Safe Routing System for Delhi NCR: A Crime-Aware Navigation Approach Using Machine Learning and OpenStreetMap

Vaibhav Singh
Department of Computer Science
Shiv Nadar University - Noida, India
Email: vs450@snu.edu.in
Github : Live — Repo

***Abstract*—Traditional navigation systems optimize routes based solely on distance and time, ignoring critical safety concerns related to crime. This paper presents a crime-aware routing system for Delhi NCR that integrates historical crime data with machine learning to provide safety-optimized navigation. We employ a Random Forest Regressor to predict crime risk scores across geographic regions using spatial interpolation with KD-trees. The system utilizes a modified Dijkstra's algorithm with crime-weighted edges on OpenStreetMap data to generate three route alternatives: shortest (distance-optimized), safest (crime-risk minimized), and balanced (multi-objective). Our implementation achieves real-time route computation in 2-5 seconds through efficient graph caching. The system covers Central Delhi (28.50-28.75°N, 77.05-77.35°E) with 150+ crime data points, and demonstrates practical applicability for safer urban navigation.**

***Index Terms*—Crime-aware routing, safe navigation, Random Forest, OpenStreetMap, spatial analysis, route optimization, Delhi NCR**

I. Introduction

A. Motivation

Urban crime is a significant concern in metropolitan areas, with criminal activity varying considerably across different geographic locations. Delhi NCR, one of India's largest metropolitan regions, faces substantial challenges related to urban safety. Traditional navigation systems like Google Maps optimize routes for distance and time but do not consider the safety implications of traversing high-crime areas.

This creates a critical gap where users, particularly vulnerable populations such as women, elderly individuals, and tourists, lack information to make safety-informed navigation decisions. Our system addresses this gap by integrating historical crime data into route planning.

B. Problem Statement

Existing navigation systems suffer from:

- No consideration of crime risk in route calculation
- Lack of safety metrics for route comparison
- Single route suggestion without alternatives
- No mechanism for users to prioritize safety over distance

C. Our Contribution

This paper presents a complete crime-aware routing system that:

1) Uses Random Forest Regressor to predict crime risk for any location
2) Implements spatial interpolation using KD-trees for efficient risk queries
3) Generates three distinct routes using modified Dijkstra's algorithm
4) Achieves real-time performance through graph and model caching
5) Provides a web-based interface using Flask and Leaflet.js

II. Related Work

A. Crime Prediction Using Machine Learning

Machine learning has been extensively applied to crime prediction. Random Forest, Support Vector Machines, and Neural Networks are commonly used algorithms. Random Forest is particularly effective for spatial crime data due to its ability to handle non-linear relationships and feature importance analysis.

B. Safe Routing Systems

Galbrun et al. [1] proposed safe route recommendations using street crime data. Their work focused on pedestrian navigation with limited scalability. Saha et al. [2] developed multi-constrained optimization but required extensive sensor infrastructure.

C. Safe Path Recommender

A GIS-based safe path recommender uses geographic information and safety-related data to suggest the safest route instead of just the shortest one. It analyzes factors like crime levels, traffic, lighting, and environmental risks to compute safer paths. This system helps users travel securely, especially in unfamiliar or high-risk areas.

D. OpenStreetMap for Routing

OSMnx library by Boeing [3] provides tools for downloading and analyzing street networks from OpenStreetMap. This has enabled researchers to build routing applications without proprietary map data.

III. Methodology

A. Crime Dataset

Our dataset contains crime statistics for Delhi NCR with the following attributes: The dataset covers 150+ police station jurisdictions in Delhi NCR with historical crime data.

B. Feature Engineering

1) Crime Severity Weighting: We developed probability-based crime severity weights that sum to 1.0, reflecting both severity and frequency:

S = 0.20 $\cdot murder + 0.18 \cdot rape + 0.18 \cdot gangrape$
$+ 0.05 \cdot robbery + 0.04 \cdot theft + 0.19 \cdot assault + 0.16 \cdot harassment$

TABLE I
CRIME DATASET FEATURES

Feature	Description
nm_pol	Police station name
murder	Number of murder cases
rape	Number of rape cases
gangrape	Number of gang rape cases
robbery	Number of robbery cases
theft	Number of theft cases
assualt murders	Assault-related murders
sexual harassement	Harassment cases
totalcrime	Total crime count
totarea	Area covered (sq meters)
lat, long	Geographic coordinates

