Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

PRAHARI: Advanced Predictive Safety Platform

Mr. Abhishek B M
Department of Computer Science and Engineering
Acharya Institute of Technology
Bengaluru, India

Priya, Aditya Kumar, Ashwin Kumar
Department of Computer Science and Engineering
Acharya Institute of Technology
Bengaluru, India

***Abstract*—Navigation apps often provide routes based on distance and not much else. Many factors can impact how safe a route is for a female walking at night or a student unfamiliar with an area in the city. PRAHARI is a web application that takes this into account. We ask two questions: How safe are the road segments on each route? How much extra distance would we have to travel to avoid the less safe segments? Using the OpenStreetMap database, we create a directed graph containing all roads. Each road segment has 7 different attributes mapped including the class of road, length of road, number of lanes, one-way, illumination tag, distance to nearest police station, and distance to nearest nightlife spot. An XGBoost regressor was fitted to give each segment a score between 1-10. Dijkstra's algorithm provides us with a shortest, safest and balanced route. The navigation service was built using FastAPI with a backend containing node.js/express and a frontend using react/leaflet. This allows us to also track the journey, report incidents and send SOS requests. For the central section of Bengaluru containing 160,077 segments, 32 s was taken to build and score the area. Each segment had a prediction. However, a problem with the original design was found while going from our campus to Majestic. The 'safest' route was 151% longer but only reduced the risk index by 1. Since we were using a cost per segment, routes consisting of fewer, longer segments were favoured. We explore why this is happening. We also looked at an alternative of normalising the distance, and tested this offline using the same data.**

***Index Terms*—Routing, OpenStreetMap, XGBoost, Dijkstra's algorithm, Women Safety, Geospatial Machine Learning, Web Application**

I. Introduction

Route planning tools have become extremely efficient at finding the fastest possible route between two locations. While the aim is simple to define and quantify, it is also the correct goal for most trips. However, many women, students and night shift workers in Indian cities will plan their route home based on other factors such as if the street is lit, is a busy main road or empty, and if there is somewhere nearby to seek assistance. Official statistics show that 441,534 cases of crimes against women were reported to the National Crime Records Bureau in 2024 [1]. While many of these crimes are unrelated to the street, knowing the overall number helps to understand why many women still consider safety when choosing their route [2].

Other safe route planning systems have used police incident data, geotagged social media and crowd sourcing [3]–[6]. However, we do not have open access to similar data on the streets in Bengaluru. To address this, we aim to determine if we can create a route where every road segment receives a consistent and interpretable score based purely on open mapped data. We can then incorporate this into a standard routing query and provide the user with information on how much longer each alternative would take.

PRAHARI is our solution. The system is a functional platform. A Python service accesses the street network from OpenStreetMap (OSM) [7]–[9]. It calculates seven features for each segment. An XGBoost model [10] is trained to map these features to a score. This scored graph is saved for later use. Three options are calculated for each request using Dijkstra's algorithm [11] via NetworkX [12]. A Node.js backend manages user accounts, tracking journeys, incidents and SOS. A React frontend colours each segment according to their score.

We have learnt that designing the edge cost function is as important as the model. Using the initial edge cost, the 'safest' route from our campus to Majestic (17.8 km) was 44.7 km long and did not significantly improve the risk index. This was due to a structural issue and is described in Section VII-D.

The key contributions of this paper are:

- An end to end system that creates a scored cached graph of roads within an area extracted from OSM using seven mapped features and the XGBoost regressor (Algorithm 1);
- A deployed three tier system providing the shortest, safest and balanced routes, journey tracking, incident reporting and SOS (Fig. 2, Algorithm 2);
- Evaluation of our road graphs in Bengaluru by measuring the graph size, training data, creation time, score range and full case study;
- Analysis of why the safety costs for each segment create long detours. A normalised length function has been created using A search and a maximum detour (Algorithm 3). This was tested offline on the same graph.

