UIDAI Report: Optimizing the Aadhaar Life-cycle

Date: 14th January, 2026

By: Nillohit Roy (Team Leader, ID: UIDAI_689)

Problem Statement

Aadhaar saturation in India is near universal for adults (over 99%) and very high overall (around 95-99% across all ages), with enrollment constantly ongoing for newborns. However, the Aadhaar ecosystem faces several challenges that generic reporting fails to capture. Our preliminary analysis identified three critical operational inefficiencies.

Hidden Stagnation: High-level state averages mask specific districts where new enrolment velocity has collapsed to near-zero, despite demographic projections indicating a growing infant population.

The Compliance Gap: A systemic disconnect exists where residents update demographic details due to migration or marriage but fail to perform mandatory bio-metric updates, degrading the bio-metric database's quality over time.

Static Resource Allocation: Operational resources (manpower and servers) are deployed based on static historical averages, leading to severe bottlenecks during temporal demand surges (e.g., specific days of the week or post-harvest seasons).

The Approach

To address these issues, we deployed a **"Describe-Diagnose-Predict"** framework:

Descriptive Analytics: Unified disparate transaction logs from different datasets to create a “Single Source of Truth” dataset for the model to predict and analyze.

Diagnostic Analytics: Utilized statistical feature engineering (Velocity and Ratios) to isolate the root causes of under-performance.

Predictive Intelligence: Deployed unsupervised learning (Isolation Forests) for anomaly detection and supervised learning (XGBoost) to predict future non-compliance risk.

Datasets Used

Aadhaar Enrolment: This dataset is based on districts of each state taken daily. The key features used from this dataset are ‘date’, ‘state’, ‘district’ and ‘age_group’. The main purpose of this dataset is to track the inflow of new identities into the ecosystem

Demographic Updates: This dataset is based on districts of each state taken daily. The key features used from this dataset are ‘date’, ‘state’, ‘district’, ‘pincode’ and ‘age_group’. The main purpose of this dataset is to identify migration patterns and administrative corrections.

Biometric Updates: This dataset is based on districts of each state taken daily. The key features used from this dataset are ‘date’, ‘state’, ‘district’, ‘pincode’ and ‘age_group’. The main purpose of this dataset is to monitor mandatory compliance at ages 5 and 15.