TABLE II
CRIME SEVERITY WEIGHTS (PROBABILITY DISTRIBUTION)

Crime Type	Weight	Rationale
Murder	0.20	Highest severity
Assault Murder	0.19	Highest severity
Sexual Harassment	0.16	Medium severity
Gang Rape	0.18	High severity
Rape	0.18	High severity
Robbery	0.05	Lower priority
Theft	0.04	Lower priority

2) Feature Set: The model uses 11 engineered features:

- **Individual crime counts**: Murder, Rape, Gang Rape, Robbery, Theft, Assault Murder, Sexual Harassment
- Total crime count
- Total area in sq. km
- **Derived features**: Crime density (crimes per sq. km), Crime severity score

Crime density is calculated as:

$$\text{Crime Density} = \frac{\text{Total Crime Count}}{\text{Area (sq. km)}}$$

3) Target Variable: Crime risk score (0-100 scale) normalized from crime density:

$$\text{Risk Score} = \frac{\text{Density} - \text{Density}_{\text{min}}}{\text{Density}_{\text{max}} - \text{Density}_{\text{min}}} \times 100$$

C. Crime Risk Prediction Model

1) Random Forest Regressor: The Random Forest model learns non-linear relationships between crime types and overall risk.

```
Algorithm 1 Model Training Process
Require: Crime dataset $D$
Ensure: Trained model $M$ and scaler $S$
1: Load crime data from CSV
2: Preprocess: compute severity, density, risk scores
3: Extract features $X$ and target $y$
4: Normalize features: $X_{scaled} = StandardScaler(X)$
5: Train Random Forest: $M = RF(X_{scaled}, y)$
6: Save model $M$ and scaler $S$ to disk
7: return $M, S$
```

2) Model Training: Training occurs once and the model is cached for subsequent use.

1) **Data split**: 122 training samples, 31 test samples
2) **Feature scaling**: StandardScaler normalization applied
3) **Model training**: RandomForestRegressor trained on scaled features
4) **Evaluation**: R², MAE, RMSE computed on test set

D. Spatial Risk Interpolation

1) KD-Tree Construction: A KD-tree is a spatial indexing structure that efficiently organizes geographic coordinates for fast nearest-neighbor retrieval. By storing all crime data points in a KD-tree, the system can quickly identify nearby crime locations for any queried point in the city. This enables $O(\log n)$ nearest-neighbor queries.

2) Inverse Distance Weighting (IDW): For locations without direct crime data, we interpolate using IDW:

```
Algorithm 2 Spatial Risk Query using IDW Interpolation
Require: Query location $(lat, lon)$, search radius $r$
Ensure: Estimated risk score $\hat{r}$
1: Compute distances $d_i = \sqrt{(lat_i - lat)^2 + (lon_i - lon)^2}$ for all neighbors
2: Compute IDW weights $w_i = \frac{1}{d_i + \epsilon}$
3: Normalize weights $w_i = \frac{w_i}{\sum_j w_j}$
4: Compute estimated risk $\hat{r} = \sum_i (risk_i \cdot w_i)$
5: return $\hat{r}$
```

We use $\epsilon = 0.0001$ to avoid division by zero.

E. Route Optimization

1) Graph Construction from OpenStreetMap: We download street network using OSMnx:

```python
import osmnx as ox

graph = ox.graph_from_bbox(
    north=28.75, south=28.50,
    east=77.35, west=77.05,
    network_type='drive', simplify=True )
```

Listing 1. OSM Graph Loading

This creates a directed graph $G = (V, E)$ where:

- V = intersections and road endpoints
- E = road segments with attributes (length, coordinates)

For Delhi, this yields approximately 1,37,958 nodes and 3,76,041 edges.

2) Edge Weight Computation: For each edge e, we compute three weight types:

1. Original Length:

$$w_{length}(e) = length_e \tag{1}$$

2. Safe Weight:

$$w_{safe}(e) = length_e \times \left(1 + 2 \times \frac{risk_e}{100}\right) \tag{2}$$

where $risk_e$ is the crime risk at the edge midpoint. The factor of 2 means high-crime roads can be up to 3× more costly.

