International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

# GuardianPath: Time-Aware Social Visibility Framework for Safe Pedestrian Route Navigation Using Machine Learning

**Afza Ruheen[1], P Bavithra Matharasi[2]**

[1,2]Department of Computer Science, Mount Carmel College(Autonomous), Bengaluru, India

**Abstract**

Every mainstream pedestrian navigation app optimises for distance or speed, and none of them account for whether the route actually feels safe to walk. A street lined with open shops at noon can turn pitch-dark and empty by midnight, yet current routing engines send a user down it regardless of the hour. This paper presents GuardianPath, a framework that takes Jane Jacobs' concept of natural surveillance, defined as safety through the sheer presence of people and open businesses, and turns it into something a computer can measure, learn, and act on, using nothing beyond free OpenStreetMap data. Five features, namely guardian proximity, active-hour flag, active POI density, anchor presence, and night penalty, are computed for road nodes in Bengaluru, India, and fed into a weighted formula that generates 12,000 labelled samples spanning every hour of the day. An XGBoost regressor trained on these samples achieves an R2 of 0.9937 with a mean absolute error of 0.008, reproducing the hand-crafted formula almost exactly. The model predictions are used within a modified Dijkstra search through a composite edge-cost function, yielding a safety-optimised route alongside the conventional shortest path. SHAP values accompany every prediction so that users can see which features drove the recommendation. On real Bengaluru routes, the safe path is found to diverge noticeably from the shortest path at night, adding only about two percent extra distance, while active POI density along the same corridor increases nearly fivefold between midnight and mid-morning, confirming that the system tracks when the city is awake versus asleep.

**Keywords:** Explainable Artificial Intelligence, Natural Surveillance, OpenStreetMap, Pedestrian Safety, Points of Interest, Safe Route Navigation, SHAP, Smart Cities, Social Visibility, Time-Aware Routing, XGBoost

## 1. Introduction

Cities in the developing world are expanding fast, and more people than ever walk as their primary mode of transit. This has made pedestrian safety a pressing concern, one that mainstream navigation tools have done remarkably little to address. Google Maps, for instance, routes a user down the shortest or quickest path without any regard for whether that path feels safe to walk. For a woman heading home alone after dark, for a college student leaving campus at midnight, or for an elderly person crossing an unfamiliar part of town, the question that matters is not "How far?" but "Will I feel safe on this route?" The safety of a given street is not a permanent attribute baked into the concrete; a stretch of road lined with open restaurants and busy cafes at 2 PM can turn into a dark, deserted corridor by 11 PM. Jacobs (1961)

described this phenomenon as natural surveillance, the informal policing that happens simply because people are around, shops are open, and the street feels alive [1]. Once those "eyes on the street" vanish for the night, so does the deterrent effect, and both actual and perceived risk climb sharply. Jacobs' ideas have existed since the 1960s, yet nobody has managed to bake them into a working navigation system.

The attempts made so far fall roughly into three categories, each with its own shortcoming. Rule-based methods [6] assign static weights to Point-of-Interest (POI) categories and treat the result as a safety score, but they treat every hour of the day the same and break down when applied to a new city. Reinforcement learning approaches [20] frame navigation as a sequential decision problem, which sounds elegant until it is realised that they require crime statistics that do not exist in any structured, geo-referenced form for most cities in the Global South. Spatial risk-mapping methods [21] produce hotspot maps via kernel density estimation or deep learning, but those maps are static snapshots that cannot be wired into a real-time router without substantial additional engineering. What is missing is a system that does all four of the following at once: (i) scores safety differently depending on the hour; (ii) runs entirely on free, open geospatial data; (iii) can explain why it picked a particular route; and (iv) computes edge-level scores fast enough for real-time graph search.