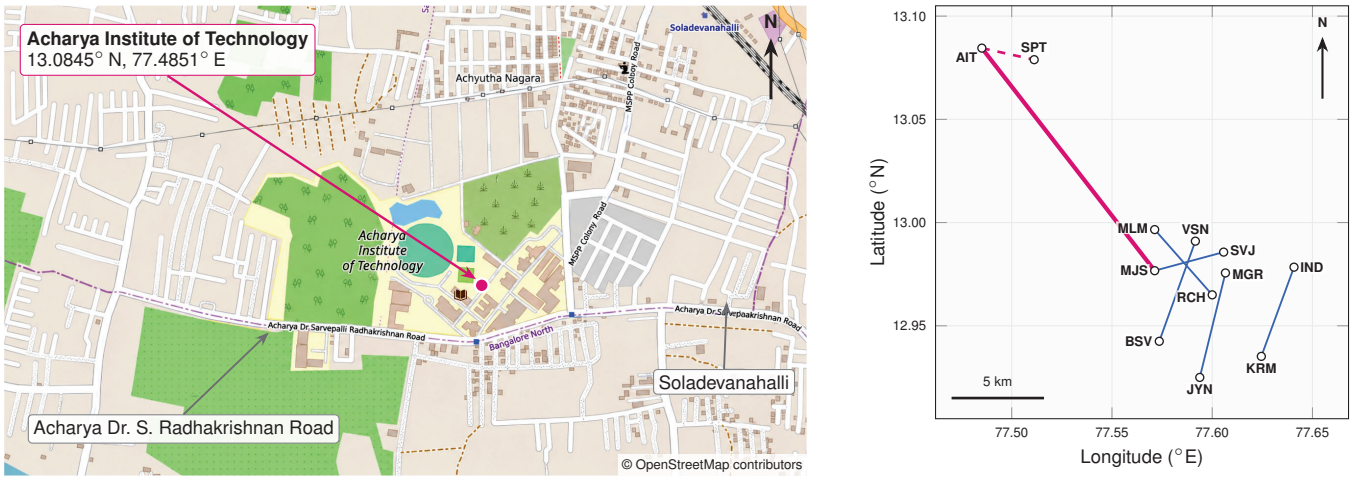
II. Related Work

A. Safety-Aware Route Search

Galbrun *et al.* [3] developed a risk model for crimes in Chicago and Philadelphia street networks. Safe routing was posed as a multi-objective shortest path problem which resulted in a small set of routes balancing distance and risk. SocRoutes [4] used sentiment in geotagged tweets as signals and found that routes avoiding strongly negative areas also avoided crime hotspots. CROWDSAFE [5] used crowdsourced

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026



(a) (b)

Fig. 1. Study area. (a) Neighbourhood of Acharya Institute of Technology in Soladevanahalli, on the north-western edge of Bengaluru. (b) Origins and destinations of the trips in Section VII; thick line: campus to Majestic (case study), dashed: campus to Sapthagiri Hospital, thin lines: central trips. AIT = campus, MJS = Majestic, SPT = Sapthagiri Hospital, MLM = Malleshwaram, RCH = Richmond Town, MGR = MG Road, JYN = Jayanagar, KRM = Koramangala, IND = Indiranagar, BSV = Basavanagudi, VSN = Vasanth Nagar, SVJ = Shivajinagar.

and official incidents to route safely on mobile devices. Mata *et al.* [6] used tweets and official crime data for Mexico City. Aljubayrin *et al.* [13] took a geometric approach favouring routes where users remain inside specific safe zones. This could be adapted to optimise for other attributes. Quercia *et al.* [14] suggested routes perceived as beautiful, peaceful and happy but with an extra few minutes walking time. Most reviews look at risk for traffic related crashes (Sohrabi *et al.* [15], Sohrabi and Lord [16]). However, one conclusion can be applied here: The usefulness of a safety aware router depends upon the accuracy of the risk estimation and its inclusion in the edge costs.

B. Women's Safety and the Street Environment

In India, SafetiPin [2] developed a mobile application to allow women to conduct safety audits which rank public spaces based on factors like lighting, security and gender diversity. In several cities they have gathered safety audit data. This data gives an insight into how safe people feel when walking around these areas. Lighting tends to be included in almost all of these audits. An analysis of 18 systematic reviews found that better lighting on streets can help significantly reduce crime [17] (Welsh and Farrington). Jacobs [18] argues that streets which are being monitored by local people and those passing by are safer. This means busy places may not always be safe, we discuss this further in Section VIII. PRAHARI does not collect safety audits but only collects information from the section of these cues that is recorded in OSM.

C. Street Networks and Learning on Map Data

OSM has been crowdsourced [7]; Barrington-Leigh and Millard-Ball estimate it contains over 80% of global roads fully mapped [19]. Attributes like lighting are however much less complete than geometries. OSMnx [9] can fetch street networks from OSM as a graph. NetworkX [12] will compute shortest paths. Gradient boosting trees [20], and specifically XGBoost [10] are a good option for tabular data. Shortest paths can be found with Dijkstra's algorithm [11], or A [21] using an admissible heuristic. To combine distance and risk as a single value we can take a weighted sum. How this weighting takes place depends on the scaling of each term [22] as shown by Marler and Arora. This is the key issue in section VII-D.

Unlike [3]–[6], the PRAHARI approach does not provide learning insights based on incident data. It provides an insight into how open data can be used in a useful application. Additionally, it provides an insight into how the edge-cost definition changes the route that users see.

III. Problem Formulation

We model an area as a directed multigraph G=(V,E). Nodes are intersections and other OSM vertices and edges are the traversable segments. Parallel edges are allowed. Each edge e has its length d_e, $a feature vector x ein R^7 and a predicted score hat t_e in [1, 10] (the high 11 - \hat{s}_e, \quad c_e \in [1, 10]$.(1) For a path P from origin o to destination z with edge set $E(P)$, the deployed system solves

$$P_{\text{short}} = \arg\min_{P} \sum_{e \in E(P)} d_e, \tag{2}$$

$$P_{\text{safe}} = \arg\min_{P} \sum_{e \in E(P)} c_e, \tag{3}$$

$$P_{\text{bal}} = \arg\min_{P} \sum_{e \in E(P)} \left[\alpha \frac{c_e}{10} + (1 - \alpha) \frac{d_e}{d_{\max}} \right], \tag{4}$$

With alpha equal to 0.6 and $d_m axbeing the longest edge in the area. If parallel edges join the same t w$ with mode speed v_m (22 km/h for driving, 4.5 km/h for

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

walking), its number of segments $n(P) = |E(P)|$ and its mean score

$$\bar{s}(P) = \frac{1}{n(P)} \sum_{e \in E(P)} \hat{s}_e. \tag{5}$$

The interface turns $\bar{s}$ into a risk index on a 0–100 scale,

$$R(P) = \text{round}\Big(100\,\frac{10 - \bar{s}(P)}{9}\Big), \tag{6}$$

labelled as lower risk (R ¡ 33), moderate risk (34- 66) or higher risk (R¿ 66). As T is proportional to L the shortest path is also the one the interface refers to as' Fastest'.

The evaluation poses three questions. RQ1: what is the cost of building and scoring an area and how is the predicted score distributed across the city scale graph? RQ2: for a real trip, how do the three routes trade off length and time against the risk index? RQ3: does the cost as defined in equations (3)–(4) behave in the way a user would expect and if not, what fixes it?

