CHALMERS
UNIVERSITY OF TECHNOLOGY

Connected Queue Warning

Proactive traffic hazard warnings based on driving patterns

Hanna Ekhage
Edvin Fredlund

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's Thesis 2025

Connected Queue Warning

Proactive traffic hazard warnings based on driving patterns

Hanna Ekhage
Edvin Fredlund

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Connected Queue Warning
Proactive traffic hazard warnings based on driving patterns
Hanna Ekhage
Edvin Fredlund

Supervisor: Andreas Perme, Volvo cars
Examiner: Fredrik Bruzelius, Mechanics and Maritime Sciences

Master's Thesis 2025
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Visualization of pheromone levels on driven roads in Bejing.

Typeset in LaTeX
Gothenburg, Sweden 2025

Connected Queue Warning
Proactive traffic hazard warnings based on driving patterns
Hanna Ekhage
Edvin Fredlund
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

This master's thesis, conducted in collaboration with Volvo Cars, investigates how individual driving behavior can be used to predict routes and issue timely warnings about upcoming traffic congestion. The system is built using real-world trajectory data from the Geolife dataset, focused on Beijing.
Routes are predicted using ant colony optimization , where pheromone weights are assigned to road segments based on previously traveled paths. Clustering is used to identify common start and end points. Destination prediction combines Random Forest, Bi-directional Long Short-Term Memory and a routing-based elimination method into an ensemble model that is continuously updated and maintains a strong prediction performance of above 69% to a maximum of 98% depending on the journey stage and the specific data split used for validation.
The system sends simulated traffic jam signals which triggers warnings when the predicted path intersects with said traffic jams. To reduce false alerts, warnings are only sent when the model's confidence is sufficiently high. The system, though tested on simulated traffic, provides a proof of concept and a foundation for future real-world applications.

Preface

This report presents the outcome of a master's thesis project carried out at the Department of Mechanics and Maritime Sciences at Chalmers University of Technology during the spring of 2025.

Acknowledgements

We would like to thank everyone that has helped us through this project. A special thank you to our supervisor Andreas Perme who has helped us find relevant software to use, and made sure that we knew where to go in case of problems. We would also like to thank our examiner Fredrik Bruzelius who has helped us a great deal in understanding what is needed from the univerity side, but also has creatively inspired us through interesting discussions centered around the work. In addition, Andreas Lindberg and the Atlas team in Volvo Cars welcomed and inspired us. Another thank you goes to Cristoffer Lejon at New Minds.

Finally, we would also like to thank our families and friends for supporting us through this work.

Hanna Ekhage, Edvin Fredlund, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ACO	Ant Colony Optimization
AI	Artificial intelligence
Beidou	Chinese satellite system
BLSTM	Bidirectional LSTM
DBSCAN	Density-based spatial clustering of applications with noise
FIL	Fresko Innovation Labs
Galileo	European satellite system
GNSS	Global Navigation Satellite System
GLONASS	Russian satellite system
GPS	Global Positioning System
LSTM	Long short-term memory
MinPts	Minimum number of points
OSM	OpenStreetMaps
OSRM	Open Source Routing Machine
RF	Random Forest

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

q	Index for start- and end cluster pairs
i, j	Index for nodes
k	Index for driving sessions

Parameters

μ	Weighted mean difference between average distance and routing distance
σ	Weighted standard deviation for difference between distance and routing distance in percentage
Q	Constant which affects the amount of added pheromones
ρ	Pheromone evaporation rate

Variables

n_q	Number of journeys between start and end clusters with index q
x_q	The percent difference between OSRM and actual distances for start and end clusters with index q
$\Delta\tau_{ij}^k$	The change in pheromones on the road segment between node i and j, for drive k
L_k	Length of driving session k
τ_{ij}	Total pheromone value on the road segment between node i and j

Contents

List of Figures

List of Tables

1
Introduction

This master thesis was conducted in collaboration with Volvo Cars, focusing on the development of a system for avoiding traffic jams by predicting the path and destination of the vehicle based on its driving history.

A model have been developed to warn the driver about upcoming traffic hazards in the predicted upcoming route, based on their driving habits. The system should also consider situations where the driving history does not support the algorithm to produce a prediction with high enough confidence. This is to not give the drivers unnecessary warnings, which might make them disregard the service. Driving history data for training and validation will be based on available vehicle log files including GNSS position time series.

1.1 Background

As the world is rapidly urbanizing, the challenge of congestion on urban roads intensifies correspondingly [Çolak et al., 2016]. Effective traffic flow control is a complex task that requires dynamic and adaptive systems. A key component in developing such systems is understanding driver behavior, which enables smarter decision-making and more efficient traffic guidance [Serok et al., 2022].

Connected functions in the car that alert the driver about upcoming incidents and hazards is a growing area that is included in the latest Euro NCAP assessment. Informing and warning about traffic jams is a prioritized part of Local Hazard detection. In developing a robust and adaptive system for vehicle destination prediction, which is connected to live updates of congestion or traffic incidents, the car can achieve a higher Euro NCAP security rating [NCAP, 2024].

1.2 Purpose

The purpose of this thesis is to analyze peoples' driving behavior, in order to warn of upcoming traffic jams, and thereby minimize idle time in traffic.

1.3 Goals

This project aims to develop a prototype system that can send warnings of upcoming traffic jams that would interfere with the drivers' normal driving patterns, well ahead of time. The core of the system should be a machine learning algorithm using three main inputs,

- The driving history of the car
- The current time and position of the car
- Information on traffic congestion

Using these inputs, the algorithm should predict if the driver is likely to be impacted by any current traffic incidents along the cars' predicted route.

1.4 Limitations

Certain parts that would be needed in a production implementation are not feasible in the time frame of this project. As a result the following limitations of the scope have been made:

- **Simulated traffic jams:** This project uses simulated traffic jams to test and validate the system. This is due to the short time frame of the project.

- **Geographical limitations:** The driving data for this project is gathered by using the Geolife dataset [Zheng et al., 2011]. This data is centered around Bejing. It is possible to use a geographically wider spread of data, but this project focuses on Bejing due to time constraints.

2
Method

The project is carried out in five main steps - data processing, the structure and use of the map, destination prediction, route prediction and traffic handling.

2.1 Data processing

This project is mainly focused around an open dataset called Geolife GPS trajectory dataset centered around Beijing [Zheng et al., 2011].

These datasets consist of positional GNSS data. GNSS stands for Global Navigation Satellite System and is a collective term that includes GPS, GLONASS, Beidou and Galileo. GNSS is a system where radio wave signals from four or more satellites are used to determine the absolute position of an object, such as a moving car. The position is calculated using trilateration from GNSS satellite signals. The resulting coordinates are expressed in terms of latitude and longitude relative to the equator and Greenwich meridian, as well as altitude above sea level. GNSS signals also contain encrypted time stamps generated by the satellites' atomic clocks [Advanced navigation, 2023]. By combining these time stamps with positional data, it becomes possible to identify driving patterns at different times of the day or week.

2.1.1 Geolife dataset

The Geolife dataset was collected by Microsoft Research Asia between 2007 and 2012 and contains 17,621 GPS trajectories from 182 users. Each trajectory is a sequence of time-stamped points with latitude, longitude, and altitude. Most trajectories are sampled densely, every few seconds or meters [Zheng et al., 2011].

2.1.2 Data filtering

The trajectory data from the Geolife dataset includes multiple types of journeys, such as walking, driving, and public transport. Since only the driving data is relevant for this project, an initial filtering step is needed. The dataset also contains unreliable segments, including large positional jumps, multiple journeys mixed into one, and disturbances in the GNSS data. Because of this, substantial filtering is required before the data can be used. In this section, only the trajectories of user 115 are shown, in order to make visualization more manageable.

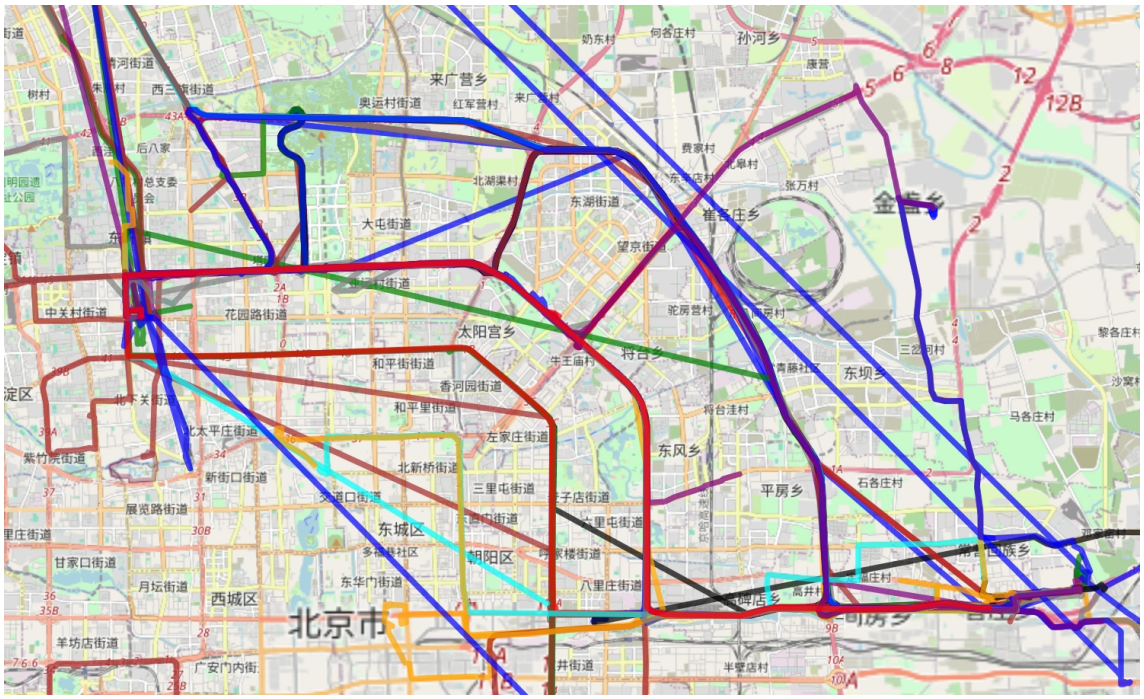

Figure 2.1: Part of User 115's trajectories around Beijing

2.1.2.1 Filtering based on journey type

Some users have labeled parts of their data with the type of journey, along with start and end times, shown in Figure 2.1. Some labeled journeys span multiple trajectory files, while other trajectories contain multiple journeys. The majority of trajectory files however, remain unlabeled.

Start Time	End Time	Transportation Mode
2008/07/08 12:35:04	2008/07/08 12:51:50	bus
2008/07/08 12:51:50	2008/07/08 12:58:15	walk
2008/07/08 12:58:15	2008/07/08 13:38:19	car
2008/07/09 01:00:07	2008/07/09 01:33:37	subway
2008/07/09 13:31:37	2008/07/09 13:40:34	walk
2008/07/10 01:29:36	2008/07/10 01:35:03	walk
2008/07/10 13:06:08	2008/07/10 13:44:20	car
2008/07/10 23:33:50	2008/07/11 00:38:09	car

Table 2.1: Example segment of labels from user 128

To filter the data, all trajectories were sequentially combined into a single Pandas DataFrame. This is a data structure in Python used to store and manipulate tabular data efficiently [Pandas, 2025]. Here, the journeys were split or joined based on the labels. This made the data quality improve significantly, but large positional jumps still remain, as shown in Figure 2.2.

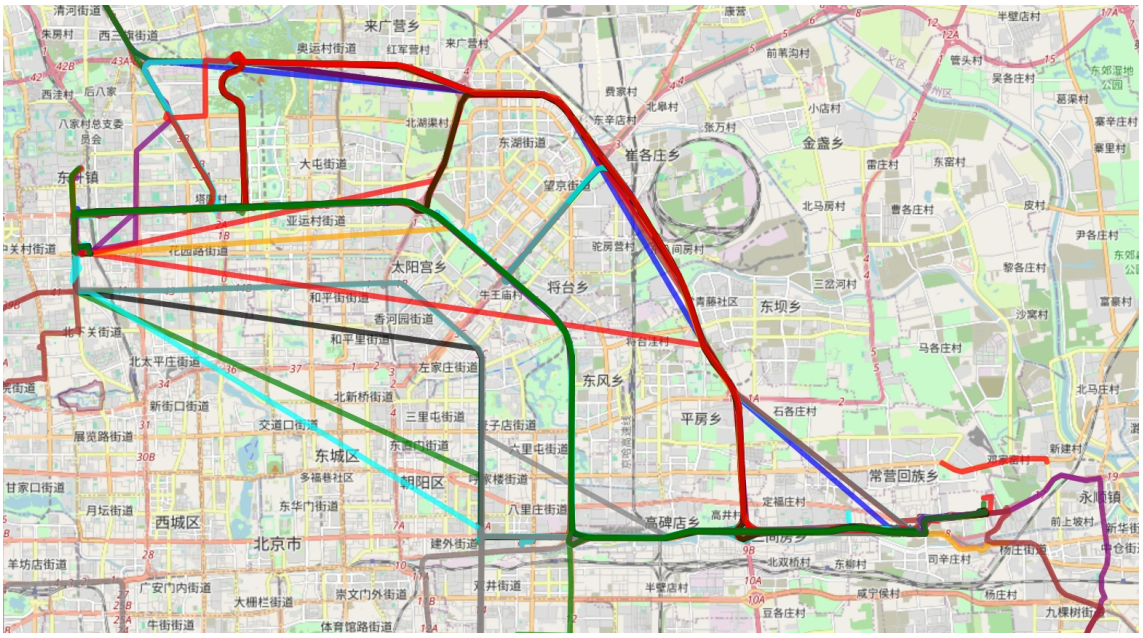

Figure 2.2: Part of user 115's trajectories after labels have been applied

2.1.2.2 Removing noise and splitting journeys

To get clear and reliable driving journeys from the dataset, additional filtering was necessary. This is to eliminate anomalies caused by GNSS signal jumps, which do not accurately represent typical vehicle movement, and to exclude trips that were too short to be considered meaningful journeys.

