LEVERAGING ARTIFICIAL INTELLIGENCE IN MODERN SUPPLY CHAIN

BY

NG QIAO YING

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS) INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

June 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I extend my deepest gratitude to my supervisor, Dr. Abdulkarim Kanaan Jebna, for the privilege of undertaking this IT project. His steady guidance, thoughtful feedback, and unwavering support have shaped a pivotal step in my journey toward a career in the IT field. I am especially grateful for the way he kindled my interest in machine learning and encouraged me whenever challenges arose; his insight and belief in my work were instrumental to the successful completion of this project.

My heartfelt thanks go to Artem for steadfast support, patience, and love, which kept me grounded and resilient during difficult moments. Your encouragement made all the difference and gave me the strength to keep moving forward.

Finally, I am profoundly grateful to my parents and family for their unconditional love and constant encouragement. Their belief in me has been a continual source of strength and motivation throughout this project.

ABSTRACT

This project addresses the challenge of last-mile delivery delays caused by urban traffic congestion by creating a smart route optimization system that combines the traffic prediction with classical pathfinding. A synthetic dataset was generated to simulate urban traffic flows, and a Long Short-Term Memory (LSTM) model was trained to forecast short-term congestion patterns. These predictions were converted into congestion factors and applied as dynamic weights within Dijkstra's algorithm to compute adaptive delivery routes. A Streamlit-based dashboard was designed to visualize model performance, predicted traffic conditions, optimized routes, and system-level evaluations in a simulated real-time environment. Evaluation results demonstrated that the LSTM model achieved reliable short-term forecasts, outperforming a baseline by more than 25% in error reduction, while the congestion-aware routing consistently avoided heavily congested edges. The prototype validates the feasibility of combining predictive analytics with graph-based optimization, offering a practical foundation for enhancing efficiency and reliability in last-mile logistics operations.

Area of Study (Minimum 1 and Maximum 2): Artificial Intelligence, Supply Chain

Keywords (Minimum 5 and Maximum 10): Traffic Prediction, Route Optimization, Smart Logistics, LSTM Model, Dijkstra's Algorithm

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
LSTM	Lont-Short Term Memory
ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
GCN	Graph Convolutional Networks
RNN	Recurrent Neural Networks
ARIMA	Autoregressive Integrated Moving Average
SVM	Support Vector Machines
RF	Random Forest
GRU	Gated Recurrent Units
CNN	Convolutional Neural Networks
GCN	Graph Convolutional Networks
k-NN	k-Nearest Neighbor
SVR	Support Vector Regression
MLP	Multi-Layer Perceptron
DSRC	Dedicated Short-Range Communication
MAPE	Mean Absolute Percentage Error
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
GPS	Global Positioning System
ITS	Information Technology System
BFS	Best-First Search
ASP	All Shortest Path
CRISP-DM	Cross-Industry Standard Process for Data Mining

Chapter 1: Introduction

Global supply chains are changing rapidly as markets expand, and customers demand faster, more reliable services. Traditional ways of managing logistics are finding it harder to keep up with challenges such as increasing delivery volumes, rising costs, and the need to stay flexible. In recent years, technologies like Artificial Intelligence (AI) have started to improve supply chains by enabling smarter decision-making and faster responses to change conditions. While warehouses have already adopted many AI-driven solutions, the logistics sector is still adapting, especially after disruptions caused by the COVID-19 pandemic and international conflicts [1]. These events exposed weaknesses in conventional logistics systems and highlighted the need for greater flexibility and resilience.

A key challenge in modern logistics is last-mile delivery which is the final step of getting goods to customers. This stage often has the highest costs and the most complexity. Traffic congestion makes last-mile delivery even harder by causing delays, raising costs, and reducing reliability [2]. In Malaysia, this problem is growing rapidly due to urbanization and the boom in e-commerce, particularly in cities such as Kuala Lumpur and Johor Bahru [3]. Studies also show that the way delivery vehicles behave, such as searching for parking or double-parking, adds to congestion and pollution. This means that last-mile delivery choices affect not only companies but also the wider traffic network and city environment [4].

Current route optimization methods mostly rely on static, historical data, which cannot handle the unpredictable nature of urban traffic. Accidents, sudden weather changes, and unexpected surges in traffic can quickly make pre-planned routes inefficient. To deal with this, short-term traffic forecasting has become increasingly important. By predicting near-future traffic patterns, it is possible to plan routes dynamically, manage congestion in real time, and reduce the risk of delays [5]. In last-mile logistics, accurate short-term forecasting makes deliveries faster, lowers costs, and improves overall service reliability.

1.1 Problem Statement and Motivation

The realisation of dependable last-mile delivery in the world of urban logistics is a strategic necessity; however, recurrent and, at other times, unpredictable gridlocks continue to affect the working efficiency [2]. There are still many routing plans that are based off the long-term plans that have been developed based on previous patterns over time, thus not capturing the short-term changes that are experienced in modern urban settings [2]. Short-term disturbances due to accidents, sudden changes in weather or unexpected demand peaks can derail timetables, swell operational expenses as well as damaging consumer satisfaction [2]. This is especially relevant in overcrowded centres like Kuala Lumpur and Johor Bahru [3].

Despite the advancements in the sphere of logistics technologies, there is a glaring gap in the technological approach of combining predictive traffic analytics with real-time routing mechanisms. The existing systems mostly do not have the ability to predict the imminent congestion or dynamically adjust the routes and hence they create suboptimal results in the dynamic changing environment of traffic.

To reduce this drawback, the current project presents a traffic-prediction model that is based on Long Short-Term Memory (LSTM) networks. Predicting the immediate state of traffic and then combing the prediction with the Dijkstra algorithm of the dynamic rerouting, the delivery paths can avoid the expected bottlenecks proactively. This approach is expected to support the efficiency of deliveries, reduce the negative impact of operational disturbances, and provide smarter, adaptive last-mile logistics.

1.2 Objectives

The purpose of this project is to design an AI solution with route-optimization that can combine short-term traffic prediction and classical path-finding methods to enhance last-mile delivery. The primary objectives are:

1. To design and implement a time-series predictive model using Long Short-Term Memory (LSTM) networks capable of forecasting short-term traffic congestion patterns in urban environments.
2. To develop a dynamic traffic-aware route optimization system that integrates predicted traffic conditions into Dijkstra's algorithm to compute adaptive delivery paths.
3. To design a traffic-aware dashboard that visualizes predicted traffic conditions, optimized routes, and key system performance metrics in a simulated real-time environment.
4. To validate the applicability of the proposed system in last-mile delivery logistics through simulation, demonstrating how traffic prediction and congestion-aware routing can reduce travel time and improve delivery reliability.

The current project will prove the relevance of a proposed system in last-mile delivery logistics with the help of simulation and, thus, the way that the prediction of the traffic can be used alongside the congestion-aware routing to minimise the travel time and increase the reliability of the delivery.

1.3 Project Scope and Direction

This development uses the application of artificial intelligence to maximise traffic-aware routes in last mile delivery. It builds a model that can forecast short-term traffic jamming and uses the projections to plan the best delivery paths in a virtual setting. Target stakeholders include logistics companies and operators of delivery services are interested in enhancing the efficiency of delivery and strengthening operational planning.

A structured traffic dataset is used to train a Long Short-Term Memory (LSTM) model in order to predict congestion patterns in the short-term. The delivery network is modelled as directed graph with the NetworkX Python library, with edges weights (predicted travel times).The algorithm of Dijkstra is then used to estimate the most efficient routes to deliveries which are below dynamic traffic conditions.

The lightweight visualization dashboard is developed to present predicted congestion levels, optimized routes, and system performance metrics. The project is limited to a simulation setting using synthetic or pre-processed traffic data and is intended as a proof of concept rather than a fully deployed system. Real-time GPS tracking, integration with commercial logistics platforms, and large-scale deployment are outside the current scope and are identified as directions for future improvement.

1.4 Contributions

The project will contribute to the development of intelligent logistics solutions by integrating short-term traffic prediction with dynamic route optimization for last-mile delivery. The system demonstrates how a LSTM model can be used to forecast near-future congestion and how these predictions can be applied within Dijkstra's algorithm to generate adaptive delivery routes. A visualization dashboard further connects prediction and routing, allowing users to observe traffic forecasts, optimized paths, and system performance in an accessible way.

From a practical perspective, the project highlights how predictive traffic-aware routing can support logistics companies in reducing delays, improving operational planning, and enhancing route flexibility. For end users, these improvements translate into faster and more reliable deliveries. Even small increases in predictive accuracy can have meaningful impacts on large-scale supply chain performance, especially in congested urban environments.

Although the system is developed and tested in a simulated environment using synthetic traffic data, it provides a proof of concept that demonstrates feasibility and sets the groundwork for future applications. With further refinement and integration of real-world traffic datasets, the framework could be extended into commercial logistics platforms to strengthen last-mile delivery efficiency.

1.5 Report Organization

This report is organized into seven chapters. Chapter 1 introduces the project background, problem statement, objectives, scope, and contributions. Chapter 2 reviews related literature on traffic prediction, route optimization, and existing systems. Chapter 3 explains the methodology based on CRISP-DM and presents the system design. Chapter 4 describes the preliminary work and setup, while Chapter 5 details the system implementation including dataset preparation, LSTM training, Dijkstra integration, and dashboard development. Chapter 6 presents the evaluation of the model and system, discusses limitations, and assesses the objectives. Finally, Chapter 7 concludes the study and provides recommendations for future improvements.

Chapter 2: Literature Review

2.1 Introduction

The chapter is a review of the available literature on two key fields that will be essential in this project which include traffic prediction and route optimization. The requirement to adapt fast has increased as the logistics systems are becoming more complex. In order to ensure higher efficiency of the last-mile delivery, one should estimate the short-term traffic situation and organise the routes that can adapt to alterations at the road.

2.2 Background Study

2.2.1 Prediction of Traffic

Traffic prediction refers to the act of predicting the future road conditions, including the number of vehicles, speed of the vehicles, and degree of congestion. This is achieved through review of the past traffic and the current data. Traffic prediction is applied in city planning, traffic control and logistics. In the case of delivery services, it will enhance the last-mile operations by minimising delays and increasing the reliability of the delivery [6].

2.2.2 Temporal Patterns and Short-Term Traffic

According to Vlahogianni [5], short-term traffic prediction refers to the prediction of road conditions in the coming minutes up to some hours. This is highly essential in real time traffic evolution, congestion alleviation and efficient route delivery planning.

Traffic patterns are also based on time. An example would include rush hours, weekend variation and some unforeseen occurrences such as accidents or poor weather. These trends need to be comprehended when developing traffic forecasting models which are applicable in the real world [7].

2.2.3 Artificial Intelligence

As IBM [7] indicated, the Artificial Intelligence (AI) is a branch of computer science that aims at developing systems which can accomplish tasks or functions that are typically performed by human intelligence. Table 1 consists of learning by experience, problem solving, decision making and even language comprehension. Nowadays, the three primary domains are used to drive modern AI: machine learning (ML), deep learning (DL) and natural language processing (NLP). These enable the computers to enhance their performance with time without requiring reprogramming each time a new task is to be performed.

AI has been used as a such tool in traffic prediction, as explained by Y. Yu [6] AI techniques, in particular, ML and DL are highly effective since they can automatically discover patterns in large traffic data sets. These datasets can also have both space (road networks, locations) and time (traffic flow by hours, days, or weeks). The short-term dynamics of traffic can also be anticipated with the help of AI models, designed to accommodate short-term changes of traffic, such as congestion, without the need to study past patterns. This capability of learning through data and responding in real time enables AI to be of particular use in developing smart logistics that can react fast and effectively to the issues in the urban traffic [4].

2.2.4 ML and DL

Machine Learning (ML) is a subset of AI that enables systems to learn through data and enhance their work without the need to programme them step-by-step. According to the explanation of IBM [7], the predictive models constructed by the ML algorithms identify patterns and patterns in data. To illustrate, with previous data of the traffic flows an ML model can learn the dynamics of the number of vehicles at each hour of the day and use the information to make predictions. The model can be improved over time as time it takes more data and thus will be more accurate and reliable in making decisions.

Machine learning Deep Learning (DL) is a more sophisticated form of machine learning. It employs the artificial neural networks that are multi-layered, and they are made to replicate the way human brain processes information. Deep learning has the capability of automatically

identifying features and patterns on extremely large and complex data. In contrast to traditional ML, where the process of feature selection is frequently manual, the process of feature selection is learned by DL models. Due to this, DL is best applicable to areas where processing of complex data like image recognition and speech processing and, more recently, traffic prediction is involved.

Long Short-Term Memory (LSTM) networks and Graph Convolutional Networks (GCN) networks have emerged as the leading DL methods in the transportation and logistics field to perform traffic forecasting tasks [6]. The LSTM networks are also designed in such a way that they capture the time dependence hence better placed in predicting changes in traffic between minute to minute, or hour to hour. In the meantime, GCNs can capture geographical correlations among roads and intersections and are applicable in the modelling of an entire city traffic network. These strengths, together with others, allow the DL methods to process the dynamic and spatio-temporal characteristics of traffic data that is crucial in enhancing the efficiency of the last-mile logistics operations.

2.2.5 Long Short- Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) networks: This is a specific form of Recurrent Neural Networks (RNNs) that was introduced by (Hochreiter and Schmidhuber, 1997) to combat the vanishing gradient problem when training sequential data [10]. The LSTM cells provide a memory cell architecture that enables the network to store, forget and retrieve information at different time steps selectively, and therefore model long-term temporal dependencies. An LSTM unit has an internal operation that is regulated with the help of three key gates:

- **Forget Gate**: Determines which information from the previous cell state should be discarded.
- **Input Gate**: Decides which new information should be stored in the current cell state.
- **Output Gate**: Controls how much of the cell state should be exposed to the next hidden state.

These gates enable LSTM networks to store relevant information in long sequences and reject irrelevant data and therefore they are especially efficient in time-series prediction tasks like short-term traffic prediction.

2.2.6 Dynamic Route Optimization

Dynamic route optimization is the process of continuously adjusting vehicle routes by incorporating real-time or predicted traffic details into the pathfinding algorithm. Unlike static routing, which uses fixed edge weights and pre-planned paths, dynamic optimization updates travel times as conditions evolve, allowing the system to anticipate congestion and select forward-looking routes. [12] demonstrate this by modifying Dijkstra's algorithm with dynamically weighted travel times obtained from traffic flow predictions, showing that adaptive routing can significantly reduce travel time compared to conventional static approaches.

2.2.7 Dijkstra's Algorithm

According to Wijaya, D. [13], the algorithm provided by Dijkstra is a technique of searching the shortest path between two points in a weighted graph. It looks at the least total cost including distance, time or price between a starting point and the rest of the points by selecting the nearest node at a time and updating the path costs. When the shortest path to a node has been located it is finalised and not modified. The subject of the final result using Dijkstra Algorithm is shown in Figure 2.4. Dijkstra algorithm provides good results by taking into account the weights of the connexions between nodes.

Procedural Steps:

1. The first step is to set the distance to the source node to zero and set all the other nodes to infinity.
2. Mark all nodes as unvisited.
3. Choose the node that has never been touched which has the least distance.
4. In the case of the present node, look at all the neighbouring nodes; should a shorter path to one of the neighbours be found, then modify the distance attribute of that neighbour.
5. After analysing all the neighbours, label the current node as visited thus confirming its minimum path
6. Then repeat the process, take the next unvisited node having the shortest distance.

7. Stop being a destination node has been visited or when all reachable nodes have been visited.

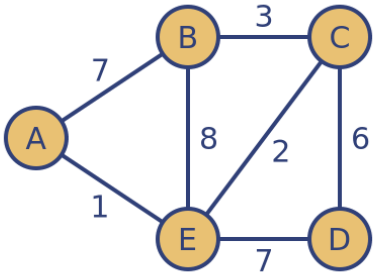


Figure 2.1: Process of Dijkstra's Algorithm

2.3 Research on AI Models for Traffic Prediction

Y. Yu [6] introduces a detailed survey of the artificial intelligence models that can be used to predict traffic, both conventional machine learning models and recent deep-learning models. The most commonly used techniques, including the Autoregressive Integrated Moving Average (ARIMA), were widely used in time-series prediction but had a gross lack of ability to capture highly nonlinear and complex dynamics of traffic flow. Traditional machine-learning models, such as Support Vector Machines (SVM) and Random Forests (RF) had a greater degree of flexibility and accuracy compared to ARIMA but often were not able to extend the long-term temporal relationships of sequential traffic information.

Deep-learning processes have proven to present distinct benefits when it comes to traffic forecasting over the past few years. Structures like the Long Short-Term Memory (LSTM) networks and the Gated Recurrent Units (GRU) have been widely used in order to capture the temporal relationships in traffic sequences. Convolutional Neural Networks (CNNs) models have been used to learn spatial patterns in grid-based traffic maps, and Graph Convolutional Networks (GCNs) have been used to conceptualize traffic systems as graphs to learn spatial dependencies across road segments and intersections. More advanced methods employ the attention systems and Transformer based systems, which dynamically detect and rank the salient features in traffic information. Hybrid models such as GCNLSTM and CNNLSTM have also been suggested that can simultaneously learn spatial and time-varying features, which can have a significant potential of better prediction accuracy.

2.3.1 Work Proposed

Y. Zhong (9) designed a Short-term traffic-congestion prediction model using a hybrid architecture of the Long Short-term Memory (LSTM) neural-network. To reduce the raw urban traffic data to salient variables, that is, vehicle speed, traffic flow, number of lanes, and a congestion index based on these parameters, the model preprocessed the raw traffic data and provided a coefficient of congestion to work out the discrete categories containing heavy congestion to free flow. The sequential learning input spatial and temporal features were obtained by spatial and temporal structuring of the inputs, and the network used two hidden layers with a sigmoid activation and an output layer with a softmax activation to classify the multi-class congestion. The approach using historical traffic data collected in Shenzhen using the city-wide surveillance system by Huawei recorded high predictive accuracy, with a Precision of 0.984, Recall of 0.975, and an F1 -Measure of 0.979. These results affirm the effectiveness of LSTM-based models in the short-term prediction of congestion in urban environments in the context of capturing temporal dynamics of traffic.