GuardianPath is presented to fill this gap. The idea, to the authors' knowledge not attempted before, is to take Jacobs' surveillance principle, turn it into five measurable features, write a rule-based formula that generates labelled training data without manual annotation, and train an XGBoost regressor [13] to approximate that formula at millisecond speed. A SHAP [18] layer is added on top so that users can see exactly which features drove each safety prediction. The contributions of this work are:

1. A formally defined, time-aware safety feature set grounded in natural surveillance theory, comprising five features derived entirely from OpenStreetMap POI data.
2. A rule-based label generation mechanism that produces 12,000 annotated training samples without crime datasets or manual labelling.
3. A trained XGBoost regression model achieving R2 = 0.9937 in approximating the rule-based formula.
4. A composite safe-routing engine using a time-dynamic edge-cost function in a modified Dijkstra search.
5. A SHAP-based explainability layer that surfaces the contribution of each feature to individual route predictions.
6. An open-data deployment as an interactive Streamlit web application on the Bengaluru urban road network.

## 2. Related Work

### 2.1 Urban Safety Theory and Rule-Based Navigation

Jacobs [1] argued that streets where people are present, shops are open, and land use is mixed tend to be safer, not because of policing, but because of passive monitoring by ordinary citizens going about their daily lives. Newman [2] built on this and formalised it as Defensible Space, the idea that well-designed physical spaces create a sense of ownership that deters crime. Jeffery [3] extended the concept further with Crime Prevention Through Environmental Design (CPTED), drawing a direct line from built-environment features to crime deterrence. Hillier and Hanson [4] added quantitative rigour to this line of thinking through Space Syntax, showing that how streets connect to one another, their integration and connectivity, has a measurable influence on where people choose to walk and how safe they feel doing so. Three decades

of CPTED field studies, reviewed by Cozens et al. [5], confirmed this: wherever shops, cafes, and public services line a road, less crime tends to be found. Laxmi et al. [6] built on these ideas with a routing system that pulled POI data from OpenStreetMap and assigned each category a static safety weight. It was a useful proof of concept, showing that open data could stand in for expensive proprietary crime datasets, but the weights were hand-tuned and the system recommended the same route at 2 PM and 2 AM.

### 2.2 Machine Learning for Route Safety

An important empirical finding came from Quercia et al. [19], who used crowdsourced perception data to show that, given the choice, pedestrians consistently picked quieter and greener routes over the shortest one, providing evidence that people care about more than distance alone. On the modelling side, gradient-boosted ensembles, particularly XGBoost [13], have proven effective on structured tabular data of the kind used in this work. The theoretical groundwork for boosting was laid by Friedman [14], and random forests [15] offer a related, somewhat older ensemble paradigm. A recurring difficulty in machine learning is the absence of labelled data. Ratner et al. [17] offered a pragmatic workaround, data programming, in which hand-written labelling functions inject domain knowledge and produce labels at scale, sidestepping manual annotation. GuardianPath borrows this spirit: the rule-based visibility formula acts as the "teacher", generating 12,000 synthetic labels so that the XGBoost model can learn the full temporal and spatial pattern of the safety function without a single hand-labelled example.

### 2.3 Spatial Risk Prediction and Urban Computing

Chainey et al. [21] showed that crime does not scatter randomly across a city; it clusters, and those clusters can be mapped using kernel density estimation. This insight launched an entire sub-field of spatial risk prediction. On the broader urban-computing front, Zheng et al. [22] surveyed how mobility traces, environmental sensors, and social-media check-ins could be combined to understand urban behaviour. Yin and Cao [11] found that POI density and category mix are strong predictors of what is happening in a neighbourhood at any given time. Noulas et al. [12] uncovered surprisingly universal temporal rhythms in how people move through cities, regardless of the city itself. All of the data used in this work sits on top of OpenStreetMap [7], a crowd-sourced map that, whatever its inconsistencies, offers global road and POI coverage at zero cost. Boeing [8] built OSMnx to make extracting and analysing OSM street networks straightforward, and Hagberg et al. [10] developed NetworkX for running shortest-path algorithms on the resulting graphs.

### 2.4 Research Gap

Putting all of this together, the picture is clear: each family of approaches gets one or two things right but falls short on the rest. Rule-based systems work with open data but ignore time and cannot explain themselves. ML navigation systems handle explainability through learned preferences but demand crowdsourced perception data that most cities lack. Spatial risk models predict crime distributions reasonably well, yet they cannot be plugged into a real-time router as-is. GuardianPath is, to the authors' knowledge, the first system designed to address all four requirements at once.

