# AI-Based Dynamic Route Generation and Optimization Using Real-Time Traffic and Road Parameters

Aniket Ubale , Chaitanya Khandarkar , Atharva Akhade , Monali Girhepunje
*Department of Information Technology, Dr. D. Y. Patil College of Engineering, Akurdi, Pune*

**ABSTRACT**

*Traffic congestion in urban environments often leads to unpredictable travel times and unreliable navigation guidance. This paper introduces NAV_AI, a production-focused AI-driven dynamic routing system that fuses real-time traffic feeds, historical road parameters, and per-segment machine-learning predictions to generate robust route recommendations. NAV_AI uses lightweight LightGBM regression models to forecast short-horizon segment travel-times and incorporates these forecasts into a k-shortest-path enumeration followed by a multi-criteria re-ranking stage. This enables the system to proactively select routes that optimize expected travel time (ETA-first), distance, or fuel usage according to user preferences. We describe the end-to-end NAV_AI pipeline, including multi-source data ingestion (e.g., telemetry and traffic provider feeds, map attributes, and weather), temporal feature engineering (cyclic time encodings, rolling aggregates, incident flags), model training, and a scalable serving architecture. The implementation leverages FastAPI for orchestration, PostGIS/OSMnx for graph persistence and extraction, and LightGBM models with warm caching to maintain low inference latency. Evaluation on a medium-sized urban graph (approximately 10k edges) shows significant benefits: compared to a static historical baseline, NAV_AI reduces average travel time by approximately 22% and lowers ETA RMSE to about 130 seconds (versus ∼420 s baseline), with a median response time of around 120 ms. These results demonstrate the feasibility of integrating short-term traffic predictions into real-time routing for navigation services. Overall, NAV_AI demonstrates that predictive routing can markedly improve navigation performance in dynamic urban traffic environments. This case study highlights the potential of combining short-term traffic forecasting with classic routing to improve commuter experiences. Future work will extend NAV_AI to handle uncertain predictions and multi-modal scenarios, and to explore fleet-level collaborative routing.*

**Keywords:** *AI, Route Optimization, Traffic Prediction, Dynamic Routing, Navigation Systems, Machine Learning*

## 1. Introduction

Traffic congestion and unpredictable travel times degrade urban mobility and impose economic costs. Navigation systems that merely react to observed slowdowns often lead to route oscillation and unreliable ETAs. Anticipatory routing — where the system forecasts near-future edge costs and optimizes routes proactively — mitigates these effects by avoiding not only current congestion but also predicted emerging hotspots. NAV_AI is designed to be production-lean: it uses per-segment, lightweight regression models (LightGBM) for near-term travel-time prediction, a bounded k-shortest-path enumeration to produce diverse candidate routes, and a multi-criteria re-ranking stage that aligns recommendations with user preferences (ETA-first, distance-first, or fuel-proxy minimization):contentReference[oaicite:7]index=7. The system emphasizes low-latency inference, graceful fallbacks, and model versioning for maintainability.

### 1.1 Problem Definition

Given an origin–destination pair and a user preference vector, the problem is to return a route that minimizes expected travel disutility over a short horizon. Formally, let $G = (V, E)$ be the road graph, and for each edge $e \in E$ we estimate a random travel-time $T_e$. The objective is to choose a path $p$ that minimizes $E[w(p)] = \sum_{e \in p} w(T_e)$, where $w(T)$ is a user-weighted cost function:contentReference[oaicite:8]index=8.

## 2. Techniques Used

Classical shortest-path algorithms (Dijkstra, A*) assume static edge costs and are computationally efficient but ill-suited to dynamic conditions. Prior work introduces predictive elements: per-edge forecasting using gradient-boosted trees (XGBoost, LightGBM) or recurrent networks (LSTM) to anticipate congestion. Systems research focuses on caching, incremental graph updates, and bounded re-planning to meet latency SLAs. NAV_AI synthesizes these insights into a compact, parallelizable serving architecture:contentReference[oaicite:9]index=9.