IV. System Architecture

Fig. 2 shows the implemented system with three tiers and some external services. Each tier is independently developed, tested and restarted.

Client tier. The client was developed using React and Vite with Leaflet through react-leaflet to provide mapping functionality. The Navigate page allows the user to input their starting point, destination (with search suggestions) and mode of travel. This provides one card for each route. Once a route is selected, the route is displayed on the map with a red-amber-green colouring depending on $\hat{s}_e$. This colour is also used on a thin ribbon at the top of the card to highlight areas of low stretch. Other pages include the dashboard, incident and community reports, emergency contacts, SOS, profile and live journeys. The sign-in functionality was provided by Clerk and the JSON Web Token is included in every API call.

Application tier. A Node.js / Express server provides a REST API. Each request checks the Clerk token. Nominatim is used to search for places and the route requests are passed to the navigation service with an internal service token. Users, saved routes, journeys, alerts, incident reports and emergency contacts are stored in a serverless PostgreSQL database (Neon) using Prisma. A Socket.IO server is used to provide real-time updates to the user's room with any new journeys or alerts. Community reports have a moderation status which can only be edited by moderators. An SOS request is stored with the user's last known location and passed to a notification service to send out emails and SMS or WhatsApp messages through Twilio.

Intelligence tier. The navigation service was developed using Python and FastAPI. The area register determines which cached graph to use for the request. If there is no cached graph, then Algorithm 1 is run in the background to build the area and the client is told that it is building. This means the first request will never cause the server to be blocked. Once the area has been completed, it is stored as a pickled graph along with a metadata file. This means it will need to be rebuilt after seven days meaning any edits to the OSM will eventually be reflected in the scores. The route service uses Algorithm 2. There is also a heatmap endpoint which provides scores for each segment in the visible area. The explanation module provides a brief description of the features included in each segment of the route (e.g. how many segments have a lighting tag). This always includes a note that the score is an estimate provided by a model and is not verified crime data. OSM data is provided using OSMnx and the Overpass API. Geocoding is provided by Nominatim.

V. Study Area and Data

A. Study Area

Acharya Institute of Technology is located in Soladevanahalli on the north-western outskirts of Bengaluru (13.0845° N, 77.4851° E) (Fig. 1a). The campus is located along an arterial road with residential layouts on either side. To the south-west are open land, and to the east there are residential lanes in a grid pattern. The campus has a wide variety of roads when travelling to the city centre ranging from village roads, through to Tumakuru Road and the dense grid of roads around Majestic, making it an ideal place to trial risk aware routing.

Two extents were created and stored in this study (Fig. 1b). The *core* extent covers central Bengaluru and contains 60,575 nodes and 160,077 directed segments. This extent was used throughout the model and for calculating routing statistics. The *automatic* extent was selected by the area registry for campus based trips and covers the north-western suburbs and city centre. This has 592,128 segments.

B. Data Sources

The road geometry and tags were obtained from OSM [8] using OSMnx [9] with network type set to `all`. This ensures that footways, steps and service roads are included in the graph. The police stations and nightlife amenities were obtained from OSM using the Overpass API within a radius of 1.2 times the half-width of the area (min. 800 m). The data was downloaded between August and October 2026. The main extent shown in Table II was created on 19 August 2026.

C. Segment Attributes

The seven variables in vector $\mathbf{x}_e$ are listed in Table I. The 'road class' and 'lighting' variables are categorical and encoded within the model pipeline. The distance variables denote the shortest distance from a segment to a point of interest of that type. Many OSM segments do not have a value for the `lit` and `lanes` variables meaning these columns describe the state of the map rather than the street. If a segment does not have the `lit` tag this should not be assumed to mean it is dark.

D. Training Target

As there was no data on the detailed nature of any incidents, we defined our training target as a heuristic score. We began with a neutral score, and added or subtracted fixed values depending on the road type (residential roads add 1.5 to the