A comparative evaluation of KNN (k-nn), LSTM and Transformer was carried out by J. Jang (10) to predict short-term travel time based on dedication short-range communication (DSRC) traffic information on suburban roads in arterial roads in Korea. Although it did not present any new architecture, the experiment showed that deep-learning methods have stronger performance compared to traditional ones. Both the LSTM and Transformer models demonstrated higher predictive accuracy compared to k -NN; in particular, the LSTM model has a Mean Absolute Percentage Error (MAPE) of 11.7 that is significantly lower than the baseline. Such findings emphasise the usefulness of LSTM networks in predicting serial traffic data and indicate their potential usefulness in the context of dynamic traffic control and logistics routing models.

M. Gmira (11) presented a travel speed prediction model that is specific to the home-delivery business and relies on supervised learning models on massive GPS dataset that included more than 2.5 million delivery points and 200,000 routes. The approach employed preset steps of preprocessing, clustering of delivery routes and imputation of missing data to

improve the quality of data. They tested several learning algorithms: k-NN, Support Vector Regression (SVR), Multi-layer Perceptron (MLP), and LSTM networks; on several datasets, the LSTM algorithm showed the best results and lowest root mean square error (RMSE) and mean absolute error (MAE). As an example, the LSTM achieved an RMSE of 6.03 km -1 h -1 and a MAE of 4.66 km -1 on computation DB1. These results prove that LSTM models are effective in short-term travel-speed forecasting, especially in urban logistics networks, and they can be solid competitors in improving efficiency in delivery in last-mile operations.

2.3.2 Evaluation metrics and Results

Jinhwan Jang (10) compared standard error metrics the predictive performance of k- nearest neighbour (k-NN), Long short-memory (LSTM), and Transformer models, in terms of the standard error metrics, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The LSTM model had much less error values than k-NN and was almost parallel in performance with Transformer model as demonstrated in Figure 2.2. A paired t-test was conducted to test the statistical significance of the differences between the MAPE values, and the results are shown in Figures 2.2 and 2.3. The received p -values (<|human|>The received p -values (<|human|>The obtained p -values (<|human|>The received p -values (<|human|>The received p -values (<|human|>The effectiveness of deep-learning models in short-term travel-time predictions is supported by the obtained p -values (less than 0.05).

Algorithm	k-NN	LSTM	Transformer
MAE (sec)	31.8	29.3	27.6
RMSE (sec)	41.2	38.0	37.3
MAPE (%)	12.4	11.7	10.3

Figure 2.2: Comparison of k-NN, LSTM, and Transformer models [9]

Statistic	k-NN	LSTM	LSTM	Transformer	k-NN	Transformer
Mean	12.4	11.7	11.7	10.3	12.4	10.3
Variance	95.2	117.9	117.9	83.7	95.2	83.7
t-value	2.53		3.18		5.26	
p-value (one-sided)	0.0058		0.0007		8.59E-8	
Number of sample	1137					

Figure 2.3: MAPE differences between models [9]

M. Gmira [11] has made a comparative analysis of a few models of supervised learning to predict short term travel speed based on real world delivery data. Models evaluated were Long Short-Term Memory (LSTM) network, Multi-layer Perceptron (MLP) network, Support vector regression (SVR), and k- nearest neighbour network (k-NN). The model efficacy was measured

using Root Mean Squared error (RMSE) and Mean Absolute error (MAE) which are standard measures of regression-based forecasting. The LSTM model recorded the lowest RMSE and MAE in all the conditions of input-feature combinations and therefore it was the best over the other methods, as shown in Figure 2.3. These findings highlight the good capabilities of LSTM networks in generating correct short-duration traveling speed forecasts, especially in the context of a logistics and transportation environment.

Comparison of alternative models.

Test instance	Model	Metric	Input vector I	Input vector II
DB0	LSTM	RMSE	**17.22**	**10.84**
		MAE	**22.86**	**12.84**
	MLP	RMSE	21.92	15.17
		MAE	29.57	17.40
	SVR	RMSE	22.85	17.03
		MAE	31.30	14.68
	k-NN	RMSE	25.00	18.51
		MAE	38.13	37.91

Figure 2.4: LSTM, MLP, SVR, and k-NN models comparison for travel speed prediction

2.3.3 Challenges

According to Vlahogianni [5], there are a number of current challenges in the short-term traffic forecasting studies. Top among them is the need to have so-called algorithms that are capable of responding to unforeseen disruption like accidents or extreme weather changes. It also highlights the creation of the models which will be able to make future predictions of traffic not on a limited range of highway situations, but on a more general level related to the urban setting and the network. Issues that surround data are still relevant especially to its solving, quality, and fusion of various data such as GPS, roadside sensors and mobile data. Other research directions of interest involve precise modelling of the dynamics, and generally high quality of model selection and generalisation, and methods of integrating predictions due to disparate models. The explainability of complex AI models, their incorporation into intelligent transportation systems (ITS) and the general use of AI to manage the traffic more adaptively are also put as the most significant directions of the future work.

Even though Y. Zhong [9] demonstrated a high predictive performance on an LSTM-based congestion forecasting quarter, multiple challenging methodological issues were apparent. The process of cleaning, normalising, and engineering features on raw traffic datasets was very time-consuming. The requirement to organise the levels of congestion necessitated the development of a congestion coefficient that was unique hence increasing complexity in the training process. Moreover, the model concentrated more on time prediction at various intersections and therefore it was limited in generalisation to extended networks. Lastly, the methodology was based on the quality and complete datasets of traffic, which cannot be guaranteed across different application settings.

Recent comparative literature has shown that although Long Short-term Memory (LSTM) networks continue to be effective in the capture of temporal traffic data, more recent models including Transformers have been shown to have slightly better performance in very a short time (Less than an hour) task [10]. However, LSTM is still being widely used because of its accuracy, interpretability, and relative simplicity. There are, however, challenges in both capturing small scale traffic variations on very short horizons and coming up with an optimum trade-off between prediction accuracy and computing cost in the real world.

2.3.4 Summary of Literature Review

Author(s)	Used Model	Dataset	Results
Y. Yu [6]	ARIMA, SVM, RF, LSTM, GRU, CNN, GCN, Transformers, Hybrids	Various Conceptual review	Deep learning (LSTM, GCN, Transformer) better captures spatio-temporal dependencies than classical models.
Y. Zhong [9]	Hybrid LSTM (2 hidden layers, sigmoid + softmax)	Urban traffic data from Shenzhen (Huawei surveillance)	LSTM model achieved high accuracy for congestion classification (F1: 0.979).
Jinhwan Jang [10]	k-NN, LSTM, Transformer	DSRC traffic data (suburban roads, Korea)	LSTM significantly outperformed k-NN (MAPE: 11.7%); Transformer showed slightly better performance.
M. Gmira [11]	k-NN, SVR, MLP, LSTM	Real-world GPS data	LSTM outperformed all models with lowest RMSE (6.03 km/h) and MAE (4.66 km/h).

Table 2.1: LR Summary table for Traffic Prediction

2.4 Proposed Work for Dijkstra's Route Optimization

A comparative evaluation of the cargo route optimisation in Indonesia was done in the study by Wijaya in which the five short-path algorithms used are a Greedy, Best-First Search (BFS), Dijkstra, A star, and Floyd-Warshall. Instead of suggesting a new methodology, the research focused on determining which algorithm was better by comparing performance, on a list of different parameters, such as cost, distance, duration, and user rating. The Python programming language was used to create implementations, which were connected with a RESTful interface, and the algorithms were tested on a dataset of inter-city freight routes. The main metrics of performance were runtime and routing accuracy, at different complexities (1 3 hops). The results showed that A* did the fastest time and Dijkstra always generated the correct path. On the other hand, Greedy and BFS showed lower reliability, in the form of looping behaviour and path generation of invalid paths.

2.4.1 Results and Evalution Metrics

Figure 2.5 summarises the relative performance of the five shortest-path algorithms and conspicuously shows that there are great differences in the computation time and the accuracy of the solutions. A + provided the best overall performance with an average run time of 7 056.67 ms and a 100-percent accuracy to find all shortest paths (ASP) in all the trial conditions (1- 2- 3-hop). The algorithm made by Dijkstra was similarly perfect, but the computational effort was slightly higher with a mean of 7404.34ms. Floyd-Warshall was as accurate as the two algorithms above but was less efficient as it took 8.97867ms on average. Conversely, Greedy algorithm had the worst performance as it had the longest run time of 11072.5 ms and gave unreliable results that contained wrong or repeated paths. The BFS algorithm had the shortest run time of 6 734.67 ms although it did not always recover the optimal shortest paths. Taken together, these findings prove that the A* algorithm provides the best trade-off between performance and accuracy in cargo route optimisation, but Dijkstra is still a reliable option in case the most essential factor in the picture is the accuracy of the route.

Algorithm	Hops	Status	Runtime (ms)	Average runtime (ms)	Description
Greedy	1	ASP	9,089	11,072.5	-Some results do not match the route
	2	ASP	8,574		and there is a possible route not found
	2	ASP	9,224		or looping forever
	3	N-ASP	17,403		-Longest average runtime
BFS	1	N-ASP	6,959	6,734.67	-Some results do not match the route
	2	ASP	6,288		-Shortest average runtime
	3	N-ASP	6,957		
Dijkstra's	1	ASP	6,986	7,404.34	-All recommendations are the best route
	2	ASP	8,106		-Longer average runtime than A*star
	3	ASP	7,121		
A*	1	ASP	6,049	7,056.67	-All recommendations are the best route
	2	ASP	8,067		-Shortest average runtime
	3	ASP	7,054		
Floyd-Warshall	1	ASP	10,971	8,978.67	-All recommendations are the best route
	2	ASP	8,363		-Longer average runtime than A*star
	3	ASP	7,602		

Figure 2.5: The optimization results [11]

2.4.2 Stregth and Weakness

A number of strengths can be identified in the study by Wijaya [13]. Its main contribution lies in the fact that it includes in one framework a comparative study of five commonly used shortest-path algorithms, Greedy, Best-First Search (BFS), Dijkstra, A, and Floyd-Warshall. This comprehensive analysis provides a balanced insight on algorithmic performance applicable in the cargo route optimisation. Moreover, that most of the presented practical criteria like the price, the distance, customer rating, and the delivery time improve the real-life applicability of the analysis. The fact that it was implemented in Python and based on a REST API highlights the relevance of the implementation in logistical realms. The study provides a helpful benchmark on choosing the suitable algorithms by comparing the accuracy of the runtime and route results under a variety of hop situations. The results show that A is the fastest, however, the algorithm created by Dijkstra reliably provides valid and optimal routes, and this is why it is the best suitable algorithm.

However, the paper has a few limitations. It was tested in a simulated environment and not in the dynamic environment of real time changes like live traffic or dynamic road conditions, thus limiting the ability to implement it in adaptive routing melting point conditions. Besides, the authors have failed to create or improve any new algorithm, instead focusing on well-known techniques. The results potentially may be limited in the generalisability of the findings to larger or more complicated logistics networks due to the restricted sample of the dataset, which consists of intercity cargo routes in Indonesia. The other disadvantage is that Greedy and BFS at times gave invalid or subsequently repeating routes, thus reducing their useful trustworthiness. Lastly, machine learning and predictive elements were not incorporated in the framework, which may have made it more flexible and responsive in changing conditions.

2.5 Proposed Work for Combination of LSTM Model and Dijkstra's Algorithm

Most existing research treats traffic prediction and route optimization as separate problems. Studies on forecasting often emphasize improving model accuracy through deep learning methods such as LSTM or Transformer networks, while routing research focuses on algorithms like Dijkstra or A* for computing shortest paths. However, fewer studies combine these two areas by embedding predictive analytics directly into routing frameworks.

Some recent works demonstrate this integration. For example, [14] developed a traffic guidance system that combines LSTM forecasts with a modified dual-tracking Dijkstra algorithm, which simultaneously considers travel distance and air quality to generate routes that avoid both congestion and highly polluted areas. Similarly, [12] proposed a dynamically weighted Dijkstra algorithm where predicted travel times, obtained through traffic flow theory models, are used as edge weights to compute time-dependent shortest paths.

These studies highlight the feasibility of integrating traffic prediction into routing optimization, but they target multi-objective such as time and pollution or theoretical flow-based approaches. In contrast, the present study focuses on applying LSTM-based short-term traffic forecasting directly as input to standard Dijkstra's algorithm, with the specific aim of improving adaptive route planning for last-mile logistics.

2.6 Review of Existing Systems

Google Maps [15] is one of the most popular navigation systems used worldwide. It helps drivers not only by providing directions but also by showing traffic conditions and estimating travel times. Google Maps gathers information from smartphones, local transport authorities, and past traffic records to estimate how busy the roads will be. Based on this information, it suggests the best routes by considering travel time, distance, and restrictions such as tolls. The interface is user-friendly, with roads color-coded to show traffic flow: green for smooth traffic, orange for moderate congestion, and red for heavy traffic. It also provides alternative routes and updates travel times as conditions change. The strength of Google Maps lies in its massive data scale and reliability. However, because it is a closed and proprietary system, its methods cannot be customized or adapted for research.

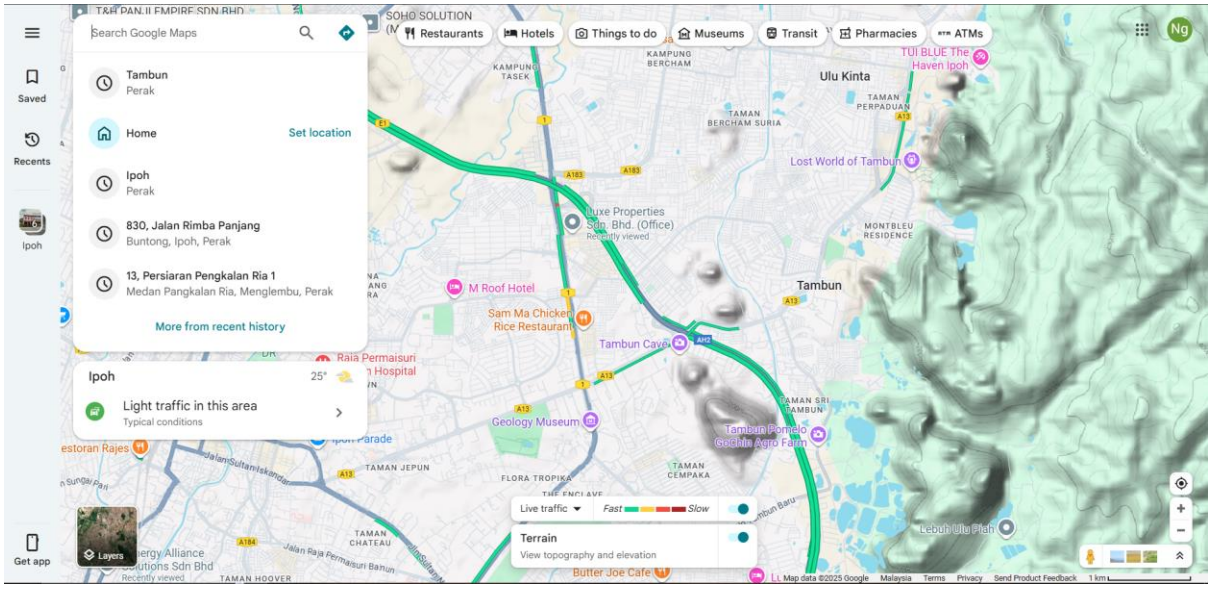


Figure 2.6: Traffic Dasboard by GoogleMaps [16]

Aimsun Live [17] is a commercial system used mainly by transport authorities and city managers. It provides real-time traffic simulation and prediction to help control traffic across entire cities. Unlike research prototypes that only test models, Aimsun Live is a fully operational platform that collects live traffic data from sensors, predicts how traffic will flow, and detects unusual events such as accidents. The information is displayed on a dashboard with a city map showing congestion levels, traffic intensity, and incidents. Additional charts allow staff to compare live and historical data. If an incident occurs, the system highlights it

immediately so operators can take quick action. Aimsun Live's strength is its ability to provide large-scale traffic management that is clear enough for non-technical staff to use. However, it is focused on network-wide traffic control rather than personal route planning. It also relies on closed, proprietary methods, limiting how much it can be customized for research purposes.

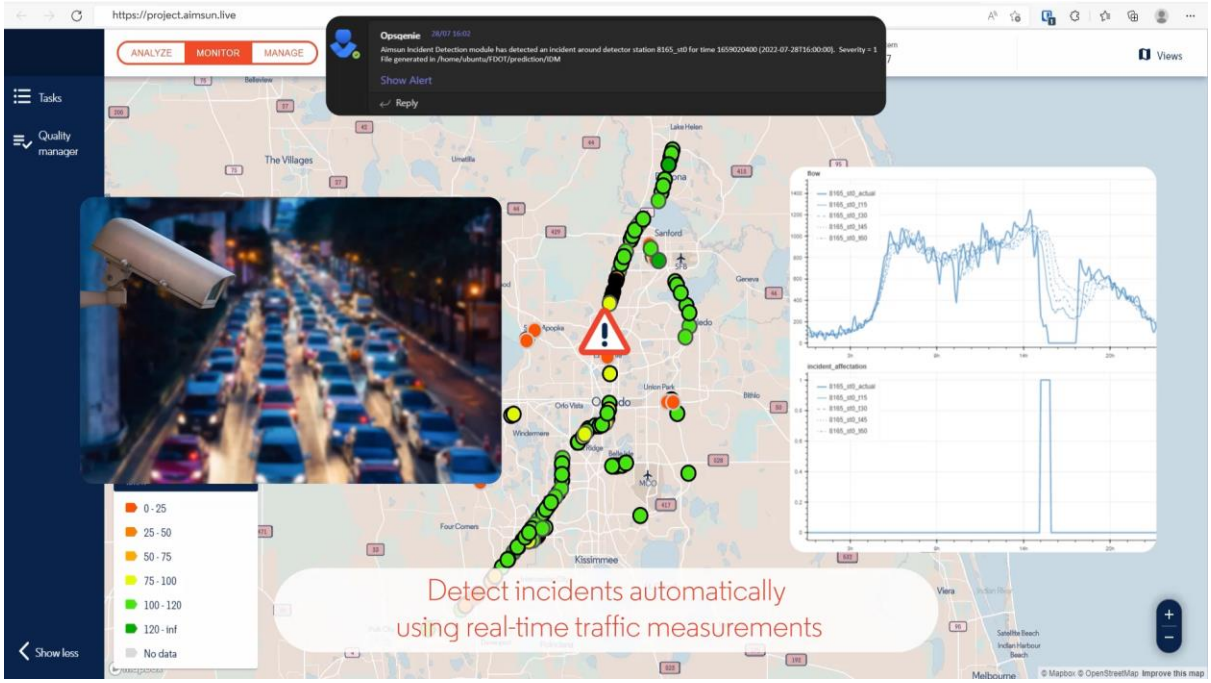


Figure 2.7: Traffic Dasboard by Aimsun [17]

AutoWiz Route Planning and Optimization [18] is a system designed for supply chain and logistics operations. It is mainly used by companies that manage fleets of vehicles, such as delivery services and courier companies. Through a web-based dashboard, managers can enter multiple delivery or pickup addresses, and AutoWiz automatically calculates the most efficient order of stops and routes for the drivers. The dashboard clearly displays all the planned routes on a map, making it easy for managers to monitor and adjust as needed. AutoWiz also supports practical logistics features such as handling multiple vehicles, setting delivery time windows, and balancing workloads between drivers. These functions help businesses reduce fuel costs, improve delivery speed, and increase customer satisfaction. The strength of AutoWiz is its focus on real-world supply chain problems and its easy-to-use interface. However, it mainly uses current road conditions and does not predict future traffic patterns.

