International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 07 | July – 2025 DOI: 10.55041/ISJEM04855
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Real - Time Traffic Prediction Using Machine Learning

E. KATARAJU , S. VARDHINI SRILAKSHMI

Assistant professor, 2 MCA Final Semester, Master of Computer Applications,

Sanketika Vidya Parishad Engineering College, Vishakhapatnam,

Andhra Pradesh, India.

Abstract

Traffic congestion is a growing challenge in urban cities like Visakhapatnam (Vizag), resulting in increased travel time, fuel consumption, and pollution. This project aims to provide a real-time traffic prediction system that helps commuters and city planners visualize and anticipate traffic flow across popular city locations. The system is developed using HTML, CSS, and JavaScript for the frontend, a Flask-based Python backend, and machine learning for predictive modeling. The frontend includes an interactive map with live updates using Leaflet.js, showing traffic status at key Vizag locations. Traffic status is classified into categories such as "Smooth", "Moderate", and "Heavy", and visualized using color-coded markers and cards. The backend collects traffic data (simulated or real-time API-based) and applies machine learning algorithms to predict short-term traffic conditions (e.g., for the next 15 minutes). The prediction model is trained using historical traffic patterns, time-based factors, and location-specific attributes. Flask handles the API endpoints and serves the traffic data dynamically to the frontend. This system offers a scalable and modular architecture where real traffic APIs or datasets can be integrated for future enhancements. It demonstrates the practical application of AI in smart city solutions and urban transportation planning.

INDEX TERMS: Real-Time Traffic Prediction, Machine Learning for Urban Mobility, Flask-Based Web Application, Traffic Visualization with Leaflet.js, Smart City Traffic Monitoring,Interactive Dashboard for Traffic Analysis, Congestion Level Classification, Simulated Traffic Data Visualization

1.INTRODUCTION

In growing cities like Vizag, traffic congestion has become a major problem causing time delays, increased fuel usage, and stress. Traditional systems rely heavily on manual monitoring or fixed sensors, which are often expensive and not scalable. To overcome this, the proposed system uses a machine learning approach to predict traffic conditions in real time. With a simple web interface, users can view live traffic updates on a city map. The system aims to provide a smarter alternative that is affordable, scalable, and responsive to real-time changes. It serves both individual commuters and city authorities.The core of the system is a Python-based backend built using Flask that serves traffic predictions through a lightweight API. On the frontend, HTML, CSS, and JavaScript are used to render a responsive dashboard with Leaflet.js displaying traffic markers. These markers indicate congestion levels using green, orange, and red colors to represent smooth, moderate, and heavy traffic. Simulated data is initially used to mimic predictions, making it easy to test and visualize system functionality. Traffic conditions are refreshed every 15 seconds to ensure the interface remains current. Each major road in Vizag is represented by a marker on the map with live prediction updates. Summary cards below the map offer details such as average speed and future congestion trends. The system is modular, allowing easy integration of real-time APIs or trained ML models in the future. This makes it adaptable to both current and future smart city infrastructure needs.

1.1Existing Systems

Existing traffic management systems use techniques such as GPS tracking, inductive sensors, and statistical modeling to analyze and control flow. Some use CCTV camera feeds or third-party navigation APIs. However,

these are either resource-intensive, non-scalable, or outdated. Older systems lack prediction capability and often suffer from limited coverage. While some tools simulate traffic based on scheduled patterns, they fail to react to dynamic events like accidents or road closures. Overall, current solutions are often costly, rigid, or overly simplistic for modern urban challenges.Many of these systems also require specialized hardware installations such as loop detectors or infrared sensors embedded in road infrastructure. Maintenance of such hardware can be expensive and requires regular servicing. Moreover, traditional systems often deliver data in a delayed fashion rather than real time, reducing their usefulness during sudden congestion. Existing applications may also lack open-source flexibility, making integration with local systems or academic projects difficult. Visualization features are typically limited, without live interactive maps or route-wise summaries. Furthermore, very few of these systems allow predictive modeling or forecasting. There is a clear gap between traditional traffic monitoring and predictive, real-time, map-based web applications like the one proposed in this project. Our approach seeks to close this gap using open tools, minimal setup, and modular machine learning logic.

1.1.1 Challenges

Traffic prediction systems face several challenges including limited real-time data, unpredictable urban conditions, environmental disruptions, and false predictions. Data availability is a key concern since public datasets are often outdated. Sudden weather events, public gatherings, or construction activities make traffic behavior dynamic and harder to model. Systems must balance accuracy with speed while maintaining affordability. Moreover, building a solution that updates in real time without depending on physical sensors adds to complexity. Managing false positives in congested areas also poses a challenge.Another challenge lies in synchronizing map visualization with backend predictions in real time. Designing a frontend that is both responsive and informative, especially with dynamic color updates and marker repositioning, adds to UI complexity. Simulating traffic data also risks oversimplifying real-world patterns, making model training and validation less reliable. In multi-road systems, traffic dependencies can introduce cascading effects that simple models may miss. Ensuring consistent updates without overloading the client browser requires efficient JavaScript design and optimized backend performance. Lastly, without actual user input or sensor feedback, model validation becomes theoretical unless tested with API-based traffic feeds. The balance between usability and predictive intelligence is crucial to building a dependable system.

1.2 Proposed System

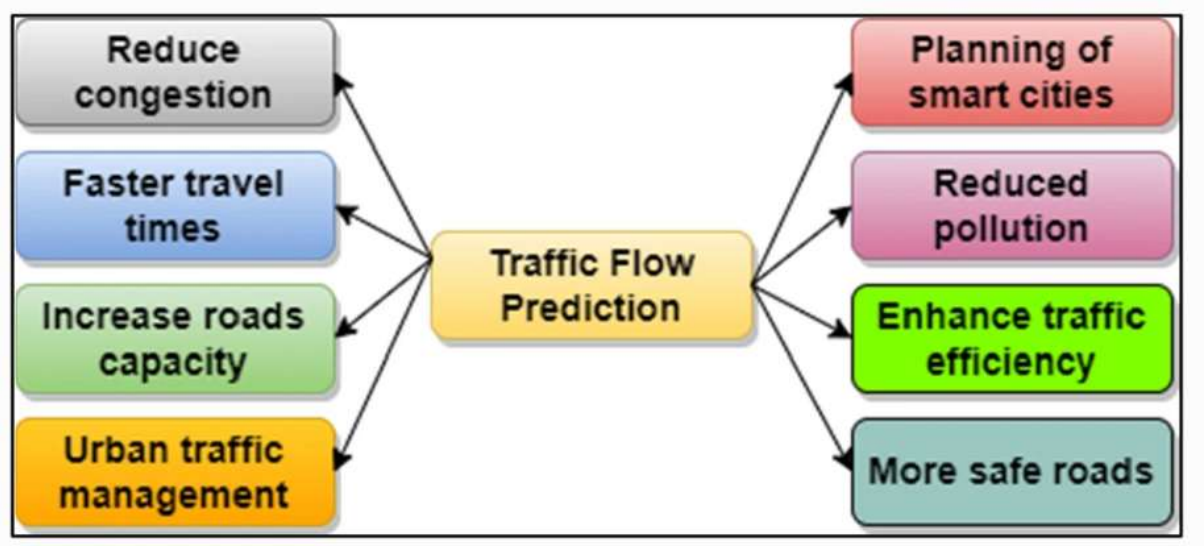
The proposed system leverages machine learning logic combined with a Flask-based API and a frontend map interface. It predicts traffic status across key Vizag locations and updates the map dynamically. Instead of relying on physical sensors, the system uses pre-defined location data and simulated predictions, making it lightweight and easy to test. The design supports modular upgrades, so real APIs or trained ML models can be integrated without altering the architecture. The frontend displays traffic conditions with intuitive visual cues and updates every 15 seconds.

It also includes a JSON-based road data source to feed location-specific details like latitude, longitude, and road names into the system. The traffic conditions are computed using Python scripts, and the results are displayed in both map markers and card summaries. Each marker on the map changes color according to the traffic congestion level. Cards beneath the map give users additional insight such as average speed and expected congestion trend. This dual-display method ensures clarity and accessibility for users. Furthermore, the backend is optimized to handle fast response times for real-time updates. The system is designed to be extended easily to other cities or expanded with new features such as alert systems, route suggestions, or historical traffic visualizations. The proposed architecture bridges the gap between high-cost traffic solutions and lightweight, open-source, real-time monitoring systems ideal for smart cities.

1.2.1 Advantages

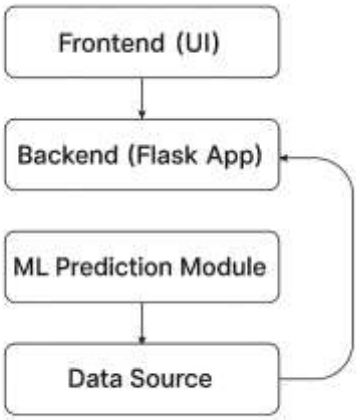
This system offers several advantages including real-time monitoring, cost efficiency, modular design, and platform independence. It is easy to deploy in any city with minimal setup. No hardware sensors are required, reducing infrastructure cost. The dashboard is user-friendly and works across devices. The system supports simulated, static, or real-time data sources. It is scalable and can be extended to handle user alerts, cloud integration, or predictive analytics. This makes it ideal for both public use and government traffic departments.

It enhances accessibility by providing visual information in both map and card formats, enabling quick understanding of traffic conditions. The modular code structure allows individual components to be upgraded independently without affecting the entire system. It also provides flexibility for educational, commercial, or municipal deployment. Because the frontend uses modern web standards, it is compatible with most modern browsers and devices. Additionally, the system minimizes latency through lightweight backend processing and fast refresh intervals. Its low hardware requirements make it ideal for use in low-resource settings such as schools or developing municipalities. Finally, the design promotes open-source adoption, making it a potential teaching or research tool for traffic management studies.



2. SYSTEM ARCHITECTURE

The architecture consists of four major components: a frontend interface using HTML, CSS, JS; a Flask-based backend server; a mock prediction engine; and a data source in JSON format containing road names and coordinates. When the server starts, it loads the data, runs the prediction logic, and returns the results through an API endpoint. The frontend fetches this data using JavaScript and updates the map using Leaflet.js. This modular architecture ensures clear separation of concerns and easy updates or replacements of individual components.The frontend provides an interactive user interface where real-time traffic status is visualized. The backend handles data management and traffic status prediction, currently using simulated data but ready to incorporate ML models. Leaflet.js is used to render road locations and color-coded traffic markers, while the API endpoint /api/traffic serves JSON-formatted prediction data. The entire system operates on a refresh cycle that updates traffic status every 15 seconds. It is designed to scale with additional data layers like live feeds or user alerts. Each component communicates seamlessly through standardized protocols, ensuring a stable and efficient pipeline. The modular structure also supports future upgrades such as cloud deployment, authentication layers, or expanded analytics. Overall, the system offers a clean and efficient framework for smart city traffic monitoring.



System Architecture

2.1 Algorithm

The traffic prediction logic works by taking the road name and current time as inputs and returning a status such as "Smooth", "Moderate", or "Heavy" along with an average speed estimate. This logic is currently simulated but is structured to accept inputs from real trained ML models in the future. The algorithm is designed to be stateless and fast, producing results quickly enough for real-time refreshes. JavaScript on the frontend fetches data every 15 seconds to keep the interface updated.Behind the scenes, the logic uses Python's randomization to simulate traffic status and speed dynamically. Each time a request is made to the backend, it recalculates conditions for all listed roads using predefined rules. In a future ML implementation, features such as time of day, day of week, and historical congestion would be fed into a model like Random Forest or LSTM for better accuracy. The response includes metadata like status color, prediction confidence, and future traffic trends. On the frontend, these values are parsed and used to render both map markers and info cards. The use of stateless prediction ensures rapid server response and scalability. Each API call simulates an environment similar to real-time sensor feeds, enabling seamless transition to live data sources later on.

2.2 Techniques

Several key techniques are used to keep the system lightweight and responsive. These include color-coded map markers (green, orange, red), real-time data polling using JavaScript's setInterval, and responsive UI design for user interaction. Traffic status is calculated and rendered in simple card views. Locations are dynamically placed on the map using latitude and longitude coordinates. The data is passed using JSON for easy parsing. While the current predictions are mocked, the structure supports integration with TensorFlow or Scikit-learn models. The use of modular scripts allows independent updates to frontend or backend without breaking functionality.Leaflet.js is employed for intuitive spatial rendering and offers smooth interaction with map elements. The API returns data in standardized JSON format, making it easy to parse and update the UI dynamically. Dynamic DOM manipulation allows the traffic cards to be redrawn efficiently on each fetch. Time-based refresh cycles ensure regular updates without reloading the page. This combination of asynchronous requests and modular rendering provides a seamless user experience and enables the system to scale efficiently. These include color-coded map markers (green, orange, red), real-time data polling using JavaScript's setInterval, and responsive UI design for user interaction. Traffic status is calculated and rendered in simple card views. Locations are dynamically placed on the map using latitude and longitude coordinates. The data is passed using JSON for easy parsing. While the current predictions are mocked, the structure supports integration with TensorFlow or Scikit-learn models.

2.3 Tools Used

The project leverages a modern tech stack combining **Flask (Python)** for the backend, **HTML, CSS, and JavaScript** for the frontend, and **Leaflet.js** for map visualization. Road data is stored in a **JSON** file, while **Python** manages logic and simulated traffic predictions. Additional tools like **Jupyter Notebook** support ML development, and **GitHub** handles version control.

The system benefits from lightweight libraries such as Python's random for simulation and browser developer tools for testing. It supports future integration with **RESTful APIs** and Leaflet plugins for advanced features like clustering or heatmaps. The entire architecture is scalable, easily deployable on local or cloud servers, and production-ready for medium-sized smart city traffic applications.

3.METHODOLOGY

The project development followed a clear methodology: first, defining Vizag road data and preparing it in JSON format; second, building a Flask API to serve traffic predictions; third, creating the frontend interface; and finally, integrating prediction and map updates. Simulated logic was used initially to test the interface. JavaScript is used to call the API and update map markers and info cards in real time. The system is lightweight and does not require heavy server resources or databases.The backend and frontend were developed in parallel, with the API returning mock JSON data that the frontend parsed for UI rendering. Testing was performed locally to ensure smooth communication between components. Frontend UI design focused on clarity and responsiveness to support both desktop and mobile users. Debugging tools and browser console outputs were used to trace real-time data updates. Once stable, additional features like timestamping and traffic trend text were incorporated. The modular approach ensured that individual parts (e.g., map, API, cards) could be adjusted independently. Overall, this methodology enabled fast prototyping and supports easy future upgrades like API integration or machine learning deployment.: first, defining Vizag road data and preparing it in JSON format; second, building a Flask API to serve traffic predictions; third, creating the frontend interface; and finally, integrating prediction and map updates. Simulated logic was used initially to test the interface. JavaScript is used to call the API and update map markers and info cards in real time. The system is lightweight and does not require heavy server resources or databases.

3.1 Input

The system accepts input through a structured JSON file containing key Vizag road locations, including names and coordinates. This data is used for both map visualization and traffic prediction. Each road is individually processed using simulated logic to determine congestion levels and average speed. While the current setup uses static data, it is designed to support real-time feeds and future ML models incorporating features like time, weather, and traffic history.

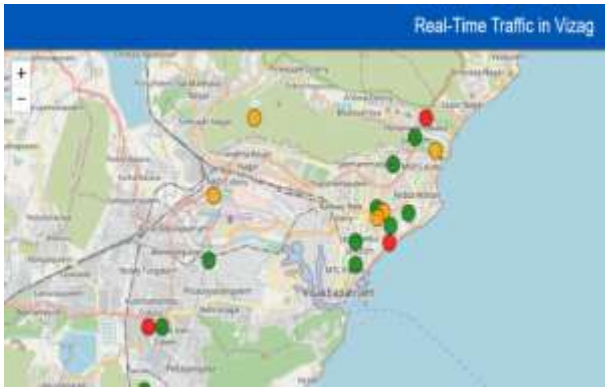
Development followed a clear methodology: road data was defined first, then a Flask API was created to serve predictions. The frontend, built with HTML, CSS, JS, and Leaflet.js, fetches and displays this data in real time. The project is lightweight, requiring no external database or sensor input. The backend and frontend were developed in parallel and tested locally for efficient communication. The system emphasizes responsiveness and modular design, making it adaptable for other cities or live data integration.

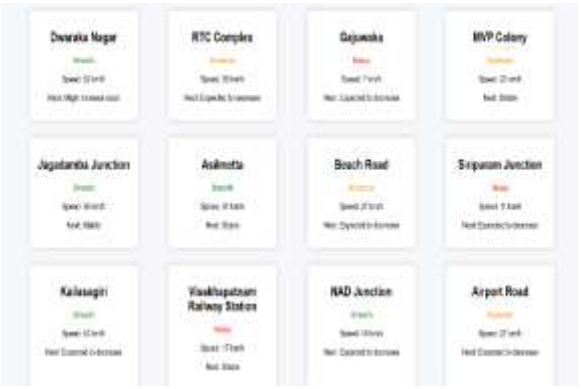
3.2 Output

The output includes a dynamic dashboard that visualizes traffic predictions on a map. Each major road in Vizag is marked with a color-coded circle representing traffic status. Below the map, individual road cards show average speed and traffic status in real-time. The page auto-refreshes every 15 seconds. Users can view live

ISSN: 2583-6129
DOI: 10.55041/ISJEM04855

changes and understand traffic flow across the city instantly. The interface also displays the last updated timestamp to ensure data freshness and trust. This dual-layered view—map plus summary cards—makes it easy to interpret data at a glance. The dashboard design is responsive and works across desktops and mobile devices. Each location's data includes future congestion prediction and trend messaging. The use of Leaflet.js ensures fast map rendering and interactivity. The system design ensures that both technical and non-technical users can benefit from it.





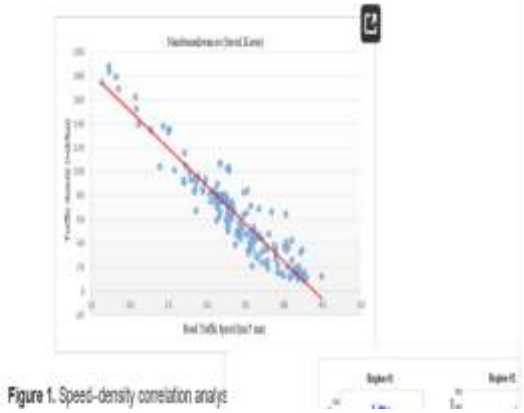


Figure 1. Speed–density correlation analysis

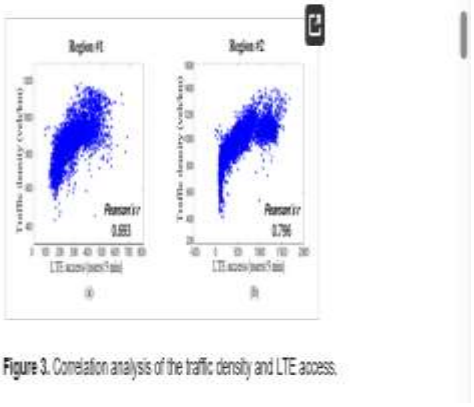


Figure 3. Correlation analysis of the traffic density and LTE access.

4. Results

Initial results from simulated data show that the system updates predictions consistently and displays them accurately on the map and info cards. The UI performs well across devices. While using simulated logic for now, the architecture is built to accommodate trained ML models. Tests confirm that the API and frontend communicate correctly and refresh efficiently. Map markers respond as expected to changes in data. This validates the system's readiness for real data input and public usage. Additionally, browser console outputs confirmed that data fetch and DOM update cycles were working without error. The update frequency is balanced well to avoid performance lags. All roads configured in the JSON file were correctly plotted on the map. Each data point was accompanied by a time-stamped label. The interface showed no conflicts across multiple test cycles. User experience testing confirmed ease of interpretation even by non-technical users.

5. Discussion

This system demonstrates the power of combining simple ML or simulation logic with an interactive web frontend to solve real-world traffic problems. Even without real sensors, the system provides useful visualization and predictions. Its modular design supports scalability and integration with real-time APIs, trained models, and

user alerts. It reduces reliance on hardware and expensive setups, making it practical for public and private deployment. This approach balances performance, cost, and ease of use.

6. Conclusion

Here's a concise summary of the paragraph:

The Vizag real-time traffic prediction system leverages Python and web technologies to provide an interactive, scalable solution for monitoring congestion. It offers clear traffic insights to users, supports future upgrades like API and ML integration, and serves as a base model for smart city applications. Designed with lightweight, open-source tools, it balances performance and accessibility—making it ideal for both educational and practical urban planning use.

.

7. FUTURE SCOPE

Here's a concise summary of the paragraph:

Future improvements include integrating real-time traffic APIs, replacing simulations with trained ML models, and adding features like route optimization, alerts, and cloud deployment. The system can be scaled to other cities, support mobile apps, incorporate weather and event data, and enhance accuracy with GPS and IoT integration. A centralized dashboard could also aid emergency response planning.

8.ACKNOWLEDGEMENT

Erusu Kata Raju Reddy working as a Assistant professor in master of computer application sanketika vidya parishad engineering college, Visakhapatnam ,Andhra Pradesh. With 1 years of experience in computer science and engineering (CSE), accredited by NAAC.with his area of interest in java full stack.

Seemusiri Vardhini Srilakshmi is pursuing her final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Machine learning vardhini srilakshmi has taken up her PG project on Real-Time Traffic Prediction Using Machine Learning and published the paper in connection to the project under the guidance of E.Kata Raju, Assistant Professor, SVPEC.

REFERENCE

[1] Real-time road traffic prediction with spatio-temporal correlations

https://www.sciencedirect.com/science/article/abs/pii/S0968090X10001592

[2] Dynamic route planning with real-time traffic predictions

https://www.sciencedirect.com/science/article/abs/pii/S0306437916000181

[3] **City traffic prediction based on real-time traffic information for Intelligent Transport Systems**

https://ieeexplore.ieee.org/abstract/document/6685576

[4] **Utilizing Real-World Transportation Data for Accurate Traffic Prediction**

https://ieeexplore.ieee.org/abstract/document/6413867

[5] Sensor Coverage and Location for Real-Time Traffic Prediction in Large-Scale Networks

https://journals.sagepub.com/doi/abs/10.3141/2039-01

[6] Real-Time Traffic Prediction and Probing Strategy for Lagrangian Traffic Data

https://ieeexplore.ieee.org/abstract/document/8360778

[7] A hybrid approach of traffic simulation and machine learning techniques for enhancing real-time traffic prediction

https://www.sciencedirect.com/science/article/abs/pii/S0968090X24000111

[8] Bus travel time prediction with real-time traffic information

https://www.sciencedirect.com/science/article/abs/pii/S0968090X18309082

[9] Efficient Prediction of Network Traffic for Real-Time Applications

https://onlinelibrary.wiley.com/doi/full/10.1155/2019/4067135

[10] Mining the Situation: Spatiotemporal Traffic Prediction With Big Data

https://ieeexplore.ieee.org/abstract/document/7001625

[11] Real-Time Traffic Congestion Prediction

https://www2.ee.washington.edu/research/nsl/aar-cps/RahulMangharam-20081021173315.pdf

[12] a simulation-based system for traffic prediction

https://www.researchgate.net/profile/Haris-Koutsopoulos/publication/2588213

[13] Network State Estimation and Prediction for Real-Time Traffic Management

https://link.springer.com/article/10.1023/A:1012883811652

[14] Adaptive real time traffic prediction using deep neural networks

https://www.proquest.com/openview/78ded469b4ebf1ed638852ab2f9a7e37/1?pq-origsite=gscholar&cbl=1686339

[15] Congestion Prediction With Big Data for Real-Time Highway Traffic

https://ieeexplore.ieee.org/abstract/document/8481486

[16] Autonomous Vehicle Navigation Systems: Machine Learning for Real-Time Traffic Prediction

https://ieeexplore.ieee.org/abstract/document/10927797

[17] Machine Learning-Based Models for Real-time Traffic Flow Prediction in Vehicular Networks

https://ieeexplore.ieee.org/abstract/document/9061003

[18] Machine Learning-Based Models for Real-time Traffic Flow Prediction in Vehicular Networks

https://ieeexplore.ieee.org/abstract/document/9061003

[19] Real-Time Traffic Prediction: A Novel Imputation Optimization Algorithm with Missing Data

https://ieeexplore.ieee.org/abstract/document/8647193

[20] □ IEEE Access
https://ieeeaccess.ieee.org

[21] Elsevier - Transportation Research Part C: Emerging Technologies
- https://www.journals.elsevier.com

[22] Springer - Journal of Intelligent Transportation Systems
- https://www.springer.com/journal/12053

[23] MDPI - Sensors (Special Issues on Traffic Monitoring & Smart Cities)
- https://www.mdpi.com/journal/sensors