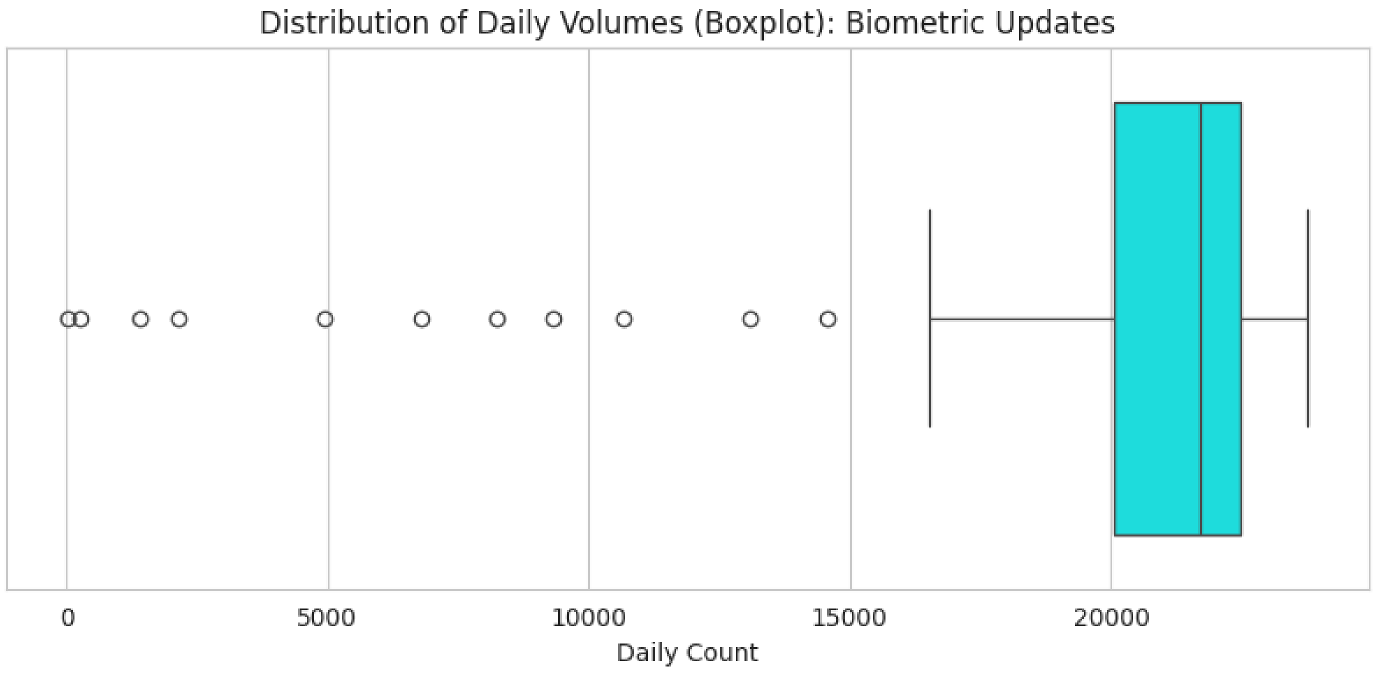
Methodology

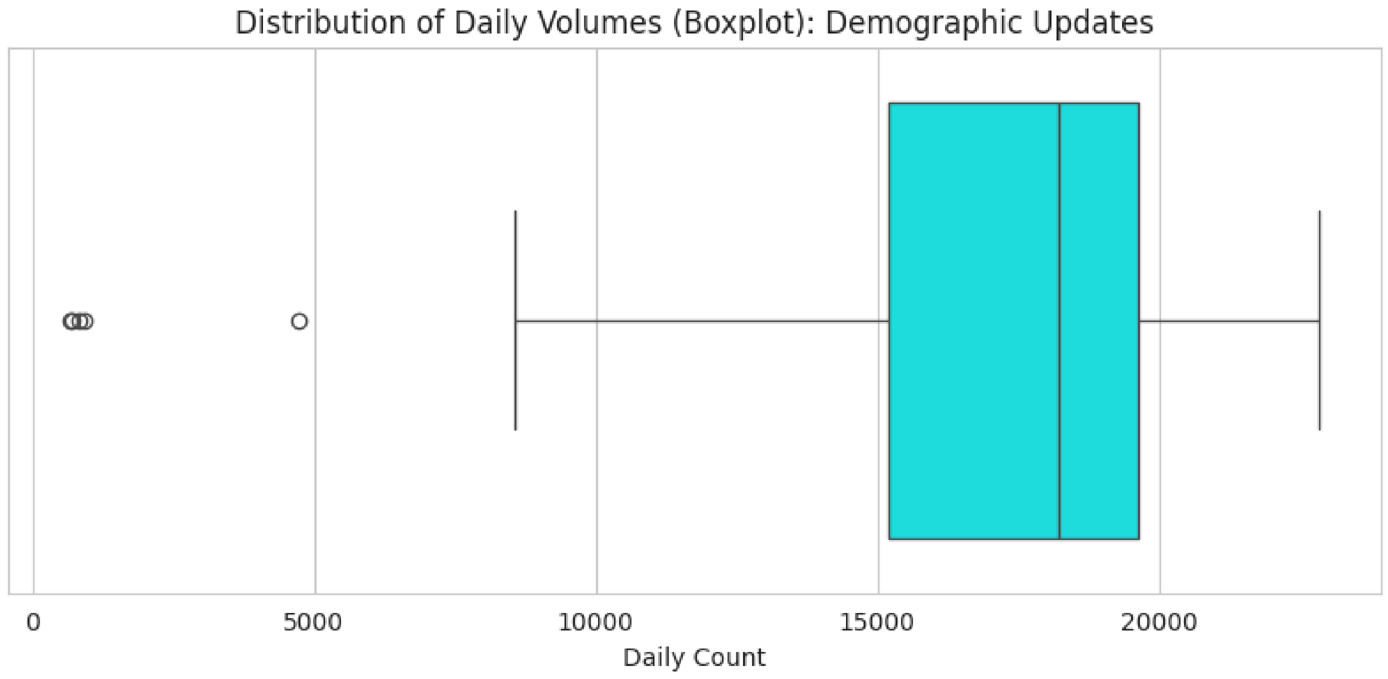
Data Pre-processing Pipeline

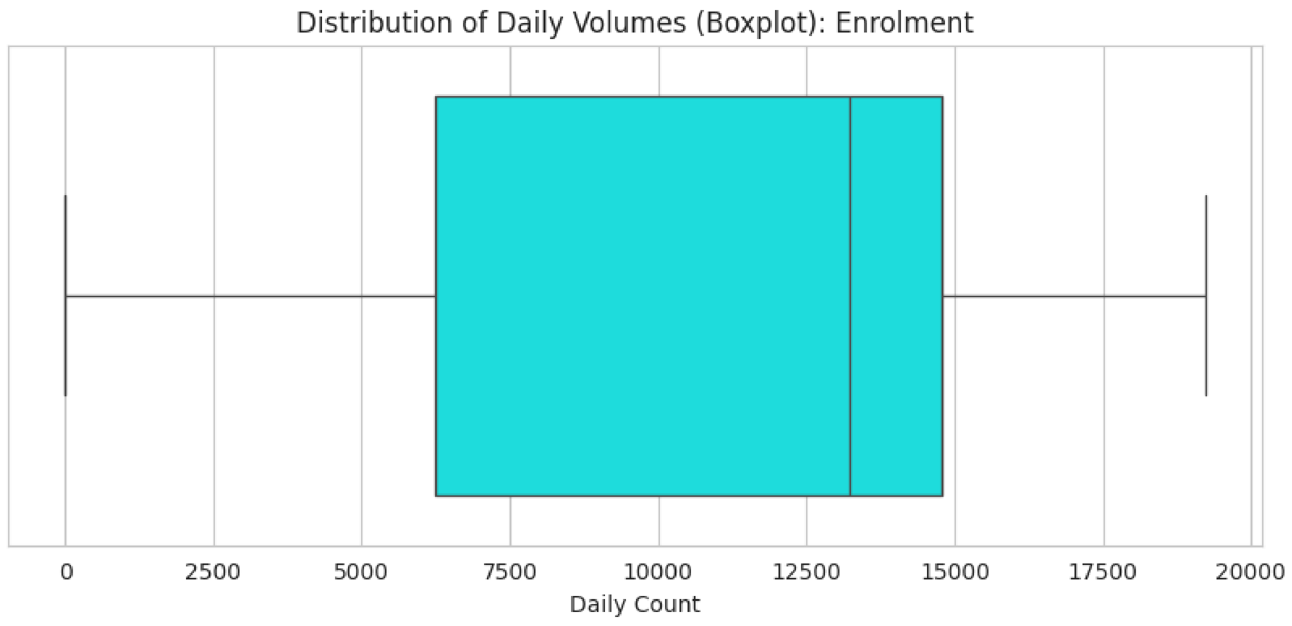
Standardization & Cleaning: We load the dataset from all the three given data sources, standardizing the text fields (normalizing the state and district names against the Local Government Directory), removing data-entry errors, dropping duplicates and finally saving it as a parquet file for faster accessing.

Temporal Aggregation: Raw transaction logs were aggregated to a District-Daily level. Missing dates were back-filled with zero-activity records to ensure continuity for Time Series forecasting.

Outlier Removal: Statistical outliers caused by data entry errors (e.g., >3 Standard Deviations from the rolling mean) were capped to prevent skewing the Machine Learning models.







Advanced Feature Engineering

