# Data Analytics Report

## Comprehensive Analysis of Aadhaar Enrolment

---

### 1. Problem Statement and Approach

#### Problem Statement

The Aadhaar ecosystem plays a critical role in identity inclusion across India. However, large-scale enrolment data often hides structural gaps, regional imbalances, operational inefficiencies, and potential data quality issues.

This project aims to systematically analyse Aadhaar enrolment data to uncover:

- Lifecycle gaps in enrolment across age cohorts
- Temporal trends and seasonal momentum
- Geographic concentration and inequality
- Operational inefficiencies at weekday and district levels
- Anomalous enrolment behaviour requiring administrative audits
- Predictive signals for future enrolment demand

#### Analytical Approach

A multi-layered analytics pipeline was designed, progressing from:

- Univariate analysis (population structure)
- Bivariate analysis (time & geography)
- Trivariate analysis (district-level inequality)
- Advanced analytics (anomaly detection, clustering)
- Predictive modelling (short-term forecasting)

The objective was not only descriptive analysis but actionable insights for policy planning, monitoring, and operational optimisation.

---

## 2. Datasets Used

### Primary Dataset

- UIDAI Aadhaar Enrolment Dataset (provided for Hackathon)
- Period covered: March 2025 – December 2025
- Data split across three CSV files and merged for analysis

### Key Columns Used

| Category | Columns |
|---|---|
| Temporal | date, year, month, month_name, weekday |
| Geography | state, district, pincode |
| Age-wise Enrolment | age_0_5, age_5_17, age_18_greater |
| Derived Metrics | total_enrolments, growth rates, ratios |

### Dataset Scale

- Records: Millions of enrolment entries
- Coverage: All States & UTs
- Granularity: District & Pincode level

---

## 3. Methodology

### 3.1 Data Cleaning & Preprocessing

- Merged multiple CSV files into a unified dataset
- Standardised state names using rule-based mapping
- Removed invalid or malformed date records
- Created derived metrics:
    - total_enrolments
    - Month-over-month growth
    - Age ratios and percentages

### 3.2 Feature Engineering

- Temporal indicators (month index, weekday)
- District-level aggregation
- Coefficient of Variation (CV) for inequality measurement

- Z-score normalisation for anomaly detection

### 3.3 Tools & Techniques

- Python (Pandas, NumPy, SciPy)
- Visualisation: Matplotlib, Seaborn
- Machine Learning:
    - Linear Regression (Forecasting)
    - K-Means Clustering
- Statistical Methods:
    - Coefficient of Variation
    - 3-Sigma anomaly detection

---

## 4. Data Analysis and Visualisation

---

### Data Loading and Preprocessing :

### Code :

```
print("\n[STEP 1/8] DATA LOADING & PREPROCESSING")
print("-" * 80)

try:
    df1 = pd.read_csv('enrolment_part1.csv')
    df2 = pd.read_csv('enrolment_part2.csv')
    df3 = pd.read_csv('enrolment_part3.csv')
    enrolment_df = pd.concat([df1, df2, df3], ignore_index=True)
    print(f"✓ Loaded {enrolment_df.shape[0]:,} records")
except FileNotFoundError:
    print("⚠ ERROR: Upload CSV files to Colab first!")
    raise

# Date parsing
enrolment_df['date'] = pd.to_datetime(enrolment_df['date'], dayfirst=True, errors='coerce')
enrolment_df = enrolment_df.dropna(subset=['date'])

# Temporal features
enrolment_df['year'] = enrolment_df['date'].dt.year
enrolment_df['month'] = enrolment_df['date'].dt.month
enrolment_df['month_name'] = enrolment_df['date'].dt.strftime('%B')
enrolment_df['weekday'] = enrolment_df['date'].dt.day_name()

# Standardize states
enrolment_df['state'] = enrolment_df['state'].astype(str).str.strip().str.title()
enrolment_df['state'] = enrolment_df['state'].str.replace(r'\s+', ' ', regex=True)
enrolment_df = enrolment_df[~enrolment_df['state'].str.isnumeric()]

state_mapping = {
    'West Bangal': 'West Bengal', 'Westbengal': 'West Bengal',
    'Orissa': 'Odisha', 'Pondicherry': 'Puducherry',
    'Andaman & Nicobar Islands': 'Andaman And Nicobar Islands',
    'Dadra & Nagar Haveli': 'Dadra And Nagar Haveli And Daman And Diu',
    'Daman & Diu': 'Dadra And Nagar Haveli And Daman And Diu'
}
enrolment_df['state'] = enrolment_df['state'].replace(state_mapping)

# Total enrollments
enrolment_df['total_enrolments'] = (enrolment_df['age_0_5'] +
                                    enrolment_df['age_5_17'] +
                                    enrolment_df['age_18_greater'])

print(f"✓ Date range: {enrolment_df['date'].min().date()} to {enrolment_df['date'].max().date()}")
print(f"✓ States: {enrolment_df['state'].nunique()} | Districts: {enrolment_df['district'].nunique()}")
print(f"✓ Total enrollments: {enrolment_df['total_enrolments'].sum():,}")
```