To remove noise from the trajectories, the speed between all points was calculated. Points with a speed exceeding 150 km/h were removed, as well as points less than 5 meters apart, to reduce data size.

To split journeys, two main methods were used. First, after the initial filtering, speed and distance betwen nodes was recalculated. If the speed still exceeded 150 km/h and the distance exceeded 2 km, a gap was assumed. Second, if a time gap of 6 minutes or more was found between two points, a new journey was assumed to begin. After this step, the trajectories look significantly cleaner, as shown in Figure 2.3.

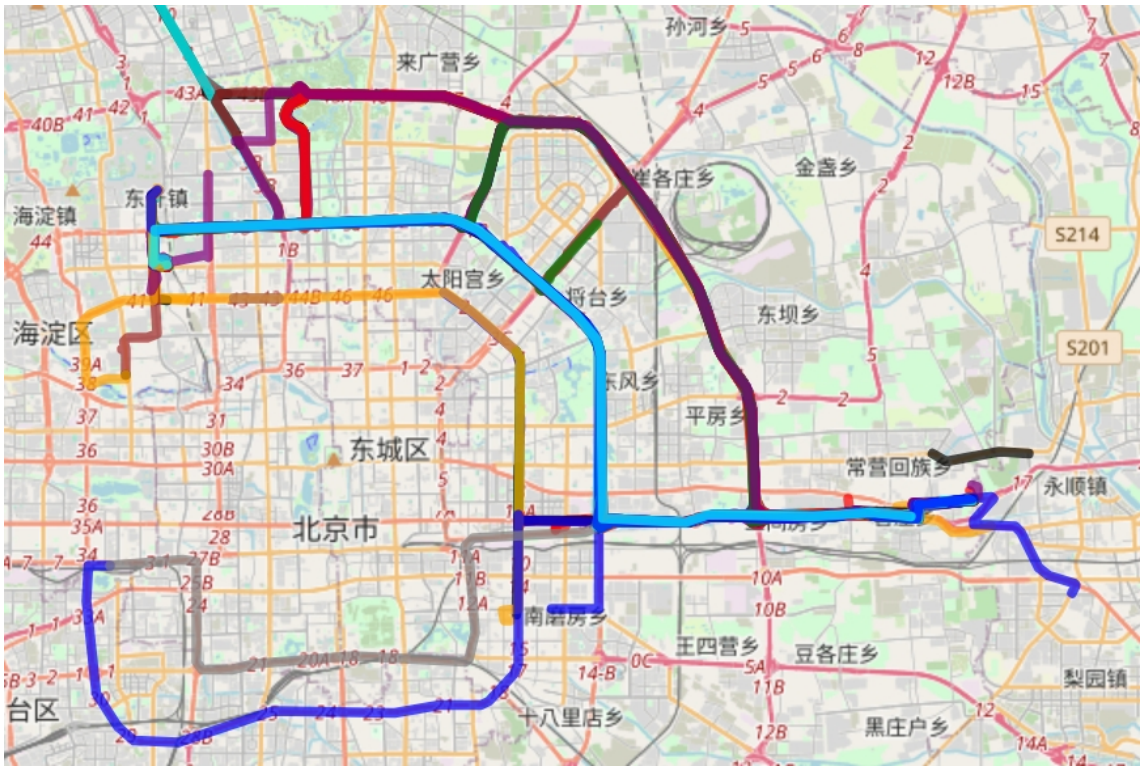

Figure 2.3: Part of user 115 trajectories after filtering

After filtering, a total of 660 journeys remain. As shown in Figure 2.4, the majority of the filtered car journeys are from user 115 and user 128. This is mainly due to a large amount of labeled car journeys form these users in the dataset. The data from user 115 and 128 are more relevant for this project than other users, because the large data size enables more accurate predictions to be made.

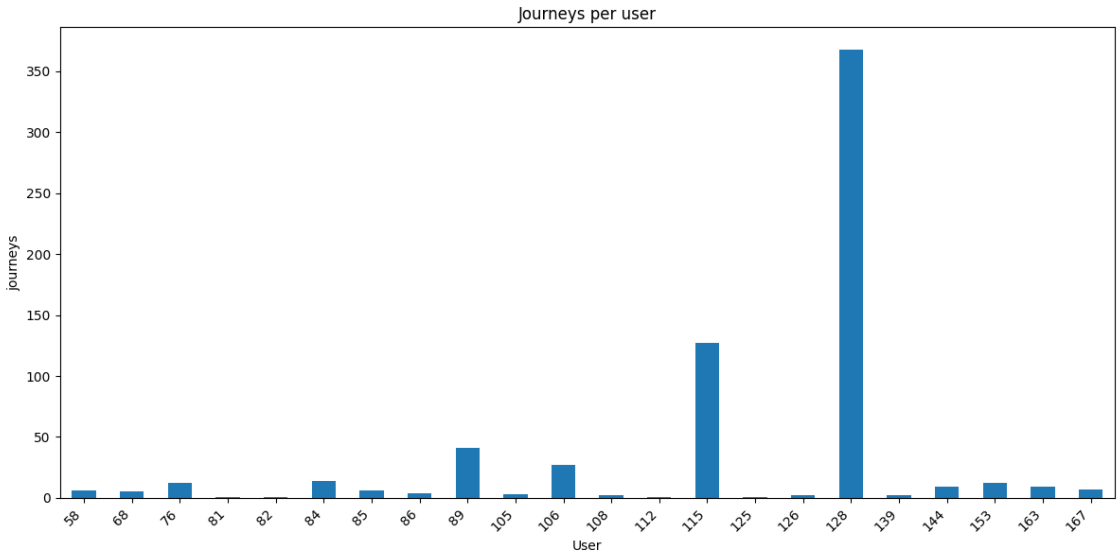

Figure 2.4: Distribution of filtered car journeys among users

2.2 The map

The map can be viewed as a graph where the roads are represented as edges, and intersections are represented as nodes. Each node and edge is assigned an id for the corresponding intersection or node in accordance to OpenStreetMaps (OSM) id listings [OpenStreetMap, 2025].

2.2.1 Constructing a graph

Maps from OSM were used for this project. Since the route planning is not distance based, but based on previous driving patterns, the map was turned into a graph, where the drivers patterns are mapped out as weights. A python package called OSMnx was used, which has functions that turns the OSM into a graph. The graph is also simplified by taking away unnecessary nodes that are outside the search area [Boeing, 2024].

2.2.2 Map matching

Although the journeys are now filtered, they still do not align with the road network, as illustrated in Figure 2.5. To address this, different open-source routing engines were evaluated.

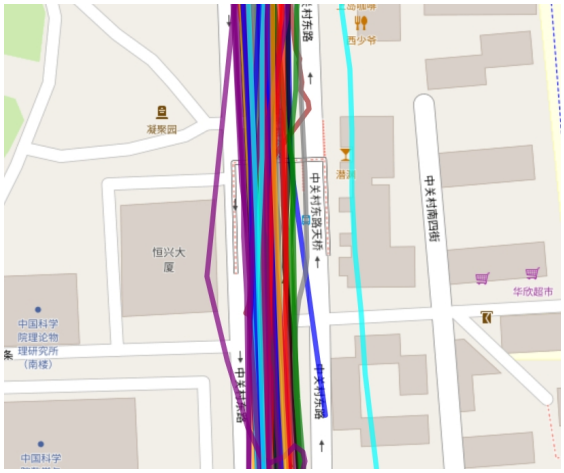

Figure 2.5: A road with unmatched journeys

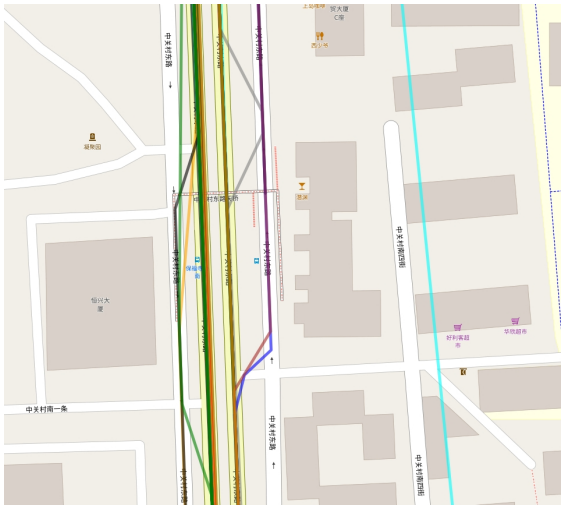

Figure 2.6: A road with map matched journeys

An engine called GraphHopper was originally used. It is strict about ensuring a contiguous route through the road network [GraphHopper, 2025]. If the coordinates drift onto a nearby road, it will sometimes try to route to that road, and then return to the original road. This results in extra loops and turns that do not actually exist, making the journeys unrepresentative and unpredictable, seen in Figure 2.8
Another open-source routing engine called Open Source Routing Machine (OSRM) were also evaluated. OSRM is an open source routing engine built on top of OSM.

A local OSRM instance was set up using Docker, based on the OpenStreetMap (OSM) map file for the Hebei region (which includes Beijing and Tianjin) [OSRM, 2025], as this corresponds to the geographical area covered by the driving data. Running the service locally using a docker container enabled fast, large-scale batch processing of trajectory data without relying on external APIs.

Using the local host, OSRM requests can be sent by formatting journey coordinates as strings. OSRM then matches the trajectory to the most likely path in the road network. The response returned new coordinates that follow the road network as can be seen in Figure 2.6. The map-matched journeys sometimes change lanes unrealistically or even appear to drive on the wrong side of the road. This is due to the low precision and noise in the original trajectories. Despite this, the results are generally still an improvement, as seen in Figure 2.7.

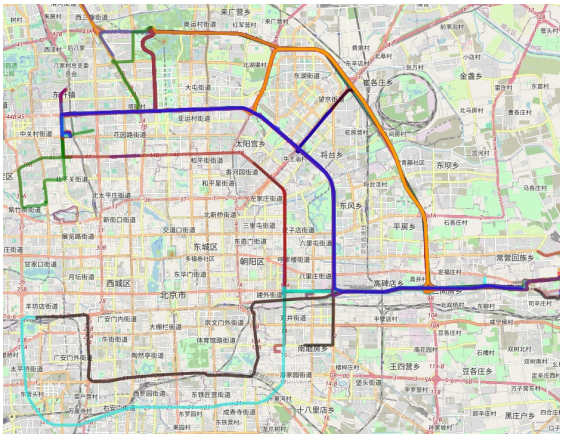

Figure 2.7: Map matching using OSRM

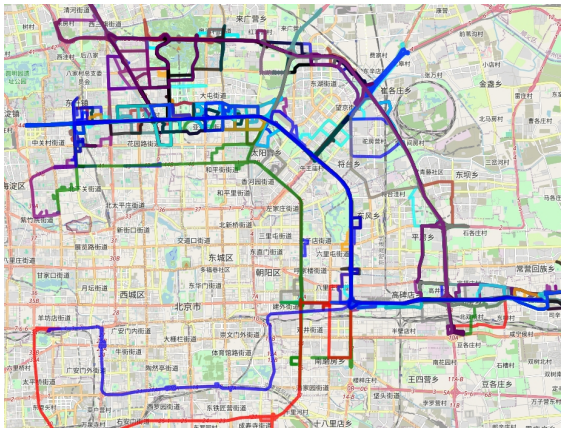

Figure 2.8: Map matching using GraphHopper

The map matching functionality provided by the open-source routing engine OSMR proved to be well-suited for the objectives of this project. Its ability to reliably align GPS traces with the road network ensures consistent and accurate route representation. Consequently, OSMR was selected as the sole map matching tool for use throughout the remainder of this thesis.

The main reason for map matching is to connect the journeys to the graph. The matched journeys were returned as both a gpx file but more importantly it returns a list of nodes representing the journey. A node in OSM is a specific point with a fixed latitude and longitude, used to shape paths like roads. When two roads meet at the same height (e.g., at an intersection), they share a node. The nodes returned by the OSRM match request were compared against a file of all junction nodes in the Hebei graph, and reduced to only include intersections. A map with only intersection nodes and edges between them can be seen in Figure 2.9.

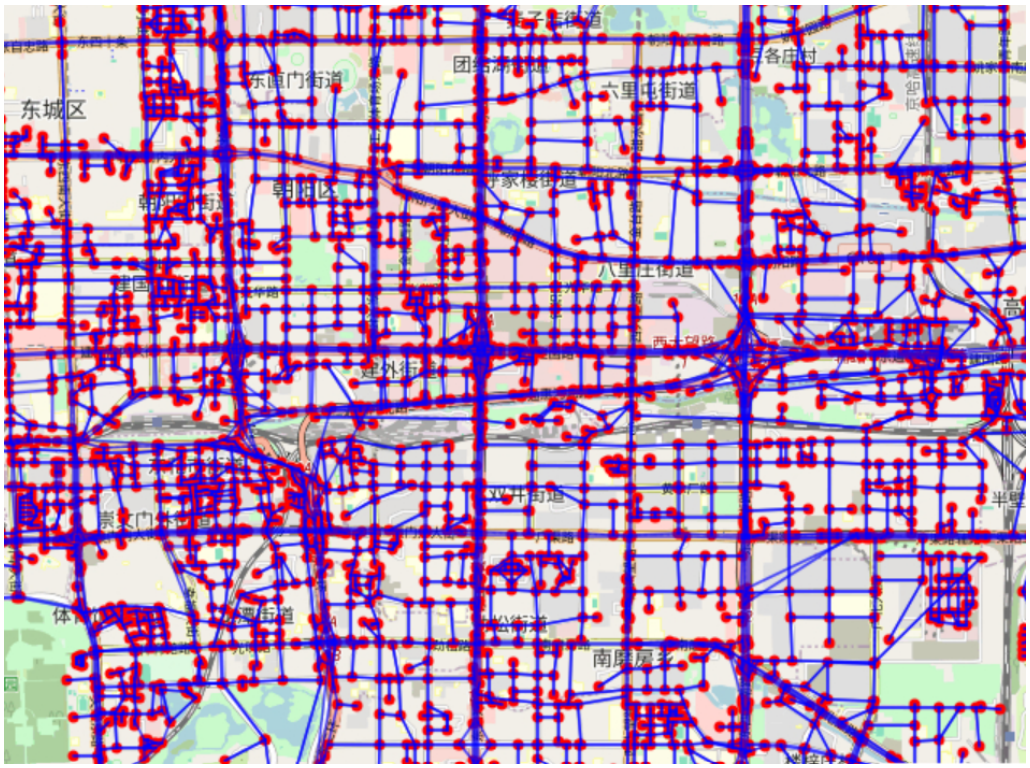

Figure 2.9: Graph of the road map of Bejing

2.2.3 Clustering

Clustering is employed to group nearby start and end points of journeys, revealing common travel origins and destinations across the map. By grouping these points into distinct clusters, key travel hubs can be identified, enhancing the accuracy of destination prediction.