3. Balanced Weight:

$$w_{balanced}(e) = 0.6 \times w_{length}(e) + 0.4 \times w_{safe}(e) \tag{3}$$

This balances distance (60%) and safety (40%).

3) Multi-Route Generation: We apply Dijkstra's shortest path algorithm with different weights:

```
Algorithm 3 Three-Route Generation
Require: Graph $G$, start node $s$, end node $t$
Ensure: Routes $R_{shortest}, R_{safest}, R_{balanced}$
 1: Find nearest nodes: $s \leftarrow nearest(start\_lat, start\_lon)$
 2: $t \leftarrow nearest(end\_lat, end\_lon)$
 3: $R_{shortest} \leftarrow Dijkstra(G, s, t, w_{length})$
 4: $R_{safest} \leftarrow Dijkstra(G, s, t, w_{safe})$
 5: $R_{balanced} \leftarrow Dijkstra(G, s, t, w_{balanced})$
 6: for each route $R$ do
 7:     Compute distance: $D_R = \sum_{e \in R} length_e$
 8:     Compute avg risk: $\bar{r}_R = \frac{\sum_{e \in R} risk_e}{|R|}$
 9:     Compute safety score: $S_R = 100 - \bar{r}_R$
10: end for
11: return Routes with metrics
```

F. Performance Optimization

1) Graph Caching: Graph caching is used to reduce the initialization time of the system. During the first run, the OpenStreetMap (OSM) graph is downloaded and processed, which may take a few minutes. This generated graph is then saved locally as a cache file. In subsequent executions, the system loads the graph directly from the cached file, reducing the loading time to just a few seconds.

2) Model Caching: Model caching is applied to avoid retraining the machine learning model each time the system runs. After the model and its scaler are trained, they are stored in local cache files. Future runs simply load these cached components, enabling instant model initialization and faster prediction without repeating the training process.

IV. Implementation

A. Technology Stack

Backend Technologies:
Python Flask(for UI) scikit-learn(Random Forest)
OSMnx NetworkX (Map Algo) Scipy (KD-Tree)

Frontend Technologies:
HTML Flask Leaflet.Js

B. Frontend Interface

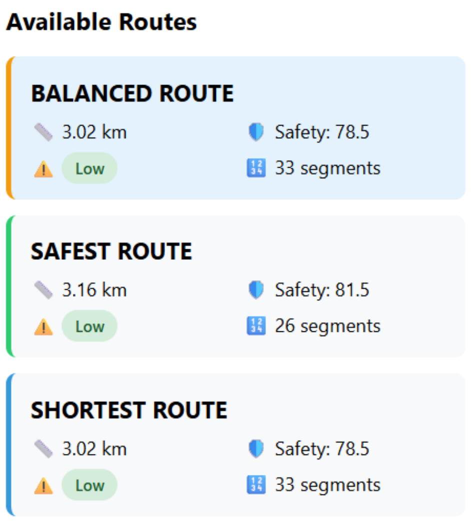
UI contains interactive map, coordinate inputs, route comparison, associated crimes and distance metrices.
Project Flow:
User enters coordinates → Backend receives request → Route optimizer finds three paths → Crime risk assessed for each segment → Results returned with visualization data → Frontend renders interactive analysis

V. Results and Evaluation

A. System Configuration

Coverage Area:

- Delhi: 28.50-28.75°N, 77.05-77.35°E
- Area: Approximately 400 km²
- Road segments: 1,37,958
- Intersections: 3,76,041


Available Routes

BALANCED ROUTE
3.02 km   Safety: 78.5
Low   33 segments

SAFEST ROUTE
3.16 km   Safety: 81.5
Low   26 segments

SHORTEST ROUTE
3.02 km   Safety: 78.5
Low   33 segments

Fig. 1. Output Logs

B. Model Performance Metrics

The trained Random Forest model demonstrates exceptional performance on the test set:

TABLE III
Crime Risk Prediction Model Performance

Metric	Test Set
R² Score	0.9922
MAE	0.52
RMSE	1.66
MAPE (%)	6.4%
Max Error	8.76

The R² score of 0.9922 indicates exceptional model accuracy, explaining 99.22% of variance in crime risk. The test set MAE of 0.52 represents minimal average prediction error, demonstrating robust generalization capability. The maximum error of 8.76 is well within acceptable bounds for practical applications.