Firstly, we calculated the enrolment, demographic and bio-metric counts. We then transformed these raw counts into "Smart Features" to feed the Machine Learning Models.

Enrolment Velocity (V_e): A derivative metric measuring the acceleration of enrolments.

$$V_e = \frac{\text{Moving Avg (7 Days)} - \text{Moving Avg Lag (7 Days)}}{\text{Moving Avg (7 Days)}}$$

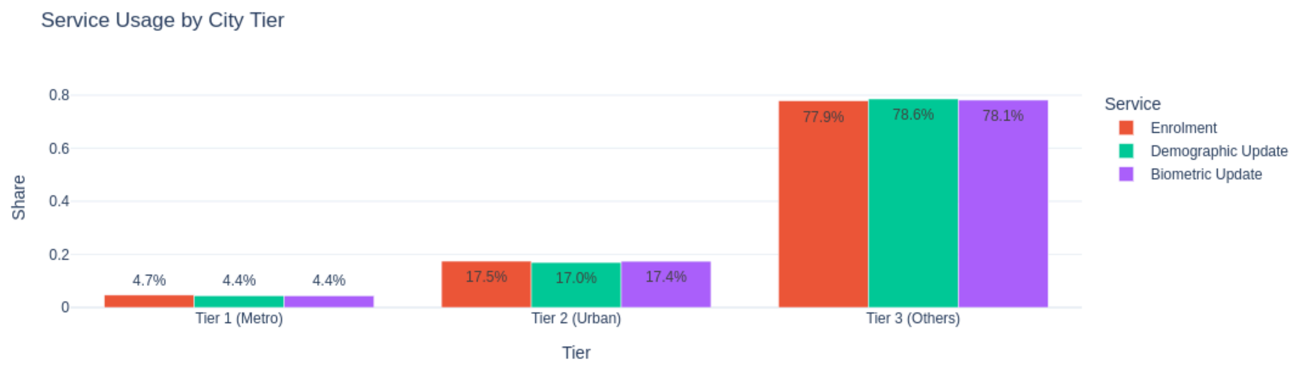
Significance: Identifies districts that are slowing down before they hit zero.

Update to Enrol Ratio: It is defined as the ratio of the total updates to the new enrolments.

$$\frac{\text{New Enrolments}}{\text{Total Updates}}$$

Significance: Distinguishes "Growth states" (Low Ratio) from "Maintenance states" (High Ratio).

Geographic Clusters: By using **K-Means Clustering,** we categorized districts into "Metro Hubs" (Tier 1), "Urban Growth Centers" (Tier 2), and "Rural Hinterlands" (Tier 3) based on transaction volume and type.