After filtering the journeys, the start and end coordinates of each journey are saved and plotted, shown in Figure 2.10. These start and end points are scattered across the city, with many points appearing isolated. In order to group these points into clusters a density-based spatial clustering of applications with noise (DBSCAN) algorithm is utilized. DBSCAN in an iterative clustering algorithm that clusters points within a certain radius of each other, discarding clusters that contain fewer start or end points than the specified minimum number of points (MinPts) [Ester et al., 1996].

Applying a DBSCAN with a MinPts of 9 points and a radius of 550 meters on the start and end points, shown in Figure 2.11, results in clusters shown in Figure 2.12.

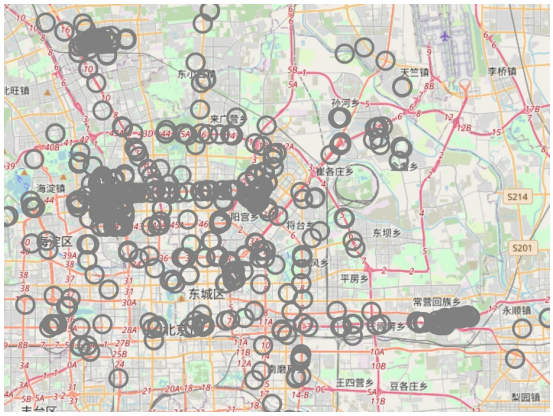

Figure 2.10: Map of Bejing with start and end points

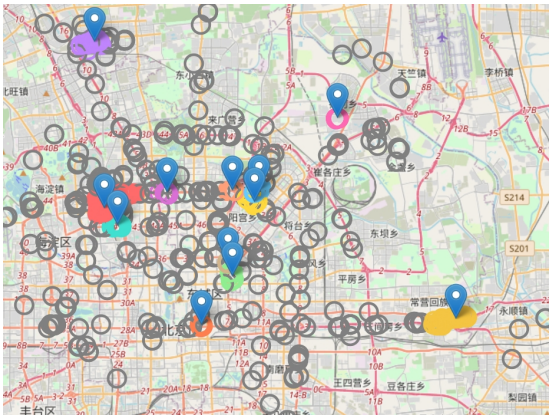

Figure 2.11: clustered start points and end points with MinPts = 9 and radius = 550m

There are still many unclustered start and end points. This is primarily due to users with limited data points, inconsistent car usage, and faulty data that causes the filtering process to split or prematurely end journeys. These isolated points cannot be used for destination prediction because a pattern cannot be established from single occurrences.

Figure 2.13 shows that clusters 0, 8, and 10 are the largest, representing most of the journeys between clusters. These clusters are highlighted in Figure 2.12.

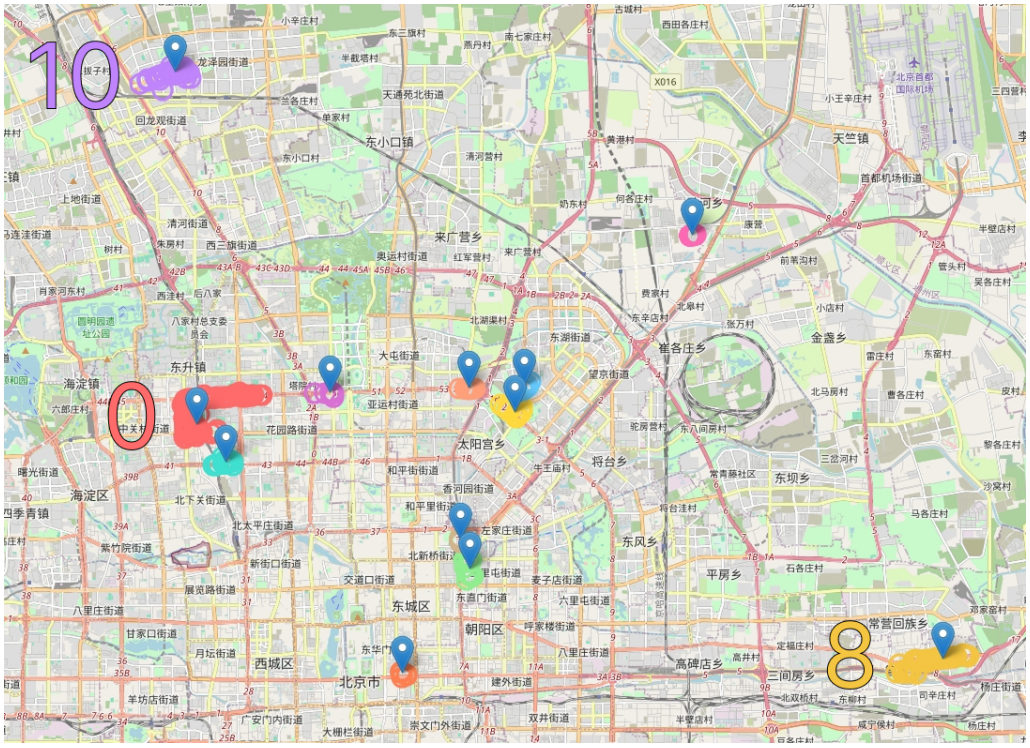

Figure 2.12: Map of Beijing with labeled clusters

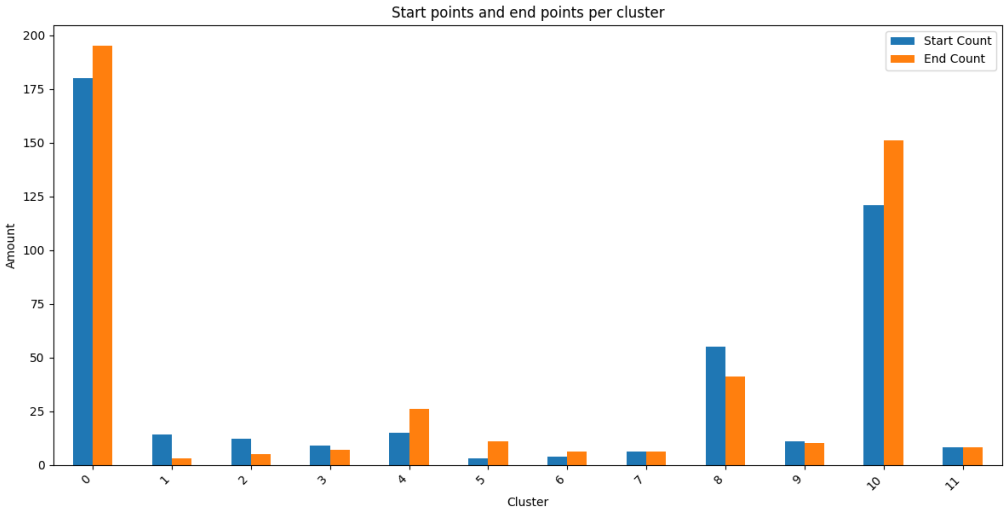

Figure 2.13: Cluster distribution of start points and end points

Clusters 8 and 10 are relatively well-defined, while cluster 0 appears too large and may benefit from being divided into smaller clusters. Figure 2.18 compares clustering with ranges of 200 and 550 for clusters 0 and 8. While splitting some clusters, for example cluster 0, could slightly enhance accuracy, others, such as cluster 8, should remain unified to accurately represent the journeys associated with them.

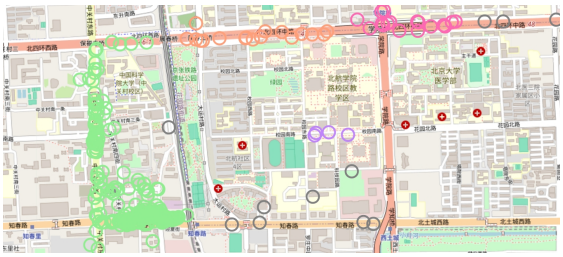

Figure 2.14: Cluster 0 with a range of 200 meters.

Figure 2.15: Cluster 8 with a range of 200 meters.

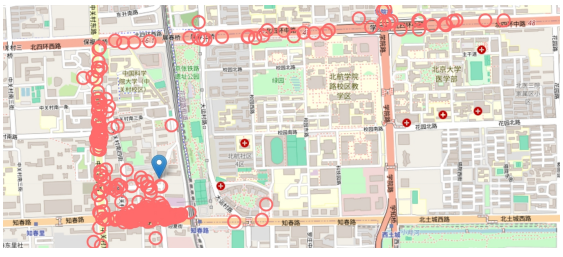

Figure 2.16: Cluster 0 with a range of 550 meters.

Figure 2.17: Cluster 8 with a range of 550 meters.

Figure 2.18: Comparison of cluster 0 and 8 with different clustering parameters.

Finding clustering parameters, that is small enough to ensure accurate predictions

but also does not require too much processing power, is challenging. Changing a single parameter can impact multiple aspects simultaneously. To define a metric for successful clustering, a Random forest model was developed, as detailed in Chapter 2.3.1. This model predicts the end cluster based on the start cluster, time of day, and day of the week.

A total of 234 clustering variations were tested with different radius and MinPts values. Due to the relatively small dataset, each variation was evaluated using 10 different training and validation splits, with the average accuracy displayed in Figure 2.19. Since the number of journeys between clusters depends on the clustering approach, Figure 2.20 presents the average accuracy multiplied by the number of journeys between clusters, representing the number of predictable, pattern-following journeys.

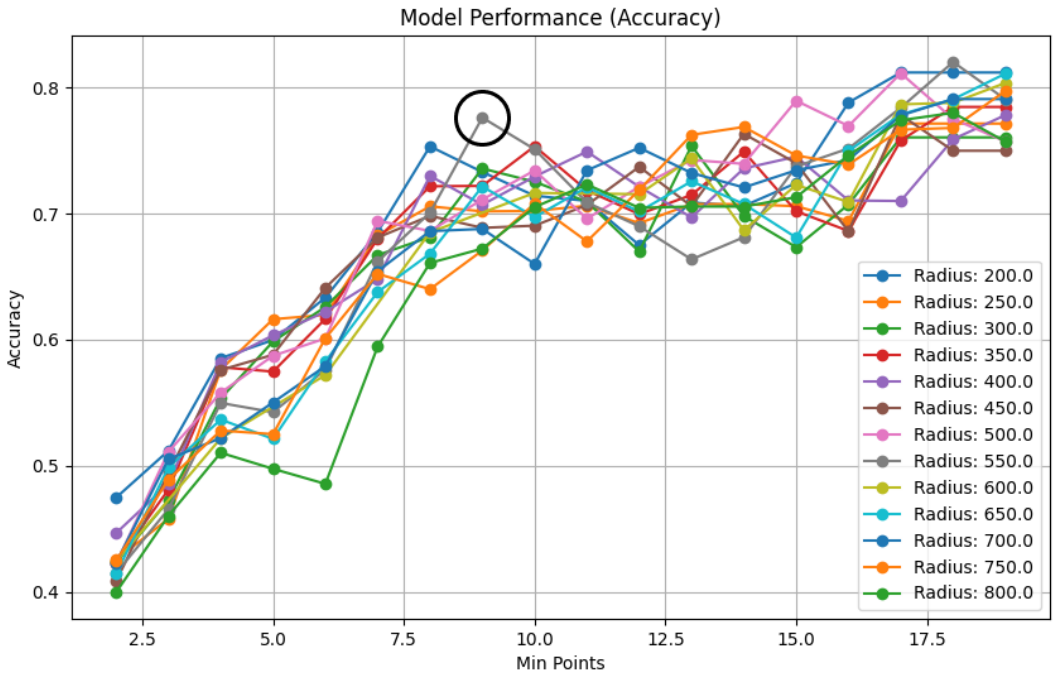

Figure 2.19: Random Tree accuracy over based on different clustering parameters, with radius = 550 and MinPts = 9 highlighted

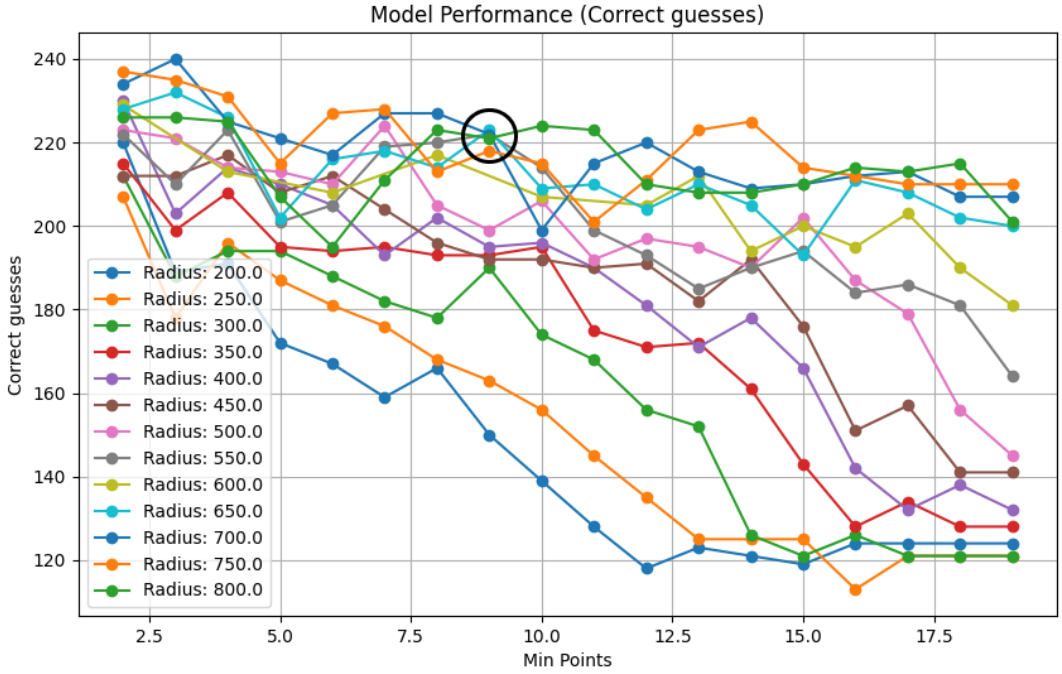

Figure 2.20: Estimated number of correct guesses based on different clustering parameters, with radius = 550 and MinPts = 9 highlighted

After analyzing the graphs, a clustering with a radius of 550 and a MinPts of 9 was selected, resulting in a total of 288 journeys between clusters. While other parameter combinations offer similar accuracy and a comparable number of estimated correct predictions, they are artificially accurate with fewer clusters. Reducing the number of clusters simplifies the prediction task but lowers fidelity, making the clusters less representative of the actual journeys.

