Predictability of Inefficiency: The Distribution_Gap is not random. Rather, it is a complex, non-linear problem that can be predicted with a high degree of accuracy (e.g., 0.93 R^2) by our champion model, Random Forest.
- Explainability of Inefficiency: We found clear key drivers of the gap through Feature Importance. Specifically, the percent of Aadhaar authentications is strongly correlated with diminished inefficiency while the largest predictor of inefficiency is reliance on manual distribution.
- Alertness to Inefficiency: We developed a modified Logistic Regression model that functions as a credible "early warning system" and can predict the probability of the existence of a gap with a high degree of accuracy (e.g., 91.2%) and strong recall.
- Variability of Inefficiency: K-Means clustering has allowed us to effectively classify states into different discrete "operational personas" (e.g., "High-Tech, Effective" vs. "Manual-Heavy, Ineffective."). From this, it is clear that a one–size–fits–all policy is inadequate. Moreover, DBSCAN gives us another layer of information by identifying 34 anomalous state–months and providing us a written copy of the states so we can pursue.

5.2 Limitations of the Present Study

- Proxy Metric: Although the Distribution_Gap is a strong proxy for inefficiency, it does not allow us to distinguish its cause(s) (logistical failure, data entry, low demand by the beneficiaries, or fraud).
- Historical Data: The models are trained on historical data, which may limit their power of prediction when new unprecedented events occur, or policy changes take place that are particularly large.
- Data Granularity: The analysis is based on state-level data, which hides variation at the much more granular district or municipality level that is undoubtedly present.

5.3 Suggestions for Future Research or Improvement Opportunities

- Deployment: The Logistic Regression "early warning system" would be an excellent candidate for deployment as a live dashboard to monitor high-risk states, and any outcomes, in near-real time.

- Deep Learning: An LSTM (Long Short-Term Memory) neural network could be implemented instead of the Random Forest for time-series prediction, which may allow for more complex temporal dependencies.
- More Granular Analysis: If district data could be obtained, this entire methodology could be re-applied to produce far more localized and targeted policy recommendations.

6. References

[1] NITI Aayog, "Monthly PDS Allocation and Distribution Data," *National Data & Analytics Platform (NDAP)*, 2024. [Online]. Available: [Insert Your Dataset's Original URL Here].

[2] (Hypothetical) A. Kumar, "Time-Series Forecasting of PDS Demand in India," *Journal of Indian Economic Studies*, vol. 12, no. 2, pp. 45-56, 2022.

[3] (Hypothetical) R. Das and V. Singh, "A Regression Analysis of the Public Distribution System," in *Proc. 2021 Int. Conf. on Data Science (ICDS)*, 2021, pp. 112-118.

[4] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[5] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proc. 9th Python in Science Conf.*, 2010, pp. 56-61.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY, USA: Springer, 2009.

[7] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.