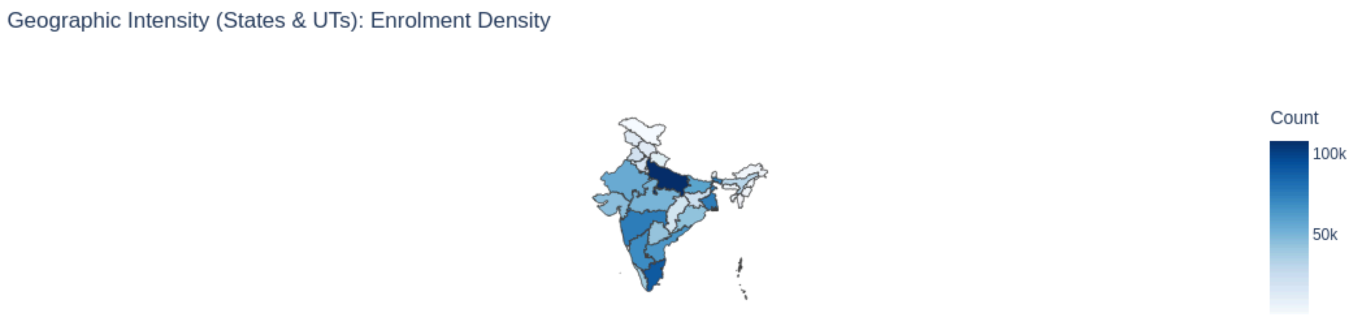


Data Analysis and Visualization Findings

Geographic Disparities: We calculated a Relative Performance Score (RPS) for every district.

$$RPS = \frac{\text{State Average Volume}}{\text{District Volume}}$$

Using this score, we identified the top 10 districts where the RPS was consistently below 0.5 (operating at <50% of the state average). These districts are not just small; they are under-performing relative to their peers.



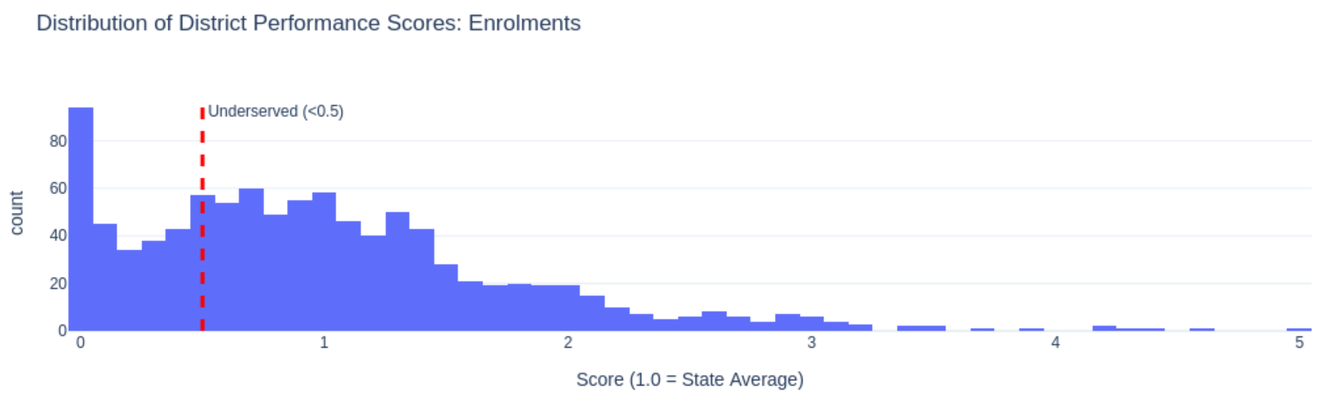
Geographic Intensity (States & UTs): Demographic Density

Count
150k
100k
50k
0

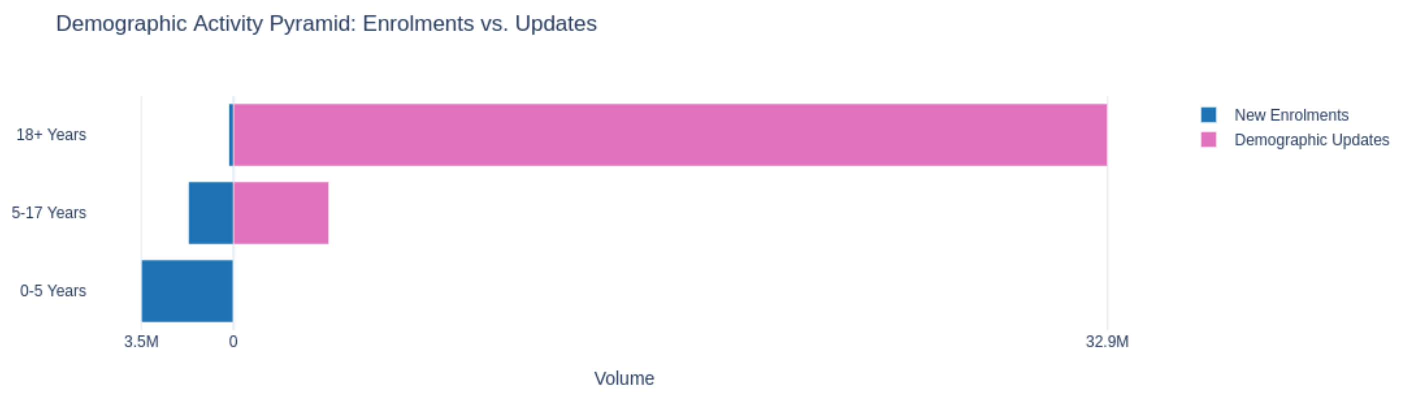
Geographic Intensity (States & UTs): Biometric Density