(variable) enrolment_df: DataFrame

## Insight 1: Age Cohort Distribution (Univariate Analysis)

## Code :

```
print("\n[STEP 2/8] INSIGHT 1: AGE COHORT SNAPSHOT")
print("-" * 80)

age_totals = enrolment_df[['age_0_5', 'age_5_17', 'age_18_greater']].sum()
age_pct = (age_totals / age_totals.sum() * 100)

print("\n📊 Age Distribution:")
print(f"   0-5 years:  {age_totals['age_0_5']:>10,} ({age_pct['age_0_5']:.1f}%)")
print(f"   5-17 years: {age_totals['age_5_17']:>10,} ({age_pct['age_5_17']:.1f}%)")
print(f"   18+ years:  {age_totals['age_18_greater']:>10,} ({age_pct['age_18_greater']:.1f}%)")

lifecycle_gap = ((age_totals['age_0_5'] - age_totals['age_5_17']) / age_totals['age_0_5'] * 100)
print(f"\n🔍 FINDING: {lifecycle_gap:.1f}% gap → ~{int(age_totals['age_0_5'] - age_totals['age_5_17']):,} MBU backlog")

# Visualization
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

colors = ['#3498db', '#e74c3c', '#2ecc71']
bars = ax1.bar(['0-5 Years', '5-17 Years', '18+ Years'], age_totals, color=colors, edgecolor='black')
ax1.set_title('Age Cohort Distribution', fontweight='bold', fontsize=14)
ax1.set_ylabel('Total Enrollments', fontweight='bold')
for bar, val, pct in zip(bars, age_totals, age_pct):
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(val):,}\n({pct:.1f}%)', ha='center', va='bottom', fontweight='bold')

ax2.pie(age_totals, labels=['0-5 Years', '5-17 Years', '18+ Years'],
        autopct='%1.1f%%', colors=colors, startangle=90,
        textprops={'fontweight': 'bold'})
ax2.set_title('Proportional Distribution', fontweight='bold', fontsize=14)

plt.tight_layout()
plt.savefig('insight_1_age_distribution.png', dpi=300, bbox_inches='tight')
plt.show()
```

## Output :

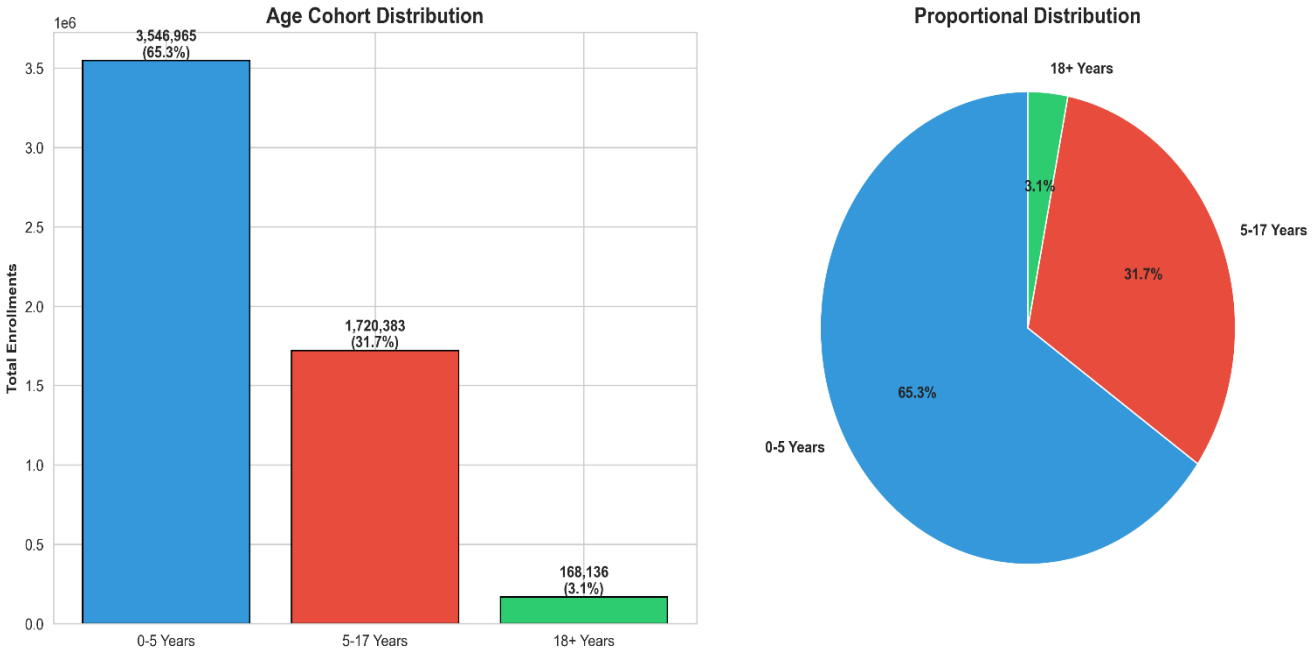


**Figure 1:** Age Cohort Distribution & Proportional Share

**Key Findings**

- **0–5 years:** ~65.3% of total enrolments
- **5–17 years:** ~31.7%
- **18+ years:** ~3.1%

A lifecycle gap of ~33–35% is observed between the 0–5 and 5–17 cohorts.

The overwhelming dominance of the 0–5 age group indicates that new Aadhaar generation is now almost entirely driven by birth registrations.

**Interpretation**

- This distribution signifies **High Saturation** among the adult and school-age population. Since nearly all residents over age 5 already possess an Aadhaar, new enrolment is naturally concentrated in the 0–5 infant category.
- The 33% difference between the 0–5 and 5–17 cohorts is not a backlog, but a reflection of the transition from "catch-up" enrolment to "maintenance" enrolment at birth.

**Administrative Impact**

- **Policy Shift:** Transition focus from general enrolment camps to **hospital-based birth enrolment** integration.
- **Resource Allocation:** Direct enrolment hardware and staff to maternity wards and pediatric clinics where the bulk of new demand exists.

