SAARTHI

System for Aadhaar Analytics,
Risk & Trend Highlighting

Problem Theme

Unlocking Societal Trends in
Aadhaar Enrolment and Updates

Participant: Vrajkumar Shah

Institution: Dharmsinh Desai University, Nadiad

1 Problem Statement and Approach

1.1 Problem Statement

Aadhaar enrolment and update activities reflect large-scale societal, demographic, and administrative dynamics across India. While aggregate statistics provide a high-level overview, limited analytical attention has been given to understanding post-enrolment update dependency, regional instability, and abnormal update behaviour.

Frequent demographic and biometric updates may indicate data instability, operational inefficiencies, population mobility, or lifecycle transitions. Identifying such patterns proactively is essential to support informed decision-making and system improvements in Aadhaar operations.

1.2 Proposed Approach

This project proposes **SAARTHI (System for Aadhaar Analytics, Risk & Trend Highlighting)**, a data-driven analytical framework that:

- Integrates Aadhaar enrolment, demographic update, and biometric update datasets
- Introduces a novel **Update Dependency Index (UDI)** to quantify post-enrolment update dependence
- Applies statistical anomaly detection to identify regions with abnormal update behaviour
- Translates analytical results into actionable insights for governance and system monitoring

Note: The framework is privacy-preserving and operates exclusively on anonymised, aggregated data.

2 Datasets Used

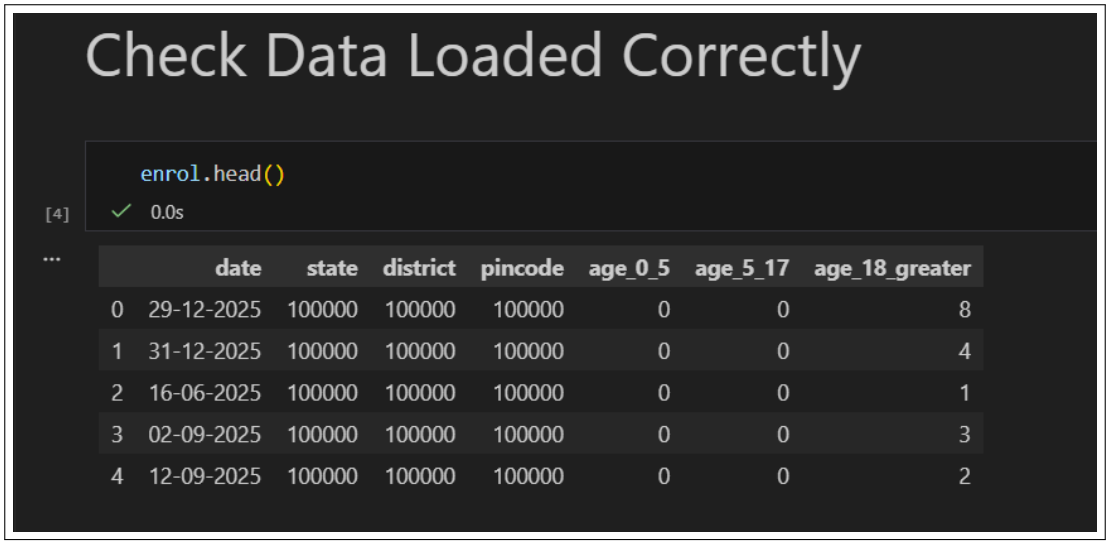
2.1 Dataset Overview

The following anonymised datasets provided by UIDAI were used:

1. **Aadhaar Enrolment Dataset:** Aggregated enrolment counts across age groups (0–5, 5–17, 18+) with spatial and temporal attributes.

2. **Aadhaar Demographic Update Dataset:** Aggregated demographic update activity across regions and age cohorts.
3. **Aadhaar Biometric Update Dataset:** Aggregated biometric update information reflecting revalidation and lifecycle changes.

