



"NAVISCAPE: An Intelligent Navigation and Risk Analysis"

KHADIJATUL KUBRA

Department of AI-DS
VTU University
Belagavi, India

Khadijatul.kubraa28@gmail.com

RAHUL YADAV

Department of AI-DS
VTU University
Belagavi, India

ry5820800@gmail.com

SANTHOSH M

Department of AI-DS
VTU University
Belagavi, India

Santhosh.muruga04@gmail.com

Ms. MOUNASHREE

Asst. Professor, Department of AI-DS VTU University, Belagavi, India

hsmounashree2711@gmail.com

Abstract—

How to Cite this Article:

KUBRA, K., YADAV, R. & M, S. (2026). NAVISCAPE: An Intelligent Navigation and Risk Analysis. International Journal of Creative and Open Research in Engineering and Management, <i>02</i>(10), 1-9. <https://doi.org/10.55041/ijcope.v2i10.076>

License:

This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.



<https://doi.org/10.55041/ijcope.v2i10.076>

Navigation apps usually minimize travel time or distance, which can send drivers through accident-prone or congested stretches. This paper presents NAVISCAPE, a navigation platform that ranks candidate routes using accident-severity prediction, traffic forecasting, live traffic data, and nearby emergency resources. An eXtreme Gradient Boosting (XGBoost) classifier was trained on 95,723 road-accident records from Karnataka, a Long Short-Term Memory (LSTM) network forecasts hourly vehicle counts, and live segment speeds come from the TomTom Flow Segment API. Routes generated through OpenStreetMap/OSRM are scored on safety (40%), traffic (30%), estimated travel time (20%), and distance (10%).

The platform also indexes 2,226 hospitals and 921 police stations, accepts crowdsourced hazard reports, and prepares authenticated SOS alerts for up to four trusted contacts. On a stratified test set of 18,790 records, XGBoost reached 52.47% accuracy, with a recall of 0.88 for grievous-injury cases. In one route comparison, nearby historical accidents fell from 37 to 8 and estimated travel time from 42.0 to 33.5 minutes. The results are encouraging, but they rest on a regional dataset and a single case study, so broader validation is still needed.

Keywords: Intelligent Transportation Systems, Risk-Aware Routing, Traffic Prediction, Long Short-Term Memory (LSTM), XGBoost, Route Optimization, Emergency Response.



I. INTRODUCTION

Growing vehicle density puts heavy pressure on urban roads, and most navigation apps respond by minimizing distance or travel time with shortest-path search. That finds fast routes, but a fast route can still pass through a cluster of severe crashes or a corridor that jams every evening. Three gaps motivate this work.

1. Safety-agnostic routing: Routing rarely penalizes segments with a history of collisions, so drivers can end up on narrow, poorly lit, or high-collision roads just to save a few minutes.
2. Reactive traffic guidance: Guidance tends to respond once a bottleneck has formed instead of using known daily patterns.
3. Weak emergency integration: The active route is seldom linked to nearby hospitals or police stations, or to an in-app way to ask for help, which matters most for vulnerable road users such as women commuting at off-peak hours.

NAVISCAPE addresses these gaps in one system that couples machine learning with geospatial analysis, with the dual aim of efficient travel and personal safety. Its contributions are an XGBoost accident-severity classifier trained on 95,723 Karnataka collision records; an LSTM traffic forecaster with a 24-hour lookback window, blended with live TomTom speeds; a route-ranking scheme that weighs historical safety, traffic, travel time, and distance; and an emergency module covering 2,226 hospitals, 921 police stations, crowdsourced hazards, JWT-based authentication, and SOS alerts for up to four trusted contacts.