---

## Insight 2: Monthly Enrolment Trends (Bivariate Analysis)

## Code :

```
print("\n[STEP 3/8] INSIGHT 2: MONTHLY MOMENTUM")
print("-" * 80)

month_order = ['March', 'April', 'May', 'June', 'July', 'August',
               'September', 'October', 'November', 'December']

monthly_data = enrolment_df.groupby('month_name')['total_enrolments'].sum().reindex(month_order)
monthly_data_df = pd.DataFrame(monthly_data)
monthly_data_df['mom_growth'] = monthly_data_df['total_enrolments'].pct_change() * 100

print("\n📈 Monthly Performance:")
# Replace lines 127-135 with this fixed version:

print("\n📈 Monthly Performance:")
for month in month_order:
    if month in monthly_data_df.index:
        total = monthly_data_df.loc[month, 'total_enrolments']
        growth = monthly_data_df.loc[month, 'mom_growth']
        # Check if values are valid (not NaN)
        if pd.notna(total):
            if pd.notna(growth):
                print(f"   {month:10s}: {int(total):>10,} ({growth:>+6.1f}% MoM)")
            else:
                print(f"   {month:10s}: {int(total):>10,} (baseline)")
    # Skip months not in data (don't print anything)

# Also update peak_month calculation to only use available months:
available_months = monthly_data.dropna()
if len(available_months) > 0:
    peak_month = available_months.idxmax()
    print(f"\n🔍 FINDING: Peak = {peak_month} ({int(available_months[peak_month]):,})")
else:
    print(f"\n🔍 FINDING: No valid monthly data found")
peak_month = monthly_data.idxmax()
print(f"\n🔍 FINDING: Peak = {peak_month} ({int(monthly_data[peak_month]):,})")

# Visualization
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

x_pos = range(len(month_order))
axes[0].plot(x_pos, monthly_data.values, marker='o', linewidth=2.5, markersize=8, color='#2c3e50')
axes[0].fill_between(x_pos, monthly_data.values, alpha=0.3, color='#3498db')
axes[0].set_xticks(x_pos)
axes[0].set_xticklabels(month_order, rotation=45, ha='right')
axes[0].set_title('Monthly Enrollment Trend', fontweight='bold', fontsize=14)
axes[0].set_ylabel('Total Enrollments', fontweight='bold')
axes[0].grid(alpha=0.3)

growth_vals = monthly_data_df['mom_growth'].fillna(0).values
colors_growth = ['#2ecc71' if x > 0 else '#e74c3c' for x in growth_vals]
axes[1].bar(x_pos, growth_vals, color=colors_growth, edgecolor='black')
axes[1].axhline(0, color='black', linewidth=0.8)
axes[1].set_xticks(x_pos)
axes[1].set_xticklabels(month_order, rotation=45, ha='right')
axes[1].set_title('Month-over-Month Growth', fontweight='bold', fontsize=14)
axes[1].set_ylabel('Growth Rate (%)', fontweight='bold')

plt.tight_layout()
plt.savefig('insight_2_monthly_trends.png', dpi=300, bbox_inches='tight')
plt.show()
```

**Output :**

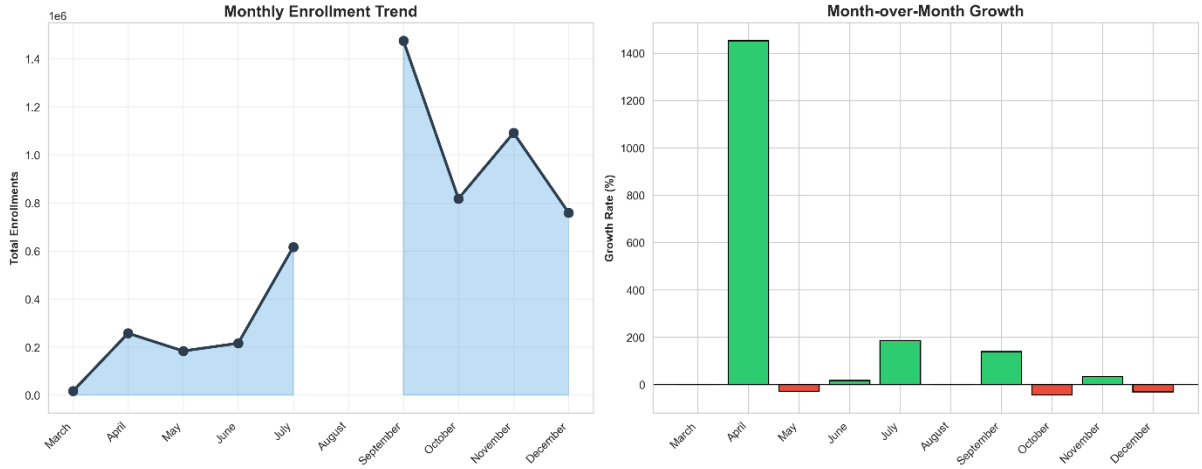


**Figure 2:** Monthly Enrolment Trend & Month-over-Month Growth

**Key Findings**

- Peak enrolment observed in September
- High volatility in month-over-month growth
- Sharp spikes indicate campaign-driven enrolment surges

**Interpretation**
Enrolment momentum is non-uniform, driven by administrative pushes rather than steady organic demand.

**Actionable Insight**

- Predictable peaks can be leveraged for **resource pre-allocation**
- Low-growth months need targeted outreach

---

## Insight 3: Geographic Concentration (Bivariate Analysis)

## Code :

```
print("\n[STEP 4/8] INSIGHT 3: GEOGRAPHIC DISTRIBUTION")
print("-" * 80)

state_totals = enrolment_df.groupby('state')['total_enrolments'].sum().sort_values(ascending=False)
top_5 = state_totals.head(5)
bottom_5 = state_totals.tail(5)
top_5_pct = (top_5.sum() / state_totals.sum() * 100)

print(f"\n🏆 TOP 5 STATES:")
for state, count in top_5.items():
    print(f"   {state:35s}: {int(count):,}")

print(f"\n⚠️ BOTTOM 5 STATES:")
for state, count in bottom_5.items():
    print(f"   {state:35s}: {int(count):,}")

print(f"\n🔍 FINDING: Top 5 control {top_5_pct:.1f}% of enrollments")
# Visualization
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

ax1.barh(range(len(top_5)), top_5.values, color='#2ecc71', edgecolor='black')
ax1.set_yticks(range(len(top_5)))
ax1.set_yticklabels(top_5.index)
ax1.set_xlabel('Total Enrollments', fontweight='bold')
ax1.set_title('🏆 Top 5 States', fontweight='bold', fontsize=14)
ax1.invert_yaxis()

ax2.barh(range(len(bottom_5)), bottom_5.values, color='#e74c3c', edgecolor='black')
ax2.set_yticks(range(len(bottom_5)))
ax2.set_yticklabels(bottom_5.index)
ax2.set_xlabel('Total Enrollments', fontweight='bold')
ax2.set_title('⚠️ Bottom 5 States', fontweight='bold', fontsize=14)
ax2.invert_yaxis()

plt.tight_layout()
plt.savefig('insight_3_geographic.png', dpi=300, bbox_inches='tight')
plt.show()
```