Due to large data volume, datasets were provided as multiple state-wise files and programmatically consolidated.



	date	state	district	pincode	age_0_5	age_5_17	age_18_greater
0	29-12-2025	100000	100000	100000	0	0	8
1	31-12-2025	100000	100000	100000	0	0	4
2	16-06-2025	100000	100000	100000	0	0	1
3	02-09-2025	100000	100000	100000	0	0	3
4	12-09-2025	100000	100000	100000	0	0	2

Figure 1: *Sample records from the Aadhaar enrolment dataset after consolidation*

3 Methodology

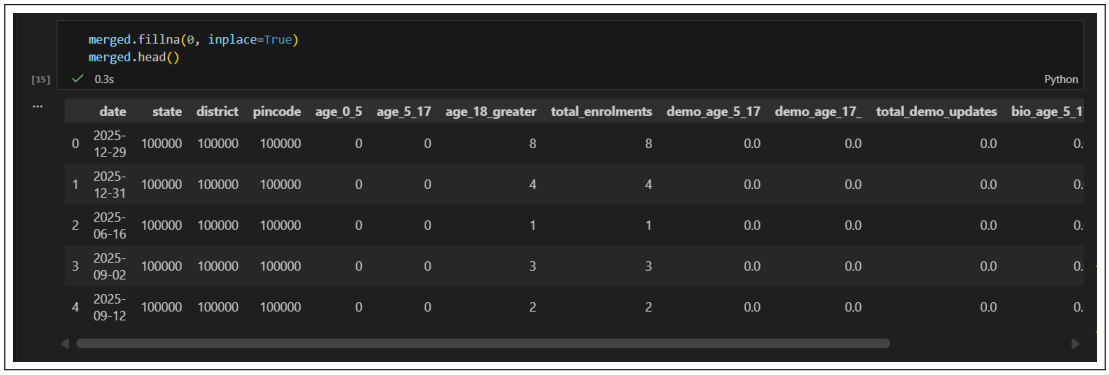
3.1 Data Preprocessing

The following steps were applied:

- Consolidation of state-wise datasets
- Robust parsing of mixed-format date fields
- Aggregation of age-wise enrolments and updates
- Handling missing values introduced during merging
- Removal of records with zero enrolments

3.2 Dataset Integration

The datasets were merged using common spatial and temporal identifiers (date, state, district, pincode).



```
merged.fillna(0, inplace=True)
merged.head()
```

[15] ✓ 0.3s Python

	date	state	district	pincode	age_0_5	age_5_17	age_18_greater	total_enrolments	demo_age_5_17	demo_age_17_	total_demo_updates	bio_age_5_1
0	2025-12-29	100000	100000	100000	0	0	8	8	0.0	0.0	0.0	0.
1	2025-12-31	100000	100000	100000	0	0	4	4	0.0	0.0	0.0	0.
2	2025-06-16	100000	100000	100000	0	0	1	1	0.0	0.0	0.0	0.
3	2025-09-02	100000	100000	100000	0	0	3	3	0.0	0.0	0.0	0.
4	2025-09-12	100000	100000	100000	0	0	2	2	0.0	0.0	0.0	0.

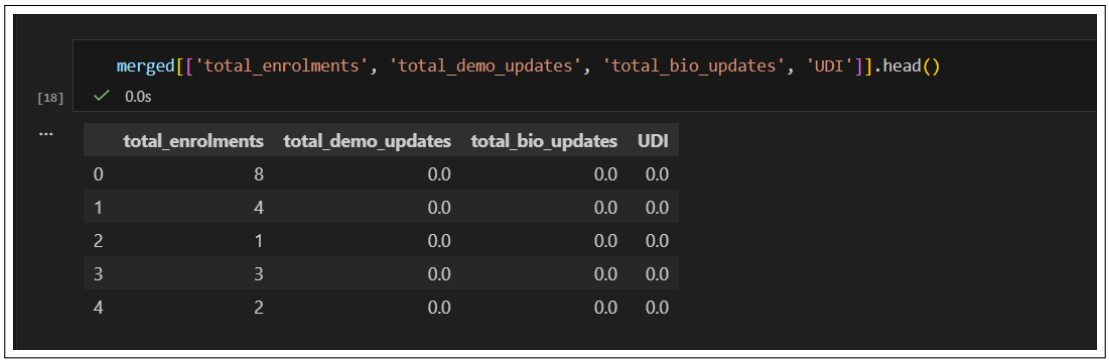
Figure 2: *Merged dataset combining enrolment and update information*