II. LITERATURE REVIEW

Road-network routing has long relied on Dijkstra's algorithm and A* search [1]. Preprocessing methods such as Contraction Hierarchies and Customizable Contraction Hierarchies [2] make queries on large networks nearly instantaneous. These methods optimize a scalar edge cost, usually distance or travel time. Multi-objective planners balance other criteria such as tolls, electric-vehicle energy use, and fuel economy, but empirical accident risk is still rarely part of everyday route recommendations [6]. NAVISCAPE adds

historical risk and live traffic to the cost of a route alongside time and distance.

Traffic-flow prediction has moved from statistical models such as ARIMA and vector autoregression to recurrent networks that learn non-linear daily cycles [3]. LSTM networks were designed to capture longer-term dependencies in sequences [4]. Recent work uses hybrid spatial-temporal models, but inside a low-latency REST service the model has to be lightweight enough for sub-second inference alongside live API calls.

Accident severity has been predicted with decision trees, Random Forests, support vector machines, and gradient-boosted trees. Road geometry, lighting, surface dampness, and spatial clustering are the factors most often linked to severity. XGBoost suits tabular crash registries because it handles sparse features, uses a regularized objective, and builds trees efficiently [5]. Accident records are usually imbalanced and affected by reporting bias, so models should be judged on per-class precision and recall, not accuracy alone.

III. METHODOLOGY

A. System Architecture and Block Diagram

NAVISCAPE uses a decoupled, multi-tier, service-oriented design (Fig. 1) that separates the client, application services, machine-learning models, and storage, so routing, traffic analysis, risk scoring, hazard handling, authentication, and emergency actions can be developed as separate backend services. The client is a React 18 and Vite single-page application with Leaflet maps on OpenStreetMap tiles, styled with Tailwind CSS and Recharts. A FastAPI backend written in Python exposes endpoints for route generation, accident-risk scoring, traffic forecasting, hazard ingestion, and emergency handling, and it loads the trained XGBoost classifier and a TensorFlow/Keras [8] LSTM model. The client talks to it over HTTPS/REST with JWT authentication. Accident records, emergency facilities, hazards, traffic data, user profiles, and emergency events are stored in SQLite in Write-Ahead Logging (WAL) mode through SQLAlchemy. External services are OSRM/OpenStreetMap for routing and the TomTom Flow Segment API [10] for live speeds.

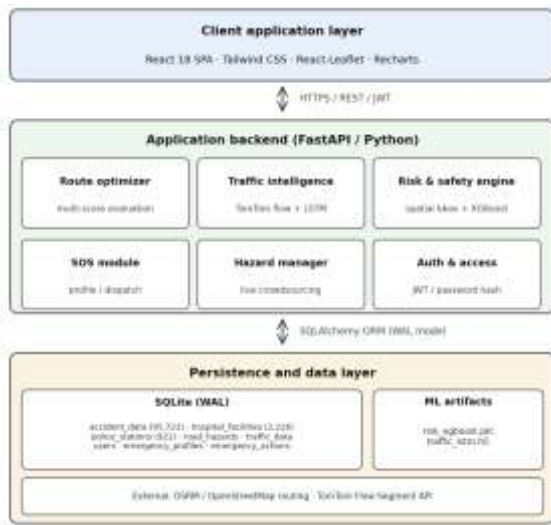


Fig. 1: High-level architecture of the NAVISCAPE platform.

B. Accident Risk Modeling Using XGBoost

The dataset holds 95,723 raw collision records from regional traffic registries in Karnataka [9]. After dropping records without coordinates, road conditions, or severity labels, 93,948 remained. The records span 38 districts but unevenly, with Bengaluru, Tumakuru, and Belagavi among those with the most incidents (Fig. 2), so the data do not represent all Karnataka roads equally. Severity has four classes: Damage Only (0), Simple Injury (1), Grievous Injury (2), and Fatal (3), making up roughly 7.8%, 19.5%, 47.4%, and 25.3% of the records.