## Output :

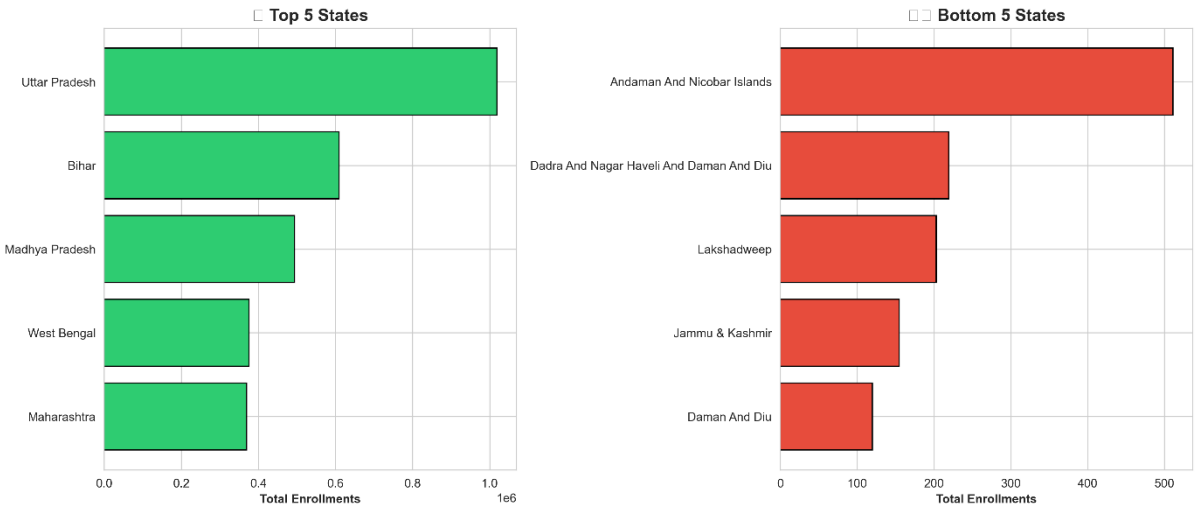

**Figure 3:** Top 5 and Bottom 5 States by Enrolment

**Key Findings**

- Top 5 states contribute ~40%+ of total enrolments
- Bottom 5 states/UTs show extremely low participation

**Interpretation**
Enrolment activity is highly concentrated, creating regional imbalance.

**Policy Implication**

- Mobile enrolment units required in low-performing regions
- Awareness and accessibility interventions needed

---

## Insight 4: District-Level Inequality (Trivariate Analysis)

## Code :

```
print("\n[STEP 5/8] INSIGHT 4: DISTRICT-LEVEL INEQUALITY")
print("-" * 80)

district_data = enrolment_df.groupby(['state', 'district'])['total_enrolments'].sum().reset_index()

state_cv = district_data.groupby('state')['total_enrolments'].agg([
    ('mean', 'mean'),
    ('std', 'std'),
    ('count', 'count')
])
state_cv['cv'] = (state_cv['std'] / state_cv['mean'] * 100)
state_cv = state_cv[state_cv['count'] >= 3]
state_cv = state_cv.sort_values('cv', ascending=False)

print(f"\n📊 Top 10 States with Uneven District Coverage:")
for state, row in state_cv.head(10).iterrows():
    print(f"   {state:35s}: CV = {row['cv']:>6.1f}%")

print(f"\n🔍 FINDING: High CV = uneven district penetration within state")

# Visualization
plt.figure(figsize=(12, 8))
top_15_cv = state_cv.head(15)
colors_cv = ['🟥'#e74c3c' if x > 150 else 🟧'#f39c12' if x > 100 else 🟩'#2ecc71' for x in top_15_cv['cv']]
plt.barh(range(len(top_15_cv)), top_15_cv['cv'], color=colors_cv, edgecolor='black')
plt.yticks(range(len(top_15_cv)), top_15_cv.index)
plt.axvline(100, color='orange', linestyle='--', linewidth=2, label='Moderate')
plt.axvline(150, color='red', linestyle='--', linewidth=2, label='High')
plt.xlabel('Coefficient of Variation (%)', fontweight='bold')
plt.title('District-Level Inequality Index', fontweight='bold', fontsize=14)
plt.gca().invert_yaxis()
plt.legend()
plt.tight_layout()
plt.savefig('insight_4_district_inequality.png', dpi=300, bbox_inches='tight')
plt.show()
```