Count
150k
100k
50k
0

```
Top 10 Most Lagging Districts (Potential Outreach Targets):
              state             district  Volume    State_Avg  \
774      Tamil Nadu            Tiruvarur       1  1964.956522
753      Tamil Nadu         Namakkal   *       1  1964.956522
954     West Bengal              Hooghiy       1  1506.560000
936     West Bengal  24 Paraganas South       1  1506.560000
942     West Bengal              Burdwan       1  1506.560000
951     West Bengal        East Midnapur       1  1506.560000
523     Maharashtra            Hingoli *       1  1425.113208
46   Andhra Pradesh        Visakhapatanam       1  1358.148936
684       Rajasthan              Balotra       1  1306.285714
718       Rajasthan             Salumbar       1  1306.285714

     Performance_Score
774           0.000509
753           0.000509
954           0.000664
936           0.000664
942           0.000664
951           0.000664
523           0.000702
46            0.000736
684           0.000766
718           0.000766
```



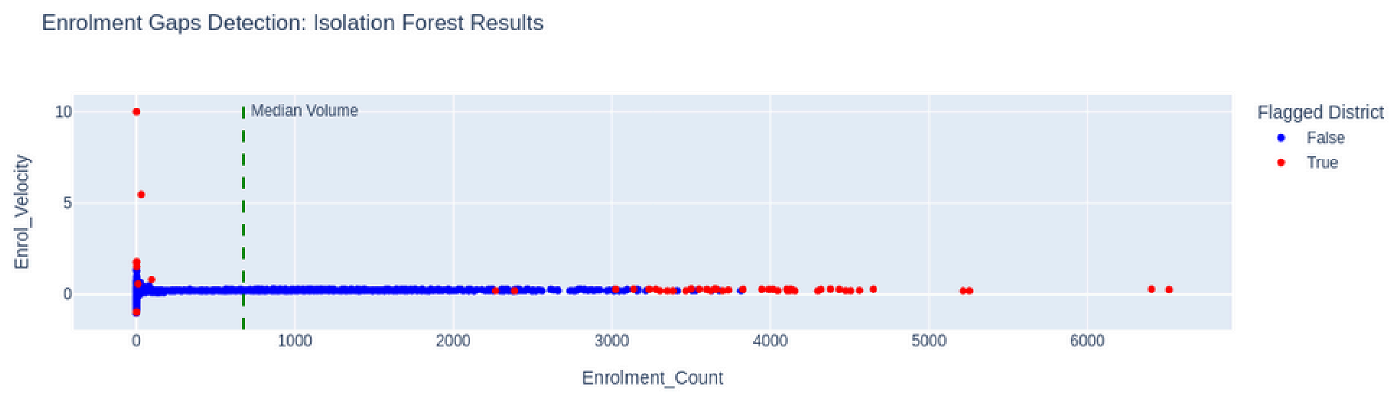
Demographic Shift: The Age Pyramid chart reveals a stark structural contrast between "Growth" (Enrolments) and "Maintenance" (Updates), resembling two opposing geometric shapes. The enrolment bars extend significantly to the left for the **0-5 Age Group**, creating a wide base that sharply tapers off for older age groups. The update bars extend massively to the right for the **18+ Age Group**, with a smaller secondary spike for the **5-17 Age Group**. The ecosystem has transitioned into a "Maintenance Phase." The massive volume of adult updates indicates a mobile workforce constantly correcting demographic details to access financial and government services. The secondary spike in the 5-17 range corresponds to the mandatory bio-metric updates required at ages 5 and 15.



Machine Learning Insights

Anomaly Detection: Operational Irregularities

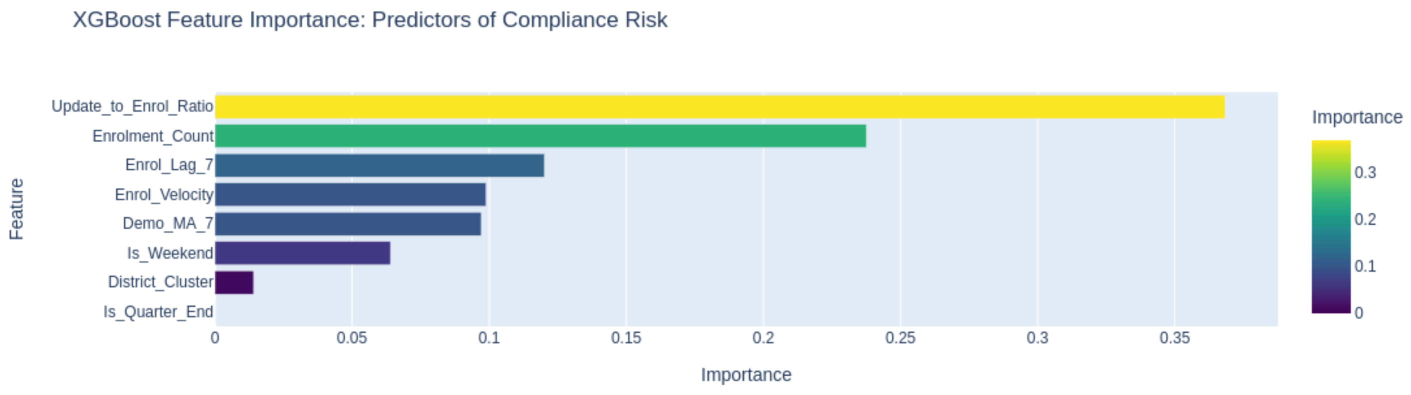
Using Isolation Forest, we scanned the timeline for anomalies. The model detected specific states with a very high enrolment velocity. We flagged the specific districts based on the unusual enrolment velocity as red while the rest blue.