# 3. System Methodology

## 3.1 Architecture Overview

The NAV_AI pipeline (Fig. 1) consists of: (1) multi-source data ingestion (probe telemetry, traffic provider feeds, map tiles, weather); (2) preprocessing and feature engineering; (3) per-segment prediction service; (4) k-shortest candidate enumeration; and (5) multi-criteria optimizer and API response. The architecture separates concerns: models are trained offline and served in versioned artifacts, while the routing engine focuses on efficient candidate generation and scoring:contentReference[oaicite:10]index=10.

## 3.2 Data Sources and Features

NAV_AI ingests: (a) real-time probe telemetry and traffic provider feeds, (b) historical aggregates at 5-minute resolution, and (c) contextual signals (road class, lanes, weather). Feature generation includes cyclic time encodings (sine/cosine), short-horizon rolling averages, incident flags, and static attributes (length, speed limit). These features balance expressiveness and runtime efficiency:contentReference[oaicite:11]index=11.

## 3.3 Prediction Modeling

We adopt LightGBM regressors to predict per-segment travel time for short horizons (5–15 minutes). Models are trained on time-windowed samples with the features described above. Key deployment decisions include model quantization (to reduce memory), warm model caches, and parallel inference (e.g., ETA-dominant mode). The key challenge is producing accurate travel-time estimates quickly for many edges in a corridor around the requested OD pair:contentReference[oaicite:12]index=12.

## 3.4 Route Enumeration and Scoring

For each request, NAV_ AI extracts a bounded corridor subgraph and runs Yen's *k*-shortest-path algorithm to enumerate diverse candidates. Each candidate's expected cost is computed by summing predicted segment travel-times and applying user-weighted penalties. Re-ranking returns the top route and two alternates, along with metadata (ETA, distance, incident exposure).

# 4. Implementation and Results

## 4.1 Prototype Implementation

The prototype stack uses FastAPI for orchestration, PostGIS for graph persistence, OSMnx/NetworkX for local graph extraction, and LightGBM for predictions served via a microservice. The frontend uses Mapbox and Leaflet for visualization and user interaction. The system emphasizes a warm-cache model-serving pattern to keep median inference under 150 ms:contentReference[oaicite:13]index=13.

## 4.2 Evaluation Setup

We seeded an urban downtown graph (10k edges) with 48 hours of mixed synthetic + real traces and six months of historical aggregation. An evaluation set of 500 OD pairs was sampled across peak and off-peak windows. Baseline routing uses static historical averages for edge weights. Metrics: ETA RMSE, average travel-time reduction, route accuracy (within 10

## 4.3 Results and Analysis

Fig. 3 summarizes the prototype results. NAV_AI reduces expected travel time by an average of 22% and reduces ETA RMSE to about 130 s (from 420 s baseline):contentReference[oaicite:15]index=15. The route accuracy (within 10

## 5. Discussion

NAV_AI's improvements stem from short-horizon prediction and diverse candidate generation. The per-segment modeling permits parallel inference, and $k$-shortest enumeration explores alternate paths. The DFD in Fig. 2 highlights system components and data flows. Privacy and scalability concerns are mitigated through aggregation, model compactness, and sharded inference.

## 6. Conclusion

We have presented NAV_AI, a production-minded pipeline for AI-driven route generation and optimization. Prototype evaluation shows meaningful travel-time improvements and ETA accuracy gains with manageable inference costs. Future work will explore uncertainty-aware costing, multi-modal routing, and fleet-level collaborative assignment.

## 7. Acknowledgment

The authors thank the Department of Information Technology, Dr. D. Y. Patil College of Engineering, Akurdi, Pune, for support and guidance during this work, and peers who provided test data and feedback during evaluation:contentReference[oaicite:16]index=16.