After deciding on the clustering parameters, further filtering was applied. All journeys starting or ending at an unclustered point were removed, together with all journeys shorter than 1000 meters, as well as all journeys that start and end at the same point, resulting in Figure 2.21.

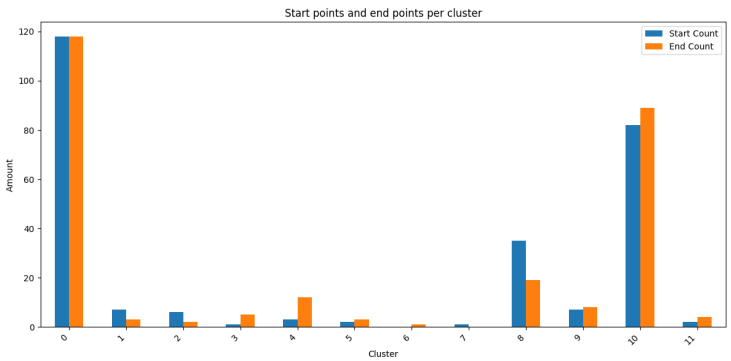

Figure 2.21: Start and endpoints after Filtering

Since clusters 6 and 7 each contain only a single start or end point, they were also removed. The final filtering resulted in 254 journey from 6 users across 10 clusters, shown in Figures 2.22 and 2.23.

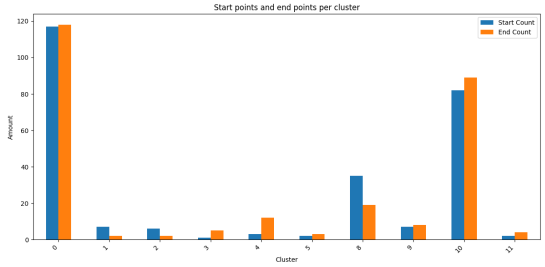

Figure 2.22: Start and endpoints after removing cluster 6 and 7

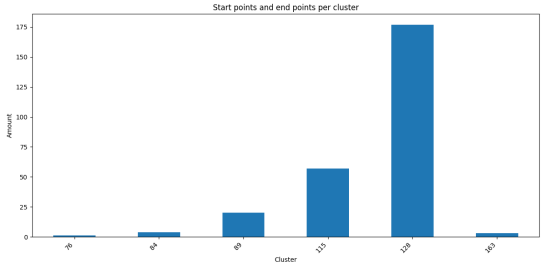

Figure 2.23: Number of journeys per user after filtering

2.2.3.1 Data structure

All filtered, matched, and clustered journeys are stored in a CSV file, with the following information.

- **Drive ID**: Unique identifier for the journey, matching GPX files of the journey before and after map matching.
- **OSM Nodes**: List of node IDs representing intersections along the route after map matching.
- **Start and End Labels**: Cluster labels for the journey's starting and ending points.
- **Start and End Coordinates**: Latitude and longitude of the journey's start and end.
- **Start Time**: The journey's starting time in ISO 8601 format [ISO, 2019].
- **Distance**: The map-matched distance in meters.
- **User**: The ID of the user who recorded the journey.

2.2.4 Adding weights based on driving patterns

By using ant colony optimization (ACO), the preferred routes of the driver can be found. ACO is an optimization model based on the natural pathfinding of ants. When serching for food, they send out ants on different paths, where all ants release pheromone. When one of the ants return after finding food, the pheromone level on that path will be stronger since an ant has traversed that route twice [Dorigo and Stützle, 2004].

In this project, one driving session of a car will be modeled in the same way as one ant is modeled. This means that the car will leave virtual pheromones, that is recorded and added to the graph as weights.

According to Doringo et. al., the pheromone level, that indicates to ants that other ants have chosen that way, can be updated in the following way [Dorigo et al., 2006].

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ traversed edge } (i,j) \text{ in its route,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

In this project, the $\Delta\tau_{ij}^{k}$ will represent the added pheromones of one driving session k for the car on a piece of road between intersection i and j. Q is a constant, which is set to $Q = 1$, and L_k is the length of driving session k. In order to add all the driving sessions, τ_{ij} will be updated in the following way, starting at zero.

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2.2}$$

τ_{ij} represents the total pheromone level on the road between intersection i and j. ρ is the pheromone evaporation rate, which should be in the interval $0 \leq \rho \leq 1$. $(1-\rho)$ stands for the decrease of pheromones over time, and $\rho = 0.0001$ for this project, because the pheromones should not disappear too quickly, but it is also important to lower the value on roads that have not been traveled in a long time. These pheromones represents the driving history of the car [FIL, 2023].

The updated pheromones can be used in a few different settings. Most importantly, all roads that has been driven is updated in a large map, and saved for reference to future journeys. This shows all roads that has ever been driven by the driver, and how many times they were driven on, based on pheromone level and time since it was driven on. When a new journey is in session, the pheromones are updated a single road id at a time, in order to get the correct pheromone levels as soon as possible. The pheromone indications can also be combined with the clusters from chapter 2.2.3.

2.2.4.1 Pheromones between clusters

The pheromones between clusters are stored in separate dictionaries. Each dictionary represents the roads traveled between cluster A and cluster B, along with the associated pheromone levels, where A and B indicate the start and end clusters among all possible combinations of clusters. After identifying which clusters the driver was traveling between, only the roads that the driver had driven on previously were considered to predict a path. This is visualized for journeys between cluster 0 to 8, in fig. 2.24. The color of the path shows which roads are more used than others with a brighter yellow indicating more previous journeys. The green dot represents cluster 0 and the red dot represent cluster 8.

Figure 2.24: The pheromones between cluster 0 and 8.

The driver choosing a previously driven route can be predicted, because humans tend to take the same route twice, over choosing a new route. This is because it brings a feeling of security and and saves energy when you do not need to look up how to use a new road [Colonna et al., 2016].

2.3 Destination prediction

To predict the route, a destination first needs to be predicted. Destination prediction can be split in two categories, prediction before the journey and prediction during the journey.

2.3.1 Pre journey prediction

To predict journey destinations before departure, a random forest classifier is used to model driver habits based on previous behavior. A random forest is an ensemble learning method that combines multiple decision trees trained on different parts of the data. Each tree contributes a vote, and the final prediction is determined by majority voting. The Random Forest model was chosen since it is less prone to overfitting and does a better job at generalization than normal classification trees and regression models [Breiman, 2001].

The model was trained to predict the end cluster of a journey using three key features: the start cluster, the time of day, and the day of the week. These features were extracted from the timestamp of each journey using a custom function, `get_time_weekday_from_iso`, which maps ISO 8601 timestamps to both weekday and a categorical time range. These time ranges were chosen based on assumptions about human behavior. Identifying optimal time ranges could be explored further in future work.

- morning (05:00 to 09:59)
- midday (10:00 to 14:59)
- afternoon (15:00 to 20:59)
- night (21:00 to 04:59)

Another advantage of using a Random Forest model is its ability to evaluate feature importance by analyzing which features are most frequently used in the decision trees that yield accurate predictions. In our case, both the start time and the starting cluster contribute significantly and roughly equally to the model's decisions, while the day of the week has considerably less impact, as shown in Table 2.2.

Feature	**Importance**
start_time	0.4336
start_cluster	0.4282
weekday	0.1382

Table 2.2: Feature importance based on Random Forest analysis.

While including the end cluster from the user's previous journey might seem reasonable to help predict journey sequences, this was intentionally excluded. The dataset consists of journeys from multiple users and does not cover all journeys made by any individual. As a result, personal journey history is inconsistent and potentially misleading. The final model configuration used a test size of 25% and employed 100 decision trees. Model performance is evaluated using accuracy, which ranged from 68% to 89% depending on the data split, with an average precision of 79.76% across 100 different train/validation splits. In addition to the predicted labels, the model also recorded the predicted probabilities for each possible end cluster. These probabilities were stored in a dictionary for use in later stages of the prediction pipeline.

2.3.2 During journey destination prediction

During an ongoing journey, two different strategies are employed to improve destination prediction: one that eliminates unlikely end clusters by routing, and another that actively predicts the most probable end cluster based on the remaining clusters and the trajectory driven so far.

The GNSS trajectory is continually map-matched during driving, and all intersection nodes traversed are stored. As each new intersection node is crossed, the program updates its internal state [Casabianca et al., 2021].

2.3.2.1 Routing based elimination

For every new intersection node passed during a journey, the program iterates through the list of possible end clusters and sends a routing request to OSRM. The request routes from the start cluster, through the current intersection, and then to each potential end cluster. An example of this process is shown in Figure 2.26, where the green marker represents the start point, the blue marker is the current location, and the red markers are the predicted end clusters. Figure 2.25 shows the actual route taken during the journey.

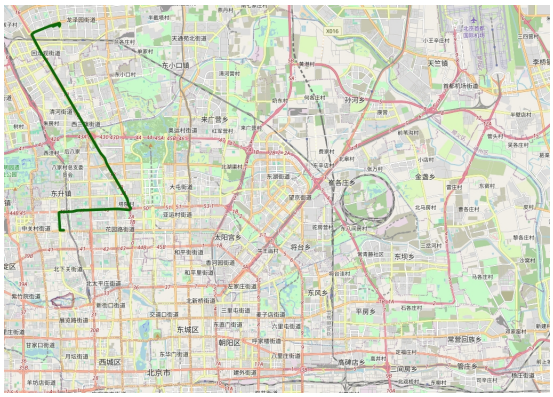

Figure 2.25: Actual route taken during the journey.

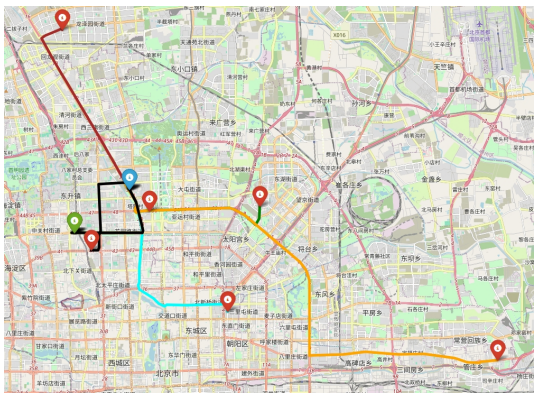

Figure 2.26: OSRM routing from start, via current point, to possible end clusters.

The routed distance is then compared to the average distance between the start and end cluster. If the OSRM distance is significantly longer, the corresponding end cluster is removed from the list of potential destinations. However, since people do not always follow the most efficient route, this method must account for some deviation. The relationship between the average recorded journey distances and OSRM-generated routing distances is illustrated in Figure 2.27, where the x-axis shows the recorded journey distances and the y-axis shows the corresponding OSRM routing distances. A summary of the numerical results is provided in Table 2.3.