Each incident i is described by its latitude, longitude, and label-encoded weather, road condition, and surface condition:

$$x_i = [\text{latitude}_i, \text{longitude}_i, \text{weather}_i, \text{road-condition}_i, \text{surface-condition}_i]^T$$

XGBoost minimizes a regularized multi-class objective, cross-entropy loss plus a penalty on tree complexity:

$$L(\theta) = -\sum_i \sum_k y_{ik} \log \hat{p}_{ik}(x_i) + \sum_m \Omega(f_m), \quad \Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2$$

Hyperparameters were 150 estimators, maximum depth 6, learning rate 0.08, and multi-class log loss for evaluation. We split the data 80/20, giving 75,158 training samples and 18,790 stratified test samples. At scoring time, the predicted class probabilities are combined with the severity weights defined in Section III-E to form a continuous risk surface that penalizes candidate routes.

C. Traffic Congestion Prediction Using LSTM

Hourly vehicle counts follow non-linear daily cycles, so the LSTM forecasts counts at target junctions along primary Bangalore corridors from the previous 24 hours. If v_t is the count at hour t , the model maps the last 24 observations to the next count:

$$[v_{t-24}, \dots, v_{t-2}, v_{t-1}] \rightarrow \hat{v}_t, \quad v' = (v - v_{\min}) / (v_{\max} - v_{\min})$$

Inputs are scaled to $[0, 1]$ with min-max normalization. The network has a 64-unit LSTM layer that returns sequences, 20% dropout, a 32-unit LSTM layer, a second 20% dropout layer, a 16-unit ReLU dense layer, and a linear output unit. It was trained for 20 epochs with the Adam optimizer [7] and mean squared error (MSE) loss, using a sequential 80/20 split.

D. Live and Predicted Traffic Intelligence

Dense route polylines from the routing engine are thinned to probe waypoints about every 0.6 km using the Haversine distance, and each probe queries the TomTom Flow Segment API for current and free-flow speed. Near the eight monitored arterial junctions (including Silk Board, Hebbal, KR Puram, Marathahalli, Whitefield, Banashankari, and Jayanagar), the LSTM and hour-of-day prediction is blended with the live reading:

$$S_{\text{traffic}} = 0.60 \cdot S_{\text{current}} + 0.40 \cdot S_{\text{predicted}}$$

Here S_{current} is the speed-ratio score defined in Section III-E. If no predicted junction data lie within 1.5 km, the system falls back to live segment flow alone.

E. Multi-Criteria Route Optimization

For a set of candidate routes $R = \{R_1, \dots, R_k\}$, NAVISCAPE computes normalized safety, traffic, travel-time, and distance scores and recommends the route with the highest weighted total.

Safety score: Historical accident records within 0.3 km of route waypoints are retrieved and weighted according to severity: 1.00 for fatal accidents, 0.75 for grievous injuries, 0.50 for simple injuries, and 0.25 for damage-only cases. The weighted accident penalty is subtracted from a baseline score of 100, with the minimum safety score limited to 10.



Traffic, ETA, and distance scores. The speed-ratio score $S_{current}$ compares current to free-flow speed and is capped at 100 ($S_{traffic}$ equals $S_{current}$ wherever no forecast is blended in). ETA and distance scores compare each route with the best option in the candidate set and are bounded between 10 and 100. ETA is the base OSRM duration plus traffic and hazard delays.

$$S_{current} = \min(100, 100 \times v_{current} / v_{free-flow})$$

$$S_{ETA}(R_i) = \text{clamp}[10, 100] (\min_j T_j / T_i \times 100)$$

$$S_{distance}(R_i) = \text{clamp}[10, 100] (\min_j D_j / D_i \times 100)$$

Overall score. The overall score is:

$$S_{overall} = 0.40 S_{safety} + 0.30 S_{traffic} + 0.20 S_{ETA} + 0.10 S_{distance}$$

The top-scoring route is recommended with a plain-language reason, such as lower historical accident exposure or avoidance of a congestion hotspot. All four component scores are oriented so that higher is better, which keeps the weighted sum consistent.