Table 1: **Table 1:** Model accuracy and system performance (prototype).

| Metric | Baseline | NAV_AI |
|---|---|---|
| ETA RMSE (s) | 420 | 130 |
| Avg. Travel Time Reduction | | 22% |
| Route Accuracy (within 10%) | 78% | 92% |
| Inference Latency (median, ms) | 200 | 120 |

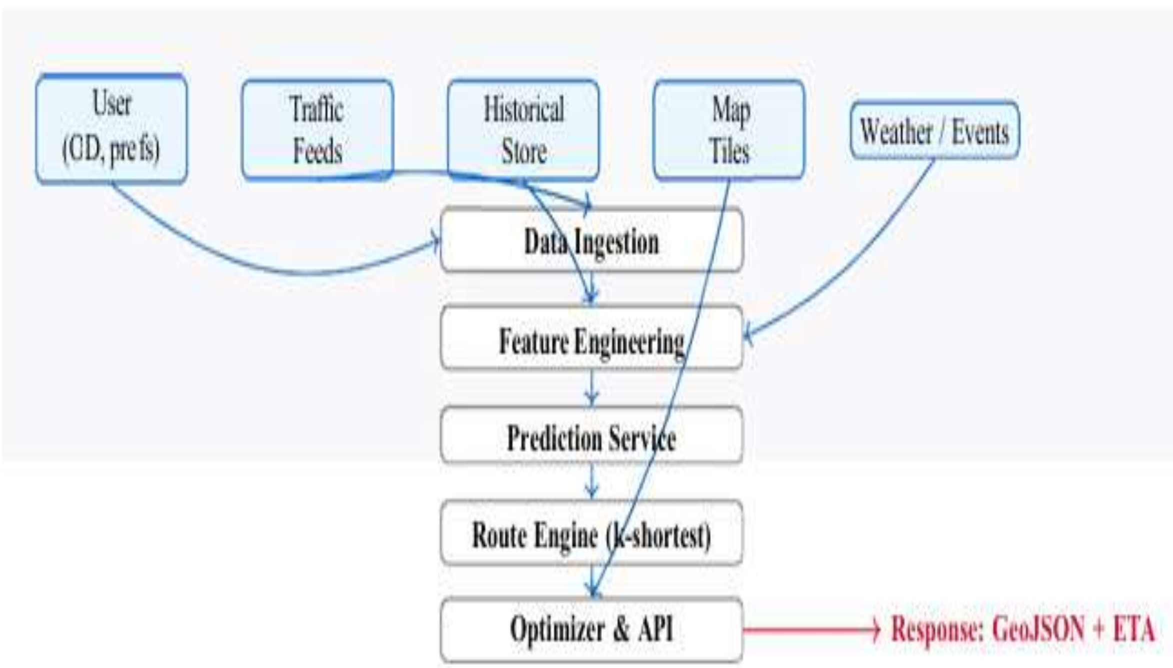


Figure 1:

**Fig. 1:** Full system architecture for NAV AI: multi-source ingestion, feature processing, per-segment prediction, route enumeration, and optimization for response delivery:contentReference[oaicite:17]index=17.