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

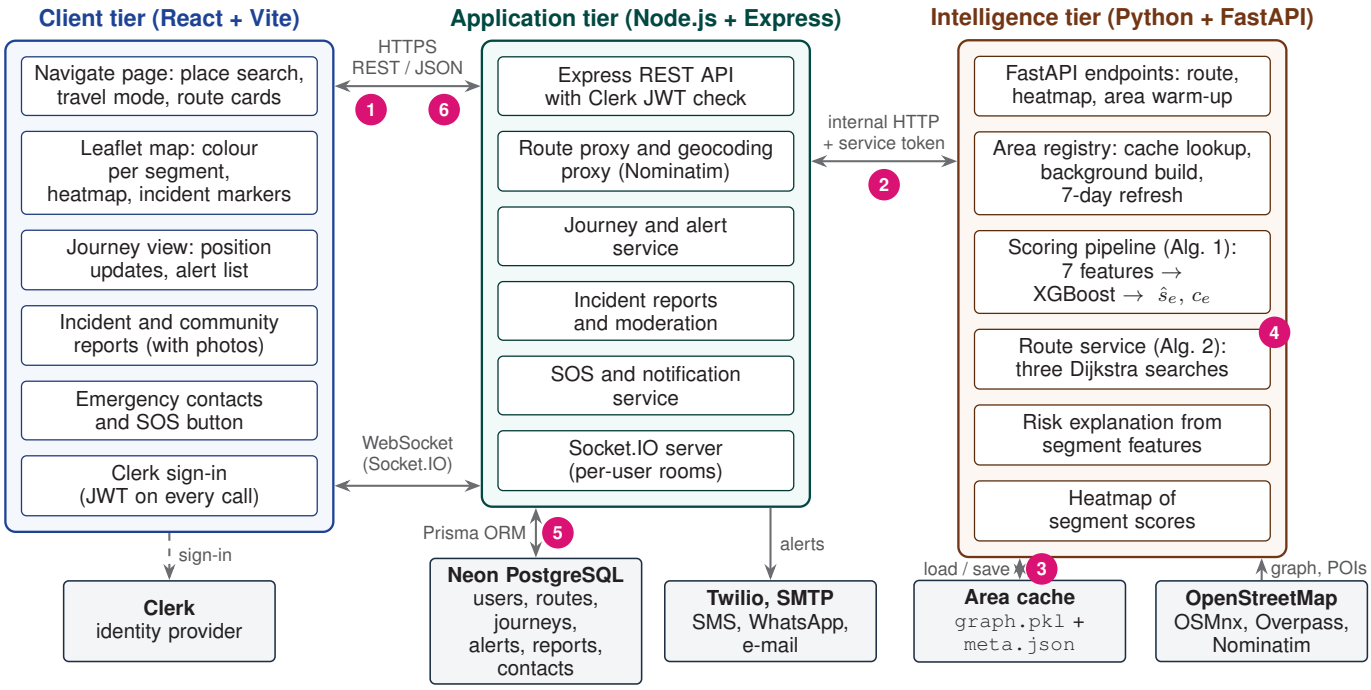


Fig. 2. Deployed architecture of PRAHARI. The numbered markers follow one route request: (1) the client sends the trip; (2) the backend checks the sign-in token and calls the navigation service; (3) the service loads the scored area from the cache or builds it using Algorithm 1; (4) Algorithm 2 computes the three routes; (5) the backend stores them for future use; (6) the client draws them in segments.

TABLE I
SEGMENT ATTRIBUTES USED BY THE MODEL

Sym.	Attribute	OSM source	Type
r_e	Road class	`highway`	categorical
ℓ_e	Segment length (m)	geometry	numeric
n_e	Number of lanes	`lanes`	numeric, sparse
o_e	One-way flag	`oneway`	binary
g_e	Lighting tag	`lit`	categorical, sparse
p_e	Distance to police (m)	`amenity=police`	numeric
q_e	Distance to nightlife (m)	nightlife amenities	numeric

score), lighting tag, proximity to a police point and proximity to a nightlife location. We then added a small random noise term and clipped the score to be between $[1, 10]$. A model was trained using XGBoost on this target. This model will learn to smooth out the rules and fill in the gaps. This is what is used to score the network at runtime, however it cannot be any more accurate in the real world than the rules it has learned from. All numbers in section VII should be considered with this in mind.

VI. Methodology and Algorithms

A. Building and Scoring an Area

Algorithm 1 is the offline portion of the pipeline. It is run once per area and the results cached. The graph is downloaded. From here, the bounding box is calculated to determine the radius of the point of interest query. Features and heuristic targets are calculated for each edge. The rows are then split randomly into 80% for training and 20% held out using scikit-learn [23]. The regressor is then fit and applied to *every* segment. This is because the router must have a score for each edge. The predictions are clamped between [1,10] and converted into costs using (1). If there was no row for a given edge, it would default to the neutral score of 5. This didn't happen in any of the builds.

In an earlier version of this step we found out that we needed to be careful about what is actually being passed onto the graph. We lost the index of the edges between the features and scoring steps causing almost every segment to default to the neutral score. The few segments which matched would get the noisy training labels instead of the predictions. This resulted in all routes in the city having a risk index of 56 (as given by (6) for $\bar{s} = 5$). By adding back in the index and attaching the predictions, this was solved.

B. Generating Three Routes

Algorithm 2. The source and target are rounded to the nearest nodes in the graph. Three Dijkstra searches are performed on the cached graph to find the shortest path between the source and target for (2)–(4). Each search with a binary heap takes $O((|V|+|E|)\log|V|)$ [24]. The service returns the total distance L, time T, number of segments n, average segment size $\bar{s}$, and score for each segment R. It also returns the explanation statements described in Section IV.

C. Journeys, Reports and SOS

The journey is initiated by using a stored route. The client continually updates their location and this is stored by the

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Algorithm 1 Building and scoring an area (offline)

Input: area centre $\mathbf{a}$ and radius ρ
Output: scored graph G with $\hat{s}_e$ and c_e on every edge

$G \leftarrow \textsc{OSMnx}(\mathbf{a}, \rho,\ \text{type} = \texttt{all})$
$w \leftarrow$ longer side of the bounding box of G
$\mathcal{A}_{\text{pol}}, \mathcal{A}_{\text{nite}} \leftarrow \textsc{Overpass}\big(\text{radius} = \max(0.6w,\ 800\,\text{m})\big)$
for each edge $e \in E$ **do**
    $\mathbf{x}_e \leftarrow (r_e, \ell_e, n_e, o_e, g_e, p_e, q_e)$ ▷ Table I
    $y_e \leftarrow \text{clip}\big(h(\mathbf{x}_e) + \varepsilon_e,\ 1,\ 10\big)$ ▷ heuristic
end for
split $\{(\mathbf{x}_e, y_e)\}$ at random: 80% train, 20% held out
$f_\theta \leftarrow$ XGBoost regressor fitted on the training rows
for each edge $e \in E$ **do**
    $\hat{s}_e \leftarrow \text{clip}\big(f_\theta(\mathbf{x}_e), 1, 10\big)$, or 5 if $\mathbf{x}_e$ is missing
    $c_e \leftarrow 11 - \hat{s}_e$
end for
write G and its metadata (counts, build time) to the cache
return G