Predicting Non-Compliance

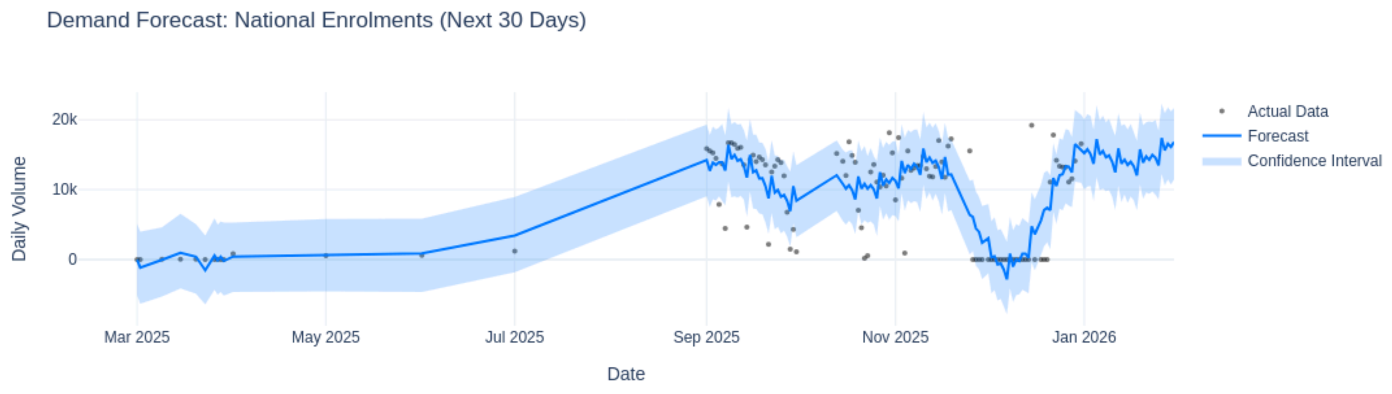
We trained an XGBoost Classifier to predict districts at high risk of falling into the "Low Biometric Compliance" category and achieved a 95.82% accuracy. The model revealed that **“Updates to Enrolment ratio”** is the strongest predictor of compliance risk. This means that citizens update their address when they move but often neglect the bio-metric update. This "Lag" is the critical window for intervention.