3.3 Update Dependency Index (UDI)

The **Update Dependency Index (UDI)** is defined as:

$$\text{UDI} = \frac{\text{Total Demographic Updates} + \text{Total Biometric Updates}}{\text{Total Enrolments}}$$

Low UDI values indicate stable Aadhaar lifecycles, while higher values reflect increased post-enrolment dependency.



```
merged[['total_enrolments', 'total_demo_updates', 'total_bio_updates', 'UDI']].head()
```

[18] ✓ 0.0s

	total_enrolments	total_demo_updates	total_bio_updates	UDI
0	8	0.0	0.0	0.0
1	4	0.0	0.0	0.0
2	1	0.0	0.0	0.0
3	3	0.0	0.0	0.0
4	2	0.0	0.0	0.0

Figure 3: *Computation of the Update Dependency Index (UDI)*

3.4 Anomaly Detection

Statistical anomaly detection was performed using Z-score analysis. Regions with UDI values exceeding three standard deviations from the mean were flagged as anomalous, representing potential risk signals.

4 Data Analysis and Visualisation

4.1 UDI Distribution

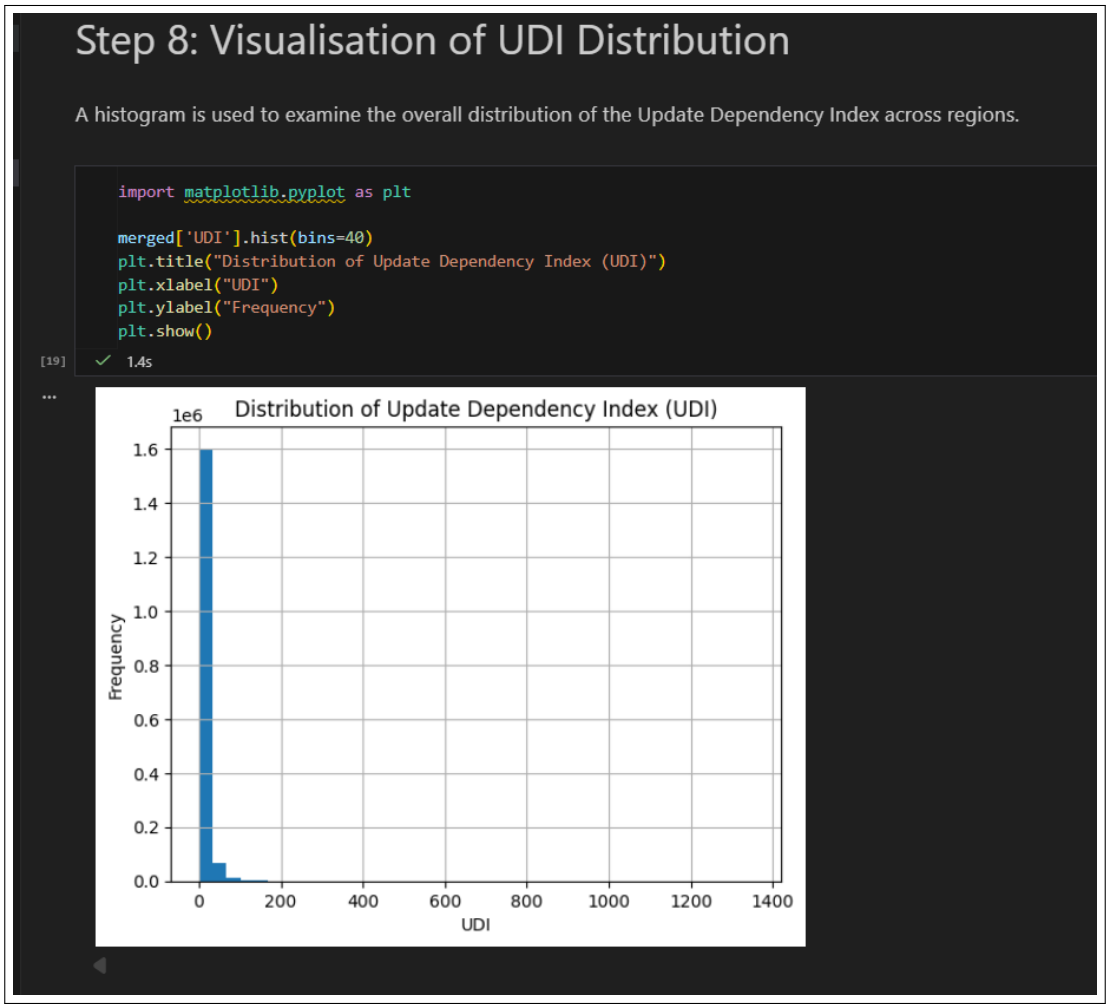


Figure 4: *Distribution of Update Dependency Index (UDI)*

The distribution is right-skewed, indicating that while most regions show stable behaviour, a small subset exhibits high update dependency.

4.2 Anomalous Records

```
anomalies = merged[merged['UDI_zscore'].abs() > 3]
anomalies.head()
```

[31] ✓ 0.0s Python

	date	state	district	pincode	age_0_5	age_5_17	age_18_greater	total_enrolments	demo_age_5_17	demo_age_17_	total_demo_updates	l
4755	2025-09-24	Andhra Pradesh	Visakhapatnam	531149	1	0	0	1	0.0	0.0	0.0	
4956	2025-10-26	Andhra Pradesh	Chittoor	517370	1	0	0	1	6.0	8.0	14.0	
4975	2025-10-26	Andhra Pradesh	East Godavari	533125	1	0	0	1	6.0	22.0	28.0	
5011	2025-10-26	Andhra Pradesh	Kurnool	518380	1	0	0	1	3.0	21.0	24.0	