Algorithm 2 Three routes on the scored graph (deployed)

Input: scored graph G, origin o, destination z, mode m
Output: $P_{\text{short}}, P_{\text{safe}}, P_{\text{bal}}$ with their summaries

$o' \leftarrow$ nearest node to o; $z' \leftarrow$ nearest node to z
$d_{\max} \leftarrow \max_{e \in E} d_e$
$w_{\text{short}}(e) \leftarrow d_e$; $w_{\text{safe}}(e) \leftarrow c_e$
$w_{\text{bal}}(e) \leftarrow 0.6\, c_e/10 + 0.4\, d_e/d_{\max}$
for $k \in \{\text{short}, \text{safe}, \text{bal}\}$ **do**
    $P_k \leftarrow \textsc{Dijkstra}(G, o', z', w_k)$ ▷ least-c_e edge
    $L_k \leftarrow \sum_{e \in E(P_k)} d_e$; $T_k \leftarrow L_k / v_m$; $n_k \leftarrow |E(P_k)|$
    $\bar{s}_k \leftarrow \frac{1}{n_k} \sum_{e \in E(P_k)} \hat{s}_e$
    $R_k \leftarrow \text{round}\big(100\,(10 - \bar{s}_k)/9\big)$
    attach $\hat{s}_e$ of every segment and the explanations
end for
return $\{(P_k, L_k, T_k, n_k, R_k)\}$

backend. If any alerts are triggered, they are sent back to the client using Socket.IO. An incident report will include a type, description, optional image and location. This will be unverified until viewed by a moderator. An SOS request will store the last known location and alert the users emergency contacts. These features are not part of the routing process. Storing an SOS event does not confirm that any messages have been sent.

D. A Length-Normalised Refinement

Section VII-D shows that (3) and (4) give incentives to short paths. Algorithm 3 is the alternate algorithm that we have implemented and tested offline. This algorithm penalises the risk by a factor of the time taken to traverse instead of the number of segments. Given a traversal time of t_e and normalised risk value of $r_e = (10 - \hat{s}_e)/9 \in [0, 1]$, we compute the cost of an edge as follows:

$$w\lambda(e) = t_e\,(1 + \lambda\, r_e), \tag{7}$$

This will give the shortest path when $\lambda = 0$. The higher λ is the more we avoid low scoring segments by a factor of how long they would take to traverse. The *exposure* of a path is given by $X(P) = \sum_{e \in E(P)} t_e r_e$.

There are three more changes associated with (7). In each mode only the relevant edges are available. That is, there are no footways, stairs, paths, pedestrian roads, cycleways, car tracks or motorways. Any pedestrian can also walk along a one way street both ways. The speed of driving on different roads varies depending on their type. Motorways have a speed 1.6 times the basic speed of 22 km/h. Living streets have a speed 0.45 times that of the basic speed. The average score of a route is weighted by the length of each segment. That is, a 20 m alley is less important than a 600 m main road.

For a grid of eight weights , potential routes are found. Since the cost of a given route is linear in (cost = t e + t e r e), the best cost for a given is given by the lower envelope of all the linear functions and is thus a concave function. If the same route is the best for two points in the grid, the line representing it will be tangent to the concave function at those two points and will therefore be the best route in between. This is exploited in Algorithm 3. For each search the algorithm performs an A [21] search with a heuristic of h(v) = 0.95(v,z)/v max, where is the Euclidean distance and v max is the maximum speed possible for that mode. Since all edges have a cost d e/v max, this will never over-estimate. The fastest route is labelled *Fastest*. The least exposed route out of those that are no more than 35% slower is labelled *Safest*. The route found for =2 (or the highest weight below this that fits within the cap) is labelled *Balanced*. If two routes are identical, they are combined into one route and all of their labels are assigned.

VII. Experiments and Results

A. Setup

The area-based scenarios executed on a MacBook laptop running macOS. The time recorded are real-world times taken from the service logs. The routing scenarios in Section VII-D were run offline on a Linux container using copies of the cached graph. These times are therefore indicative. The core extent was used for RQ1 and RQ3, whilst the trip around campus for RQ2 was run using the automatic extent.

B. RQ1: Building and Scoring an Area

As shown in Table II, 31.9s and 33.1s were taken to fully create the model, including feature extraction and training. For this reason, it would not be feasible to recreate the model for each request. However as the registry only uses it once per area, this is an acceptable time. All 160,077 segments had a model prediction and no segments were assigned the neutral default.

The predictions range from 2.46 to 8.18 with a standard deviation of 0.97. The model is therefore using only a section of the range of 1-10. Most segments are around one away from the mean. This is not an issue with XGBoost, but is due to the target being based on only a few broad rules. This does affect