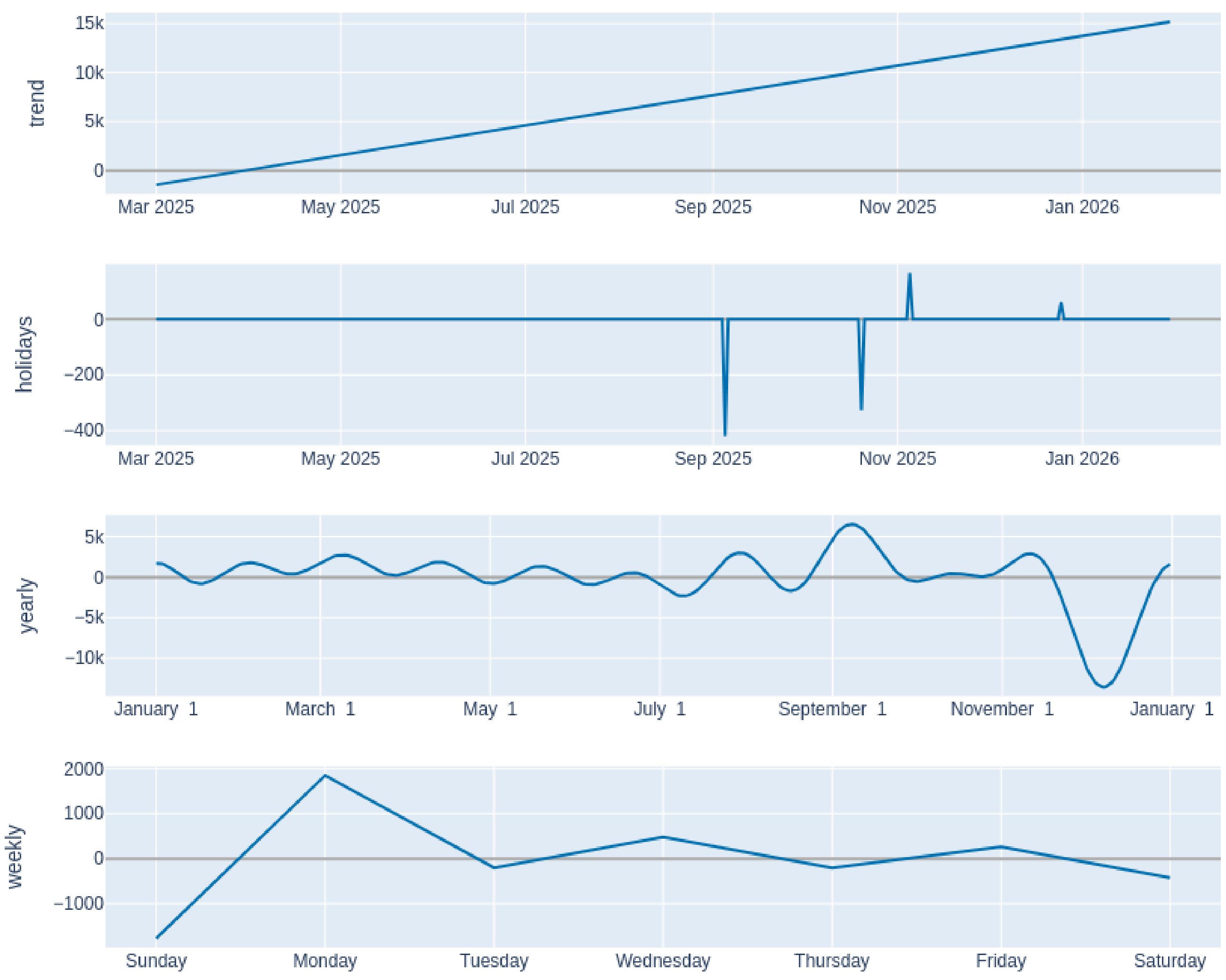
```
--- XGBoost Performance Report ---
Accuracy: 95.82%
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     13791
           1       0.93      0.92      0.92      5179

    accuracy                           0.96     18970
   macro avg       0.95      0.94      0.95     18970
weighted avg       0.96      0.96      0.96     18970
```



Forecasting

We utilized Facebook Prophet to forecast daily demand for the next 30 days and observed two things. There is going to be an increase in enrolments starting from January 2026 and most of the demands will be on Monday. This implies that instead of having equal staffs on all working days, some days needs special attention.





Recommendations

The "Sync-Update" Protocol

To implement an update software that triggers a "Biometric Nudge." This means when a resident a demographic update, the system should check the last bio-metric timestamp. If greater than 5 years, it should mandate or strongly recommend a bio-metric update in the same session.

Targeted Mobile Deployment

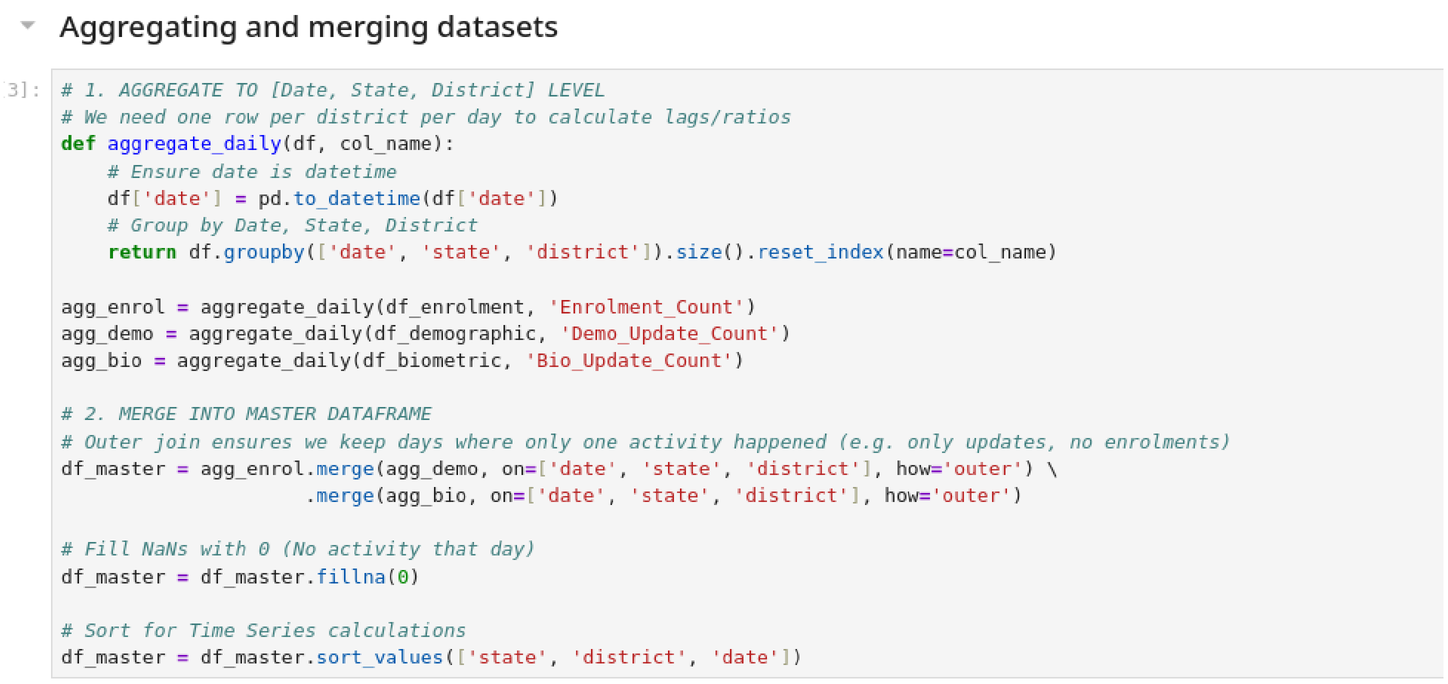
Deploy Mobile Enrolment Vans specifically to the **10 Districts** where the RPS is relatively low. The fixed centers have a higher chance of failing in these areas due to which the RPS is showing a low value. Mobile units may address the accessibility gap.