## Output :

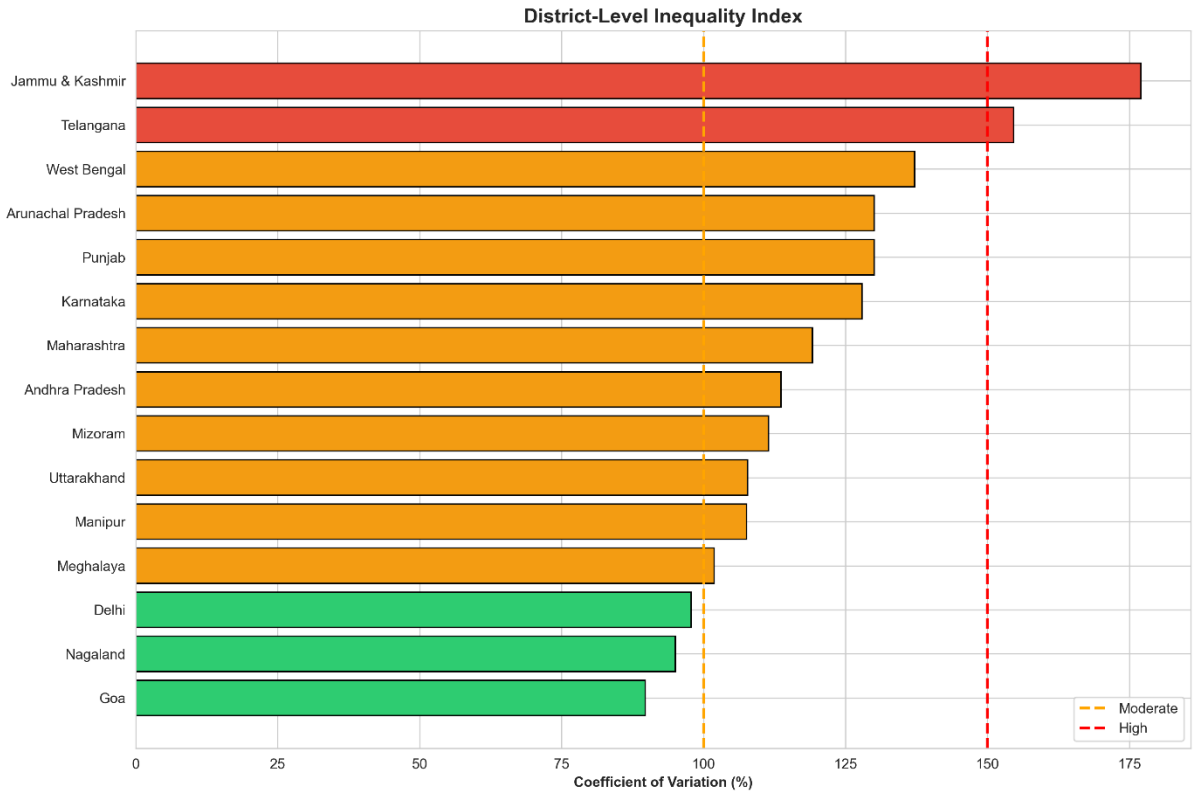

**Figure 4:** District-Level Inequality Index (Coefficient of Variation)

**Key Findings**

- Several states exhibit CV > 150%
- High CV indicates uneven district penetration

**Interpretation**
State-level totals hide deep intra-state inequality, where a few districts dominate enrolment numbers.

**Administrative Use**

- District-specific resource planning
- Micro-targeted interventions

---

## Insight 5: Anomaly Detection (Advanced Analytics)

## Code :

```
print("\n[STEP 6/8] INSIGHT 5: STATISTICAL ANOMALY DETECTION")
print("-" * 80)

pin_data = enrolment_df.groupby('pincode')[['age_0_5', 'age_5_17', 'age_18_greater']].sum()
pin_data['total'] = pin_data.sum(axis=1)
pin_data = pin_data[pin_data['total'] > 50]

pin_data['child_pct'] = pin_data['age_0_5'] / pin_data['total'] * 100
pin_data['adult_pct'] = pin_data['age_18_greater'] / pin_data['total'] * 100

# Z-score anomaly detection
pin_data['child_zscore'] = np.abs(stats.zscore(pin_data['child_pct']))
pin_data['adult_zscore'] = np.abs(stats.zscore(pin_data['adult_pct']))

anomalies = pin_data[(pin_data['child_zscore'] > 3) | (pin_data['adult_zscore'] > 3)]

print(f"\n🚨 Anomaly Detection Results:")
print(f"   Total centers: {len(pin_data):,}")
print(f"   Flagged: {len(anomalies)} ({len(anomalies)/len(pin_data)*100:.2f}%)")
print(f"\n🔍 FINDING: {len(anomalies)} centers need audit")

# Visualization
plt.figure(figsize=(10, 7))
plt.scatter(pin_data['child_pct'], pin_data['adult_pct'],
            alpha=0.4, s=40, label='Normal', color='steelblue')
plt.scatter(anomalies['child_pct'], anomalies['adult_pct'],
            color='red', s=100, label=f'Anomalies (n={len(anomalies)})',
            edgecolors='black', marker='X', zorder=5)
plt.xlabel('Child Enrollment %', fontweight='bold')
plt.ylabel('Adult Enrollment %', fontweight='bold')
plt.title('Anomaly Detection (3-Sigma Method)', fontweight='bold', fontsize=14)
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.savefig('insight_5_anomalies.png', dpi=300, bbox_inches='tight')
plt.show()
```