## 3. System Architecture

When a user submits a routing query, GuardianPath runs through four stages, one after another (see Figure 1).

1. Data Collection. The road network and nearby POIs are pulled from OpenStreetMap through the Overpass API, the road graph is loaded via OSMnx [8] as a NetworkX MultiDiGraph, and each POI is tagged with its type and opening hours.

2. Feature Engineering and Labelling. For every road node, five safety features are computed and combined with a weighted formula into a visibility label V ∈ [0, 1]. Doing this for 500 sampled nodes at each of the 24 hours yields 500 × 24 = 12,000 labelled samples.
3. Model Training. An XGBoost regressor [13] learns from these labels (80/20 split). Once trained, it replaces the formula at runtime, producing predictions in milliseconds.
4. Routing and Explanation. A composite cost function mixes a heuristic danger score with the XGBoost prediction and feeds it into Dijkstra's algorithm, producing a safety-aware route alongside the ordinary shortest path. SHAP [18] values indicate which features mattered most.

The complete stack is implemented in Python: OSMnx, XGBoost, NetworkX, scikit-learn [16], SHAP, GeoPandas [9], Folium for maps, and Streamlit for the front end. The entire pipeline runs on a plain CPU; no GPU is required.

**Figure 1: System Architecture of GuardianPath**

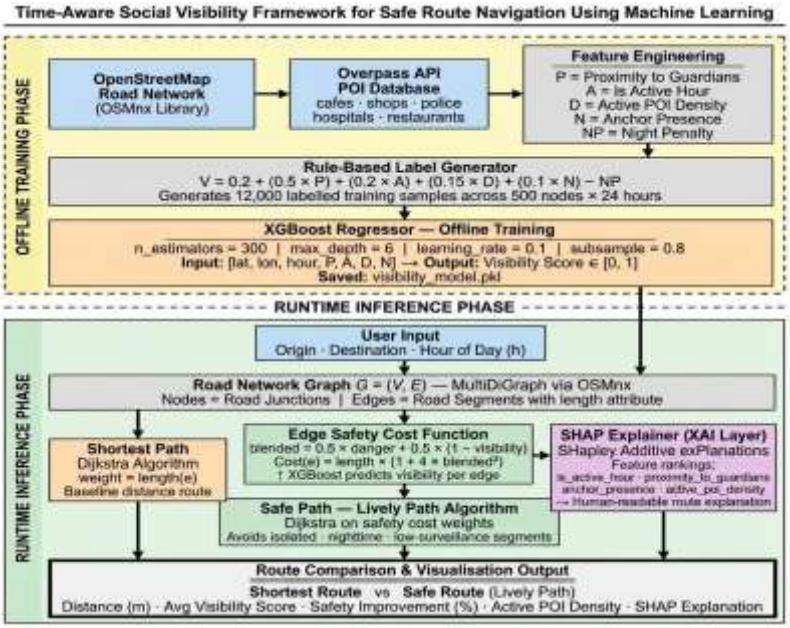


Figure 1 shows the pipeline proceeding from OSM data collection through feature engineering, XGBoost training, and safety-weighted Dijkstra routing with SHAP explainability.

## 4. Safety Features and Label Generation

This section describes the five features computed for each road node. Each one captures a different facet of spatial, temporal, or social context. After the features are defined, the formula that combines them into a single visibility label used for training is presented.

### 4.1 Feature 1 — Proximity to Guardian POIs (P)

This feature asks how close a given road node is to a place where people are likely to be present. Such places are referred to as "guardian POIs": cafes, pharmacies, restaurants, retail shops, and the like. A KD-tree spatial index [23] is built over all POIs within 500 metres and the following is computed:

$$P = \min\left( (1/5) \Sigma\, i \in N(500\ m)\ \max(0,\ 1 - di/ri),\ 1 \right) \qquad (1)$$

where di is the straight-line distance to the i-th POI (converted from degrees to metres using lat × 111,000 and lon × 108,000), ri is the influence radius assigned to that POI type (cafes are assigned 100 m, hospitals 400 m; these are configurable), and N(500 m) is the set returned by the KD-tree within the 500 m search

window. The five nearest contributions are averaged and the result is clamped to [0, 1].