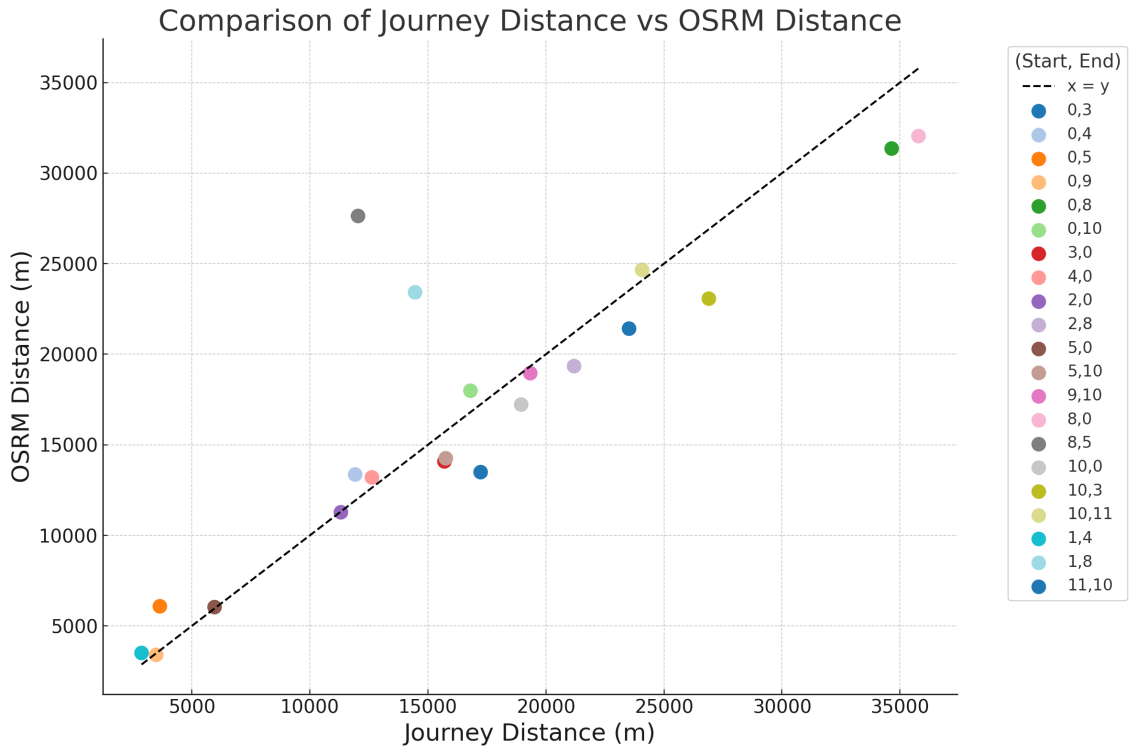

Figure 2.27: Comparison of average journey distances and OSRM routing distances.

Start, End	Avg. Journey Distance (m)	OSRM Distance (m)	Journey Count (n_q)	Percent Difference (x_q)
0,3	17236	13492	3	-21.72
0,4	11921	13362	8	12.08
0,5	3647	6086	2	66.85
0,9	3491	3403	7	-2.51
0,8	34647	31357	16	-9.50
0,10	16805	17988	79	7.04
3,0	15702	14085	1	-10.30
4,0	12631	13202	3	4.52
2,0	11316	11277	3	-0.34
2,8	21193	19344	1	-8.72
5,0	5965	6040	1	1.26
5,10	15760	14258	1	-9.53
9,10	19337	18955	7	-1.98
8,0	35784	32043	34	-10.45
8,5	12043	27635	1	129.48
10,0	18952	17221	76	-9.14
10,3	26903	23071	2	-14.24
10,11	24075	24656	4	2.41
1,4	2872	3509	1	22.18
1,8	14457	23421	2	62.00
11,10	23523	21415	2	-8.96

Table 2.3: Comparison of average journey distances and OSRM routing distances across clusters.

The percent differences shown in Table 2.3 are used to compute the weighted standard deviation using Equations 2.3 and 2.4, where n_q represents the number of journeys between start and end clusters, and x_q represents the percent difference between OSRM and actual distances.

$$\mu = \frac{\sum_{q=1}^{n} n_q x_q}{\sum_{q=1}^{n} n_q} \tag{2.3}$$

$$\sigma = \sqrt{\frac{\sum_{q=1}^{n} n_q (x_q - \mu)^2}{\sum_{q=1}^{n} n_q}} = 14.38 \tag{2.4}$$

The resulting weighted standard deviation is 14.38%, which is larger than the deviation observed for all major journeys (0→8, 8→0, 0→10, 10→0).
To ensure robustness while still allowing for natural deviations in routes, the threshold for routing-based end cluster elimination is set to 15% for normal clusters, if the destination probability for the chosen cluster is higher than 80% the threshold is increased to 30%. 30% is greater than two standard deviations and should include more than 95% of the journeys.

2.3.2.2 End cluster prediction based on trajectories

To predict the destination cluster based on its driven path so far, a deep learning model inspired by the architecture presented in [Casabianca et al., 2021] is used. The journey is represented by a sequence of intersection nodes. The model takes the node sequence and starting cluster as an input feature and tries to predict the end cluster. The model architecture is shown in Figure 2.28 and described below.

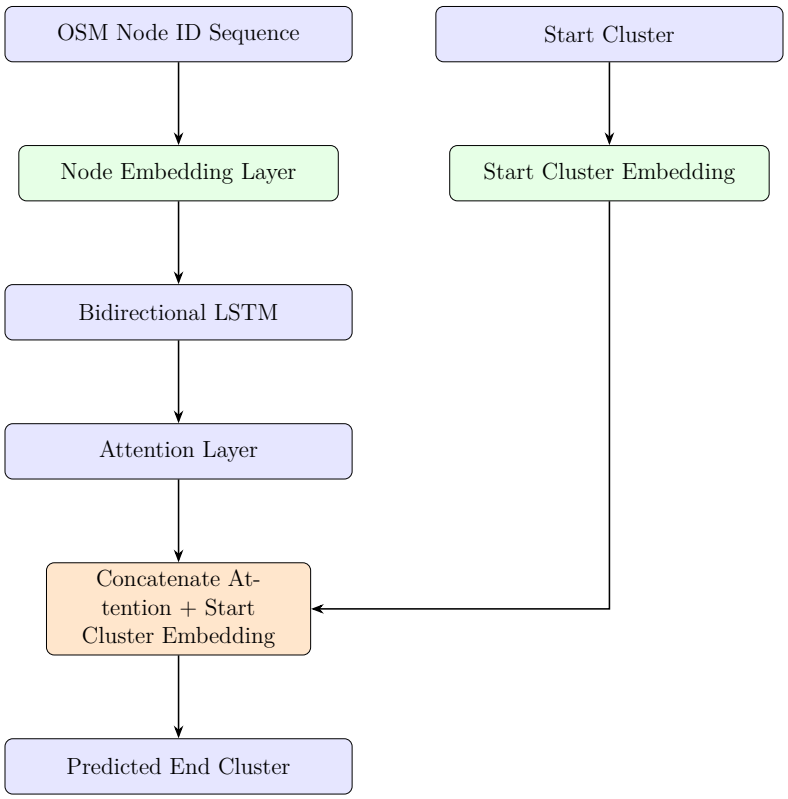

Figure 2.28: Model architecture for predicting end cluster using OSM node sequence and start cluster

- **Node Embeddings**: The OSM node IDs are first mapped into dense vectors using an embedding layer, allowing the model to capture spatial relationships and similarities between locations. The start cluster is embedded in a similar way.
- **Bidirectional LSTM (BLSTM)**: The embedded node sequence is passed through a bidirectional LSTM, which processes the sequence in both forward and backward directions. This helps the model capture the global journey context rather than only focusing on the most recent steps.
- **Attention Layer**: An attention mechanism is applied to the LSTM outputs to weigh the most relevant parts of the sequence for destination prediction. This results in a summary vector that emphasizes informative node transitions.
- **Final Prediction**: The attention-weighted summary vector, start cluster, embedding, feature are concatenated and passed through a dense output layer that predicts the end cluster.

In predictive modeling, typical dataset splits for training and validation are commonly in the range of 70%/30%, 75%/25%, or 80%/20%. Given the relatively small size of the dataset in this project, selecting an appropriate split required balancing two competing needs: providing sufficient data for effective model training, while also ensuring that the validation set includes enough diverse scenarios to enable meaningful performance evaluation. A split where 75% of the journeys make up the training data, with the remaining 25% used for validation, was therefore chosen. This configuration offered a practical compromise, ensuring that the model had enough data to learn from while still allowing meaningful evaluation across multiple scenarios [EMB, 2024].

During training, both the loss and accuracy are computed for each epoch and stored. The loss reflects the model's performance on the training data, lower values indicate better fit to the training set. The accuracy measures how well the model predicts the end cluster in the validation set and is reported for both the full journey and for only the first 60% of each journey. Measuring the accuracy after only 60% of the journey is useful for finding a model that works well with incomplete data, since the model will be used mid-journey to make predictions.

Although the loss generally decreases as the number of training epochs increases, a lower training loss does not necessarily indicate improved model performance. If the validation accuracy remains constant while the loss continues to drop, it is often a sign of overfitting - where the model becomes too specialized to the training data and struggles to generalize to new data. To address this, we monitor validation accuracy and stops the training when validation accuracy doesn't improves over 10 consecutive epochs. Ending training based on validation is a common method for used to reduce overfitting in deep learning algorithms and is used by [Casabianca et al., 2021]. However, since accurate predictions for partial journeys is a key goal for our purpose. The best-performing model is selected based both the validation accuracy at 100% of the journey and 60% of the journey. If the 100% validation accuracy has plateaued, but the accuracy at 60% of the journey has improved then training is continued. The first model to achieve the highest validation accuracy for both is

saved and used for the remainder of the project.

a lower loss is not always more desirable, as it is often indicates overfitting. To counteract this, validation accuracy for 100% and 60% of the journey is used to select the best performing model. Training stops the 100% validation accuracy has not improved or stayed the same while the 60% improves for 10 consecutive epochs. The first model that achieves the highest validation accuracy is saved and used in the remainder of the project.

[Casabianca et al., 2021]

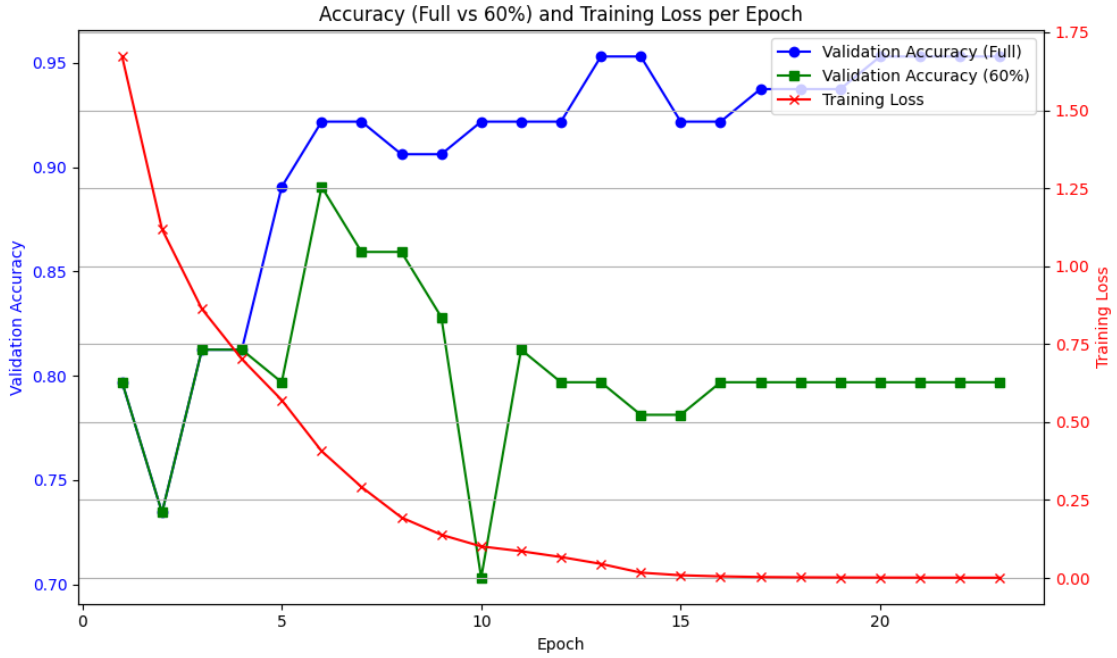

Figure 2.29: Example of model training with best result at epoch 13

2.4 Route prediction

The driver's habits are reflected in the τ_{ij} calculated in chapter 2.2.4. However, general driving patterns can vary based on the time of day and day of the week. To improve prediction accuracy, a time-based filter is applied.

The GNSS data includes timestamps and dates for each test drive. When predicting a route, the system first considers the current time and day. In addition to this temporal context, the driver's historical behavior is also incorporated. If the predicted route for the current time is not taken, the model prioritizes destinations the driver frequently visits.

2.4.1 Route based on destination

The start and end cluster is calculated in accordance to chapter 2.2.3. When they have been calculated, the roads that have previously been traversed between the given clusters are looked at. The road with the most pheromones is predicted to be the most likely road to start the route.

2.4.2 During journey route prediction

One important part of the route prediction is to continuously update the prediction during the journey. As the car is driving to certain destinations, paths will become increasingly unlikely. By eliminating these paths and recalculating the route and destination during the journey, a more confident prediction can be made.

The algorithm is looking at the current road id and which node id is the next on the path. In this way, on route prediction can be done by assuming which road is going to be chosen next, and updating where the car is live.

For predictions that are further ahead on the route, the algorithm looks at which clusters still are possible to travel to. All clusters that have been previously visited are possible to be visited again, but during the journey there will be less and less possible clusters left.

2.5 Sending warnings based on traffic situations

In the case of traffic jams, the car is to be warned in order to be able to change route to the chosen destination. This could be done by an icon on the dashboard and via the digital display.

2.5.1 Simulating slow traffic

This project is limited to using simulated traffic jam signals in order to validate the system. These are created by manually declaring specific road IDs in the system as experiencing slow traffic.

2.5.1.1 Traffic node alert logic

To assess whether a vehicle is at risk of encountering a known traffic node before reaching it, the system analyzes the journey path in real-time based on predicted end clusters and historical trajectories.
For each predicted end cluster, which comes with an associated probability from the model seen in section 2.3.1, the algorithm performs the following steps:

1. Retrieves a set of historical journeys between the start cluster and a predicted end cluster from the list.
2. Checks whether any of the predefined traffic nodes appear in those historical journeys.

3. If a traffic node is found in a journey it checks weather the current node also appears in the same historical journey. If it does:
   (a) It computes the number of intersections (`nodes_until_traffic`) between the current node and the traffic node.
   (b) It aggregates the probability of reaching that traffic node based on the model's predicted probability for the end cluster.