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

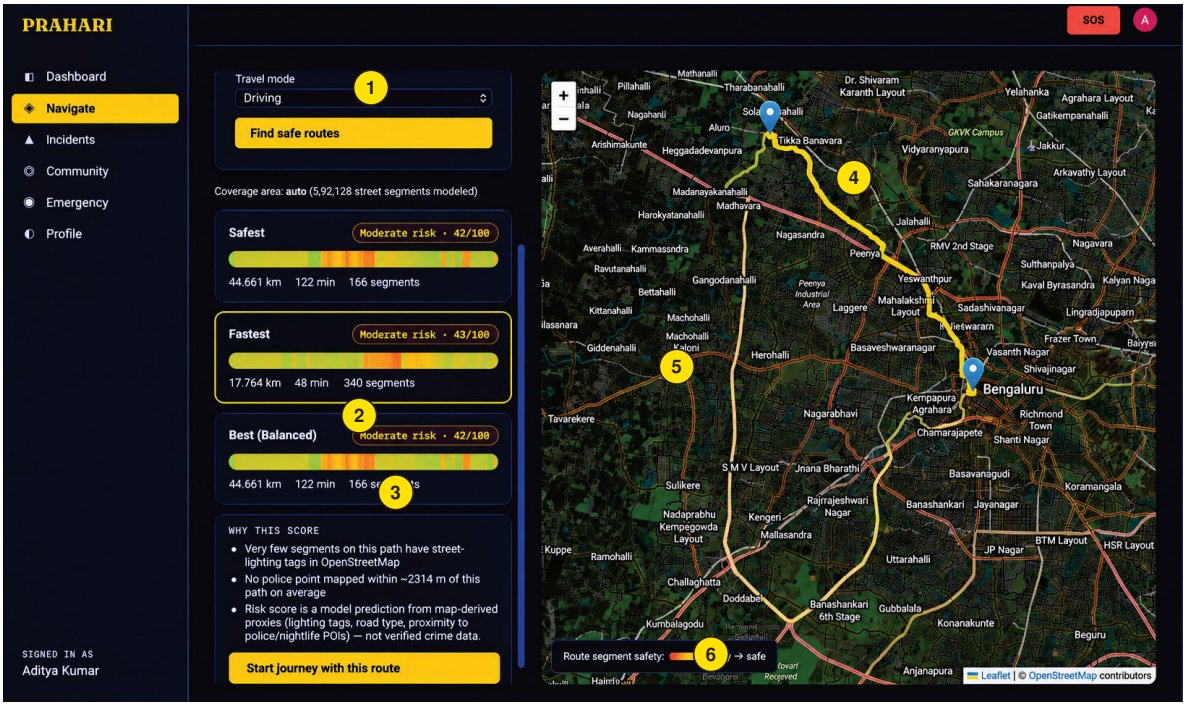


Fig. 3. The PRAHARI prototype answering a driving request from the campus to Majestic; the pins mark the start and the destination. (1) Request form with start, destination and travel mode. (2) Coverage of the automatic extent (592,128 segments). (3) Route cards with risk index, length, time and segment count; the ribbon on each card colours its segments by score. (4) The selected route (Fastest, 17.8 km along Tumakuru Road), drawn segment by segment from red (lower score) to green (higher score). (5) The alternative labelled Safest (44.7 km), which loops round the western edge of the city. (6) Legend of the segment colour scale. Map data © OpenStreetMap contributors.

Algorithm 3 Length-normalised route selection (offline variant)

Input: G, o', z', mode m; $\Lambda = (0, 0.5, 1, 2, 4, 8, 16, 32)$; cap $\delta_c = 0.35$; $\lambda_b = 2$
Output: labelled routes

keep the edges usable in mode m; $t_e \leftarrow d_e / v_m(e)$; $r_e \leftarrow (10 - \hat{s}_e)/9$
$w_\lambda(e) \leftarrow t_e(1 + \lambda r_e)$
$Q_0 \leftarrow \mathrm{A}^*(w_{\Lambda_0})$; $Q_7 \leftarrow \mathrm{A}^*(w_{\Lambda_7})$; REFINE$(0, 7)$
$\mathcal{C} \leftarrow$ distinct paths among $Q_0, \ldots, Q_7$
$P_{\text{fast}} \leftarrow \arg\min_{P \in \mathcal{C}} T(P)$
$\mathcal{C}' \leftarrow \{P \in \mathcal{C} : T(P) \le (1 + \delta_c)\, T(P_{\text{fast}})\}$
$P_{\text{safe}} \leftarrow \arg\min_{P \in \mathcal{C}'} \sum_{e \in E(P)} t_e r_e$
$P_{\text{bal}} \leftarrow Q_j$ for the largest $\Lambda_j \le \lambda_b$ with $Q_j \in \mathcal{C}'$ (else P_{fast})
merge identical paths and keep all their labels
return the labelled routes

procedure REFINE(i, j)
  if $j - i \le 1$ **or** $Q_i = Q_j$ **then**
    $Q_k \leftarrow Q_i$ for all $i < k < j$; **return**
  end if
  $k \leftarrow \lfloor (i + j)/2 \rfloor$; $Q_k \leftarrow \mathrm{A}^*(w_{\Lambda_k})$
  REFINE(i, k); REFINE(k, j)
end procedure

TABLE II
AREA GRAPH, TRAINING DATA AND BUILD STATISTICS

Quantity	Value
Core extent (central Bengaluru)	
Graph nodes	60,575
Directed road segments	160,077
Attributes per segment	7
Training rows (80%)	128,061
Held-out rows (20%)	32,016
Segments scored by the model	160,077 (100%)
Segments on the neutral default	0
Predicted score, minimum–maximum	2.46–8.18
Predicted score, standard deviation	0.97
Area build time (two builds)	31.9 s, 33.1 s
Automatic extent (campus to city centre)	
Road segments modelled	592,128

the downstream processing however. As the mean of a route will be based on a few hundred segments, there isn't much difference between each of the alternatives. This can be seen in RQ2.