### 4.2 Feature 2 — Active Hour Flag (A)

A simple yes-or-no flag indicating whether the hour is one when shops and offices are likely to be open:

$A = 1$, if $7 \leq h < 22$; $A = 0$, otherwise (2)

Between 7 AM and 10 PM, streets typically have foot traffic and open storefronts; outside that window, most commercial activity shuts down.

### 4.3 Feature 3 — Active POI Density (D)

This feature approximates how busy an area is at a given moment, by scaling the proximity score according to the hour:

$D = P \times 1.0$, if $A = 1$; $D = P \times 0.3$, if $A = 0$ (3)

During daytime, density equals the full proximity score, since shops are open and the street is alive. At night, it is reduced to 30% because only a handful of establishments, such as 24-hour pharmacies and late-night fast food, remain active.

### 4.4 Feature 4 — Anchor Presence (N)

This feature captures whether round-the-clock safety institutions, such as police stations, hospitals, and pharmacies, are nearby, using a scaled-down version of the proximity score:

$N = 0.5 \times P$ (4)

The 0.5 multiplier reflects the fact that anchors provide a more diffuse, background sense of security rather than the direct effect of an open cafe nearby.

### 4.5 Feature 5 — Night Penalty (NP)

Late at night, surveillance drops sharply. This is modelled as a flat deduction: $NP = 0.3$, if $h \geq 22$ or $h < 6$; $NP = 0.0$, otherwise (5)

Subtracting 0.3 during the 10 PM to 6 AM window is deliberately aggressive; it forces the model to treat nighttime roads as substantially more dangerous unless strong guardian features compensate.

### 4.6 Rule-Based Visibility Label

The ground-truth visibility label V for each training sample is given by:

$V = \text{clip}(0.2 + 0.5P + 0.2A + 0.15D + 0.1N - NP + \varepsilon, 0, 1)$ (6)

The 0.2 baseline represents a bare minimum of visibility, since even a residential lane has street lights. The Gaussian noise term $\varepsilon \sim N(0, 0.1)$ prevents the model from memorising the formula exactly and introduces the kind of scatter that would be seen in real-world measurements. Table 1 summarises each weight and the reasoning behind it.

**Table 1: Feature Weights in the Rule-Based Visibility Formula**

| Feature | Symbol | Weight | Justification |
|---|---|---|---|
| Base score | — | 0.20 | Minimum visibility on any road |
| Guardian proximity | P | 0.50 | Strongest signal, people nearby |
| Active hour | A | 0.20 | Time defines social context |
| Active POI density | D | 0.15 | Busyness adds surveillance |

| | | | |
|---|---|---|---|
| Anchor presence | N | 0.10 | 24/7 passive safety |
| Night penalty | NP | −0.30 | Reduced natural surveillance |

### 4.7 Worked Examples

Three examples illustrate the formula. Example 1, daytime busy market at 09:00: with a cafe 40 m away (P = 0.80), six of ten POIs open (D = 0.80), a police station 300 m away (N = 0.40), active hour (A = 1), and no night penalty, V = 0.96, classified as High Safety. Example 2, isolated road at 23:00: nearest shop 160 m away (P = 0.20), no POIs open (D = 0.06), no nearby anchor (N = 0.10), outside active hours (A = 0), night penalty applied, V = 0.02, classified as Very Unsafe. Example 3, evening near a police station at 18:00: restaurant 10 m away (P = 0.95), four of ten POIs open (D = 0.95), police station 100 m away (N = 0.475), active hour (A = 1), no night penalty, V = 1.065, clipped to 1.00.

## 5. XGBoost Regression Model

### 5.1 Training Dataset

500 road nodes were sampled from the Bengaluru walking graph (bounding box: 12.95°N–13.01°N, 77.56°E–77.65°E), and the feature vector and rule-based visibility label were computed at each of the 24 hours, yielding 500 × 24 = 12,000 labelled samples. POIs are fetched via the Overpass API and stored in a GeoDataFrame.