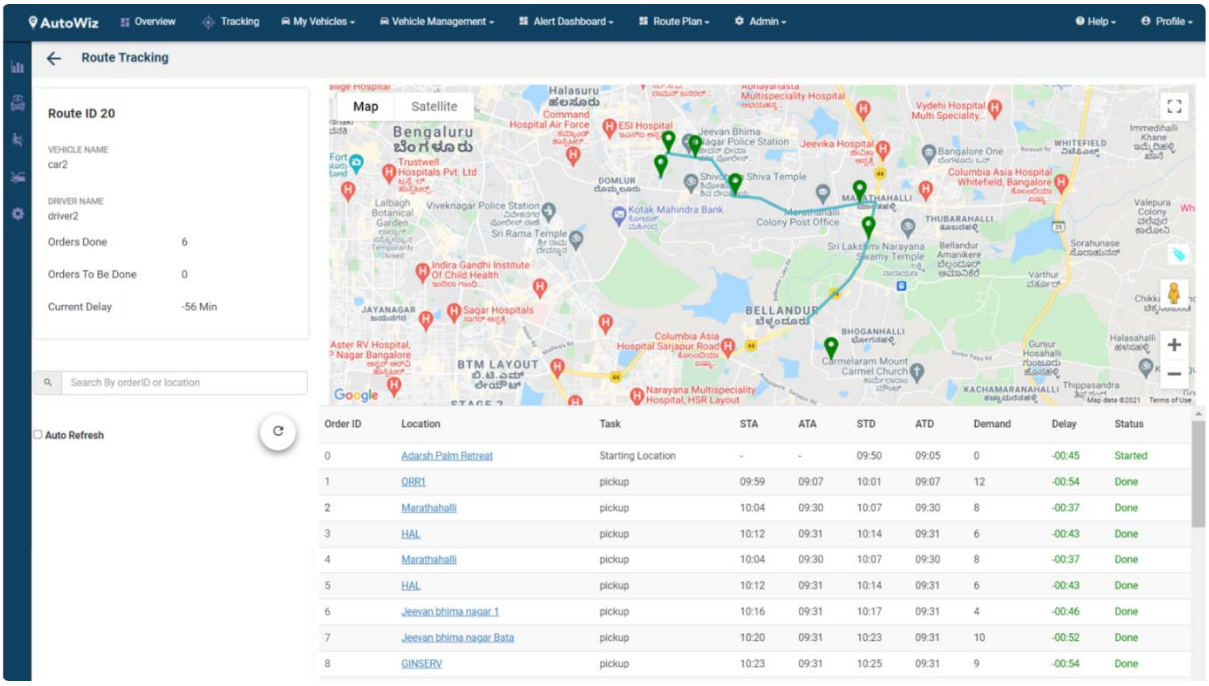


Figure 2.8: Routing Dashboard by AutoWiz [18]

2.6.1 Comparison Table

System	Users	Purpose	Strengths	Limitations
Aimsun [17]	Transport agencies and city traffic control centers	Real-time traffic simulation, prediction, and incident detection for whole cities	Strong for large-scale city traffic management, easy for non-technical staff, integrates prediction with monitoring	Focused on network-wide management, not individual route optimization, closed system with proprietary models
AutoWiz [18]	Logistics and supply chain companies	Plans and optimizes delivery or pickup routes for fleets	Practical for supply chain use, supports multiple vehicles, time windows, and workload balancing, easy-to-use	Relies on current road conditions only, does not forecast future traffic or adapt routes ahead of time

Google Maps [15]	General drivers and the public	Provides navigation with live traffic updates and dynamic route suggestions	Huge data coverage, reliable travel time estimates, user-friendly	Proprietary system, not customizable for research, designed for general use not specialized applications

Table 2.2:Comparison Table between Existing Systems

From this comparison, it is clear that existing systems such as Google Maps, Aimsun Live, and AutoWiz provide powerful solutions in their respective areas. Google Maps excels in public navigation with real-time traffic updates, Aimsun Live supports city-wide traffic management for authorities, and AutoWiz focuses on supply chain efficiency through route planning. However, these systems share common gaps: they are either proprietary and not customizable, or they do not emphasize forecasting future traffic conditions. The proposed LSTM–Dijkstra system addresses these gaps by combining traffic prediction with route optimization in an open and flexible framework. This makes it suitable for personalized and adaptive planning, particularly in supply chain operations where anticipating delays is critical.

Chapter 3: Proposed Method/Approach

The current project was conducted on the use of CRISP-DM methodological framework, which involves the stages of Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. The first stage was the determination of the presence of traffic congestion as an issue and clarification of its implications on route optimisation. The sequential training of the traffic data followed as a preparation step that was used as an input of a Long Short-Term Memory (LSTM) model to predict the short-term traffic. The resulting predictions were then combined with the Dijkstra algorithm to make traffic-aware routing and the rest of the analytical pipeline available through a dashboard allowing the visualisation of forecasts, congestion level, and optimal routes.

3.1 Methodology

3.1.1 Cross-Industry Standard Process for Data Mining (CRISP-DM)

CRISP-DM framework has been considered suitable as the main frame of reference when it comes to leading the subsequent stages of the model development, which is why its introduction to this study is justified. It makes sure that there is an orderly systematic way of approaching data-mining and machine-learning tasks. This test methodology has six consecutive steps that advance each other to increase the success of the project at the end. It has since become the most popular method of data-research, data-analytical and data-mining projects [19]. The schematic view of the CRISP-DM method is presented in Figure 3.1.

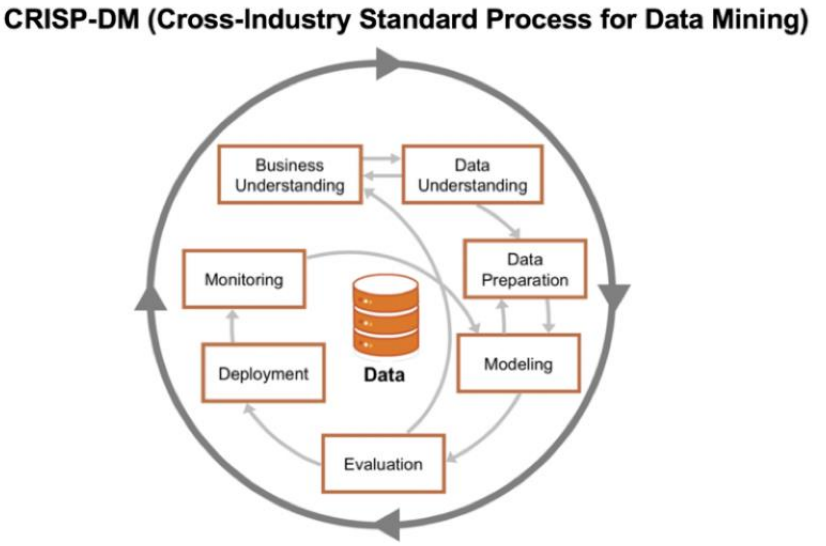


Figure 3.1 Crisp-DM Process Diagram

3.1.2 Business Understanding

The business underpinning step determines the theoretical basis of the research by defining the grand goals of the research with specific analytical targets. The main objective of this inquiry is to improve the efficiency of the last-mile delivery system in the Malaysian urban logistics environment, in which unpredictable traffic jams have become a continually encountered challenge. Traditional types of prospective routing that modern logistics are strategic with cannot react to such sharp changes in traffic routing patterns, which can often result in delays, high delivery costs, and the decreased level of customer satisfaction.

The project suggests a predictive framework, which represents a combination of a Long Short-Term Memory (LSTM) predictive model to forecast the short-term traffic with the Dijkstra algorithm to optimise dynamic routes. The key business goal includes the supply of logistics operators with real-time decision-making tools that can enhance delivery reliability, improve the flexibility of the operations and increase the sensitivity to congestion in the urban centres like Kuala Lumpur and Johor Bahru.

Placing the system into the context of the greater business environment is the challenge of fast growth of e-commerce, the diversity of traffic congestion perishes, and the need of projection of the logistics system, which is the subject of the proposed architecture the possibility to predict the result of better route planning. The resultant output is a prototype solution to integrate technical innovation with realistic industry needs and hence promote more efficient and reliable last-mile delivery services.

3.1.3 Data Understanding

This phase is an important step because it sets the foundation for how the modeling will be carried out later [20]. For this project, the main focus is on two parts namely short-term traffic prediction and route optimization. In order to achieve this, the data needs to provide information that can represent traffic conditions over time as well as the structure of the road network. Ideally, real-world traffic datasets from government portals or open research projects would be used. However, it is difficult to obtain data that contains all the attributes required for both tasks. Many available datasets are either too aggregated, missing important details, or only cover certain aspects such as overall flow counts without including junction-level or edge-based details. Since Dijkstra's algorithm requires road connectivity information and LSTM models need continuous time-series data, this becomes a major limitation when relying on external datasets.

Therefore to overcome this challenge, the project makes use of a self-generated synthetic dataset. Although it is artificial, it is designed to resemble the kind of information that would typically be found in an actual traffic network. The dataset is made up of four main parts. The first is a list of junctions, which act as the nodes in the road network. The second part is a set of edges that connect these junctions, where each edge represents a road segment with basic attributes such as its starting point, ending point, and a base travel cost. The third part is a time-series of vehicle counts at 5-minute intervals, which provides the temporal element needed for short-term traffic prediction. Finally, the fourth part is a set of congestion factors assigned to each edge, which adjust the travel costs depending on traffic conditions.

Together, these components give a structured view of how traffic flows across the network and how conditions change over time. While the data is not collected from real traffic sensors, it is built in such a way that it can mimic typical traffic situations, such as higher volumes during peak hours and lighter flow during off-peak periods. This makes it possible to test the models in a controlled environment and still get meaningful results. At the same time, the limitation of using synthetic data is acknowledged, as it cannot fully capture unpredictable events like accidents, sudden weather changes, or irregular driver behavior. Even so, the dataset provides a practical solution for demonstrating the modeling approach in this study, as it includes all the necessary attributes to connect traffic prediction with route optimization.

3.1.4 Data Preparation

This data-cleaning is a very important and often time-consuming step in any data science project, often making up 50270 perf of the project workload (Smith et al., 2021). This step will consist of transforming the raw synthetic data into a machine-computer-readable format that can be presented as a structured machine-learning-ready dataset to be used to train a prediction model based on LSTM. The conversion is necessary because the quality of the data that is ready to use affects directly the ability of the model to learn traffic patterns and how accurately it can assist with route optimization through Dijkstra algorithm.

The first part of preparation involves organizing the dataset so that the required attributes are clearly defined and consistently formatted. Since the dataset is time-series based, particular attention is given to the timestamp field. Each record is aligned into 5-minute intervals, which ensures that traffic flow is represented at a fixed and regular frequency. This consistency is crucial because the LSTM model depends on sequential data to identify temporal patterns.

Next, additional features are generated to help the model capture seasonality and traffic fluctuations. For example, the hour of the day, day of the week, and peak-hour indicators are derived from the timestamps. Cyclical transformations such as sine and cosine functions are also applied to represent daily and weekly cycles more effectively. At the same time, rolling statistics such as moving averages and lagged values are introduced to provide the model with information about short-term trends and variations in traffic flow.

Once the features are created, the data is cleaned to remove inconsistencies, fill missing values, and ensure that all inputs are usable for modeling. Scaling is then applied to normalize the numerical values so that different features remain on a comparable scale, preventing large variations in one variable from dominating the model's training process.

Finally, the prepared dataset is converted into sequential input–output pairs suitable for supervised learning. This is done by slicing the time-series into windows of fixed length, where a sequence of past values is used to predict the next traffic value. The data is then divided into three sets: 70% for training, 20% for validation, and 10% for testing. This split allows the model to learn from one portion of the data, tune its performance on another, and finally be evaluated on unseen data to measure its generalization ability.

In summary, this phase transforms the synthetic dataset into a form that captures both temporal and contextual information about traffic flow. By structuring the data in this way, the LSTM model is provided with the inputs it needs to learn meaningful patterns and make accurate short-term predictions, which are later applied in route optimization.

3.1.5 Modeling

In this phase, machine-learning models are used on the ready dataset to create a prediction model [22]. The type of neural network chosen in this project is a Long Short-Term Memory (LSTM) because of its application in time-series problems where the present is determined by past values. As opposed to the traditional models, which look at each record separately, LSTMs are designed to acquire temporal dynamics, by holding salient memory information of past time steps. As a result, they will also be able to capture a combination of short-term fluctuations in traffic and recurring traffic patterns, say, morning and evening rush.

Although pre-trained LSTM models were considered at the start of the project, most available ones are designed for other fields such as language or finance. The few traffic-related models that exist do not include the attributes required here, such as junction-level vehicle counts and congestion values. Because of this, the model will be trained from scratch using the synthetic dataset prepared earlier.

The model is planned with stacked LSTM layers that learn temporal patterns, dropout layers that reduce overfitting, and a final dense layer that produces the traffic prediction. The model input will be a fixed sequence of past traffic values along with time-related features, and the output will be the predicted traffic value for the next time step. Training will be carried out using the Adam optimizer, with Huber loss chosen as the loss function because it balances sensitivity to small errors with robustness against larger fluctuations. Early stopping will also be applied to prevent overfitting and to stop training once the model no longer improves.

After training, the model will be tested on unseen data to evaluate its performance. Standard metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Squared Error (MSE) will be used to measure accuracy and reliability.

The purpose of this stage is not only to produce accurate traffic forecasts but also to supply meaningful inputs for the routing process. By connecting the predicted traffic values to Dijkstra's algorithm, the system will be able to recommend routes that take future congestion into account instead of relying only on static or historical averages.

3.1.6 Evaluation

This phase is an important step in for the framework, as it determines how well the trained model performs on new, unseen data [23]. Once the LSTM model has been trained, its performance will be evaluated to check how well it can predict short-term traffic conditions. To do this, a portion of the dataset is kept aside for testing and is not used at all during training or validation. This ensures that the results reflect the model's ability to generalize to new, unseen data rather than just repeating patterns it has already learned.

The evaluation will be performed through comparing the values of predicted traffic with the determined values of real traffic presented in the test dataset. This will be done using standard error measures, as the Mean Absolute Error (MAE), the Root Mean Squared Error (RMSE) and the Mean Squared Error (MSE). The metrics give different views about accuracy: MAE is the mean difference between predictions and actual values, RMSE underlines more drastic errors and MSE is the general estimate of the variance between the predictions and the ground truth. The visualisation of the predictions will also be done compared to real values so as to provide a better picture of how effective the model would be in capturing the trends in traffic.

Some approaches to the methods of earlier stages of development, like cross-validation, can also be utilised to test the stability of the model on different data division and to reduce overreliance on a single training set. However, actual performance reporting will mainly be based on the holdout test set, because it will be most consistent with the way this model performs in a real implementation.

The evaluation is not only about numerical accuracy but also about whether the predictions are useful in practice. In this project, the forecasts will serve as inputs to Dijkstra's algorithm, so they need to be consistent and reliable enough to support meaningful route optimization. If

the results show weaknesses such as underfitting, overfitting, or poor handling of fluctuations, further tuning or retraining will be considered before moving on to deployment.

In summary, this phase ensures that the LSTM model is both accurate and practical for use in traffic prediction. By validating its performance on unseen data, the project can be confident that the model provides a solid foundation for the integration of predictive traffic data into route optimization.

3.1.7 Deployment

The final stage of the data-mining workflow is called deployment [24], after which the predictive abilities of the model are implemented in a deployable artefact that is intended to generate the tangible utility. In this project the implementation stage is geared towards implementation of the predictive model and routing algorithm hence creates a complete operation system that can be subject to empirical test and presentation. This is achieved through a dashboard built with Streamlit, which acts as the central platform for displaying both the traffic predictions from the LSTM model and the optimized routes generated by Dijkstra's algorithm. The dashboard provides a clear and interactive interface so that the results can be viewed and interpreted easily.

The process begins with the LSTM model, which generates short-term predictions of traffic flow at 5-minute intervals. These predictions are then translated into congestion factors that represent the level of traffic on each road segment. The congestion factors are used as weights in the road network graph, allowing Dijkstra's algorithm to compute the most efficient route under the predicted traffic conditions. In this way, the system is able to adapt routes to changing congestion levels instead of relying only on static distances or historical averages.

The dashboard serves as the main point of interaction, allowing users to experiment with predictions and see how routing decisions change under different traffic scenarios. Traffic forecasts can be visualized as graphs, showing how congestion is expected to rise or fall over time. At the same time, the dashboard displays the current road network together with the shortest path identified by Dijkstra's algorithm. When congestion levels change, the updated path is shown immediately, making it possible to observe the impact of predicted traffic on

route selection. The dashboard also includes options to view comparisons between actual and predicted traffic values, which helps in evaluating the performance of the prediction model.

Several tools and libraries are used to support deployment. TensorFlow and Keras are responsible for running the LSTM prediction model, while Pandas and NumPy handle data processing and feature generation. NetworkX is used to represent the traffic network and implement Dijkstra's algorithm. Streamlit acts as the web application layer, bringing these components together into an accessible and user-friendly dashboard. Matplotlib is used for plotting traffic trends and visual comparisons.

In summary, the deployment integrates prediction, optimization, and visualization into a single system. By presenting the results on a Streamlit dashboard, the project moves beyond isolated model development and demonstrates how predictive analytics can be combined with classical pathfinding techniques in a practical way. The deployed prototype shows how traffic-aware routing can be applied in a clear, interactive, and accessible format, highlighting the potential of such an approach in real-world transport applications.

