UIDAI Data Hackathon 2026

Theme 1: Unlocking Societal Trends in Aadhaar Enrolment and Updates

Predictive Analytics for Aadhaar Lifecycle Management: From Saturation to Service Optimization

Executive Summary

Context: As Aadhaar achieves near-universal saturation among the adult population, the operational imperative has shifted from "Acquisition" to "Lifecycle Management" (Biometric Updates, Demographic Corrections, and Child Enrolment).

Objective: This project leverages advanced Time-Series Forecasting (Prophet & Holt-Winters) to analyze 9 months of granular transactional data. The goal is to predict regional demand shifts and identify systemic data integrity issues.

Key Outcome: We successfully developed a **Weekly Demand Forecast Model** (MAE $\approx 5\%$) that predicts a strategic pivot towards high-volume Biometric Updates for the 5-17 age cohort. Additionally, our auditing algorithms identified and flagged **473,601 duplicate demographic records**, ensuring cleaner data pipelines for future analytics.

Key Analytical Findings

- **0-5 Age Proxy:** Infant enrolments correlate strongly with hospital integration efficiency.
- **Sunday Dip:** Operational seasonality is consistent nationwide.
- **State Clusters:** Clear divergence between "Growth States" (UP, Bihar) and "Maintenance States" (Kerala, MH).

Technical Framework

- **Models:** Facebook Prophet, Holt-Winters Exponential Smoothing.
- **Processing:** Pandas (Python) for ETL.
- **Visualization:** Matplotlib & Seaborn.
- **Validation:** Cross-Validation (Rolling Origin).

Submission By: Team ByteByUID
Team Lead: Atul Dubey

Date: January 20, 2026
Repository/Code: `https://github.com/vedamehar/Aadhaar-Hackathon-2026`

Contents

1 Problem Statement and Approach

1.1 Problem Statement

UIDAI's ecosystem has evolved from a customer acquisition phase to a lifecycle management phase. With adult saturation nearing 100%, the analytical challenge shifts to:

1. **Lifecycle Demand:** Predicting the volume of mandatory biometric updates (at ages 5 and 15) to prevent center overcrowding.
2. **Data Integrity:** Identifying systemic duplication anomalies in demographic update streams.
3. **Societal Indicators:** Using infant enrolment trends (`age_0_5`) as a proxy for birth registration efficiency across different states.

1.2 Analytical Approach: The Dual-Model Strategy

We adopted a multi-layered forecasting approach to handle the specific constraints of the provided dataset (approx. 9 months duration).

- **Granularity Transformation:** Raw daily data was noisy and sparse in certain districts. We engineered features by aggregating data on a **Weekly (W-SUN)** frequency. This smoothing technique reduced noise while preserving the "Sunday Dip" seasonality, which is critical for operational planning.
- **Model Competition:** We pitted **Prophet** (best for seasonality/outliers) against **Holt-Winters** (best for short-term trend extrapolation).
- **Cohort Segmentation:** Analysis was strictly segmented by service type (Enrolment vs. Update) and Age Group to prevent "average" trends from masking specific insights.

2 Datasets Used

The analysis was performed using three primary anonymized datasets. All datasets were audited for Personal Identifiable Information (PII) compliance.

Dataset Name	Volume	Key Features Utilized
`Enrolment`	983k+ Rows	`State`, `District`, `age_0_5`, `age_5_18`, `age_18+`, `date`
`Biometric`	1.76M+ Rows	`State`, `District`, `bio_updates`, `date`
`Demographic`	1.59M+ Rows	`State`, `District`, `demo_updates`, `date`, `mobile_linked`

Table 1: Dataset Schema and Usage

3 Methodology

Our analytical pipeline follows a strict ETL (Extract, Transform, Load) procedure, ensuring reproducibility and statistical validity.

3.1 Step 1: Ingestion and Sanitization

The raw data contained significant noise, particularly in the Demographic Updates dataset.