### 5.2 Model Configuration

Table 2 lists the XGBoost hyperparameters. The ten input features are: lat, lon, hour_of_day, day_of_week, is_night, is_peak_hour, proximity_to_guardians, is_active_hour, active_poi_density, and anchor_presence. The target is the rule-based visibility score $V \in [0, 1]$.

**Table 2: XGBoost Hyperparameters**

| Parameter | Value |
|---|---|
| Number of estimators | 300 |
| Maximum depth | 6 |
| Learning rate (η) | 0.1 |
| Subsample ratio | 0.8 |
| Column sample by tree | 0.8 |
| Objective | reg:squarederror |
| Random state | 42 |

### 5.3 Training Procedure

The dataset is split into 9,600 training and 2,400 test samples (80/20, random seed 42). XGBoost trains sequentially: each tree fits the residuals left by all preceding trees, with corrections scaled by η = 0.1:

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x) \quad (7)$$

where $F_m(x)$ is the ensemble prediction after m trees, $h_m(x)$ is the m-th tree's output, and $F_0$ is the mean of all training labels. After 300 rounds, the final prediction is:

$$\hat{V}(x) = F_0 + \eta \sum_{m=1}^{300} h_m(x) \quad (8)$$

### 5.4 Evaluation Results

An R2 of 0.9937 indicates that the XGBoost model has learned the teacher formula almost perfectly. For practical purposes, the compact model reproduces the five-feature scoring function with negligible error, which is exactly what is required: a fast predictor that can be embedded in the Dijkstra loop without recomputing the full rule-based pipeline at every node.

**Table 3: XGBoost Model Performance on Held-Out Test Set**

| Metric | Value | Interpretation |
|---|---|---|

| | | |
|---|---|---|
| R2 Score | 0.9937 | Explains 99.37% of label variance |
| MAE | 0.0080 | Average error of 0.008 on a [0, 1] scale |
| RMSE | 0.0175 | Root mean square error of 1.75% |

## 6. Safe Routing Algorithm

### 6.1 Edge Cost Function

For an edge (u, v) with physical length $\ell$, the composite safety cost is:

$$C(u, v) = \ell \times (1 + 4 \cdot b^2) \quad (9)$$

where b is a blended danger score:

$$b = 0.5 \cdot D_{heuristic} + 0.5 \cdot (1 - \hat{V}) \quad (10)$$

The heuristic component is:

$$D_{heuristic} = W_{poi} \cdot (1 - \rho) + W_{guard} \cdot (1 - g) + W_{time} \cdot tr \quad (11)$$

where $\rho$ is the active POI density at the edge midpoint, g is the guardian proximity score, tr is the time-of-day risk factor (0.8 at night, 0.3 during evening, 0.5 during daytime), and the weights shift with the hour as shown in Table 4. The result is that a well-lit, busy edge ($\hat{V} \approx 1$) costs roughly its physical length, while a dark, deserted edge ($\hat{V} \approx 0$) is inflated by up to 5×, causing Dijkstra to steer away from it unless there is no alternative.

**Table 4: Time-Dynamic Routing Weights**

| Time Band | Wguard | Wpoi | Wtime |
|---|---|---|---|
| Day (06:00–16:59) | 0.35 | 0.30 | 0.15 |
| Evening (17:00–21:59) | 0.35–0.40 | 0.30–0.35 | 0.15–0.30 |
| Night (22:00–05:59) | 0.40 | 0.35 | 0.30 |

### 6.2 Route Differentiation

The safe route is also required to look meaningfully different from the shortest path; otherwise, showing two lines on the map would add little value. After the shortest path is computed, the set of edges Eshort that it uses is identified and a 3× cost penalty is applied to each of them during the safety Dijkstra run. This pushes the safe route onto alternative streets unless every alternative is substantially longer.