Chapter 4: System Design Diagram

4.1 Use Case Diagram

The use-case diagram under Figure 4.2.1 outlines the interactions between the User and the Admin in the Traffic Prediction and Routing System that has been designed in terms of optimising the final delivery routes.

On the User side, all interactions occur through the dashboard, which serves as the main interface. The User can access the dashboard to view traffic predictions, visualize the traffic network, plan shortest paths using Dijkstra's algorithm, compare alternative routes, and perform simple what-if analysis (e.g., adjusting congestion scenarios). By centralizing these functions in a user-friendly interface, the system ensures that the User can focus on decision-making without needing to interact directly with the underlying models or algorithms.

On the Admin side, the focus is on maintaining and improving the backend of the system. The Admin manages data loading, cleaning, and preprocessing to ensure the quality of inputs. The Admin also trains the LSTM model on the prepared dataset, exports congestion factors for use in routing, and evaluates model performance using standard metrics. These tasks ensure that the prediction model remains accurate, reliable, and ready for integration with the dashboard.

In summary, the User is responsible for interacting with the dashboard to obtain predictions and optimized routes, while the Admin ensures that the supporting data and models are properly prepared, trained, and evaluated. This division of responsibilities provides a clear boundary between end-user decision-making and backend system management.

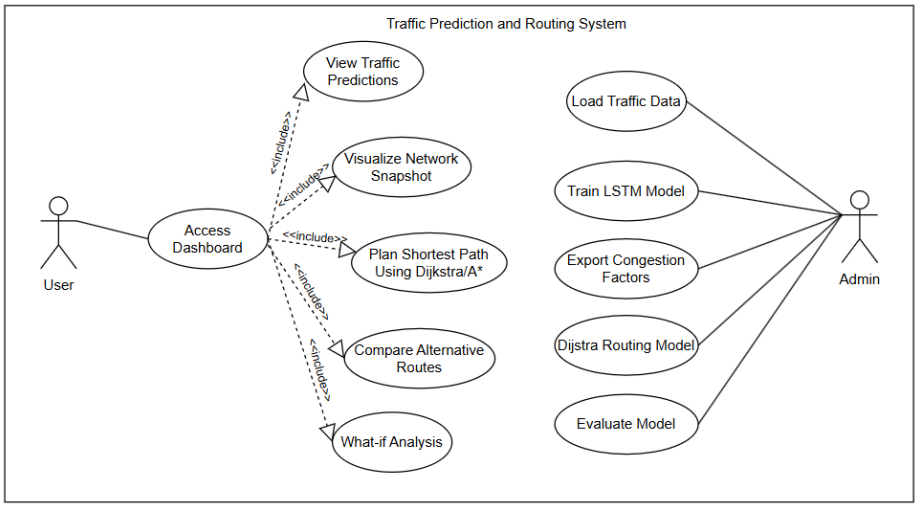


Figure 4.1 Use Case Diagram of Traffic Prediction and Routing System

4.2 Activity Diagram

The activity diagram in Figure 3.x illustrates the sequence of actions that take place between the User, the Dashboard, and the underlying Models when using the Traffic Prediction and Routing System.

On the User side, the process starts when the User accesses the dashboard. Through the dashboard interface, the User can provide the source and destination for the journey. The User also has the option to adjust parameters such as rain or temperature in order to simulate different traffic conditions. These inputs allow the system to perform traffic prediction and rerouting under various scenarios.

Within the Dashboard system, several tasks are managed. First, the dashboard displays general information such as the About page and model evaluation results to give the User an overview of system performance. It also provides an auto-updating map that visualizes future traffic congestion predictions. Once the User has entered their routing request, the dashboard retrieves the required data, applies preprocessing steps, and prepares it for the prediction model. After that, congestion estimation is performed using the forecast results. These congestion values are then supplied to the routing module, where Dijkstra's algorithm calculates the most efficient route based on predicted conditions.

In the Models section, the LSTM model plays a central role. It processes the preprocessed data and generates traffic predictions at 5-minute intervals for each junction. These forecasts are sent back to the Dashboard, where they are converted into congestion factors for each road segment. This ensures that the routing algorithm uses up-to-date predicted traffic conditions rather than static averages.

Finally, the Dashboard system compiles the results and presents them to the User. The output includes the shortest path, a set of alternative routes, and a summary of the result such as estimated travel time. By combining traffic forecasts with route optimization, the system provides a practical tool for planning routes that account for both current and predicted congestion levels.

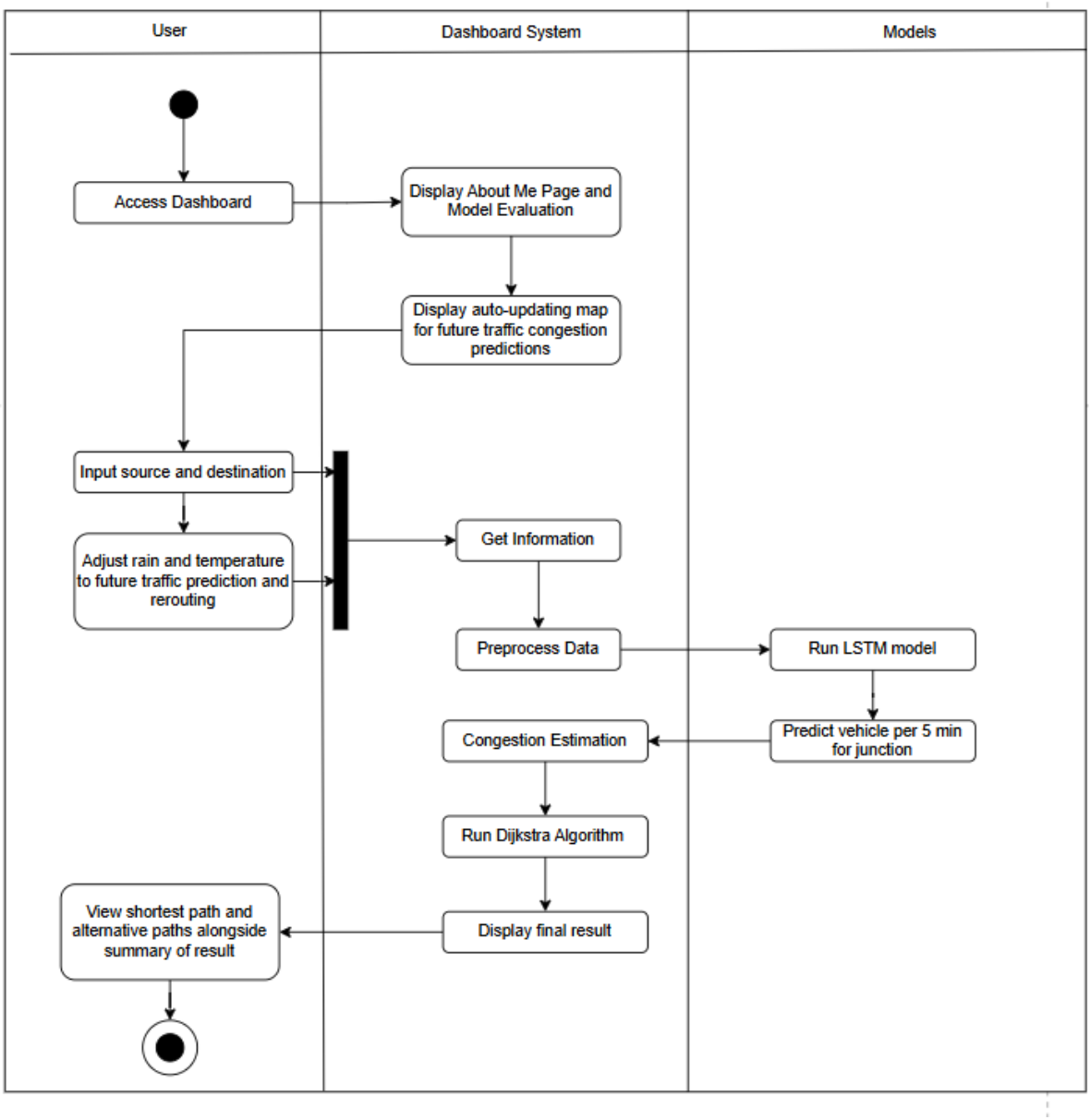


Figure 4.2: Activity Diagram of the Traffic Prediction and Routing System

4.3 Dashboard System

The created dashboard provides an interactive baord that allows users to explore the traffic prediction and route optimization system. Users interact with it in the following way:

Settings Panel as shown in Figure 4.3 and Figure 4.4:

- Users can switch between light and dark themes, adjust figure size, and select visualization options such as congestion colormaps and predicted flow styles.
- Additional controls allow for “what-if” analysis by adjusting rainfall or temperature, enabling users to test how environmental factors might influence congestion.
- Live updates can be toggled on or off, with adjustable refresh intervals, and background predictions can be configured by setting the prediction horizon and frequency.

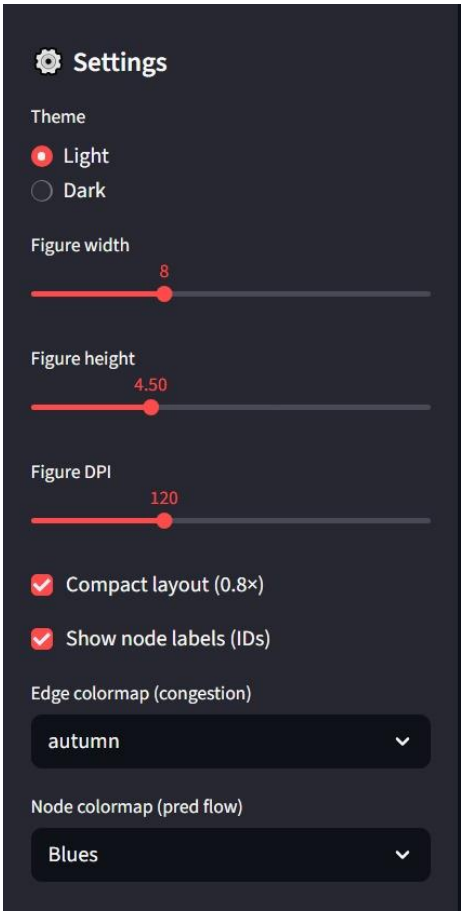


Figure 4.3: Setting Panel 1

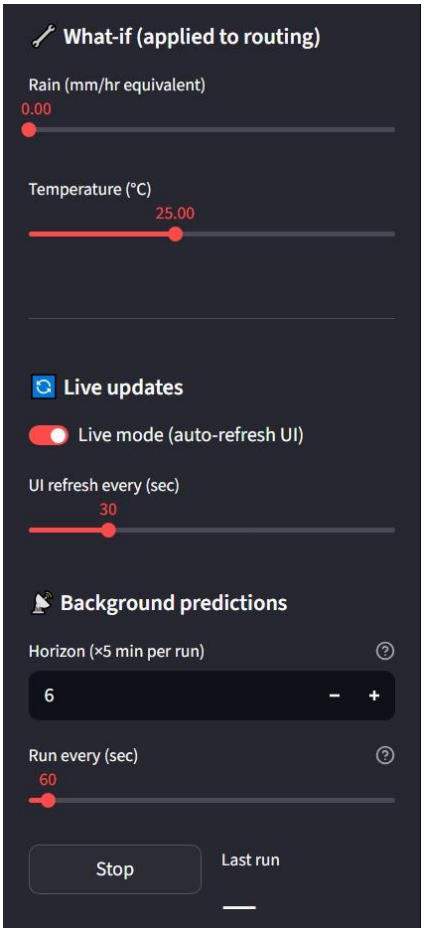


Figure 4.4: Setting Panel 2

Model Performance as shown in Figure 4.5:

- Displays summary of results.
- Shows traffic patterns over time.

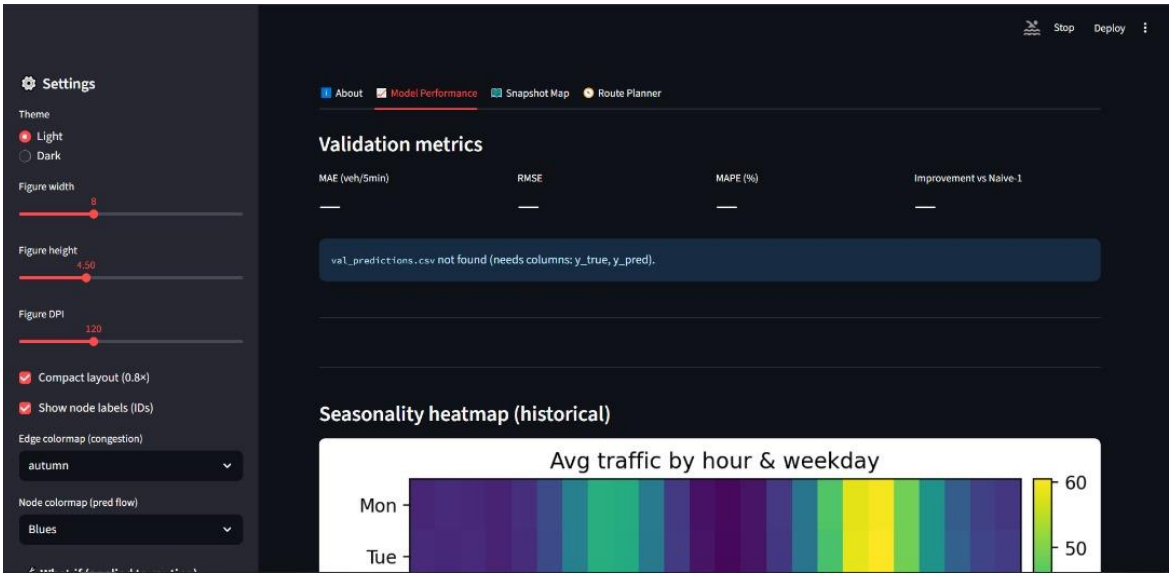


Figure 4.5 Model Performace

Snapshot Map as shown in Figure 4.6:

- Presents a visual snapshot of the road network at a chosen time window.
- Edge thickness and color indicate congestion intensity, while node size and color represent predicted flow levels. This allows users to quickly identify bottlenecks or congested routes.

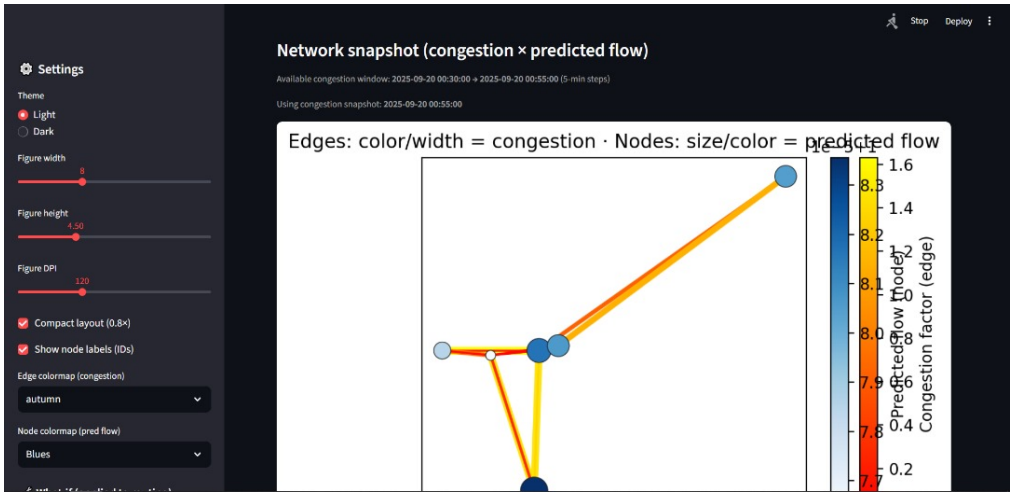


Figure 4.6 Snapshot Map

Route Planner as shown in Figure 4.7 and Figure 4.8:

- Users can select source and destination nodes.
- The system computes and displays the best route along with estimated travel time (ETA).
- Alternative routes are also displayed for comparison, with clear visualization of origin/destination, selected path, and congestion levels.

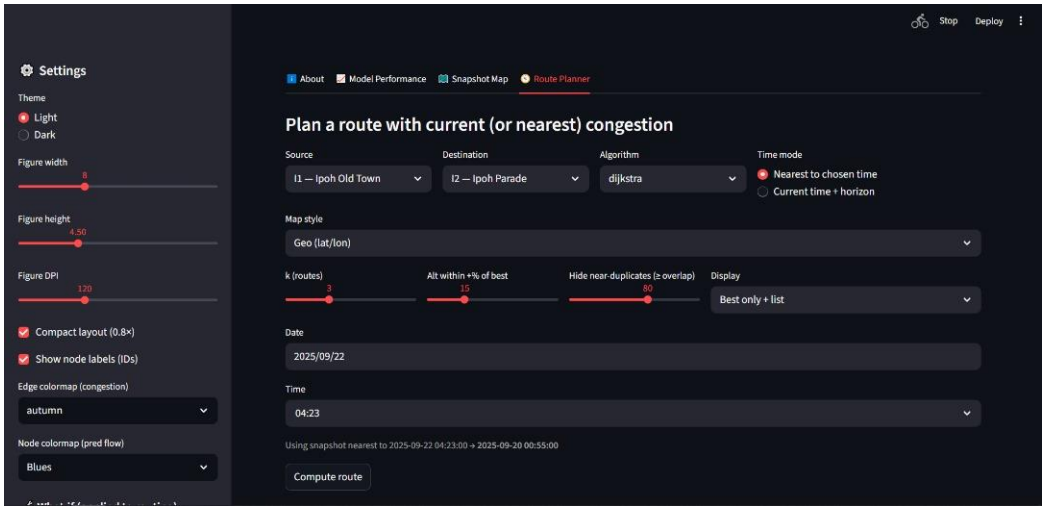


Figure 4.7 Route Planner 1

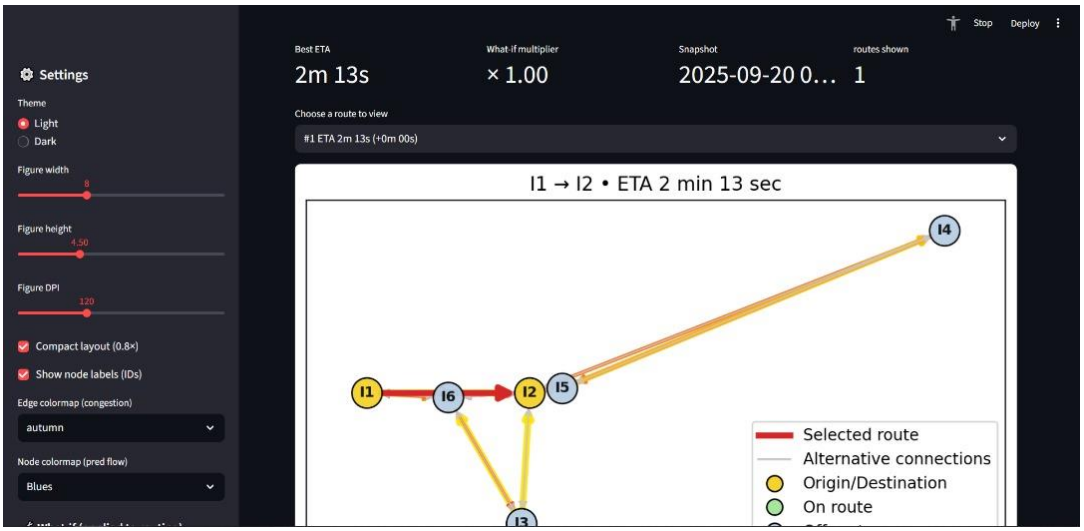


Figure 4.8 Route Planner 2

This interactive flow demonstrates how predictive congestion data can directly inform routing decisions in last-mile delivery scenarios. As the dashboard is currently a prototype, its design and functionality can be refined further in future work to support real-time deployment and integration with logistics platforms.