After doing this for all journeys for all predicted end clusters the program continues.

1. A traffic warning is triggered under the following conditions:
   - If the vehicle is predicted to reach the traffic node in fewer than x intersections and the cumulative probability of all journeys reaching the traffic node exceeds y%.
2. If a traffic warning is triggered and the journey ends up driving into the traffic node it is counted as a successful warning.
3. If a traffic warning is triggered and the journey does not end up driving into the traffic node it is counted as a false warning.
4. If the vehicle pass the traffic node before a traffic warning is triggered, the system returns a "too late" status.

3

Results

3.1 Random forest

Since the dataset is relatively small, with only 254 journeys from 6 different users, the performance of the Random Forest model is sensitive to how the data is split between training and validation. To capture this variability, the model was evaluated 100 times using different seeds for the data split. The resulting distribution of accuracies is shown in Figure 3.1.

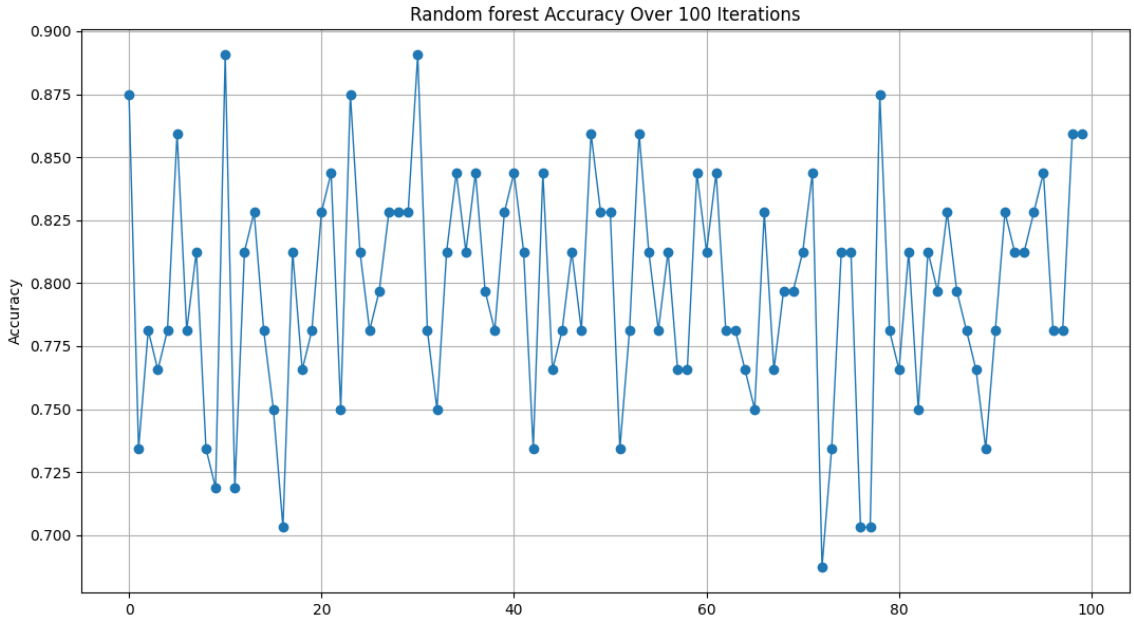

Figure 3.1: Accuracy of the Random Forest model over 100 different data splits.

The average accuracy across all runs is 79.76%, though individual results vary considerably. Table 3.1 shows the feature importance values for the run with the lowest accuracy, while Table 3.2 shows the same for the run with the highest accuracy.

Feature	Importance
start_times	0.5001
start_cluster	0.3967
weekday	0.1031
Accuracy	68.75%

Table 3.1: Feature importance and accuracy for the lowest accuracy data split, random state = 72.

Feature	Importance
start_times	0.4551
start_cluster	0.4226
weekday	0.1223
Accuracy	89.06%

Table 3.2: Feature importance and accuracy for the highest accuracy data split, random state = 30.

These tables show that a balance between the features start times and start clusters is desirable with a slight over reliance on start_times. Also that the over reliance on the start cluster and weekday features results in a worse precision.
For the remainder of the results, an average data split with random state = 26 will be used to accurately represent the average capabilities of the model, with a accuracy of 79.68% Shown in table 3.3.

Feature	Importance
start_cluster	0.4299
start_times	0.4206
weekday	0.1494
Accuracy	79.68%

Table 3.3: Feature importance and accuracy for the average accuracy data split, random state = 26.

3.2 BLSTM end cluster prediction

The trajectory-based prediction task was trained using a bidirectional LSTM model with an attention mechanism, inspired by a similar project by Casabianca et al [Casabianca et al., 2021]. To illustrate the results clearly, three graphs are shown for each of the chosen data splits. The first graph (Figure 3.8) shows the average prediction accuracy when validating on only the first x nodes (from 1 to 117). The second graph (Figure 3.9) shows the accuracy when validating on the first x% of the journey (from 1% to 100%). The third figure presents the training details, as described in Section 2.3.2.2.

3.2.1 BLSTM prediction for the lowest accuracy data split

The best model from the split where the random state was 72, was saved at epoch 19, achieving a final accuracy of 93.75% at 100% journey completion and 90.62% at 60%. As shown in Figure 3.2, the model reaches peak performance after just 54 nodes. Figure 3.3 shows that accuracy at 60% and 100% journey completion are relatively similar. However, this performance is not consistent throughout training. As seen in Figure 3.4, the 60% accuracy fluctuates significantly before and after epoch 19.

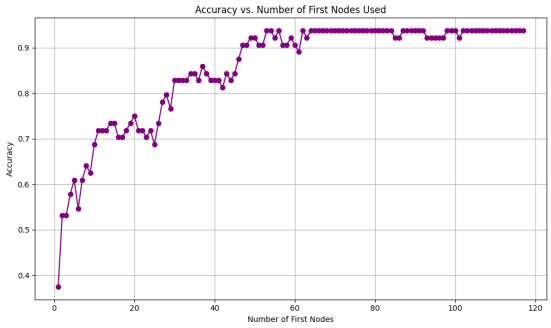

Figure 3.2: Prediction accuracy for the lowest accuracy data split, as a function of the number of nodes used (1 to 117).

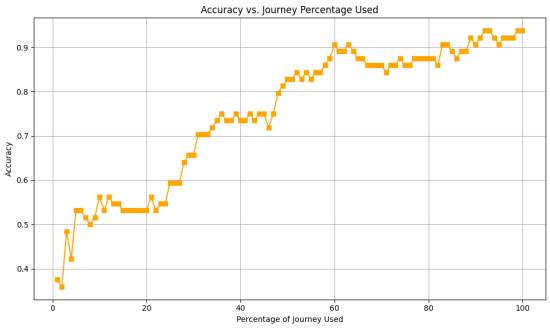

Figure 3.3: Prediction accuracy for the lowest accuracy data split, as a function of the percentage of the journey observed (1% to 100%).

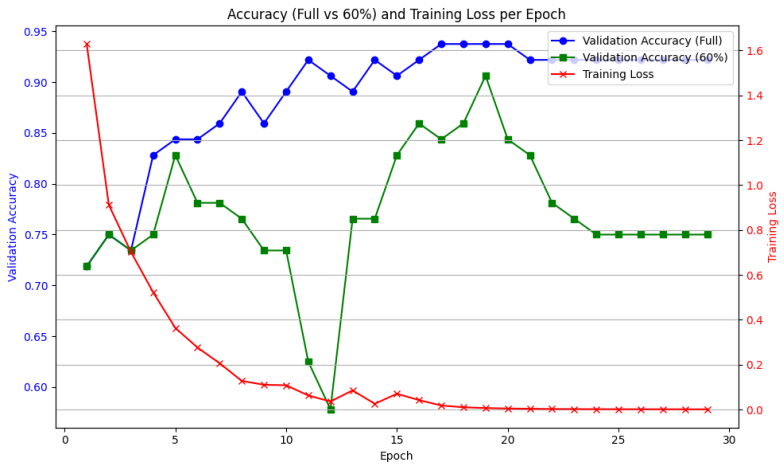

Figure 3.4: Training and validation loss/accuracy curves for the lowest accuracy data split during training.

3.2.2 BLSTM prediction for the highest accuracy data split

The model where the random state was 30, was saved at epoch 16, with a final accuracy of 100% at full journey validation and 90.20% at 60%. As shown in Figure 3.5, the model reaches peak accuracy after 81 nodes. Figure 3.6 shows a relatively stable

accuracy of around 87%+-4% from 20% to 65% of the journey, with a noticeable increase at 70%. Full accuracy is only reached with complete journeys.

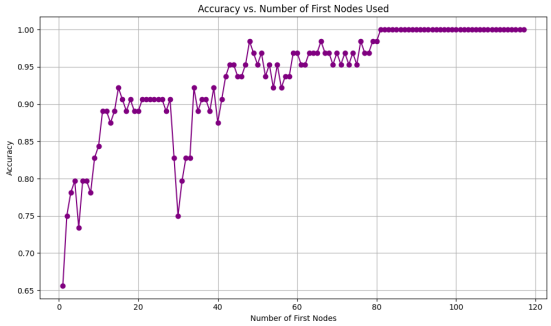

Figure 3.5: Prediction accuracy for the highest accuracy data split, as a function of the number of nodes used.

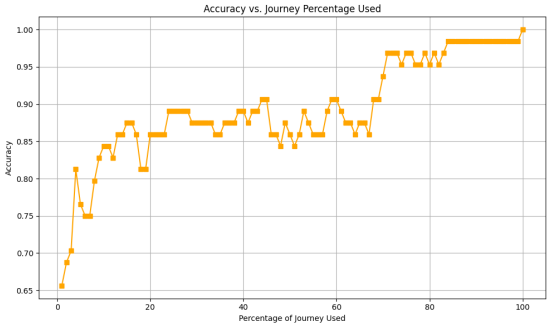

Figure 3.6: Prediction accuracy for the highest accuracy data split, as a function of the percentage of the journey used.

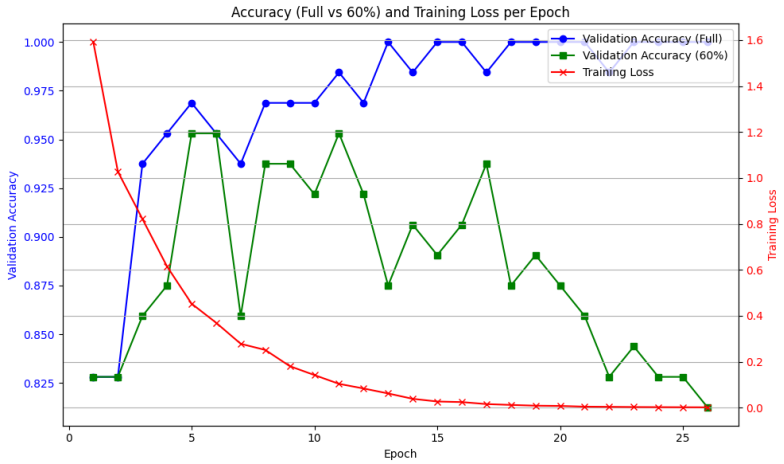

Figure 3.7: Training and validation loss/accuracy curves for the highest accuracy data split during training.

3.2.3 BLSTM prediction for the average accuracy data split

The model where the random state was 26, was saved at epoch 13, achieving a final accuracy of 95.31% at full journey completion and 79.69% at 60%. As shown in Figure 3.8, the model is highly volatile when validating with fewer than 35 nodes, after which it shows a steady improvement in accuracy, peaking at 73 nodes. Figure 3.9 shows a similarly unstable performance until about 30% of the journey, after which accuracy improves consistently, reaching its maximum just before the end of the journey.

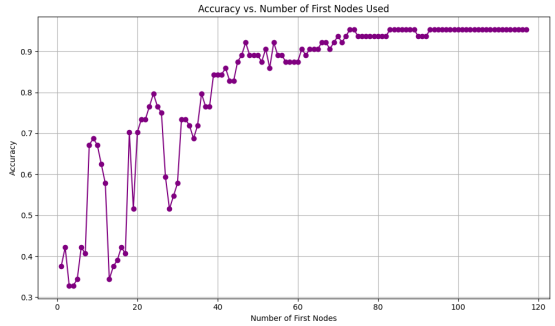

Figure 3.8: Prediction accuracy for the average accuracy data split, as a function of the number of nodes used.

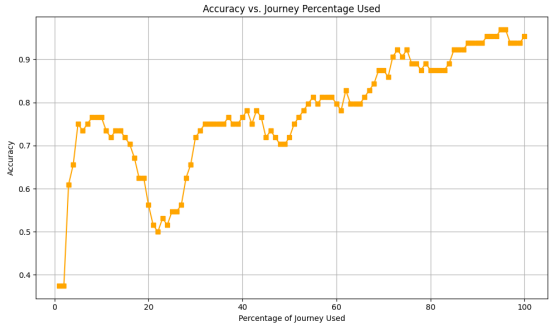

Figure 3.9: Prediction accuracy for the average accuracy data split, as a function of the percentage of the journey used.

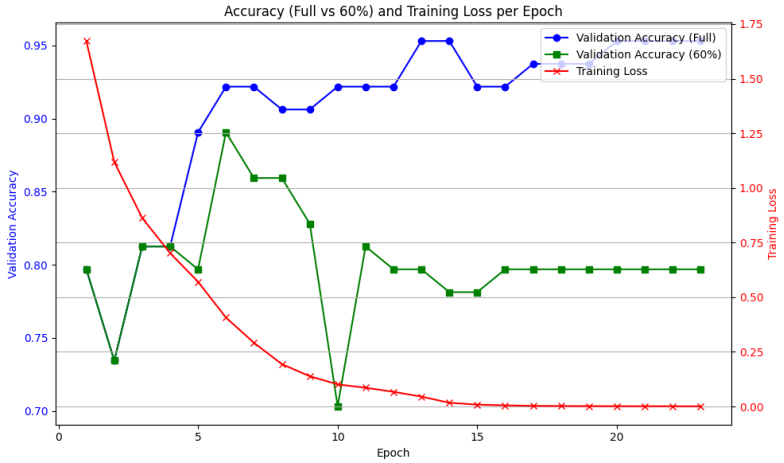

Figure 3.10: Training and validation loss/accuracy curves for the average accuracy data split during training.

