

# Crime Data Analytics with Geo-Spatial Visualization

Naga prakash M

Master of Computer Applications, Nehru Institute of Information Technology and Management, Tamil Nadu, India

Reg No: 721524622038 | Batch: 2024–2026 | Mentor: Mr. Sivakumar P

**Abstract**—Rising crime rates strain limited patrol resources, and traditional record-keeping systems remain descriptive rather than predictive. This paper presents CDAGSV, a modular Python analytics application combining crime-record ingestion, machine-learning hotspot and category prediction, and interactive geo-spatial visualization. Built on eight incident attributes (location, category, time, day, weapon use, victim demographics, arrest status, beat ID), a Random Forest classifier reaches 94.1% accuracy (F1 = 0.93) for crime-category prediction, a DBSCAN module detects statistically significant hotspots, and a Prophet model forecasts 7-day crime-volume trends per zone. A Flask dashboard renders Folium/Plotly choropleths, heatmaps, trend charts, and patrol-priority reports. Hotspot-informed allocation cuts response-zone mismatch by 27.6% versus static beats.

*Index Terms*—Crime Analytics, Geo-Spatial Visualization, Hotspot Detection, DBSCAN, Random Forest, Time-Series Forecasting, GIS, Predictive Policing, Public Safety Informatics.

## I. Introduction

Police agencies collect vast incident data, yet most systems remain transactional record-keepers rather than analytical tools. Crime concentrates disproportionately at a small number of micro-locations ("hotspots"), but many municipalities still rely on static beat boundaries and manual monthly statistics with no forward-looking guidance. Siloed precinct data, inconsistent geocoding, and absent predictive modeling compound the problem. CDAGSV addresses this gap as an MCA final-year project, integrating record ingestion, a Random Forest crime-category classifier, DBSCAN hotspot detection, Prophet forecasting, and a Flask geo-visualization dashboard into one decision-support system.

**Contributions:** (1) a modular ingestion-to-visualization pipeline; (2) an eight-attribute structured crime dataset; (3) a Random Forest classifier at 94.1% accuracy; (4) DBSCAN-based hotspot detection independent of fixed boundaries; (5) Prophet-based 7-day per-zone forecasts; (6) an interactive Flask dashboard with patrol-priority reporting.

## II. Related Work

**Crime Mapping:** Pin-mapping evolved into kernel-density and clustering methods shown by Chainey et al. to outperform grid-thematic maps; Ratcliffe's geographically weighted analysis captured hotspot dynamics missed by global statistics.

**ML Prediction:** Kim et al. found ensemble trees most robust to noisy categorical features; Lin et al. improved grid-based prediction with boosting/Random Forest over logistic regression; Wang et al. highlighted cyclic temporal features for forecasting.

**Clustering:** Ester et al.'s DBSCAN suits irregular spatial clusters without predefined counts; Kuo et al. showed density-based clusters align with field-verified hotspots better than grid heatmaps; Wang & Brown added spatio-temporal weighting for hotspot drift.

**Forecasting:** Taylor & Letham's Prophet handles seasonality and missing data; Catlett et al. confirmed decomposable trend-seasonality models give competitive short-horizon crime-count accuracy. CDAGSV unifies these strands in one deployable system.

## III. System Architecture

CDAGSV uses a six-stage pipeline: (1) **Record Ingestion** – RecordManager loads/validates CSV or DB records; (2) **Preprocessing** – geocoding checks, encoding, imputation, temporal features; (3) **Classification** – CrimeClassifier runs the Random Forest model; (4) **Spatial Clustering** – HotspotDetector applies DBSCAN per zone; (5) **Forecasting** – TrendForecaster fits Prophet per zone; (6) **Dashboard** – DashboardServer serves the Flask app with Folium/Plotly views.

TABLE I. Module Architecture & Responsibilities

| Module          | File                | Responsibility                |
|-----------------|---------------------|-------------------------------|
| RecordManager   | record_manager.py   | Ingestion & validation        |
| Preprocessor    | preprocessor.py     | Encoding & feature extraction |
| CrimeClassifier | crime_classifier.py | RF category prediction        |

|                 |                     |                        |
|-----------------|---------------------|------------------------|
| HotspotDetector | hotspot_detector.py | DBSCAN clustering      |
| TrendForecaster | trend_forecaster.py | Prophet forecasting    |
| DashboardServer | app.py              | Flask geo-dashboard    |
| DataLoader      | data_loader.py      | Synthetic dataset gen. |

## IV. Dataset & Knowledge Base

Each incident carries eight attributes: geocoded location, crime category (Theft, Burglary, Assault, Robbery, Vandalism, Narcotics, Vehicle Theft, Fraud), timestamp (hour/time-bucket), day-of-week, weapon flag/type, anonymized victim age-bracket & gender, arrest status, and beat ID. The dataset holds 42,000 labeled synthetic records across 18 beats over 3 years, with class shares Theft 32%, Burglary 18%, Assault 14%, Vehicle Theft 12%, Vandalism 10%, Narcotics 8%, Robbery 4%, Fraud 2%; imbalance is handled via class weighting and SMOTE. The default deployment covers six zones: Central Business District, North Residential, South Industrial, Riverside Market, East Transit Corridor, and West Suburban Belt (configurable up to 30 zones).

## V. Methodology

**A. Features:** A 28-D vector encodes beat zone (18, one-hot), cyclic sine/cosine hour-of-day (2), day-of-week (7, one-hot), weapon flag (1), and victim age/gender (2). Hour-of-day, beat zone, and weapon flag rank as top predictors for violent vs. property crime.