Chapter 5: System Implementation

5.1 Hardware Setup

The hardware involved in this project is a laptop.

Description	Specifications
Laptop model	ASUS ROG GL552JX
Laptop processor	Intel(R) Core(TM) i7-4720HQ
OS	Windows 10
GPU	NVIDIA GeForce GTX 950M
Storage	256GB
RAM	8GB

Table 5.1 Laptop Specifications

5.2 Software Setup

Visual Studio Code (VSC) was selected as the primary development environment for this project because it is lightweight, cross-platform, and supports a wide range of extensions for Python, TensorFlow, and Streamlit. Its built-in debugging tools and Git integration make it suitable for both rapid prototyping and version-controlled development, while the user-friendly interface allows smooth management of multiple files and workflows [Microsoft, n.d.]. In addition to its technical features, VS Code is also the most widely used editor among developers worldwide, as shown in the Stack Overflow Developer Survey 2023, where it

ranked first in popularity. This combination of functionality and industry adoption makes VS Code a reliable choice for the development of the traffic prediction and routing system [25].

Before beginning the project, it is necessary to install the required Python libraries and packages. This setup is carried out using the integrated terminal in Visual Studio Code. To organize the project properly, a new folder is first created to store all code files and dependencies. The folder is then opened in the editor by selecting "Open Folder…", as illustrated in Figure 5.1.

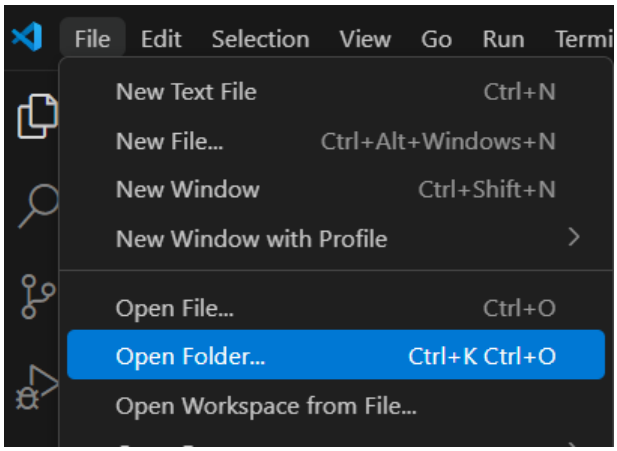


Figure 5.1: Screenshot of Open Folder

Once the project folder is open, a virtual environment (env) is created within the terminal. This environment ensures that all installed libraries are isolated and managed specifically for this project, preventing conflicts with global system packages. The command used to create the environment is "python -m venv .venv" .After activation, the required libraries such as TensorFlow, Pandas, NumPy, Streamlit, and NetworkX can be installed directly into this environment.

This setup provides a clean and controlled workspace, allowing the project to be developed, tested, and deployed without affecting other Python environments on the system.

Then, in the terminal, the virtual environment must be activated before installing the required libraries. As shown in Figure 5.2, once the environment is active, the prompt is prefixed with (.venv), which confirms that all subsequent installations will be contained within this environment. After activation, the necessary libraries for the project can be installed using the command "pip instal -r requirements.txt", with the actual installation process illustrated in

Figure 5.3. The installation process is shown in Figure 5.4. When the installation completes, the libraries are ready to be used for the development of the traffic prediction and routing system.

```
D:\Traffic Prediction LSTM>"D:\Traffic Prediction LSTM\.venv\Scripts\activate.bat"

(.venv) D:\Traffic Prediction LSTM>.venv\Scripts\activate
```

Figure 5.2: Screenshot of how to activate created environment

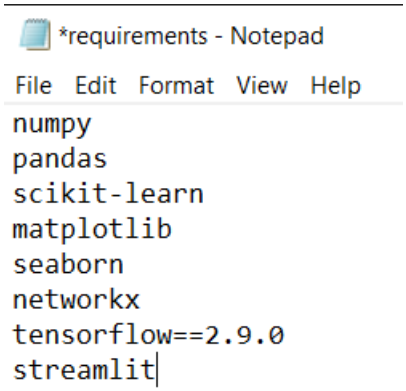


Figure 5.3: Screen of requirements.txt

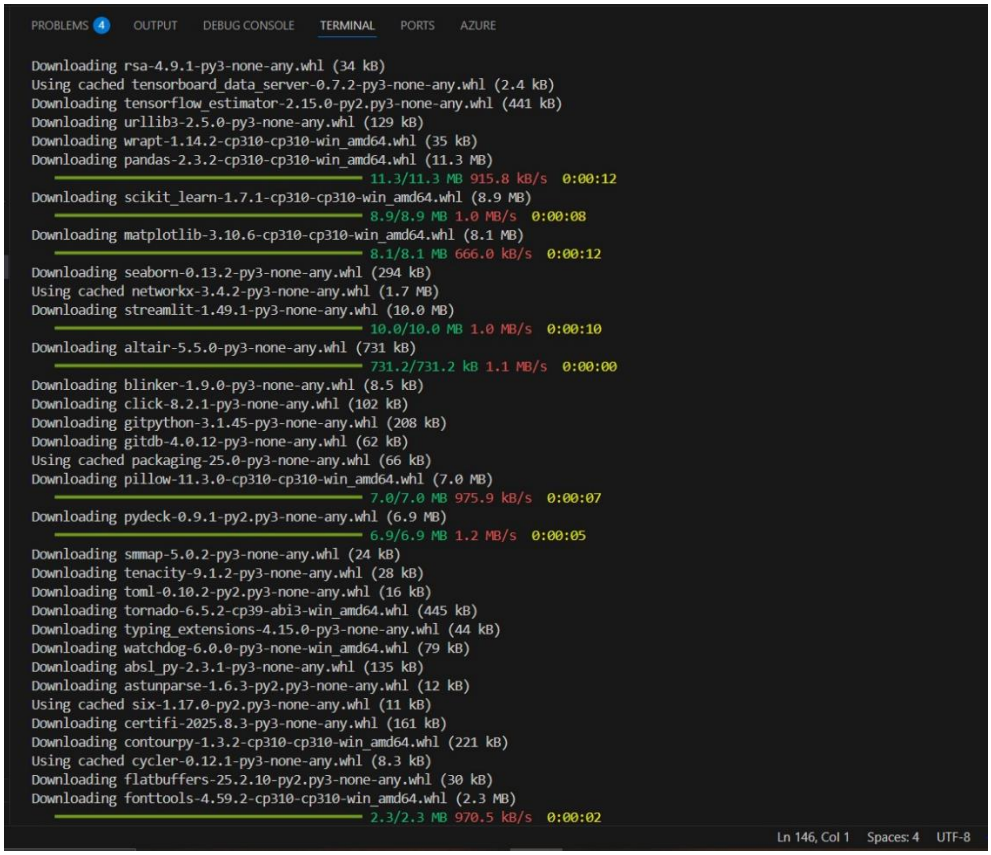


Figure 5.4: Process of Installing Libraries

5.2.1 Configuring GPU Acceleration for TensorFlow

TensorFlow was set up to use the laptop's NVIDIA GeForce GTX 950M graphics card instead of relying only on the CPU to make model train faster. Using the GPU helps speed up tasks like training the LSTM model, which would otherwise take much longer.

For this setup, the versions of the software had to match correctly. Based on TensorFlow's guide, CUDA Toolkit 11.2, cuDNN 8.1.1, Python 3.9, and TensorFlow 2.9.0 were installed. The NVIDIA graphics driver was updated first and checked using the nvidia-smi command to confirm the GPU was detected. After that, the CUDA toolkit was installed, followed by the cuDNN files, which were copied into the CUDA folders on the system. Finally, TensorFlow was installed inside the project's virtual environment.

A short Python script was run to check if TensorFlow could detect the GPU. The output showed that the GPU was available and being used, and the training logs also confirmed that cuDNN was loaded. This meant the setup was successful, and the LSTM model could now be trained with GPU acceleration, making the process more efficient compared to running on CPU alone.

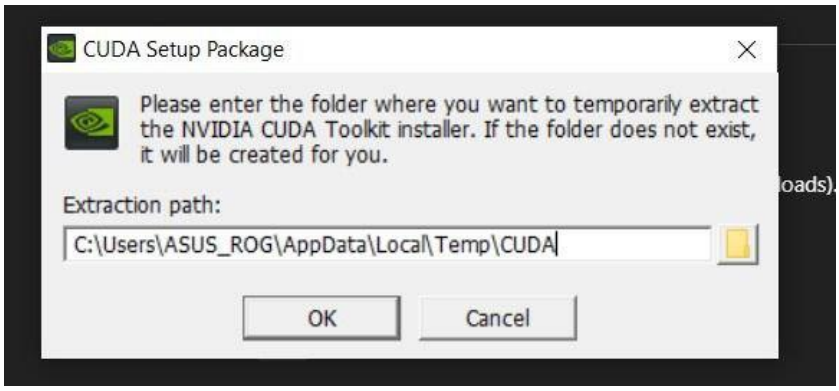


Figure 5.5: Installing CUDA

5.3 Settings and Configurations

After the installation of all required libraries, the project files are organized inside a dedicated folder to ensure a clean and manageable structure. The overall layout of the folder is shown in Figure 5.6. The automatically generated folders such as __pycache__ can be ignored, as they are only created by Python when the code is executed.

In this project, the main files used are:

- traffic_prediction_lstm.py – the script responsible for training and running the LSTM prediction model.
- export_congestion.py – converts the model's prediction output into congestion factors that can be used for routing.
- dashboard.py – the main file that runs the Streamlit dashboard, which integrates the prediction model and routing algorithm into an interactive interface.

Other files such as map_view.py, route_mark.py, and the CSV result files were used mainly during testing in the early stages of development. They helped in checking outputs and validating intermediate results but are not the core components of the final system.

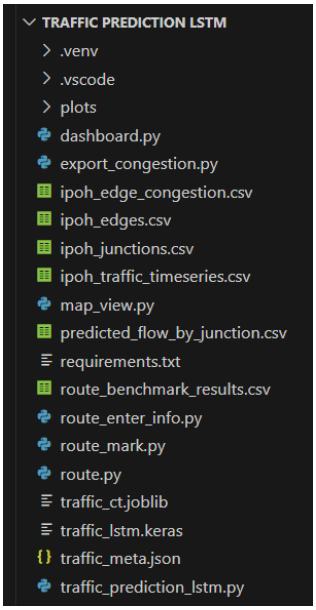


Figure 5.6: Files Created in Folder

This organization ensures that the essential files for model training, congestion export, and dashboard deployment are clearly separated, while supporting files remain available for testing and reference. All Python scripts can be run from the terminal using the format "python filename.py", with the virtual environment activated beforehand to ensure that the correct libraries are used. An example of how to run the file is shown in Figure 5.7.

```
PS D:\Traffic Prediction LSTM> & "D:\Traffic Prediction LSTM\.venv\Scripts\Activate.ps1"
(.venv) PS D:\Traffic Prediction LSTM> python traffic_prediction_lstm.py
```

Figure 5.7: Command to Run File

This part of the report explains how the system was implemented in practice, from preparing the dataset and training the LSTM model, to exporting congestion factors and deploying the dashboard for real-time interaction. Code snippets are included to serve as proof of implementation.

5.4 Traffic Prediction File

The synthetic traffic dataset was stored in CSV format and contains junction IDs, timestamps, and vehicle counts for each 5-minute interval. The dataset was loaded into Python using Pandas and sorted to ensure time-order consistency. Figure 5.8 shows that the timestamp column is parsed as a datetime object to allow time-based feature creation. Sorting by junction and timestamp ensures that traffic sequences remain continuous, which is essential for time-series modeling with LSTM.

```
df = pd.read_csv(args.csv_traffic, parse_dates=["timestamp"]) \
       .sort_values(["junction_id","timestamp"]).reset_index(drop=True)
```

Figure 5.8: Segment of Code 1

Then, to improve predictive performance, additional features were created from the raw vehicle counts. These features below helped the model to capture cyclical traffic behavior and short-term fluctuations:

- Cyclical features such as sine encodings of the hour of the day.
- Lag features to include past traffic counts.
- Rolling averages and standard deviations to represent trends and variability.

The snippet of the code is shown in Figure 5.9. These engineered features enrich the dataset, giving the model context about recent values and recurring daily cycles. These transformations ensure that the LSTM model has both immediate past context and information about repeating daily cycles, which improves its ability to predict traffic fluctuations

```
tgt = "vehicles_per_5min"
grp = df.groupby("junction_id", sort=False)
df["veh_lag1"]     = grp[tgt].shift(1)
df["veh_ma_3"]     = grp[tgt].transform(lambda s: s.rolling(3).mean().shift(1))
df["veh_ma_12"]    = grp[tgt].transform(lambda s: s.rolling(12).mean().shift(1))
df["veh_std_12"]   = grp[tgt].transform(lambda s: s.rolling(12).std().shift(1))
df["rain_mm_lag1"] = grp["rain_mm"].shift(1)
df["temp_c_lag1"]  = grp["temp_c"].shift(1)
```

Figure 5.9 Segment of Code 2

The dataset was also split into training (70%), validation (20%), and testing (10%) sets as shown in Figure 5.10. Unlike random splitting, this was done in a time-aware manner to preserve sequence order. This setup shown in Figure reflects real-world use cases where models learn from past data and predict future, unseen traffic conditions.

```
train_frac, val_frac, test_frac = 0.70, 0.20, 0.10
assert abs(train_frac + val_frac + test_frac - 1.0) < 1e-6

df["t_train"] = df.groupby("junction_id")["time_index"].transform(
    lambda s: int(np.floor(s.quantile(train_frac))))
df["t_val"]   = df.groupby("junction_id")["time_index"].transform(
    lambda s: int(np.floor(s.quantile(train_frac + val_frac))))
```

Figure 5.10: Segment of Code 3

Then, for the modeling part, the LSTM model was implemented as a function (build_model) to allow flexibility in defining the input shape and prediction horizon. The architecture consists of two stacked LSTM layers, each followed by dropout, and a dense output layer that produces forecasts for the defined horizon. The implementation is shown below in Figure 5.11.

- The first LSTM layer (64 units) captures sequential dependencies and returns sequences to the next layer.
- The second LSTM layer (32 units) processes the encoded representation and produces a condensed sequence.
- Dropout (0.1) is applied between layers to reduce overfitting.
- The Dense layer outputs predictions for the specified horizon (default = 1 step, but can support multi-step forecasting).

```
def build_model(input_shape: Tuple[int, int], horizon: int) -> tf.keras.Model:
    return Sequential([
        LSTM(64, return_sequences=True, input_shape=input_shape, recurrent_dropout=0.0),
        Dropout(0.1),
        LSTM(32, recurrent_dropout=0.0),
        Dense(horizon)
    ])
```

Figure 5.11: Segment of Code 4

In summary, the dataset preparation, feature engineering, and model definition steps ensured that the LSTM was provided with structured inputs suitable for learning both

temporal dependencies and recurring traffic cycles. The trained model would later be integrated into the congestion export and dashboard system.

5.5 Traffic Congestion Factor

Once the LSTM model generated predictions, the next step was to convert these values into congestion factors that could be used by the routing algorithm. This task was handled by the export_congestion.py script, which takes predicted junction-level flows and translates them into edge-level congestion weights.

The process begins by rounding the current timestamp to the nearest 5-minute slot and retrieving the corresponding predicted traffic flows. These values are then distributed across the outgoing edges of each junction using a softmax function based on speed limits, which ensures that faster roads receive proportionally more traffic. In the full implementation, an additional shaping step is applied, either using a simple demo mode for testing or a realistic mode that simulates road capacity effects, ensuring that congestion factors better reflect real-world traffic dynamics.

Finally, the resulting values are stored as edge-level congestion factors, which act as weights for routing algorithms such as Dijkstra. The outputs are saved into two CSV files: ipoh_edge_congestion.csv, which records the congestion factor for each edge, and predicted_flow_by_junction.csv, which contains per-junction flows used for visualization on the dashboard. The core part of this implementation is shown in Figure 5.12.

```
def predict_step_for_all(ts_for_step: pd.Timestamp,
                         latest_by_j: dict,
                         ct, model, feat_cat, feat_num):
    rows_out = []
    yhat_by_j = {}
    for j, blob in latest_by_j.items():
        ctx = blob["ctx"][feat_cat + feat_num]
        Xctx = ct.transform(ctx)
        Xseq = np.expand_dims(Xctx, 0)
        yhat = float(np.expm1(model.predict(Xseq, verbose=0)).ravel()[0])  # vehicles/5min
        yhat_by_j[j] = yhat
        rows_out.append({"junction_id": j, "pred_flow": yhat})
    pred_junc_df = pd.DataFrame(rows_out)
    # Map junction flow → edges
    edges = pd.read_csv(CSV_EDGES).copy()
    edges["from_junction"] = edges["from_junction"].astype(str)
    pred_junc_df["junction_id"] = pred_junc_df["junction_id"].astype(str)
    edges = edges.merge(pred_junc_df, left_on="from_junction", right_on="junction_id", how="left")
    edges["pred_flow"] = edges["pred_flow"].fillna(0.0)
```

Figure 5.12: Segment of Code 5

These congestion factors are then used by the dashboard to perform congestion-aware route planning, as described in the following section.