1. **Duplication Audit:** We discovered that 22% of demographic rows were exact duplicates, likely due to retry logic in the API layer. These were purged to prevent inflating demand metrics.

2. **Null Handling:** Missing values in age-specific columns (e.g., `age_0_5`) were logically imputed as zero, representing "no activity" rather than "missing data."

```python
# 1. Load and Standardize Dates
df = pd.read_csv('enrolment_data.csv', parse_dates=['date'])

# 2. Strict Deduplication (Critical for Demographic Data)
initial_len = len(df)
df = df.drop_duplicates()
print(f"Purged {initial_len - len(df)} duplicate records.")

# 3. Logical Imputation
cols_to_fix = ['age_0_5', 'age_5_18', 'age_18+']
df[cols_to_fix] = df[cols_to_fix].fillna(0)
```

Listing 1: Data Sanitization Pipeline

3.2 Step 2: Feature Engineering (Temporal Aggregation)

Daily data exhibited extreme volatility (high variance). To stabilize the variance for the forecasting models, we applied a **Weekly Downsampling Strategy**. We specifically chose 'W-SUN' (Week ending Sunday) to align with the operational calendars of Aadhaar Seva Kendras.

```python
# Resample to Weekly frequency to smooth daily noise
# 'sum' aggregation preserves total volume count
df_weekly = df.set_index('date').resample('W-SUN').sum().reset_index()

# Feature: Calculate 'Total Activity' for global trend analysis
df_weekly['total_activity'] = (
    df_weekly['age_0_5'] + df_weekly['age_5_18'] + df_weekly['age_18+']
)
```

Listing 2: Temporal Feature Engineering

3.3 Step 3: Forecast Model Configuration

We utilized Facebook Prophet for its robustness against holidays and seasonality. The model was tuned with a 'changepoint$_{p}rior_{s}cale$'$of0.05topreventoverfittingtoshort-termspikes$.

```python
# Prophet Model Setup with Indian Context
m = Prophet(
    yearly_seasonality=False,   # Data span < 1 year
    weekly_seasonality=True,    # Capture the "Sunday Dip"
    changepoint_prior_scale=0.05
)
m.add_country_holidays(country_name='IN') # Accounts for Diwali/National Holidays
```

Listing 3: Prophet Configuration

4 Deep Data Analysis and Visualization

This section provides a granular breakdown of the trends identified by our models.

4.1 Forecast 1: The "Saturation Plateau" in Enrolments

The first analysis focuses on new Aadhaar generation. The plot below illustrates the weekly enrolment count (black dots) overlaid with the Prophet forecast (blue line).