### 6.3 Time-Aware POI Activity Mapping

Not all POI types are relevant at every hour. The day is split into four bands, namely morning (06:00–09:59), daytime (10:00–17:59), evening (18:00–21:59), and night (22:00–05:59), and each is associated with the POI types that would realistically be found open. At night, this is largely hospitals, police stations, pharmacies, petrol stations, ATMs, hotels, and 24-hour fast food. During the day, the full range of shops, restaurants, banks, parks, and transit stops comes into play. Where an OSM opening_hours tag exists, it is parsed directly; otherwise, the time-band fallback is used. In either case, the active density score at a given

hour reflects what is actually open rather than a static average.

## 7. SHAP Explainability Layer

To justify telling a user that a particular route is safer, the system must be able to explain why. For this reason, GuardianPath feeds every XGBoost prediction through SHAP (SHapley Additive exPlanations) [18], which decomposes the prediction into signed contributions from each input feature. For a prediction $\hat{V}(x)$, SHAP computes a set of values $\{\phi 1, \phi 2, ..., \phi k\}$ satisfying:

$$\hat{V}(x) = \phi 0 + \Sigma\ (j=1 \text{ to } k)\ \phi j \qquad (12)$$

where $\phi 0$ is the base (expected) value and each $\phi j$ quantifies the marginal contribution of feature j. Positive $\phi j$ values indicate features that increase visibility, such as high guardian proximity at night, while negative values indicate features that suppress it, such as the night penalty flag. In the Streamlit interface, SHAP results appear as a plain-English explanation panel. For each road segment, the system identifies the top factors, translates them into readable statements, for example "Near Guardian: 95% proximity to police/hospital", and displays them alongside the predicted visibility score. The goal is to allow the user to examine the reasoning behind why a stretch was rated safe or flagged, rather than requiring them to accept the recommendation without explanation.

## 8. Experimental Results

### 8.1 Experimental Setup

GuardianPath was tested on the walking network of central Bengaluru, India, a bounding box from 12.95°N to 13.01°N and 77.56°E to 77.65°E, covering a mix of residential lanes, commercial corridors, and arterial roads. All runs used a standard laptop CPU; no GPU was involved.

### 8.2 Route Comparison Experiments

Two origin-destination pairs were selected and run at different hours to determine whether the router changes its recommendation depending on the time.

**Figure 2: Route Comparison at Night — Good Hope School to Mount Carmel College (Red = Shortest Path, Green Dashed = Safe Path)**


Scenario 1, Night. Good Hope School to Mount Carmel College, evaluated between 22:00 and 06:00. At night, urban activity is minimal and the guardian proximity and anchor presence features carry the highest

safety weights. As shown in Figure 2, the shortest path and the safe path exhibit clear spatial divergence: the safe route re-routes through corridors with higher concentrations of late-night establishments near the Shivajinagar corridor. After dark, the city is mostly shut. Both paths encounter 136 active POIs, but the safe route packs them more tightly, 15.6 POIs/km versus 14.0 on the shortest path, despite adding only 70 metres (4.69 km vs. 4.62 km). Most of those late-night POIs are fast-food outlets, exactly the kind of 24-hour establishment the night-band filter is designed to identify. The recommendation engine's verdict was: "Safe route is recommended. Trade-off: 2% longer but 3% safer."

**Figure 3: Route Comparison at 10:00 AM — Coles Park to Mount Carmel College (Red = Shortest Path, Green Dashed = Safe Path)**


Scenario 2, Daytime. Coles Park to Mount Carmel College, evaluated at 10:00 AM. At this hour, the two routes diverge less dramatically. The daytime dynamic weights (Wguard = 0.35, Wpoi = 0.30, Wtime = 0.15) are set conservatively, as most roads are well-populated during daylight. Mid-morning tells a different story: of 1,238 nearby POIs, 674 are open, mostly restaurants, and 443 guardian facilities line the safe route. The safe path is 0.4 km longer (4.84 vs. 4.44 km) but scores only 87% against the shortest route's 88%. Because the gap is negligible, the engine's recommendation was: "Both routes are safe at this time. The shortest route is recommended."