5.6 Dashboard File

The final stage of the system implementation was to integrate the traffic prediction model and congestion export process into a user-friendly interface. This was achieved through the dashboard.py script, which was developed using Streamlit. The dashboard provides a central platform for interacting with the system, allowing users to view traffic predictions, assess model performance, visualize congestion, and compute optimized routes.

The dashboard is organized into four main tabs. The About tab gives a high-level overview of the system workflow, while the Model Performance tab displays accuracy metrics such as MAE, RMSE, and MAPE, alongside plots comparing actual and predicted flows. The Snapshot Map tab visualizes congestion levels across the network by coloring edges and scaling node sizes based on predicted flows. Finally, the Route Planner tab allows users to select a source and destination, then computes the shortest path using Dijkstra's algorithm with congestion factors as edge weights.

In addition to route planning, the dashboard supports what-if analysis through interactive sliders. These allow users to adjust variables such as rainfall or temperature, which scale the congestion factors and demonstrate how adverse conditions might impact traffic. A background prediction loop can also be activated, automatically refreshing congestion data at regular intervals by calling the export_congestion.py script. This ensures that the dashboard remains up to date with the latest forecasts.

Chapter 6: Evaluation

6.1 Introduction

In this chapter, the evaluation of the LSTM model that has been designed to predict short-term traffic is provided. The purpose of the evaluation will be to evaluate the predictive ability of the model on future traffic flow, as well as to identify whether its outputs are accurate enough to use them as a way of giving information that can support route optimisation. A validation dataset and unseen hold-out test dataset was utilised, and the performance of the LSTM was compared with a basic baseline, Naive(1), that assumes that the next value will be the same as the last value. The benefit of this comparative approach is a strict evaluation of the additional value brought by the LSTM model.

6.2 Model Architecture

This is the final LSTM model which was built with these settings:

- Two LSTM layers (64 units and 32 units)
- A dropout layer (0.1) to reduce overfitting
- A dense output layer for producing the forecast

In total, the model had about 34,700 trainable parameters, which is small enough to train efficiently but large enough to capture repeating traffic patterns s shown in Figure 6.1.

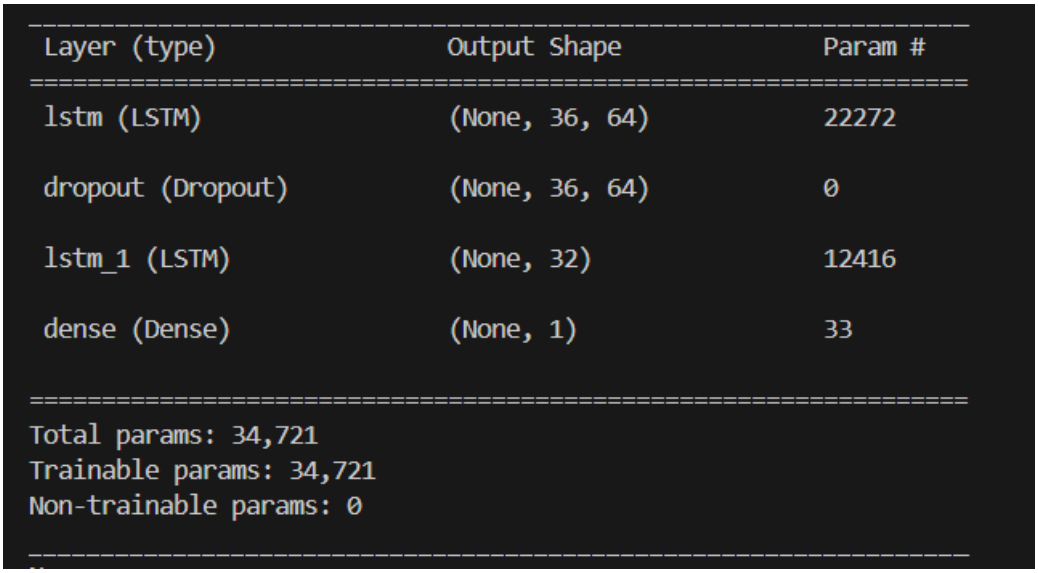


Figure 6.1: Model summary screenshot

6.3 Training Process

The data set was divided so that 70 percent of it was used in the training set, 20 percent in the validation set and the rest 10 percent in the testing set. During training:

- Early stopping was applied so the training would stop once performance stopped improving.
- A learning rate scheduler (ReduceLROnPlateau) adjusted the learning rate when progress slowed, helping the model learn more smoothly.

The model stabilized after about 30 epochs, showing that it was neither underfitting nor overfitting as shown in Figure 6.2.

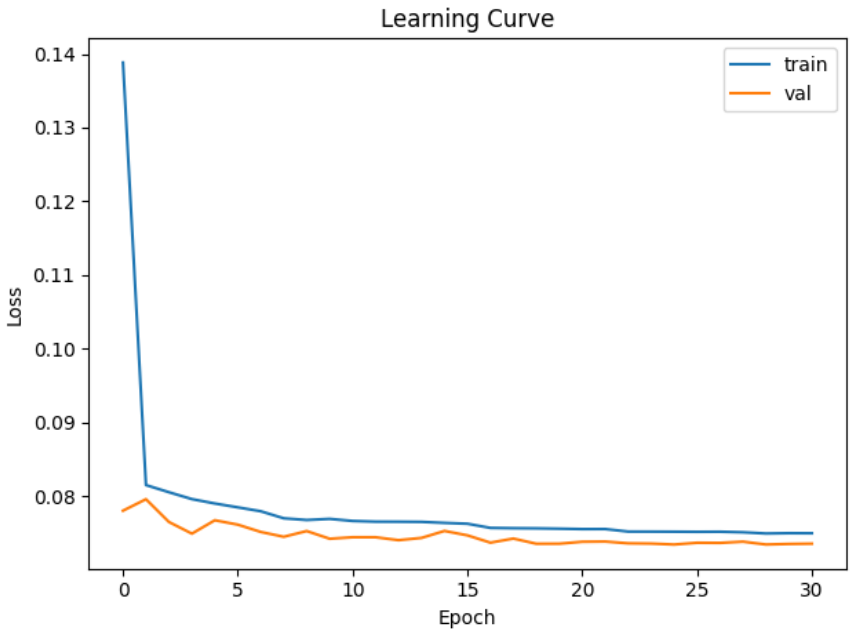


Figure 6.2: Learning curve screenshot (training vs validation loss across epochs)

6.4 Validation Results

On the validation set, the LSTM achieved the following performance:

- MAE: 3.70 vehicles / 5 min
- RMSE: 4.62
- MAPE: 16.97%

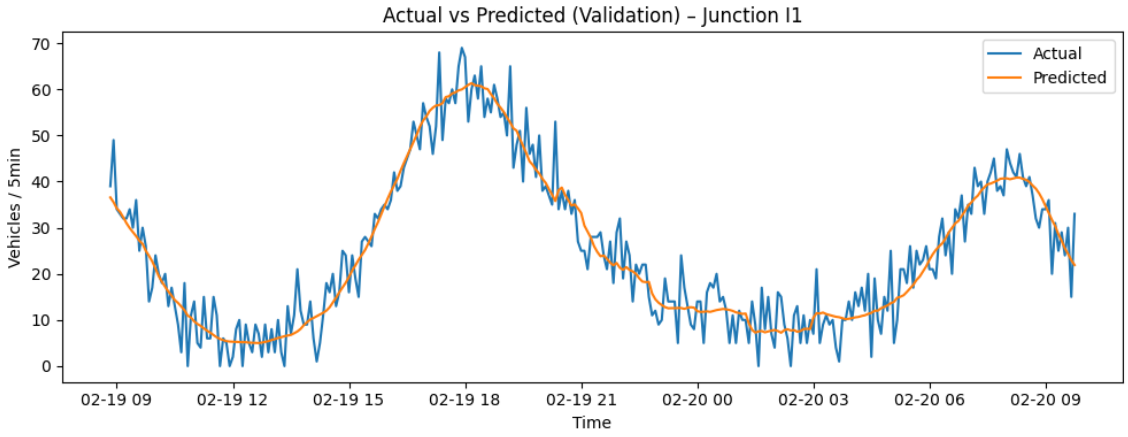
These results indicate that, on average, predictions were within about 3–4 vehicles of the actual recorded values. When compared with the Naive(1) baseline (which simply repeats the last observed traffic value), the LSTM showed clear improvements in Figure 6.3:

- 26.9% lower MAE
- 27.2% lower RMSE

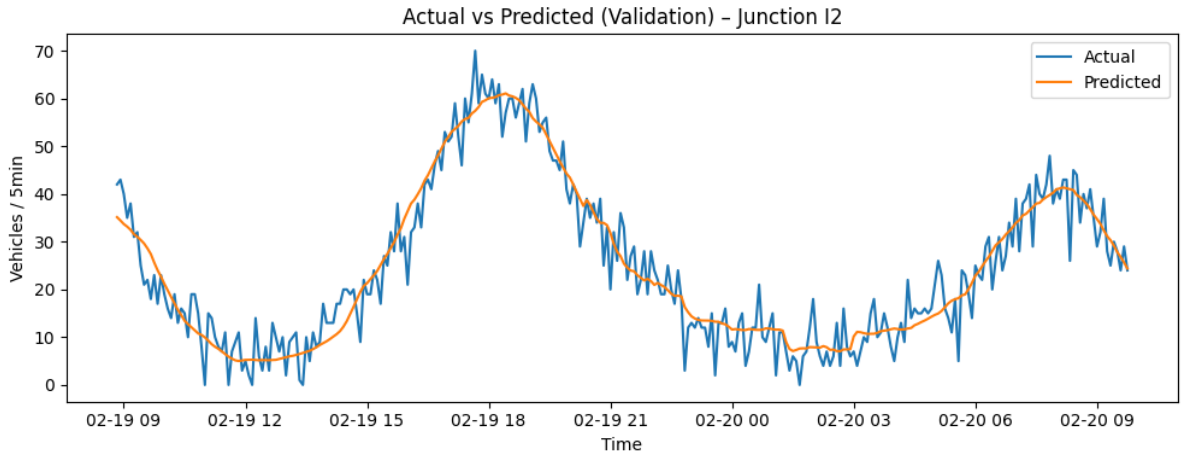
This demonstrates that the model successfully captures short-term traffic dynamics that a persistence model cannot. Figures 6.4 to Figure 6.9 shows the Actual vs Predicted plots for validation for Junctions I1–I6.

```
================ Validation Metrics ================
Model    -> MAE 3.701 | RMSE 4.623 | MSE 21.368 | MAPE 16.97% | sMAPE 26.53%
Naive(1) -> MAE 5.064 | RMSE 6.354
Improvement vs Naive(1): MAE 26.9% | RMSE 27.2%
====================================================
```

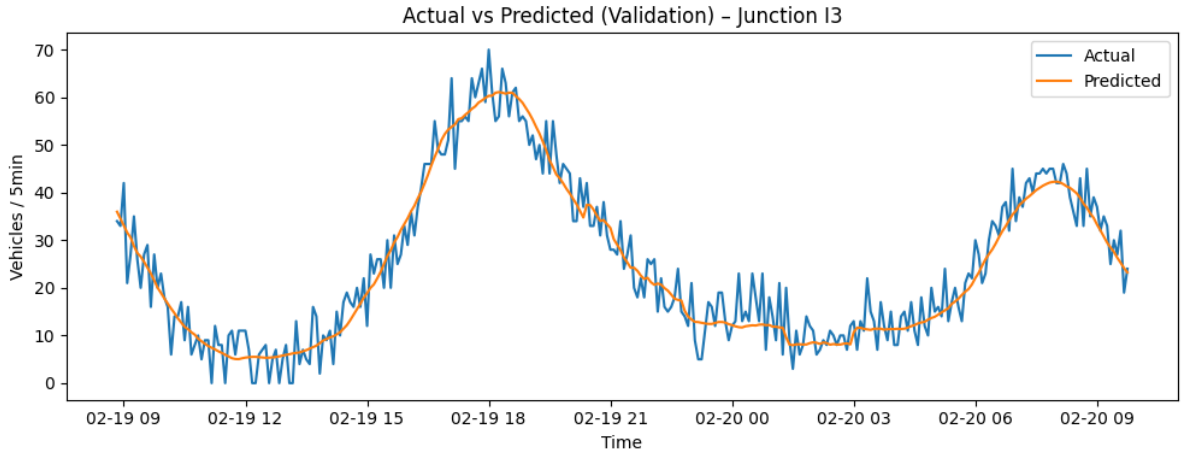
Figure 6.3: Validation metrics and comparison with baseline



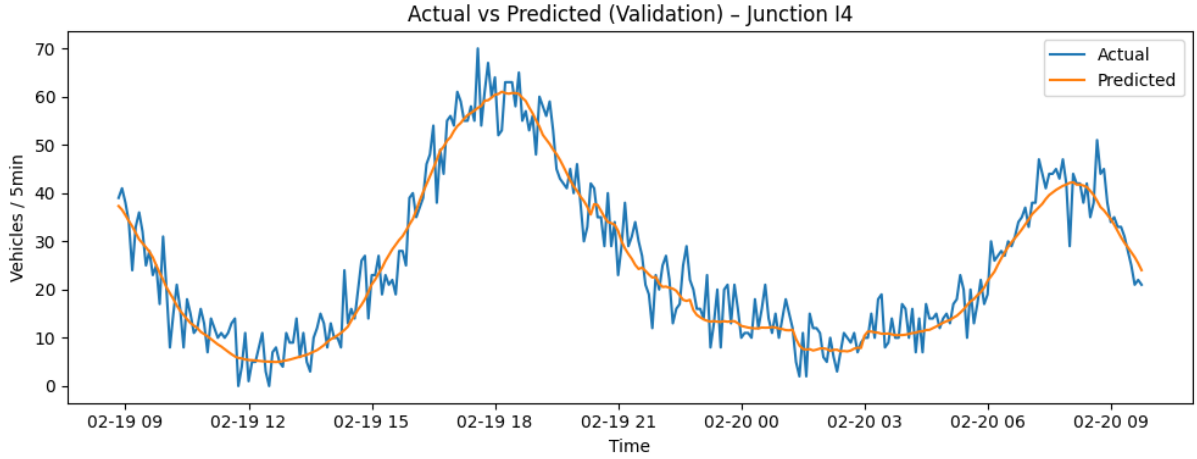
Figures 6.4: Actual vs Predicted plots for validation (Junctions I1)



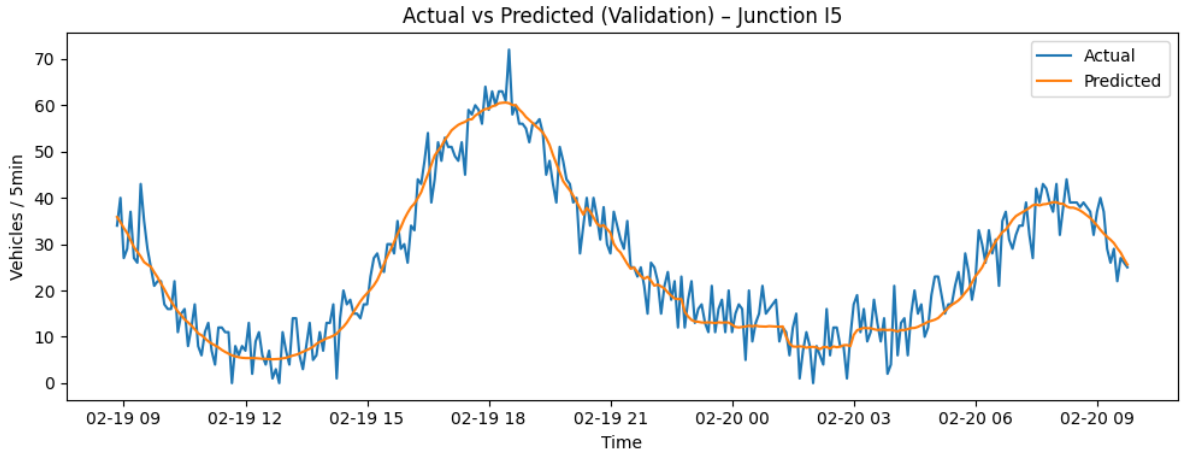
Figures 6.5: Actual vs Predicted plots for validation (Junctions I2)



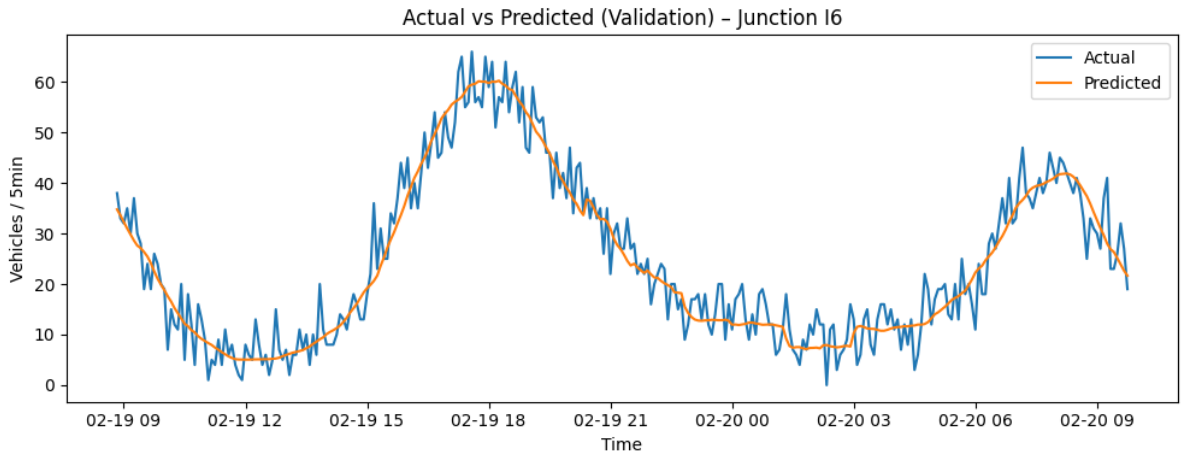
Figures 6.6: Actual vs Predicted plots for validation (Junctions I3)



Figures 6.7: Actual vs Predicted plots for validation (Junctions I4)



Figures 6.8: Actual vs Predicted plots for validation (Junctions I5)



Figures 6.9: Actual vs Predicted plots for validation (Junctions I6)

These graphs show that the predicted curves closely follow actual traffic flows, including peak periods, proving that the model generalizes across different junctions.