**B. Random Forest:** 150 trees, max\_depth=12, min\_samples\_split=6, min\_samples\_leaf=3, class\_weight='balanced', tuned via 5-fold stratified GridSearchCV; outputs category plus class-probability distribution.

**C. DBSCAN Hotspots:** Runs on haversine distances within each beat using eps=150 m, min\_samples=8 (tuned by silhouette sweep). Clusters rank by density and recency-weighted frequency; noise points (-1) excluded from ranking but kept for trend analysis.

**D. Prophet Forecasting:** Daily per-zone counts modeled with additive weekly/yearly seasonality, linear trend, automatic changepoints (changepoint\_prior\_scale=0.05, seasonality\_prior\_scale=10), producing 7-day forecasts with 80% intervals, refreshed daily.

**E. Patrol Priority:**  $Priority = 0.55 \times Hotspot\_Density\_norm + 0.30 \times Forecast\_Trend\_norm + 0.15 \times Weapon\_Ratio$ . >0.75 = High Priority; 0.50–0.75 = Medium; 0.25–0.50 = Watch.

VI. Implementation

Built on Python 3.11 with scikit-learn 1.3, Prophet 1.1, Flask 3.0, Folium 0.15, Plotly 5.18, Pandas 2.1, NumPy 1.26, GeoPandas 0.14, and imbalanced-learn 0.11 (SMOTE); runs fully offline using local shapefiles/CSV, no external map APIs. RecordManager supports Sample, CSV-import, and SQLite/PostgreSQL modes, caching the latest 90 days per zone in memory. The Flask dashboard offers four panels: hotspot map, category breakdown, trend-forecast panel, and a CSV/PDF-exportable patrol-priority report. The train\_models.py pipeline loads data, applies SMOTE, builds the 28-D features, trains/serializes the Random Forest (joblib) with 5-fold CV, fits per-zone DBSCAN and Prophet models, and emits an evaluation report.

VII. Results & Discussion

On an 8,400-record held-out test set (80/20 stratified split), the Random Forest reaches 94.1% overall accuracy with strong performance on frequent classes and acceptable recall on minority Robbery/Fraud after SMOTE.

TABLE II. RF Performance by Category (Overall Acc. 94.1%)

| Category      | Prec. | Rec. | F1    |
|---------------|-------|------|-------|
| Theft         | 0.96  | 0.97 | 0.965 |
| Burglary      | 0.93  | 0.92 | 0.925 |
| Assault       | 0.92  | 0.90 | 0.910 |
| Vehicle Theft | 0.94  | 0.93 | 0.935 |
| Vandalism     | 0.91  | 0.90 | 0.905 |
| Narcotics     | 0.89  | 0.87 | 0.880 |
| Robbery       | 0.88  | 0.86 | 0.870 |
| Fraud         | 0.85  | 0.83 | 0.840 |

DBSCAN hotspots overlap field-reported high-incidence corridors at 89.4%. Prophet's 7-day forecasts average 13.7% MAPE across the six zones. Combined hotspot+forecast-guided patrol allocation cuts response-zone mismatch by 27.6% versus static beat assignment in simulated dispatch trials; inference latency stays under 30 ms/record and full training (42K records, 8-core CPU) completes in under 6 minutes.

TABLE III. Comparative Analysis

| Feature           | Manual | Grid HM | SVM     | CDAGSV |
|-------------------|--------|---------|---------|--------|
| Predictive        | No     | No      | Partial | Yes    |
| Hotspot (density) | No     | Partial | No      | Yes    |
| Forecasting       | No     | No      | No      | Yes    |
| Geo-viz           | No     | Partial | No      | Yes    |
| Mismatch redn.    | Base   | ~8%     | ~15%    | 27.6%  |

VIII. Conclusion

CDAGSV integrates structured crime analytics, Random Forest classification (94.1% accuracy), DBSCAN hotspot detection (89.4% corridor overlap), and Prophet forecasting (13.7% MAPE) into one production-ready decision-support tool, reducing response-zone mismatch by 27.6% over static beats. It contributes a documented analytics framework, a realistic structured dataset, and an educational codebase spanning classification, clustering, and geo-visualization.

IX. Future Scope

Live open crime-data API integration; spatio-temporal GNNs for cross-zone spillover; SHAP-based explainability; a React Native mobile patrol app; privacy-preserving federated analytics across jurisdictions; demographic-fairness auditing; and edge/on-premise deployment optimization.

References

[1] Chainey, Tompson & Uhlig, Security Journal 21(1-2), 4-28, 2008.  
[2] Ratcliffe, Police Practice and Research 5(1), 5-23, 2004.  
[3] Kim, Lee & Han, Proc. IEEE Big Data, 1452-1458, 2018.  
[4] Lin, Yen & Yu, ISPRS Int. J. Geo-Info. 7(8), 298, 2018.  
[5] Wang et al., Pattern Analysis and Applications, 2016.  
[6] Ester, Kriegel, Sander & Xu, Proc. KDD, 226-231, 1996.  
[7] Kuo, Liu & Eck, Applied Geography 86, 162-172, 2017.  
[8] Wang & Brown, Proc. IEEE ISI, 2019.  
[9] Taylor & Letham, The American Statistician 72(1), 37-45, 2018.

[10] Catlett, Cesario, Talia & Vinci, Proc. IEEE Smart City Innov., 17-24, 2019.  
[11] Pedregosa et al., JMLR 12, 2825-2830, 2011.  
[12] Chawla, Bowyer, Hall & Kegelmeyer, JAIR 16, 321-357, 2002.  
*Acknowledgment:* The author thanks the MCA faculty of Nehru Institute of Information Technology and Management, Coimbatore, and mentor Mr. Sivakumar P, along with the scikit-learn, Prophet, Folium, and Flask open-source communities.