F. Emergency and Personal Safety Orchestration

In safety-critical situations a navigation tool should connect to emergency services. The emergency module indexes 2,226 hospitals (with emergency trauma classification) and 921 state police stations by coordinates, so the nearest medical and police facilities along a user's route can be found with spatial queries. The hazard module takes crowdsourced reports of potholes, waterlogging, poor street lighting, construction obstacles, and similar problems, supports bounding-box queries over them, and adds their delay penalties to the route ETA.

Authenticated users register an emergency mobile number, an optional email, and two to four trusted contacts with relationship labels, and must consent to location sharing before the safety profile counts as complete. To avoid accidental triggers, the SOS button has to be held for three seconds and then confirmed. On confirmation the system captures the device GPS position (about ± 60 m in Fig. 3), stores an immutable Emergency Event, and keeps an auditable action trail with states such as EMERGENCY_REGISTERED, WHATSAPP_PREPARED, and POLICE_CALLED. The active-emergency screen then offers one-tap actions: call a contact, call 112, view the location, or open WhatsApp with a prepared, localized message that includes a live map link. Delivery is never assumed. The user presses Send inside WhatsApp and taps "I sent the alert" to confirm, and the system reports delivery only after that confirmation.

G. Prototype Interface

New users register with a name, email, and a password of at least six characters, followed by a verification step. The route planner takes a source and destination, searches for routes, and shows the nearest police station and hospital, with map layers for hospitals, police stations, and traffic. The dashboard (Fig. 4) summarizes each user's journeys: routes completed, average empirical safety score, total distance and travel time, the safest and lowest-scoring trips, and short written insights. Scores of 80 or above are labeled optimal, 60–79 moderate, and below 60 caution. The dashboard also shows live hazard counts, accident statistics for the Karnataka dataset, and a map of 3,000 incident.

IV. RESULTS AND DISCUSSION

A. Accident Severity Classification

Table 1 gives class-level results on the 18,790 held-out test records.

Severity class	Precision	Recall	F1-score	Support
Damage Only	0.13	0.00	0.00	1,465
Simple Injury	0.51	0.31	0.39	3,667
Grievous Injury	0.53	0.88	0.66	8,903
Fatal	0.51	0.18	0.27	4,755
Macro average	0.42	0.34	0.33	18,790
Weighted average	0.49	0.52	0.46	18,790
Overall accuracy	—	—	0.5247	18,790

Table 1: XGBoost accident-severity classification.



Overall accuracy was 52.47%. Recall for grievous injury was 0.88, so the model catches most cases of the class that matters most for casualty prevention. Recall was only 0.00 for damage-only and 0.18 for fatal cases, so the model is not reliable across all severity levels. The near-zero damage-only recall probably reflects reporting bias, since non-injury damage incidents are under-reported in police registries compared with injury and fatal events. The high grievous-injury recall is useful as one signal in the route-risk surface, but calibration, better handling of class imbalance, and external validation are needed first.

B. Traffic Forecasting

The LSTM model was trained for 20 epochs using an 80/20 sequential split of historical traffic readings. Peak-hour patterns were considered when analyzing congestion. However, the model's forecasting performance has not yet been quantified using test-set MSE, MAE, or RMSE, and further evaluation is required.

Pairing the 24-hour lookback window with live TomTom flow lets the system catch sudden bottlenecks while still expecting the usual daily congestion. We have not yet computed a test MSE, MAE, RMSE, or baseline comparison, so the forecasting accuracy of the LSTM is not quantified here.

C. Route-Comparison Evaluation

Candidate routes between urban hubs were scored under varying road conditions. Table 2 compares a conventional shortest-distance route with the route NAVISCAPE recommends.