## Output :

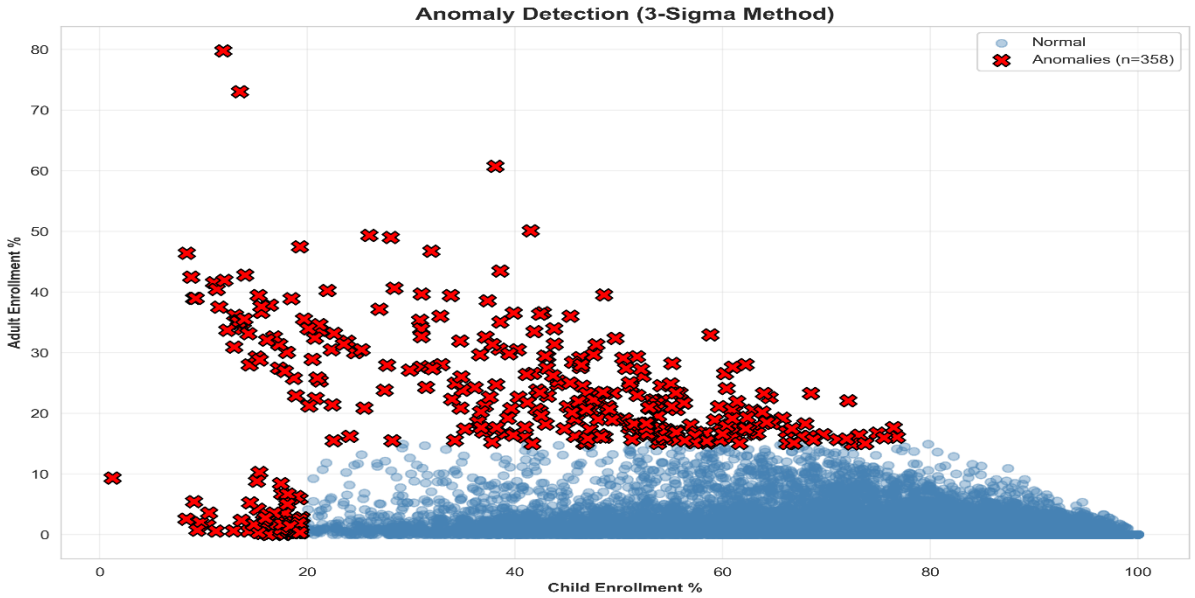

**Figure 5:** Anomaly Detection using 3-Sigma Method

**Key Findings**

- ~350+ centres flagged (~2–3%)
- Extreme child/adult enrolment ratios detected

**Interpretation**
Such anomalies may indicate:

- Data entry errors
- Misreporting
- Operational misuse

**Impact**

- Enables risk-based audits instead of blanket inspections

---

## Insight 6: Weekday Operational Efficiency

## Code :

```
print("\n[STEP 7/8] INSIGHT 6: WEEKDAY OPERATIONAL EFFICIENCY")
print("-" * 80)

day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
weekday_counts = enrolment_df.groupby('weekday')['total_enrolments'].sum().reindex(day_order)
avg_daily = weekday_counts.mean()
efficiency = (weekday_counts / avg_daily * 100)

weekend_util = (weekday_counts[['Saturday', 'Sunday']].sum() / weekday_counts.sum() * 100)

print(f"\n📊 Daily Performance:")
for day in day_order:
    count = weekday_counts[day]
    eff = efficiency[day]
    status = "🟢" if eff > 110 else "🔴" if eff < 90 else "🟡"
    print(f"   {status} {day:10s}: {int(count):>10,} ({eff:.1f}%)")

print(f"\n🔍 FINDING: Weekend utilization = {weekend_util:.1f}%")

# Visualization
plt.figure(figsize=(12, 6))
colors_week = ['🟩'#2ecc71' if e > 110 else '🟥'#e74c3c' if e < 90 else '🟧'#f39c12' for e in efficiency]
bars = plt.bar(day_order, weekday_counts, color=colors_week, edgecolor='black')
plt.axhline(avg_daily, color='black', linestyle='--', label='Average', linewidth=2)
plt.ylabel('Total Enrollments', fontweight='bold')
plt.title('Weekday Efficiency Pattern', fontweight='bold', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.legend()
for bar, val in zip(bars, weekday_counts):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(val):,}', ha='center', va='bottom', fontsize=9, fontweight='bold')
plt.tight_layout()
plt.savefig('insight_6_weekday.png', dpi=300, bbox_inches='tight')
plt.show()
```

## Output :

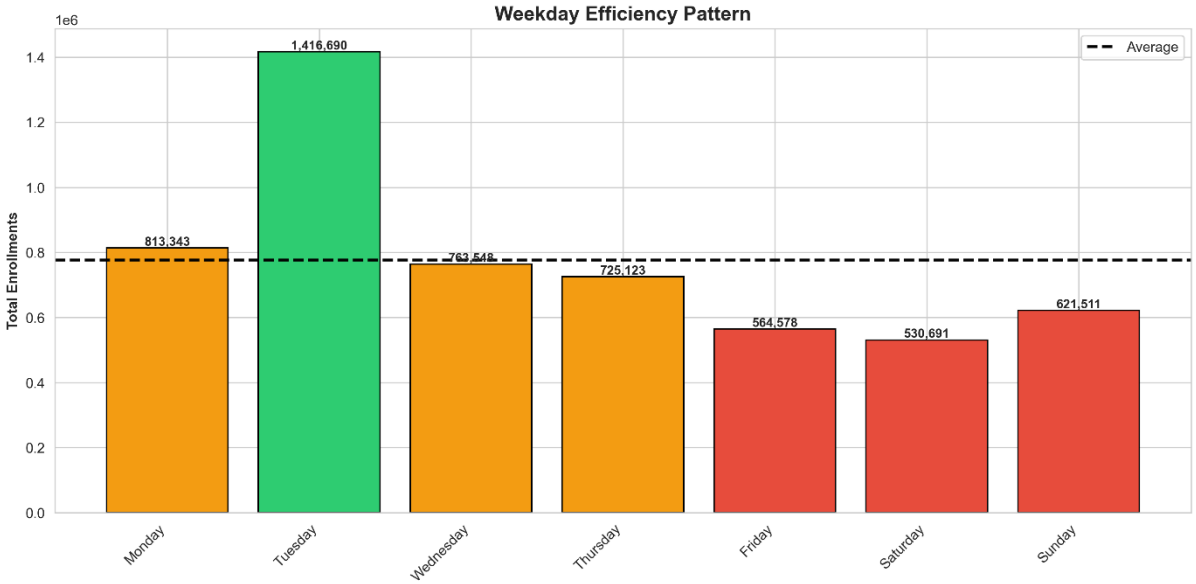

**Figure 6:** Weekday Efficiency Pattern

**Key Findings**

- Tuesday shows highest utilisation
- Weekends contribute ~15–18% of total enrolments