C. RQ2: Campus to Majestic

The prototype provides a route for a driving request from Acharya Dr. S. Radhakrishnan Road (outside campus) to Nadaprabhu Kempegowda Station (Majestic) as shown in Fig. 3. The three routes are given in Table III. The fastest route takes the route through Tumakuru Road, Peenya and

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

TABLE III
ROUTES RETURNED FOR THE CAMPUS–MAJESTIC TRIP (DRIVING)

Route	L (km)	T (min)	n	L/n (m)	R
Fastest (shortest)	17.764	48	340	52.2	43
Safest	44.661	122	166	269.0	42
Balanced	44.661	122	166	269.0	42
Safest vs. fastest	+151.4%	+154.2%	−51.2%	×5.15	−1

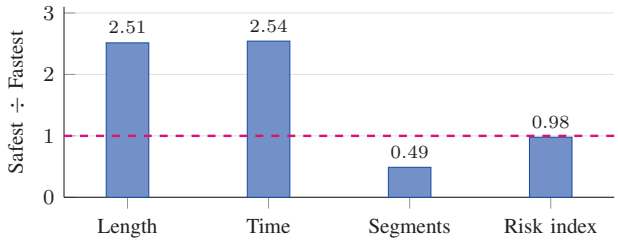


Fig. 4. Safest route relative to the fastest for the campus–Majestic trip (dashed line: ratio of 1). Length and time grow by a factor of about 2.5 and the segment count halves, while the risk index falls by about 2%.

Yeshwanthpur and then heads into the centre via Malleswaram. The safest route goes around the western edge of the city and approaches from the south. This route is the same as the safest route.

The figures are very straightforward and difficult to argue. The safest route is 26.9 km (151.4%) longer and 74 min slower (154.2%). However, the risk index decreases from 43 to 42. Both routes are in the moderate range. The ratios can be seen in Fig. 4. No one should be advised to take 74 minutes longer to save one point in the 100 point index. Any user seeing this card would quickly lose faith in using the safest route. The safest route can also be seen to have 51% fewer segments (on average 5 times longer) in one column of the table (Table III).

D. RQ3: Why the Safest Route Was So Long

The cause lies in (3). Because $c_e = 11 - \hat{s}_e$, the objective can be written as

$$\sum_{e \in E(P)} c_e = n(P)\,\big(11 - \bar{s}(P)\big), \tag{8}$$

This can be expressed as the number of segments multiplied by the average cost per segment. Since the scores are very similar, the second element does not change much between the different potential paths. Therefore, the search tends to pick the path with the *least number of segments*. From equations (8) and Table III, the objective function is approximately 790 for the safest and 1,650 for fastest path for the campus trip. This means that by using the detour, the objective function has doubled but the average value has barely changed. This is simply due to the detour having long segments.

The balanced objective function (4) also suffers from this issue. The distance element $d_e/d_{\max}$ will be small for any segments much shorter than the longest in the area. The safety element adds an amount of 0.6 ce/10 0.29 per segment. Once again the objective function is approximately equal to the number of segments. This means the balanced and safest routes shown in Table III are the same. This is a scaling issue, where the weights of 0.6 and 0.4 have become meaningless (Marler and Arora [22]).

TABLE IV
OFFLINE ROUTING CHECKS ON BENGALURU GRAPHS

Trip	Observation
Deployed costs (2)–(4)*; the search ignores the mode*	
Malleshwaram → Richmond Town	3.1 km (55%) of the path shown for driving ran on footways.
MG Road → Jayanagar	"Safest" path about 10% longer than the shortest and no safer; it used a segment scored 3.29 that the shortest avoided.
Length- or time-normalised costs (Section VI-D)	
Five central trips (driving)	0 m of footways, steps or paths on every route (Algorithm 3).
Malleshwaram → Richmond Town (walking)	Lowest segment score on the safest route 5.58 versus 3.21 on the fastest, for +0.3 min (Algorithm 3).
Koramangala → Indiranagar	With cost $d_e(1 + \lambda r_e)$, one path stays optimal for every $\lambda \leq 20$.
Campus → Sapthagiri Hospital	Fastest, safest and balanced coincide; one card with three labels (Algorithm 3).
Latency, core extent	0.4–1.5 s per request; about 1 s once per mode for the road index.

From Table IV we can see some further offline tests. The search does not take into account the mode of transport. This meant that the driving route between Malleshwaram and Richmond Town had 3.1km (55

These issues have been resolved by using normalised costs. All driving routes for the 5 central trips did not have any footway. The safest route walking between Malleshwaram and Richmond Town increased the minimum score per segment by 5.58 from 3.21 but only added 0.3 mins. This is probably something a pedestrian would be willing to pay. For 2 trips the truthful answer was just 1 route. This is shown in the improved interface as 1 card with multiple labels. Finally, in the randomized tests on the core graph, A never produced a higher cost path than the Dijkstra search from NetworkX. Most trips only required 2 searches as only the first and last elements of the λ grid were required.

VIII. LIMITATIONS AND ETHICAL CONSIDERATIONS

Target Validation As the model was trained to learn our rules, showing the model fits to the target is not a good indication of safety. Therefore, we do not show results for error on held-out data as this would just indicate how well the XGBoost model fits to our rules. To ensure that the model is performing well on real data, there should be independent outcomes such as geocoded incidents or audits (such as those collected by SafetiPin [2]). The split should also be blocked by location (e.g. [25]) and not randomly as currently, neighbour segments are leaking into the held-out rows.

Map Data If the tag for `lit` or `lanes` is missing from the map, this does not mean that the street has those properties.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Police and nightlife depend on what has been mapped by volunteers. Therefore, having a police station nearby does not indicate anything about response times. The sign for nightlife is also questionable as busy venues mean more eyes on the street [18]. The score does not take into account the time of day, traffic and weather. All travel speeds are constant.

Supporting Functions The journey check function that was deployed did not compare the user location to the destination but rather the line of the journey. This caused false alerts. The client socket also did not include the sign in token meaning that the alerts were not sent to the user's room. These issues have been resolved and tested but not merged yet. SOS dispatch was only tested in simulated mode.