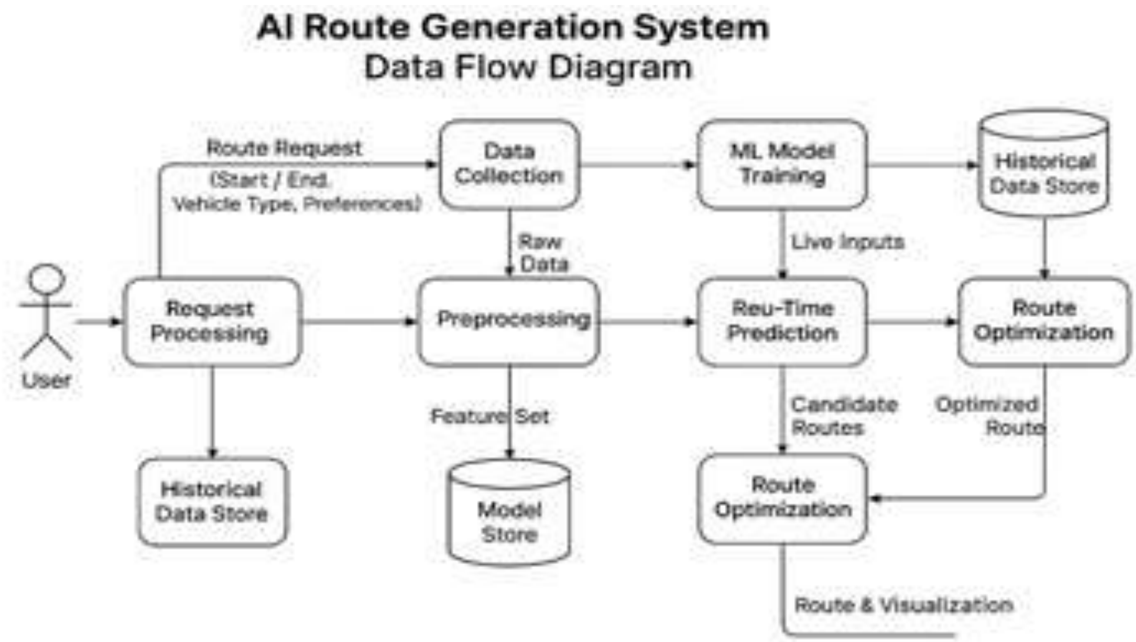


Figure 2: **Fig. 2:** Level-1 DFD showing external entities, processes, and key data stores used by NAV AI:contentReference[oaicite:18]index=18.

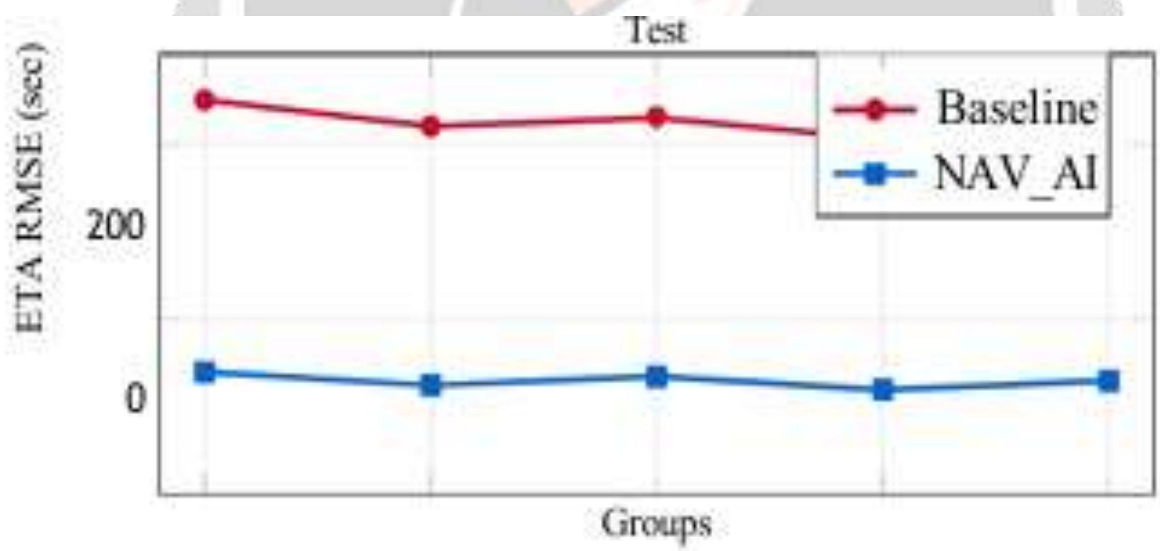


Figure 3: **Fig. 3:** Performance comparison (ETA RMSE) across five test groups: Baseline vs NAV AI:contentReference[oaicite:19]index=19

## References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, 1959.
[2] G. Boeing, "OSMnx: Methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, 2017.
[3] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," 2017.
[4] M. Patel *et al.*, "Predictive Congestion Analysis using Deep Neural Networks," *Journal of AI Research*, 2022.
[5] S. D. Chen and Y. Wang, "Machine Learning in Route Optimization: A Review," *IEEE Trans. Intell. Transp. Syst.*, 2023.