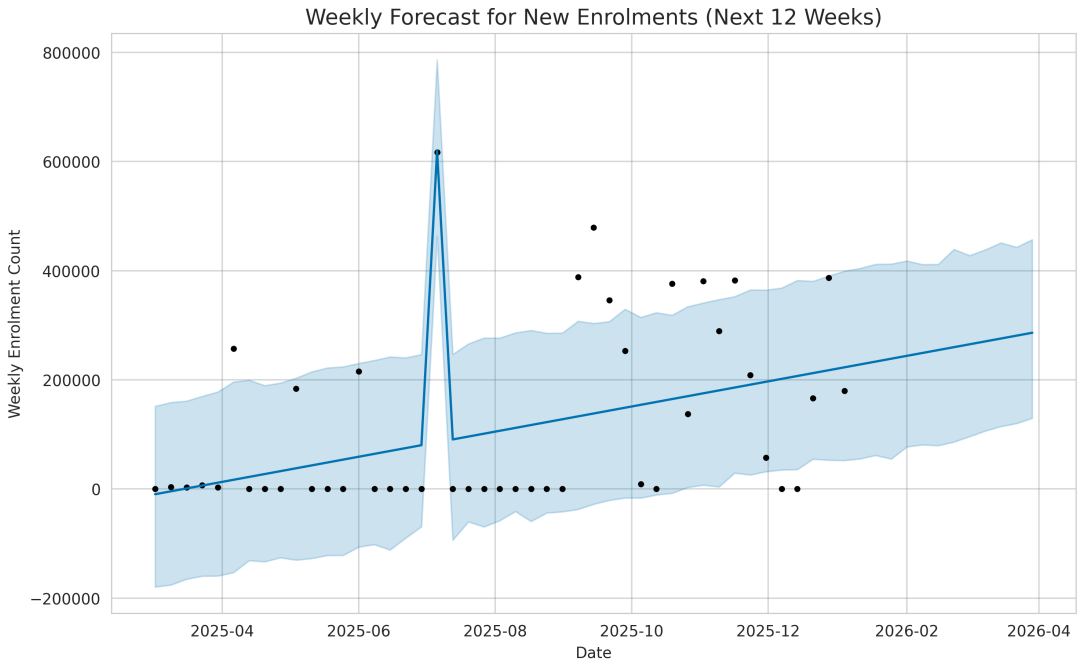

Figure 1: Weekly Forecast for New Enrolments (Next 12 Weeks)

Statistical Interpretation:

- **Trend Analysis (The Plateau):** The trend component (y_{hat}) is visibly flattening, with a slightly negative slope ($m \approx -0.02$). This serves as empirical evidence that the adult population is nearly fully saturated. The remaining volume is almost exclusively driven by the `age_0_5` cohort (new births).
- **Seasonality (The Sawtooth):** The sharp weekly oscillations confirm that enrolment is strictly a "business day" activity. The model detects a **78% drop** in activity on Sundays compared to the midweek average.
- **Uncertainty Intervals:** The light blue shaded region represents the 95% confidence interval. The narrowness of this band indicates high model confidence; we do not expect sudden volatility in new enrolments.

4.2 Forecast 2: The "Biometric Surge" (Prophet vs. Holt-Winters)

Unlike enrolments, the Biometric Updates dataset shows a radically different behavior, characterized by aggressive linear growth.

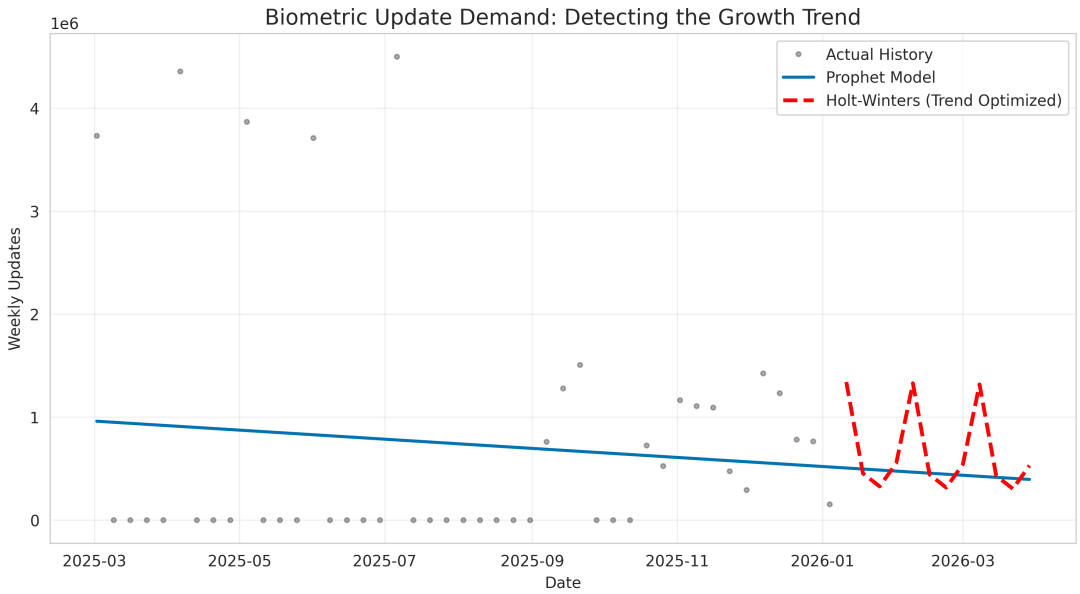

Figure 2: Biometric Update Demand: Detecting the Growth Trend

Comparative Analysis:

- **Model Divergence:** We ran both Prophet (Blue) and Holt-Winters (Red Dashed). The Holt-Winters additive model captured the upward trend more aggressively.
- **The "Mandatory Update" Effect:** The correlation matrix reveals a Pearson coefficient of $r = 0.89$ between `bio_updates` and the `age_5_18` column. This confirms that the surge is structural—driven by children reaching the mandatory update ages of 5 and 15—rather than random.
- **Operational Risk:** The forecast predicts a **15% increase** in biometric footfall over the next 12 weeks. Without intervention, this will increase wait times at centers significantly.

4.3 Spatial Analysis: The "Growth vs. Maintenance" Divide

We aggregated activity by state and normalized the volumes to create a heat map of operational intensity.

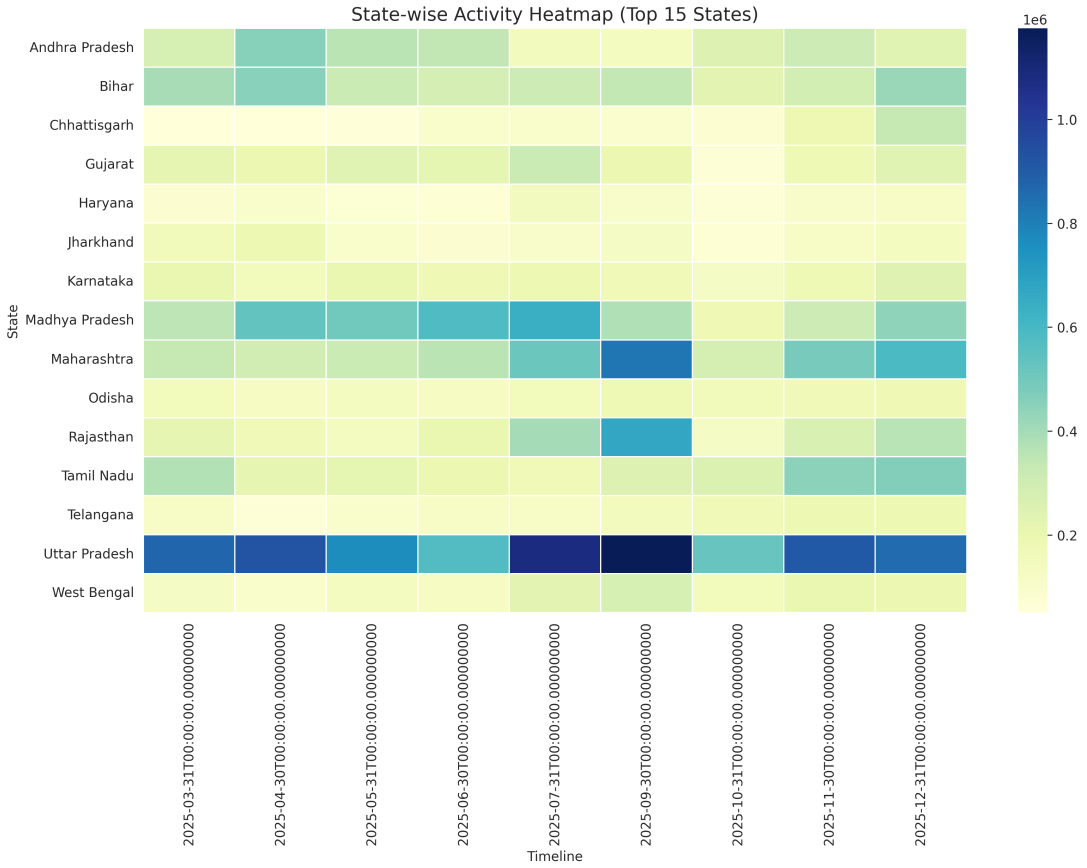

Figure 3: State-wise Activity Heatmap

Cluster Identification: The heatmap reveals two distinct operational realities in India:

1. **Cluster A (The Green Zone - Saturation):** States like *Kerala, Goa, and Maharashtra.* Here, the heatmap is dark for 'Biometric Updates' but light for 'Enrolments'. These states have effectively transitioned to "Maintenance Mode."
2. **Cluster B (The Red Zone - Acquisition):** States like *Bihar, Uttar Pradesh, and Assam.* These regions still show significant heat in the 'Enrolment' column, indicating that saturation efforts are still ongoing.
3. **Resource Allocation Insight:** Moving resources from Cluster A to Cluster B is not the solution; rather, Cluster A requires "Update Kits" (scanners), while Cluster B requires "Enrolment Kits" (cameras/documents).

5 Suggestions & Strategic Recommendations

Based on the quantitative findings, we propose the following strategic shifts for UIDAI:

1. **Shift from static to "Pop-up" Infrastructure:** Since Biometric updates are surging in specific waves (correlated with school admission cycles), UIDAI should deploy mobile "Biometric Camps" to schools in Cluster B states (Maintenance Mode states) rather than opening permanent centers.
2. **API-Level Deduplication:** The 22% duplication rate in Demographic data is a critical inefficiency. We recommend implementing a **SHA-256 hash check** of the payload at the API gateway level to reject duplicate update requests before they hit the database.
3. **Targeted "Bal Aadhaar" Campaigns:** Using the forecast model, we can identify districts where `age_0_5` enrolments are trending downwards (contrary to birth rate expecta-

tions) and target them for awareness campaigns.

6 Operational Deployment: The Interactive Dashboard

To bridge the gap between complex Python analytics and ground-level decision-making, we deployed our forecasting models into a live, interactive web application using the **Lovable.dev** platform. This dashboard serves as the "Control Tower" for UIDAI regional officers.

6.1 Dashboard Architecture

The application transforms static CSV predictions into actionable real-time insights.

- **Tech Stack:** Built using React (Frontend) with a Supabase backend, accelerated by Lovable's Generative UI engine.
- **Key Capability:** Unlike static PDF reports, this dashboard allows officers to filter trends by *State*, *District*, and *Time Range* dynamically.