C. Feature Importance Analysis

TABLE IV
Top 10 Most Important Features

Feature	Importance
Crime Density	0.9723
Total Area	0.0080
Assault Murders	0.0059
Murder	0.0057
Rape	0.0026
Theft	0.0014
Sexual Harassment	0.0012
Crime Severity	0.0012
Robbery	0.0009
Gang Rape	0.0006

Crime density dominates feature importance at 97.23%, validating our approach to normalize crime counts by geographic area. This indicates that crimes per unit area is the strongest predictor of overall risk.

D. Sample Route Comparison

Test case: Connaught Place to India Gate (3.2 km)
Analysis:

TABLE V
ROUTE COMPARISON EXAMPLE

Metric	Shortest	Balanced	Safest
Distance (km)	3.02	3.02	3.16
Risk	21.5	21.5	18.5
Risk	Low	Low	Low
Safety Score	78.5	78.5	81.5

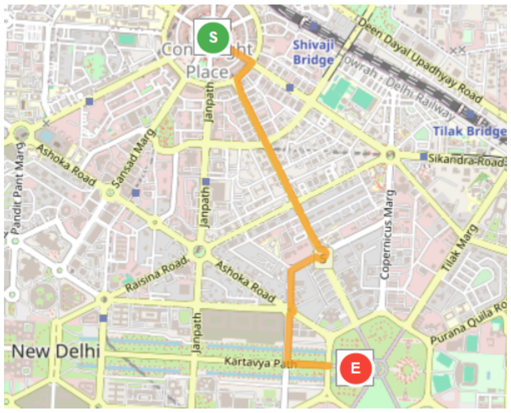


Fig. 2. Balanced Path in Map

VI. Discussion

A. Achievements

- Real time end to end route computation
- Web-based interface for easy access
- Caching reduces initialization by 96%
- Spatial interpolation fills data gaps using KD-tree enables fast queries

B. Limitations

- Small dataset
- Less coverage only to Delhi
- No time or weather consideration
- Coordinate entry only (no address search)

C. Future Enhancements

- Add address geocoding and increase area coverage
- Integration with existing navigation apps
- Real-time crime data integration

VII. Conclusion

This paper presented a complete crime-aware routing system for Delhi that successfully integrates machine learning with geographic information systems. Our key contributions include:

1) Random Forest-based crime risk prediction with 0.99 R^2 score
2) Efficient spatial interpolation using KD-trees
3) Multi-objective routing providing three distinct alternatives
4) Real-time performance through intelligent caching
5) Complete open-source implementation
6) Route Crime Analysis

The system demonstrates practical applicability for safer urban navigation.

Acknowledgments

We acknowledge the Delhi Police for crime data and the OpenStreetMap community for street network data.

References

[1] M. Galbrun, K. Puolamäki, and T. Tatti, "Safe Route Recommendation for Pedestrians Using Street Crime Data," in *Proc. ACM SIGSPATIAL*, 2016.

[2] A. Saha, S. Seal, and A. Dey, "Safe Navigation via Multi-Constrained Route Optimization," in *Proc. ACM SIGSPATIAL*, 2020.

[3] G. Boeing, "OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126-139, 2017.

[4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[5] National Crime Records Bureau, "Crime in India 2022," Ministry of Home Affairs, Government of India, 2022.

[6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[7] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269-271, 1959.

[8] D. Shepard, "A Two-dimensional Interpolation Function for Irregularly-spaced Data," in *Proc. 23rd ACM National Conference*, 1968, pp. 517-524.

[9] A. Jain, H. Kumar, D. Parashar, and S. Sharma, Geographical Information System Based Safe Path Recommender, International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 10, pp. 3863–3868, August 2019.

[10] S. Garg, A. Mittal, and V. Sathiyasuntharam, "Deep Learning Based Model to Recommend Safe Route Navigation System," in IEEE Conference Proceedings, IEEE, 2020.

Author Biography

Vaibhav Singh is a 3rd year student in the Department of Computer Science at Shiv Nadar University, specializing in Data Science and Machine Learning. This work was completed as part of 'Foundations of Data Science' Course under the supervision of Prof. Suchi Kumari.