Dynamic Rostering

Shift 15% of center staff hours as well as servers and resources from mid-week (Wednesday/Thursday) to Mondays. From the forecasting, there is a high probability that there will be a huge demand surge in Mondays as compared to other days of the week. This will reduce wait times without increasing total headcount costs.

Technical Appendix

Data Cleaning and Merging Logic



Aggregating and merging datasets

3]:

```
# 1. AGGREGATE TO [Date, State, District] LEVEL
# We need one row per district per day to calculate lags/ratios
def aggregate_daily(df, col_name):
    # Ensure date is datetime
    df['date'] = pd.to_datetime(df['date'])
    # Group by Date, State, District
    return df.groupby(['date', 'state', 'district']).size().reset_index(name=col_name)

agg_enrol = aggregate_daily(df_enrolment, 'Enrolment_Count')
agg_demo = aggregate_daily(df_demographic, 'Demo_Update_Count')
agg_bio = aggregate_daily(df_biometric, 'Bio_Update_Count')

# 2. MERGE INTO MASTER DATAFRAME
# Outer join ensures we keep days where only one activity happened (e.g. only updates, no enrolments)
df_master = agg_enrol.merge(agg_demo, on=['date', 'state', 'district'], how='outer') \
                     .merge(agg_bio, on=['date', 'state', 'district'], how='outer')

# Fill NaNs with 0 (No activity that day)
df_master = df_master.fillna(0)

# Sort for Time Series calculations
df_master = df_master.sort_values(['state', 'district', 'date'])
```

Feature Engineering

[9]:
```python
def create_lag_features(df):
    df_feat = df.copy()

    # We must group by District to ensure lags don't bleed between districts
    grouper = df_feat.groupby(['state', 'district'])

    # 1. Rolling Averages (Smooth out daily noise)
    # 7-Day Moving Average
    df_feat['Enrol_MA_7'] = grouper['Enrolment_Count'].transform(lambda x: x.rolling(window=7, min_periods=1).mean())
    df_feat['Demo_MA_7'] = grouper['Demo_Update_Count'].transform(lambda x: x.rolling(window=7, min_periods=1).mean())

    # 30-Day Moving Average (Long-term trend)
    df_feat['Enrol_MA_30'] = grouper['Enrolment_Count'].transform(lambda x: x.rolling(window=30, min_periods=1).mean())

    # 2. Lag Features (Value exactly 1 week ago)
    df_feat['Enrol_Lag_7'] = grouper['Enrolment_Count'].shift(7)

    # 3. Velocity (Growth Rate)
    # (Current MA_7 - Previous MA_7) / Previous MA_7

    prev_ma_7 = df_feat['Enrol_MA_7'].shift(1)

    # Define Epsilon to prevent division by zero (0.000001)
    epsilon = 1e-6

    df_feat['Enrol_Velocity'] = (df_feat['Enrol_MA_7'] - prev_ma_7) / (prev_ma_7 + epsilon)

    # Fill NaN values generated by shifting (first 7-30 days will be NaN)
    df_feat = df_feat.fillna(0)

    # Capping extreme velocity values to keep the model stable.
    df_feat['Enrol_Velocity'] = df_feat['Enrol_Velocity'].clip(lower=-10, upper=10)

    return df_feat

df_features = create_lag_features(df_master)
df_features.head(10)
```

Predictive Modeling

Train XGB Model

[8]:
```python
# 1. SPLIT DATA
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# 2. TRAIN XGBOOST
xgb_model = XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=5,
    random_state=42,
    eval_metric='logloss'
)

xgb_model.fit(X_train, y_train)

# 3. EVALUATE
y_pred = xgb_model.predict(X_test)

print("--- XGBoost Performance Report ---")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2%}")
print(classification_report(y_test, y_pred))
```

The code to all the notebooks is available in the following GitHub repository: https://github.com/nillohitroy/UIDAI