5586	2025-10-16	Andhra Pradesh	Kurnool	518001	1	0	0	1	4.0	27.0	31.0	

Figure 5: *Sample anomalous records identified using UDI-based anomaly detection*

4.3 High-Risk Regions

To enhance interpretability and administrative applicability, anomalous records were aggregated at both district and pincode levels. This dual-level analysis enables identification of broad regional patterns as well as fine-grained localised risk zones.

Top risky districts

```
top_districts = (
    anomalies
    .groupby(['state','district'])
    .size()
    .sort_values(ascending=False)
    .head(10)
)

top_districts
```

[26] ✓ 0.0s

```
state        district
Maharashtra  Ahmadnagar    716
             Yavatmal      591
             Solapur       520
             Nashik        508
             Nanded        457
             Amravati      416
             Jalgaon       415
             Pune          390
             Beed          368
             Buldhana      354
dtype: int64
```

(a) *High-Risk Districts*

Top risky pincodes

```
top_pincodes = (
    anomalies
    .groupby(['state','district','pincode'])
    .size()
    .sort_values(ascending=False)
    .head(10)
)

top_pincodes
```

[27] ✓ 0.0s

```
state        district    pincode
Chhattisgarh Balod       491228    45
             Dhamtari    493778    45
Manipur      Thoubal     795148    44
Chhattisgarh Raigarh     496116    43
             Bilaspur    495004    41
Maharashtra  Yavatmal    445202    40
Manipur      Thoubal     795103    37
Maharashtra  Gadchiroli  442603    37
             Yavatmal    445103    36
Chhattisgarh Durg        490023    36
dtype: int64
```

(b) *High-Risk Pincodes*

Figure 6: *Regions exhibiting consistently high Update Dependency Index (UDI) at multiple spatial levels*