Metric	Conventional	NAVISCAPE
Distance	14.2 km	15.6 km (+9.8%)
Base OSRM duration	28.0 min	30.5 min
Traffic/hazard delay	+14.0 min	+3.0 min
Reported total ETA	42.0 min	33.5 min (-20.2%)
Nearby historical accidents	37	8 (-78.4%)
High-risk hotspots	3	0
Empirical safety score	54.2/100	88.6/100
Traffic-flow score	48.0/100	82.5/100
Overall score	52.8/100	84.1/100

Table 2: Reported comparison of route options.

In this example the conventional route is shorter but passes through three known accident clusters and heavy bottlenecks. NAVISCAPE picks a route 1.4 km longer, and nearby historical accidents drop from 37 to 8, estimated ETA falls from 42.0 to 33.5 minutes (about 20.2%), and the overall score rises from 52.8 to 84.1. That is the trade-off the scoring is meant to make. The route comparison is presented as a single illustrative case study. The original origin, destination, and test date were not recorded, and the comparison has not been repeated across multiple route pairs. Therefore, the results should be interpreted as preliminary rather than as evidence of general performance.

D. System Latency

In local tests, the spatial accident bounding-box query averaged 28 ± 6 ms for a 15 km route with 25 sampled points. Parallel asynchronous TomTom polling took 350 ± 85 ms, and end-to-end route scoring stayed under 500 ms. That is fast enough for interactive use in our local setup. Hardware details, network conditions, repetition counts, and percentile latencies are not reported, so these numbers should not be extrapolated to production.

E. Limitations

The accident data cover some districts far better than others, and historical crashes say little about lighting, crime, or other aspects of personal safety. Recall of 0.18 for fatal and 0.00 for damage-only cases shows that accuracy alone is not enough for safety-critical use. The traffic forecaster has no numerical error metrics or baseline yet, and the route comparison is a single example that others cannot yet replicate. In deployment, API outages, GPS error, and stale hazard reports could also affect behavior, and SOS alerts reach contacts only if the user presses Send in WhatsApp. NAVISCAPE is decision support: it does not guarantee that a route is safe or that help has been dispatched.

V. CONCLUSION

NAVISCAPE balances travel efficiency with road safety and personal security by combining an XGBoost severity classifier trained on 95,723 crash records, an LSTM traffic forecaster, live TomTom speeds, and a multi-criteria route score, together with an emergency



module covering 2,226 hospitals, 921 police stations, crowdsourced hazards, and SOS alerts for up to four trusted contacts. XGBoost reached 52.47% accuracy and 0.88 recall for grievous injury. In the route comparison, a 1.4 km longer route cut nearby historical accidents from 37 to 8 and estimated travel time from 42.0 to 33.5 minutes. These results suggest that routing on shortest path or duration alone ignores safety information that is already available, but stronger validation and better per-class performance are needed.

Future work includes evaluating spatio-temporal graph neural networks such as STGCN, adding real-time weather data, running field trials with vehicle or edge-IoT telemetry, reporting forecasting MAE/RMSE against persistence and statistical baselines, checking probability calibration, and repeating the route comparison on geographically separate regions with published origin–destination pairs, latency percentiles, and uncertainty estimates.

REFERENCES

- [1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [2] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Experimental Algorithms*, Berlin, Heidelberg: Springer, 2008, pp. 319–333.
- [3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.
- [6] D. R. Carmody and R. B. Sowers, "Tradeoffs between safety and time: A routing view,"

Transportation Research Part C: Emerging Technologies, vol. 108, pp. 357–377, 2019.

- [7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, 2015.
- [8] F. Chollet et al., "Keras: Deep learning for humans," GitHub repository, 2015. [Online]. Available: <https://github.com/keras-team/keras>
- [9] S. Shubham, "Karnataka Accident Dataset," Kaggle Datasets, 2023. [Online]. Available: <https://www.kaggle.com/datasets/shubham2703/karnat-aka-accident-dataset>
- [10] TomTom N.V., "TomTom Traffic Stats & Flow Segment Data API Documentation," 2024. [Online]. Available: <https://developer.tomtom.com/traffic-api>