6.5 Test Results (Holdout Set)

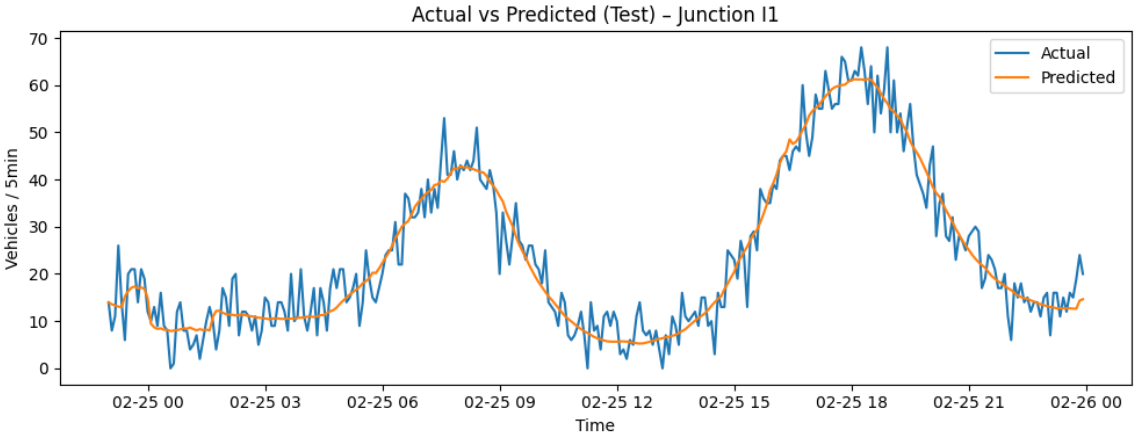
The holdout test set, which contains unseen data, was used to evaluate the model's generalization ability. The results were as follows in Figure 6.10:

- MAE: 3.73 vehicles / 5 min
- RMSE: 4.67
- MAPE: 17.16%

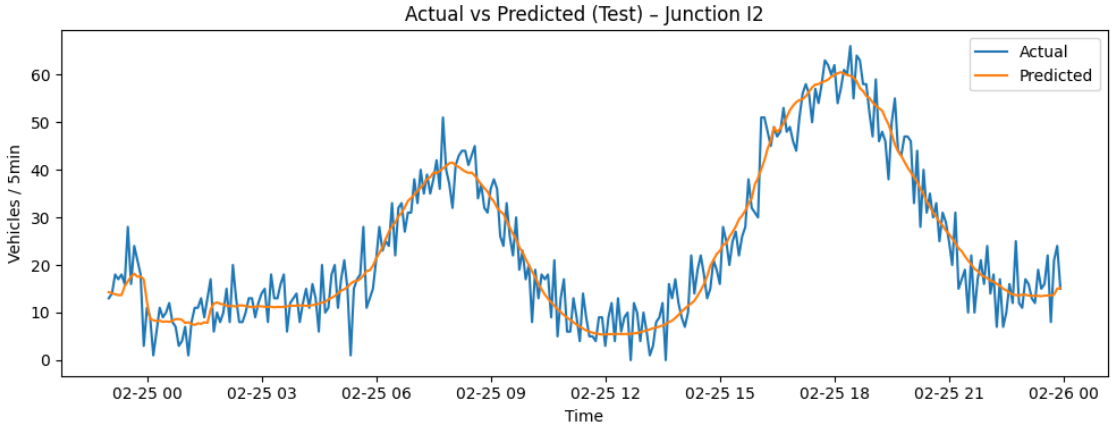
The similarity between validation and test results shows that the model is not overfitted and performs consistently on unseen data. Figures 6.11 to Figure 6.16 shows the Actual vs Predicted plots for test for Junctions I1–I6.

```
================ TEST Metrics (holdout) ================
Model     -> MAE 3.730 | RMSE 4.674 | MSE 21.848 | MAPE 17.16% | sMAPE 26.56%
Naive(1) -> MAE 5.121 | RMSE 6.439
========================================================
```

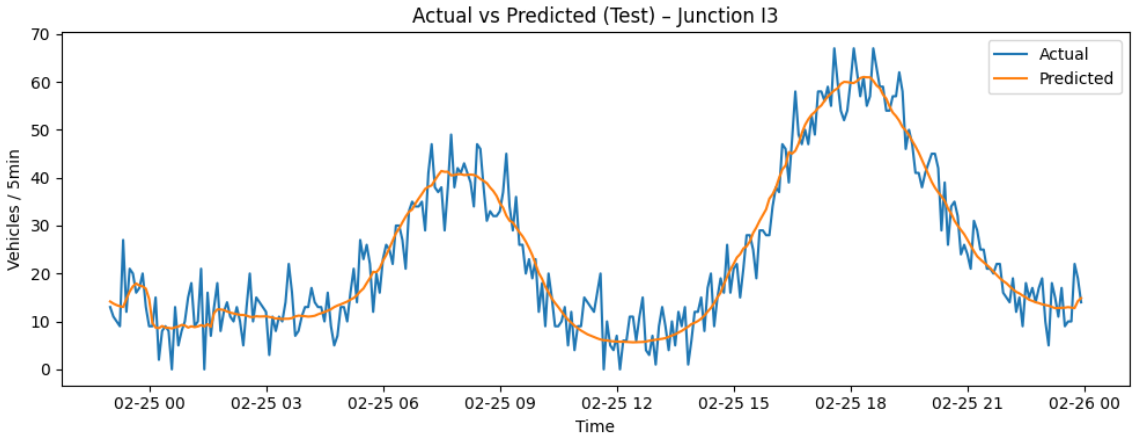
Figure 6.10: Test metrics and comparison with baseline.



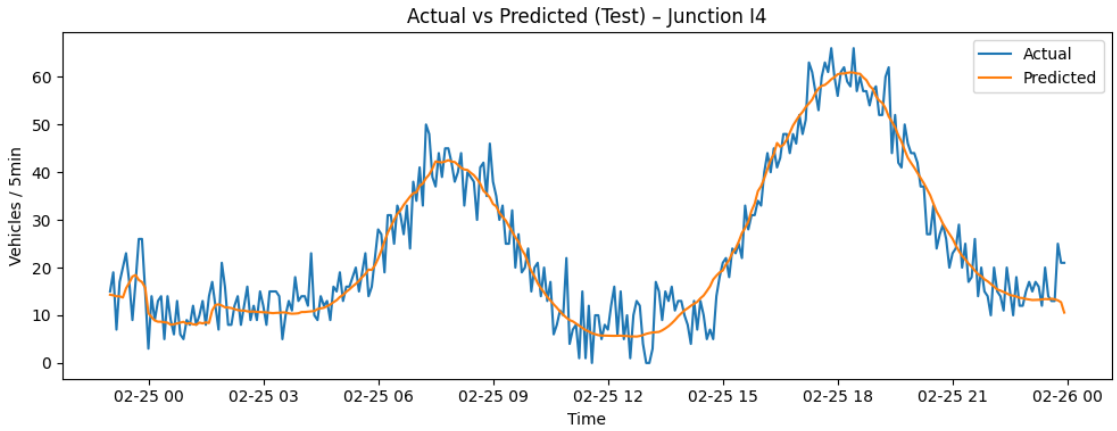
Figures 6.11: Actual vs Predicted plots for test (Junctions I1)



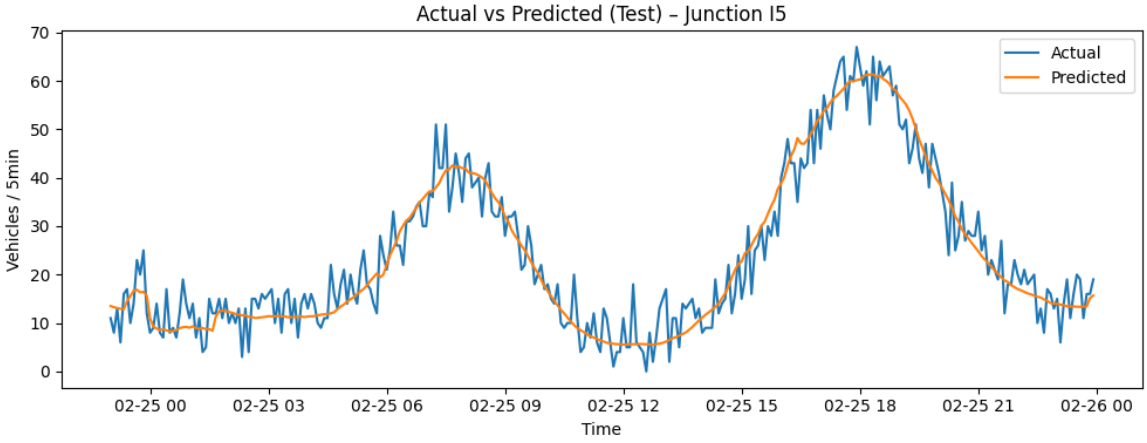
Figures 6.12: Actual vs Predicted plots for test (Junctions I2)



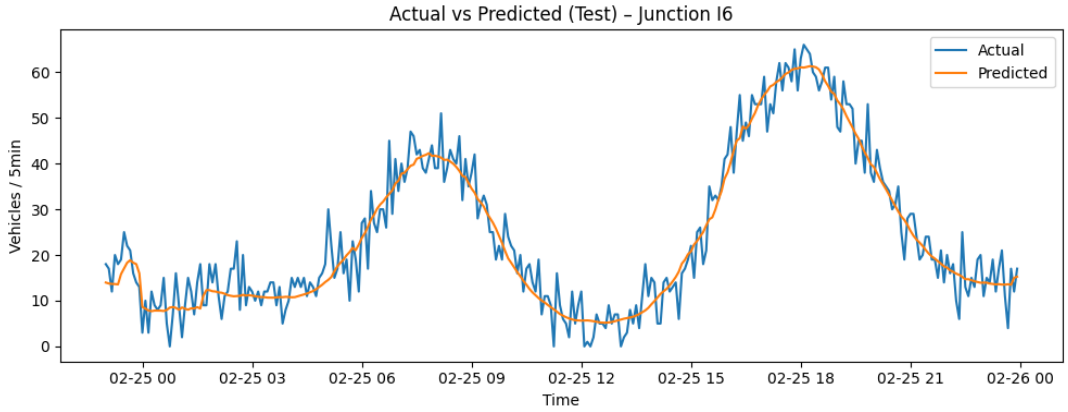
Figures 6.13: Actual vs Predicted plots for test (Junctions I3)



Figures 6.14: Actual vs Predicted plots for test (Junctions I4)



Figures 6.15: Actual vs Predicted plots for test (Junctions I5)



Figures 6.16: Actual vs Predicted plots for test (Junctions I6)

The predictions closely match the actual values, especially during rush-hour peaks, further confirming the model's robustness.

6.6 Error Analysis

Moreover, to gain deeper insights into the model's performance, several diagnostic plots were examined in addition to the main metrics. Figure 6.17 shows a calibration plot.This scatter plot compares the predicted values with the actual observed values. The points align closely with the diagonal reference line, showing that the model predictions are well-calibrated. In practice, this means that when the model predicts higher or lower traffic values, they generally match the true recorded counts.

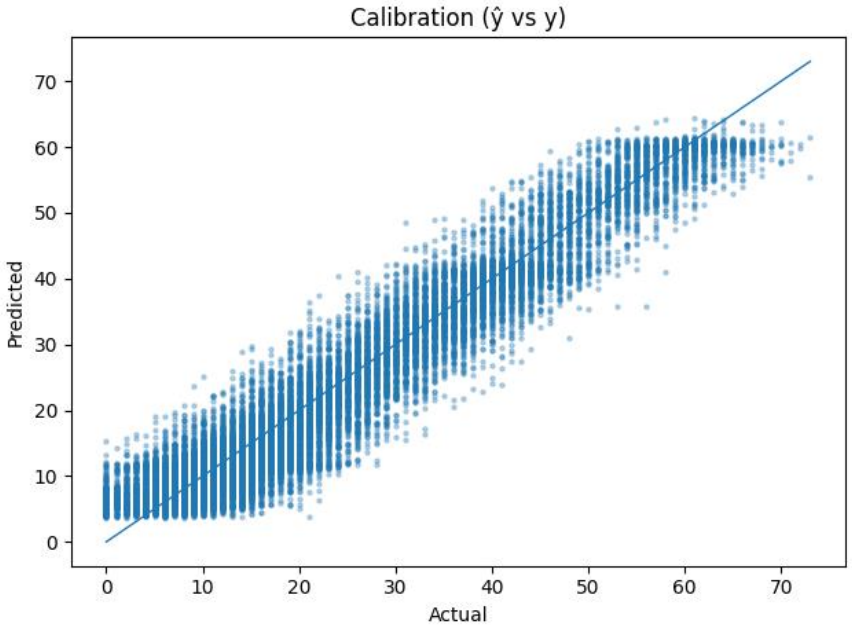


Figure 6.17: Calibration Plot

Figure 6.18 shows an error histogram. The distribution of errors (predicted – actual) is centered around zero, with most errors falling within a narrow range. This indicates that the model does not consistently overestimate or underestimate traffic. The roughly normal shape of the histogram also suggests that errors are random rather than systematic, which is desirable in forecasting tasks.

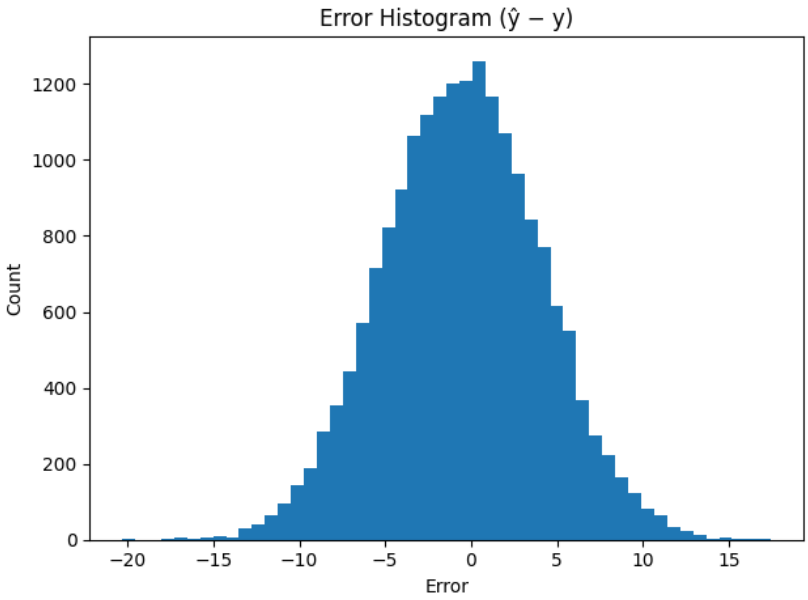


Figure 6.18: Error Histogram

Figure 6.19 shows the per-junction MAE analysis. The Mean Absolute Error was calculated separately for each of the six junctions in the dataset. Results showed very little variation, with values ranging between 3.65 and 3.78 vehicles per 5 minutes. This consistency confirms that the model performs equally well across all parts of the network, instead of being biased toward certain locations.

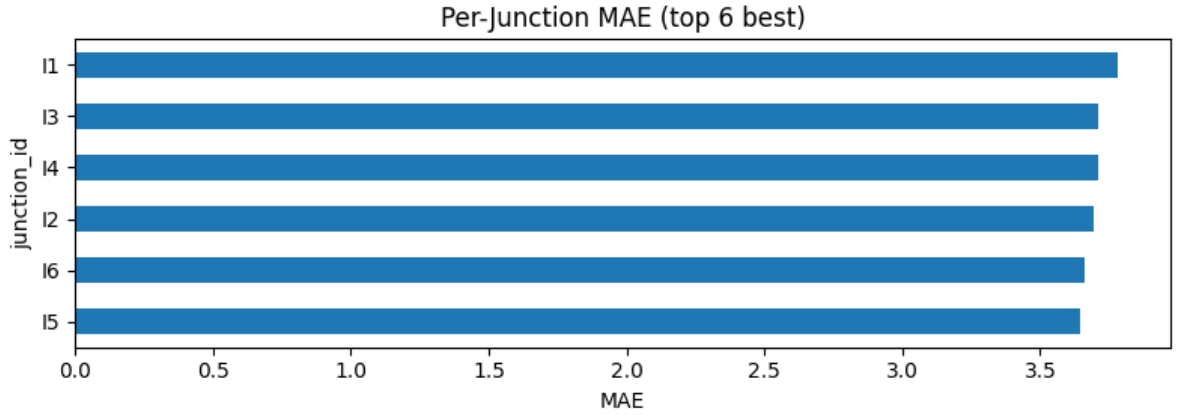


Figure 6.19 Per-Juction MAE Analysis

Lastly, Figure 6.20 shows the seasonality heatmap which illustrates recurring traffic patterns across different hours of the day and days of the week. Clear peaks can be observed during the morning and evening rush periods, while traffic volumes are noticeably lower during nighttime hours. These patterns confirm the presence of daily and weekly cycles in the dataset

and support the conclusion that the model was able to capture meaningful temporal dynamics relevant for short-term traffic forecasting.