Reflecting on the three model iterations, we observe that the model performs reliably after around 40 nodes and reaches near-peak performance after 60 nodes. Additionally, the accuracy at 60% and at full journey completion are not necessarily correlated. In many cases, 60% accuracy tends to decline as the model becomes more overfitted with additional training epochs.

3.3 Destination prediction

To evaluate the performance of the different destination prediction methods, all scenarios are tested at every 10% interval of the journey, from 0% to 100%. This allows for a gradual understanding of how each model performs as more information becomes available during the trip. The description of these scenarios can be seen below.

- **Random Forest:** The Random Forest model is calculated at the start of each journey and, as can be seen in the figures below, does not change during the test.

- **BLSTM:** The BLSTM model generally improves the further the car has traveled, since it gains access to more nodes for the prediction.

- **Combined Random Forest and BLSTM:** When combining the Random Forest and BLSTM models, the Random Forest prediction is trusted until 40 nodes have been traveled. After this, the confidence of the Random Forest prediction is checked. If the confidence for the most likely cluster is higher than 50%, the Random Forest is still trusted; otherwise, the BLSTM model is used. After 60 nodes, only the BLSTM model is used. The 40 and 60 node cutof points are based on section 3.2

- **Routing-Based Elimination:** This method is heavily based on the initial guess from the Random Forest model and has not been tested independently. It starts working after 10 nodes, since a certain distance must be traveled before the routed distance through the current node can differ from the calculated average distance between the clusters. After this point, it activates every 4 nodes. This frequency is due to the high computational cost of making multiple routing requests (potentially requiring routes to 8 end clusters each time), and the fact that there often isn't a significant change in distance traveled at every node.

- **Full Combination:** Combining all three methods follows similar principles to those described above.

Looking at the worst data split, we can see that the BLSTM is highly volatile but generally performs better than the combined version with Random Forest. This is likely due to the data split being particularly unfavorable for the Random Forest model. While the performance of the BLSTM is not penalized as sharply. We also observe that the routing-based scenarios perform better overall, with accuracy steadily increasing throughout the journey. The RF + Routing scenario surpasses others after 60–70%, while the fully combined model becomes the best at 50% and again from 80% onward, ending with an accuracy of approximately 89.06%.

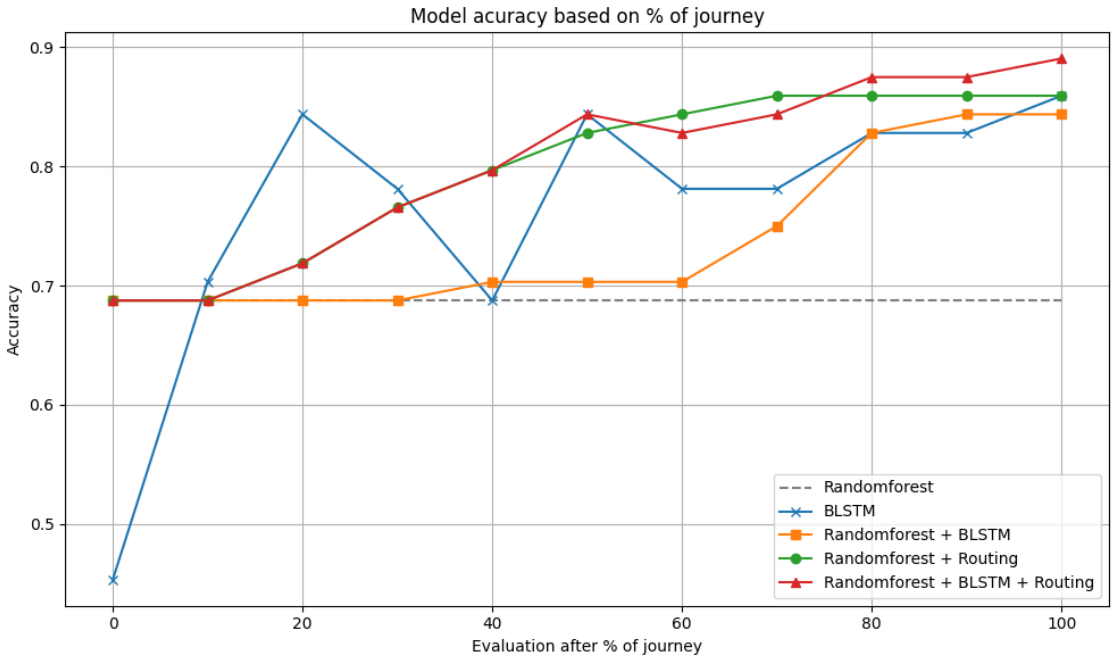

Figure 3.11: Performance comparison of all models on the worst data split.

In the best data split, the BLSTM performs worse than the combined BLSTM + Random Forestapproach until 40% of the journey is completed. Both routing-enhanced scenarios perform similarly, with the model using all three methods slightly outperforming the Random Forest + Routing at 60%. All models except for Random Forest and Random Forest + BLSTM end with an accuracy of approximately 96.87%.

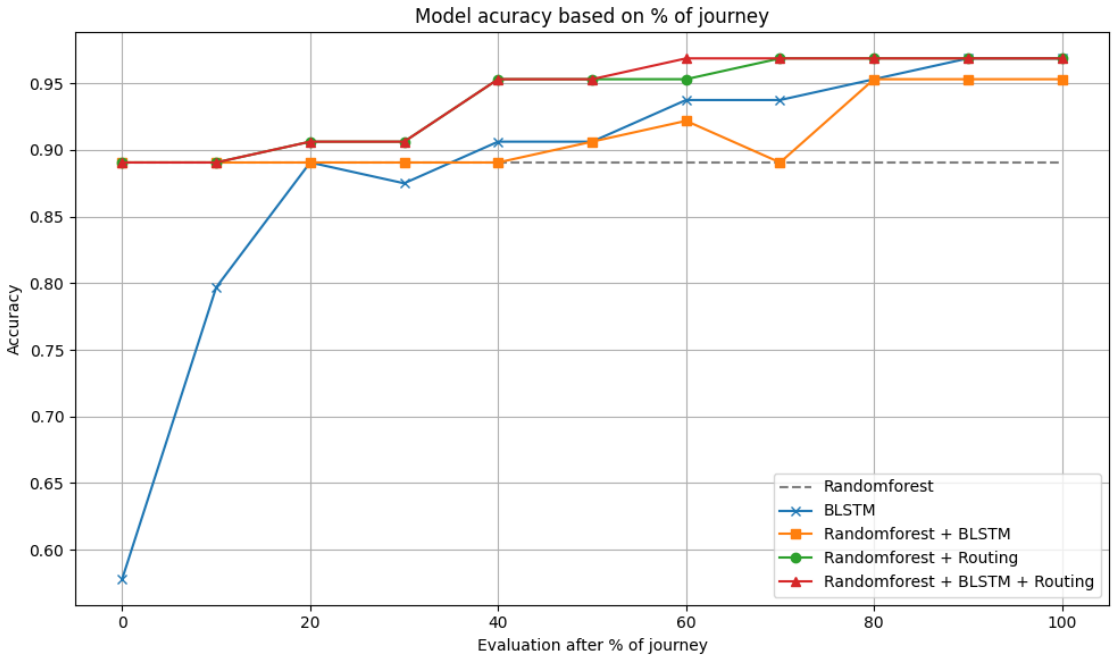

Figure 3.12: Performance comparison of all models on the best data split.

The average data split shows a different trend. The BLSTM performs significantly worse than the other models, only surpassing the Random Forest at around 70% of the journey. This result aligns more closely with expectations and highlights the benefit of the ensemble approach, which aims to provide robust predictions from start to finish. The routing-based models show a steady improvement up to 70%, where the fully combined model slightly drops. The models with routing peaks at 90.63%. While the BLSTM model continues reaches an accuracy of 92.19% at 100% of the journey completed.

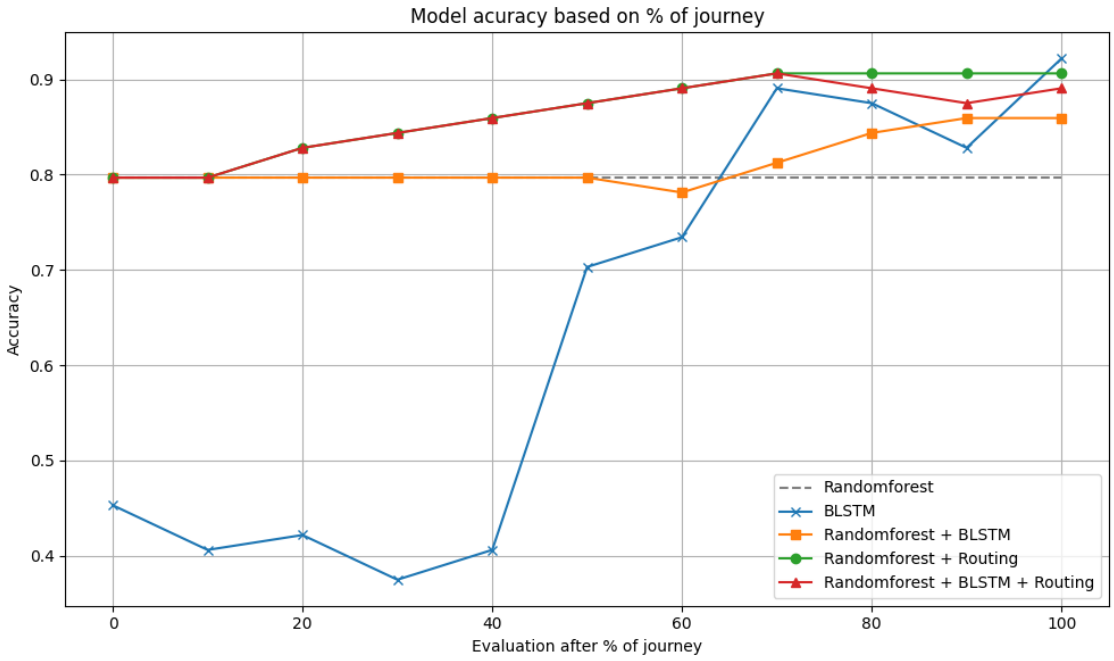

Figure 3.13: Average performance of all models on the average data split.

3.4 Traffic warning evaluation

The following tables present an evaluation of the system's ability to warn the driver of upcoming traffic congestion using the combined model with all Random forest + Routing + BLSTM. Each evaluation is based on two factors: the warning threshold — how many nodes before the congestion the system should issue a warning, and the model's prediction confidence, tested at thresholds above 40% and 90%. A warning can be considered in three different ways:

- **Correct** if the driver was warned in time before entering the congestion.
- **False** if the driver was warned, but was not actually going to drive through the congestion.
- **Too Late** if the system failed to warn the driver in time, causing them to enter the congestion.

For the evaluation, five commonly occurring nodes were selected to act as traffic congestion points. These nodes were chosen in connection with important intersec-

tions where the route differs between journeys and are shown in Figure 33.14. In addition to these five nodes, a sixth node was selected at random along the route of the current journey, at least 15 nodes after the starting point.

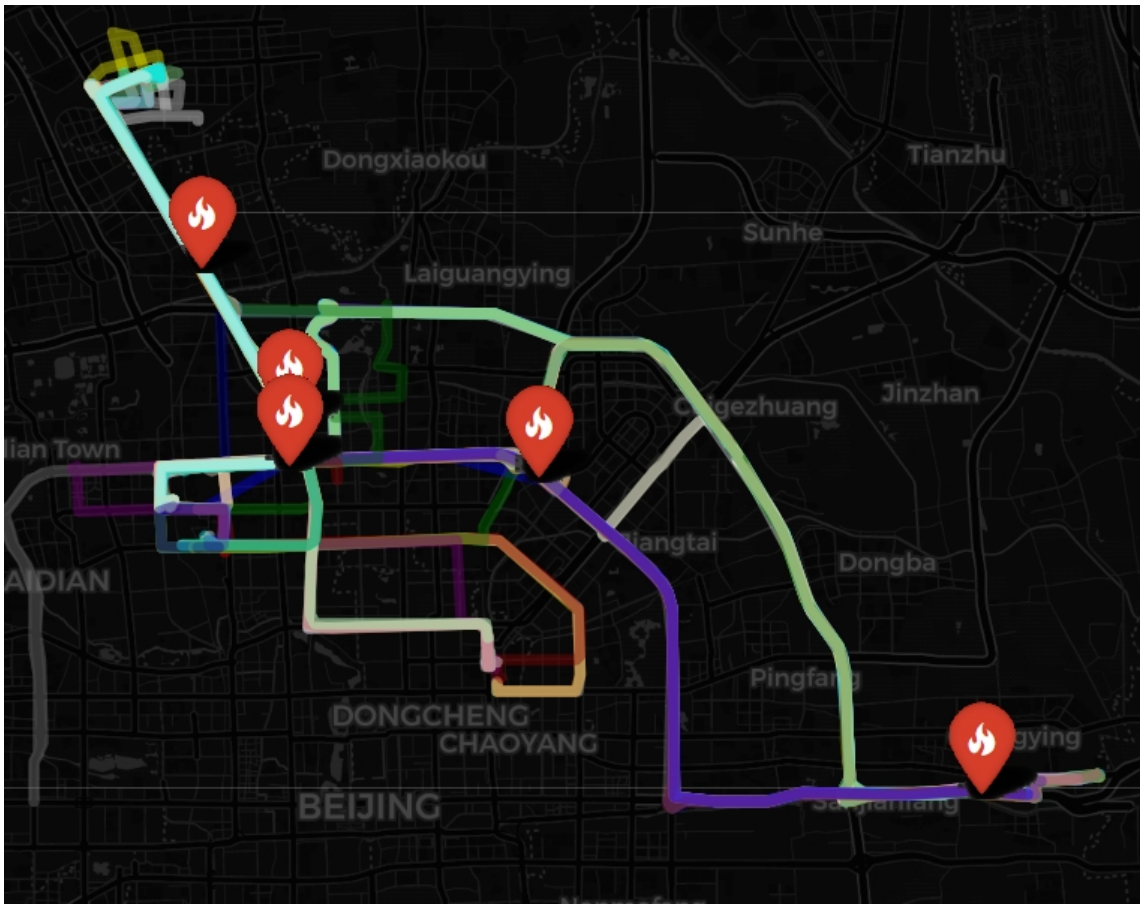

Figure 3.14: Map showing the location of the selected nodes of congestion

For the worst data split (random state = 72), the overall performance is around 75-80%, with a lot of false warnings and late detections. The best result here is with a 5-node threshold and 40% confidence, where 82.81% of warnings were correct. Higher confidence thresholds tend to result in more late warnings across all thresholds while slightly decreasing the false warnings, at the cost of a slightly less correct warnings.