**Table 5: Cross-Scenario POI Density Comparison**

| Metric | Night (22:00) | Day (10:00) | Ratio |
|---|---|---|---|
| Active POIs | 136 | 744 | 5.5× |
| Active Density (POIs/km) | 14.0 | 95.4 | 6.8× |
| Guardian Facilities | 88 | 443 | 5.0× |

| | | | |
|---|---|---|---|
| Safety Score (Safe Route) | 3% higher | Equal | — |

### 8.3 Time-Aware Active POI Density Analysis

Comparing the two scenarios side by side makes the temporal sensitivity clear. The same streets that host 744 active POIs at 10 AM are left with just 136 at night, roughly a fivefold drop. Active density shows an even sharper contrast, jumping from 14.0 POIs/km after dark to 95.4 POIs/km in broad daylight, a 6.8× swing. All of this variation comes from the time-band filter; the road graph itself does not change. In other words, the framework tracks when the city wakes up and shuts down, rather than treating every hour with the same static profile.

### 8.4 SHAP Feature Attribution

The SHAP values indicate which features the model relies on most. At night, is_night is by far the largest drag on visibility scores, pulling every segment down. Guardian proximity partially offsets this, boosting scores wherever a hospital or 24-hour chemist is nearby. During the day, the pattern reverses: is_active_hour becomes the dominant positive factor, and active_poi_density contributes a large positive push because many more establishments are open. This pattern is consistent with expectations: at night, the model relies on is_night and guardian_proximity, while during the day, is_active_hour and active_poi_density take over. This is exactly the behaviour encoded into the features, and observing SHAP reproduce it independently builds confidence that the explanations are trustworthy rather than post-hoc rationalisations.

### 8.5 System Performance

In practical terms, a user waits about ten seconds for the first route while caches warm up, and under five seconds for every route after that, well within the patience threshold of someone standing on a street corner deciding which way to walk. No GPU is required.

**Table 6: System Latency Benchmarks**

| Operation | Latency |
|---|---|
| City graph load (local cache) | ~4 s |
| OSM POI fetch (Overpass cache hit) | <1 s |
| Node proximity cache build | ~3–5 s (once) |
| Node RT density cache build | ~3–5 s (once) |
| Dijkstra — safe route computation | ~2–4 s |
| Total first-route query latency | ~10–15 s |
| Subsequent route queries (same area) | <5 s |

## 9. Discussion

Taken together, the experiments show that Jacobs' surveillance concept can be turned into a working, explainable routing system that behaves differently at night than during the day. A corridor scoring 95.4 active POIs/km at 10 AM drops to 14.0 at midnight, and the router responds by rerouting through whatever

late-night establishments it can find. The SHAP layer confirms that the safety features the model relies on are the ones that would be expected: guardian proximity dominates at night, and active-hour status takes over during the day. That said, several constraints bound the current implementation. The cumulative node-scoring approach used by Dijkstra's algorithm can favour a route that aggregates most of its safety from a dense POI cluster near one endpoint, even when the rest of the path is poorly monitored. A minimum-link formulation, bounding the route score by its weakest segment rather than its sum, would better reflect the continuous-surveillance requirement of real pedestrian safety and is the most targeted algorithmic improvement for future work. At present, the entire system is anchored to Bengaluru; it cannot be applied to Mysuru or Mumbai without re-downloading the road graph and retraining the model, which is inconvenient but not fundamentally difficult. A deeper concern is that the labels come from the rule-based formula rather than from actual crime data or user-reported incidents. The framework measures proxy indicators of safety, such as people nearby, shops open, and anchors present, but it cannot confirm whether a location that appears safe by those proxies is truly safe. Validated pedestrian-safety datasets for Indian cities remain scarce, so this limitation is partly inherent to the problem domain itself. Four directions stand out for future work: (i) a mobile app with live GPS and turn-by-turn directions, since a Streamlit dashboard is not practical on a street corner; (ii) incorporating actual crime or incident records through a Bayesian model that treats the rule-derived scores as a prior and updates them with real data; (iii) adding a weather penalty, since rain empties sidewalks just as effectively as nightfall; and (iv) allowing users to specify any bounding box so that the system builds the graph on the fly, removing the restriction to a single city.