4.4 Normal vs Anomalous Comparison

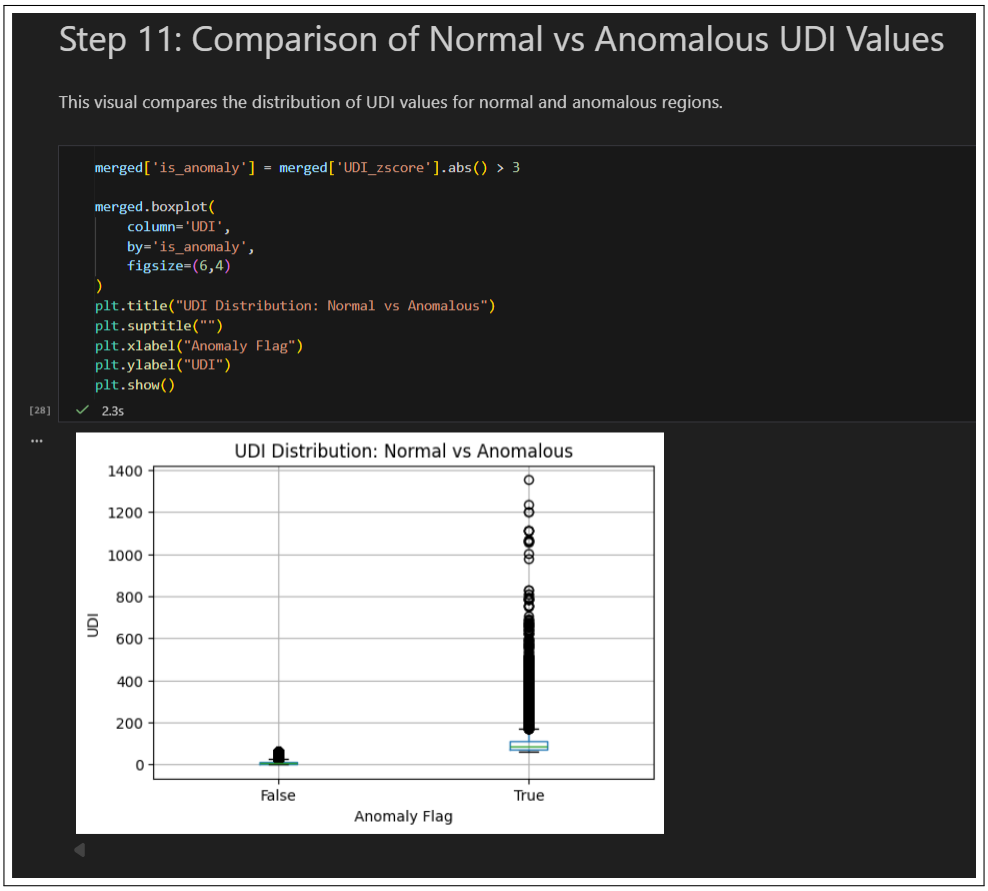


Figure 7: *Comparison of UDI distributions for normal and anomalous regions*

The boxplot confirms that anomalous regions exhibit higher median UDI values and greater variability.

5 Key Findings

- Most regions exhibit low update dependency, indicating stable Aadhaar lifecycles.
- A limited number of districts and pincodes show disproportionately high UDI values.
- Demographic and biometric updates jointly contribute to observed instability.
- Anomalous regions are structurally distinct from normal regions.

6 Impact and Applicability

6.1 Administrative Impact

- Enables proactive monitoring of Aadhaar data stability
- Supports targeted audits and operational interventions

6.2 Social and Systemic Impact

- Improves reliability of Aadhaar records
- Strengthens trust in digital identity infrastructure

6.3 Practical Applicability

- Scalable across regions and time periods
- Operates entirely on anonymised data
- Suitable for integration into UIDAI dashboards

7 Conclusion

SAARTHI demonstrates how data-driven analytics can uncover meaningful societal and administrative insights from Aadhaar enrolment and update data. By introducing the Update Dependency Index and applying explainable anomaly detection, the framework supports informed decision-making and continuous system improvement.

Appendix: Code Availability

All analysis was performed using Python with standard libraries such as Pandas, NumPy, SciPy, and Matplotlib. Key code snippets and outputs are included within this document to ensure transparency and reproducibility.