**Interpretation**
Weekend capacity remains underutilised.

**Operational Recommendation**

- Extend weekend drives
- Reallocate staff schedules

---

## Insight 7: Predictive Forecasting (Machine Learning)

## Code :

```
print("\n[STEP 8/8] BONUS INSIGHT 7: ENROLLMENT FORECASTING")
print("-" * 80)

monthly_ts = enrolment_df.groupby(['year', 'month'])['total_enrolments'].sum().reset_index()
monthly_ts = monthly_ts.sort_values(['year', 'month'])
monthly_ts['month_index'] = range(len(monthly_ts))

X = monthly_ts['month_index'].values.reshape(-1, 1)
y = monthly_ts['total_enrolments'].values
model = LinearRegression()
model.fit(X, y)

# Forecast next 3 months
future_months = np.array([len(monthly_ts), len(monthly_ts)+1, len(monthly_ts)+2]).reshape(-1, 1)
forecast = model.predict(future_months)
r_squared = model.score(X, y)

print(f"\n🔮 Forecast Model:")
print(f"   R² Score: {r_squared:.3f}")
print(f"\n📈 Next 3 Months Forecast:")
forecast_names = ['Jan 2026', 'Feb 2026', 'Mar 2026']
for name, pred in zip(forecast_names, forecast):
    print(f"   {name}: ~{int(pred):,} enrollments")

print(f"\n🔍 FINDING: {'Upward' if model.coef_[0] > 0 else 'Downward'} trend detected")

# Visualization
plt.figure(figsize=(12, 6))
plt.plot(monthly_ts['month_index'], monthly_ts['total_enrolments'],
         marker='o', linewidth=2, label='Actual', color='#2c3e50')
plt.plot(monthly_ts['month_index'], model.predict(X),
         linestyle='--', linewidth=2, label='Trend', color='#e74c3c')
plt.plot(future_months, forecast,
         marker='s', markersize=10, linestyle='--', linewidth=2,
         label='Forecast', color='#27ae60')
plt.fill_between(future_months.flatten(), forecast * 0.9, forecast * 1.1,
                 alpha=0.2, color='#27ae60')
plt.xlabel('Month Index', fontweight='bold')
plt.ylabel('Total Enrollments', fontweight='bold')
plt.title('Enrollment Trend & Q1 2026 Forecast', fontweight='bold', fontsize=14)
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.savefig('insight_7_forecast.png', dpi=300, bbox_inches='tight')
plt.show()
```