## 10. Conclusion

GuardianPath shows that Jacobs' half-century-old surveillance insight can be turned into a practical, time-aware routing system that runs on nothing but free OpenStreetMap data and a commodity CPU. Five interpretable features were defined, 12,000 labelled samples were generated through a rule-based formula, and an XGBoost regressor was trained that approximates the formula with an R2 of 0.9937. Embedding the model predictions into a modified Dijkstra search produces safety-optimised routes that differ meaningfully from the shortest path at night yet add negligible distance during the day. SHAP analysis shows that the model has learned temporally coherent representations, with guardian proximity mattering most at night and active-hour status mattering most during the day, which is consistent with the underlying theory. The result is an end-to-end system that routes, explains, and responds to time, all within a few seconds, without proprietary data or specialised hardware.

## Acknowledgement

The authors acknowledge the OpenStreetMap community for maintaining the geospatial data infrastructure that underpins this work. This research received no external funding.

## References

1. J. Jacobs, The Death and Life of Great American Cities, Random House, New York, 1961.
2. O. Newman, Defensible Space: Crime Prevention Through Urban Design, Macmillan, New York, 1972.
3. C.R. Jeffery, Crime Prevention Through Environmental Design, Sage Publications, Beverly Hills, 1971.

4. B. Hillier, J. Hanson, The Social Logic of Space, Cambridge University Press, Cambridge, 1984.
5. P.M. Cozens, G. Saville, D. Hillier, Crime prevention through environmental design (CPTED): A review and modern bibliography, Property Management, 23 (5), 328–356, 2005.
6. A. Laxmi, et al., POI-based safe route navigation using OpenStreetMap, Proc. IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bengaluru, India, 2022.
7. M. Haklay, P. Weber, OpenStreetMap: User-generated street maps, IEEE Pervasive Computing, 7 (4), 12–18, 2008.
8. G. Boeing, OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks, Computers, Environment and Urban Systems, 65, 126–139, 2017.
9. K. Jordahl, et al., geopandas/geopandas: v0.8.1, Zenodo, 2020.
10. A.A. Hagberg, D.A. Schult, P.J. Swart, Exploring network structure, dynamics, and function using NetworkX, Proc. 7th Python in Science Conference (SciPy), Pasadena, CA, USA, 11–15, 2008.
11. Z. Yin, L. Cao, Mining point-of-interest data from social networks for urban land use classification and disaggregation, Computers, Environment and Urban Systems, 53, 36–46, 2015.
12. A. Noulas, S. Scellato, R. Lambiotte, M. Pontil, C. Mascolo, A tale of many cities: Universal patterns in human urban mobility, PLoS ONE, 7 (5), e37027, 2012.
13. T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 785–794, 2016.
14. J.H. Friedman, Greedy function approximation: A gradient boosting machine, Annals of Statistics, 29 (5), 1189–1232, 2001.
15. L. Breiman, Random forests, Machine Learning, 45 (1), 5–32, 2001.
16. F. Pedregosa, et al., Scikit-learn: Machine learning in Python, Journal of Machine Learning Research, 12, 2825–2830, 2011.
17. A.J. Ratner, C.M. De Sa, S. Wu, D. Selsam, C. Re, Data programming: Creating large training sets, quickly, Proc. 30th Conference on Neural Information Processing Systems (NeurIPS), Barcelona, Spain, 3567–3575, 2016.
18. S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Proc. 31st Conference on Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4765–4774, 2017.
19. D. Quercia, R. Schifanella, L.M. Aiello, The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city, Proc. 25th ACM Conference on Hypertext and Social Media, Santiago, Chile, 116–125, 2014.
20. P. Mirowski, et al., Learning to navigate in cities without a map, Proc. 32nd Conference on Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 2419–2430, 2018.
21. S. Chainey, L. Tompson, S. Uhlig, The utility of hotspot mapping for predicting spatial patterns of crime, Security Journal, 21 (1–2), 4–28, 2008.
22. Y. Zheng, L. Capra, O. Wolfson, H. Yang, Urban computing: Concepts, methodologies, and applications, ACM Transactions on Intelligent Systems and Technology, 5 (3), 38, 2014.
23. J.L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM, 18 (9), 509–517, 1975.
24. E.W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik, 1 (1), 269–271, 1959.