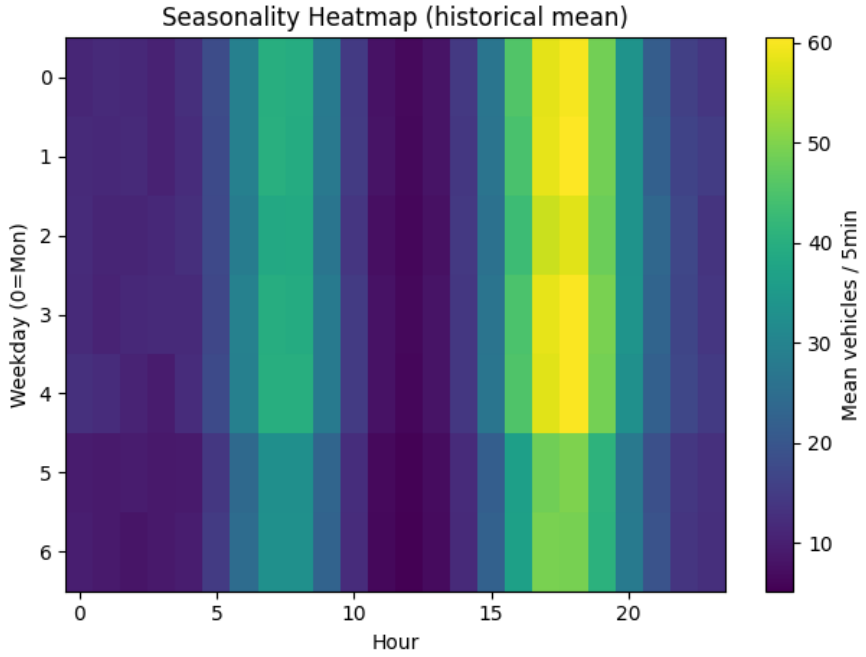


Figure 6.20: Seasonality Heatmap

Overall, these analyses show that the model is not only accurate in terms of numerical metrics but also reliable, well-balanced across junctions, and capable of reflecting real-world traffic behavior.

6.7 System-Level Evaluation (Dashboard Snapshot)

In addition to evaluating the LSTM model's predictive accuracy, the system was also assessed on how effectively it integrates these predictions into the routing dashboard. Figure 6.21 shows a traffic snapshot generated by the dashboard.

- Edges (roads) are color and width coded to represent congestion levels (congestion factor). Roads with higher congestion appear thicker and in warmer colors (red/yellow).
- Nodes (junctions) are scaled and colored according to the predicted traffic flow, with larger and darker nodes representing higher volumes of vehicles.
- The visualization allows users to immediately identify congested routes and high-flow junctions, which supports better route planning decisions.

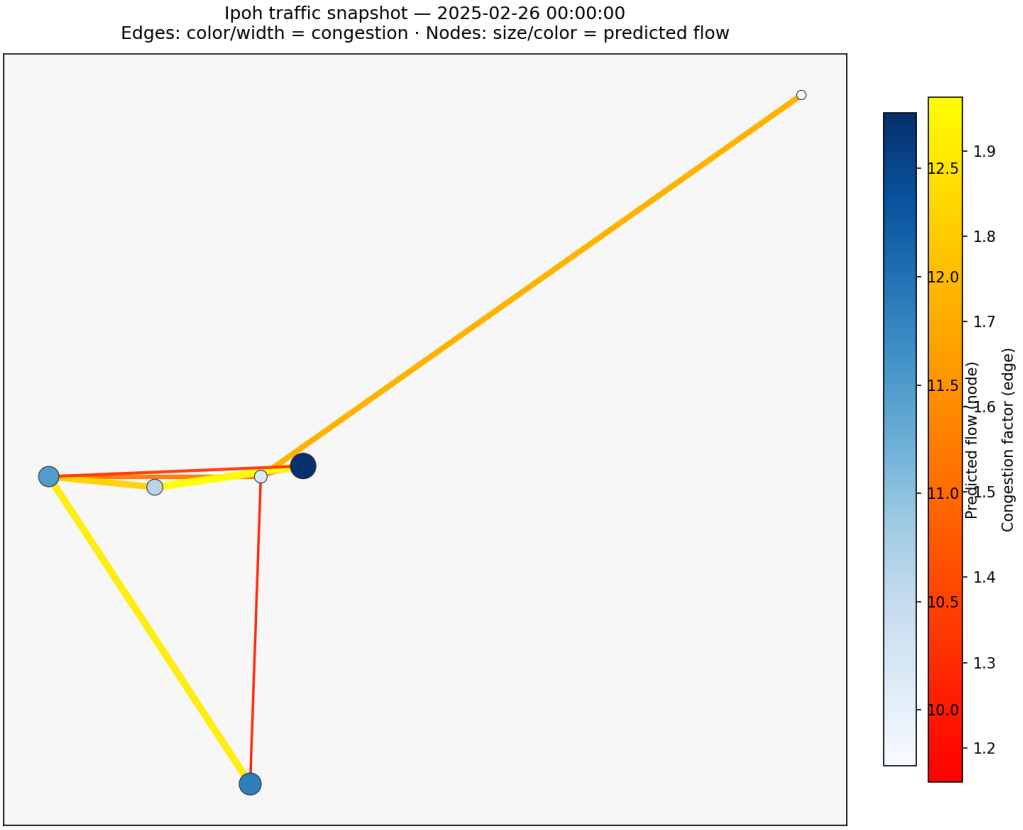


Figure 6.21 Traffic Snapshot

This demonstrates that the system does not only produce accurate numerical forecasts but also provides an intuitive visualization layer that makes the results practical for decision-making.

6.8 Project Challenges and Limitations

One of the major challenges encountered in this project was the difficulty of finding prior research or journal articles that directly addressed the integration of short-term traffic prediction with route optimization using Dijkstra's algorithm. While there is a significant amount of work on traffic forecasting and on routing separately, very few studies have successfully combined the two into a single system. Existing models often focus only on prediction without linking their outputs to routing decisions. Furthermore, most publicly available models were not trained with the necessary attributes, such as congestion factors or edge-level travel times, to support dynamic route planning. This limitation meant that I could not rely on pre-trained models and instead had to self-train an LSTM model from scratch to ensure compatibility with Dijkstra's algorithm.

Another significant limitation was the availability of data. Public traffic datasets are often incomplete, aggregated at the city or region level, or missing crucial attributes such as junction-level flows, connectivity between edges, or congestion indicators. Since these features are essential for both traffic forecasting and routing, the lack of a complete dataset made it difficult to test the system under realistic conditions. To address this, I generated a synthetic dataset that replicated typical traffic behavior, including peak-hour surges and off-peak flows, while ensuring all necessary attributes were included. Although this enabled the system to function as intended, it is important to acknowledge that synthetic data cannot fully capture the irregularities of real-world traffic, such as accidents, road closures, or sudden weather disruptions.

Reflecting on these challenges and limitations, they shaped the uniqueness of the project. The absence of ready-made solutions pushed me to design a custom system tailored specifically to the integration of LSTM-based forecasting with Dijkstra routing. Similarly, working with synthetic data provided the opportunity to test the methodology in a controlled environment, even though it reduced the realism of the results. Overall, these obstacles highlighted the need for innovation when resources are limited and demonstrated that meaningful results can still be achieved through creative problem-solving and adaptation.

6.9 Objective Evaluation

This project aimed to create an intelligent vehicle route-optimisation system that provides a combination of AI-mediated traffic predictions with traditional graph-based route-finding algorithms, which would improve the activities of the last-mile logistics. Advancement made as compared to every individual goal is as summarised below.

Objective 1 - To design and implement a time-series predictive model using Long Short-Term Memory (LSTM) networks capable of forecasting short-term traffic congestion patterns in urban environments.

This objective was successfully achieved. A stacked LSTM model was developed and trained using a synthetic dataset designed to simulate urban traffic conditions at junction and edge levels. The model produced accurate short-term forecasts, achieving validation and test MAE values of approximately 3.7 vehicles per 5 minutes, and consistently outperformed the Naive(1) baseline by over 25%. Evaluation results also showed that the model generalized well to unseen data and performed consistently across all junctions.

Objective 2 - To develop a dynamic traffic-aware route optimization system that integrates predicted traffic conditions into Dijkstra's algorithm to compute adaptive delivery paths.

This objective was also achieved. Predictions from the LSTM model were exported as edge-level congestion factors, which were then incorporated into a weighted network graph. Dijkstra's algorithm was applied using these dynamic weights to compute shortest paths that adapt to predicted congestion levels. The system was demonstrated successfully in the dashboard, showing that routes change dynamically in response to varying traffic conditions.

Objective 3 - To design a traffic-aware dashboard that visualizes predicted traffic conditions, optimized routes, and key system performance metrics in a simulated real-time environment.

This objective was fully achieved. A Streamlit-based dashboard was implemented with multiple functionalities such as model performance tab showing evaluation metrics and validation plots, a snapshot map visualizing congestion across the network and predicted junction flows, and a route planner that computes optimized paths based on predicted congestion. Additional features such as live updating, what-if analysis (for the effect of rainfall),

and visual comparisons between actual and predicted flows enhanced the usability of the dashboard.

Objective 4 - To validate the applicability of the proposed system in last-mile delivery logistics through simulation, demonstrating how traffic prediction and congestion-aware routing can reduce travel time and improve delivery reliability.

This objective was achieved within the simulation scope. A comparative case was conducted where static distance-based routing was contrasted with predictive routing that included LSTM-generated congestion factors. Results showed that the predictive framework avoided congested routes and reduced overall travel time, while also producing more stable routes under varying conditions. These findings confirm that the system can enhance delivery efficiency and reliability in last-mile logistics, even though evaluation was limited to synthetic data.

To summarize, all four objectives were successfully achieved. The system was able to forecast short-term traffic patterns, integrate these predictions into a routing framework, and present the results in a clear dashboard. Furthermore, the project validated the potential benefits of predictive traffic-aware routing in last-mile logistics, showing how such a system can contribute to more efficient and reliable supply chain operations. While testing on real-world datasets remains a recommended next step, the prototype has demonstrated its feasibility and effectiveness within a simulated environment.

6.10 Concluding Remark

This chapter has presented the evaluation of the end-to-end traffic prediction and route optimization system. The system integrates a self-trained Long Short-Term Memory (LSTM) model with Dijkstra's algorithm to provide traffic-aware route planning, supported by an interactive Streamlit dashboard. The evaluation results demonstrated that the LSTM model was able to forecast short-term traffic flows with good accuracy, achieving mean absolute errors of around 3–4 vehicles per 5-minute interval and showing over 25% improvement compared to a Naive(1) baseline. Additional diagnostic plots, including calibration graphs, error distributions, and per-junction performance, confirmed that the model generalized well and produced stable results across the network.

When compared against the project's original objectives, the system met its key goals. The LSTM model successfully captured short-term traffic dynamics, Dijkstra's algorithm was effectively adapted to include congestion-aware routing, and the Streamlit dashboard delivered a user-friendly platform to visualize predictions, optimized routes, and system performance metrics. These elements came together to form a working prototype that demonstrates how AI-based forecasting can be combined with classical pathfinding to support intelligent last-mile logistics decisions.

However, the project also faced certain limitations, particularly in data availability. Since publicly available traffic datasets often lack the attributes required for route optimization, a synthetic dataset was used. While this allowed the system to be developed and tested, it does not capture all the unpredictability of real-world traffic conditions. Despite this, the project provides a solid foundation for further work, showing that such an approach is feasible and holds promise for real-world deployment once suitable datasets are available.

Chapter 7: Conclusions and Recommendations

7.1 Conclusion

A traffic prediction and path optimization framework was created in this project, that is, the use of a Long Short-term memory (LSTM) model plus the Dijkstra algorithm and visualised with the usage of Streamlit dashboard. The framework was trained on an artificially created dataset that was designed to simulate urban traffic dynamics and obtained correct forecasts on short terms which have mean absolute errors of about 3-4 vehicles/5 minutes and it improves by more than 25 per cent compared to a Naive(1) baseline.

The integration of predicted traffic flows into Dijkstra's algorithm showed how congestion-aware routing can adapt delivery paths dynamically, rather than relying on static distances. The dashboard provided a clear interface to visualize predictions, network congestion, and optimized routes, fulfilling all objectives set at the beginning of the project.

While synthetic data limited the realism of testing, the project demonstrated the feasibility and practicality of combining AI-based forecasting with graph-based optimization. It serves as a strong foundation for further development and real-world application in logistics and traffic management.

7.2 Recommendations for Future Work

Although this project has delivered as promised, there are a number of areas through which this can be improved. The most important way to improve it would be to test the system by applying real traffic data instead of using a synthetic one. This kind of empirical validation would make the predictive and routing outputs more realistic and provide a better picture of the performance that the system would have had should it be deployed in a real-life urban situation.

There is another area that can be refined, and that is the dashboard interface. It currently offers handy features such as traffic prediction, traffic congestion maps and road direction. However, later versions may add new interactive capabilities, such as users being able to compare multiple routing routes at once, exporting traffic and routing outcomes in downloadable reports, or a history of past prediction. These additions would raise the functionality and ergonomic nature of the dashboard.

Finally, even though the combination of LSTM and Dijkstra algorithms turned out to be effective in the present study, it can be encouraged to pursue other ways of approach as well. Some of the possible replacements include GRU networks and the A+ search algorithm among others. They would be compared and analysed to decide whether these options introduce better speed or precision, thus bringing the privilege of providing the system to a higher degree of flexibility and to more challenging traffic situations.

Overall, the suggestions above are supposed to be extensions of the current system rather than to replace it. Further test and specific adjustments could make the system more refined and reliable in taking care of the real-life traffic management and logistical optimization.

REFERENCES

[1] MIDA, “Supply Chain: Transforming logistics and warehousing,” *MIDA*, December 25, 2025. https://www.mida.gov.my/mida-news/supply-chain-transforming-logisticsand-warehousing/

[2] Industrial Distribution. “How Traffic Congestion Affects Your Supply Chain.” *Industrial Distribution*, February 13, 2018. https://www.inddist.com/logistics/article/13775513/how-traffic-congestion-affects-your-supply-chain

[3] Khairina Fikri. “Logistics Industry: Navigating the Local Supply Chain in Malaysia.” *Ajobthing*, 2024. https://www.ajobthing.com/resources/recruiter-advice/logistics-industry-navigating-the-local-supply-chain-in-malaysia

[4] Y. Zhang, A. Bhaskar, E. Chung, and L. Duan, “Assessing the impacts of last mile delivery strategies on delivery vehicles and traffic network performance,” *Transportation Research Part C: Emerging Technologies*, vol. 144, p. 103852, 2022. doi: 10.1016/j.trc.2022.103852.

[5] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we’re going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.

[6] Y. Yu, Y. Zhang, and Y. Wang, "Traffic Prediction Using Artificial Intelligence: Review of Recent Advances and Emerging Opportunities," *Transportation Research Part C: Emerging Technologies*, vol. 140, p. 103719, 2022.

[7] IBM, "Artificial Intelligence," IBM Think, 2024. Available: https://www.ibm.com/think/topics/artificial-intelligence. [Accessed: 26-Apr-2025].

[8] Hasanain Kureshi et al. , “AI for Traffic Prediction and Management,” *REST Journal on Banking, Accounting and Business,*27 October, 2024, https://doi.org/10.46632/jbab/3/3/9

[9] Y. Zhong, X. Xie, J. Guo, Q. Wang, and S. Ge, "A new method for short-term traffic congestion forecasting based on LSTM," *IOP Conference Series: Materials Science and Engineering*, 2018, vol. 383, p. 012043.

[10] J. Jang, "Travel Time Prediction Using Machine Learning Algorithms: Focusing on k-NN, LSTM, and Transformer," *The Open Transportation Journal*, vol. 18, pp. e26671212356139, 2024.

[11] M. Gmira, M. Gendreau, A. Lodi, and J.-Y. Potvin, "Travel speed prediction based on learning methods for home delivery," *EURO Journal on Transportation and Logistics*, vol. 9, 2020, Art. no. 100006.

[12] K. Udhan, R. Sawant, S. Patil, and P. Jain, "Vehicle route planning using dynamically weighted Dijkstra's algorithm with traffic prediction," Procedia Computer Science, vol. 200, pp. 287–295, 2022, doi: 10.1016/j.procs.2022.01.214.

[13] Wijaya, D. R., Athallah, A., Noor'afina, T. N., Telnoni, P. A., & Budiwati, S. D. (2023). Cargo Route Optimization Using Shortest Path Algorithms: Runtime and Validity Comparison. *Journal of Computer Science, 19*(11), 1369-1379. https://doi.org/10.3844/jcssp.2023.1369.1379

[14] H. Zhang, Y. Zhang, Z. Wu, Y. Jiang, and S. Jin, "Traffic guidance based on LSTM neural network and dual tracking Dijkstra algorithm," Journal of Physics: Conference Series, vol. 1648, no. 4, p. 042030, 2020, doi: 10.1088/1742-6596/1648/4/042030.

[15] Google, "Find traffic & incidents on Google Maps," *Google Maps Help*. [Online]. Available: https://support.google.com/maps/answer/3093389. [Accessed: Sept. 20, 2025].

[16] Google, "Google Maps," *Google Maps*. Available: https://www.google.com/maps/@4.6125797,101.1214479,14z/data=!5m2!1e4!1e1?entry=ttu&g_ep=EgoyMDI1MDkxNy4wIKXMDSoASAFQAw%3D%3D. [Accessed: Sept. 22, 2025].

[17] Aimsun, "About Aimsun," *Aimsun Official Website*. Available: https://www.aimsun.com/about-aimsun/. [Accessed: Sept. 20, 2025].

[18] AutoWiz, "Route Planning and Optimization," *AutoWiz Official Website*. Available: https://www.autowiz.in/routeplanning.html. [Accessed: Sept. 20, 2025].

[19] N. Hotz, "What is CRISP DM? - Data Science Process Alliance," *Data Science Process Alliance*, Jan. 19, 2023. https://www.datascience-pm.com/crisp-dm-2/

[20] IBM, "Data Understanding Overview," *Ibm.com*, Aug. 17, 2021. ibm.com/docs/en/spss-modeler/saas?topic=understanding-data-overview (accessed Apr. 21, 2025).

[21] IBM, “Data Preparation Overview,” *Ibm.com*, Aug. 17, 2021. https://www.ibm.com/docs/en/spss-modeler/saas?topic=preparation-data-overview (accessed Apr. 21, 2025).

[22] IBM, “Modeling Overview,” *Ibm.com*, Aug. 17, 2021. https://www.ibm.com/docs/en/spss-modeler/saas?topic=modeling-overview (accessed Apr. 21, 2025).

[23] IBM, “Evalution Overview,” *Ibm.com*, Aug. 17, 2021. https://www.ibm.com/docs/en/spss-modeler/saas?topic=evaluation-overview (accessed Apr. 21, 2025).

[24] IBM, “Deployment Overview,” *Ibm.com*, Aug. 17, 2021. https://www.ibm.com/docs/en/spss-modeler/saas?topic=deployment-overview (accessed Apr. 21, 2025).

[25] Microsoft, “Visual Studio Code documentation.” https://code.visualstudio.com/docs (accessed Sep. 20, 2025).

APPENDIX

POSTER