Ethics Privacy A single number could cause a neighbourhood to appear dangerous when there is little evidence. PRAHARI makes it clear that the scores are estimates and gives an explanation based on the map features. Community reports are not verified until they have been checked by a moderator. Journeys and reports contain personal location data and can only be accessed through authenticated accounts. There still needs to be a limit on how long this data is stored before being released publicly. No user study has been done.

IX. Conclusion and Future Work

PRAHARI creates a weighted road network from open source map data in order to provide shortest, safest and balanced routes. We were able to create a weighted network with 160,077 segments in central Bengaluru in around 30 seconds. Every segment was predicted. The most useful insight was learning from our mistake. By weighting each segment by safety cost, we found that a route of length 17.8 km became 44.7 km longer due to the weighting. This only gave an increase of 1 point. This meant that we were rewarding long segments rather than safer routes. By weighting each segment with a risk value per unit time travelled, taking into account the travel mode, and limiting the detour, we created sensible routes using the same data.

As next steps we will make the improved search our default, compare our scores to external incident/audit data with spatial blocking, include time of day features and perform a user study with women who commute in Bengaluru.

Acknowledgment

We would like to thank the Department of Computer Science and Engineering, Acharya Institute of Technology, Bengaluru for providing academic support and the project guide for providing detailed feedback on our initial draft. Map data ©by OpenStreetMap contributors.

References

[1] National Crime Records Bureau, *Crime in India 2024*. New Delhi, India: Ministry of Home Affairs, Government of India, May 2026.

[2] K. Viswanath and A. Basu, "SafetiPin: An innovative mobile app to collect data on women's safety in Indian cities," *Gender & Development*, vol. 23, no. 1, pp. 45–60, 2015, doi: 10.1080/13552074.2015.1013669.

[3] E. Galbrun, K. Pelechrinis, and E. Terzi, "Urban navigation beyond shortest route: The case of safe paths," *Inf. Syst.*, vol. 57, pp. 160–171, Apr. 2016, doi: 10.1016/j.is.2015.10.005.

[4] J. Kim, M. Cha, and T. Sandholm, "SocRoutes: Safe routes based on tweet sentiments," in *Proc. 23rd Int. Conf. World Wide Web (WWW Companion)*, Seoul, Korea, Apr. 2014, pp. 179–182, doi: 10.1145/2567948.2577023.

[5] S. Shah, F. Bao, C.-T. Lu, and I.-R. Chen, "CROWDSAFE: Crowd sourcing of crime incidents and safe routing on mobile devices," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2011, pp. 521–524.

[6] F. Mata *et al.*, "A mobile information system based on crowd-sensed and official crime data for finding safe routes: A case study of Mexico City," *Mobile Inf. Syst.*, vol. 2016, Art. no. 8068209, 2016, doi: 10.1155/2016/8068209.

[7] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct.–Dec. 2008, doi: 10.1109/MPRV.2008.80.

[8] OpenStreetMap contributors, "OpenStreetMap." [Online]. Available: https://www.openstreetmap.org. Accessed: Aug. 19, 2026.

[9] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Comput., Environ. Urban Syst.*, vol. 65, pp. 126–139, 2017, doi: 10.1016/j.compenvurbsys.2017.05.004.

[10] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[11] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959, doi: 10.1007/BF01386390.

[12] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proc. 7th Python Sci. Conf. (SciPy)*, Pasadena, CA, USA, 2008, pp. 11–15.

[13] S. Aljubayrin, J. Qi, C. S. Jensen, R. Zhang, Z. He, and Z. Wen, "The safest path via safe zones," in *Proc. IEEE 31st Int. Conf. Data Eng. (ICDE)*, 2015, pp. 531–542.

[14] D. Quercia, R. Schifanella, and L. M. Aiello, "The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city," in *Proc. 25th ACM Conf. Hypertext Social Media (HT)*, Santiago, Chile, 2014, pp. 116–125, doi: 10.1145/2631775.2631799.

[15] S. Sohrabi, Y. Weng, S. Das, and S. G. Paal, "Safe route-finding: A review of literature and future directions," *Accident Anal. Prev.*, vol. 177, Art. no. 106816, 2022, doi: 10.1016/j.aap.2022.106816.

[16] S. Sohrabi and D. Lord, "Navigating to safety: Necessity, requirements, and barriers to considering safety in route finding," *Transp. Res. C, Emerg. Technol.*, vol. 137, Art. no. 103542, 2022, doi: 10.1016/j.trc.2021.103542.

[17] B. C. Welsh and D. P. Farrington, "Effects of improved street lighting on crime," *Campbell Syst. Rev.*, vol. 4, no. 1, pp. 1–51, 2008, doi: 10.4073/csr.2008.13.

[18] J. Jacobs, *The Death and Life of Great American Cities*. New York, NY, USA: Random House, 1961.

[19] C. Barrington-Leigh and A. Millard-Ball, "The world's user-generated road map is more than 80% complete," *PLoS ONE*, vol. 12, no. 8, Art. no. e0180698, 2017, doi: 10.1371/journal.pone.0180698.

[20] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001, doi: 10.1214/aos/1013203451.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[22] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: New insights," *Struct. Multidiscip. Optim.*, vol. 41, no. 6, pp. 853–862, Jun. 2010, doi: 10.1007/s00158-009-0460-7.

[23] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[25] D. R. Roberts *et al.*, "Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure," *Ecography*, vol. 40, no. 8, pp. 913–929, 2017, doi: 10.1111/ecog.02881.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)