Warning Threshold (nodes before traffic)	Confidence > 40%	Confidence > 90%
20	Correct: 76.56% False: 14.06% Too Late: 9.38%	Correct: 76.56% False: 12.50% Too Late: 10.94%
10	Correct: 81.25% False: 7.81% Too Late: 10.94%	Correct: 78.12% False: 6.25% Too Late: 15.62%
5	Correct: 82.81% False: 4.69% Too Late: 12.50%	Correct: 78.12% False: 4.69% Too Late: 17.19%

Table 3.4: Traffic warning performance for the lowest accuracy data split (random state = 72).

In the best case (random state = 30), accuracy is relatively high for all settings. The best result is again at 5 nodes and 40% confidence, where 95.31% of the warnings were correct and only 3.12% were too late. Increasing the confidence threshold reduces the number of false warnings slightly, but leads to more late ones resulting in fewer corect warnings, especially at shorter thresholds.

Warning Threshold (nodes before traffic)	Confidence > 40%	Confidence > 90%
20	Correct: 89.06% False: 7.81% Too Late: 3.12%	Correct: 87.50% False: 7.81% Too Late: 4.69%
10	Correct: 87.50% False: 6.25% Too Late: 6.25%	Correct: 85.94% False: 4.69% Too Late: 9.38%
5	Correct: 95.31% False: 1.56% Too Late: 3.12%	Correct: 89.06% False: 1.56% Too Late: 9.38%

Table 3.5: Traffic warning performance for the highest accuracy data split (random state = 30).

The average case (random state = 26) lands somewhere in the middle. Once again, the highest correct rate is at 5 nodes and 40% confidence, with 85.94%. False warnings are surprisingly low for this data split, especially for the 10 node runs. Increasing the confidence threshold for this split does nothing to reduce the number of false warnings, it only increases the number of late warnings.

Warning Threshold (nodes before traffic)	Confidence > 40%	Confidence > 90%
20	Correct: 82.81% False: 6.25% Too Late: 10.94%	Correct: 81.25% False: 6.25% Too Late: 12.50%
10	Correct: 82.81% False: 1.56% Too Late: 15.62%	Correct: 84.38% False: 1.56% Too Late: 14.06%
5	Correct: 85.94% False: 1.56% Too Late: 12.50%	Correct: 81.25% False: 1.56% Too Late: 17.19%

Table 3.6: Traffic warning performance for the average accuracy data split (random state = 26).

Overall, we see the same pattern in all data splits: lowering the confidence threshold gives more correct warnings overall, but can sometimes increase the number of false ones. A higher threshold results in fewer warnings, which leads to more cases where the driver ends up in congestion without being warned. False warnings can be annoying and, depending on how they are presented, might reduce trust in the system. But missing a warning altogether means the driver ends up in a traffic jam they could have avoided. A transparent system that shows where traffic is expected, by highlighting it directly on the car's navigation tool, could reduce the frustration caused by false warnings. The information is still correct, there is traffic, it just does not impact the driver. This might not reduce trust as much, since the driver can judge the relevance. In contrast, a system that only sends a vague alert like "traffic ahead" runs the risk of being wrong and losing credibility. With a more transparent approach, it is arguably better to give a few extra warnings than to leave the driver unaware of relevant ones.

Worth to note is that a few of the false warnings are due to bad data, as the matching of the roads sometimes doesn't follow reality, making journeys jump back and fourth between parallel roads seen in Figure 2.6 or skipping some intersections entirely, resulting in some traffic nodes being narrowly missed, when in reality they should have been driven on.

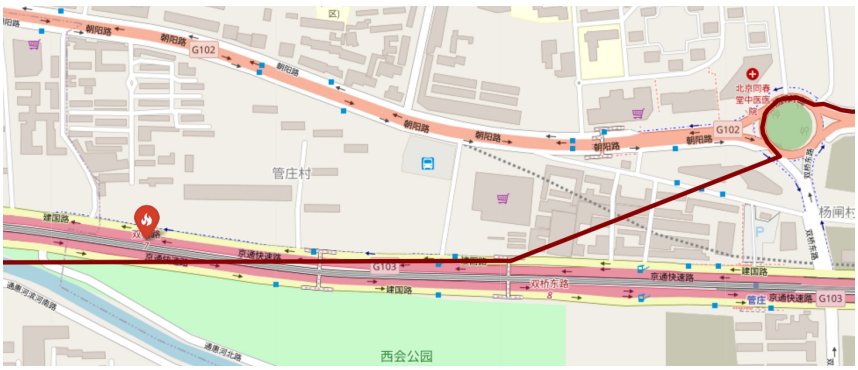

Figure 3.15: Example of a false "False" warning

4

Discussion

While the project gave some valuable insights, some parts of this project did not progress as expected or required more time than anticipated for evaluation. This discussion aims to highlight these challenges to prevent similar issues in future work and to offer insights and inspiration for potential improvements.

4.1 Data

Some obstacles regarding the data of this project were found, and is discussed in the following segments.

4.1.1 Data from personal vehicles

The data used in this project came from an open dataset of vehicles in Bejing. Although it offered a large volume of trip information, it was not ideal for modeling individual driving habits. If this project is to be further developed, then it would be beneficial to find driving data from personal vehicles. The data used in this project was gathered by phone, and let the users share their data when they wanted to, which led to gaps in the data. In the Bejing dataset, there was also a mixture of different users, and therefore the larger clusters was locations like the university or a central street. The goal for this project was not to find the popular places in Bejing, but to understand real peoples driving habits, in order to warn them in correct situations.

4.1.2 Data quality and filtering

The raw data needed thorough filtering to ensure useability. Many sequences were incomplete or contained abrupt transitions, such as mid-route app closures or changes in transportation mode. These inconsistencies caused uncertainty in determining true start and end points for journeys. The dataset also included a wide variety of vehicle types, as well as data for multiple users. It was hard to get enough suitable data from this dataset, since a lot of it had to be filtered out, and could not be used. If the mapping was on the car instead of in an app, the end cluster could be used as the next journeys start cluster, but now it had to be calculated for each new journey.

4.1.3 Impact of poor data quality

Because the data had gaps, many start and end points were unreliable. In a case where many faulty "starts" and "stops" happens i a close radius, creates a new cluster which might confuse the algorithm into believing that the driver is on its way there. resulting in needing a high threshold for creating new clusters, 9 or more, start or end points. Given more time, even more resources would have been allocated to improving the filtering, after spending more than half of the project time on pre processing the data trying to optimize filtering, it was decided to instead try to remove faulty journeys, resulting in only around a fourth of the journeys from the initial filtering remaining, meaning a relatively limited dataset of only 255 journeys.

4.1.4 Impact of a limited dataset

Training and testing an AI model with a limited dataset can significantly affect its performance and reliability. When the data available is not sufficiently large or diverse, the model may struggle to learn generalizable patterns. This often leads to overfitting, where the model performs well on the training data but poorly on new, unseen examples.
In the context of this project, the limited dataset means that the model's understanding of different journey patterns and traffic scenarios may not be fully representative. As a result, the warning system could behave inconsistently when applied to real-world driving. For example, it might issue accurate warnings in situations it has seen before but fail to generalize to slightly different routes or traffic conditions. A smaller dataset also makes the model more sensitive to the specific split used for training and testing. As seen in the result, changing the random seed can noticeably impact performance. This variability underlines the importance of using a larger and more varied dataset to build a robust model.

4.2 Geographical spread

The geographical spread of the data was limited to the region of Bejing. If this project is to be integrated in a car's system, it is possible to use maps from all around the world. However, scaling the system would also require efficient data handling and potentially region-specific model tuning, which was beyond the scope of this project but should be considered in future development.

4.3 Time-of-day and seasonal driving trends

Another aspect for further exploration involves analyzing patterns in driving behavior over time. This includes examining trip frequency and variability across different times of day, days of the week, and even across seasons or months. Studying these trends at finer time intervals may reveal insights that enhance the accuracy of predictive models.

5
Conclusion

This thesis demonstrates that a combined approach using Random Forest, routing-based elimination, and Bidirectional LSTM (BLSTM) models is effective for predicting driving destinations and issuing early warnings for upcoming traffic congestion. The ensemble model was capable of correctly predicting upcoming traffic incidents in 58 out of 64 scenarios, achieving a correctness rate of over 90% in the best-case configurations.

The Random Forest model alone showed moderate reliability, with an average prediction accuracy of around 79.7%, though its performance varied significantly depending on the data split. The BLSTM model offered stronger results, achieving up to 100% accuracy in favorable splits and 95.3% on average by the end of each journey. However, its performance was highly volatile in early-stage predictions and sensitive to overfitting.

By combining the strengths of each method - leveraging Random Forest's early-stage stability, BLSTM's sequential learning capabilities, and routing-based spatial reasoning - the full system maintained high accuracy throughout the journey, even under less ideal data splits. The combined model not only enhanced final destination predictions but also improved the timeliness and reliability of traffic warnings, particularly when tuned with dynamic confidence thresholds and node-based trigger points.

5.1 Future work

Despite these promising results, the system's accuracy and robustness could be further improved. Key opportunities for future work include:

- Using data from personal vehicles with consistent logging.
- Matching the data to the road network with more advanced matching methods.
- Enhancing route continuity by linking consecutive trips.
- Expanding to multi-region or global models.
- Integrating real-time traffic feeds and hazard data.
- Testing the system performance in live vehicle environments.
- Exploring optimal time split strategies for analysis.

In conclusion, the hybrid modeling approach proposed in this thesis offers a solid

foundation for intelligent, proactive traffic management systems. With further refinement, it holds strong potential to contribute meaningfully to safer and more efficient driving experiences.

Bibliography

[ISO, 2019] (2019). ISO 8601: Date and Time Format. Available: https://www.iso.org/iso-8601-date-and-time-format.html.

[Advanced navigation, 2023] Advanced navigation (2023). Global navigation satellite system (gnss) and satellite navigation explained.

[Boeing, 2024] Boeing, G. (2024). Modeling and analyzing urban networks and amenities with osmnx. page 16.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

[Casabianca et al., 2021] Casabianca, P., Zhang, Y., Martínez-García, M., and Wan, J. (2021). Vehicle destination prediction using bidirectional lstm with attention mechanism. *Sensors*, 21:8443.

[Çolak et al., 2016] Çolak, S., Lima, A., and González, M. C. (2016). Understanding congested travel in urban areas. *Nat Commun*, 7:10793.

[Colonna et al., 2016] Colonna, P., Intini, P., Berloco, N., and Ranieri, V. (2016). The influence of memory on driving behavior: How route familiarity is related to speed choice. an on-road study. Available at: https://www.sciencedirect.com/science/article/pii/S0925753515002738 (Accessed: 2025-06-05).

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and St¨utzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):12.

[Dorigo and Stützle, 2004] Dorigo, M. and Stützle, T. (2004). Ant colony optimization. *A Bradford book*.

[EMB, 2024] EMB, T. (2024). Train and test data: Best practices for machine learning. Available at: https://blog.emb.global/train-and-test-data/ (Accessed: 2025-06-13).

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 226–331.

[FIL, 2023] FIL (2023). Ant colony optimization,a guide to swarm intelligence for optimization.

[GraphHopper, 2025] GraphHopper (2025). Graphhopper map matching api documentation. Available at: https://docs.graphhopper.com/openapi/map-matching/postgpx (Accessed: 2025-06-05).

[NCAP, 2024] NCAP, E. (2024). Assessment protocol – safety assist. Available at: https://www.euroncap.com/en/for-engineers/protocols/safety-assist/ (Accessed: 2025-02-03).

[OpenStreetMap, 2025] OpenStreetMap (2025). Node. *OpenStreetMap Wiki*.

[OSRM, 2025] OSRM, P. (2025). Osrm api documentation v5.24.0. Available at: https://project-osrm.org/docs/v5.24.0/api/ (Accessed: 2025-06-05).

[Pandas, 2025] Pandas (2025). pandas.dataframe. Available at: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html (Accessed: 2025-06-11).

[Serok et al., 2022] Serok, N., Havlin, S., and Blumenfeld Lieberthal, E. (2022). Identification, cost evaluation, and prioritization of urban traffic congestions and their origin. *Scientific Reports*, 12(1):13026.

[Zheng et al., 2011] Zheng, Y., Fu, H., Xie, X., Ma, W.-Y., and Li, Q. (2011). *Geolife GPS Trajectory Dataset – User Guide*, geolife gps trajectories 1.1 edition. Available at: https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY