Traffic-Sense: Harnessing Interpretable Machine Learning to Revolutionize Urban Traffic Congestion Prediction

Project Team
Group 26

Sl. No.	Reg. No.	Student Name
1	20ETCS002004	Aditya Kumar
2	20ETCS002007	Aishwarya V
3	20ETCS002141	Syed Khubaib Shaheen
4	20ETCS002147	Utkarsh Singh Bharadwaj

Supervisor: Dr. Yogesh K

B. Tech. in Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY

M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Bengaluru -560 054

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

This is to certify that the Project title ***"Traffic-Sense: Harnessing Interpretable Machine Learning to Revolutionize Urban Traffic Congestion Prediction****" is a bonafide work carried out in the Department of Computer Science and Engineering by Aditya Kumar, Aishwarya V, Syed Khubaib Shaheen, Utkarsh Singh Bharadwaj, bearing Reg. Nos 20ETCS002004, 20ETCS002007, 20ETCS002141, 20ETCS002147 respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

December 2023

Dr. Yogesh K M
(Supervisor)
Assistant Professor

Dr. Rinki Sharma
Professor and Head – Dept. of CSE

Dr. Jagannathrao Venkatrao Desai
Professor and Dean - FET

Declaration

Traffic-Sense: Harnessing Interpretable Machine Learning to Revolutionize Urban Traffic Congestion Prediction

The project work is submitted in partial fulfillment of academic requirements for the award of B. Tech. Degree in the Department of Computer Science and Engineering of the Faculty of Engineering and Technology of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of university regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

Sl. No.	Reg. No.	Student Name	Signature
1	20ETCS002004	Aditya Kumar	
2	20ETCS002007	Aishwarya V	
3	20ETCS002141	Syed Khubaib Shaheen	
4	20ETCS002147	Utkarsh Singh Bharadwaj	

Date: 28th December 2023

Acknowledgements

It is with extreme pleasure and pride that we present our B-Tech. dissertation “Traffic-Sense: Harnessing Interpretable Machine Learning to Revolutionize Urban Traffic Congestion Prediction”. We would like to express our sincere thanks and gratitude to the following people, who stood by us throughout, helping us with much required inputs, guidance, knowledge and supported us. We take great pleasure to express our sincere thanks and gratitude to academic project guide Dr. Yogesh K M Asst. Professor Department of CSE, for his support, guidance and suggestions throughout the project which is leading this project for the completion.

We express our sincere thanks to, Dr Jagannathrao Venkatrao Desai, our respected Dean and to Dr. Rinki Sharma, Head of Department of Computer Science and Engineering, for their kind cooperation and support toward out dissertation, and to the management of Ramaiah University of Applied Science for their continued support. We are thankful to the staff members of the Computer Science and Engineering, RUAS for giving us good support and suggestion.

Lastly, we would like to thanks our parents and friends for their continued support, encouragement and motivation and God for paving our way of success in this object.

Abstract

This project deals with traffic prediction that can be done in intelligent transportation systems which involve the prediction between previous year's data set and the recent year data which ultimately provides the accuracy and mean square error. This prediction will be helpful for the people who are in need to check the immediate traffic state. The traffic data is predicated on a basis of 1 hour time gap. Also, Live statistics of the traffic is analyzed from this prediction which makes it easier to analyze. The system compares the data of all roads and determines the most populated roads of the city.

The project employs a diverse set of interpretable Machine learning techniques, ensuring transparency and comprehensibility in predicting congestion. Through extensive feature engineering and data preprocessing, we harness the potential of various urban data sources, including traffic flow, weather conditions, and special events. Our models provide actionable insights for urban planners, traffic management authorities, and commuters alike. This report not only outlines the technical aspects of our machine learning models but also emphasizes the importance of transparency in fostering public trust. As urban areas continue to grow, "Traffic Sense" serves as a crucial tool for enhancing the efficiency of transportation systems, ultimately contributing to the development of smarter and more liveable cities.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction

The ever-growing pace of urbanization has transformed cities into dynamic hubs of activity, but with this transformation comes a critical concern—urban congestion. As city populations surge and transportation systems evolve, the need for effective strategies to predict, understand, and alleviate congestion becomes paramount. Enter the "Traffic Sense" project, a forward-looking initiative leveraging interpretable machine learning models to unravel the intricacies of urban mobility.

Urban congestion poses a significant challenge to the efficiency and sustainability of modern cities. In response to this, the "Traffic Sense" project leverages the power of interpretable machine learning models to predict and understand urban congestion patterns. Congestion hinders efficient transportation, limiting access to essential services and hindering economic growth. Heavy traffic congestion can also result in road safety hazards, increasing the risk of accidents and injuries. Addressing traffic congestion is crucial for enhancing urban mobility, reducing environmental impact, and improving overall quality of life in cities. Accurate prediction of traffic congestion can aid in proactively planning traffic management strategies and optimizing transportation systems.

Beyond the inconvenience faced by commuters, congestion poses broader challenges, affecting economic productivity, environmental sustainability, and overall urban well-being. The "Traffic Sense" project acknowledges these challenges and seeks to redefine the way we approach urban congestion. By employing interpretable machine learning models, the project not only aims to provide accurate predictions but also prioritizes transparency, ensuring that stakeholders, from urban planners to the general public, can comprehend the intricacies of traffic dynamics.

This introduction sets the stage for a transformative journey toward smarter, more resilient cities. The "Traffic Sense" project endeavours to empower decision-makers with insights that transcend traditional traffic prediction models, paving the way for a more sustainable and seamless urban future. As we delve into the nuances of urban congestion, the "Traffic Sense" initiative stands as a beacon of innovation, pointing towards a future where cities are not just larger but also more intelligent and responsive in managing their intricate traffic landscapes. Some of the challenges

Urban congestion poses a significant challenge to the efficiency and sustainability of modern cities. In response to this, the "Traffic Sense" project leverages the power of interpretable machine learning models to predict and understand urban congestion patterns. Congestion hinders efficient transportation, limiting access to essential services and hindering economic

growth. Heavy traffic congestion can also result in road safety hazards, increasing the risk of accidents and injuries. Addressing traffic congestion is crucial for enhancing urban mobility, reducing environmental impact, and improving overall quality of life in cities. Accurate prediction of traffic congestion can aid in proactively planning traffic management strategies and optimizing transportation systems.

1.2 Scope

The "Traffic Sense" project encompasses a comprehensive exploration of urban congestion prediction using interpretable machine learning models. The scope of this endeavour extends across multiple dimensions, aiming to address key aspects of urban mobility and congestion management. The project's primary objectives and scope include:

1. **Prediction Accuracy:** Develop and implement interpretable machine learning models with a focus on accuracy in predicting urban congestion. Evaluate and refine these models to ensure robust performance in diverse urban scenarios.

2. **Transparency and Interpretability:** Emphasize the interpretability of machine learning models, enabling stakeholders to understand the factors influencing congestion predictions. This transparency is vital for building trust among urban planners, traffic management authorities, and the general public.

3. **Data Integration:** Explore and integrate diverse urban data sources, including traffic flow data, weather conditions, and special events, to enhance the predictive capabilities of the models. Assess the impact of various factors on congestion dynamics.

4. **User-Friendly Interface:** Develop an intuitive and user-friendly interface to present congestion predictions and insights. Ensure that the interface caters to the needs of diverse users, including urban planners, traffic managers, and the general public.

5. **Scalability:** Design the project with scalability in mind to accommodate the evolving dynamics of urban environments. Consider the potential expansion of the project to cover larger geographical areas and increasing data volumes.

6. **Public Engagement:** Implement strategies for public engagement and awareness, fostering a sense of community involvement in congestion management. This could include interactive platforms, educational campaigns, and feedback mechanisms.

7. **Collaboration Opportunities:** Explore opportunities for collaboration with local authorities, transportation agencies, and academic institutions to leverage expertise, share data, and enhance the overall impact of the project.

By addressing these facets, the "Traffic Sense" project aspires to provide a holistic and impactful approach to urban congestion prediction, contributing to the development of smarter and more efficient cities.

1.3 Literature Survey

A critical and thorough examination of previously published studies and academic papers is called a literature review. It gives a broad review of the state of knowledge at the moment, points out areas where research is lacking, and creates the framework for a new investigation or project. The insights gained from the literature review underscore the relevance of the "Traffic Sense" project in the contemporary discourse on urban congestion prediction. By drawing on the strengths of interpretable machine learning models, the project seeks to address the identified gaps in understanding and mitigating congestion, contributing to the advancement of data-driven solutions in urban planning and transportation management.

Table 1: Literature Survey

Sl. No.	Author(s)	Journal Name and Year of Publication	Research Focus	Conclusions Derived via Authors	Limitations in the Study
1	Xiaoming Yuan	Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction	Federated Deep Learning based on FedSTN for Traffic Flow Prediction	Captures long-term spatial-temporal information, shares short-term hidden information, captures semantic features. Simulations demonstrate effectiveness.	Federated learning's reliance on constant information exchange between local models and the central server may lead to increased communication overhead, affecting system efficiency, particularly in scenarios with limited network bandwidth.

2	MedinaSalgado, Boris, Eddy SanchezDelaCruz	Urban traffic flow prediction techniques: A review	Smart techniques for analyzing mobility data for urban traffic flow prediction	Emphasizes the impact of urban population growth in Latin America on traffic problems. Advocates for intelligent transport systems using IoT and algorithms.	The temporal aspects of traffic flow prediction, such as the influence of time of day or specific events, could be further emphasized to enhance the temporal

Sl. No.	Author(s)	Journal Name and Year of Publication	Research Focus	Conclusions Derived via Authors	Limitations in the Study
					accuracy of the models.
3	Yi Zhang	Research on urban traffic industrial management under big data: Taking traffic congestion as an example.	Traffic flow prediction model with CNN, LSTM, and attention mechanism	Utilizes CNN for spatial features, LSTM with attention for dynamic historical influence, and weighted integration. Provides more accurate predictions.	Not specified
4	Jiawan Zhang	Applying Artificial Intelligence and Deep Belief Network to predict traffic congestion evacuation performance in smart cities.	Traffic flow prediction using Deep Belief Network algorithm	Compares models and shows DBN's higher prediction accuracy, effectively suppressing congestion in smart cities. Provides experimental references for future smart city construction.	Not specified
5	M Saleem	Smart cities: Fusion-based intelligent traffic congestion control system for vehicular networks using machine learning techniques.	Fusion-based intelligent traffic congestion control system using machine learning	Proposes FITCCS-VN to collect traffic data, improve traffic flow, and decrease congestion with high accuracy.	Not specified

6	Yuanbo Xu	Dynamic Traffic Correlationbased SpatioTemporal Graph Convolutional Network	Urban traffic is characterized by high dynamic spatio-temporal correlations, introducing uncertainty and complexity to traffic status prediction.	Utilizes DTC-STGCN for accurate traffic flow and speed forecasting in ITS. Outperforms state-ofthe-art baselines in longterm and peak time prediction.	In many urban traffic scenarios, spatiotemporal dependencies among traffic data change over time, making the use of a fixed adjacency matrix insufficient to describe dynamic dependencies.
7	Ouallane, Asma Ait, Assia Bakali,	Fusion of engineering insights and	Traffic Management Systems using	Reviews current approaches, summarizes benefits and drawbacks,	Lack of research covering the entire
	Ayoub Bahnasse, Said Broumi, and Mohamed Talea	emerging trends: Intelligent urban traffic management system	traffic data and emerging technologies	analyzes three main subsystems, and discusses future research directions.	urban road traffic management system.
8	Roy C. Park	Urban traffic accident risk prediction for knowledgebased mobile multimedia service	Accident prediction model using deep learning for risk assessment	Achieves 75% accuracy and 81% recall. Guides accident-prone regions and road risk levels for optimization.	Risk level can change based on each situation.
9	Jhilik Bhattacharya, and Prasenjit Basak	Multistep traffic speed prediction	Deep learningbased approach for accurate traffic forecasting	Addresses congestion due to increasing vehicle usage. Considers spatial and temporal dependencies.	Not specified
10	Yash Modi	A comprehensive review on intelligent traffic management using machine learning algorithms.”	Algorithms for smart traffic signal management, traffic flow prediction, congestion detection, and automatic signal detection	Explores AI and ML solutions for traffic congestion, pollution reduction, and improved traffic efficiency.	Not specified

11	AA Kashyap	Traffic flow prediction models–A review of deep learning techniques	Review of deep learning for traffic flow prediction	Highlights the importance of traffic flow prediction in intelligent transport systems. Discusses various deep learning architectures.	Not specified
12	Junchen Jin	GAN-Based Short-Term Link Traffic Prediction Approach for Urban Road Networks Under a Parallel Learning Framework	Short-term traffic speed prediction with PL-WGAN for urban road networks	Uses WGAN for robust data-driven traffic modeling. Combines generative and discriminative neural networks. Provides scalable and effective traffic prediction solution.	Lack of specific limitations mentioned.

2. Background Theory

In this comprehensive chapter, we explore the theoretical and technical underpinnings of the proposed project, beginning with an exploration of the programming languages, frameworks, and Integrated Development Environments (IDEs) crucial for its realization.

2.1 Programming languages, Frameworks and IDEs

1. **Python:** Python plays a fundamental role, particularly in the fields of machine learning. Renowned for its versatility, Python offers a wide range of libraries and frameworks specifically designed for machine learning, text processing, and data analysis. Utilizing Python version 3.11, we implemented various machine learning algorithms on our dataset to obtain the desired results.

2. **Visual Studio Code:** Visual Studio Code (VS Code) stands out as a proficient code editor, known for its efficiency and compatibility with various programming languages, including Python. In our project, we utilize VS Code to run the Flask Application and create a webpage that features an input box. This interactive interface enables users to input text, and the webpage dynamically displays the corresponding output.

3. **Google Colab:** Google Colab serves as a crucial cloud-based Python development environment, providing free access to Graphics Processing Unit (GPU) resources. Integral to our project, Google Colab facilitates the execution of different codes, including tasks such as dataset uploading, pre-processing, word embedding techniques, and the application of diverse algorithms.

4. **Flask:** Flask assumes a central role as the backend technology, seamlessly connecting the frontend user interface with the image prediction and machine learning components. By integrating Flask into our architecture, we establish a functional link that produces the desired output of predicting road traffic congestion, which is then showcased on a dedicated webpage.

5. **Machine Learning:** An area of artificial intelligence called machine learning has had a big impact on a lot of different industries, such finance, healthcare, and transportation. They offer sophisticated predictive and analytical capabilities by empowering computers to learn and make judgments without the need for explicit programming. In both machine learning and deep learning, algorithms like XGBoost, Random Forest, SVM, KNN, and Decision Trees are used; each provides a unique methodology. These methods are appropriate for a variety of issues and datasets because each has advantages and disadvantages of its own. As they develop further, they find uses in a variety of fields, spurring innovation and advancement in the AI space.

a. **Support Vector Machines (SVM):** A supervised learning algorithm used for regression and classification applications is SVM. It builds a hyperplane, or a collection of hyperplanes, to divide the data points into various classes. In order to improve generalization performance, support vector machines (SVM) seek to identify the ideal hyperplane that optimizes the margin between the classes. By transforming the data into a higher-dimensional space through kernel functions, SVM is capable of handling classification jobs that are both linear and non-linear. SVM is widely utilized in many fields, such as bioinformatics, image recognition, and text categorization.

b. **K Nearest Neighbors:** The k-nearest neighbors algorithm, sometimes referred to as KNN or k-NN, is a non-parametric supervised learning classifier that groups individual data points based on closeness in order to classify or predict them. Although it can be applied to classification or regression issues, it is usually employed as a classification algorithm, based on the idea that comparable points can be located next to each other.

c. **XG-Boost:** XG-Boost is a distributed gradient boosting library that has been developed to achieve maximum efficiency, versatility, and portability. It uses the Gradient Boosting framework to implement machine learning algorithms. With the use of parallel tree boosting, or GBDT or GBM, which is another name for XG-Boost, numerous data science issues can be quickly and precisely resolved.

d. **Decision Trees:** Decision Trees are basic but effective machine learning algorithms that build a tree-like representation of decisions and their potential outcomes. Every leaf node in the tree indicates a class or a prediction, and every internal node in the tree represents a judgment based on a feature. Figure 1. The features that best split the data are used to determine class division, while the values of the attributes of the provided data are used to group the data. The information gain or entropy, which is defined as follows, is used to determine the best attribute at each internal node: $Entropy = -\sum c \in C$ $p(C)$ log2 $p(c)$. The property with the lowest entropy is the best one. Decision trees handle both numerical and categorical data and are simple to understand and show. However, if they are not regularized appropriately, they may experience overfitting. The performance of Decision Trees is frequently enhanced by ensemble techniques such as Random Forests.

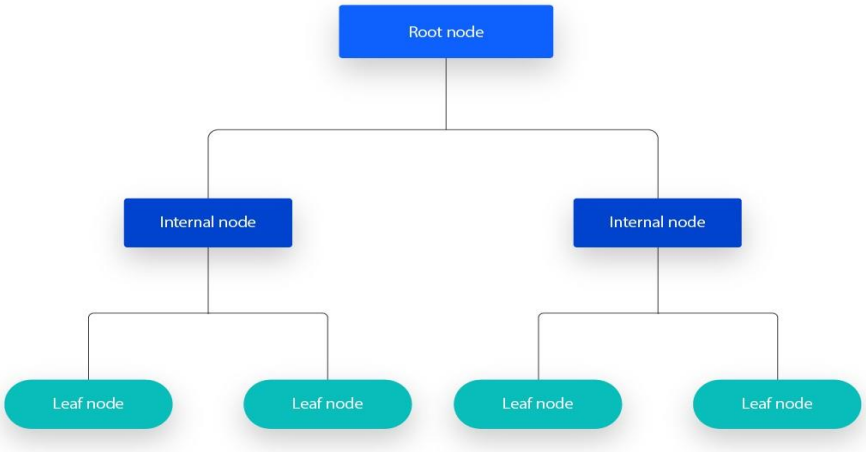

Figure 1: Decision tree with root node, decision nodes and terminal nodes.

e. **Random Forest:** Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions. Each decision tree is trained on a random subset of the training data and random subsets of the features. During prediction, each decision tree in the forest independently predicts the outcome, and the final prediction is determined by majority voting or averaging, Figure 2. Random forest uses bagging and feature randomness to generate these random subsets to ensure low correlation among the tree.

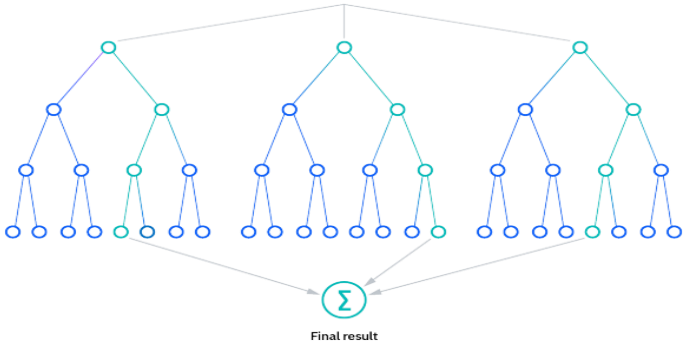

Figure 2: Random Forest structure consisting of multiple decision trees combined to produce a result.

2.2 Conclusion

Within this comprehensive chapter, we have meticulously outlined the array of technologies that have been strategically chosen for the successful completion of our project. We have offered a full explanation of why they were chosen, ensuring that the project's title and goal are in sync to establish a clear and focused trajectory. This alignment forms a solid foundation for the subsequent phases of the project.

3. Aims and Objectives

3.1 Title

Traffic-Sense: Harnessing Interpretable Machine Learning to Revolutionize Urban Traffic Congestion Prediction

3.2 Aim

The aim of the project is to develop a robust and accurate system that can anticipate traffic congestion in urban areas. By leveraging historical traffic data, real-time sensor information, and machine learning algorithms, the project aims to accurately predict traffic congestion patterns, optimize traffic flow, and implement effective management strategies to alleviate congestion and enhance transportation efficiency.

3.3 Objectives

The primary objectives and goals of the project are:

1. To conduct a comprehensive literature review on existing research related to urban traffic congestion prediction using machine learning and image analysis techniques.
2. To collect real-time traffic camera images or video streams from various urban areas for the purpose of training and testing the machine learning model.
3. To apply image preprocessing techniques, including image enhancement, noise removal, and distortion correction, to improve the traffic camera photos' dependability and quality.
4. To segment the traffic camera images into distinct regions, such as vehicles, road infrastructure, and other relevant objects, to enable effective feature extraction.
5. To develop a user-friendly web-based application that integrates the interpretable machine learning model, allowing users to access real-time traffic congestion predictions and visualize the important contributing factors.

3.4 Methods & Methodology

The methodology for this project can be structured in the following key ways:

1. **Literature Review:**

a) Identify and review relevant academic papers, research articles, and studies related to methodologies for image analysis, machine learning, and urban traffic congestion prediction.

b) Use machine learning models to analyze current approaches, industry best practices, and difficulties in the field of traffic congestion prediction.

c) Synthesize the findings to inform the development of TrafficSense and establish a strong foundation for the project.

2. **Data Collection:**

a) Collect real-time traffic camera images or video streams from diverse urban areas, capturing various traffic conditions and scenarios.

b) Ensure data privacy and compliance with ethical guidelines while obtaining the necessary permissions for data collection.

3. **Image Pre-processing:**

a) Apply image pre-processing techniques to enhance the quality of traffic camera images, including noise removal, contrast enhancement, and distortion correction.

b) Normalize the images to ensure uniformity and consistency in the data.

4. **Image Segmentation:**

a) Segment the pre-processed traffic camera images into relevant regions, such as vehicles, road infrastructure, and other objects, using segmentation algorithms.

b) Extract features from each segmented region, including vehicle count, density, speed, lane occupancy, and road occupancy.

5. **Feature Selection:**

a) Make use of feature selection techniques to determine which features are most pertinent and significant in predicting traffic congestion.

b) Select features that have a significant impact on traffic conditions and can provide valuable insights into congestion factors.

6. Interpretable Machine Learning Model Development

a) Develop interpretable machine learning models, such as rule-based, linear, or decision tree models for forecasting traffic congestion.

b) Implement techniques that provide explanations for model predictions, ensuring interpretability and transparency.

7. Model Training & Testing:

a) To train the machine learning models and evaluate their effectiveness, divide the dataset into training and testing sets.

b) To assess the efficacy of the model, use suitable assessment metrics such as accuracy, precision, recall, and F1 score.

8. Web-Based Application Development:

a) Design and develop a user-friendly web-based application that integrates the trained interpretable machine learning model.

b) Provide a real-time traffic congestion prediction feature accessible to users through the web interface.

9. Performance Evaluation and Visualization:

a) Evaluate the performance of Traffic-Sense using real-world traffic data and compare the model predictions with actual traffic conditions.

b) Visualize the important contributing factors influencing traffic congestion predictions to enhance interpretability.

10. Documentation & Reporting:

a) Maintain detailed documentation of the development process, methodologies, and findings throughout the project.

b) Prepare a comprehensive report summarizing the methodology, results, and conclusions of Traffic-Sense.

c) Present the research findings to relevant stakeholders, academia, and potential users to share insights and potential impact.

In conclusion, the meticulous consideration of methods and methodology in this study lays a robust foundation for the subsequent phases of research. The methodological choices made reflect a strategic alignment with the research objectives, ensuring the collection of relevant and reliable data. The comprehensive nature of the selected methods, coupled with rigorous data analysis techniques, is poised to yield nuanced insights into the intricacies of the research question.

Furthermore, the transparent and replicable nature of the chosen methodology enhances the credibility of the study. By adhering to established research practices and incorporating appropriate validation measures, this research seeks to contribute to the broader academic discourse in a meaningful and reliable manner.

As the study transitions to the next phases, the efficacy of the chosen methods remains paramount. The success of our research hinges on the careful execution of the outlined methodology, and it is through this methodological rigor that we aim to unearth valuable contributions to the understanding of our research area.

4. Problem Solving Approach

4.1 Problem Solving Approach

The "Traffic-Sense" project adopts a holistic problem-solving approach, aiming to address the multifaceted challenges associated with urban congestion prediction. The following key strategies outline our comprehensive approach to solving the identified problems:

1. **Data Integration and Diversity:** Recognizing the complexity of urban environments, our project places a strong emphasis on integrating diverse data sources. By amalgamating traffic flow data, weather conditions, and special event information, we aim to create a comprehensive dataset that captures the dynamic nature of congestion triggers.

2. **Interpretable Machine Learning Models:** In response to the need for transparency in congestion prediction, the project leverages interpretable machine learning models. This approach not only ensures accurate predictions but also enables understanding of the factors influencing congestion dynamics. By prioritizing interpretability, we enhance the trustworthiness of our predictions.

3. **Actionable Insights:** The project's goal is not only to predict congestion but to provide actionable insights for decision-makers. Through the interpretation of machine learning model outputs, users can make well-informed decisions about traffic control, strategies, infrastructure planning, and urban development, contributing to effective congestion mitigation.

4. **User-Friendly Interface:** Recognizing the diverse user base, including urban planners, traffic managers, and the general public, the project incorporates a user-friendly interface. This interface serves as a platform for presenting congestion predictions and insights in an accessible manner, fostering engagement and collaboration.

5. **Scalability and Adaptability:** The "Traffic-Sense" project is designed with scalability in mind. The methodology and models employed should be adaptable to varying urban landscapes and data volumes. This scalability ensures the longevity and applicability of the project as cities evolve and expand.

6. **Public Engagement and Awareness:** Acknowledging the importance of community involvement, the project implements strategies for public engagement and awareness. Interactive platforms, educational campaigns, and feedback mechanisms are integrated

to encourage active participation and cooperation in congestion management efforts.

By combining these elements in our problem-solving approach, the "Traffic-Sense" project strives to provide a comprehensive and efficient resolution to the problems relating to the forecasting of urban congestion, supporting the creation of more sustainable and intelligent cities.

4.2 Requirements, both functional and non-functional

Functional Requirements:

1. Data Integration: A variety of urban data sources, such as information on traffic patterns, meteorological conditions, and special events, should be seamlessly integrated by the system. In order to guarantee the accuracy and consistency of integrated data, it should carry out preprocessing and data cleaning.
2. Prediction Model: In order to predict urban congestion transparently and accurately, the system needs to use interpretable machine learning models. It ought to enable model updating and training in response to changing traffic patterns.
3. User Interface: A user-friendly interface should be developed to present congestion predictions and insights. The interface should be accessible to various user groups, including urban planners, traffic managers, and the general public.
4. Actionable Insights: The system should generate actionable insights based on the interpretation of machine learning model outputs. Insights should be presented in a clear and understandable format to facilitate decision-making.
5. Scalability: The system should be designed to scale seamlessly, accommodating varying urban landscapes and data volumes. It should handle increased computational loads as the project expands to cover larger geographical areas.
6. Notification System: Implement a notification system to alert relevant stakeholders of predicted congestion events. Notifications should be timely and customizable based on user preferences.

Non-Functional Requirements:

1. Performance: The system should demonstrate high performance, providing realtime or near-real-time congestion predictions. Response times for user interactions with the interface should be minimal.
2. Scalability: In order to accommodate the growing amount of data and user interactions, the system needs to be scalable. It should maintain performance levels as the project expands to cover additional regions.
3. Reliability: The system should be reliable, with a high level of accuracy in congestion predictions. It should have mechanisms in place to handle and recover from system failures gracefully.
4. Security: Implement robust security measures to protect sensitive urban data. Ensure user authentication and authorization mechanisms to control access to system features.
5. Usability: The user interface should be intuitive and easy to navigate. Provide documentation and training materials to facilitate user adoption.
6. Interpretability: Ensure the interpretable machine learning models used are capable of explaining predictions in a clear and understandable manner. Incorporate features that allow users to delve into the factors influencing congestion predictions.
7. Collaboration: Facilitate collaboration with external partners and stakeholders through data-sharing mechanisms. Provide APIs or integration points for seamless collaboration with other urban planning tools and systems.

These functional and non-functional requirements serve as a foundation for developing and assessing the success of the "Traffic Sense" project. They guide the design, implementation, and evaluation of the system to ensure it meets the needs of stakeholders and addresses the challenges associated with urban congestion prediction.

4.3 Design

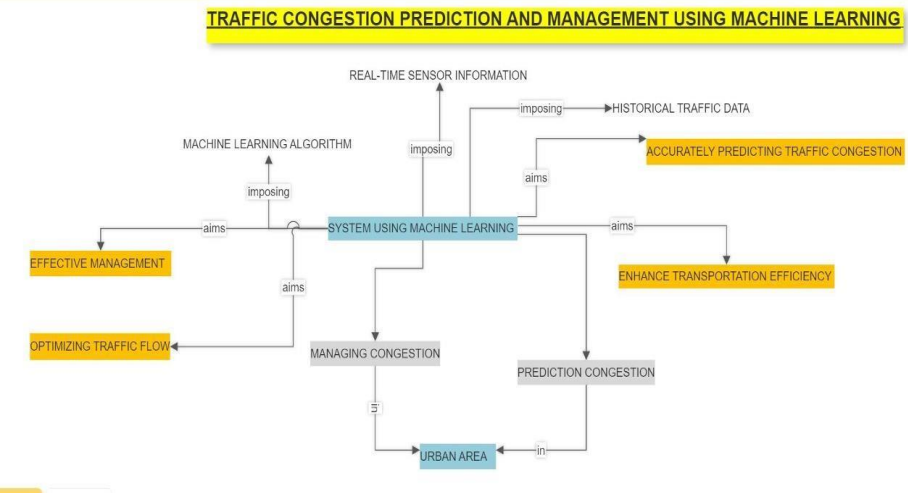

Figure 3: Block Diagram

In the quest for a smarter and more responsive urban mobility landscape, the design system of "Traffic-Sense" stands as a pivotal element in translating conceptual ideas into a tangible and functional solution. The intricacies of urban congestion prediction necessitate a wellthought-out and cohesive design, seamlessly integrating data, algorithms, and user interfaces. This section outlines the fundamental principles and components of the system design, depicted succinctly in the accompanying flowchart.

Sequence diagram:

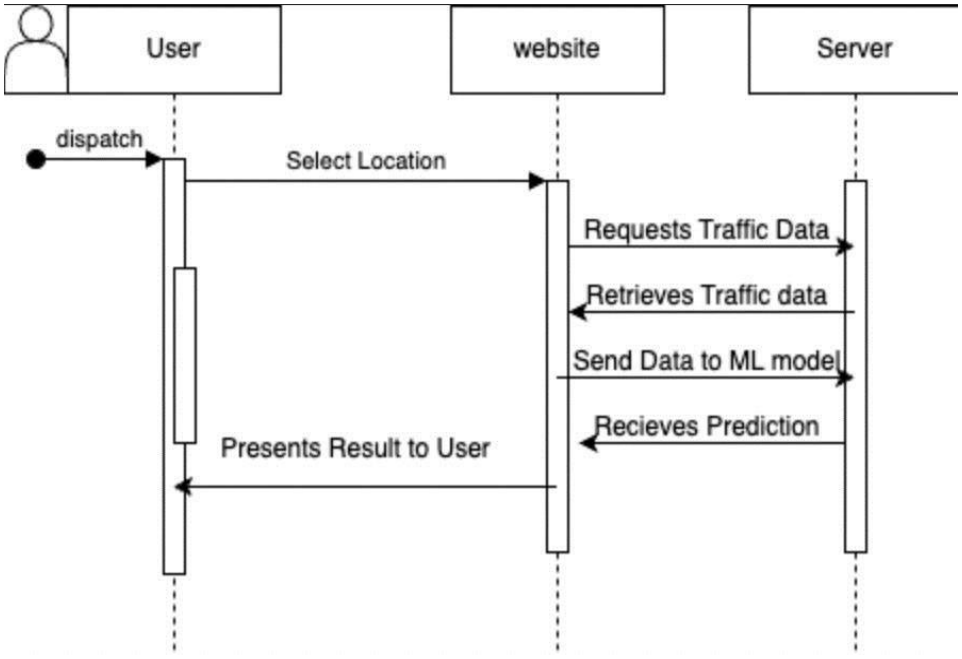

Figure 4: Sequence diagram

4.4 Implementation

The implementation phase of the "Traffic Sense" project stands as a testament to the fusion of technological innovation and the real-world dynamics of traffic in Bengaluru. As we venture into the details of this phase, it becomes evident that the heartbeat of our approach lies in the integration of on-the-ground data captured from eight strategic locations within the bustling city of Bengaluru.

These eight distinct places—each a microcosm of the city's diverse traffic patterns—have been carefully chosen to enrich the project's machine learning models. Through a collection of photographs depicting traffic scenarios at these locations, the system undergoes a training process, becoming attuned to the nuanced patterns and challenges unique to each locale. The incorporation of real-world data from prominent spots such as Silk Board Junction, MG Road, and Electronic City infuses the models with a localized understanding of Bengaluru's traffic landscape.

The user interface, an integral part of our implementation, encapsulates the essence of usercentric design. Users interact with the system by selecting specific locations of interest, and in return, the trained models provide real-time insights into the traffic status of these places. Whether it is an assessment of clear conditions, heavy traffic, or the occurrence of an accident, the system transforms raw images into actionable information for urban planners, traffic managers, and the public.

Implementation of the model:

To implement the model, we first process all the images in our dataset, convert them to grayscale and apply gaussian filters to remove noise and to blur the image. Grayscale conversion is necessary as it reduces color-based complications during the processing.Then the image segmentation is done by detecting the contours and drawing them.

```
train_path = '/content/drive/MyDrive/trafficnet_dataset_v1/train'
test_path = '/content/drive/MyDrive/trafficnet_dataset_v1/test'
segmented_images = '/content/drive/MyDrive/segmented_trafficnet_dataset'

def img_preprocess(image_path, output_path):
    original_image = cv2.imread(image_path)
    target_size = (256, 256)
    resized_image = cv2.resize(original_image, target_size)

    gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    _, thresholded = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # contours
    contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    segmented_image = resized_image.copy()
    cv2.drawContours(segmented_image, contours, -1, (0, 255, 0), 2)

    # Save the segmented image with the same folder and subfolder structure
    output_folder = os.path.dirname(output_path)
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    cv2.imwrite(output_path, segmented_image)

# Function to process images in a folder and save segmented images
def process_images_in_folder(folder_path, output_folder):
    for filename in os.listdir(folder_path):
        if filename.endswith('.jpg') or filename.endswith('.png'):
            image_path = os.path.join(folder_path, filename)
            output_path = os.path.join(output_folder, filename.replace('.jpg', '_segmented.jpg').replace('.png', '_segmented.png'))
            img_preprocess(image_path, output_path)

# Process images in train dataset subfolders and save segmented images in segmented dataset
for subfolder in os.listdir(train_path):
    subfolder_path = os.path.join(train_path, subfolder)
    if os.path.isdir(subfolder_path):
        segmented_output_folder = os.path.join(segmented_images, 'train', subfolder + '_segmented')
        print(f"Segmenting images in train/{subfolder} folder and saving segmented images in {segmented_output_folder}:")
```

Figure 5: Image preprocessing and segmentation

Histogram of gradients or HOG features are extracted for feature extraction. The HOG features are 1-D NumPy arrays which are collected for each image in the dataset and stored in the local drive.

```python
from skimage.feature import hog
from skimage import exposure

# Extract HOG features from an image and save as a numpy array
def extract_hog_features(image_path):
    original_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)  # Read the image in grayscale

    # HOG feature extraction
    features = hog(original_image, orientations=9, pixels_per_cell=(8, 8),
                   cells_per_block=(2, 2), visualize=False, multichannel=False)
    return features

# Process images in a folder and save HOG features in a new directory
def process_images_in_folder(folder_path, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    for filename in os.listdir(folder_path):
        if filename.endswith('_segmented.jpg') or filename.endswith('_segmented.png'):
            image_path = os.path.join(folder_path, filename)
            output_path = os.path.join(output_folder, filename.replace('_segmented.jpg', '_hog_features.npy').replace('_segmented.png', '_hog_features.npy'))
            features = extract_hog_features(image_path)
            np.save(output_path, features)

# Paths to your train and test datasets
train_path = '/content/drive/MyDrive/segmented_trafficnet_dataset/train'
test_path = '/content/drive/MyDrive/segmented_trafficnet_dataset/test'

# Feature directory path
feature_output_path = '/content/drive/MyDrive/features_extracted_dataset'

# Process images in train dataset subfolders and save HOG features in respective subdirectories
for subfolder in os.listdir(train_path):
    subfolder_path = os.path.join(train_path, subfolder)
    if os.path.isdir(subfolder_path):
        hog_output_folder = os.path.join(feature_output_path, 'train',subfolder)
        print(f"Extracting HOG features from images in train/{subfolder} folder and saving features in {hog_output_folder}:")
        process_images_in_folder(subfolder_path, hog_output_folder)

# Process images in test dataset subfolders and save HOG features in respective subdirectories
for subfolder in os.listdir(test_path):
    subfolder_path = os.path.join(test_path, subfolder)
    if os.path.isdir(subfolder_path):
        hog_output_folder = os.path.join(feature_output_path, 'test', subfolder)
        print(f"Extracting HOG features from images in test/{subfolder} folder and saving features in {hog_output_folder}:")
        process_images_in_folder(subfolder_path, hog_output_folder)
```

Figure 6: Extracting HOG features from the images

These features are loaded and labelled in the environment to be trained for model.

```python
from sklearn import svm
import os
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# load HOG features and labels
def load_features_and_labels(folder_path):
    features = []
    labels = []

    for class_folder in os.listdir(folder_path):
        class_folder_path = os.path.join(folder_path, class_folder)
        if os.path.isdir(class_folder_path):
            label = class_folder
            for filename in os.listdir(class_folder_path):
                if filename.endswith('hog_features.npy'):
                    features_path = os.path.join(class_folder_path, filename)
                    feature_vector = np.load(features_path)
                    features.append(feature_vector)
                    labels.append(label)
                    print(f"Loaded feature from {features_path} for label {label}")

    return np.array(features), np.array(labels)

# Paths to your feature directories
train_feature_path = '/content/drive/MyDrive/features_extracted_dataset/train'
test_feature_path = '/content/drive/MyDrive/features_extracted_dataset/test'

# Load training features and labels
X_train, y_train = load_features_and_labels(train_feature_path)

# Load testing features and labels
X_test, y_test = load_features_and_labels(test_feature_path)
print("X_test shape:", X_test.shape)
print("X_train shape:", X_train.shape)
```

Figure 7: Labelling the features

Our next task is model selection for classification of the images. Various models can be chosen for the same but we will choose the one which is the most efficient one.

Models Trained:

1. Support Vector Machines (SVM):

```
# Create an SVM classifier
svm_classifier = svm.SVC(kernel='poly', C=100.0)

# Train the classifier on the reduced features
svm_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm_classifier.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

Figure 8: Model training for svm classifier

```
Accuracy: 0.70125
Confusion Matrix:
[[134  13  20  33]
 [ 19 147  11  23]
 [ 29   3 145  23]
 [ 30  22  13 135]]
Classification Report:
                        precision    recall  f1-score   support

      accident_segmented       0.63      0.67      0.65       200
 dense_traffic_segmented       0.79      0.73      0.76       200
          fire_segmented       0.77      0.72      0.75       200
sparse_traffic_segmented       0.63      0.68      0.65       200

                accuracy                           0.70       800
               macro avg       0.71      0.70      0.70       800
            weighted avg       0.71      0.70      0.70       800
```

Figure 9: model metrics and results

2. KNN classifier

```python
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier(n_neighbors=3)

# Fit the classifier to the training data
knn_classifier.fit(X_train, y_train)
y_pred=knn_classifier.predict(X_test)
print("KNN classifier")
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

Figure 10: Model training KNN classifier

```
KNN classifier
Accuracy: 0.39375
Confusion Matrix:
[[102  52   1  45]
 [ 68 105   1  26]
 [ 88  52   9  51]
 [ 63  38   0  99]]
Classification Report:
                         precision    recall  f1-score   support

      accident_segmented       0.32      0.51      0.39       200
 dense_traffic_segmented       0.43      0.53      0.47       200
          fire_segmented       0.82      0.04      0.09       200
sparse_traffic_segmented       0.45      0.49      0.47       200

                accuracy                           0.39       800
               macro avg       0.50      0.39      0.35       800
            weighted avg       0.50      0.39      0.35       800
```

Figure 11: KNN metrics and results

3. Random forest classifier

```python
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# For illustration purposes, let's assume you've already encoded your labels using LabelEncoder
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier
rf_classifier.fit(X_train, y_train_encoded)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test_encoded, y_pred)
conf_matrix = confusion_matrix(y_test_encoded, y_pred)
classification_rep = classification_report(y_test_encoded, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

Figure 12: Random forest classifier

```
Accuracy: 0.58125
Confusion Matrix:
[[105  30  28  37]
 [ 19 125  24  32]
 [ 45   4 123  28]
 [ 34  36  18 112]]
Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.53      0.52       200
           1       0.64      0.62      0.63       200
           2       0.64      0.61      0.63       200
           3       0.54      0.56      0.55       200

    accuracy                           0.58       800
   macro avg       0.58      0.58      0.58       800
weighted avg       0.58      0.58      0.58       800
```

Figure 13: Model metrics and results for random forest

4. XGB classifier

```
xgb_classifier = xgb.XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,max_depth=3,subsample=0.8,
    colsample_bytree=0.8)
# Train the classifier
xgb_classifier.fit(X_train, y_train_encoded)

# Make predictions on the test set
y_pred = xgb_classifier.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test_encoded, y_pred)
conf_matrix = confusion_matrix(y_test_encoded, y_pred)
classification_rep = classification_report(y_test_encoded, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

Figure 14: XGBoost Classifier

```
Accuracy: 0.64125
Confusion Matrix:
[[125  19  25  31]
 [ 14 142  19  25]
 [ 42   6 125  27]
 [ 33  28  18 121]]
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.62      0.60       200
           1       0.73      0.71      0.72       200
           2       0.67      0.62      0.65       200
           3       0.59      0.60      0.60       200

    accuracy                           0.64       800
   macro avg       0.64      0.64      0.64       800
weighted avg       0.64      0.64      0.64       800
```

Figure 15: Model metrics and results for XGB classifier

5. Decision tree classifier

```
import os
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Create a Decision Tree classifier
# dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier = DecisionTreeClassifier(random_state=42, max_depth=20)

# Train the classifier
dt_classifier.fit(X_train, y_train_encoded)

# Make predictions on the test set
y_pred = dt_classifier.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test_encoded, y_pred)
conf_matrix = confusion_matrix(y_test_encoded, y_pred)
classification_rep = classification_report(y_test_encoded, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

Figure 16: Decision Tree classifier

```
Accuracy: 0.4025
Confusion Matrix:
[[75 36 46 43]
 [53 94 18 35]
 [45 42 71 42]
 [43 47 28 82]]
Classification Report:
              precision    recall  f1-score   support

           0       0.35      0.38      0.36       200
           1       0.43      0.47      0.45       200
           2       0.44      0.35      0.39       200
           3       0.41      0.41      0.41       200

    accuracy                           0.40       800
   macro avg       0.40      0.40      0.40       800
weighted avg       0.40      0.40      0.40       800
```

Figure 17: Results and metrics for decision tree classifier

Model selection

As we can see that the SVM classifier gives the best results, we are selecting it for our model training and prediction purposes.

Prediction function:

```
import cv2
from skimage.feature import hog
from skimage import exposure
from google.colab.patches import cv2_imshow

def preprocess_image(image_path):
    # Read the image in grayscale
    original_image = cv2.imread(image_path)
    target_size = (256, 256)
    resized_image = cv2.resize(original_image, target_size)

    gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    _, thresholded = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # contours
    contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    segmented_image = resized_image.copy()
    cv2.drawContours(segmented_image, contours, -1, (0, 255, 0), 2)
    return segmented_image

def extract_hog_features(image_path):
    # Preprocess the image
    segmented_image = preprocess_image(image_path)
    segmented_image = cv2.cvtColor(segmented_image, cv2.COLOR_BGR2GRAY)
    features = hog(segmented_image, orientations=9, pixels_per_cell=(8, 8),
                   cells_per_block=(2, 2), visualize=False, multichannel=False)

    return features

# Example usage
image_path = '/content/traffic-jam-at-road.webp'

segmented_image = preprocess_image(image_path)
cv2_imshow( cv2.imread(image_path))
cv2_imshow(segmented_image)
hog_features = extract_hog_features(image_path)

def predict_image(hog_features):
    hog_features=hog_features.reshape(1,-1)
    prediction = svm_classifier.predict(hog_features)
    return prediction

predictions=predict_image(hog_features)
print(predictions[0])
```

Figure 18: Prediction Function

Implementation for Website and Server:

Within the digital corridors of the "Traffic Sense" project, the convergence of two integral components—website and server—forms the technological backbone that powers its predictive prowess. This section unveils the intricate codebase of both the website and server, providing a dual perspective into the carefully orchestrated symphony of algorithms, data flow, and user interaction.

The attached code snippets offer a behind-the-scenes glimpse into the meticulous craftsmanship that brings the project to life. The website's code, with its intuitive design and user-centric functionalities, encapsulates the essence of user experience, presenting predictions and insights in a comprehensible manner. Simultaneously, the server-side code orchestrates the seamless integration of diverse data sources and the deployment of interpretable machine learning models.

As we navigate through these code snippets, it becomes apparent that the design principles extend beyond visual aesthetics. The clarity and efficiency of the code are paramount, ensuring not only the functionality of the website but also the scalability and adaptability required to meet the demands of dynamic urban environments.

Snippets for Website:

trafic.html × server.py 8

trafic.html > style > body

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Traffic Checker in Bengaluru</title>
</head>
<body>
    <div class="container">
        <h1>Traffic Checker</h1>
        <img src="background.jpg" alt="City Traffic" class="background-image">
        <label for="area">Select Area:</label>
        <select id="area" onchange="getTrafficConditions()">
            <option value="" disabled selected>SELECT AREA</option>
            <option value="area1">Peenya Industrial, Bengaluru</option>
            <option value="area2">Vidhan Soudha, Bengaluru</option>
            <option value="area3">Lal Bagh, Bengaluru</option>
            <option value="area4">Cubbon Park, Bengaluru</option>
            <option value="area5">Lumbini Gardens, Bengaluru</option>
            <option value="area6">Bannerghatta National Park, Bengaluru</option>
            <option value="area7">Chunchi Falls, Bengaluru</option>
            <option value="area8"> ISKCON Temple, Bengaluru</option>
        </select>
        <div id="trafficStatus" class="traffic-status">
            <!-- Display traffic status here -->
        </div>
        <div><img id="imageElement" src="" alt="Description of the image"></div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Figure 19: Code for Website

```
trafic.html > style > body
<style>
    body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        margin: 0;
        padding: 0;
        background-image:  url(background_image2.jpg);
        background-size: cover;
        background-attachment: fixed;
        color : #333;
    }

    .container {
        position: relative;
        max-width: 600px;
        margin: 50px auto;
        text-align: center;
        background-color: #fff;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        overflow: hidden;
    }

    .background-image {
        width: 100%;
        height: auto;
        position: absolute;
        top: 0;
        left: 0;
        z-index: -1;
        opacity: 0.3;
    }

    h1 {
        margin-top: 40px;
        color: #333;
```

Figure 20: Code for Website

```
trafic.html  server.py 8
trafic.html > style > body
    h1 {
        margin-top: 40px;
        color: #333;
    }

    label {
        font-weight: bold;
        display: block;
        margin-top: 20px;
        color: #333;
    }

    select {
        padding: 12px;
        font-size: 16px;
        width: 80%;
        margin: 10px auto;
        color: #333;
    }

    .traffic-status {
        margin-top: 20px;
        padding: 20px;
        border: 1px solid #ccc;
        background-color: #f9f9f9;
        border-radius: 8px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        color: #333;
    }

    p {
        margin: 0;
    }
```

Figure 21: Code for Website

```
trafic.html ×  server.py 8
trafic.html > style > h1
    }

    /* Additional styling for better responsiveness */
    @media only screen and (max-width: 600px) {
        .container {
            border-radius: 0;
        }

        select {
            width: 100%;
        }
    }
</style>

<script>
function getTrafficConditions() {
    // Assuming you have a traffic API endpoint
    const selectedArea = document.getElementById("area").value;

    // Replace the following URL with your actual traffic API endpoint
    // const apiUrl = `https://api.example.com/traffic?area=${selectedArea}`;

    // fetch(apiUrl)
    //     .then(response => response.json())
    //     .then(data => {
    //         displayTrafficConditions(data);
    //     })
    //     .catch(error => {
    //         console.error('Error fetching traffic data:', error);
    //         displayTrafficConditions({ status: 'Error fetching data' });
    //     });
    var data = {'place': selectedArea};

    // fetch('http://127.0.0.1:5000/predict', {
    //     method: 'POST',
    //     headers: {
```

Figure 22: Code for Website

```
trafic.html ×  server.py 8
trafic.html > style > {} @media only screen and (max-width: 600px) > .container
    // fetch('http://127.0.0.1:5000/predict', {
    //     method: 'POST',
    //     headers: {
    //         'Content-Type': 'application/json',
    //     },
    //     body: JSON.stringify(data),
    //     })
    //     .then(response => response.json())
    //     .then(data => {
    //         displayTrafficConditions(data);
    //     })
    //     .catch((error) => {
    //         console.error('Error fetching traffic data:', error);
    //         displayTrafficConditions({ status: 'Error fetching data' });
    //     });

    fetch('http://127.0.0.1:5000/predict',{
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(data),
            }
            )
            .then(response => response.json())
            .then(data => {
                // Display JSON data
                displayTrafficConditions(data.output1);

                const imageData = atob(data.image);

            // Convert the binary data to a Uint8Array
            const arrayBuffer = new Uint8Array(imageData.length);
            for (let i = 0; i < imageData.length; i++) {
                arrayBuffer[i] = imageData.charCodeAt(i);
            }
```

Figure 23: Code for Website

```
trafic.html ×  server.py 8
trafic.html > script > getTrafficConditions
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(data),
            }
            )
            .then(response => response.json())
            .then(data => {
                // Display JSON data
                displayTrafficConditions(data.output1);

                const imageData = atob(data.image);

            // Convert the binary data to a Uint8Array
            const arrayBuffer = new Uint8Array(imageData.length);
            for (let i = 0; i < imageData.length; i++) {
                arrayBuffer[i] = imageData.charCodeAt(i);
            }

            // Create a Blob from the Uint8Array
            const imageBlob = new Blob([arrayBuffer], { type: 'image/jpeg' });

            // Create an object URL from the Blob
            const objectURL = URL.createObjectURL(imageBlob);
                // Set the image source
                document.getElementById('imageElement').src = objectURL;
            })
            .catch(error => console.error('Error fetching data and image:', error));
}

function displayTrafficConditions(data) {
    const trafficStatusDiv = document.getElementById("trafficStatus");
    trafficStatusDiv.innerHTML = `<p><strong>Traffic Conditions:</strong> ${data.prediction}</p>`;
    // You can customize the display based on the structure of your API response
}
</script>
```

Figure 24: Code for Website

Snippets for the Server:

```
server.py > ...
from flask import Flask, render_template, request, jsonify
from flask_cors import CORS  # Import the CORS module
import joblib
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
import random
from skimage.feature import hog
from skimage import exposure
import base64

app = Flask(__name__)
CORS(app)


def get_random_image_path():

    folder_path=r'C:\Users\utkarsh singh\Desktop\26_project\image_data'

    # Get a list of all files in the folder
    images = os.listdir(folder_path)

    # Select a random image file
    random_image = random.choice(images)

    # Construct the full path to the random image
    random_image_path = os.path.join(folder_path, random_image)

    return random_image_path


def preprocess_and_segment_image(image_path):
    original_image = cv2.imread(image_path)
    target_size = (256, 256)
    resized_image = cv2.resize(original_image, target_size)
```

Figure 25: Code for the server

```python
    return random_image_path


def preprocess_and_segment_image(image_path):
    original_image = cv2.imread(image_path)
    target_size = (256, 256)
    resized_image = cv2.resize(original_image, target_size)

    gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    _, thresholded = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # contours
    contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    segmented_image = resized_image.copy()
    cv2.drawContours(segmented_image, contours, -1, (0, 255, 0), 2)
    return segmented_image

def extract_hog_features(seg_img):

    seg_img=cv2.cvtColor(seg_img, cv2.COLOR_BGR2GRAY)
    # Apply HOG feature extraction
    features = hog(seg_img, orientations=9, pixels_per_cell=(8, 8),
                   cells_per_block=(2, 2), visualize=False)

    return features


# Load the pre-trained scikit-learn model
model = joblib.load('svm_classifier.joblib')


@app.route('/predict', methods=['POST'])
def predict():
```

Figure 26: Code for the server

```python
    seg_img=cv2.cvtColor(seg_img, cv2.COLOR_BGR2GRAY)
    # Apply HOG feature extraction
    features = hog(seg_img, orientations=9, pixels_per_cell=(8, 8),
                   cells_per_block=(2, 2), visualize=False)

    return features


# Load the pre-trained scikit-learn model
model = joblib.load('svm_classifier.joblib')


@app.route('/predict', methods=['POST'])
def predict():

    data = request.get_json(force=True)

    image_path=get_random_image_path()
    with open(image_path, 'rb') as image_file:
        image_data = base64.b64encode(image_file.read()).decode('utf-8')


    seg_img=preprocess_and_segment_image(image_path)
    feature=extract_hog_features(seg_img)
    feature=np.expand_dims(feature, axis=0)
    prediction = model.predict(feature)
    output = {'prediction': prediction[0]}

    return jsonify(output1=output, image=image_data)


if __name__ == '__main__':
    app.run(debug=True)
```

Figure 27: Code for the server

4.5 Results

Screenshots:

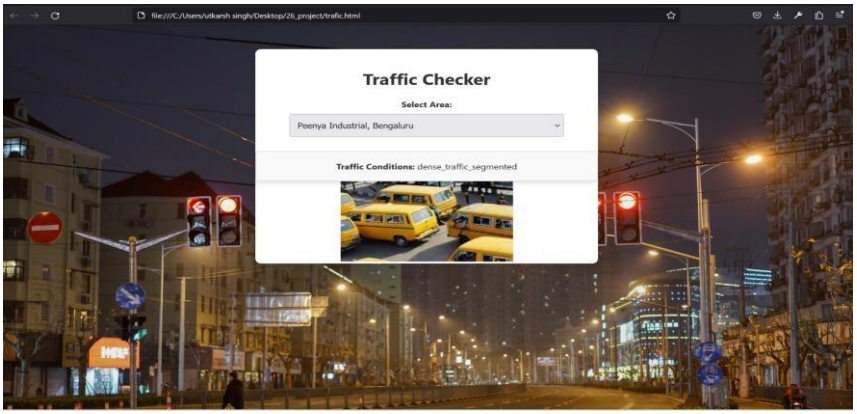

Figure 27: Website screenshots

Figure 28: Website screenshots

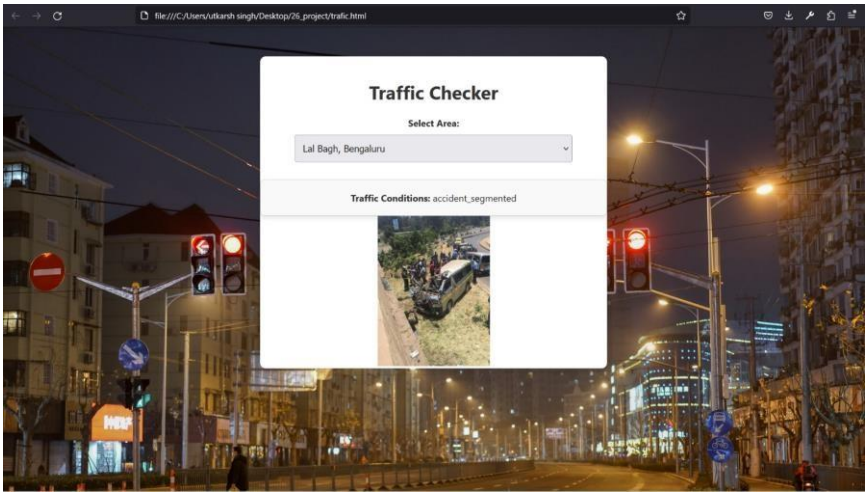

Figure 29: Website screenshots

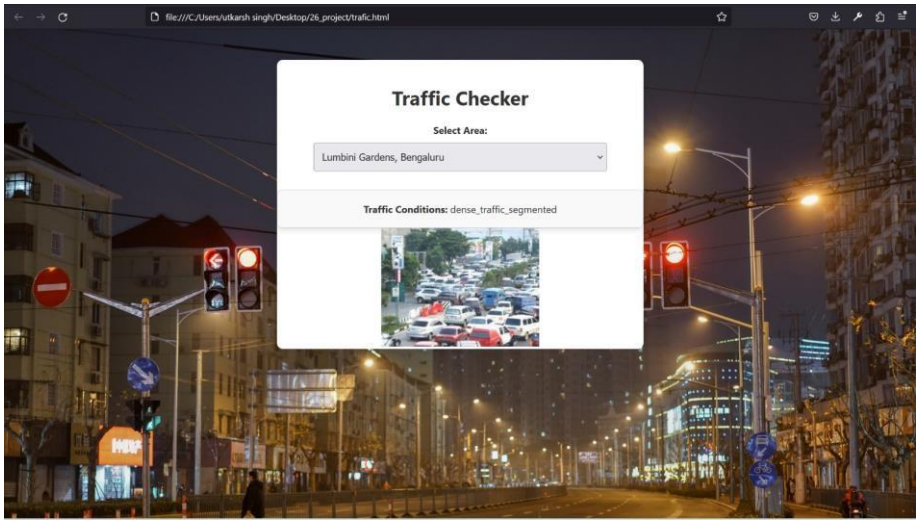

Figure 30: Website screenshots

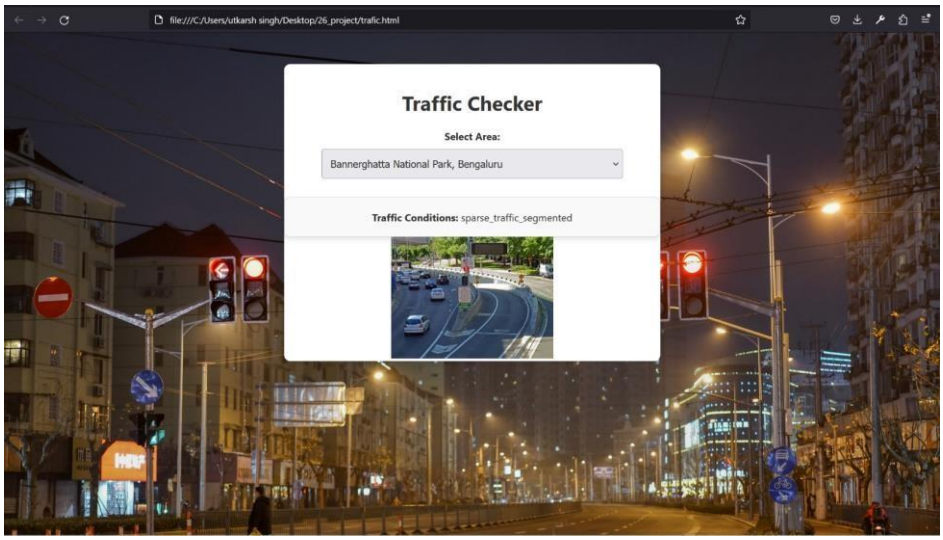

Figure 31: Website screenshots

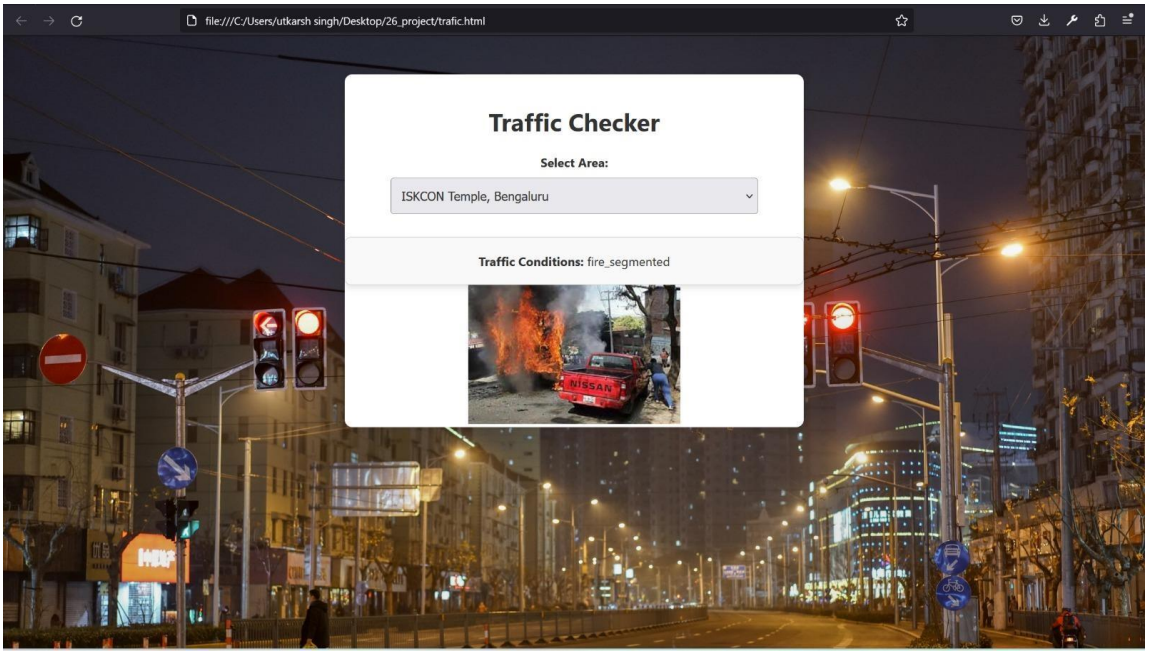

Figure 32: Website screenshots

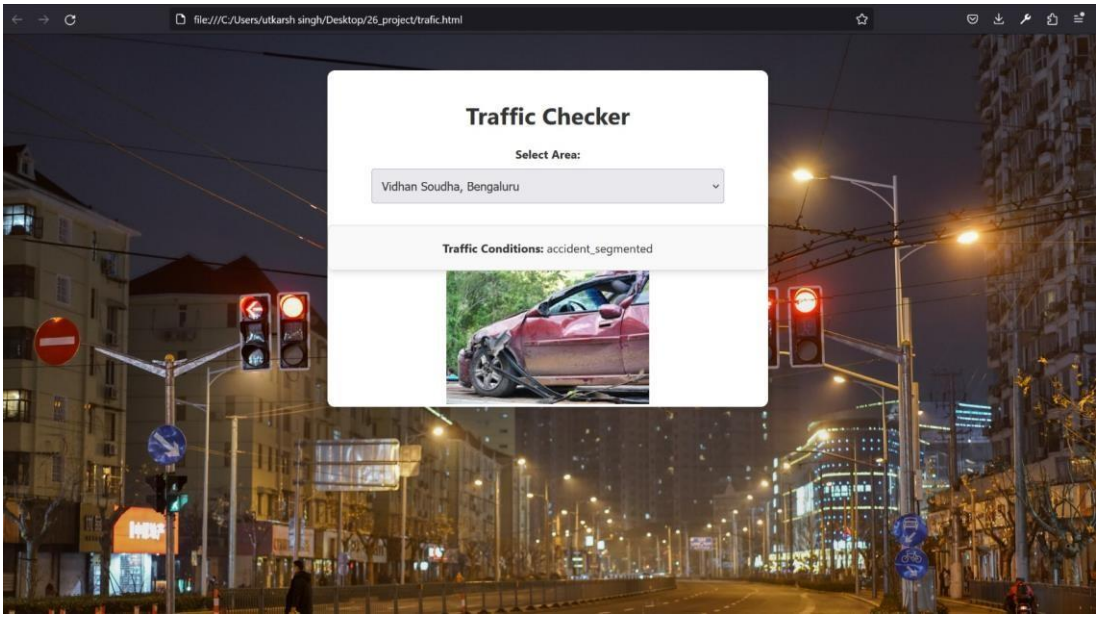

Figure 33: Website screenshots

5. Project Costings

This section presents a breakdown of the costs associated with developing the system prototype, based on the work covered in the preceding sections. the price of the used software and hardware.

Over the past three months, the entire team has been working hard. Report writing is another ongoing task for the project that is done after every goal is accomplished.

Components	**Cost**
Ide(VS code)	Free
Google Colab	Free
Data set(Traffic dataset)	Free
Flask framework	Free
Labour Cost	None
Total cost	0

Table 2: Project cost

Our project is free of charge, demonstrating our commitment to effectiveness. Utilizing the resources at our disposal, we guarantee quality without incurring costs. This cost-free approach demonstrates our dedication to providing value in an easy-to-use manner.

6. Conclusions and Suggestions for future work

6.1 Conclusion

In conclusion, our project undertakes a thorough literature review, delving into pertinent papers and studies on traffic congestion prediction. We meticulously analyze state-of-the-art methodologies, acknowledging existing challenges and establishing a robust foundation for our work. The subsequent phase involves the ethical collection of real-time traffic camera data from diverse urban settings, ensuring data privacy and compliance with ethical guidelines. To improve the quality of traffic camera images, we apply sophisticated image pre-processing techniques like distortion correction, noise reduction, and contrast enhancement. Moving forward, we employ image segmentation to delineate relevant regions and extract key features, including vehicle count, density, speed, lane occupancy, and road occupancy. The culmination of these processes' manifests in the creation of a user-friendly web-based application. This application seamlessly integrates a trained, interpretable machine learning model, offering users real-time traffic congestion predictions through an accessible web interface. Our project addresses urban traffic challenges, providing a valuable tool for users to plan efficient routes and navigate congested areas effectively. Through innovation and technology, we contribute to enhancing the overall efficiency of urban transportation systems.

6.2 Suggestion for future work

1. API Integration: Implement APIs to seamlessly integrate external data sources, such as weather APIs for real-time weather updates or social media APIs to capture events that might impact traffic.
2. Real-time Updates: Enhance the web application to provide real-time traffic updates, leveraging WebSocket or Server-Sent Events (SSE) to push live predictions to users as they occur.
3. Geospatial Visualization: Integrate geospatial visualization libraries like Leaflet or Mapbox to represent traffic predictions on an interactive map, providing users with an intuitive and comprehensive view of congestion patterns.
4. Model Versioning: Implement a model versioning system to easily switch between different machine learning models. This facilitates model updates and comparisons, ensuring the deployment of the most accurate and relevant model at any given time.
5. User Feedback Mechanism: Provide a feedback feature in the application so that users can correct errors or offer suggestions. This can help prediction models that are based on user feedback get better.

REFERENCES

1. 2Yuan, Xiaoming, Jiahui Chen, Jiayu Yang, Ning Zhang, Tingting Yang, Tao Han, and Amir Taherkordi. "Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction." IEEE Transactions on Intelligent Transportation Systems (2022).
2. 1 Medina-Salgado, Boris, Eddy Sanchez-DelaCruz, Pilar Pozos-Parra, and Javier E. Sierra. "Urban traffic flow prediction techniques: A review." Sustainable Computing: Informatics and Systems 35 (2022): 100739.
3. Zhang, Yi, Shuwang Yang, and Hang Zhang. "Research on urban traffic industrial management under big data: Taking traffic congestion as an example." Journal of Advanced Transportation 2022 (2022).
4. Chen, Gen, and Jiawan Zhang. "Applying Artificial Intelligence and Deep Belief Network to predict traffic congestion evacuation performance in smart cities." Applied Soft Computing 121 (2022): 108692.
5. M Saleem, Muhammad, Sagheer Abbas, Taher M. Ghazal, Muhammad Adnan Khan, Nizar Sahawneh, and Munir Ahmad. "Smart cities: Fusion-based intelligent traffic congestion control system for vehicular networks using machine learning techniques." Egyptian Informatics Journal 23, no. 3 (2022): 417-426.
6. Xu, Yuanbo, Xiao Cai, En Wang, Wenbin Liu, Yongjian Yang, and Funing Yang. "Dynamic traffic correlations based spatio-temporal graph convolutional network for urban traffic prediction." Information Sciences 621 (2023): 580-595.
7. Ouallane, Asma Ait, Assia Bakali, Ayoub Bahnasse, Said Broumi, and Mohamed Talea. "Fusion of engineering insights and emerging trends: Intelligent urban traffic management system." Information Fusion (2022).
8. Park, Roy C., and Ellen J. Hong. "Urban traffic accident risk prediction for knowledge-based mobile multimedia service." Personal and Ubiquitous Computing (2022): 1-11.
9. Modi, Shatrughan, Jhilik Bhattacharya, and Prasenjit Basak. "Multistep traffic speed prediction: A deep learning based approach using latent space mapping

considering spatio-temporal dependencies." Expert Systems with Applications 189 (2022): 116140.

10. Modi, Yash, Ridham Teli, Akshat Mehta, Konark Shah, and Manan Shah. "A comprehensive review on intelligent traffic management using machine learning algorithms." Innovative infrastructure solutions 7, no. 1 (2022): 128.

11. Kashyap, Anirudh Ameya, Shravan Raviraj, Ananya Devarakonda, Shamanth R. Nayak K, Santhosh KV, and Soumya J. Bhat. "Traffic flow prediction models–A review of deep learning techniques." Cogent Engineering 9, no. 1 (2022): 2010510.

12. Jin, Junchen, Dingding Rong, Tong Zhang, Qingyuan Ji, Haifeng Guo, Yisheng Lv, Xiaoliang Ma, and Fei-Yue Wang. "A GAN-based short-term link traffic prediction approach for urban road networks under a parallel learning framework." IEEE Transactions on Intelligent Transportation Systems 23, no. 9 (2022): 1618516196.

Plagiarism Report

Traffic-Sense Harnessing Interpretable Machine Learning to Revolutionize Urban Traffic Congestion Prediction

ORIGINALITY REPORT

20%	16%	14%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to M S Ramaiah University of Applied Sciences Student Paper	8%
2	ak-tyagi.com Internet Source	1%
3	Submitted to De La Salle University Student Paper	1%
4	www.researchgate.net Internet Source	1%
5	ouci.dntb.gov.ua Internet Source	1%
6	iarjset.com Internet Source	<1%
7	www.mdpi.com Internet Source	<1%
8	Submitted to Kingston University Student Paper	<1%