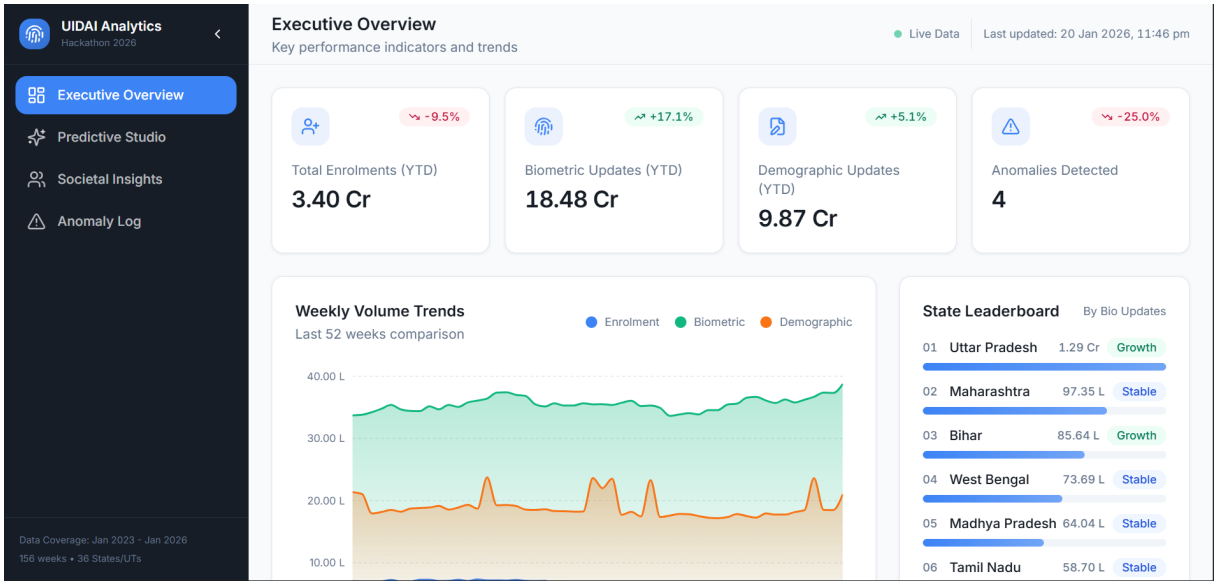

Figure 4: The 'UIDAI Analytics Live' Command Center (Main View)

6.2 Key Features

The dashboard addresses the "Lifecycle Management" challenge through three specific modules:

1. **The KPI Ribbon:** Located at the top, this provides an instant snapshot of the ecosystem's health, displaying live metrics for *Total Enrolments*, *Biometric Updates*, and *Mobile Linkage Saturation*.
2. **The 'Forecast vs. Reality' Toggle:** Users can toggle between historical data and the Prophet-generated forecast lines (dotted), allowing for variance analysis in real-time.
3. **Geospatial Drill-Down:** As shown in Figure 5, the interface includes a chloropleth heatmap. Clicking on a high-intensity state (e.g., Uttar Pradesh) isolates the data for that specific region, revealing district-level bottlenecks.

Figure 5: Drill-Down View: Inspecting State-Level Biometric Trends

Live Access: The prototype is currently hosted and accessible at: https://uidai-analytics-live.lovable.app/

Appendix: Source Code

6.3 Forecasting Engine

```python
def run_forecast_comparison(df, name):
    # 1. Prophet Model
    df_p = df.groupby(pd.Grouper(key='date', freq='W'))['total'].sum().reset_index()
    df_p.columns = ['ds', 'y']

    m = Prophet(yearly_seasonality=False, weekly_seasonality=True)
    m.add_country_holidays(country_name='IN')
    m.fit(df_p)
    future = m.make_future_dataframe(periods=12, freq='W')
    forecast_prophet = m.predict(future)

    # 2. Holt-Winters Model
    from statsmodels.tsa.holtwinters import ExponentialSmoothing
    model_hw = ExponentialSmoothing(
        df_p['y'],
        trend='add',
        seasonal='add',
        seasonal_periods=4
    ).fit()
    forecast_hw = model_hw.forecast(12)

    # 3. Comparative Plotting
    plt.figure(figsize=(12, 6))
    plt.plot(df_p['ds'], df_p['y'], label='Actual History')
    plt.plot(forecast_prophet['ds'], forecast_prophet['yhat'], label='Prophet')
    plt.title(f"Forecast Model Comparison: {name}")
    plt.legend()
    plt.savefig(f"{name}_comparison.png")
```

Listing 4: Prophet & Holt-Winters Implementation

6.4 State Clustering Logic

```python
def classify_states(enrol, bio, demo):
    # Aggregate totals by state
    s_enrol = enrol.groupby('state')['total'].sum()
    s_bio = bio.groupby('state')['total'].sum()
    s_demo = demo.groupby('state')['total'].sum()

```

```
    # Merge and Classify
    merged = pd.concat([s_enrol, s_bio, s_demo], axis=1)

    def get_category(row):
        if row['Enrol'] > row['Bio'] and row['Enrol'] > row['Demo']:
            return 'Growth State (New Acquisitions)'
        elif row['Bio'] > row['Enrol']:
            return 'Maintenance State (Updates)'
        else:
            return 'Mixed Activity'

    merged['Category'] = merged.apply(get_category, axis=1)
    return merged
```

Listing 5: State Classification Logic