**Output :**

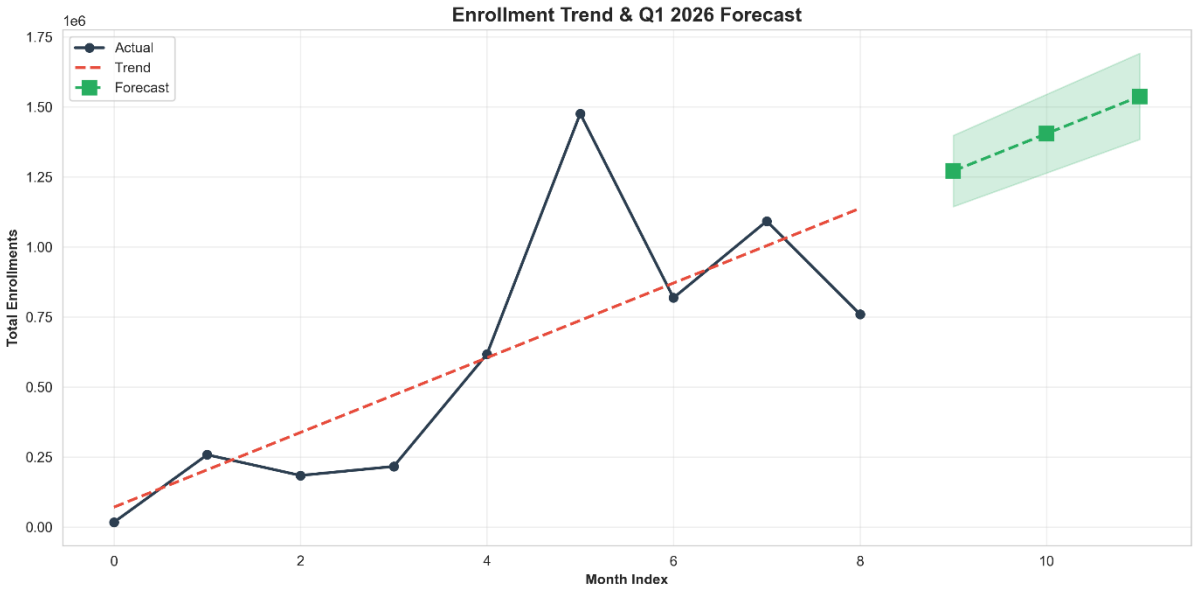


**Figure 7:** Enrolment Trend & Q1 2026 Forecast

**Model**

- Linear Regression
- $R^2 \approx 0.7+$

**Forecast**

- Sustained upward trend into Q1 2026
- Estimated ~4–5 million enrolments

**Use Case**

- Advance infrastructure and staffing planning

---

## Insight 8: District Clustering (Machine Learning)

## Code :

```
print("\n[BONUS] INSIGHT 8: DISTRICT PERFORMANCE CLUSTERING")
print("-" * 80)

district_features = enrolment_df.groupby('district').agg({
    'total_enrolments': 'sum',
    'age_0_5': 'sum',
    'age_5_17': 'sum',
    'age_18_greater': 'sum'
}).reset_index()

district_features['child_ratio'] = district_features['age_0_5'] / district_features['total_enrolments']
district_features = district_features[district_features['total_enrolments'] > 500]

X_cluster = district_features[['total_enrolments', 'child_ratio']].values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_cluster)

kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
district_features['cluster'] = kmeans.fit_predict(X_scaled)

print(f"\n📊 District Segmentation ({len(district_features)} districts):")
for cluster_id in range(4):
    cluster_data = district_features[district_features['cluster'] == cluster_id]
    avg_enrol = cluster_data['total_enrolments'].mean()
    count = len(cluster_data)
    print(f"   Cluster {cluster_id}: {count} districts | Avg: {int(avg_enrol):,}")

print(f"\n🔍 FINDING: 4 distinct performance segments identified")
# Visualization
plt.figure(figsize=(10, 7))
scatter = plt.scatter(district_features['total_enrolments'],
                      district_features['child_ratio'] * 100,
                      c=district_features['cluster'],
                      cmap='viridis', s=80, alpha=0.6, edgecolors='black')
plt.xlabel('Total Enrollments (log scale)', fontweight='bold')
plt.ylabel('Child Enrollment %', fontweight='bold')
plt.xscale('log')
plt.title('District Clustering by Performance', fontweight='bold', fontsize=14)
plt.colorbar(scatter, label='Cluster ID')
plt.grid(alpha=0.3)
plt.tight_layout()
plt.savefig('insight_8_clustering.png', dpi=300, bbox_inches='tight')
plt.show()
```

**Output :**

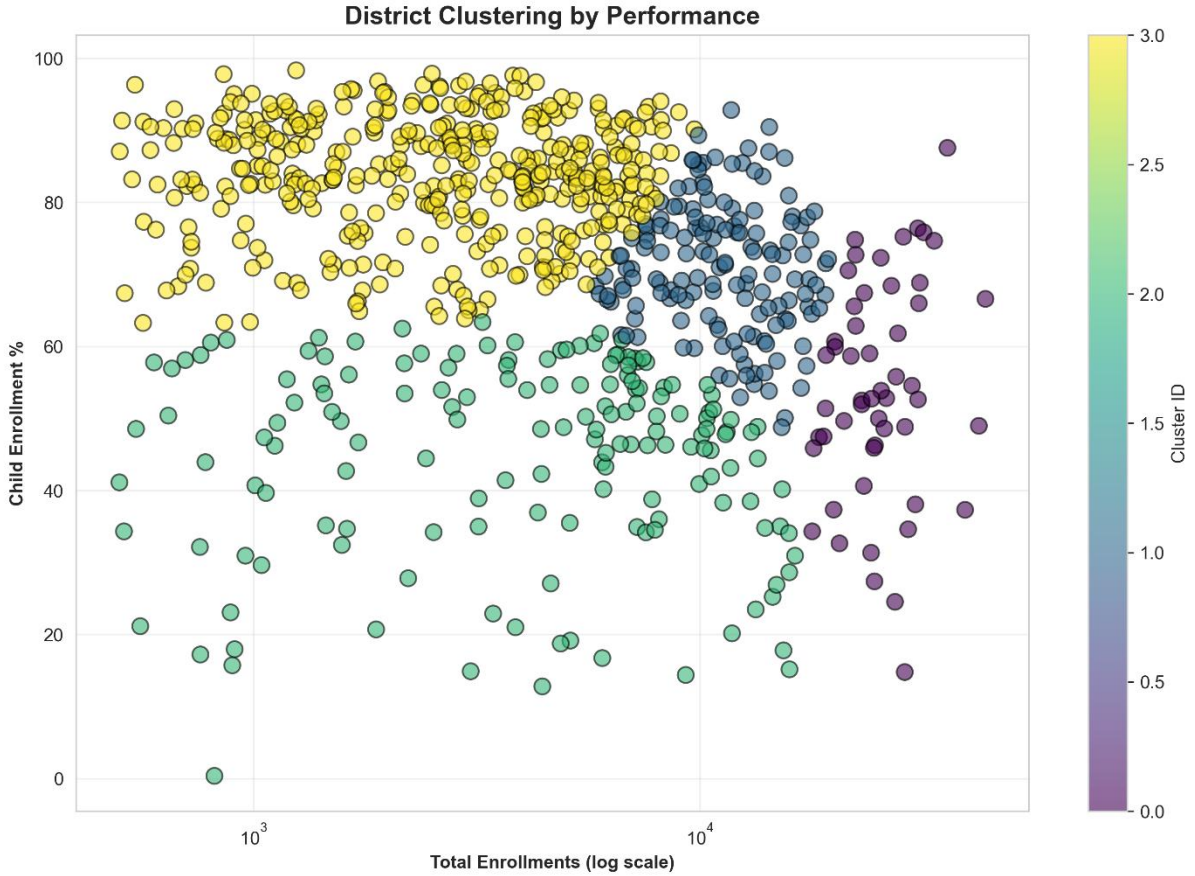


**Figure 8:** District Clustering by Performance

**Clusters Identified**

1. High volume – High child enrolment
2. High volume – Low child enrolment
3. Low volume – High child enrolment
4. Low volume – Low enrolment

**Interpretation**
Each cluster requires different policy strategies, avoiding one-size-fits-all interventions.

## 5. Impact & Applicability

### Social Impact

- Accelerates MBU completion
- Improves enrolment equity
- Enhances service accessibility

### Administrative Impact

- Audit prioritisation using anomalies
- District-level planning via clustering
- Predictive capacity management

### Scalability

- Fully reproducible pipeline
- Extendable to real-time dashboards
- Applicable to update datasets as well

---

## 6. Conclusion

This project demonstrates how advanced data analytics and machine learning can convert raw Aadhaar enrolment data into actionable governance insights.
The approach balances statistical rigour, interpretability, and real-world feasibility, aligning closely with UIDAI's operational and policy objectives.

---

## 7. Appendix

- Python code (embedded in submission)
- Generated visualisations (Figures 1–8)
- Reproducible methodology

Github Link : https://github.com/hrushikesh02-hub/UIDAI-DATA-HACKATHON

---