A Novel Spatiotemporal Framework for Efficient Traffic Prediction and Visualization

by

Sifatul Mostafi

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Faculty of Engineering and Applied Science

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

December 2021

THESIS EXAMINATION INFORMATION

Submitted by: **Sifatul Mostafi**

Master of Applied Science in **Electrical and Computer Engineering**

Thesis title: A Novel Spatiotemporal Framework for Efficient Traffic Prediction and Visualization

An oral defense of this thesis took place on December 17, 2021 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Ying Wang
Research Supervisor	Dr. Khalid Elgazzar
Examining Committee Member	Dr. Masoud Makrehchi
Thesis Examiner	Dr. Akramul Azim, Ontario Tech University

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

This thesis proposes an efficient traffic prediction framework to estimate congestion at intersections depending on neighboring road links. The framework encompasses three major components, data extraction, Bayesian Linear Regression-based traffic prediction model, and an interactive map-based traffic simulator to visualize the results. To collect traffic data, we have developed an open-source web-based data scraper tool to extract and export publicly available traffic data from the Google Maps web interface. We also developed a Bayesian Linear Regression-based traffic prediction model to estimate traffic congestion that leverages Bayesian inference to facilitate model interpretability and quantify model uncertainty. The experiments show that Bayesian linear regression modeling can be trained on small data observations to quantify model uncertainty and predict traffic congestion without sacrificing interpretability and accuracy compared to the frequentist approach. We have also developed a web-based traffic simulator to simulate linear regression-based traffic prediction models and visualize the results on interactive maps.

Keywords: traffic data extraction; bayesian linear regression; traffic simulator

AUTHOR'S DECLARATION

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Sifatul Mostafi

STATEMENT OF CONTRIBUTIONS

The contributions that accompany this thesis include an accepted paper in the 2021 IEEE International Symposium on Networks, Computers and Communications (ISNCC 2021) and an accepted paper in the 2021 IEEE World Forum on Internet of Things (WF-IOT 2021). This is described in more detail below.

1. Referred Conference Proceedings:

Sifatul Mostafi and Khalid Elgazzar, "An Open Source Tool to Extract Traffic Data from Google Maps: Limitations and Challenges" in Proc. International Symposium on Networks, Computers and Communications, Dubai, United Arab Emirates, 2021. In this paper, the authors developed an open-source web-based data scraper tool to extract and export available traffic data from the Google Maps web interface in multiple usable formats. The tool provides a user-friendly interface that enables users to visually mark the locations of interests and flexibly determine the required periods for data collections. Performance evaluation shows that the tool can retrieve traffic data from Google Maps in a linear time complexity with no significant computational overhead. Limitations and challenges to develop such tools are also investigated.

2. Referred Conference Proceedings:

Sifatul Mostafi, Taghreed Alghamdi and Khalid Elgazzar, "A Bayesian Linear Regression Approach to Predict Traffic Congestion" in Proc. IEEE World Forum on Internet of Things, New Orleans, Louisiana, USA, 2021. In this paper, the authors proposed a simple Bayesian Linear Regression approach for spatiotemporal traffic congestion prediction that leverages Bayesian inference to facilitate model interpretability and quantify model uncertainty. The model is evaluated in terms of mean absolute error (MAE) and root mean squared error (RMSE). The experiment shows that Bayesian linear regression modelling can be trained on small data observations to quantify model uncertainty and predict traffic congestion without sacrificing interpretability and accuracy in comparison with the frequentist approach.

ACKNOWLEDGEMENTS

I would like to acknowledge everyone who played a role in my academic accomplishments including my family, friends and colleagues.

First of all, my supervisor Dr. Khalid Elgazzar for providing me with the opportunity to work in the IoT Research Lab in Ontario Tech University where I have conducted this research work. I would like to thank him for providing me with proper guidelines and supporting me in every phase of this academic journey.

Last but not least, I would like to thank my committee members, each of whom has provided patient advice and guidance throughout the research process. Thank you all for your support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

ITS	Intelligent Transportation System
GPS	Global Positioning System
API	Application Programming Interface
CEM	Competitive Expectation Maximization
DOM	Document Object Model
JSON	JavaScript Object Notation
XML	Extensible Markup Language
CSV	Comma Separated Value
GMM	Gaussian Mixture Model
MMSE	Minimum Mean Square Error
GIS	Geographical Information System
MS	Milliseconds
ISP	Internet Service Provider
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
LCG	Linear Conditional Gaussian
NUTS	No U Turn Sampler
MCMC	Markov Chain Montel Carlo
HPD	Highest Posterior Density
NN	Neural Network
LSTM	Long Short Term Memory
ARIMA	Auto Regressive Integrated Moving Average
SARIMA	Seasonal Auto Regressive Integrated Moving Average

Chapter 1

Introduction

1.1 Introduction

Spatiotemporal Traffic prediction of a road can play a vital role in finding out the optimal and alternative route from a source to destination, developing an Intelligent Transportation System (ITS), planning and constructing a new road, controlling traffic lights and traffic load to reduce traffic congestion, and developing city planning and structures in complex and heterogeneous urban settings. Spatiotemporal Traffic congestion prediction is one of the most ever-growing traffic research domains where different types of methodologies, approaches, tools and techniques are used for fast and reliable traffic prediction. Generally, a traffic congestion prediction framework consists of major components like data collection and preprocessing, traffic modeling and simulation, traffic congestion prediction and visualization. In this thesis we propose a spatiotemporal traffic congestion prediction framework that consists of three major components: 1) a data collection tool, 2) a traffic prediction model, and 3) a traffic simulation and visualization software. We developed an open-source tool to extract time-series traffic data from Google Maps. Based on the collected traffic data, we developed a Bayesian linear regression based spatiotemporal traffic prediction model. Lastly, we built ”RegTraffic”, an interactive map based traffic simulator to visualize the results.

1.1.1 Google Maps as a Source of Traffic Data

With the increase of vehicles, passengers, roads and the number of trips per day, a large volume of traffic data is generated every day. These traffic data can be useful in planning optimal routes, designing traffic control and monitoring systems, implementing intelligent transportation systems, and many more. Researchers have used different data collection techniques to collect and preprocess traffic data with different technical aspects and operational characteristics. Global Positioning System (GPS) based crowd-sourcing technologies have been widely used to collect traffic data as it leverages collective traffic information from the crowd through their internet-enabled devices in real-time, which has enormous potential to be an alternative to the traditional traffic data collection techniques [1]. Studies show that traffic data collection through crowdsourcing provides reliable travel time estimates with reasonable accuracy [2].

Google Maps uses a hybrid positioning system to collect user location as passive crowdsourced information for their platforms from mobile phones running the Android operating system. Mobile phone users all around the world support this crowdsourced hybrid geolocation system of Google Maps. Google Maps also integrates region-specific local information to validate information gathered from GPS crowdsourcing. Studies reveal that data retrieved from Google Maps provides reasonably accurate and feasible traffic data [3–5]. Thus, Google Maps traffic data has been widely used both in academia and industry in application domains like traffic monitoring systems, vehicle routing, travel time prediction, smart parking systems and so on. The Google Maps Application Programming Interfaces (APIs) provide a variety of functionalities for accessing contents from Google Maps that lead to the exploration of different applications based on Google Maps APIs [6]. These APIs are not free and come with a paid subscription. Google Maps also provides traffic data with limited features through their publicly available web interface which is not in a usable format and requires manual efforts to apply in research and development. An open-source, and publicly available tool to retrieve and preprocess these publicly available traffic data from Google Maps would benefit the research community to access and utilize this data in their research and development efforts.

1.1.2 Bayesian Linear Regression

Linear regression has been studied and applied extensively in practical applications like prediction and forecasting due to its simplicity and interpretability [18]. Linear regression also has been widely applied in traffic modeling applications like traffic forecasting and traffic flow estimation [19, 20]. Linear regression is a statistical approach to explain a response variable through a linear combination of one or many explanatory variables associated with their coefficients. Frequentist and Bayesian approaches are two major approaches of linear regression [21].

Frequentist linear regression requires a high number of observations to generate statistically significant model parameters as it cannot leverage prior information and only learns from the training dataset. This approach fits well only with a linear relationship as it provides a single point estimate for the model parameters using the maximum likelihood estimation [22]. On the other hand, Bayesian linear regression is a probabilistic approach to linear regression where the model parameters are estimated based on Bayesian inference. Unlike the frequentist approach, Bayesian linear regression can leverage any non-informative priors such as a normal distribution for inference and fits smaller or missing data observations to provide statistically significant outcome [23]. Instead of estimating a single point, a posterior probability distribution can be drawn for all the model parameters based on the training dataset and the prior distribution [24]. This approach can also quantify the uncertainty in the model by assigning confidence intervals to the model parameters and predictor variables. The mean values of these posterior probability distributions are taken into account while making predictions. Linear regression based on Bayesian inference works well for traffic data that shows non-linearity and high variance. [25].

1.1.3 Traffic Simulation and Visualization

With the advancement of computer technologies and software engineering, computer based traffic simulation has become a popular approach for traffic analysis in support of the evaluation and design of Intelligent Transport Systems (ITS) [45]. Traffic simulation software with the ability to emulate the variability of spatial and temporal components in traffic phenomena makes it a practical tool for capturing and explaining complex traffic systems. The purpose of developing traffic simulation tools is to experiment with varieties of strategies in traffic modeling [46]. Traffic simulation software tools and models built on real-life traffic data are widely applied in supporting real-time traffic decision and management solutions.

Regression modeling approaches facilitate traffic modeling and prediction of the traffic congestion in a road segment based on the adjacent spatial information and temporal conditions [18]. Regression based traffic simulation helps in analyzing complex traffic structures which is a useful method for the development and planning of traffic systems and networks. Traffic infrastructure maintenance, optimal traffic regulators deployment and cost-effective practical infrastructures require analysis of complex traffic scenarios and events [47]. Hence, regression based traffic congestion estimation and computerized simulation is a suitable option for policymakers to analyze different complex traffic scenarios and take actions accordingly [48].

1.2 Motivation

The motivation of this thesis is three-fold:

1. There are open map service providers to offer traffic data through their free APIs which are not reasonably accurate to use in research [7]. Many third-party tools are also available online that provide utilities to format traffic data from the Google Maps web interface. These online tools either lack features or are proprietary. Manual retrieval and preprocessing of these publicly available traffic data from the Google Maps web interface is very time-consuming,

inefficient and error-prone. A pixel positioning based image processing technique is proposed to extract traffic layer data from the Google Maps web interface which is computationally expensive and lacks in usability [8]. Therefore, an open-source tool to acquire and preprocess publicly available traffic data from Google Maps is required.

2. In the past attempts, researchers utilized Bayesian inference based modeling for traffic flow forecasting. They mainly focused on improving forecasting accuracy by combining Bayesian inference with advanced techniques like principal component analysis for supervised feature extraction [29], competitive expectation-maximization (CEM) for model parameter estimation [30], and data transformation and redistribution [31]. Unlike the frequentist approach, these combined Bayesian inference based modeling approaches lack interpretability. Therefore, we require a traffic prediction model that can: (1) incorporate both spatial and temporal components, (2) work well with small or missing data points by leveraging prior information, (3) quantify the uncertainty in the model, (4) provide simplicity and interpretability, yet achieving competitive accuracy in comparison with frequentist approach.

3. A lot of microscopic and macroscopic traffic simulators have been developed including SUMO [54], Aimsun [57], TraffSim [58], SUMMIT [59], SifTraffic [60] and VISSIM [50]. These simulators have practical use cases in traffic analysis including traffic flow measurement, multi-agent simulation, particle based simulation and so on. However, these simulators face challenges in simulating road traffic congestion in heterogeneous road transportation networks with a small amount of real-time data [46]. A regression modeling based traffic simulator is needed to visualize traffic congestion of connected road segments using interactive geographical maps.

1.3 Problem Statement

This thesis address the problem statements as follows:

First of all, existing online tools to extract traffic data from Google Maps either lack features or are proprietary. Manual retrieval and preprocessing of these publicly available traffic data from the Google Maps web interface is very time-consuming, inefficient, error-prone, computationally expensive and lacks in usability. We require a web based open-source tool to acquire and preprocess publicly available traffic data efficiently from the Google Maps web interface. The tool must provide good usability and offer multiple features that are either available in Google Maps or can be extracted from existing features.

Secondly, unlike the frequentist approach, existing Bayesian inference based modeling approaches lack interpretability and can not quantify model uncertainty in terms of spatiotemporal components. We require a Bayesian inference based traffic prediction model that can: (1) incorporate both spatial and temporal components, (2) work well with small or missing data points by leveraging prior information, (3) quantify the uncertainty in the model, (4) provide simplicity and interpretability, yet achieving competitive accuracy in comparison with frequentist approach.

Lastly, existing traffic simulators are not specifically designed to simulate and visualize traffic congestion of connected road segments predicted by regression based traffic modeling in interactive geographical maps. Therefore, a flexible and adaptable traffic simulation software system is required to conduct regression modeling that provides functionalities to simulate and visualize traffic congestion of connected road segments using interactive maps.

1.4 Thesis Contribution

Our main contributions in this thesis are as follows:

1. We developed a web based open-source tool to acquire and preprocess publicly available traffic data from the Google Maps web interface. We designed our tool to provide good usability and offer multiple features that are either available in Google Maps or can be extracted from existing features. We evaluated the performance of the tool for further optimization and

enlisted the limitations and challenges of developing such a tool.

2. We developed a Bayesian inference based traffic prediction model that can: (1) incorporate both spatial and temporal components, (2) work well with small or missing data points by leveraging prior information, (3) quantify the uncertainty in the model, (4) provide simplicity and interpretability, yet achieving competitive accuracy in comparison with frequentist approach.

3. We built a web based traffic simulation software system that is able to conduct regression modeling and provide functionalities to simulate and visualize traffic congestion of connected road segments using interactive maps.

1.5 Thesis Organization

This thesis is organized as follows.

Chapter one introduces the concept of Google Maps as a source of traffic data extraction, Bayesian linear regression and traffic simulator for regression analysis. This chapter also indicates the gaps and challenges in the existing literature of these concepts, how we address these challenges and how we contribute to overcoming these challenges. In chapter two, we provide a comprehensive background study on Google Map based traffic data extraction tools, Bayesian linear regression based traffic prediction models and traffic simulators. In chapter three, we describe the development of an open-source tool to extract traffic data from Google Maps. We developed a mathematical model to evaluate the performance of the tool. We also point out the limitations and challenges. In chapter four, we propose a novel Bayesian linear regression based traffic prediction model. We describe the temporal feature extraction process to incorporate into the prediction model, quantify model uncertainty, discuss the results, and compare our model with other state-of-the-art frequentist approaches. In chapter five, we develop a traffic simulator software for regression based traffic simulation and visualization. We plug in our developed linear regression based traffic prediction model described in Chapter four with the traffic simulator. We simulate a real-life traffic scenario and visualize the outcome in the traffic simulator. Lastly, we provide a conclusion in chapter six where we mentioned some intuition regarding the techniques and methodology that we followed, and how our proposed methodologies overcome the existing challenges. We also highlight the limitations and challenges of our proposed system and the scope at which the contributions of the thesis can be extended in the future.

Chapter 2

Background and Related Work

2.1 Introduction

In this chapter, we describe backgrounds related to traffic data extraction from Google Maps, Bayesian linear regression based traffic modeling and different types of traffic simulators. Also, we discuss the related work, state of the art approaches, their advantages and lacking. Section 2.2 provides a comprehensive review of Google Maps as an authentic and reliable source of traffic data. In this section, we further investigate and identify the barriers to access traffic data due to the limitations and drawbacks of existing tools and APIs. Section 2.3 highlights the past attempts to predict traffic congestion using the Bayesian linear regression approach. Section 2.4 describes traffic simulation tools and provides a brief comparison. Lastly, section 2.5 summarizes the contents of the chapter.

2.2 Traffic Data Extraction from Google Maps

Google Maps is a web mapping service developed by Google which is the dominant provider of transport information and innovation. It works as a search tool to provide location based utilities [9]. Google Maps utilizes GPS data from the Google Maps application on smartphones through crowdsourcing [10, 11]. Google Maps also has access to local municipality data through contracts such as road-specific information, road types, road works, and speed limits. Google uses this data to design algorithms to calibrate and fine-tune predicted travel time constantly. Therefore, Google Maps provides accurate travel time predictions than the travel times that are systematically recorded by Uber [12]. O–D travel time matrix can be retrieved through Google Maps API that refers to an organized format of travel time between multiple origins and destinations for many spatial analysis tasks [13]. Google Maps is considered the most popular type of flexible transit service provider as it does not have to rely on geographic information. Thus, their web technologies establish a smooth scheduling system to retrieve precise data of road networks from Google servers [14]. This includes the development of modern WebGIS applications, path planning, and induction of traffic congestion. Due to an enriched programming API, Google's Web map service is one of the most widely used mapping services [15]. It also offers the functionality to customize and configure selected maps in any web browser through graphical visualization [16].

Google Maps has been proved to provide high-quality real-time traffic data through an experiment where data obtained from an intelligent transportation system are compared with Google Maps traffic data in Hong Kong. The outcomes demonstrate that the evaluated journey time was consistent in most routes throughout the entire day in both sources. The numerical differences in terms of statistical p-value measurements were also acceptable. Google Maps surpasses ITS in accessing journey time data for location based applications due to the high deployment and maintenance cost of ITS [3]. The researcher has also shown that big data retrieved from Google Maps traffic API is a feasible data source to conduct advanced research on a city road system as these big data hold high spatial and temporal aspects of the traffic situation. Although data provided by Google Maps may not reflect the whole traffic congestion, its data, with high spatial resolution, shows little deviation in finding traffic congestion patterns [4]. In another experiment, the traffic data provided by Google Maps are compared with a traffic dataset collected through sensors installed on different road segments in the city of Paris. Google Maps traffic data achieved an overall accuracy of 95.8% in fluid traffic situations [5]. Due to the credibility of Google Maps traffic data, the data has been used in a variety of traffic-related research including traffic visualization, monitoring and simulation, travel time prediction, congestion analysis, route planning, traffic light control, accident detection, traffic impact analysis, etc.

Google Maps provides Distance Matrix API to provide the functionality to users and developers to access their traffic data including travel time from source to destination [26]. Accessing these APIs requires a paid subscription. OpenStreetMap is an open-source alternative to Google Maps traffic data [27]. It is less efficient in terms of accessibility and geospatial accuracy [7]. OutSrcaper is a web based and third-party tool to extract traffic data from Google Maps which also has pricing tags [28]. As a result, researchers have been looking for ways to extract publicly available traffic data from the client-side of the Google Maps web interface without using external tools like OutScraper. Traffic layers in Google Maps are provided in the form of rendered images that show the state of the traffic congestion on different road segments using four different colors where each color represents a specific traffic congestion level [5]. Caiza et al. [8] shows how image processing techniques could be applied to extract congestion data from the Google Maps traffic layer by adopting a relationship between pixel positions of the display to geographical coordinates.

A comparison among the Google Maps Distance Matrix API, OutScraper and our proposed tool is listed in Table 2.1 in terms of some key measurements. Unlike Google Maps Distance Matrix API, the OutScraper tool does not have an API to access data as it sends requests to Google Maps server from the front end and performs data scraping once desired data is available in the document object model (DOM). A user must figure out and provide the address string separately in the Outscraper. On the other hand, our tool provides an overlay on top of the Google Maps web interface to let users select input parameters directly from Google Maps and interact with the tool in real-time which is more convenient and user-friendly. Our tool also serves the dataset in JSON, XML, CSV and XLXS format based on the user preference which is not available in the other options. The tool is not a standalone traffic data provider, but rather it only automates the process of acquiring, preprocessing and formatting publicly available data in Google maps. Any data that is not publicly accessible

Table 2.1: Comparison of Google Maps Traffic Data Access Tools

Comparison Parameters			Google Map Distance Matrix API	Outscraper	Proposed Tool
Required Input Parameters	Origin		1. Address String 2. Latitude/Longitude Coordinate 3. Place ID 4. Plus Code 5. Encoded Polylines	1. Address String	1. Real time location search 2. Latitude/Longitude Coordinate 3. Select on Google Maps
	Destination		Same format as for the origin	Same format as for the origin	Same format as for the origin
	Distance Unit		1. Metric 2. Imperial	1. Metric 2. Imperial	1. Metric
	Arrival time		arrival_time	N/A	Select "Arrive by"
	Departure Time		departure_time	Select from UI	Select "Depart at"
	Traffic Model		1. best_guess 2. pessimist 3. optimist	N/A	N/A
Output	File Format				
	Parsing Needed		Yes	Yes	Yes
	Features	Datetime	Yes	Yes	Yes
		Start Location	Yes	Yes	Yes
		End Location	Yes	Yes	Yes
		Start Latitude/Longitude	Yes	Yes	Yes
		End Latitude/Longitude	Yes	Yes	Yes
		Mid Latitude/longitude	Yes	Yes	Yes
		Maximum, Minimum and Average Duration	Yes (In terms of Best guess, Pessimist and Optimist Traffic Model)	Yes	Yes
		Distance	Yes	Yes	Yes
		Congestion Unit	Yes	Yes	Yes
		Time Zone	Yes	UTC	No
		Fare	Yes	No	No
Open Source			No	No	Yes

through the Google Maps web interface is beyond the scope of the tool.

2.3 Bayesian Linear Regression based Traffic modeling

Zhang et al. [29] propose a Bayesian networks based Gaussian Mixture Model (GMM) for traffic flow forecasting. GMM is an unsupervised clustering technique that lacks the ability to estimate from a new sample observation. GMM cannot obtain optimal model parameters through traditional maximum likelihood estimation methods. As a result, a competitive expectation-maximization (CEM) algorithm is applied to estimate the maximum likelihood by approximating the joint probability distribution. Model parameters estimation through the CEM algorithm is only compatible with mixture models and has a heavier computational burden [32]. Also, the principal component analysis is further carried out before applying the CEM algorithm to reduce the linear dimensionality and improve the forecasting accuracy. Pincipal component analysis is an unsupervised approach that only provides major components in the form of eigenvectors based on the high variance in the feature space. It does not provide any statistical measurement for feature selection. Principal components are linear combinations of standardized features that are less readable and hard to interpret [33]. The performance of the model is evaluated based on the criterion of the minimum mean square error (MMSE), which is not directly interpretable in terms of measurement unit [34]. Later on, Sun et al. [30] also carried out a similar type of experiment where they did not address or overcome the issues of interpretation and simplification.

Zhu et al. [31] integrate linear conditional Gaussian (LCG) modeling with Bayesian networks for spatiotemporal short-term traffic flow prediction using time series traffic dataset. Time series of traffic speed is redistributed in different arbitrary ranges as a categorical variable and combined with continuous traffic volume in the Bayesian network for better prediction accuracy. Some of these approaches consider traffic flow in terms of traffic count, which is the number of vehicles on some road link, and requires further adjustment to plug into a regression model [35]. This approach only provides a probability distribution for continuous variables and left out discrete variables in the Bayesian network. Experiments on the synthesized dataset generated through a microscopic simulation indicate a biased outcome which results in less interpretability of the model parameters.

Unlike feature density estimation using an unsupervised approach like Gaussian Mixture modeling, we propose a supervised Bayesian network based linear regression model that can leverage prior belief to estimate model parameters. The model leverages the Markov Chain Monte Carlo algorithm and samples a posterior probability distribution for each of the model parameters associated with both continuous and discrete independent variables to quantify model uncertainty. The mean of a posterior probability distribution is considered as the best estimation of the corresponding model parameter. The best estimation of each model parameter is multiplied with corresponding input variables to predict the congestion index. The model performance is evaluated in terms of the root mean squared error which is directly interpretable in terms of measurement units. Instead of selecting temporal features through the principal component analysis, we leverage exploratory data analysis for feature extraction and selection. We represent traffic congestion of a road in terms of an average traffic speed for simplicity and better interpretability of the model.

2.4 Traffic Simulator

Traffic simulators are commonly divided into two categories, microscopic traffic simulators [50, 51, 54] and macroscopic traffic simulators [52].

FreeSim traffic simulator is designed to conduct real-time freeway traffic simulation [53]. SUMO (Simulation of Urban Mobility) is a microscopic traffic simulator that is developed to process complex and large road networks [54]. SUMO is widely used in many applications including traffic flow modeling [55] and color mapping Google Maps routes [56]. Aimsun is a traffic simulator software for modeling smart mobility [57]. Traffsim software simulator is widely used for modeling isolated traffic control strategies [58]. SUMMIT traffic simulator provides functionalities to simulate urban driving in large traffic scenarios with massive and mixed traffic [59]. SimTraffic is a practical software tool to conduct simulations of practical traffic applications [60]. VISSIM is a microscopic traffic simulator for behavior based multi-purpose traffic flow simulation [50].

Wang et al. [61] explored different methods of correcting traffic simulation modeling based on linear regression. Golovnin et al. [62] took a web-oriented approach to simulate road traffic, especially in urban conditions. Mizuta et al. [63] evaluated the traffic flow near intersections of a metropolitan city to understand how agent based traffic simulators work to approximate vehicle behaviors.

These state of the art simulators and simulation techniques do not provide functionalities to simulate and visualize regression based modeling in an interactive map. Therefore, we need a traffic simulator software that can be customized to simulate and visualize regression based traffic models like Bayesian linear regression in a more interactive and dynamic way for better interpretation of the model results and decision making support. A comparison among the existing traffic simulators is listed in Table 2.2 in terms of some key measurements.

Table 2.2: Comparison of Traffic Simulators

	Spatiotemporal Traffic Modeling	Regression Modeling	Geographical Map Visualization	Web Interface
RegTraffic	Yes	Yes	Yes	Yes
FreeSim [53]	Yes	No	No	No
SUMO [54]	Yes	No	Yes	No
Aimsun [57]	Yes	No	Yes	No
TraffSim [58]	Yes	No	No	No
SUMMIT [59]	Yes	No	Yes	No
SimTraffic [60]	Yes	No	Yes	No
VISSIM [50]	Yes	No	Yes	No

2.5 Summary

In this chapter, we introduced the background of Google Maps based traffic data extraction, Bayesian Linear Regression based traffic prediction modeling and different types of Traffic Simulators. We surveyed the literature and described the research work related to the focus of this thesis. We described the advantages and lacking the state of the art approaches for Google Map based traffic data collection, Bayesian linear regression modeling and different types of traffic simulators. We also provided a comprehensive comparison of the existing approaches compared to the proposed solution.

Chapter 3

An Open Source Tool to Extract Traffic Data from Google Maps

3.1 Introduction

In this chapter, we describe the development of an open-source web based data scraper tool to extract and export available traffic data from the Google Maps web interface in multiple usable formats. The tool provides a user-friendly interface that enables users to visually mark the locations of interests and flexibly determine the required periods for data collections. Performance evaluation shows that the tool can retrieve traffic data from Google Maps in a linear time complexity with no significant computational overhead. Limitations and challenges to develop such tools are also investigated.

3.2 Proposed Methodology

Our approach is to develop a data extraction technique to search and retrieve specific data in the document of the object model of the Google Maps web interface. Once the traffic data is retrieved, we preprocess these data and store them in multiple user formats. We also develop a web interface as another layer on top of the Google Maps web interface for the users to select a specific time, date and location for traffic data extraction. We also provide a panel in the web interface for the user to select a set of features to include in the dataset.

3.2.1 Data Extraction

Data interfacing, acquisition, and preprocessing are the major steps of any data collection technique [17]. In our proposed approach, data interfacing involves the selection of input parameters and features. Input parameters are specific time, date and location to extract traffic data whereas the features are the specific column in the dataset. Data acquisition is processed by sending an HTTP request to Google Server and data retrieval from the document object model. The data preprocessing part consists of data validation, feature extraction, data formation and data storage. Each one of these operations is dependent on its predecessors and cannot proceed until the previous operation is completed as shown in Figure 3.1.

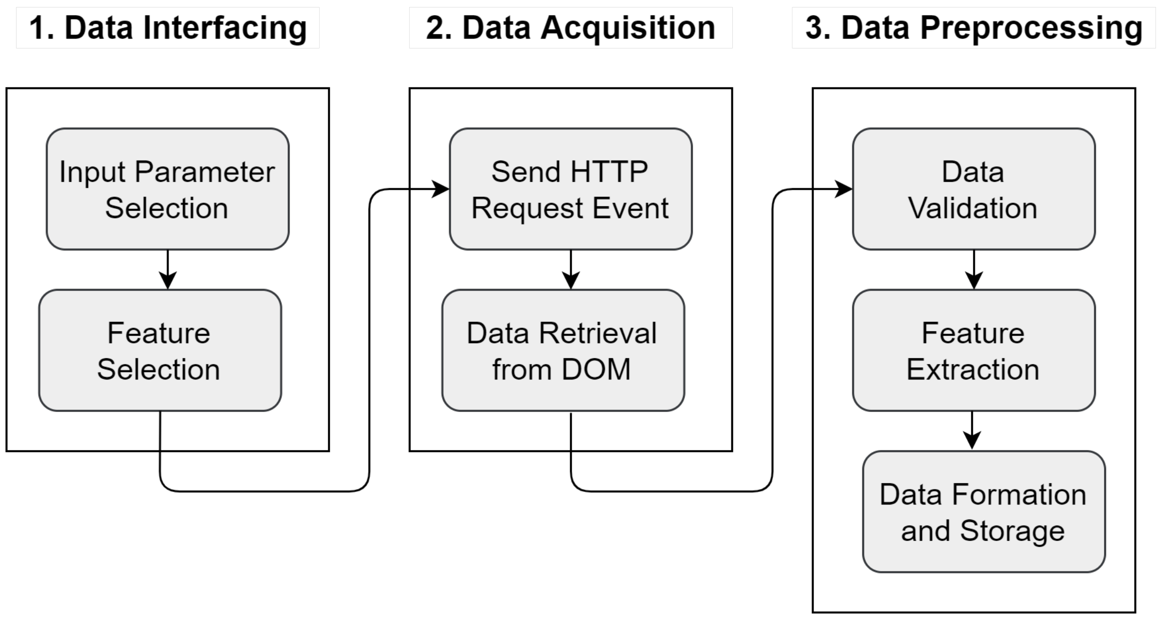


Figure 3.1: Processing Pipeline of the Tool to Extract Traffic Data from the Google Maps

An asynchronous callback function operates at the core of the system that integrates the data acquisition and preprocessing part. To extract observation of a time step, the function generates a DOM event to send a data request to Google Server and waits until returned by the server and becomes available in the DOM. There is an unequal time lag between the moment the click event is fired and the data being available in DOM. To address this issue, the callback function operates after a certain interval specified as a hyperparameter in the system through performance evaluation. In the data validation phase, the system checks for data duplication for multiple time steps. After data validation, the system extracts new features like average traffic duration and traffic congestion index. After feature extraction, the system formats data in multiple formats and saves it.

Algorithm 1 presents the pseudo-code of the callback function. The function takes three input parameters n, r, t where n is the number of roads, r is the road segment and t is the starting time. The output dataset file is represented by c. Here, 1 ms is assigned to the variable $callbackInterval$ as a hyperparameter. In section 3.3, we demonstrate further analysis to find the optimal value of the callback interval.

3.2.2 User Interface

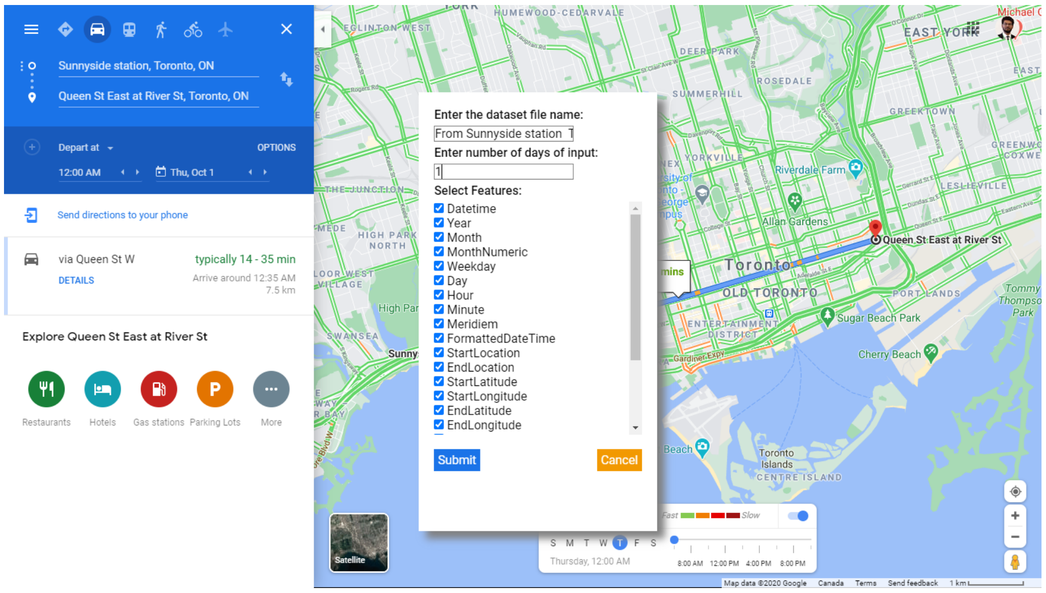
The tool provides an interactive interface for users to provide input parameters and interact with the system.

1. The tool requires a user to select the starting point, endpoint of a road segment, departure time and departure date as input parameters.

2. The tool provides a user interface to specify the name of the dataset file and the number of days and a list of checkboxes where each checkbox represents a feature of the dataset 3.2.

3. The tool validates input parameters and provides an error message using an alert box.

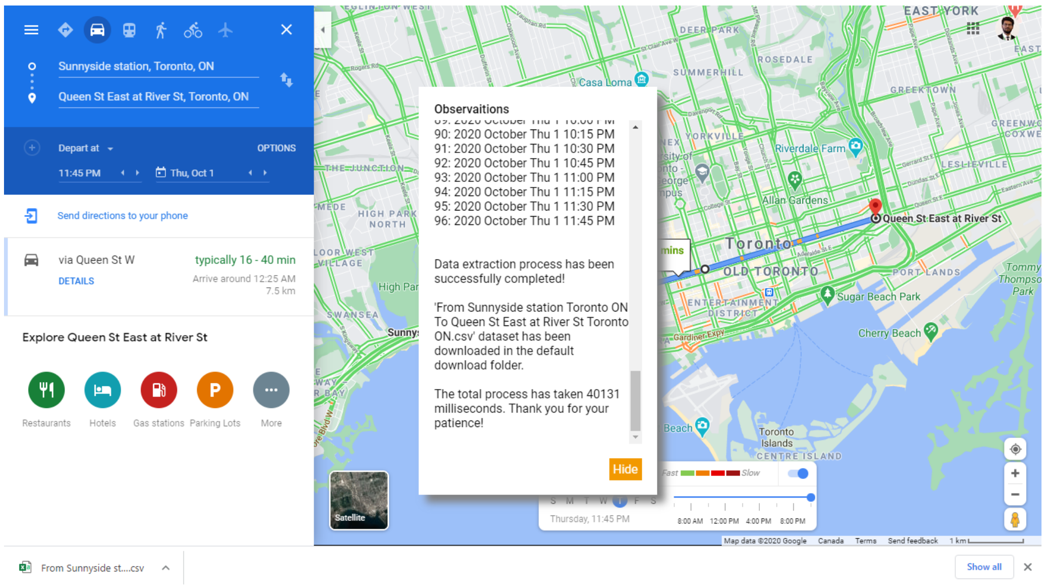
4. The tool shows the ongoing process and provides a success message once the data extraction process is completed 3.3.



Enter the dataset file name:
From Sunnyside station T
Enter number of days of input:
1
Select Features:
- [x] Datetime
- [x] Year
- [x] Month
- [x] MonthNumeric
- [x] Weekday
- [x] Day
- [x] Hour
- [x] Minute
- [x] Meridiem
- [x] FormattedDateTime
- [x] StartLocation
- [x] EndLocation
- [x] StartLatitude
- [x] StartLongitude
- [x] EndLatitude
- [x] EndLongitude

Submit Cancel

Figure 3.2: User Interface: Asking for Input Parameters for Data Extraction



Observations

```
90: 2020 October Thu 1 10:15 PM
91: 2020 October Thu 1 10:30 PM
92: 2020 October Thu 1 10:45 PM
93: 2020 October Thu 1 11:00 PM
94: 2020 October Thu 1 11:15 PM
95: 2020 October Thu 1 11:30 PM
96: 2020 October Thu 1 11:45 PM
```

Data extraction process has been successfully completed!

'From Sunnyside station Toronto ON To Queen St East at River St Toronto ON.csv' dataset has been downloaded in the default download folder.

The total process has taken 40131 milliseconds. Thank you for your patience!

Hide

Figure 3.3: User Interface: Success Message Once Data Extraction is Completed

Algorithm 1: Traffic Data Extraction of a Road Segment

```
1  function extract $(n, r, t)$ Input   : Number of days, $n$
                 Road Segment, $r$
                 Time, $t$
   Output: Dataset File, $c$
2  $obs \leftarrow n$ * 4 * 24
3  $previousMin \leftarrow NULL$
4  $lapStartTime \leftarrow CurrentTime()$
5  $lapEndTime \leftarrow NULL$
6  $lapTime \leftarrow NULL$
7  $totalElapsedTime \leftarrow NULL$
8  $totalIteration \leftarrow 0$
9  $obsCount \leftarrow 0$
10 $callbackInterval \leftarrow 1$
11 while $Interval(callbackInterval)$ is $True$ do
12     $totalIteration++$
13     if ContentLoads() then
14         $currentMin \leftarrow GetMinute()$
15         if $previousMin$ != $currentMin$ then
16             $previousMin \leftarrow currentMin$
17             if $obsCount < obs$ then
18                 $Download(c)$
19                 $ClearInterval()$
20             end
21             $c[obsCount] \leftarrow Extract(r, s)$
22             $c[obsCount].append(NewLine)$
23             $lapEndTime \leftarrow currentTime()$
24             $lapTime \leftarrow lapEndTime$ - $lapStartTime$
25             $totalElapsedTime \leftarrow totalElapsedTime$ + $lapTime$
26             $lapStartTime \leftarrow lapEndTime$
27             $obsCount++$
28             $t \leftarrow t$ + 15
29         end
30     end
31 end
32 return $c$
```

3.2.3 Data Description

The tool provides 12 data features with different data types, units and ranges as shown in Table 3.1. Among these features, $Datetime$, $AvgDuration$, and $CongestionIndex$ features are generated through feature extraction. $Datetime$ feature consists of all the temporal features like Day, $Month$, $Year$, $Hour$, $Minute$, and $Meridie$. The $AvgDuration$ is the average of $MinDuration$ and $MaxDuration$. The $CongestionIndex$ feature represents the average speed of a road in terms of kilometers per hour and is extracted from $AvgDuration$ and $Distance$ features.

Figure 3.4 shows the time series data of a road segment starting from Sunnyside station, Toronto, ON to Queen St E and River St, Toronto, ON. The data was taken for two consecutive days, from 12:00 am on the first of October to 11:45 pm on the second of October. The figure shows how the Congestion Index, which is the average speed in Kilometer/hour, changes over two days. A high average speed indicates low congestion and a low average speed indicates high congestion. The difference between the two-time steps is 15 minutes as provided by Google Maps.

3.3 Performance Evaluation

The performance of the tool mostly depends on how fast the requested data is loaded in the DOM element of the Google Maps web interface once the request is sent to the server. Regardless of how fast the data extraction process is, the tool must wait until the requested data is retrieved and available on the Google Maps web interface. Nevertheless, the system can be further optimized so that it can gather data as soon as it is available without any significant computational overhead. Finding the right value for the $callbackInterval$ hyperparameter plays a crucial role in optimizing the performance as it modifies two important factors of the tool: the number of iterations a callback function needs to extract the data and the delay in the data extraction process once the data is available.

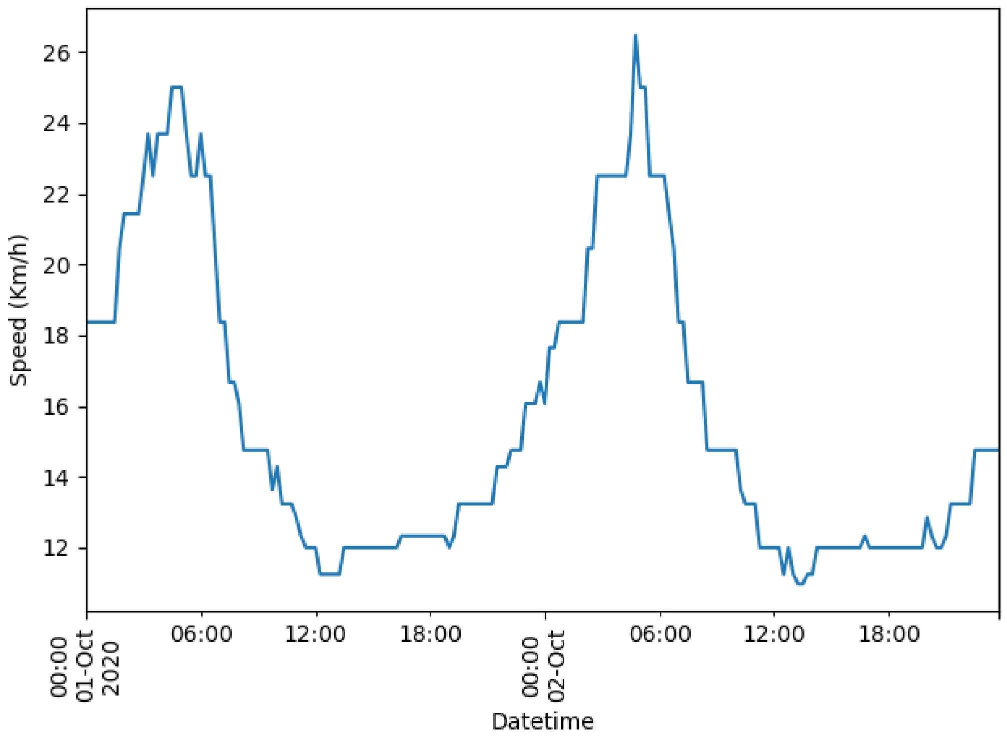


Figure 3.4: Time Series Data Extracted using the Proposed Tool from the Google Maps

Thorough sensitivity analysis of the data extraction process with respect to different hyperparameters shows that tuning the hyperparameters affect the performance of the system.

3.3.1 Mathematical modeling

The number of iterations that the callback function may go through to data observation of a time step depends on the length of the callback interval $callbackInterval$ and $lapTime$. $lapTime$ is the time duration to load the requested data once the click event is triggered as shown in Algorithm 1. There is no way to know the exact duration of $lapTime$ for a specific observation in advance as it depends on many external factors like internet speed, network congestion, web browser version as well as the specification of the machine on which the callback is executed. All these external factors are beyond the scope of the development of the tool. The $lapTime$ values are ordinal as could be

Table 3.1: Retrieved Data Description Using Data Extraction Tool from Google Maps

SL	Feature	Description	Unit	Range	Type	preprocessing	Feature-Extraction
1	Datetime	The DateTime index of time series		Any	DateTime	Yes	Yes
2	StartLocation	The location of the starting point of a road segment with Postal Code	N/A	N/A	Text	No	No
3	EndLocation	The location of the ending point of a road segment with Postal Code	N/A	N/A	Text	No	No
4	StartLatitude	The latitude of the starting point of a road segment	DD	(-)90 - (+)90	Numerical	No	No
5	EndLatitude	The latitude of the ending point of a road segment	DD	(-)90 - (+)90	Numerical	No	No
6	StartLongitude	The longitude of the starting point of a road segment	DD	(-)180 - (+)180	Numerical	No	No
7	EndLongitude	The longitude of the ending point of a road segment	DD	(-)180 - (+)180	Numerical	No	No
8	MinDuration	The minimum duration it may take to cross a road segment	Km / h	0 - ∞	Numerical	Yes	No
9	MaxDuration	The maximum duration it may take to cross a road segment	Km / h	0 - ∞	Numerical	Yes	No
10	AvgDuration	The average of minimum and maximum duration to cross a road segment	Km / h	0 - ∞	Numerical	Yes	Yes
11	Distance	The distance of the road segment in terms of kilometres	Km	0 - ∞	Numerical	Yes	No
12	Congestion Index	The traffic speed on a road segment	Km / h	0 - ∞	Numerical	Yes	Yes

different for two different sets of observations. We can set the value of the only hyperparameter $callbackInterval$ in such a way so that the tool can generate an optimal performance for any value of $lapTime$.

The $callbackInterval$ can be equal, smaller or greater than the $lapTime$. In the following subsections, we discuss the possible outcome in terms of $iteration$ and $overhead$ for different values of $callbackIntervals$ in comparison with $lapTime$.

$callbackInterval = lapTime$

This is the ideal scenario where the $callbackInterval$ is equal to the $lapTime$. The callback function can collect a data observation in each iteration as the data loading time is the same as the interval of the callback function (3.1). There is no $overhead$ as the callback function takes only one iteration to collect a data observation (3.2). It is not possible to predetermine the exact value of $lapTime$ and set the value of $callbackInterval$ accordingly to avoid any overhead. Because $lapTime$ is a random variable that varies between observations and the $callbackInterval$ remains constant as it is set as a hyperparameter at the beginning of the iteration.

$callbackInterval < lapTime$

If the value of $callbackInterval$ becomes less than $lapTime$ then the callback function iterates more than once. The callback function continues iteration after every $callbackInterval$ until it reaches the point when the requested data is loaded. Although $lapTime$ is greater than $callbakInterval$, it is not necessarily a multiple of $callbackInterval$. The number of $iteration$ of the callback function is determined by ceiling the fraction of $lapTime$ and $callbackInterval$ as shown in eq. (3.1). Eq. (3.2) shows the general formula to calculate the overhead to collect a data observation.

If the $callbackInterval$ value is smaller than the value of $lapTime$, the $overhead$ is lower and vice versa as per eq. (3.4). Since the $lapTime$ values show high variance in different observations, we need to keep $callbackInterval$ as small as possible so that the callback function can collect a data observation as soon as it is loaded. Smaller values of $callbackInterval$ results in an increasing number of iterations (3.3).

$callbackInterval > lapTime$

If the value of $callbackInterval$ is greater than $lapTime$, the callback function iterates only once to collect a data observation (3.1). A data observation is already loaded by the time the callback function completes an iteration and generates $overhead$ as shown in eq. (3.2). The higher the difference between $callbackInterval$ and $lapTime$, the bigger the $overhead$ and vice versa, see eq. (3.4).

Eq. (3.5) can be derived from eq. (3.3) and (3.4) which shows that $iteration$ is inversely proportional to $overhead$. Therefore, an increment in $iteration$ of a callback function to collect a data observation always leads to a decrease in timing $overhead$ and vice versa. This conclusion emphasizes the trade-off between time complexity (here in terms of time $overhead$) and computational complexity (here in terms of the number of $iteration$ of the callback function). In the next subsections, we illustrate the experimental setup and results to support these hypotheses.

$$iteration = \begin{cases} 1, & \text{if } callbackInterval \geq lapTime \\ Ceiling\left(\dfrac{lapTime}{callbackInterval}\right), & \\ & \text{if } callbackInterval < lapTime \end{cases} \tag{3.1}$$

$$overhead = \begin{cases} 0, & \text{if } callbackInterval = lapTime \\ (callbackInterval * iteration) - lapTime, & \\ & \text{if } callbackInterval < lapTime \\ callbackInterval - lapTime, & \\ & \text{if } callbackInterval > lapTime \end{cases} \tag{3.2}$$

$$iteration \propto \begin{cases} lapTime - callbackInterval, & \\ & \text{if } callbackInterval < lapTime \\ \dfrac{1}{callbackInterval - lapTime}, & \\ & \text{if } callbackInterval > lapTime \end{cases} \tag{3.3}$$

$$overhead \propto \begin{cases} \dfrac{1}{lapTime - callbackInterval}, \\ \qquad\qquad \text{if } callbackInterval < lapTime \\ callbackInterval - lapTime, \\ \qquad\qquad \text{if } callbackInterval > lapTime \end{cases} \tag{3.4}$$

$$overhead \propto \frac{1}{iteration} \tag{3.5}$$

3.3.2 Experimental Setup

We run our experiment in a machine with Intel(R) Core(TM) i3-4005U CPU that has 1.70 GHz and 8 GB DDR3 RAM. The machine runs a 64-Bit Windows 10 operating system with the latest version of google chrome installed at the time of the experiment. The road segment starts from the junction of Conlin Road East and Simcoe Street North (Located in Oshawa, Ontario, Canada, L1H 7K4) towards 1352-1340 Durham Regional Rd 2, Oshawa, ON. We collect time series data of that road segment for two consecutive days from 12:00 am on the 1st of October to 11:45 pm on the 2nd of October 2020. The final dataset contains a total of 192 observations for two consecutive days as each day contains 96 observations. We conduct the experiments 9 times with 9 different callback intervals between 1 ms and 1000 ms. Rather than comparing $iteration$ and $overhead$ for each of the 192 observations and different $callbackInterval$ values, the cumulative number of iterations and elapsed time (in ms) is calculated and then compared between these for different $callbackInterval$.

Our experiments and results support the desired outcome as hypothesized in the previous subsection. The experiment shows that the tool runs in a linear time with respect to the number of data observations and can be optimized by changing the value of $callbackInterval$ to the optimal point to avoid any computational overhead.

3.3.3 Results

Iteration with Different Callback Intervals

Figure 3.5 shows the number of iterations cumulatively added with observations for different values of $callbackInterval$. The lower the value of $callbackInterval$, the steeper the line, as the callback function went through more iterations to collect a data observation. On the other hand, the higher the value of $callbackInterval$, the flatter the line, as the callback function went through fewer iterations to collect a data observation.

There are a few important aspects to notice in the graph. Changes of $callbackInterval$ from 1 ms to 10 ms result in a higher change in slope in between the corresponding lines than changes of $callbackInterval$ from 500 ms to 1000 ms. The change of slope for lines corresponding to different $callbackInterval$ is not the same across different values of $callbackInterval$. Lines corresponding to lower $callbackInterval$ are comparatively more scattered than the lines corresponding to higher $callbackInterval$. The higher the value of $callbackInterval$, the lower the change in the cumulative number of iterations in between consecutive lines and the lower the value of $callbackInterval$, the higher the change in between consecutive lines. The number of total iterations decreases by a significant amount after a certain threshold even if the value of $callbackInterval$ keeps increasing.

Sometimes a line may have a sudden rise in terms of the number of iterations within a very short number of observations. For instance, the blue line corresponding to the $callbackInterval$ of 1 ms has one sudden spike around observation 135. While collecting data of the 135th observation, the callback function was stuck either because data was not loading or there was a duplication in the data. Therefore, the callback function had to iterate more times as the $callbackInterval$ for this line is only 1 ms, which results in a sudden rise in the corresponding line. Lines corresponding to comparatively lower

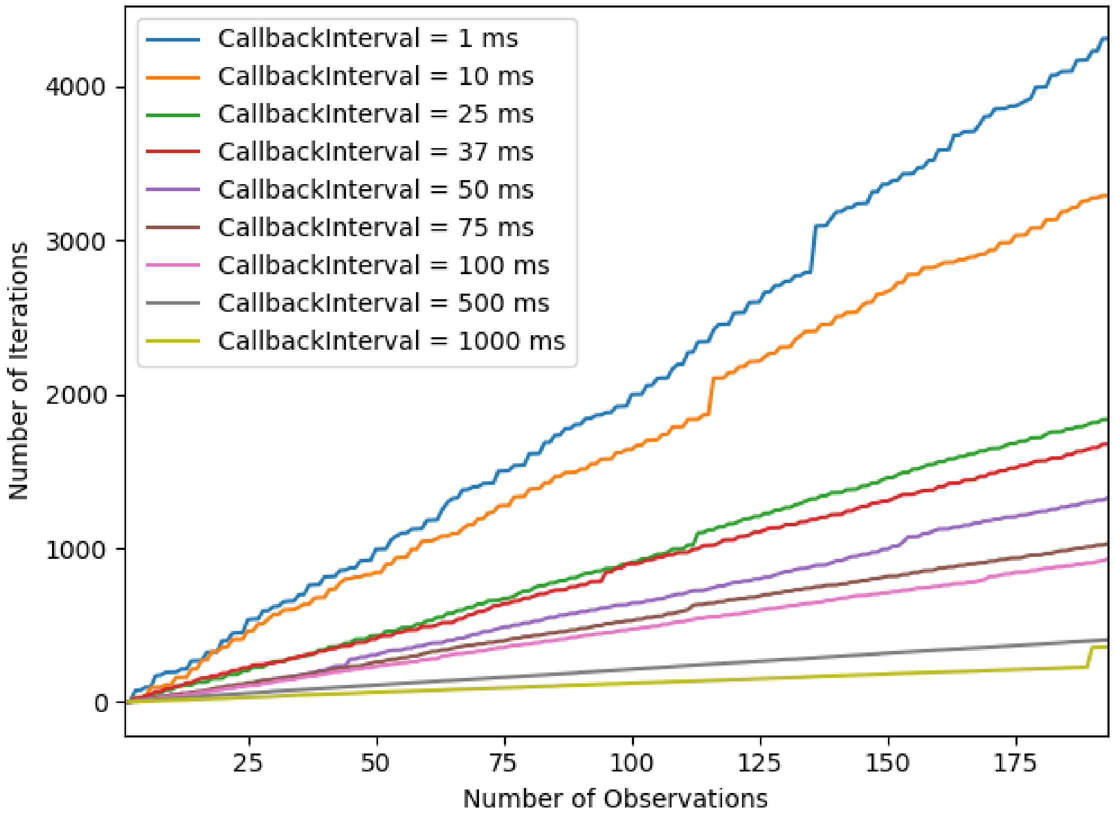


Figure 3.5: Number of Iteration with Different Values of $callbackInterval$

$callbackInterval$ also expose similar characteristics. The sudden rise is not as high as compared to the lines corresponding $callbackInterval$ of smaller values. It is not possible to determine how many sudden rises a line may have in advance. There could be none, one or more sudden rises in any line representing any value of $callbackInterval$.

Time Complexity with Different Values of $callbackInterval$

The same experiment is repeated to measure the total elapsed time with a different number of observations. The $overhead$ time of the observations is cumulatively added and plotted the total elapsed time with different observations as shown in Figure 3.6. The higher the value of $callbackInterval$, the steeper the line, as more $overhead$ timing is added on top of $callbackInterval$ to collect a data observation. On the other hand, the lower the value of $callbackInterval$, the flatter the line. This behavior shows that the tool can run in a linear time complexity with respect to the number of observations.

Figure 3.6 also shows similar kinds of characteristics as Figure 3.5. Changes of $callbackInterval$ from 1000 ms to 500 ms result in a lower change in slope in between the corresponding lines than changes of $callbackInterval$ from 10 ms to 1 ms. The slope changes for the lines corresponding to different $callbackInterval$ are not the same across different values of $callbackInterval$. The lines corresponding to higher $callbackInterval$ are comparatively more scattered than the lines corresponding to lower $callbackInterval$. The higher the value of $callbackInterval$, the lower the change is in the cumulative elapsed time in between consecutive lines and vice versa. Total elapsed time decreases by a significant amount after a certain threshold even if we keep decreasing the value of $callbackInterval$.

Trade off between the number of Iteration and Elapsed Time

The objective of our analysis is to determine an equilibrium point where the value of the hyperparameter $callbackInterval$ results in the desired output which is less computational and less time. An increase in $callbackInterval$ decreases the total number of iterations of the callback function, but increases the total elapsed time, thus delaying the overall data collection process. Similarly, a decrease in $callbackInterval$ also decreases the total elapsed time and increases the total number of iterations of the callback function. Therefore, there is a trade-off between the cumulative number of iterations and the total elapsed time.

In Figure 3.7, we plot the total elapsed time vs. the total number of iterations for different values of $callbackInterval$. The data point representing $callbackInterval$ of 1 ms

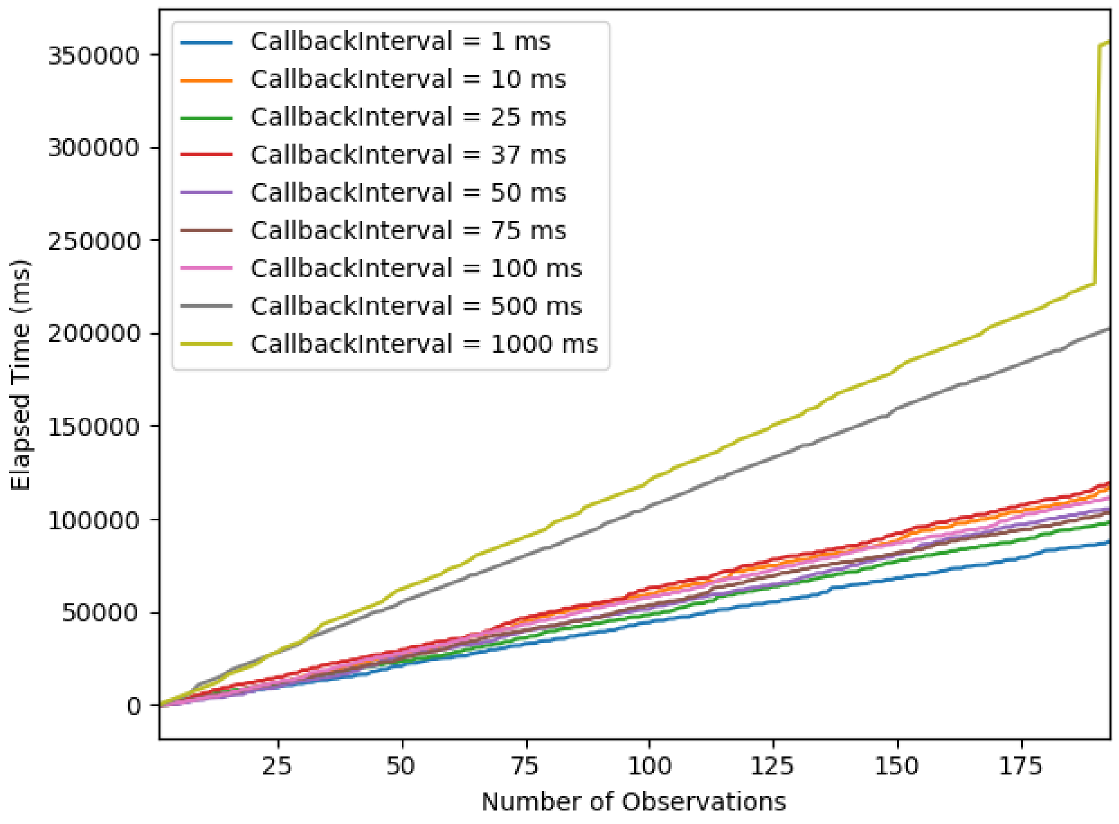


Figure 3.6: Time Complexity for Different Callback Intervals (Milliseconds)

provides the highest number of cumulative iterations with the lowest elapsed time. On the other hand, the data point representing $callbackInterval$ of 1000 ms provides the lowest number of cumulative iterations with the highest elapsed time. The difference between the $callbackInterval$ of 1000 ms and 100 ms is comparatively lower with respect to the total number of cumulative iterations than it is with respect to the total elapsed time. Moreover, the difference between $callbackInterval$ of 100 ms and 1 ms is comparatively higher for the total number of cumulative iterations than it is for the total elapsed time. As a result, we can consider $callbackInterval$ of 100 ms as an optimized equilibrium point for this dataset.

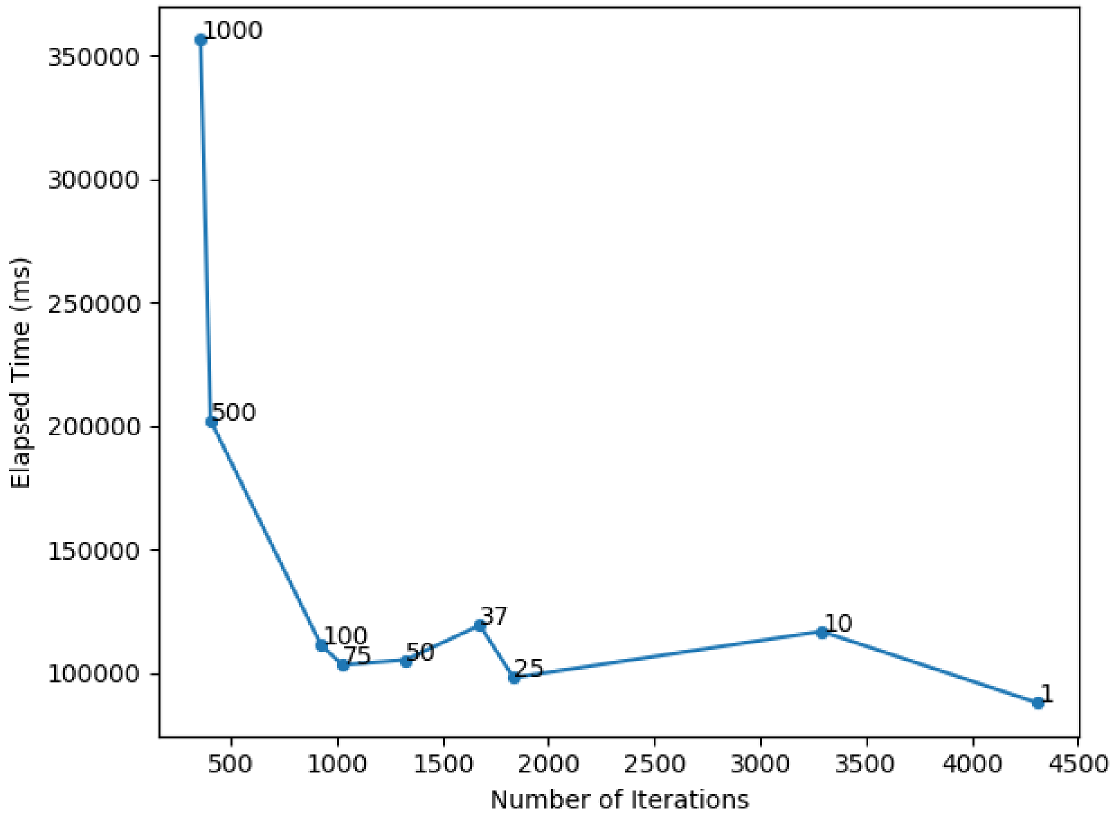


Figure 3.7: Trade off Between Number of Iteration and Time Complexity (Milliseconds)

3.4 Limitations and Challenges

The tool does not have any access to the Google Maps Distance Matrix API. It solely depends on how fast and consistent the requested data of a time step is loaded on the Google Maps web interface accessed through a web browser. Although the working procedure of the tool is technically correct, the feasibility of the tool needs to be addressed. Excessive use of the tool results in a high volume of frequent HTTP requests from the same IP address that may raise security concerns. Frequent click events generated on the Google Maps web interface may provoke some ISP to block the process assuming that it could be a malicious activity. Limited usage of the tool from the same IP address might help but this is not a feasible solution as it largely depends on the internet service provider and how Google tackles a high volume of data requests via the Google Maps web interface.

Comparing the performance of the proposed tool with existing APIs and Tools would require further investigation for different types of queries in order to show the real value of the proposed tool. It is very difficult to develop insightful quality, or performance metrics of the proposed tool as the performance of the tool is largely based on Google's data and infrastructure quality.

The premise of this work is not to get access to Google Maps data for free through the Google Maps web interface. Our intention is not to encourage the use of this tool to circumvent paying for data which would be an ethically and possibly legally questionable stance. Using these techniques to circumvent a pay-wall is almost certainly a violation of Google's terms of service. Traffic data with a limited feature is already publicly available on the Google Maps website which is open to all users for free. We can get access to this information without violating Google's terms of service. The tool would help a user to facilitate and automate the process of accessing this publicly available traffic data through web scraping and reverse-engineering which is certainly legal.

3.5 Summary

Google Maps uses GPS location based crowdsourcing to collect real-time traffic data with high precision. Access to this traffic data requires paid subscriptions to use The Google Maps Distance Matrix API. Other map services and third-party online tools are either less accurate in extracting time-specific traffic information, have limited features or are proprietary. In this chapter, we describe the development and analysis of a lightweight tool to extract publicly available traffic data from the Google Maps web interface in an automated process by leveraging web scraping. We developed a mathematical model to evaluate the performance of the tool. After conducting a performance evaluation of our tool and found that the tool is lightweight, highly accurate and efficient. Although the tool is dependent on the Document Object Model of Google Maps web interface, it is browser-independent and can be used on any latest version of a web browser. We further mentioned some key limitations and challenges based on the performance analysis.

Chapter 4

A Bayesian Linear Regression Approach to Predict Traffic Congestion

4.1 Introduction

In this chapter, a simple Bayesian linear regression approach has been described for spatiotemporal traffic congestion prediction. The model leverages Bayesian inference to facilitate interpretability and quantify uncertainty. The model is evaluated in terms of mean absolute error (MAE) and root mean squared error (RMSE). The experiment shows that Bayesian linear regression modeling can be trained on small data observations to quantify model uncertainty and predict traffic congestion without sacrificing interpretability and accuracy compared to the frequentist approach.

The remainder of this chapter is organized as follows. 4.2 demonstrates exploratory data analysis for temporal feature extraction. Section 4.3 and 4.4 shows the methodology and experimental setup, respectively. The results are discussed in Section 4.5. Lastly, Section 4.6 concludes the final remarks and provides future directions.

4.2 Exploratory Data Analysis

A road segment is determined as a specific unidirectional segment in a road that has a unique entity in a Geographical Information System (GIS) [36]. A road segment can be connected with other road segments at the extremities of its origin and end. We conducted our experiment on four connected road segments in Oshawa, Ontario, Canada. Among these 4 connected road segments, the origin of road segment 1 is connected with the ending point of road segments 2, 3 and 4. Together, these road segments form a connected road network. We represent the traffic congestion level of these 4 road segments as $Road1$, $Road2$, $Road3$ and $Road4$, respectively as shown in Figure 4.1. We would like to see how $Road2$, $Road3$ and $Road4$ collectively affect $Road1$ during a specific time of a day.

We collected the average traffic speed of each road segment every 15 minutes for an entire week from 12:00 am on March 01, 2020, to 11:45 pm on March 07, 2020. As a result, there are a total of 672 observations over 7 days of time-series data for each road segment. We extracted this traffic data from Google Maps using our scraper tool [37]. Figure 4.2 shows the time series of the average traffic speed of all the four road segments for the first two days. The y axis represents the average traffic speed in km/h, which is considered the traffic congestion index in our analysis. We can see that the time series has a cycle as the average traffic speed shows regular and predictable changes that recur every day within a certain time interval. The higher average speed indicates low traffic congestion and the low average speed indicates high traffic congestion.

Linear regression modelling estimates and explains a response variable in terms of a linear combination of explanatory variables associated with their corresponding model parameters. To build a spatiotemporal linear regression model, spatial components have to be

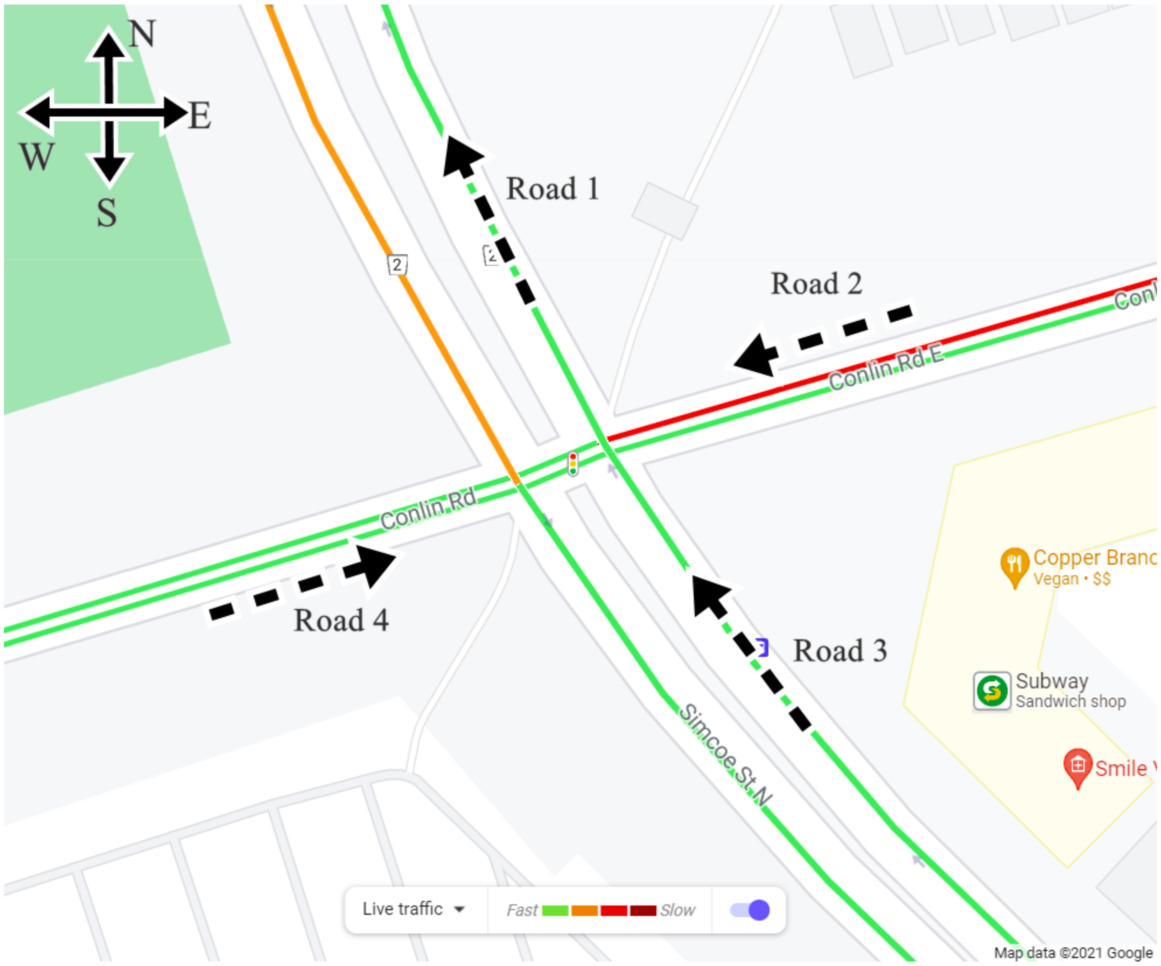


Figure 4.1: The Junction of Simcoe and Conlin in Oshawa, ON.

linearly combined with the temporal components as an explanatory variable in the regression model to explain the response variable. We consider the traffic congestion index of a road link as a spatial feature of that road link as the congestion index is calculated based on the time series traffic speed information of that specific road link location. The traffic congestion index of a road link is measured by the average traffic in that specific road link in terms of Km/h. In the Bayesian linear regression model, we estimate the traffic congestion index of a road link in terms of the traffic congestion index of some other connected road links. The traffic congestion index of these road links are considered a spatial feature

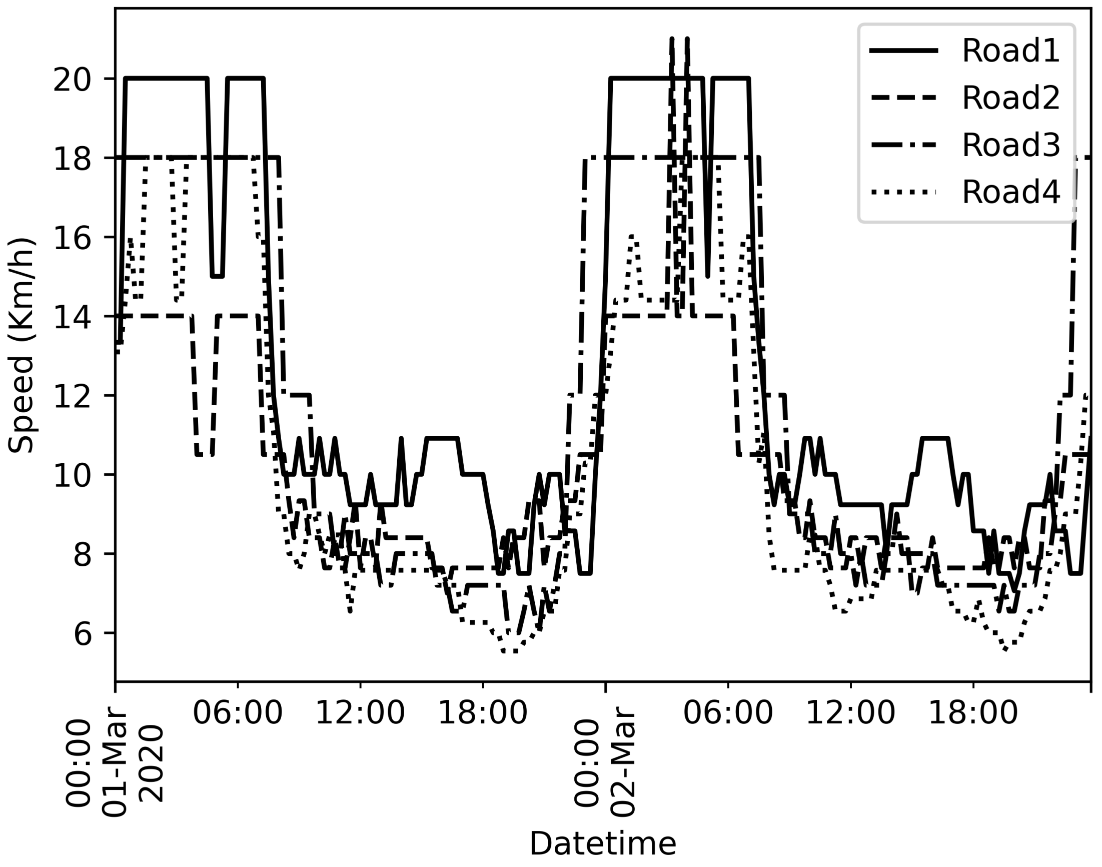


Figure 4.2: Time series of congestion throughout the week.

and are linearly combined with their associated model parameter in the Bayesian linear regression equation. The time components from time series indexes are extracted as binary categorical features and incorporated into the model as temporal components.

Adopting this approach assists the model in avoiding one-hot encoding as well as data scaling. One hot encoding generates a new feature vector for every category of a categorical variable, which might lead to a dummy variable trap and also increase data dimensionality. The data scaling process involves normalization or standardization of all the explanatory features which may result in information loss including outliers. For that, exploratory data

Table 4.1: Correlation of $Road1$ with explanatory features

Feature	**Correlation with $Road1$**
AM	0.741888
Peakhour	-0.893341
Road2	0.821926
Road3	0.825143
Road4	0.920026

analysis is conducted by plotting the density distribution of multiple time components. Only the time components with a substantial impact on the response variable are chosen. The seasonality in the time series is taken into account to extract new time components and verify their explanatory ability.

For each road segment, the hourly mean average speed is plotted as shown in Figure 4.3. The mean values show very little variance compared to each other as they seem to move together throughout the day. The average of the different means of all road segments is plotted in Figure 4.4. The horizontal line at a speed of 11.75 km/h divides the plot evenly and intersects with the total average speed at two points, one at daytime 8:00 and the other one is at 23:00. From this exploratory data analysis, a new categorical feature called $Peakhour$ is extracted that indicates a certain time interval during a day where the average traffic speed remains below 11.75 km/h. From 9:00 am to 12:00 pm, the value of $Peakhour$ would be 1, otherwise 0. Another time component is considered in the analysis as a categorical variable which is AM. The value of AM would be 1 when the meridiem is AM and 0 when it is PM.

Figure 4.5 and 4.6 shows the density distribution of $Road1$ in terms of two categorical features AM and $Peakhour$, respectively. Both of the graphs provide two separable density distributions for these two categorical features. It indicates that these features can

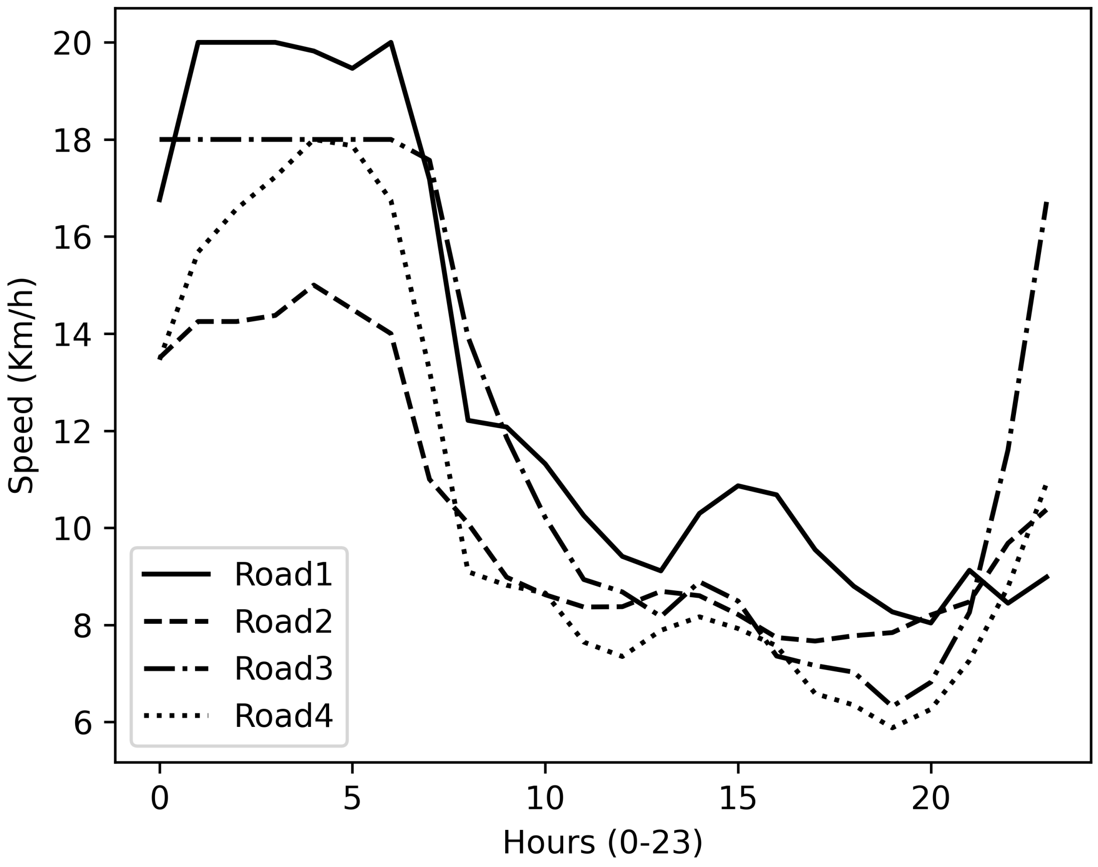


Figure 4.3: Hourly average speed throughout a day

explain the response variable $Road1$ and would play a vital role as an explanatory variable in the regression equation. To see how these categorical features along with other non-categorical features are correlated with our response variable $Road1$, The Pearson correlation coefficient of these explanatory features is shown in Table 4.1. These feature engineering steps result in our proposed Bayesian linear regression formula as shown in eq. (4.1).

$$Road1 \sim AM + Peakhour + Road2 + Road3 + Road4 \tag{4.1}$$

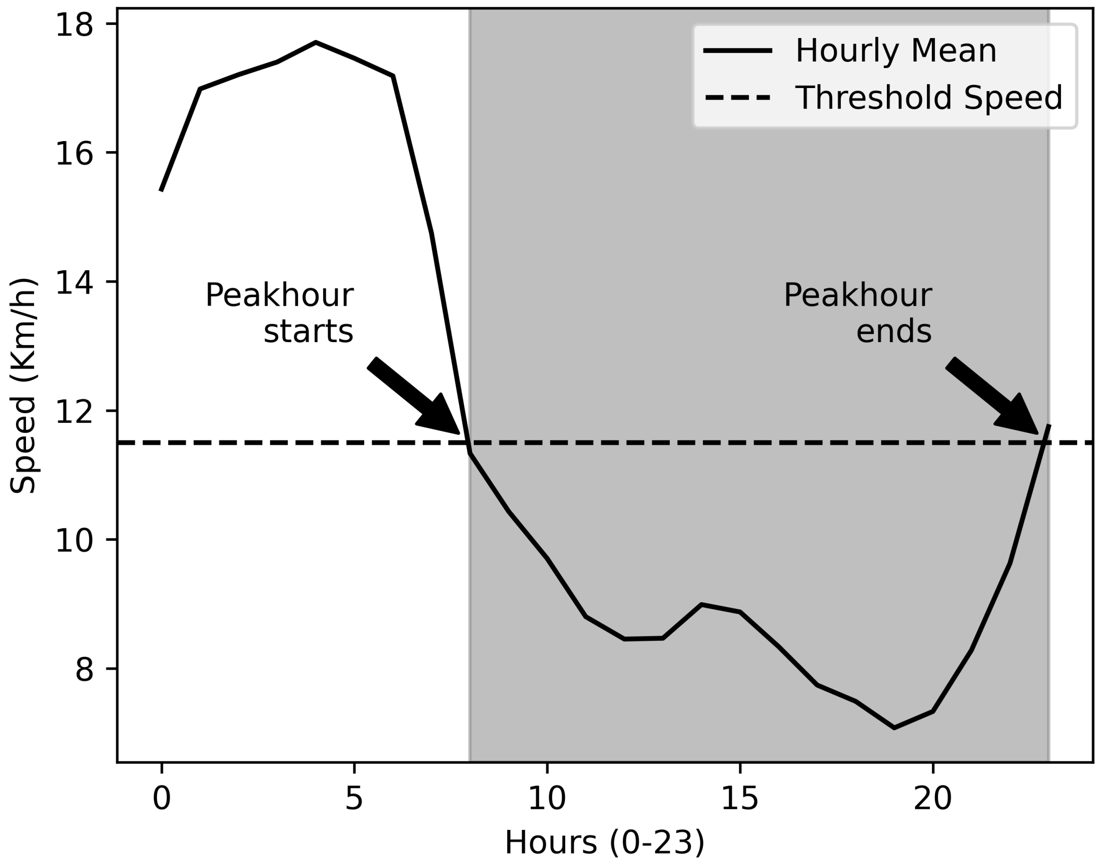


Figure 4.4: Identifying threshold for Peakhour

The corresponding Bayesian network behind the regression model is shown in Figure 4.7. There are two types of nodes in the Bayesian network: temporal and spatial. The AM and $Peakhour$ nodes represent the temporal components which are categorical features extracted from the time series. $Road2$, $Road3$ and $Road4$ represent the spatial components which are continuous features. In the Bayesian network, $Road1$ depends on these five nodes which are represented in eq. (4.1). The network is a directed acyclic graph where $Road1$ has five incoming edges in the topological order. Although the Bayesian network can have a multi hierarchy among nodes and edges, Bayesian inference-based regression

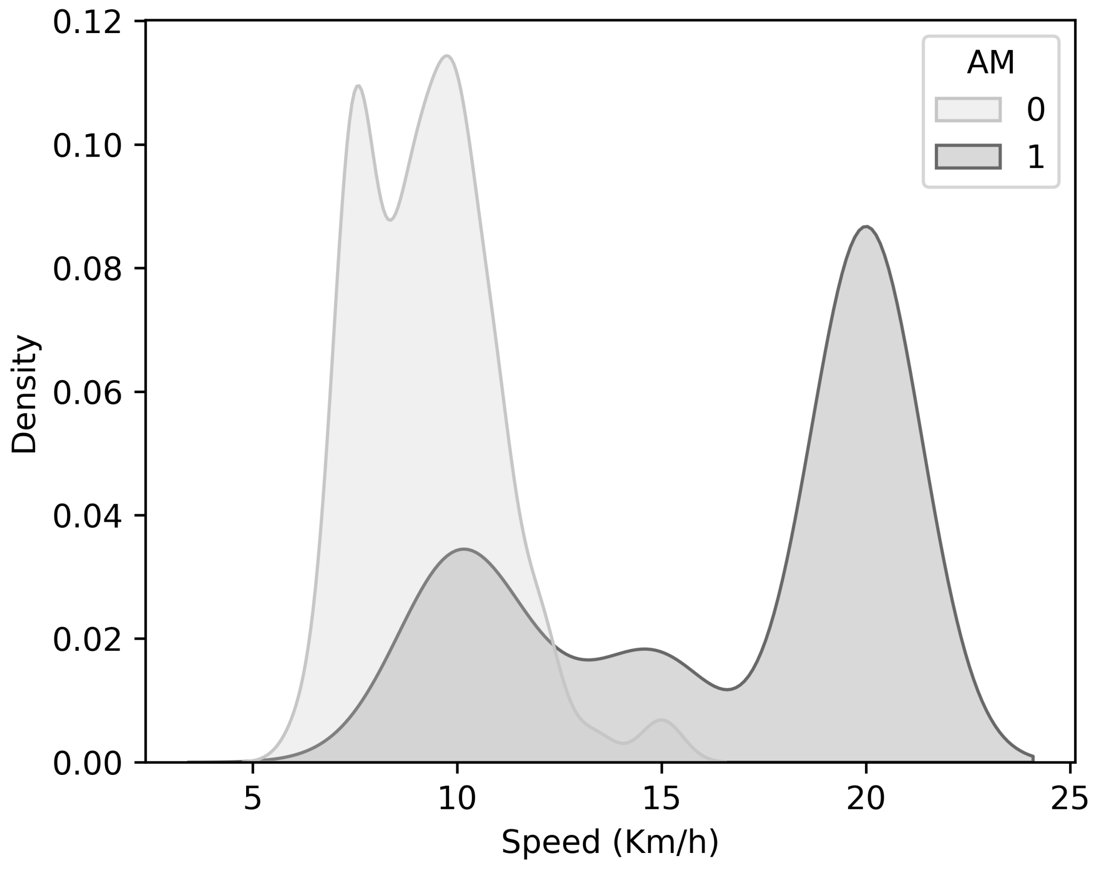


Figure 4.5: Density Distribution By AM

modeling has only one layer of explanatory variables that explains the response variable.

Bayesian inference-based regression modeling is scalable and can be incorporated with any number of explanatory variables depending on the context. For example, in our case, $Road1$ depends on $Road2$, $Road3$ and $Road4$. The model can be extended for traffic road scenarios where the traffic congestion level of a road has a direct dependency on the traffic congestion level of n number of roads.

$$Road1 \sim AM + Peakhour + Road2 + Road3 + Road4 \tag{4.2}$$

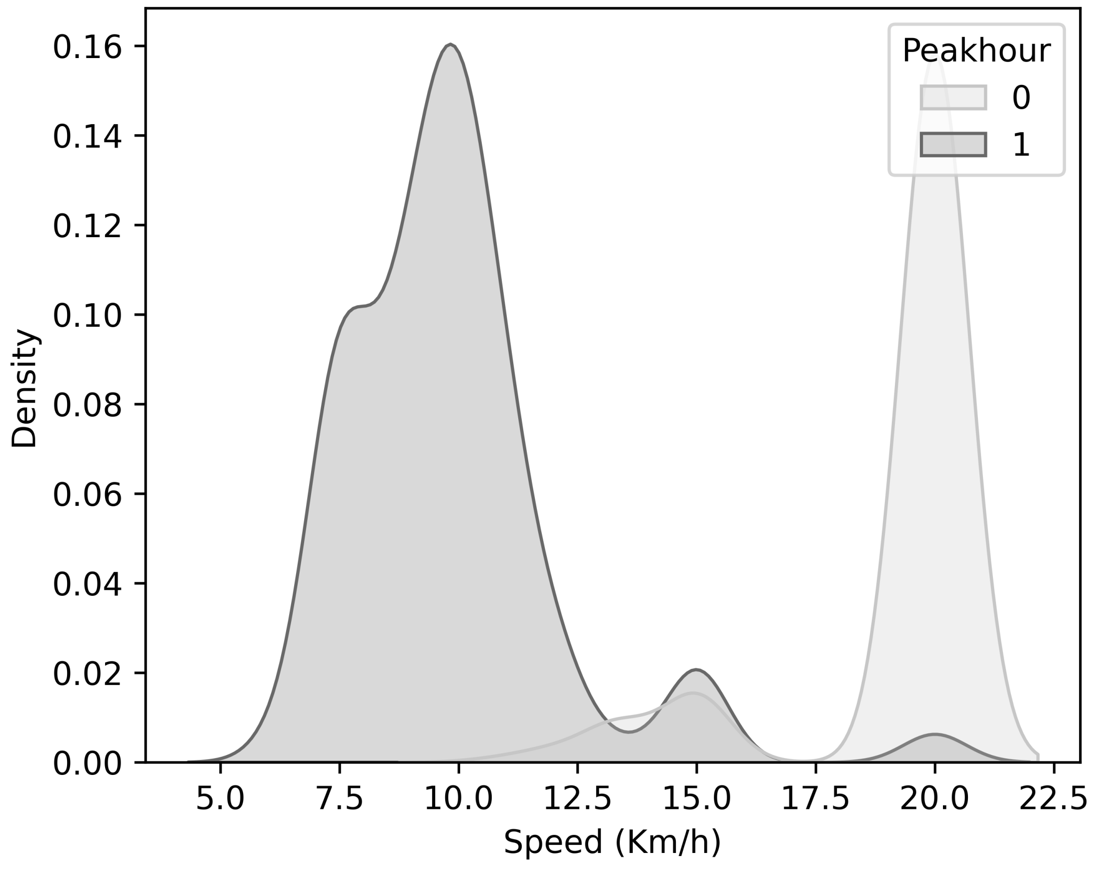


Figure 4.6: Density Distribution By Peakhour

4.3 Methodology

Instead of finding a single point estimate, Bayesian linear regression formulates a probability distribution of the outcomes. The response variable is drawn from a probability distribution rather than estimated as a single value. Eq. (4.3) is a Bayesian linear regression model that samples the response variable from a normal distribution.

$$y \sim N(\beta^T X, \sigma^2 I) \tag{4.3}$$

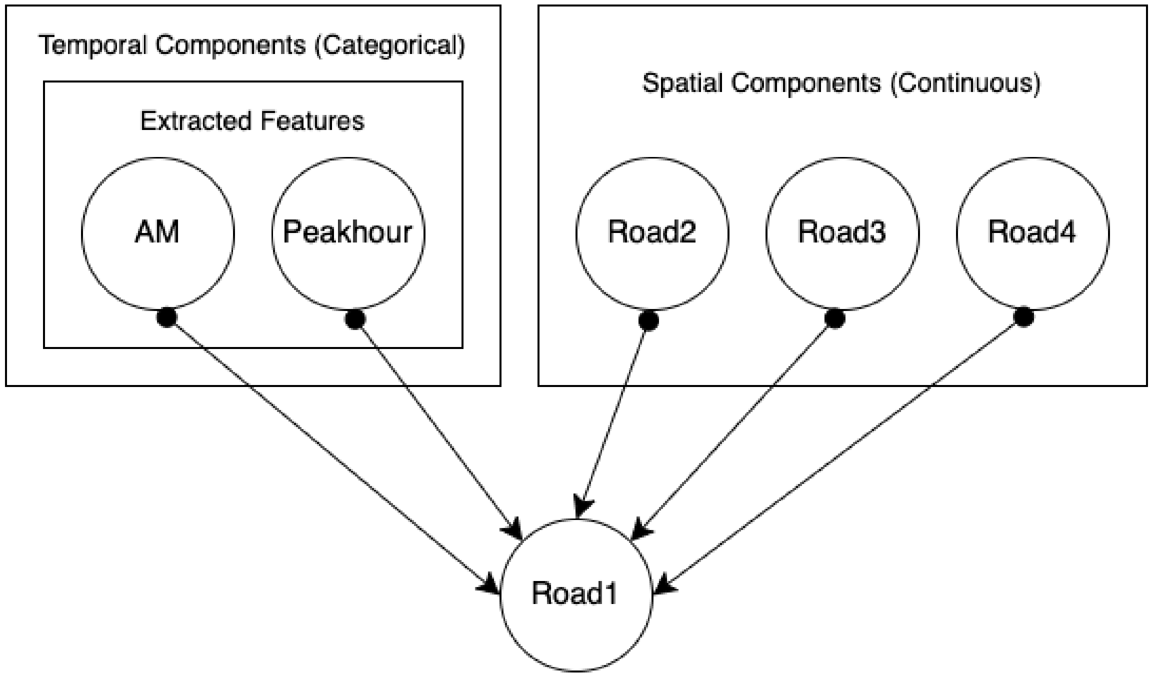

Temporal Components (Categorical)
Extracted Features
AM
Peakhour
Spatial Components (Continuous)
Road2
Road3
Road4
Road1

Figure 4.7: Structure of the Proposed Bayesian Network

In eq. (4.3), the response variable y is generated from a Gaussian normal distribution, which is characterized by a mean and variance. The mean in the Bayesian linear regression is the multiplication of the transpose of the coefficient matrix and the predictor matrix. Due to the multi-dimensional formulation of the model, the square of the standard deviation, σ is multiplied by the identity matrix I to get the variance as shown in the equation.

Rather than finding the best estimate of the response variable, Bayesian linear regression provides the posterior probability distribution. The model learns from the training dataset and estimates the model parameters from a probability distribution along with the response variable.

$$P(\beta \mid y, X) = \frac{P(y \mid \beta, X) * P(\beta \mid X)}{P(y \mid X)} \tag{4.4}$$

$$P(Posterior) = \frac{Likelihood * Prior}{Normalization} \tag{4.5}$$

Eq. (4.4) refers to the Bayes Theorem which is the fundamental building block of Bayesian linear regression. Here, $P(\beta \mid y, X)$ is the posterior probability distribution of the model parameters, $P(y \mid \beta, X)$ is the likelihood of the data, $P(\beta \mid X)$ is the prior probability of the parameters, and $P(y \mid X)$ is the normalization constant. The posterior distribution of the model parameters is proportional to the multiplication of the likelihood of the data and the prior probability of the parameters.

Fewer data points make the posterior distribution more spread out. With the increase of data points, the data likelihood gets prioritized over the initially estimated prior. In the case of very large data points, the mean of the posterior probability distribution for the model parameters converges to the values obtained from linear regression. Evaluating the posterior distribution for continuous variables is intractable. To overcome this issue, sampling methods like Monte Carlo Markov Chain are used to draw samples from the posterior probability distribution to approximate the posterior [38].

The sampling method that is used in our methodology is the No-U-Turn Sampler (NUTS), which generates samples from the posterior distribution [43]. The NUTS mechanism can explore high-dimensional distributions unlike the default Gibbs sampler, which results in a better convergence [39]. One of the main features of NUTS that makes it a good candidate for our model is avoiding the random walk when exploring samples. Another important feature of NUTS is automatic parameter tuning. Obtaining the optimal parameter setting to draw the samples leads to improving the performance of our Bayesian model.

In NUTS, we are required to specify two values for parameter tuning. The first one is to set the step size ϵ along with the required number of steps L in each chain [40]. Setting a large step-size increases the efficiency of accepting and rejecting the generated samples. Although the rate of rejecting the generated samples in NUTS is higher than in the Gibbs sampler, using NUTS will generate samples from a complex distribution that are adequate to estimate the optimal posterior [41].

We describe the posterior probability distribution and evaluate how the proposed model works for test observations. We evaluate the effect of the model explanatory variables on the target variable $Road1$. We apply Bayesian linear regression and compare the results with two frequentist linear regression approaches, multiple linear regression and ElasticNet regression. ElasticNet regression is a regularized linear regression approach that combines lasso and ridge methods to incorporate L1 and L2 penalties within the model for prediction [42]. Table 4.2 shows all approaches and their tuning parameters compared to the proposed Bayesian linear regression model.

4.4 Experimental Setup

The dataset is split into 70% for training and 30% testing. The training and testing dataset are randomly selected with a random state 42. The experiments run on Google Colaboratory using 12GB NVIDIA Tesla K80 GPU. The prior for the data likelihood is set as a Normal Distribution. The posterior is sampled by the Markov Chain Monte Carlo (MCMC) sampler. The MCMC sampling is conducted using No-U-Turn Sampler [43]. Both the number of samples and tuning steps are specified as 1000. The number of chains is set to 2 to run 2 Markov sampling chains in parallel to check whether both converge to the expected outcome.

Table 4.2: Different Regressors and their Parameters

Approach	**Parameters**
Linear Regression	Default Parameters

ElasticNet Regression	alpha=1.0
	L1_ratio=0.5

Bayesian Linear Regression	Samples=1000
	Tune=3
	Chain=1.0
	Prior=Normal Distribution

4.5 Results and Discussions

4.5.1 Posterior Probability Distributions

The outcome of our Bayesian linear regression is the distribution of the model parameters. The model does not provide an exact estimate for a feature, but the mean value of the distribution can be considered as an accurate estimate for the feature. The benefit of having posterior probability distribution is that the model also provides an entire range of values that shows the uncertainty of the true values. The left side of Figure 4.8 shows the posterior probability distribution for both the categorical and non-categorical model parameters. The progression of the drawn samples of each distribution is shown on the right side of the trace plot.

Table 4.3 lists the posterior probability distributions of all model parameters after training the model. The mean values are the most likely estimate for each correspondent parameter. SD is the standard deviation in the data likelihood that indicates uncertainty in $Road1$. If we increase our sample size, the uncertainty in the model will decrease. HPD 3% and HPD 97% refer to the 3% and 97% Highest Posterior Density, respectively, which is a credible interval for the model parameters. The upper and lower bound of HPD limits and standard deviation show the confidence in the model parameters.

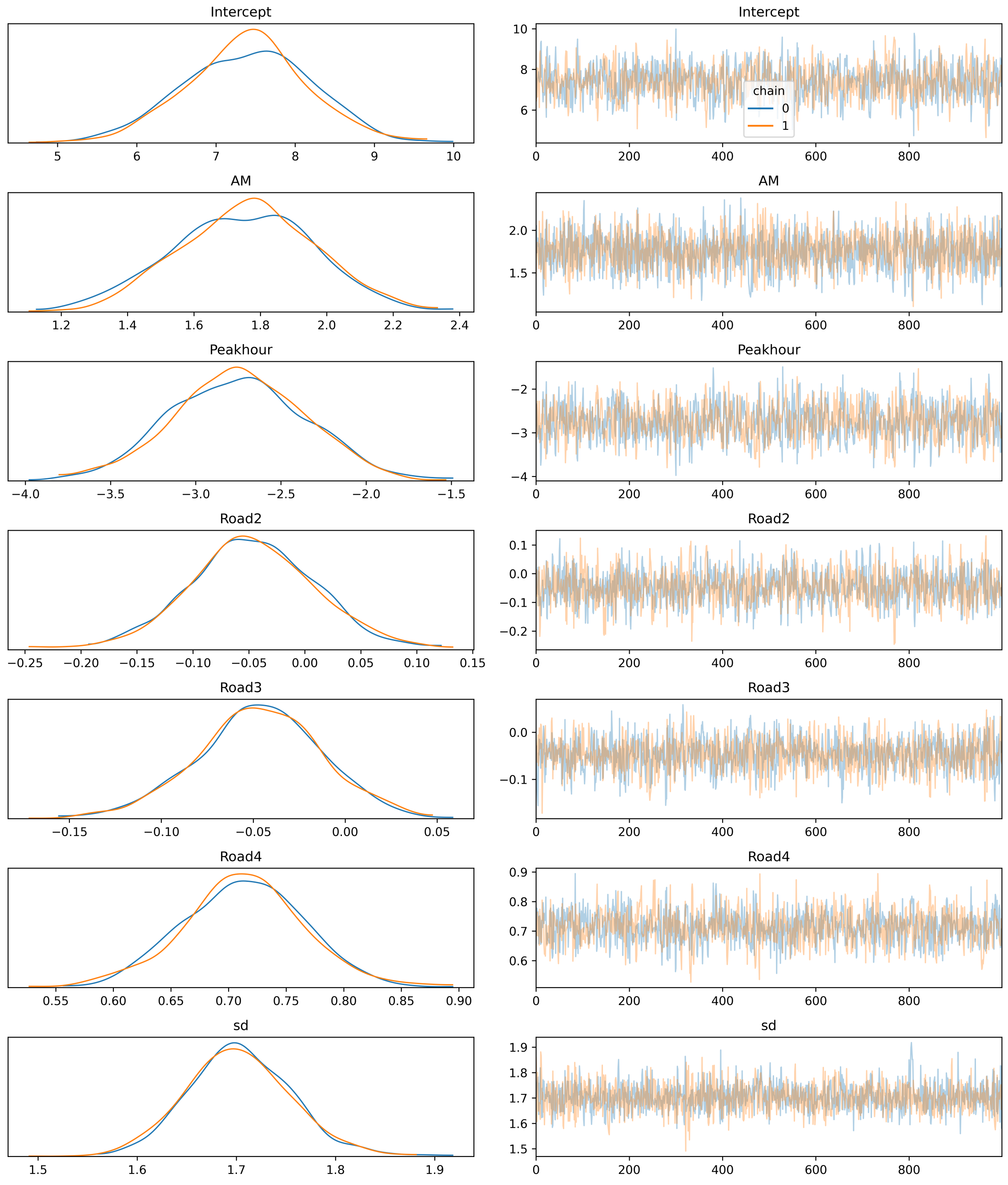


Figure 4.8: Posterior Probability Distribution

Table 4.3: Summary of Posterior Probability Distribution

Variable	Mean	Standard Deviation	HPD 3%	HPD 97%
Intercept	7.416	0.765	5.879	8.799
AM	1.756	0.215	1.373	2.181
Peakhour	-2.752	0.386	-3.478	-2.019
Road2	-0.048	0.057	-0.153	0.060
Road3	-0.048	0.033	-0.109	0.013
Road4	0.714	0.054	0.606	0.811
SD	1.700	0.053	1.601	1.796

The mean of a posterior probability distribution is taken as the best estimate of that model parameter. These mean estimates of these model parameters are put together in eq. (4.1) to derive a new eq. (4.6) that represents the Bayesian linear regression equation. The model can be easily interpreted like the frequentist approach as every unit of change in an explanatory variable will affect a corresponding coefficient unit amount of change in the response variable.

Although the Pearson correlation coefficient between the target variable $Road1$ and explanatory variables $Road2$ and $Road3$ is positive, the corresponding sign of these two input variables is negative as shown in eq. (4.6). This is due to multicollinearity, where input features explaining the output variable are also correlated with each other [44]. Thus, a change in an input feature causes a change in the correlated input feature. The proposed model fails to identify the effect of an input feature on the target variable due to the confounding behavior of another input feature. Input features with multicollinearity correspond to high standard error and are statistically insignificant in explaining the target variable. Taking out highly correlated features through feature selection could be an option to get rid of multicollinearity.

$$\begin{aligned} Road1 = 7.4163 * Intercept + 1.7561 * AM - 2.7517 * Peakhour \\ -0.0477 * Road2 - 0.0479 * Road3 + 0.7139 * Road4 + 1.7003 * SD \end{aligned} \tag{4.6}$$

4.5.2 Test Observations

To demonstrate how the proposed model works, we test some observations from the testing dataset and compare the actual outcomes with the predicted ones. To construct the probability density function for $Road1$, we provide the values of the model parameters of a test observation in eq. (4.6). To represent a test observation in the testing dataset, we use a set of values that corresponds to the value for each one of the features in the same order as shown in eq. (4.1).

Four random observations from the testing dataset along with the probability density function of $Road1$ are shown in Figure 4.9, 4.10, 4.11 and 4.12. The dotted lines represent the true value of $Road1$ and the straight lines are the mean of the probability distribution which represents the best estimate for the distributions. The estimated value provided by the model is very close to the true value in Figure 4.9, 4.10, and 4.12. However, the model does not provide close estimation as shown in Figure 4.11.

4.5.3 New Observations

We inject some new observations in our model to evaluate its performance for new and modified observations. We plug four new observations into our model and estimate the value of $Road1$ as shown in Figure 4.13, 4.14, 4.15 and 4.16. The model provides a posterior distribution of all possible values of the response variable for every new combination of explanatory variables. The mean estimate of these possible estimations is highlighted

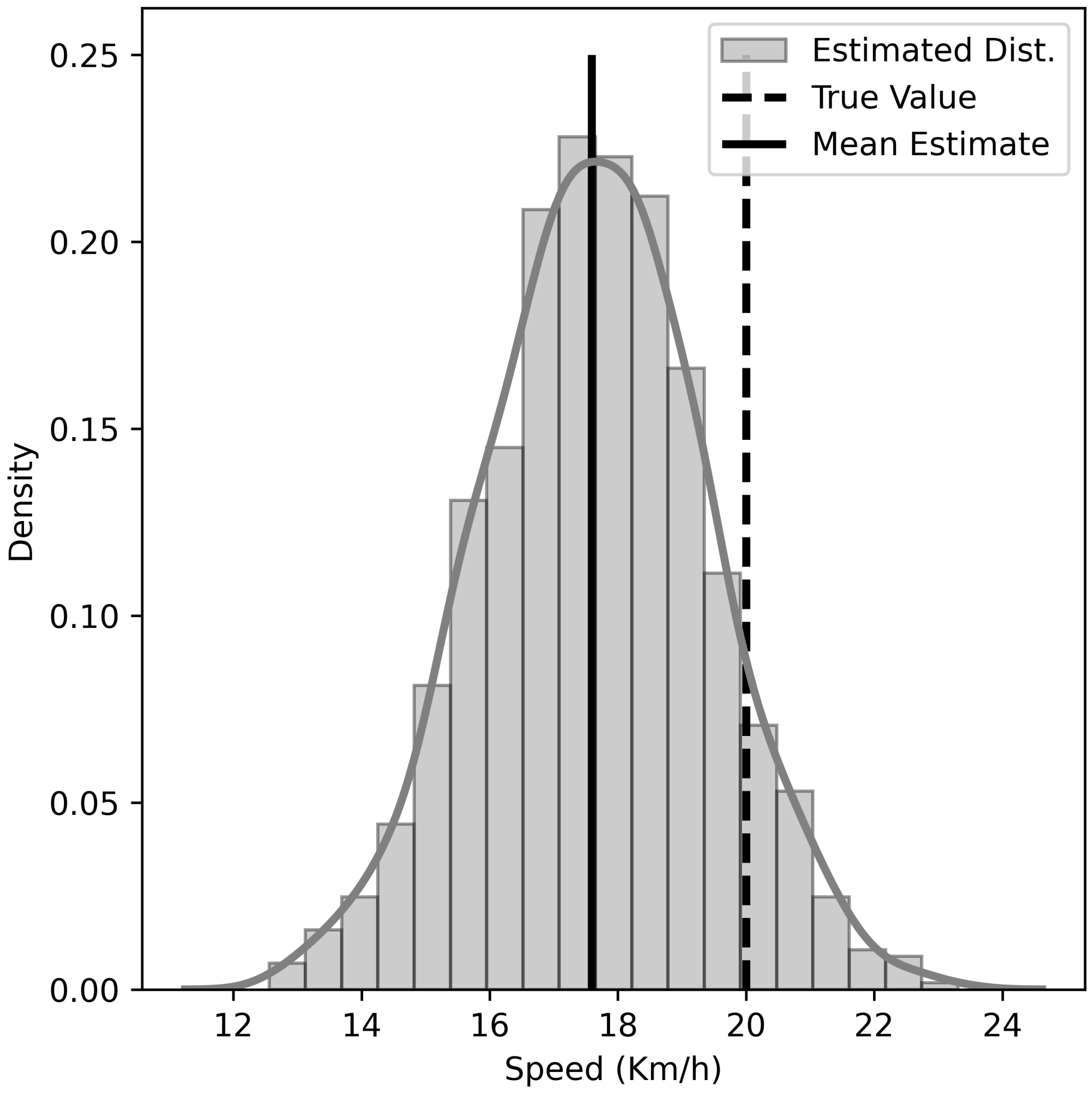


Figure 4.9: Model Output for the Test observation (1,0,21,18,14)

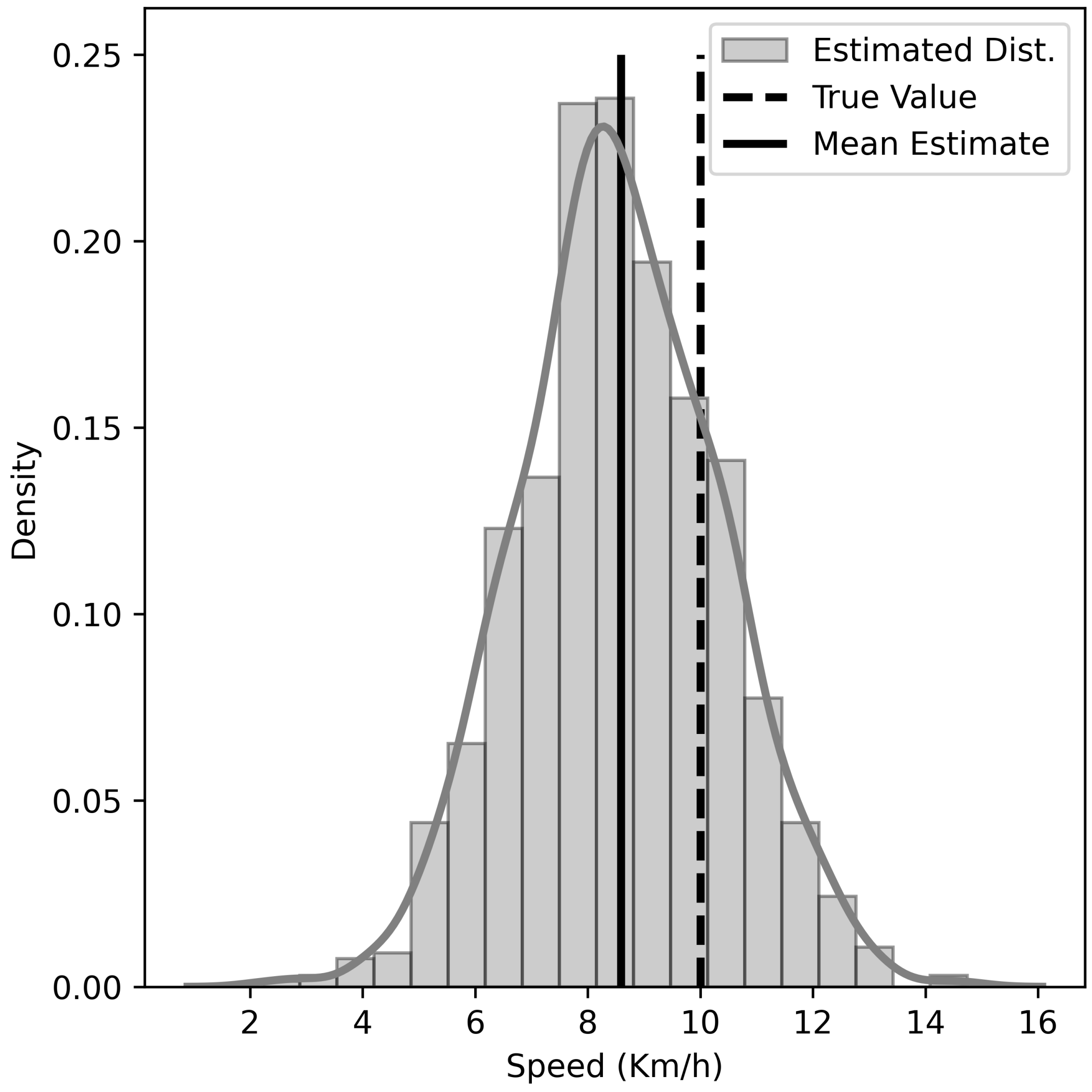


Figure 4.10: Model Output for the Test observation (0,1,7.63,8,6.55)

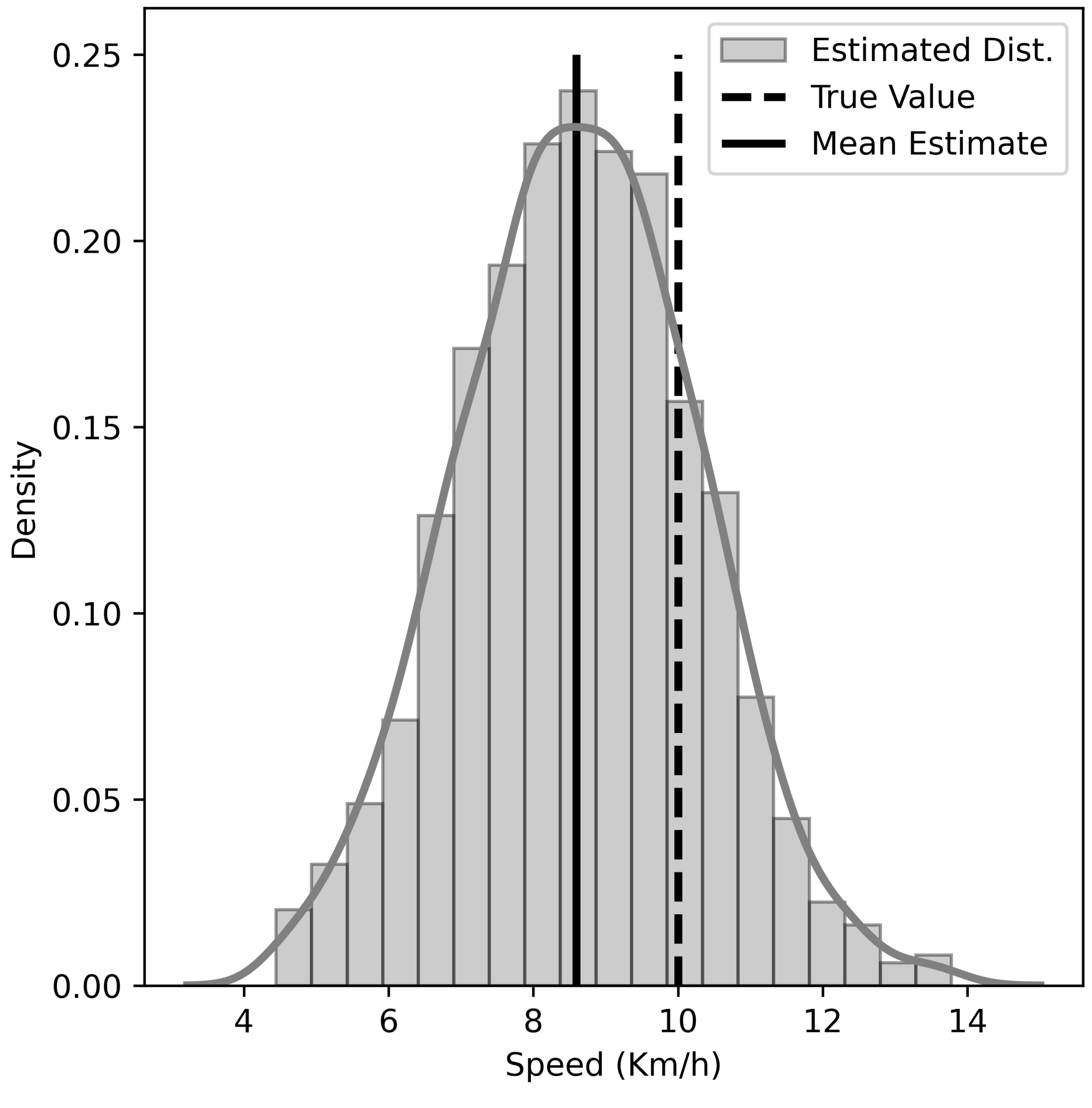


Figure 4.11: Model Output for the Test observation (0,1,8.4,7.2,6.54)

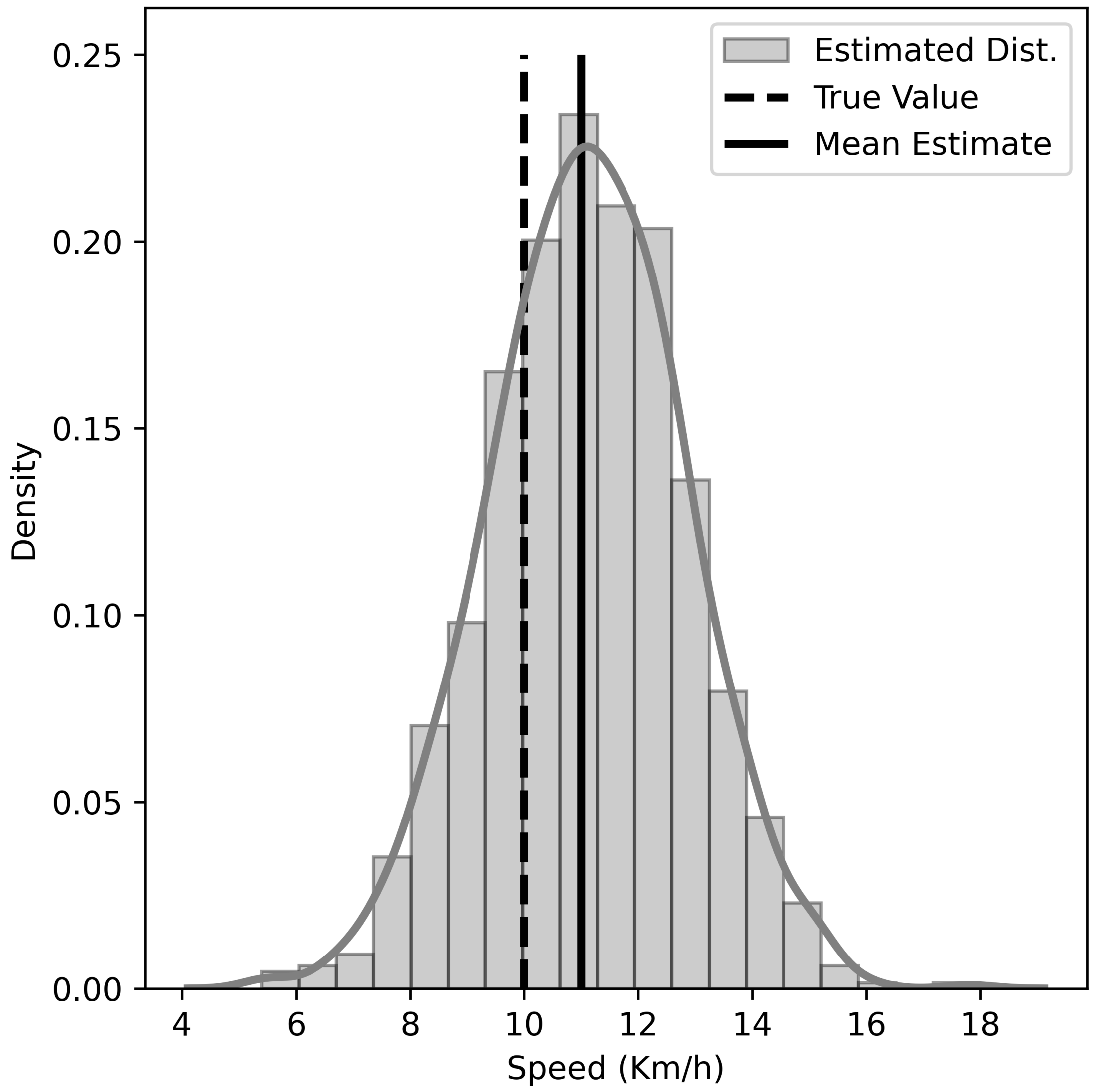


Figure 4.12: Model Output for the Test observation (1,1,8.4,9,7.58)

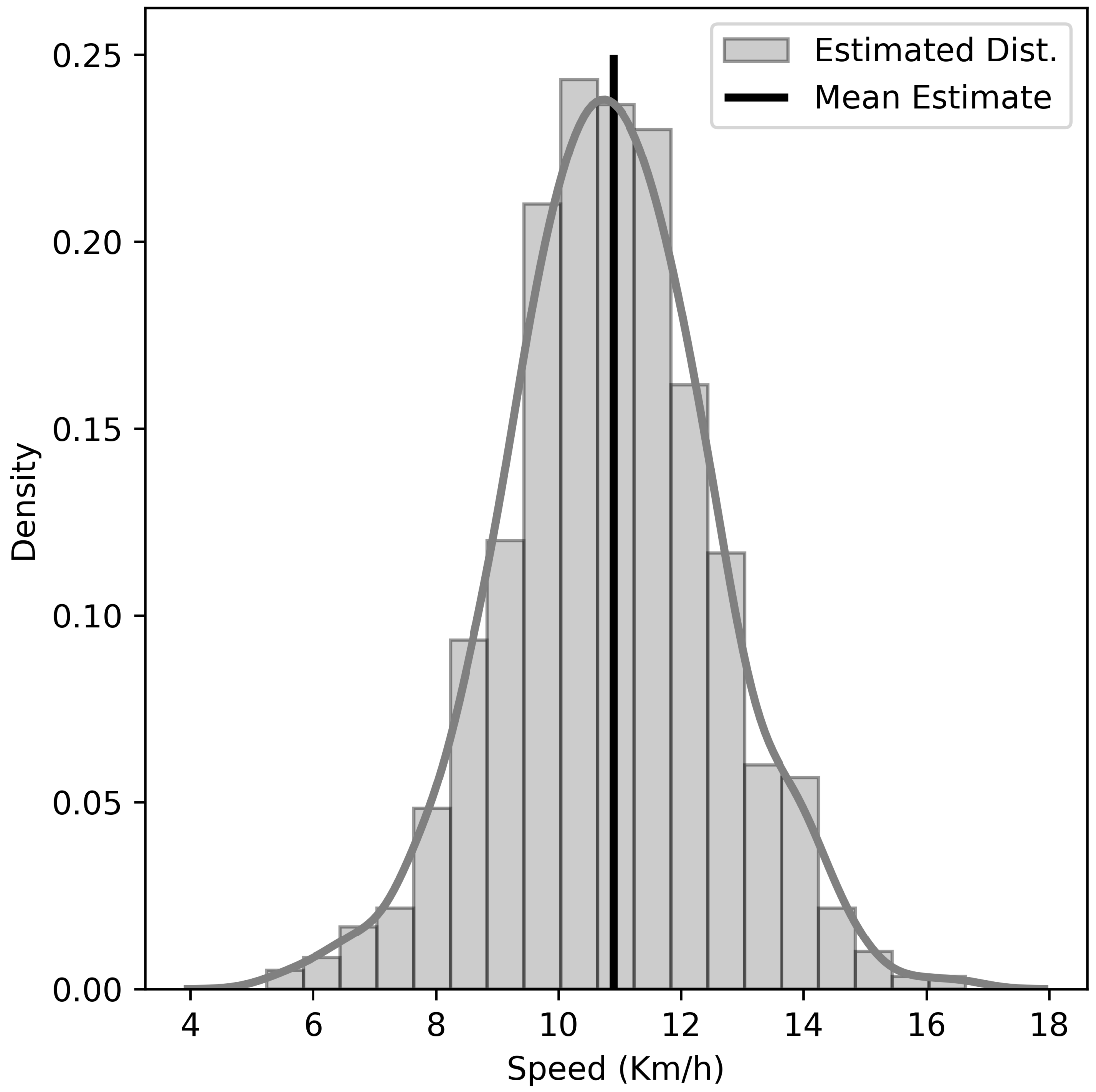


Figure 4.13: Model Output for the New Observation (1,1,5,6,7)

with the vertical straight line. The posterior probability distribution shows the highest probability density near the mean estimation. The deviation from the mean in both directions decreases the probability density. This observation supports the idea that the mean estimation of traffic congestion will be very close to the true value.

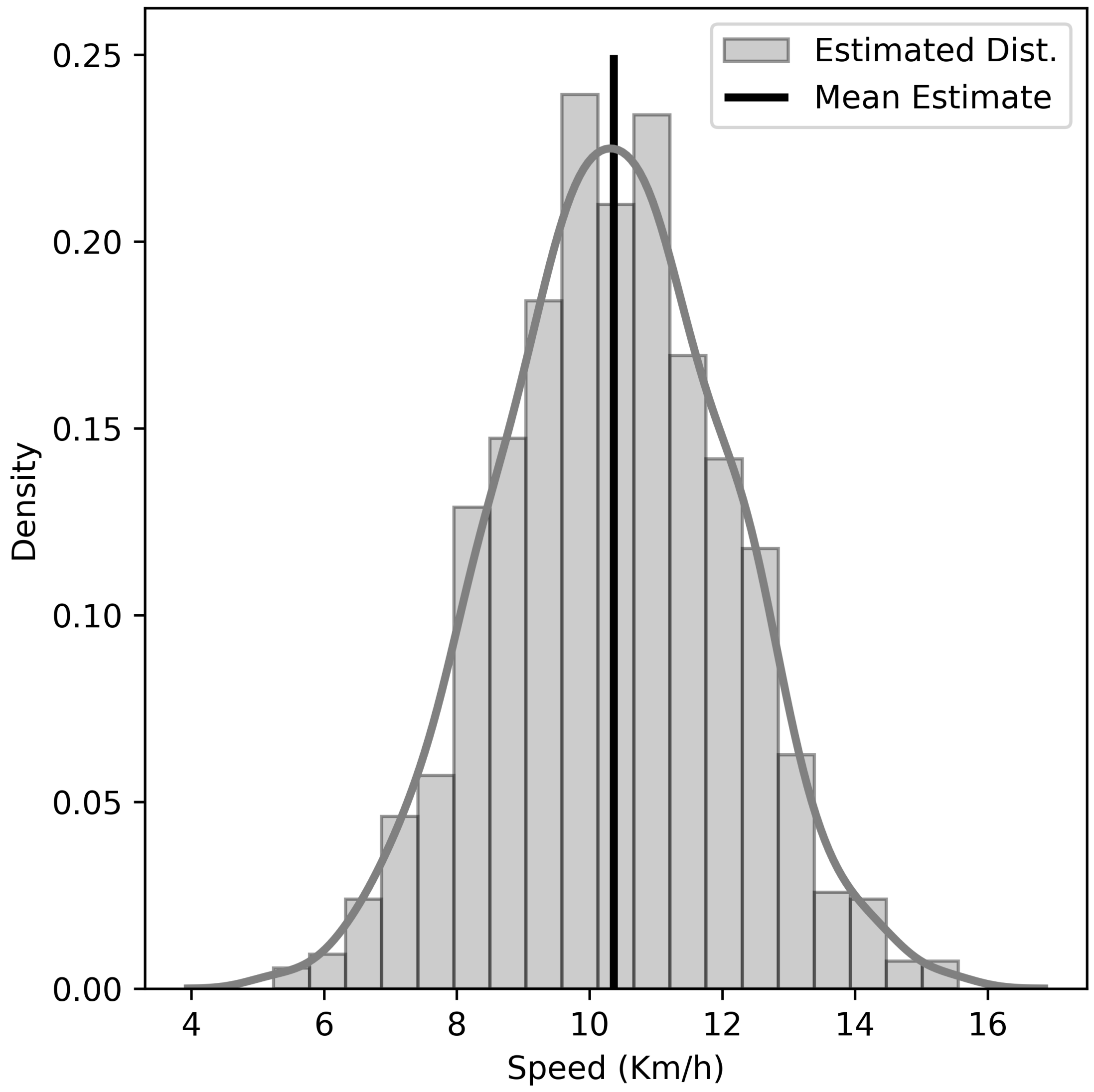


Figure 4.14: Model Output for the New Observation (0,0,6,7,5)

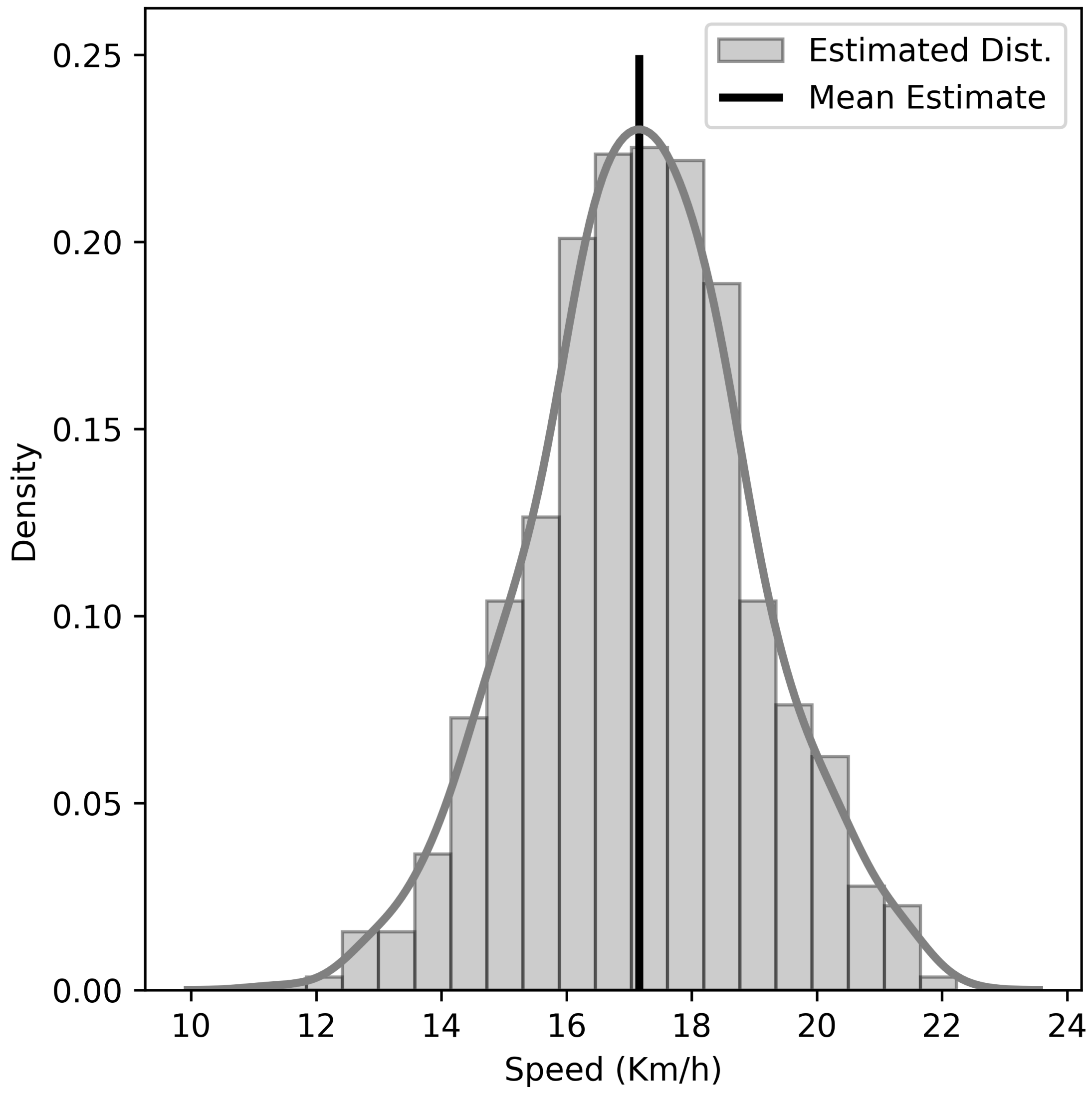


Figure 4.15: Model Output for the New Observation (1,0,15,12,13)

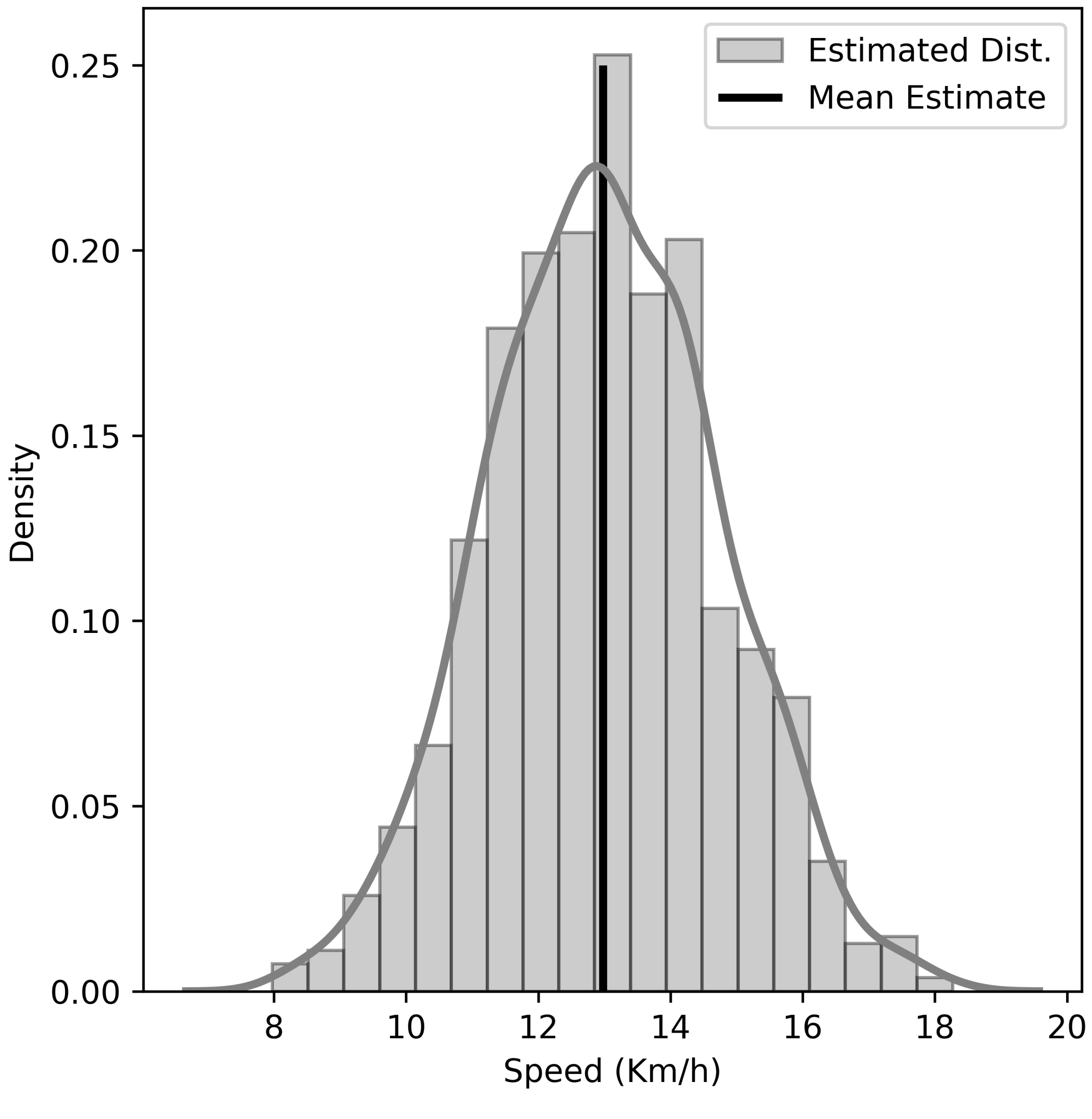


Figure 4.16: Model Output for the New Observation (0,,7,11,9)

4.5.4 Effect of Model Variables

To evaluate the effect of an individual explanatory variable on $Road1$, we iterate through all possible values of that variable while holding all other variables as constant. Then, we can see how the estimated value of $Road1$ changes with respect to that particular variable. We generate a range of values for the query variable and find the estimates of $Road1$ across this range of the posterior distribution. We assume that all model parameters except $Road1$ (the response variable) are at their median value.

In Figure 4.17, 4.18, 4.19, 4.20 and 4.21, there are 100 lines in each plot because the estimated value of $Road1$ vs. the range of a query variable has been sampled from 100 samples of test dataset. The uncertainty in model parameters can be explained by the distribution of these lines. If the lines are more spread out, then the uncertainty in that model parameter increases. Lines will be more spread out with smaller or missing data observations in the training dataset.

Samples drawn from AM and $Peakhour$ show a unidirectional pattern and do not spread out much as shown in Figure 4.19 and Figure 4.20, respectively. But for the query variables, $Road2$ and $Road3$, the lines drawn from the sample have both positive and negative slopes that cancel out each other and result in insignificant model parameters as shown in eq. (4.6). Unlike these two variables, $Road4$ has a strong positive effect on $Road1$ as shown in Figure 4.21.

4.5.5 Comparison With Other Approaches

Bayesian linear regression has advantages over the frequentist linear regression approaches in terms of quantification of model uncertainty, utilization of prior probability distribution and addressing missing data as shown in Table 4.4. The mean absolute error (MAE) and

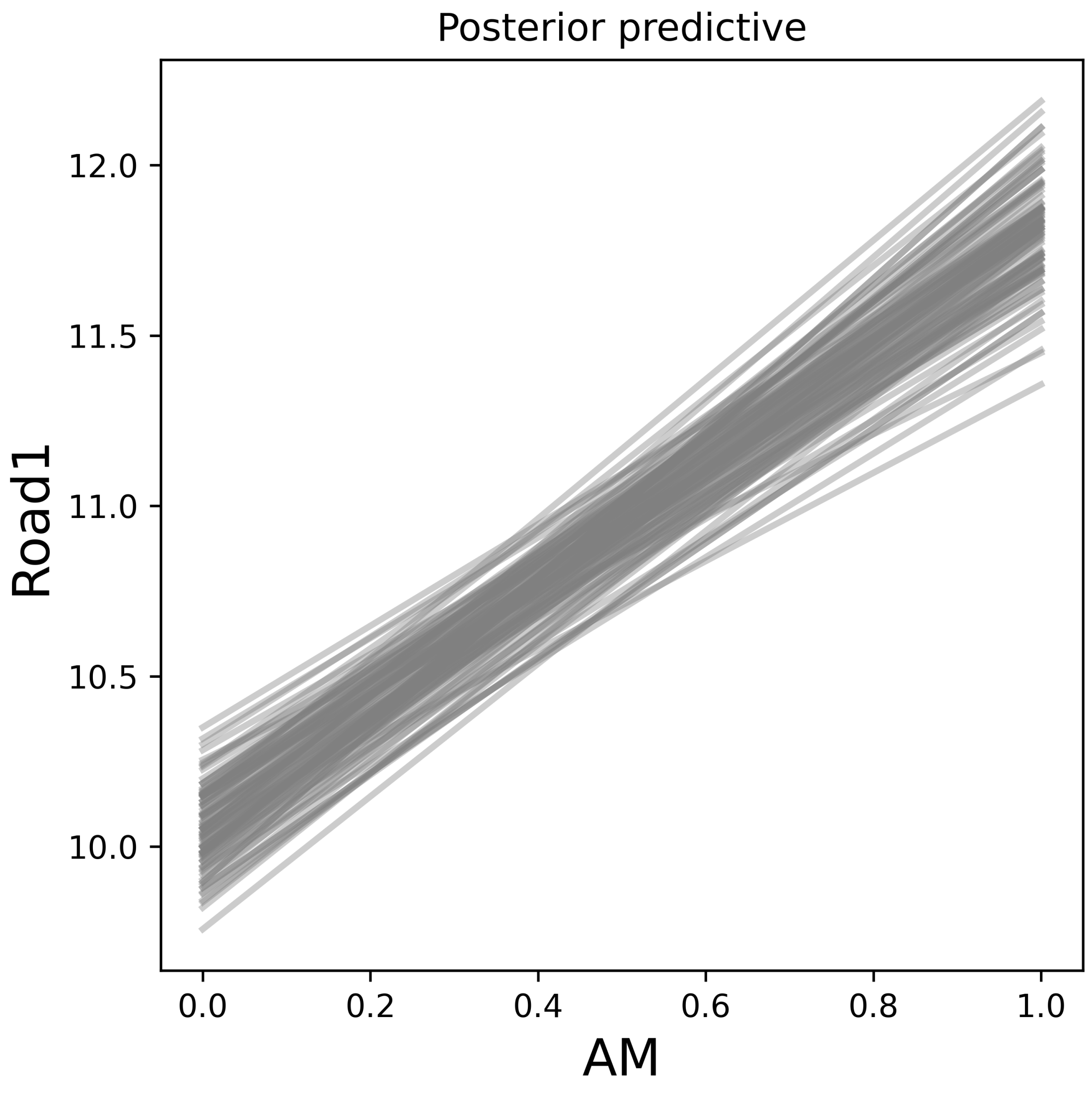


Figure 4.17: Model Variable Effect of AM

Table 4.4: Bayesian Linear Regression Model Comparison Based on Different Features

Prediction Approach	**Interpretability**	**Quantify Uncertainty**	**Prior**	**Missing Data**
Bayesian Linear Regression	Yes	Yes	Yes	Yes
Multiple Linear Regression	Yes	No	No	No
Elastic Net Regression	Yes	No	No	No

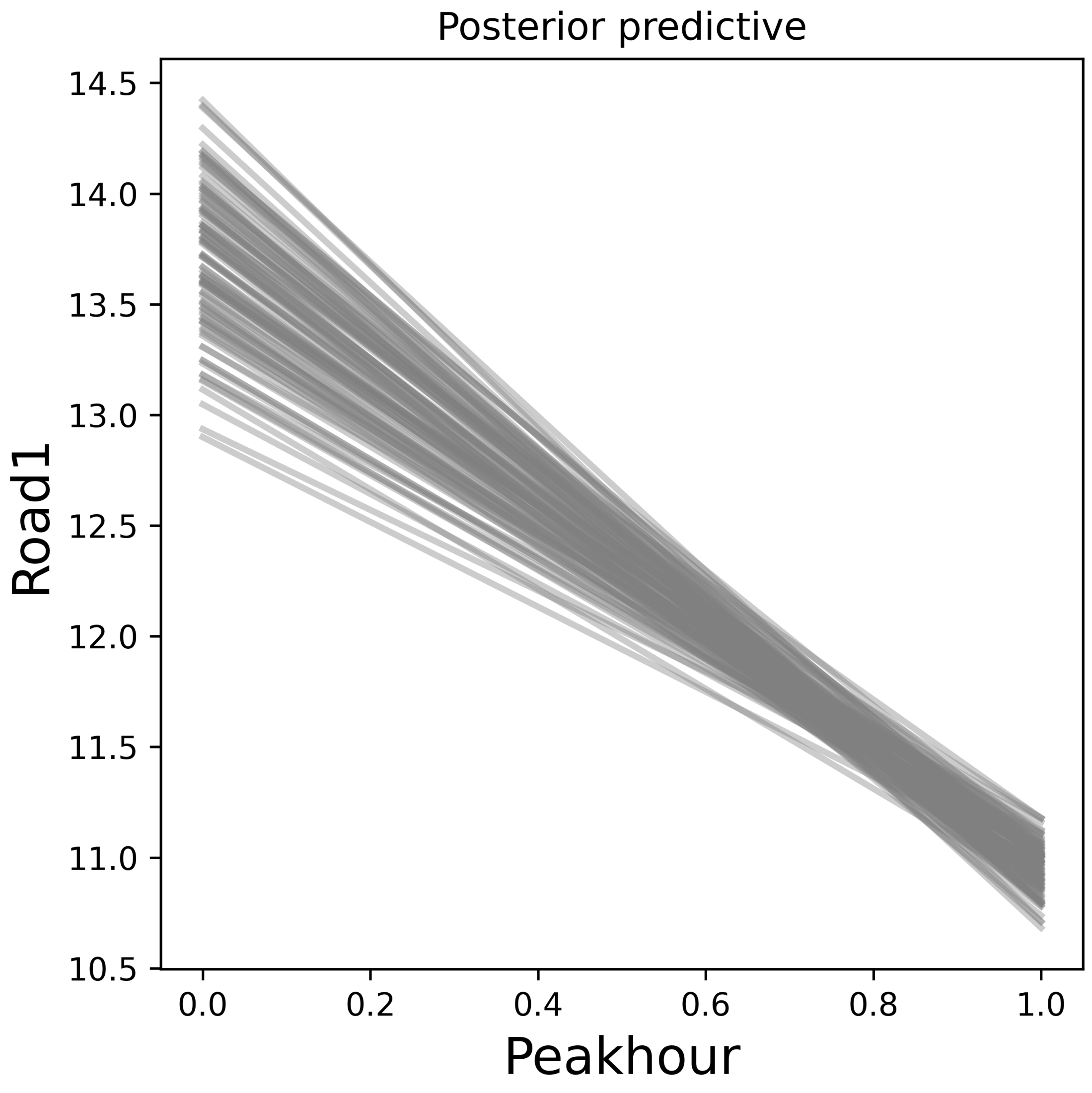


Figure 4.18: Model Variable Effect of $Peakhour$

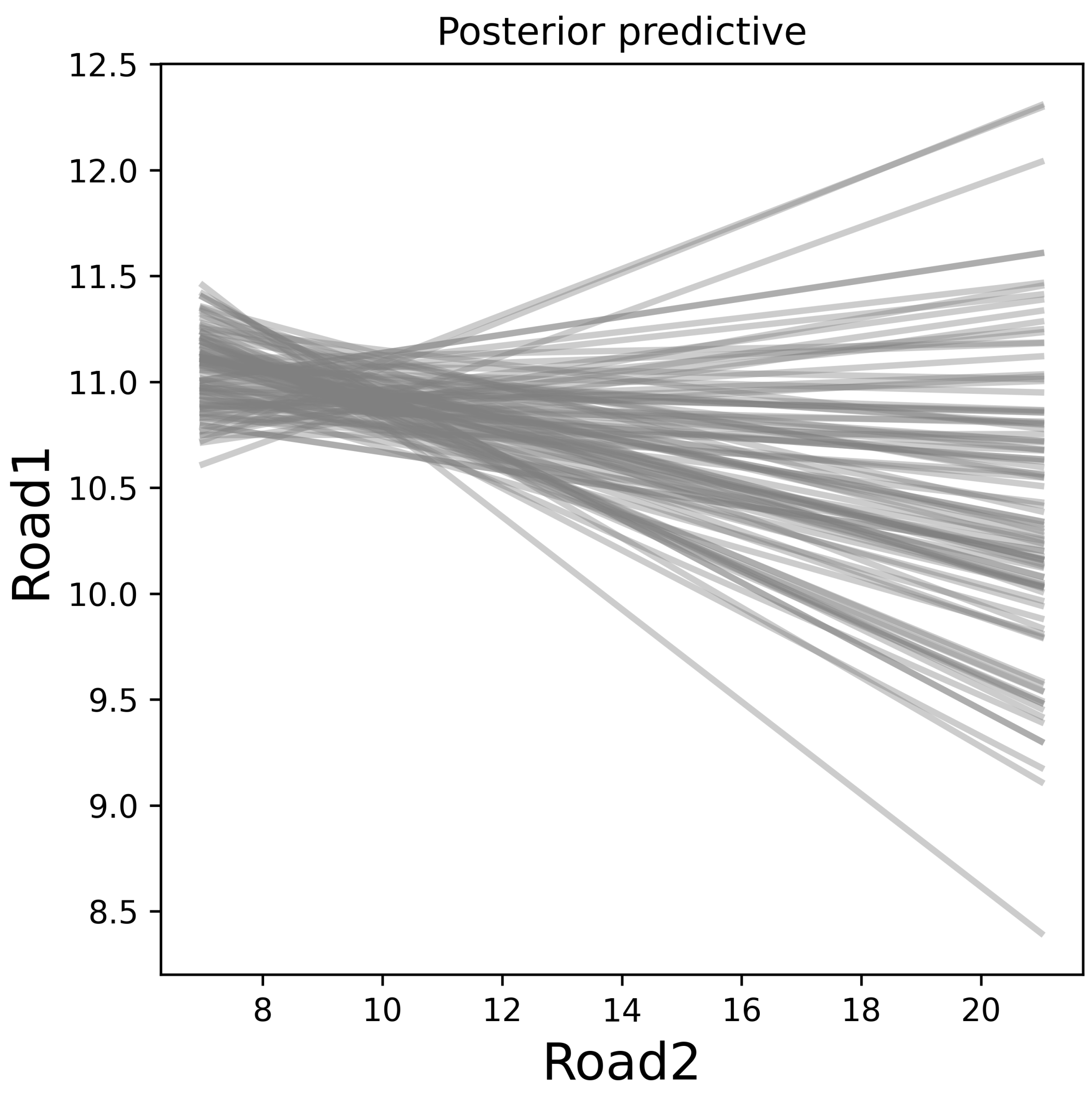


Figure 4.19: Model Variable Effect of $Road2$

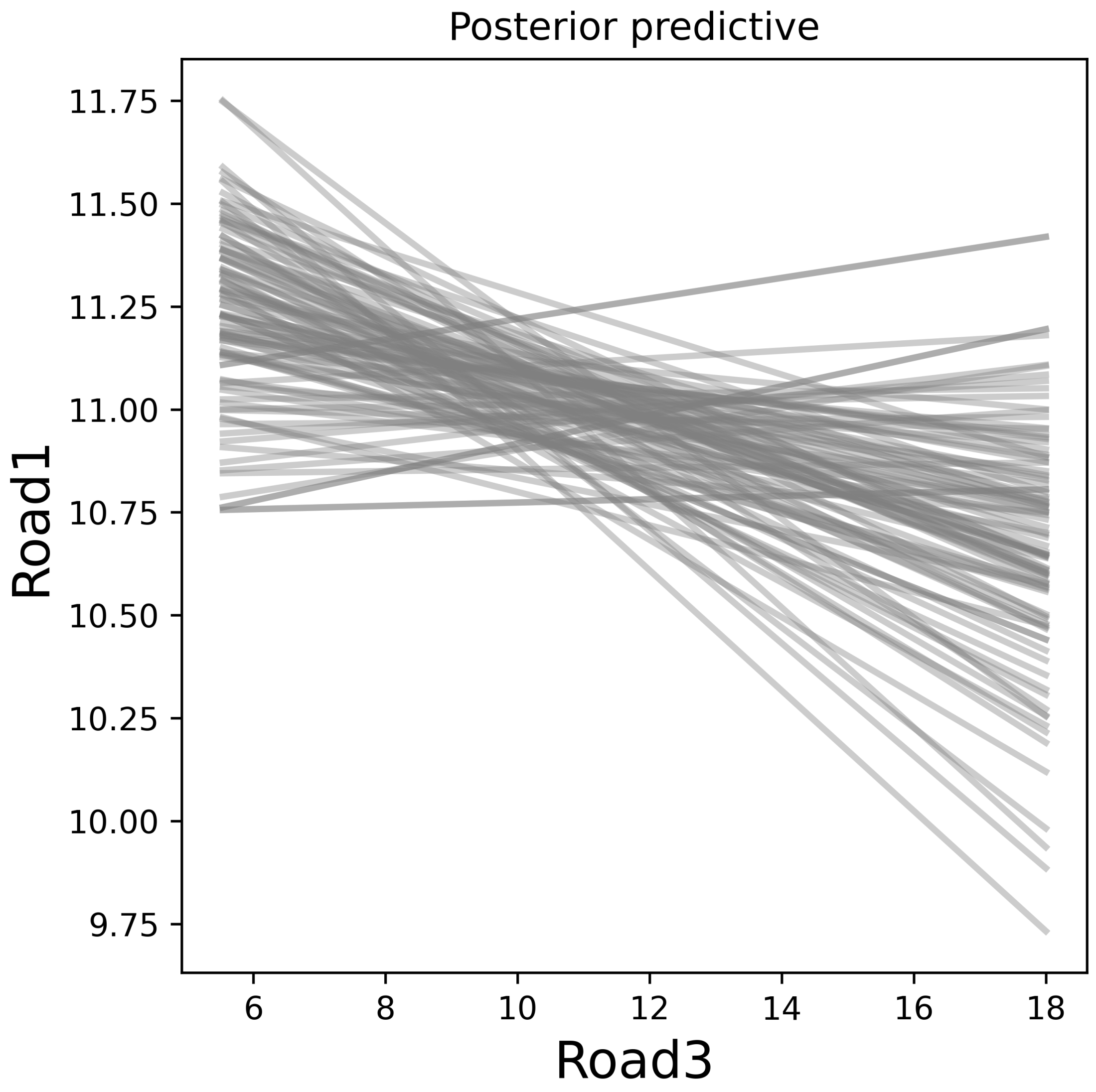


Figure 4.20: Model Variable Effect of $Road3$

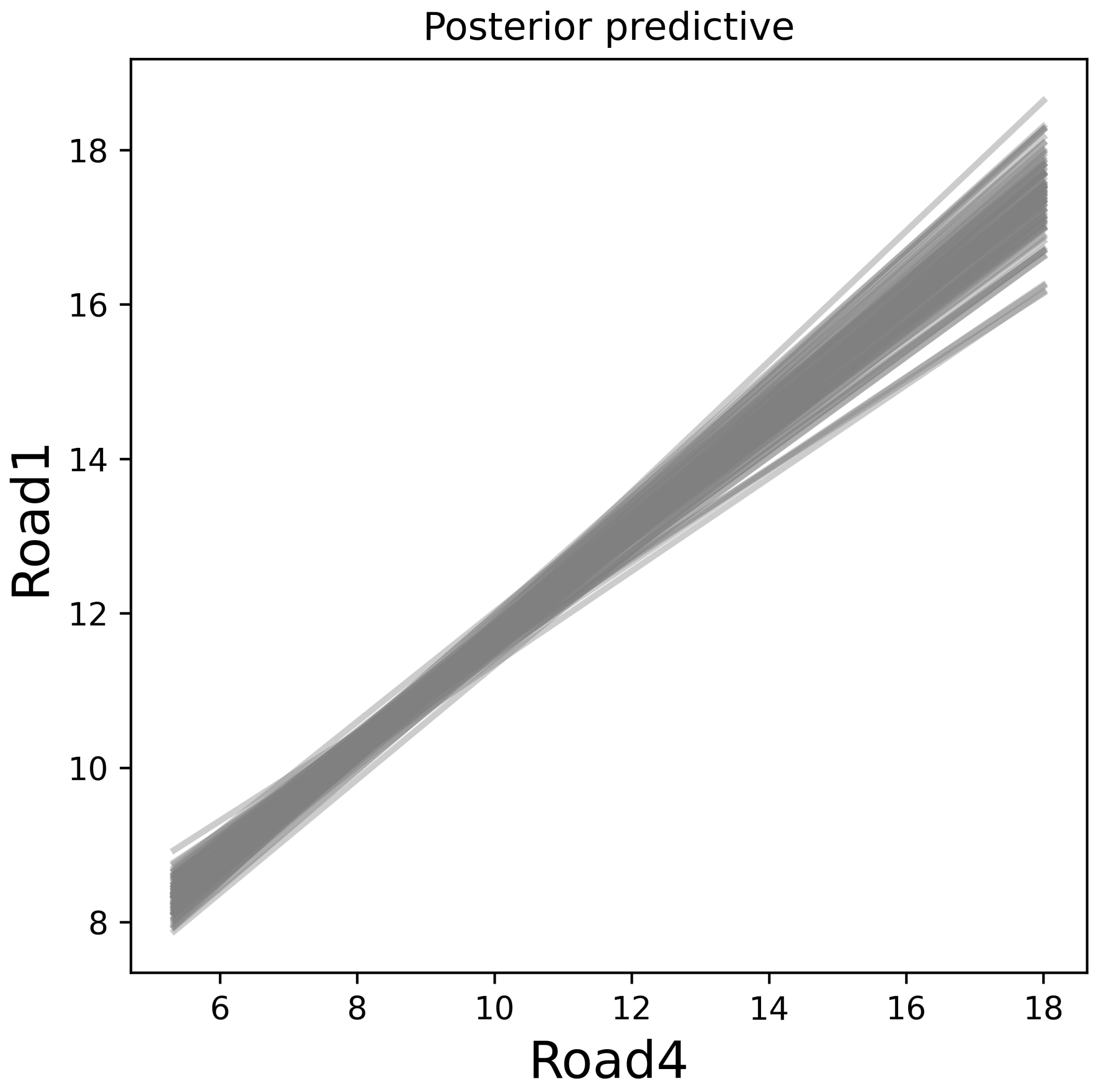


Figure 4.21: Model Variable Effect of $Road4$

root mean squared error (RMSE) are calculated for each model for a side-by-side comparison of model accuracy based on the test dataset. A baseline is hypothesized in the model comparison which is the mean of all possible observations of the traffic speed. For Bayesian linear regression, the means estimated from the posterior probability distribution is considered as the most likely estimation for each of the model parameters. Bayesian linear regression outperforms the state of the art approaches in terms of accuracy as it has the least MAE and RMSE values. Figure 4.22 shows the comparison of these models along with the baseline in terms of the increasing order of MAE and RMSE.

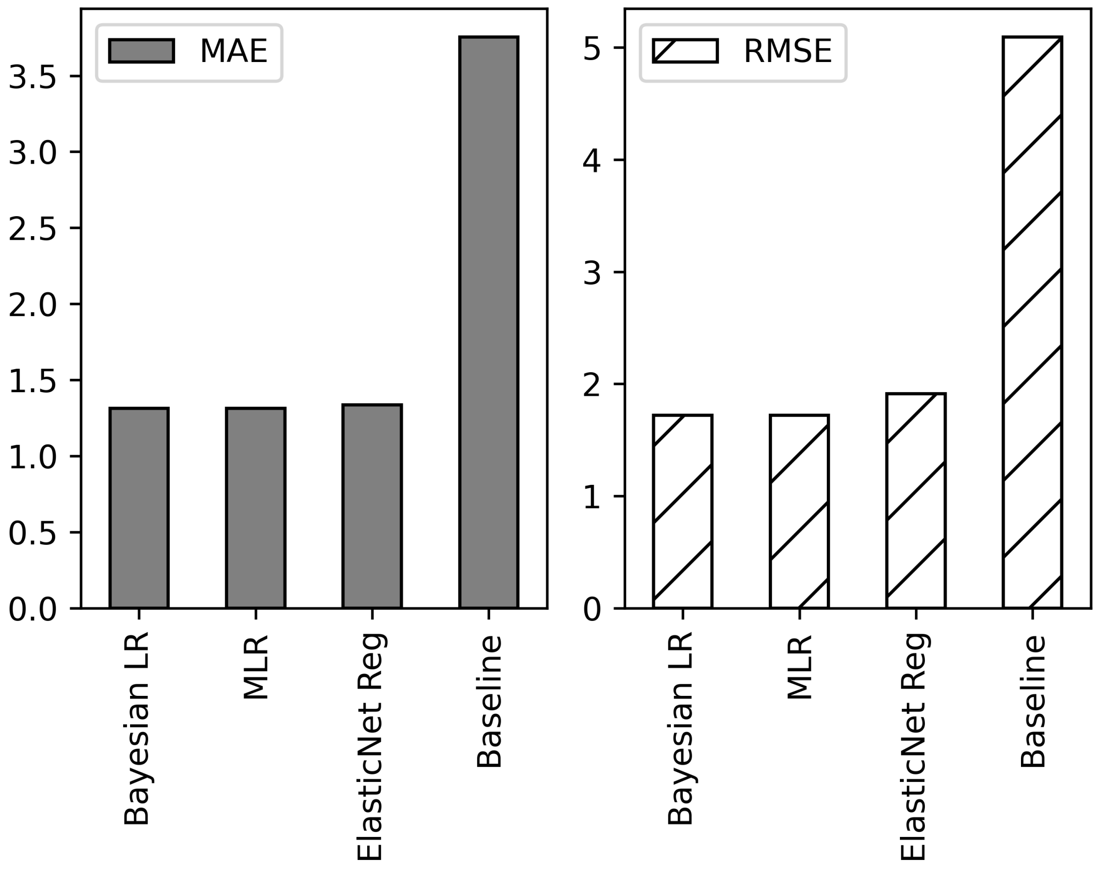


Figure 4.22: Comparison of Bayesian Linear Regression Approach with Different Frequentist Linear Regression Approaches

4.6 Summary

The objective of this work is not to develop a regression model that provides the highest prediction accuracy, but to provide a simple and interpretable model that can fill the gap of state of the art frequentist linear regression approaches. We propose an efficient Bayesian linear regression model for spatiotemporal traffic congestion prediction that matches the interpretability and accuracy of the state of the art frequentist approaches, yet deals with smaller or missing observations by leveraging prior information. The model can quantify uncertainty through the effect of model variables. We conduct exploratory data analysis to extract temporal features and incorporate them into the model. This model could be adopted with complex traffic scenarios, where traffic data observations are non-linear and interrupted. Also, this probabilistic inference based regression approach addresses issues like scalability, model uncertainty and interpretation need to be addressed.

Chapter 5

RegTraffic: An Interactive Map based Traffic Simulator for Regression Analysis

5.1 Introduction

Regression analysis is a dynamic approach for spatiotemporal traffic modeling to predict traffic congestion of a road or area. Software based traffic simulation helps in analyzing complex traffic structures and is a useful method for the development and planning of traffic systems and networks. Regression based traffic simulation can simulate traffic congestion of a road segment based on the different types of events like accidents, roadblocks and bad weather conditions. There is no dedicated traffic simulation tool to simulate and visualize traffic congestion of connected road segments predicted by regression-based traffic modeling. This chapter describes a simple web-based and dynamic traffic simulation software called RegTraffic to simulate and visualize traffic congestion of connected road segments using an interactive map.

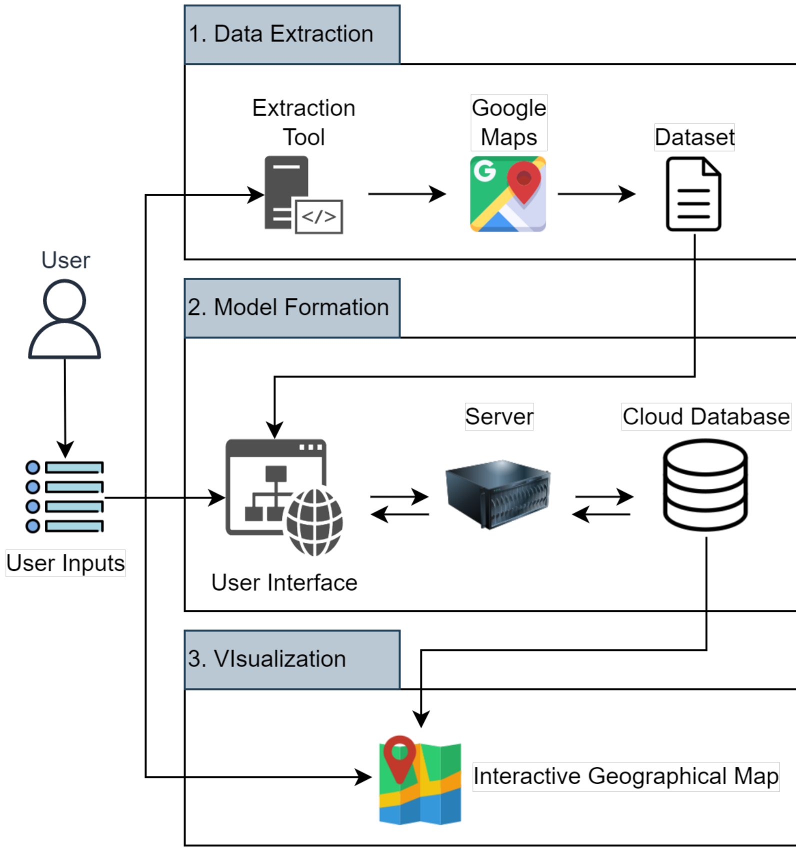


Figure 5.1: System Architecture of the Proposed RegTraffic Simulator

5.2 System Architecture

RegTraffic consists of three core system components that together complete the simulator system. These core components are traffic data extraction, model formation and visualization as shown in Figure 5.1.

The first component of the RegTraffic is traffic data extraction which is described in chapter 3. In this process, a user selects the starting point and ending point of the route of interest and specifies the time range. The traffic data extraction tool gathers time series information of the “congestion index” of that road segment every 15 minutes throughout the time range from Google Maps. The congestion index is defined by the average speed of that road segment in terms of kilometers per hour. At the end of the process, the tool auto-generates a time series dataset that has a unique name as provided by the user when adding a spatial feature. RegTraffic stores the dataset in CSV format for convenient data reading and manipulation in the regression formation step.

The second component of the RegTraffic is model formation which is described in chapter 4. The proposed novel Bayesian linear regression approach combines both spatial and temporal features to form a spatiotemporal traffic prediction model. The approach describes a novel exploratory data analysis technique to extract temporal features from time series data of spatial features. The backend of the RegTraffic consists of a server and cloud database. The server stores the time series dataset of spatiotemporal features through file processing and forms a Bayesian linear regression model based on user inputs. The cloud database is used to store metadata of time series dataset, measurements like congestion coefficients, and regression model parameters including intercept and regression coefficients.

The third component of the RegTraffic is a visualization interface where RegTraffic provides a geographical map based interactive visualization interface where the model parameters associated with spatiotemporal components can be integrated with user inputs to predict traffic congestion. The user would be able to input new data observations for both spatial and temporal features and visualize the effect on the area of interest in the visualization panel.

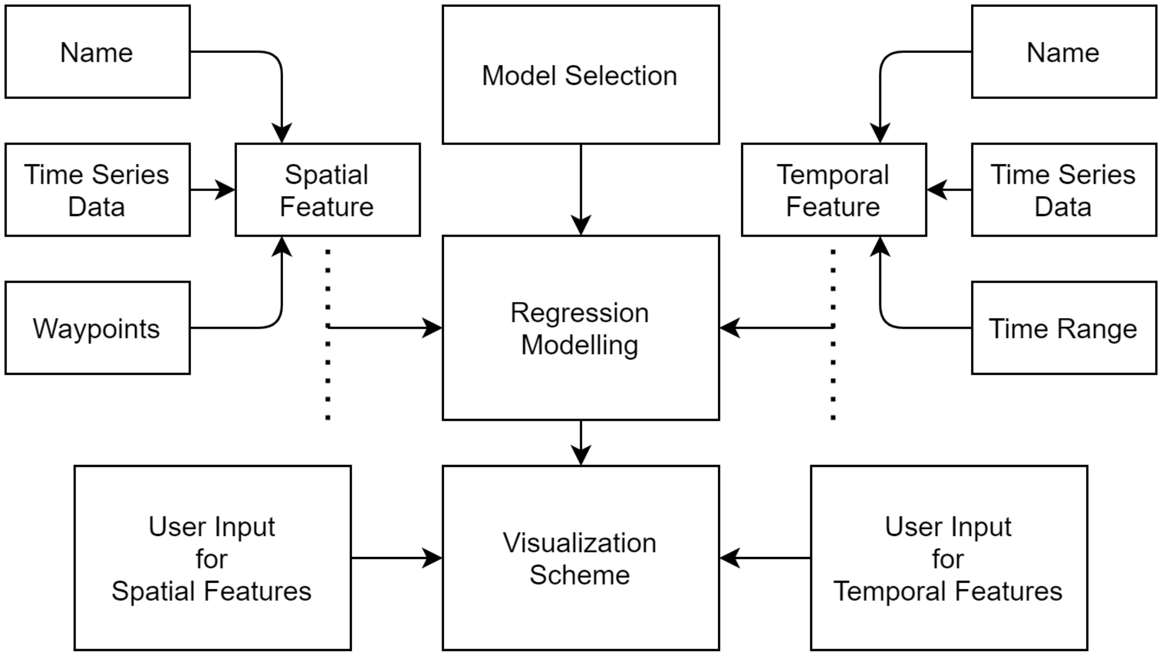


Figure 5.2: Processing Pipeline of the Proposed RegTraffic Simulator

5.3 Processing Pipeline

Figure 5.2 shows a detailed processing pipeline of RegTraffic where users need to provide a unique name and corresponding time series data for both the spatial and temporal feature. For adding a spatial feature in RegTraffic a user needs to provide a set of latitude and longitude as the waypoints of the route of that spatial feature. For adding a temporal feature a user needs to provide a specific time range for that temporal feature. Once the features are added to RegTraffic a user can build a regression model by selecting a set of spatial and temporal features. A user can explicitly determine which spatial feature will be the dependent feature in the regression modeling. Once the model is formed, the model parameters are stored in the cloud database. In the visualization interface, the user can provide new observations for all the independent features in the regression model and visualize the outcome of the dependent feature on the geographical interactive map in real time.

5.4 Modeling and Simulation

Figure 5.3 shows a traffic road junction. In this junction, we can consider a road link as the spatial road feature that is dependent on one or many connected spatial road features. For example, in this junction, we consider the road link $\hat{y}$ as a spatial feature and modelled it as a dependent variable in our regression modeling. We consider the road links x_1^s, x_2^s up to the road link x_n^s as the independent spatial feature in our regression modeling. Note that the dependent road link $\hat{y}$ is going out from the junction whereas the independent road links $x_1^s, x_2^s, ..., x_n^s$ are going towards the junction. Our proposed traffic modeling approach described in chapter 4 indicates that the dependent spatial feature must be an outgoing road link and the independent spatial features must be incoming road links with respect to the orientation of the junction. Our proposed regression model incorporates a set of temporal features that can be extracted from both independent and dependent spatial features through exploratory data analysis. The specific number of temporal features and independent spatial features are arbitrary and dependent on the specific road junction and their orientation.

Here, X_S is defined as a set of independent spatial features $\left\{x_1^s, x_2^s, .., x_{n_s}^s\right\}$ as shown in eq. (5.1).

$$X_S = \left\{x_1^s, x_2^s, .., x_{n_s}^s\right\} \tag{5.1}$$

The cardinality of set X_S is defined as n_s as shown in eq. (5.2).

$$n_s = |X_S| \tag{5.2}$$

Similarly, X_T is a set of temporal features $\left\{x_1^t, x_2^t, .., x_{n_t}^t\right\}$ as shown in eq. (5.3).

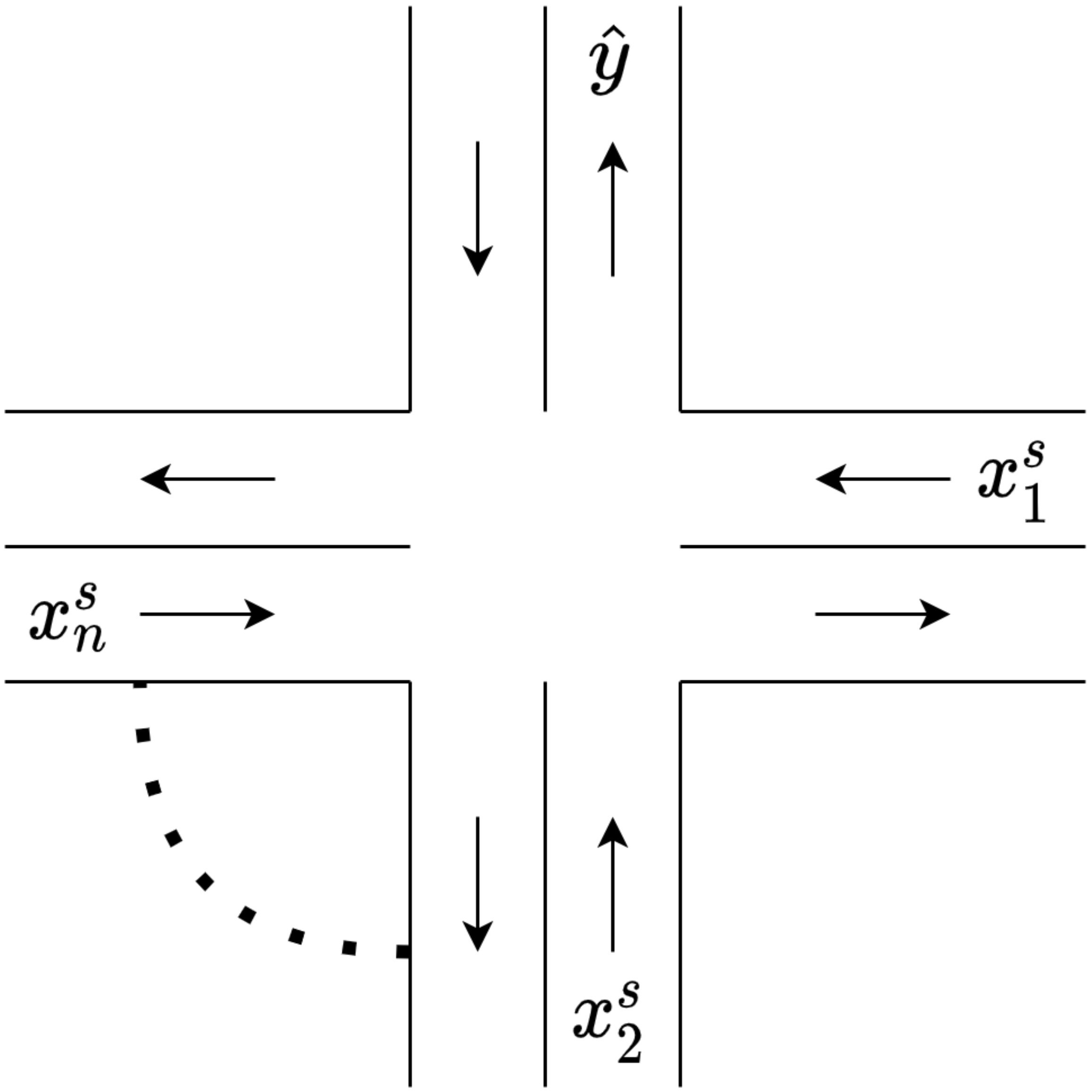


Figure 5.3: Traffic Road Junction

$$X_T = \left\{x_1^t, x_2^t, .., x_{n_t}^t\right\} \tag{5.3}$$

The cardinality of set X_T is defined as n_t as shown in eq. (5.4).

$$n_t = |X_T| \tag{5.4}$$

The set of temporal features X_T is the output of function f which takes in the set of spatial features X_S as an input. The function f is a many to many function that takes in a set of spatial features and conducts exploratory data analysis to extract a set of temporal features as shown in eq. (5.5)

$$X_T = f_{n_s \to n_t}(X_S) \tag{5.5}$$

In this modeling, we define the set X as a union set of the set of temporal features X_T and spatial features X_S as shown in eq. (5.6).

$$X = X_T \cup X_S \tag{5.6}$$

RegTraffic forms a regression model through a linear combination of both temporal and spatial explanatory features in order to explain the dependent spatial feature $\hat{y}$ as shown in eq. (5.7). In this equation, all the independent features are associated with their corresponding regression coefficient. α indicates the bias and ϵ refers to the error term in the regression equation.

$$\hat{y} = \sum_{i=1}^{n_t} \beta_i^t x_i^t + \sum_{i=1}^{n_s} \beta_i^s x_i^s + \alpha + \epsilon \tag{5.7}$$

In the regression eq. (5.7), every explanatory temporal feature from set X_T is associated with a regression coefficient from set β_T as shown in eq. (5.8).

$$\beta_T = \left\{\beta_1^t, \beta_2^t, .., \beta_{n_t}^t\right\} \tag{5.8}$$

Similarly, in the regression eq. (5.7), every explanatory spatial feature from set X_S is associated with a regression coefficient from set β_S as shown in eq. (5.9).

$$\beta_S = \left\{\beta_1^s, \beta_2^s, .., \beta_{n_s}^s\right\} \tag{5.9}$$

Here, β is defined as the union of set β_T and β_S

$$\beta = \beta_T \cup \beta_S \tag{5.10}$$

Eq. (5.11) refers to the Bayes Theorem which is the fundamental building block of Bayesian linear regression. Here, $P(\beta \mid \hat{y}, X)$ is the posterior probability distribution of the model parameters, $P(\hat{y} \mid \beta, X)$ is the likelihood of the data, $P(\beta \mid X)$ is the prior probability of the parameters, and $P(\hat{y} \mid X)$ is the normalization constant. The posterior distribution of the model parameters is proportional to the multiplication of the likelihood of the data and the prior probability of the parameters. A detailed description of the model is described in chapter 4.

$$P(\beta \mid \hat{y}, X) = \frac{P(\hat{y} \mid \beta, X) * P(\beta \mid X)}{P(\hat{y} \mid X)} \tag{5.11}$$

Once the regression model is built, the user can provide new observations for independent spatial features X_S and independent temporal features X_T into the model. Based on the new observation the model incorporates the regression coefficients associated with the explanatory variables and predicts the output for the dependent variable $\hat{y}$. A user can associate an event with a specific value and provide it as an input for any independent spatial feature in the regression equation.

Here, X_E is defined as a set of events $\left\{X_1^E, X_2^E, .., X_{n_E}^E\right\}$ as shown in eq. (5.12).

$$X_E = \left\{X_1^E, X_2^E, .., X_{n_E}^E\right\} \tag{5.12}$$

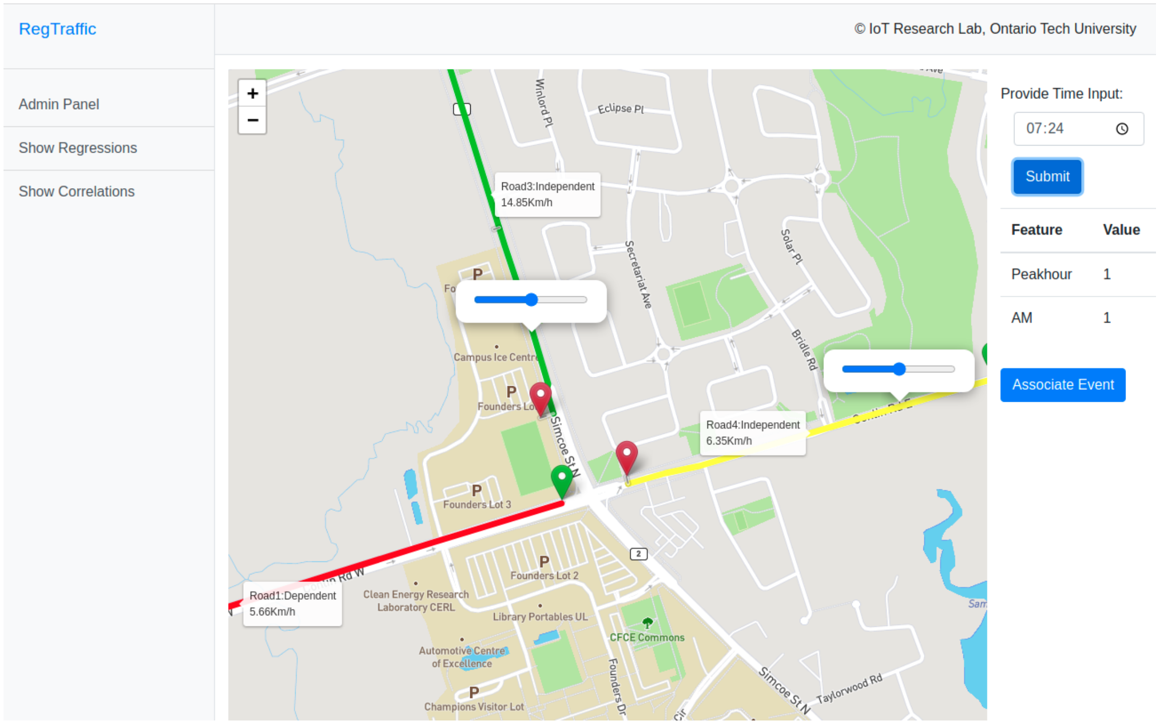
The cardinality of set X_E is defined as n_E as shown in eq. (5.13).

$$n_E = |X_E| \tag{5.13}$$

After event integration, the independent spatial features associated with an event are added into the eq. (5.7). For the independent spatial features associated with an event, we need to replace the values for the set of independent spatial features X_S with the set of values for events X_E as shown in eq. (5.14).

$$\hat{y} = \sum_{i=1}^{n_t} \beta_i^t x_i^t + \sum_{i=1}^{n_s} \beta_i^s x_i^s + \sum_{i=1}^{n_E} \beta_i^E x_i^E + \alpha + \epsilon \tag{5.14}$$

Figure 5.4 describes a sample simulation procedure of a road junction where $Road1$ is a dependent road link and $Road3$ and $Road4$ are independent road links. Therefore, in the corresponding regression model, the $Road1$ is considered as a dependent spatial feature and $Road3$ and $Road4$ are considered as independent spatial features. Based on the spatial features two new temporal features are extracted which are $Peakhour$ and Am. RegTraffic shows the location of the road links on an interactive geographical map where the user can provide new observations for independent road links and temporal features to predict the outcome of the dependent road link. As shown in the figure, the user sets the congestion



Provide Time Input: 07:24

Feature	Value
Peakhour	1
AM	1

Figure 5.4: Regression Analysis in RegTraffic Simulator

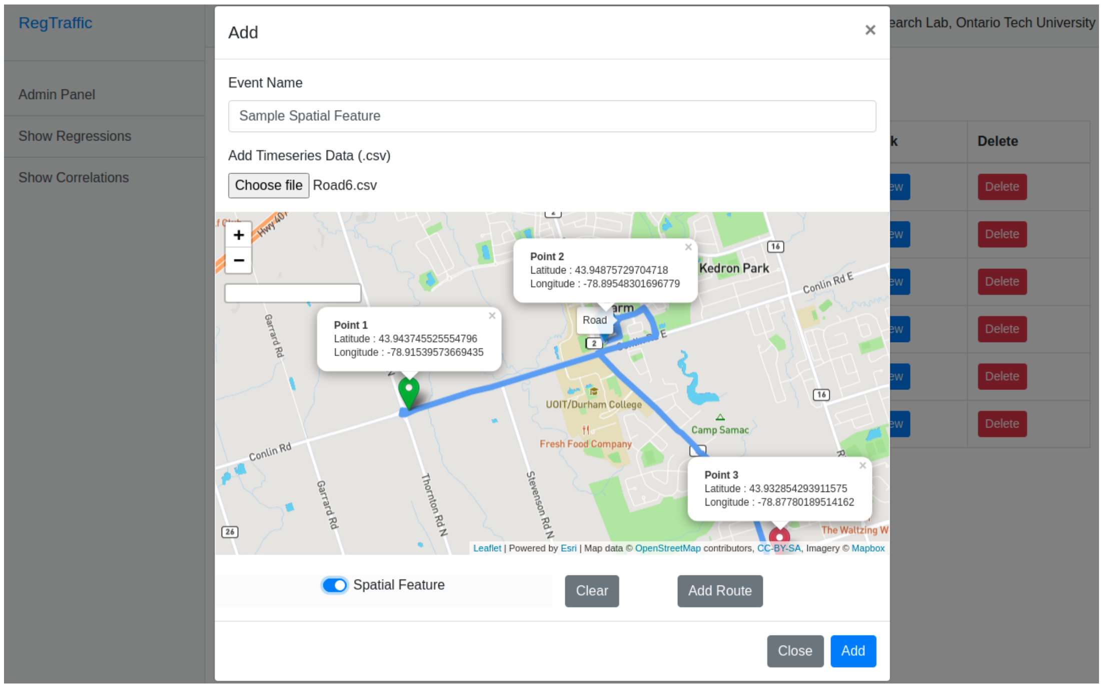
index of $Road3$ and $Road4$ as 14.85 kilometers per hour and 6.35 kilometers per hour respectively. The user also needs to provide the specific time as an input for the temporal features $Peakhour$ and AM. RegTraffic calculates the value for the temporal features from the time input provided by the user and incorporates these values along with the input values for independent spatial features to predict the congestion index of dependent road link $Road1$. Based on the input values provided by the user, RegTraffic predicts the congestion index of the road link $Road1$ which is 5.66 kilometers per hour in this case.

5.5 User Interface

5.5.1 Spatial Feature Interface

RegTraffic has a separate interface for adding a spatial feature for regression analysis that takes the name of the spatial feature along with a time series dataset in CSV format as an input dataset for that specific spatial feature for regression analysis as shown in Figure 5.5. To keep track of different spatial features, RegTraffic renames the dataset file as the feature name specified by the user. The interface provides a dedicated panel to select the waypoints of that spatial feature. RegTraffic requires every spatial feature to have spatial components like latitude and longitude. Using a switch button available on the top right corner, the user can activate the panel to input the spatial components of the spatial feature. Once the switch button is activated, a user can click on a specific position on an interactive map to complete a route. RegTraffic accepts latitude and longitude of not only starting and ending points of a road segment but also allows a user to add more points in between the starting and ending points. In this way, a user would be able to add waypoints to complete a route that can have multiple destinations as precisely as possible.

Every point on the map is marked with a marker. The starting point is highlighted with a green marker, the ending point is highlighted with a red marker. Any other point except the starting and ending point is marked with a regular blue marker. Once a marker is added, RegTraffic stores the latitude and longitude information of that marker and shows it on top of the marker as a bounded tooltip. A user would be able to click on top of the marker to show or hide the tooltip. There is a button at the right side of the map panel to clear any unwanted points that are added to the map unintentionally. Once all the points are marked the user can complete the route by pressing the button called ”Add Route” that is situated under the clear button at the right panel. RegTraffic generates a light-blue poly-line on top


RegTraffic

Admin Panel

Show Regressions

Show Correlations

Add

Event Name

Sample Spatial Feature

Add Timeseries Data (.csv)

Choose file Road6.csv

Point 1
Latitude : 43.943745525554796
Longitude : -78.91539573669435

Point 2
Latitude : 43.94875729704718
Longitude : -78.89548301696779

Point 3
Latitude : 43.932854293911575
Longitude : -78.87780189514162

Spatial Feature

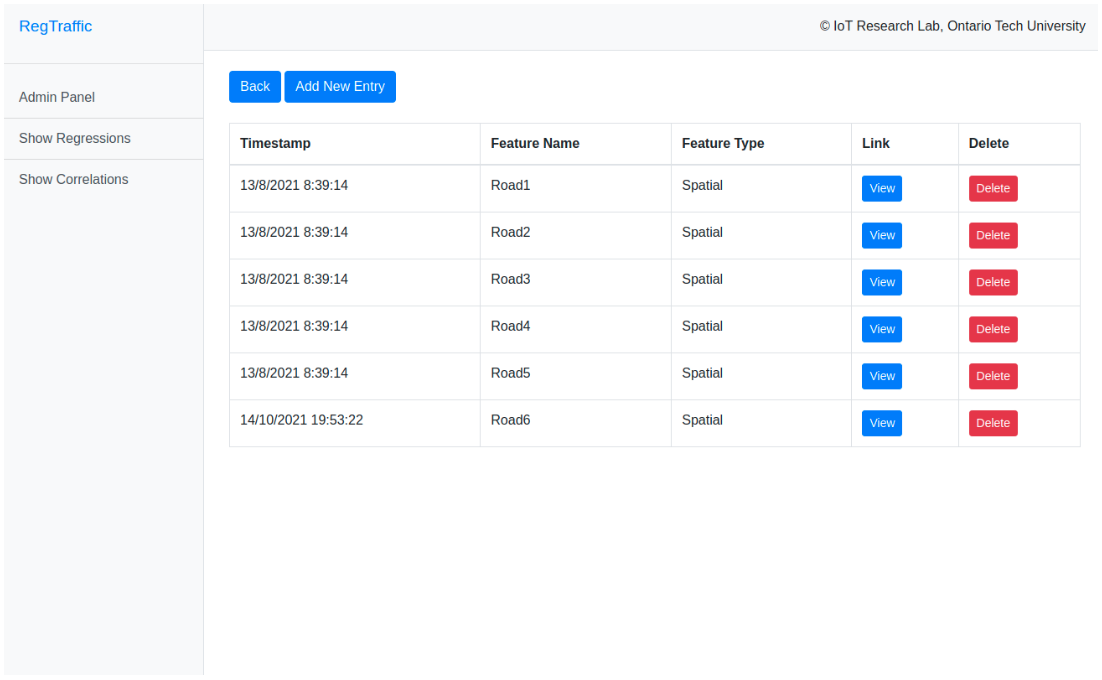
Figure 5.5: Spatial Feature Creation in RegTraffic Simulator

of the route to highlight it to the user. Along with the poly-line, it also added the name of the feature on top of the poly-line. Once all the inputs are completed, the user can add all the information of the spatial feature into RegTraffic.

RegTraffic has a dashboard for the spatial feature where all the spatial features are listed along with their timestamp, name, and type as shown in Figure 5.6. The dashboard also provides an option to delete a spatial feature from RegTraffic. Along with this information, every entry has a link to view the times series data of the spatial feature as shown in Figure 5.7.

5.5.2 Temporal Feature Interface

Just like adding the spatial feature, RegTraffic has a separate interface to add a temporal feature. RegTraffic takes the name of the temporal feature along with the time series dataset



RegTraffic

© IoT Research Lab, Ontario Tech University

Admin Panel

Show Regressions

Show Correlations

Back | Add New Entry

Timestamp	Feature Name	Feature Type	Link	Delete
13/8/2021 8:39:14	Road1	Spatial	View	Delete
13/8/2021 8:39:14	Road2	Spatial	View	Delete
13/8/2021 8:39:14	Road3	Spatial	View	Delete
13/8/2021 8:39:14	Road4	Spatial	View	Delete
13/8/2021 8:39:14	Road5	Spatial	View	Delete
14/10/2021 19:53:22	Road6	Spatial	View	Delete

Figure 5.6: Spatial Feature Dashboard in RegTraffic Simulator

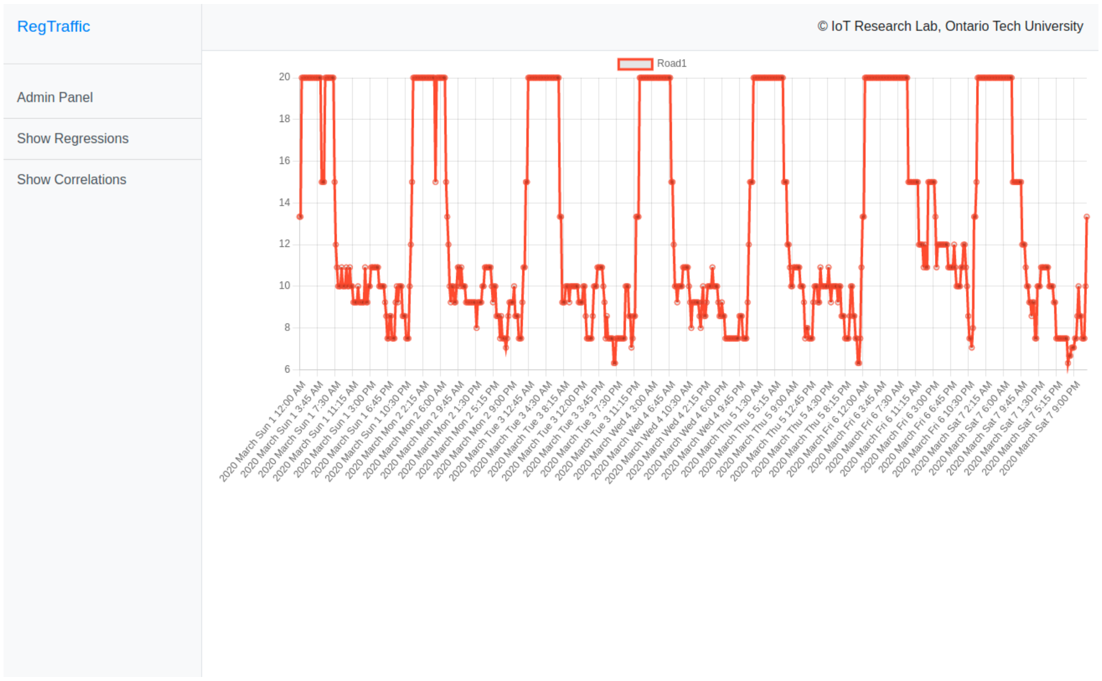


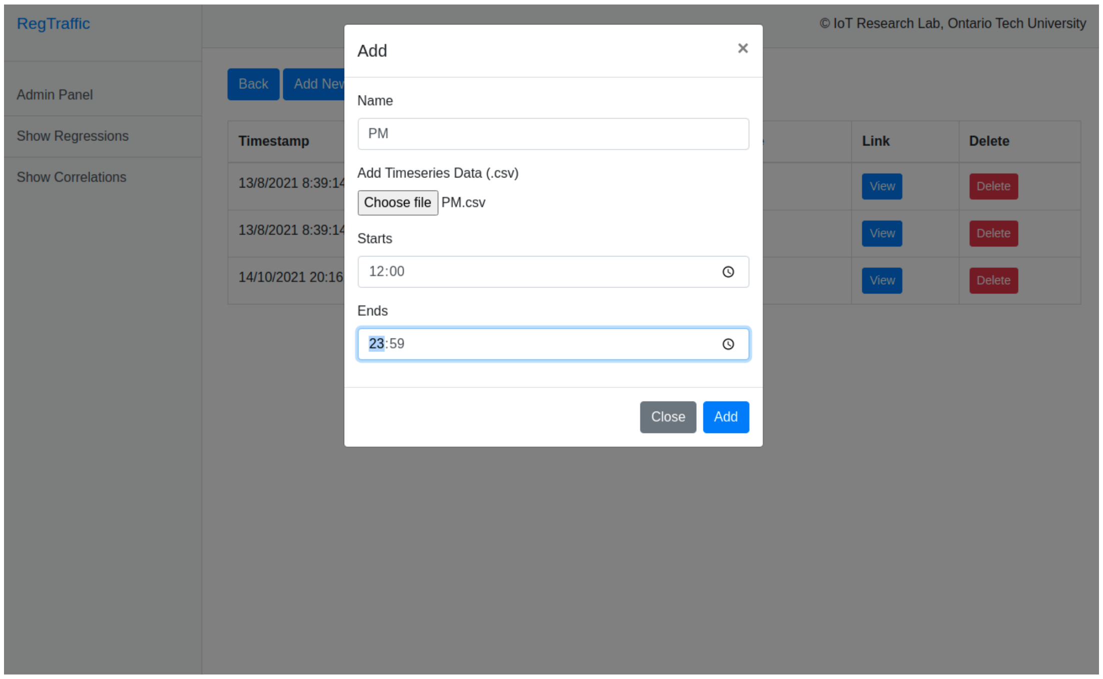
Figure 5.7: Spatial Feature Description in RegTraffic Simulator

of the temporal feature in CSV format. RegTraffic renames the dataset file with the name of the temporal feature to avoid multiple datasets with the same name which are representing different temporal features. RegTraffic is designed to take only categorical temporal features like peak hour and meridiem. To calculate the time range of these categorical features along with the time series, RegTraffic takes two-time inputs from the user. These time inputs are the starting time and ending time of a temporal feature. The time inputs are added in a 24-hour format. To simplify the time inputs, RegTraffic only lets a user choose a specific hour and minutes for a temporal feature. There is a button at the bottom to add all this information for a temporal feature as shown in Figure 5.8.

RegTraffic has a dashboard for the temporal feature where all the temporal features are listed along with their timestamp, name, and type as shown in Figure 5.9. The dashboard also provides an option to delete a temporal featureRegTraffic. Along with this information, every entry has a link to view the times series data of the spatial feature as shown in Figure 5.10.

5.5.3 Correlation Analysis Interface

The system provides a panel to select a set of features including both spatial and temporal features and generate a correlation index among them. The system further provides a feature to select the method for calculating the correlation index. There are three types of methods that are made available to the system to calculate the correlation index which are Pearson Correlation, Spearman Correlation and Kendall Tau Correlation. Once a user selects a set of features, then the user needs to select the correlation calculation method. A user can denote the calculation of correlation of a specific set of features and methods with a unique name and save it based on its name as shown in Figure 5.11.



Name: PM

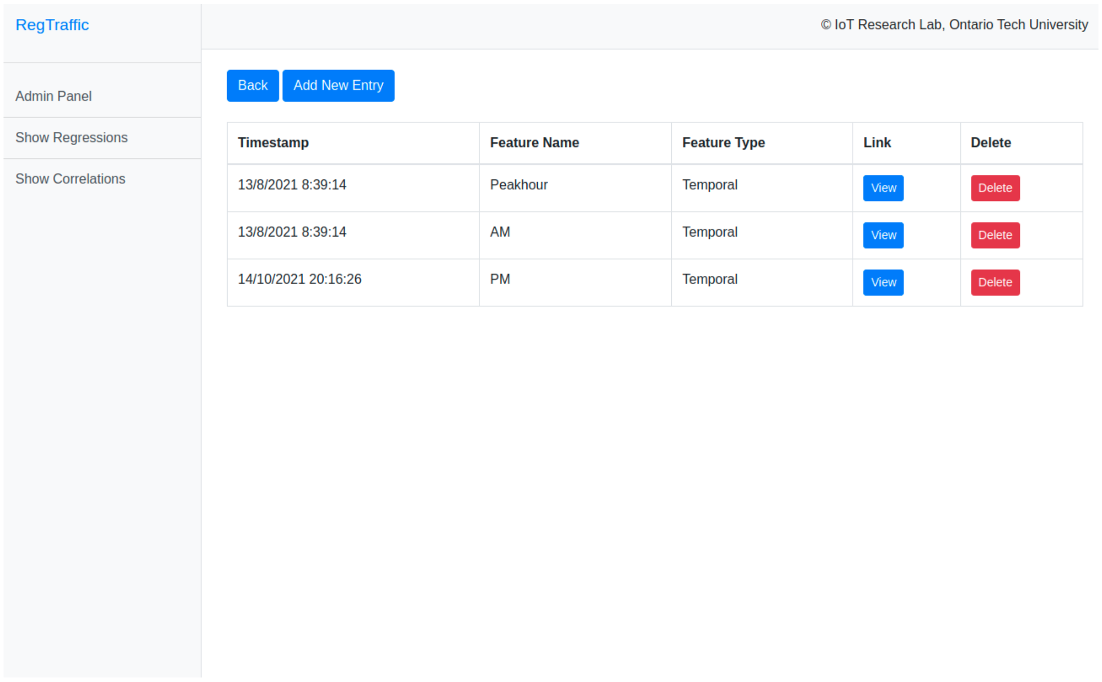
Add Timeseries Data (.csv): Choose file PM.csv

Starts: 12:00

Ends: 23:59

Timestamp		Link	Delete
13/8/2021 8:39:14		View	Delete
13/8/2021 8:39:14		View	Delete
14/10/2021 20:16		View	Delete

Figure 5.8: Temporal Feature Creation in RegTraffic Simulator



Timestamp	Feature Name	Feature Type	Link	Delete
13/8/2021 8:39:14	Peakhour	Temporal	View	Delete
13/8/2021 8:39:14	AM	Temporal	View	Delete
14/10/2021 20:16:26	PM	Temporal	View	Delete

Figure 5.9: Temporal Feature Dashboard in RegTraffic Simulator

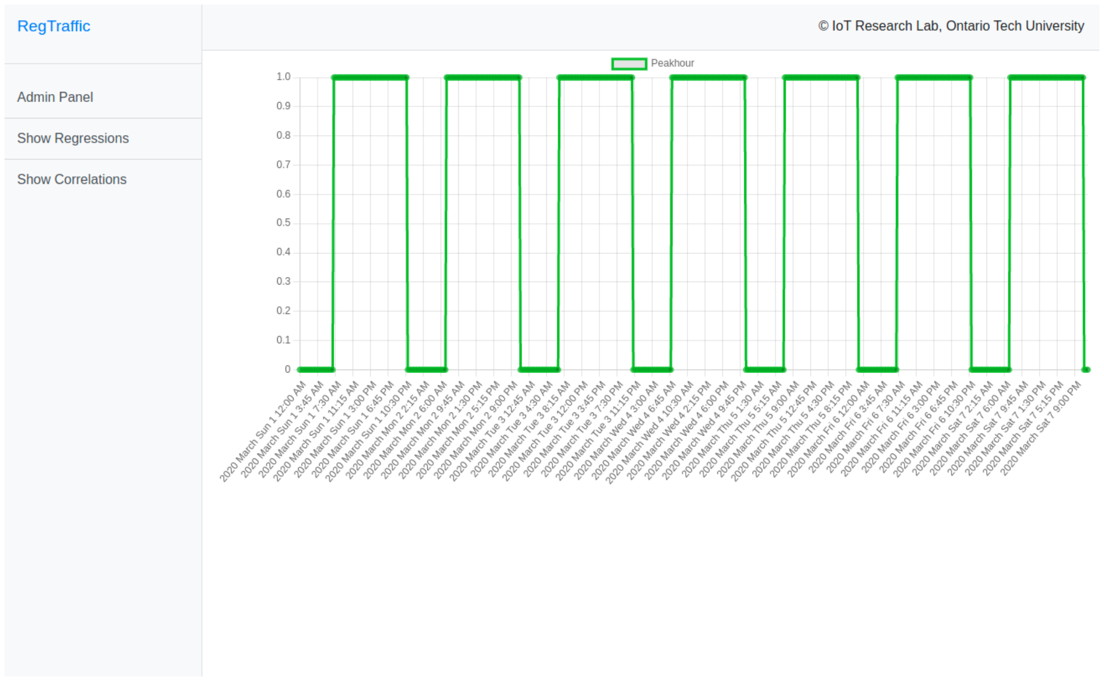
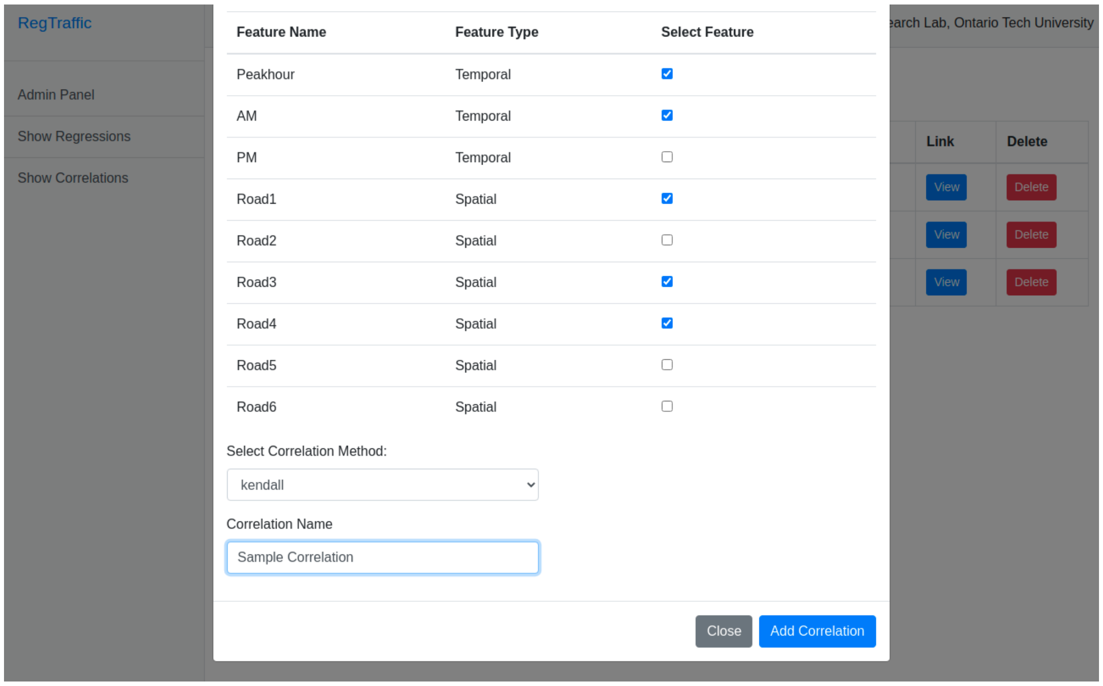


Figure 5.10: Temporal Feature Description in RegTraffic Simulator

The system has a dashboard for correlation where all regression entities are listed along with their timestamp, name, and correlation method as shown in Figure 5.12. The dashboard also provides an option to delete a regression entity from RegTraffic. Along with this information, every entry has a link to explore the regression entity in detail as shown in Figure 5.19.

5.5.4 Event Interface

One of the core features of RegTraffic is event incorporation with the regression model. RegTraffic provides an interface where a user can create an event by providing a unique name of the event, the corresponding traffic speed of a road segment once that event occurs and the time range of the event as shown in Figure 5.14. Once the event is created it is stored in the database. RegTraffic has a dashboard where all event entities are listed along

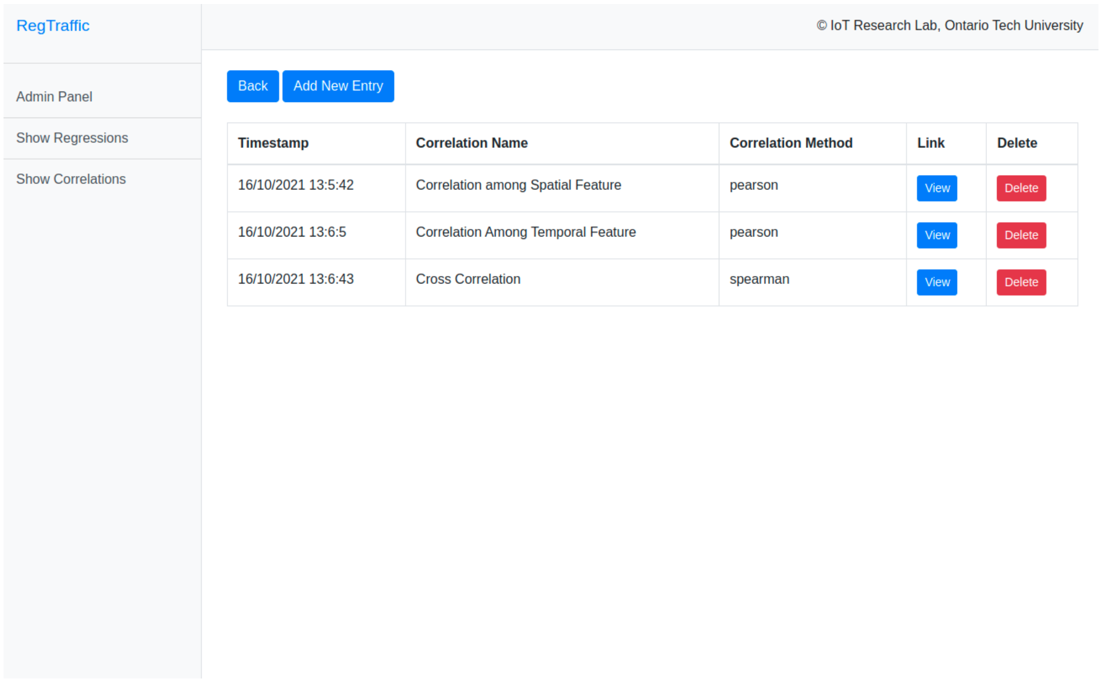


Feature Name	Feature Type	Select Feature
Peakhour	Temporal	☑
AM	Temporal	☑
PM	Temporal	☐
Road1	Spatial	☑
Road2	Spatial	☐
Road3	Spatial	☑
Road4	Spatial	☑
Road5	Spatial	☐
Road6	Spatial	☐

Select Correlation Method: kendall

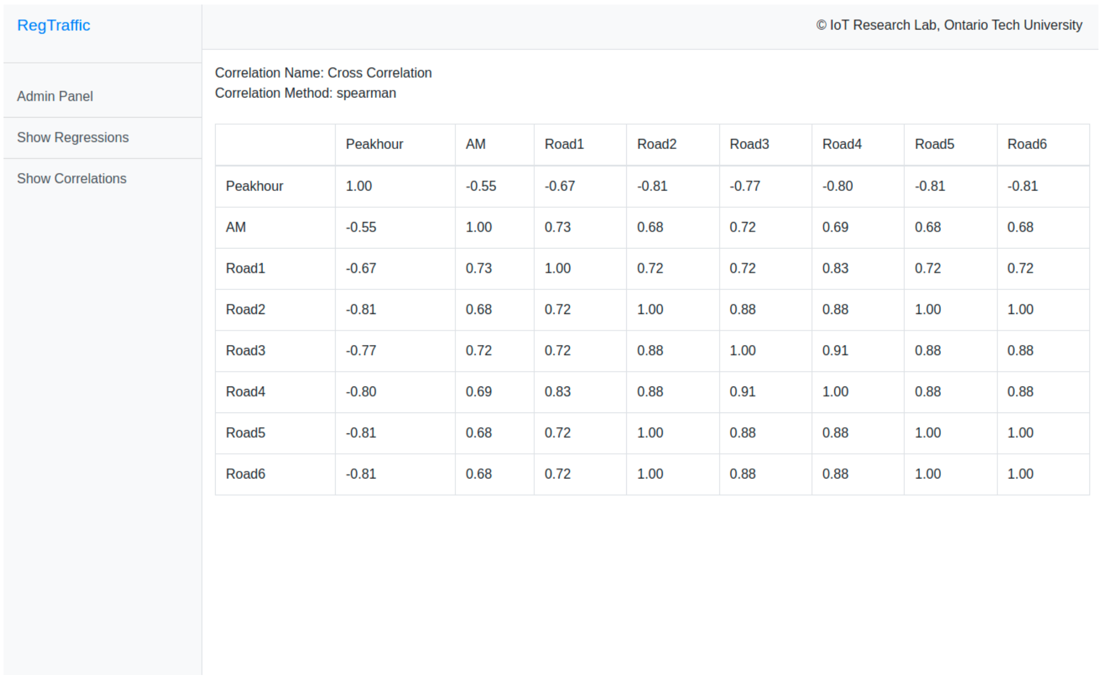
Correlation Name: Sample Correlation

Figure 5.11: Correlation Creation in RegTraffic Simulator



Timestamp	Correlation Name	Correlation Method	Link	Delete
16/10/2021 13:5:42	Correlation among Spatial Feature	pearson	View	Delete
16/10/2021 13:6:5	Correlation Among Temporal Feature	pearson	View	Delete
16/10/2021 13:6:43	Cross Correlation	spearman	View	Delete

Figure 5.12: Correlation Dashboard in RegTraffic Simulator



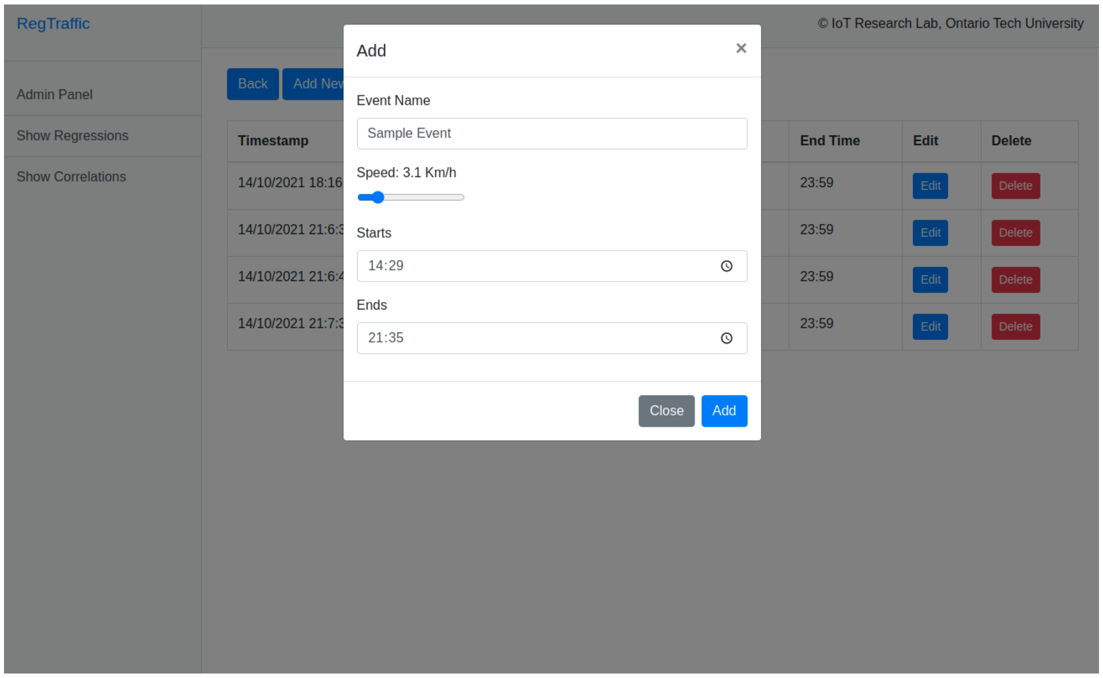
Correlation Name: Cross Correlation
Correlation Method: spearman

	Peakhour	AM	Road1	Road2	Road3	Road4	Road5	Road6
Peakhour	1.00	-0.55	-0.67	-0.81	-0.77	-0.80	-0.81	-0.81
AM	-0.55	1.00	0.73	0.68	0.72	0.69	0.68	0.68
Road1	-0.67	0.73	1.00	0.72	0.72	0.83	0.72	0.72
Road2	-0.81	0.68	0.72	1.00	0.88	0.88	1.00	1.00
Road3	-0.77	0.72	0.72	0.88	1.00	0.91	0.88	0.88
Road4	-0.80	0.69	0.83	0.88	0.91	1.00	0.88	0.88
Road5	-0.81	0.68	0.72	1.00	0.88	0.88	1.00	1.00
Road6	-0.81	0.68	0.72	1.00	0.88	0.88	1.00	1.00

Figure 5.13: Correlation Description in RegTraffic Simulator

with their timestamp, name, speed value and time range as shown in Figure 5.15. The dashboard also provides an option to edit and delete an event entity from RegTraffic.

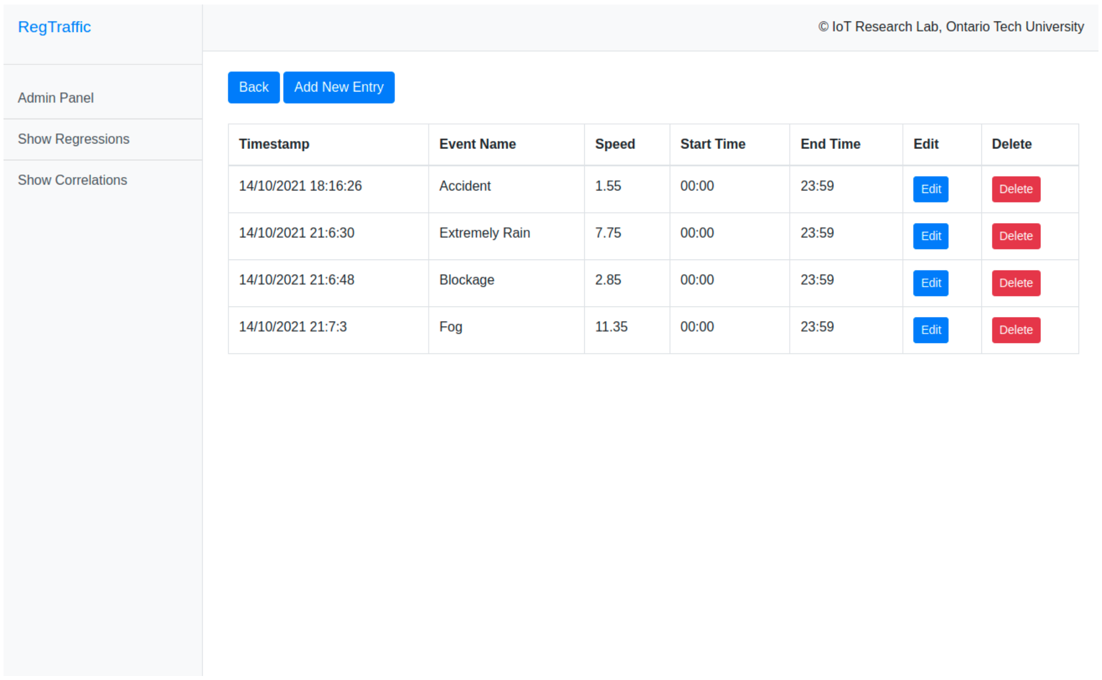
In the simulation and visualization interface, a user would be able to associate an event with an independent spatial feature by using a modal menu as shown in Figure 5.16. From the modal, a user would be able to see some key information of regression modeling like coefficient value, dependency, type of feature and can associate an event with a spatial feature by choosing that event from a select option. If there is no need to associate any event with a specific spatial feature, the user can provide 'None'. Once these inputs are filled and the user submits the simulation inputs, RegTraffic automatically incorporates the effect of these events on the corresponding road segments.



Event Name: Sample Event
Speed: 3.1 Km/h
Starts: 14:29
Ends: 21:35

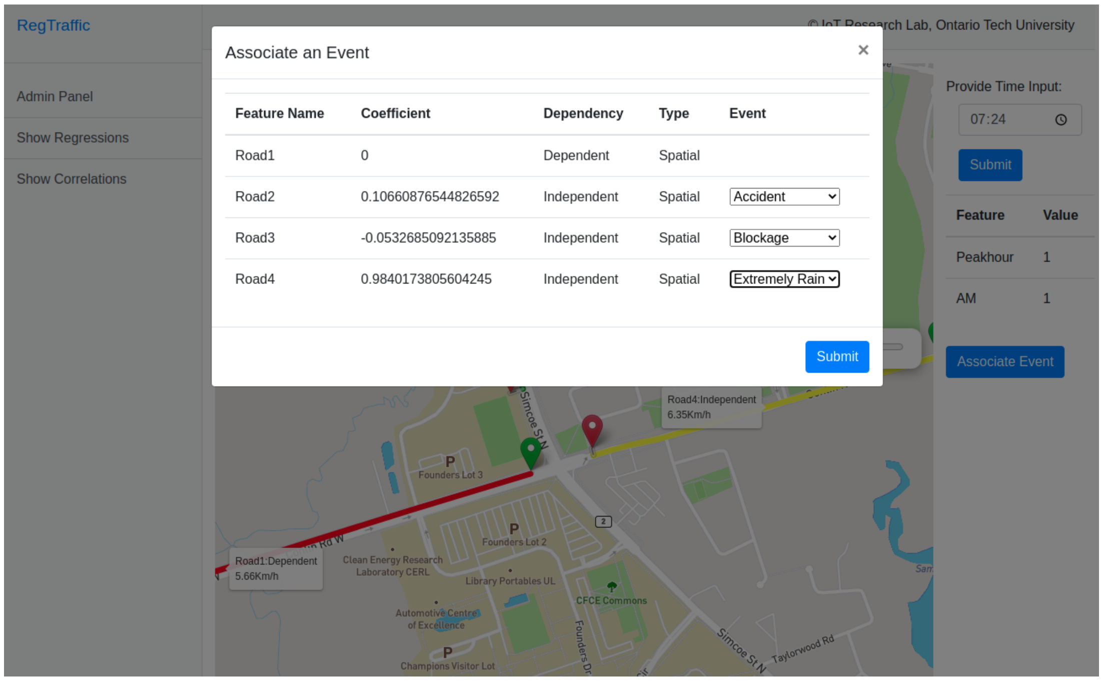
Timestamp	End Time	Edit	Delete
14/10/2021 18:16	23:59	Edit	Delete
14/10/2021 21:6:3	23:59	Edit	Delete
14/10/2021 21:6:4	23:59	Edit	Delete
14/10/2021 21:7:3	23:59	Edit	Delete

Figure 5.14: Event Creation in RegTraffic Simulator



Timestamp	Event Name	Speed	Start Time	End Time	Edit	Delete
14/10/2021 18:16:26	Accident	1.55	00:00	23:59	Edit	Delete
14/10/2021 21:6:30	Extremely Rain	7.75	00:00	23:59	Edit	Delete
14/10/2021 21:6:48	Blockage	2.85	00:00	23:59	Edit	Delete
14/10/2021 21:7:3	Fog	11.35	00:00	23:59	Edit	Delete

Figure 5.15: Event Dashboard in RegTraffic Simulator



Associate an Event

Feature Name	Coefficient	Dependency	Type	Event
Road1	0	Dependent	Spatial	
Road2	0.10660876544826592	Independent	Spatial	Accident
Road3	-0.0532685092135885	Independent	Spatial	Blockage
Road4	0.9840173805604245	Independent	Spatial	Extremely Rain

Feature	Value
Peakhour	1
AM	1

Figure 5.16: Event Association in RegTraffic Simulator

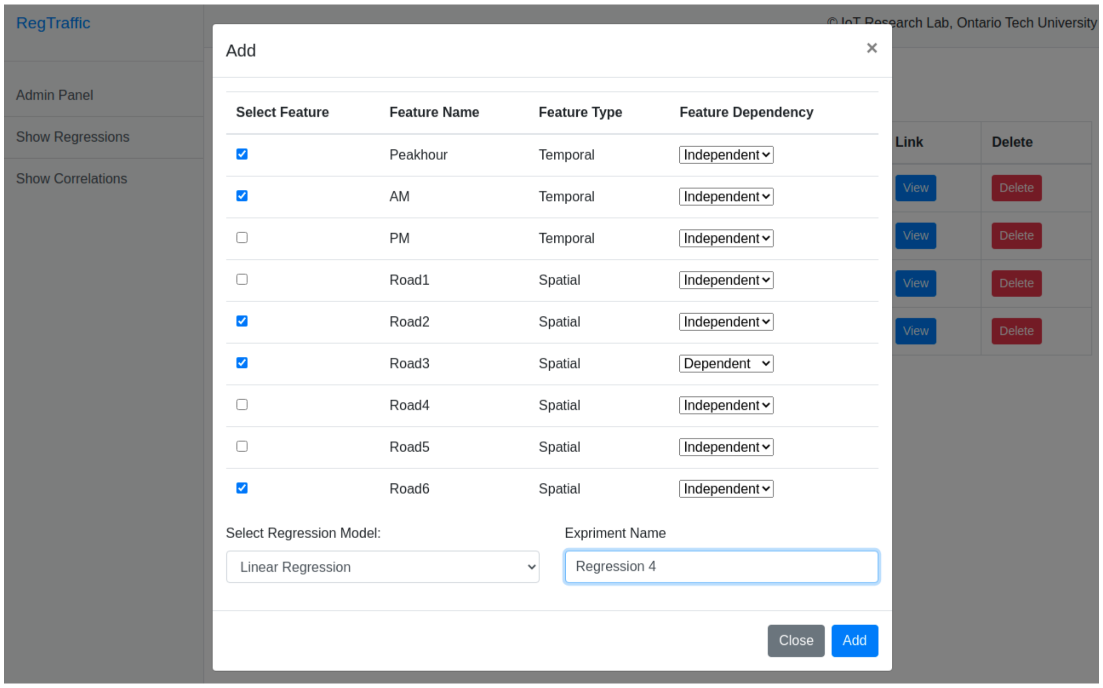
5.5.5 Regression Analysis Interface

The regression analysis interface provides all the necessary options for a user to form a regression entity. A regression entity has a dependent variable and a set of independent variables that would explain the dependent variable. RegTraffic lists all the spatial and temporal features in this interface and lets the user choose which feature to select as a dependent feature. Since the temporal components do not depend on any other features, RegTraffic does not allow a user to choose any temporal feature as a dependent feature and keeps the dependency as independent by default. However, for spatial features, it provides a select box to select the feature as a dependent or independent feature. Every set of dependent and independent features that form a regression need to have a unique identity. For that reason, RegTraffic takes the unique name of a regression from the user as well. Once the user provides all this information a new regression entity is created and saved in RegTraffic for further processing as shown in Figure 5.17.

RegTraffic has a dashboard for regression where all regression entities are listed along with their timestamp, name, and regression model as shown in Figure 5.18. The dashboard also provides an option to delete a regression entity from RegTraffic. Along with this information, every entry has a link to explore the regression entity in detail as shown in Figure 5.19.

One of the core features of RegTraffic is the interface for simulation and visualization as shown in Figure 5.19. This interface provides a panel to simulate and visualize a regression entity that is created by the user earlier. Once a regression entity is provided by the user, RegTraffic checks in the corresponding information and forms a regression model using the time series dataset of both the dependent and independent features of the regression entity. To build the regression model RegTraffic incorporates both the spatial and temporal features. The panel lists all the spatial features on top of the interactive map using polylines. Each polyline has a slider button as a tooltip on top of it. Using the slider a user would be able to change the value of that spatial feature. RegTraffic takes the value as an input for that independent feature and recalculates the value of the dependent spatial feature based on the bias and associated regression coefficients of the independent spatial features. The bias and coefficients are generated from a regression model. RegTraffic updates the value and changes the color of modified spatial features. The color is determined based on a predefined color range set. Since the value of the dependent spatial features depends on the input values of the independent spatial features, RegTraffic does not add any slider button for the dependent spatial features.

Instead of taking input for every temporal independent feature, RegTraffic only takes a time component from the user. Then it generates corresponding input values for every



Select Feature	Feature Name	Feature Type	Feature Dependency
✓	Peakhour	Temporal	Independent
☐	AM	Temporal	Independent
☐	PM	Temporal	Independent
☐	Road1	Spatial	Independent
✓	Road2	Spatial	Independent
✓	Road3	Spatial	Dependent
☐	Road4	Spatial	Independent
☐	Road5	Spatial	Independent
✓	Road6	Spatial	Independent

Select Regression Model: Linear Regression

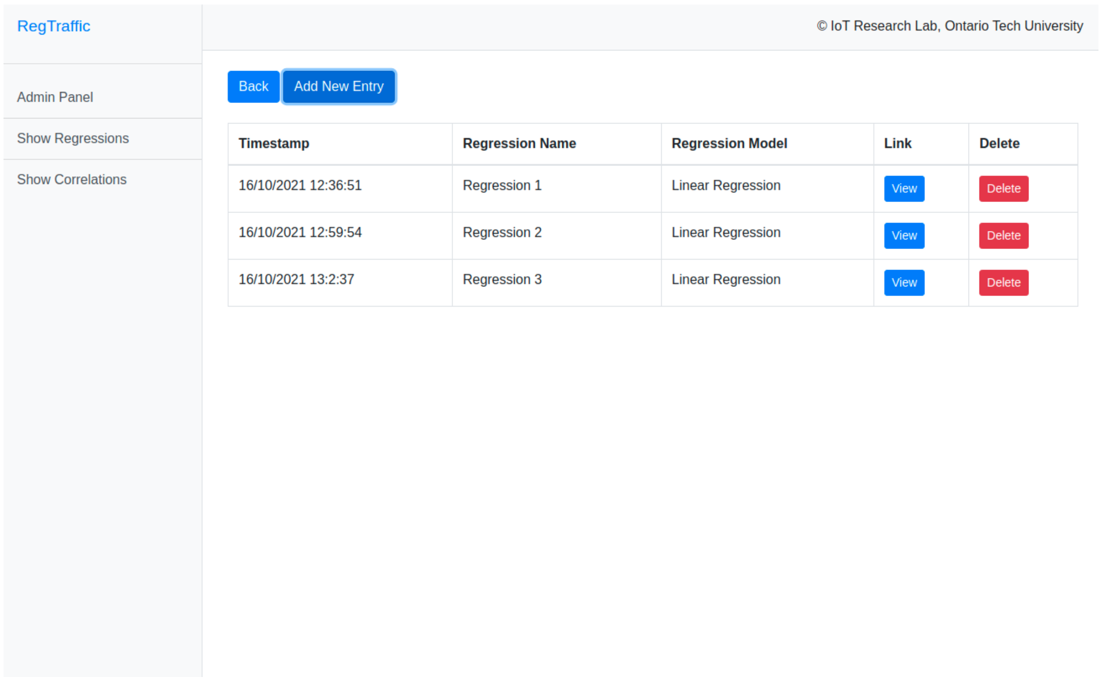
Expriment Name: Regression 4

Figure 5.17: Regression Creation in RegTraffic Simulator

temporal feature in a regression entity based on the time range of these features provided by the user during regression formation. RegTraffic provides a dynamic table at the right side of the visualization panel to list all the temporal independent features. Once the user enters the time component as an input to the regression model, RegTraffic automatically converts the corresponding input values for each one of the temporal independent features and lists them in that dynamic table so that the user can see the input values for these temporal features.

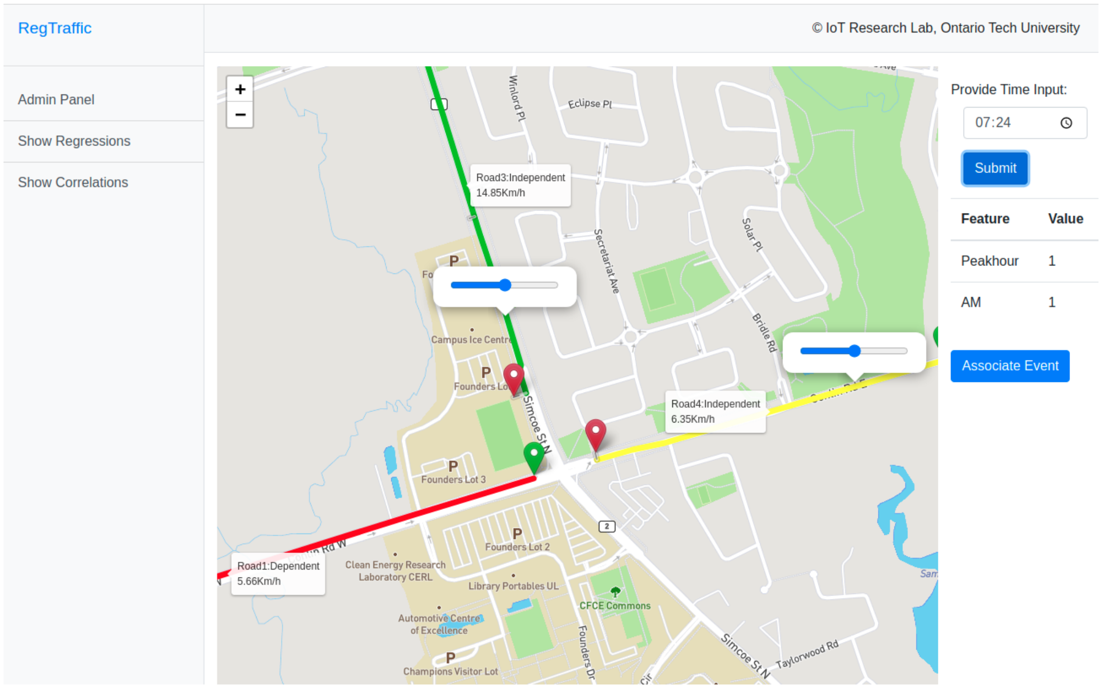
5.6 Technical Description

A traffic simulator tool should be assessed on open accessibility, operating system portability, the usability of the graphical user interface, documentation, interpretation of simulation output and performance in terms of hardware [49]. As a result, we used state of the art



Timestamp	Regression Name	Regression Model	Link	Delete
16/10/2021 12:36:51	Regression 1	Linear Regression	View	Delete
16/10/2021 12:59:54	Regression 2	Linear Regression	View	Delete
16/10/2021 13:2:37	Regression 3	Linear Regression	View	Delete

Figure 5.18: Regression Dashboard in RegTraffic Simulator



Feature	Value
Peakhour	1
AM	1

Figure 5.19: Regression Analysis in RegTraffic Simulator

technologies and adopted the best software engineering and design practices to develop our system. RegTraffic is built on using varieties of technical tools, packages, frameworks, libraries, operating systems and databases. Some of the major sides of RegTraffic are server, virtual machine, back end, front end, interactive maps, routing machine, template engine, database, regression modeling, file processing and using a tool to manage all these packages. We have checked the compatibility issues of all the packages with each other and avoided any version that has a conflict with others. We tried to avoid any unstable version of these packages to build RegTraffic. RegTraffic adopts best software development practices including the agile development process. Table 5.1 shows all the sides of the RegTraffic and lists different packages, libraries and tools under them along with the version and a short description.

Table 5.1: Technical Description of RegTraffic Traffic Simulator

Side	**Name**	**Version**	**Description**
Server	Linux	Ubuntu 20.04	Creates server to host RegTraffic
Virtual Machine	Virtualenv	3.8.x	Creates virtual environment
Back end	Python	3.8	Core programming language
	Flask	2.0	MVC Web framework
Front end	HTML	5	Markup language
	CSS	3	Provides Style
	Bootstrap	4.0	CSS Framework
	Jquery	3.5.1	Traverse DOM Tree
Maps	Leaflet.js	2.5.1	Generates Interactive Map
Routing Machine	Leaflet Routing Machine	2.5.1	Generates Route from Waypoints
Template Engine	Jinja2	3.0	Web template for python
Database	MongoDB	Cloud Atlas	Store and retrieve data
Regression	pandas	1.3.1	Manipulate and analysis data
	numpy	1.21	Handles multi dimensional arrays
	scipy	1.7	Provides scientific computing
	matplotlib	3.4.2	Plot graphs/chars
	sklearn	0.24	Provides regression algorithms
File Processing	Papaparse.js	5.0.2	Parse CSV files
Package Manager	Pip	21.2.1	Package management system

5.7 Summary

In this chapter, we illustrated the development and analysis of a web-based traffic simulator called RegTraffic to simulate and visualize road traffic congestion. RegTraffic builds regression analysis by incorporating both spatial and temporal time series data. It provides a dynamic visualization interface for a user to provide new observations for independent features of a regression model. The visualization interface shows the prediction of traffic congestion based on user input in real-time on a geographical interactive map. We described the functionalities and features of the RegTraffic including feature integration, user interface, regression model formation, and visualization. We also listed the in-depth technical description of the RegTraffic traffic simulator.

Chapter 6

Conclusion

6.1 Discussions

In this thesis, we have provided a framework that consists of three major components. First of all, we have developed a tool to automate the process of collecting time-series traffic information within a specific range of time from Google Maps in a well-organized and usable format that can be directly used in a traffic model without further preprocessing. We have introduced a traffic congestion index to define the congestion level of a road segment for a specific time. Secondly, using the tool we have collected traffic data to build a Bayesian Linear Regression model. To develop the model, we have extracted temporal components and incorporated them with spatial components to build a spatiotemporal traffic congestion prediction model that provides model variability and meaningful interpretation. Lastly, we have developed a software system that provides a map based interactive platform to visualize the outcome of the linear regression model. We hope that our traffic prediction framework would help both the industry and research community to further investigate the specific cause for traffic congestion and open a door for future contributions in these domains of research.

Google Maps collects real-time traffic data with high precision through GPS location-based crowdsourcing. The Google Maps Distance Matrix API provides access to this traffic data with paid subscriptions. Other map services and third-party online tools either provide less accurate traffic data, limited features, and/or are proprietary. In this thesis, we designed and developed a framework for traffic modeling and analysis based on historical traffic data. Our approach consists of three major components: (1) a data scraping tool to extract traffic data from Google Maps, (2) an accurate and robust traffic model to estimate the congestion level at certain road segments, (3) a visualization tool to show the results on interactive maps. To accelerate the research on the traffic domain, we have developed a lightweight tool to automate the process of formatting and extracting publicly available traffic data from the Google Maps web interface by leveraging web scraping. Our developed tool is lightweight, fast, and open source. Performance evaluation of our tool shows that it is highly accurate and efficient and can be used on any latest version of a web browser. However, the development and performance analysis of such a tool involves some key limitations and challenges as it is based on web scraping and largely depends on the Google Map web service. Our proposed approach does not violate any data copyrights, policies or regulations of Google Maps as it only automates the accumulation of publicly available traffic data from the Google Maps user-side web interface. We believe that this open-source tool accelerates research using Google Maps traffic data and encourages the research community to promote open data sharing for traffic-related research and developments.

We also introduced a novel Bayesian linear regression approach that can predict traffic congestion without any significant accuracy loss. This approach works well with smaller or missing observations by leveraging prior information, yet provides the same interpretability and some advantages over state of the art frequentist approaches. We found that exploratory data analysis is an effective tool for feature selection in time-series traffic data to overcome the drawbacks of principal component analysis. The model parameters estimated from Markov Chain Monte Carlo sampling can quantify the model uncertainty. This model could be adopted with complex traffic scenarios where traffic data observations are non-linear and interrupted. The proposed probabilistic inference based regression approach is scalable and can handle data uncertainty.

Lastly, we developed RegTraffic, a web-based traffic simulator system to simulate and visualize road traffic congestion. RegTraffic takes both spatial and temporal time series data as an input and conducts regression analysis based on user preferences. It plugs in regression coefficients with a dynamic visualization scheme where a user can visualize and predict traffic congestion in real-time. Although currently the system only supports regression-based traffic analysis and prediction, it is built as scalable and can be updated to accommodate advanced traffic modeling techniques like artificial neural networks, Long Short Term Memory, Multivariate Multi-Step Time-Series Analysis, etc. This simulator can be utilized to analyze and simulate connected road segments in a road intersection. Since the simulator implements regression models and provides real-time visualization, it can be used to simulate real-life scenarios to better understand a traffic situation and support decision-making.

6.2 Future Work

Regarding the tool, it is possible to improve the user interface and provide more functionalities to the user to extract traffic information from Google Maps. A timeline to showcase a number of uses and use restrictions based on feasibility analysis of the tool would make the tool more reliable and feasible for the user. As of now the tool only formats the data in tabular format which is widely accepted in the research community. However, allowing users to format data in other formats like JSON or any other customized format would make the tool more versatile and improve usability.

Regarding the Bayesian Linear Regression model, we plan to extend our work by applying the Bayesian regression approach in multivariate time series analysis for traffic congestion forecasting. Instead of incorporating the temporal features as a categorical feature, they can be normalized with other numerical values to provide users with flexibilities to provide any specific time input as an explanatory feature in the system. In that case, the model needs to interpret the final normalized output into the meaningful output for better interpretation of the results.

The traffic simulator for linear regression-based traffic analysis can be extended with features to incorporate other traffic analysis models including Neural Networks (NN), Long Short Term Memory (LSTM) and different time series analysis frameworks like ARIMA, SARIMA etc.

Bibliography

[1] A. Misra, A. Gooze, K. Watkins, M. Asad, and C. A. Le Dantec, “Crowdsourcing and Its Application to Transportation Data Collection and Management,” in *Transportation Research Record*, vol. 2414, no. 1, pp. 1–8, Jan. 2014.

[2] Shahram Tahmasseby, “Traffic Data: Bluetooth Sensors vs. Crowdsourcing—A Comparative Study to Calculate Travel Time Reliability in Calgary, Alberta, Canada,” in *Journal of Traffic and Transportation Engineering*, vol. 3, no. 2, Feb. 2015.

[3] Z. He, C.-Y. Chow, and J.-D. Zhang, “A Comparative Analysis of Journey Time from Google Maps and Intelligent Transport System in Hong Kong,” in *IEEE 21st International Conference on High Performance Computing and Communications*, Zhangjiajie, China, Aug. 2019, pp. 2610–2617.

[4] P. Baji, “Using Google Maps road traffic estimations to unfold spatial and temporal inequalities of urban road congestion: A pilot study from Budapest,” in *Hungarian Geographical Bulletin*, vol. 67, no. 1, pp. 61–74, Mar. 2018.

[5] H. Rezzouqi, I. Gryech, N. Sbihi, M. Ghogho, and H. Benbrahim, “Analyzing the Accuracy of Historical Average for Urban Traffic Forecasting Using Google Maps,” in *Intelligent Systems and Applications*, vol. 868, K. Arai, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2019, pp. 1145–1156.

[6] H. Li and L. Zhijian, ”The study and implementation of mobile GPS navigation system based on Google Maps,” *International Conference on Computer and Information Application*, Tianjin, 2010, pp. 87-90.

[7] M. Haklay, “How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets,” in *Environment and Planning B: Planning and Design*, vol. 37, no. 4, pp. 682–703, Aug. 2010.

[8] L. J. Caiza, R. Alvarez, L. Urquiza-Aguiar, X. Calderón-Hinojosa, and A. Zambrano, “VTM: Vehicular Traffic Monitor via Images Processing of Google Maps,” in *Proceedings of the 15th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks - PE-WASUN*, Montreal, QC, Canada, 2018, pp. 40–46.

[9] H. Williams and D. Crawford, “Google’s Earth. Google Maps the Future of Traffic and Travel Information?,” in *ITS International*, vol. 18, no. 1, 2012, Accessed: Jan. 09, 2021. [Online]. Available: https://trid.trb.org/view/1136681.

[10] P. Tafidis et al., “Can Google Maps Popular Times Be an Alternative Source of Information to Estimate Traffic-Related Impacts?,” in *Transportation Research Board 97th Annual Meeting Transportation Research Board*, 2018, Accessed: Jan. 09, 2021. [Online]. Available: https://trid.trb.org/view/1494705.

[11] S. Mishra, D. Bhattacharya, and A. Gupta, “Congestion Adaptive Traffic Light Control and Notification Architecture Using Google Maps APIs,” in *Data*, vol. 3, no. 4, p. 67, Dec. 2018.

[12] H. Wu, “Comparing Google Maps and Uber Movement Travel Time Data,” in *Transport Findings*, Nov. 2018.

[13] F. Wang and Y. Xu, “Estimating O–D travel time matrix by Google Maps API: implementation, advantages, and implications,” in *Annals of GIS*, vol. 17, no. 4, pp. 199–209, Dec. 2011.

[14] F. Qiu, W. Li, and C. An, “A Google Maps-Based Flex-Route Transit Scheduling System,” in *COTA International Conference of Transportation Professionals*, Changsha, China, Jun. 2014, pp. 247–257.

[15] T. Dimitrov Berov, “A Vehicle Routing Planning System for Goods Distribution in Urban Areas Using Google Maps and Genetic Algorithm,” in *International Journal for Traffic and Transport Engineering*, vol. 6, no. 2, pp. 159–167, Jun. 2016.

[16] P. Pokorný, ”Determining Traffic Levels in Cities Using Google Maps,” in *Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI)*, Corfu, 2017, pp. 144-147.

[17] T. Bellemans, B. De Schutter, and B. De Moor, “Data Acquisition, Interfacing and Pre-Processing of Highway Traffic Data,” in *Telematics Automotive*, Birmingham, United Kingdom, 2000, Accessed: Jan. 09, 2021. [Online]. Available: https://trid.trb.org/view/669866.

[18] X. Yan and X. Su, *Linear regression analysis: theory and computing*. Singapore, Hackensack, NJ: World Scientific, 2009.

[19] L. Pun, P. Zhao and X. Liu, ”A Multiple Regression Approach for Traffic Flow Estimation,” in *IEEE Access*, vol. 7, pp. 35998-36009, 2019.

[20] I. Alam, D. Md. Farid, and R. J. F. Rossetti, “The Prediction of Traffic Flow with Regression Analysis,” in *Emerging Technologies in Data Mining and Information Security*, Singapore, 2019, pp. 661–671.

[21] J. Wakefield, “Bayesian and Frequentist Regression Methods,” in *Springer Science & Business Media*, 2013.

[22] D. C. Montgomery, E. A. Peck, and G. G. Vining, “Introduction to linear regression analysis”, in *5th ed. Hoboken, NJ: Wiley*, 2012.

[23] I. Castillo, J. Schmidt-Hieber, and A. van der Vaart, “Bayesian linear regression with sparse priors,” in *Annals of Statistics*, vol. 43, no. 5, pp. 1986–2018, Oct. 2015.

[24] J. Pek and T. Van Zandt, “Frequentist and Bayesian approaches to data analysis: Evaluation and estimation,” in *Psychology Learning & Teaching*, vol. 19, no. 1, pp. 21–35, Mar. 2020.

[25] D. G. T. Denison, C. C. Holmes, B. K. Mallick, and A. F. M. Smith, “Bayesian Methods for Nonlinear Classification and Regression,” in *John Wiley & Sons*, 2002.

[26] ”Overview — Distance Matrix API”, Google Developers. https://developers.google.com/maps/documentation/distance-matrix/overview (accessed Nov. 28, 2021).

[27] ”OpenStreetMap”, OpenStreetMap. https://www.openstreetmap.org/ (accessed Nov. 28, 2021).

[28] ”Google Maps Traffic Data Extractor - Free Tier — Outscraper”, Nov. 16, 2019. https://outscraper.com/google-maps-traffic-extractor/ (accessed Nov. 28, 2021).

[29] C. Zhang, S. Sun and G. Yu, ”A Bayesian network approach to time series forecasting of short-term traffic flows,” in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, Washington, WA, USA, 2004, pp. 216-221.

[30] S. Sun, C. Zhang and G. Yu, ”A Bayesian network approach to traffic flow forecasting,” in *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 124-132, March 2006.

[31] Z. Zhu, B. Peng, C. Xiong, and L. Zhang, “Short-term traffic flow prediction with linear conditional Gaussian Bayesian network,” in *Journal of Advanced Transportation*, vol. 50, no. 6, pp. 1111–1123, 2016.

[32] B. Zhang, C. Zhang, and X. Yi, “Competitive EM algorithm for finite mixture models,” in *Pattern Recognition*, vol. 37, no. 1, pp. 131–144, Jan. 2004.

[33] R. Bro and A. K. Smilde, “Principal component analysis,” in *Analytical Methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[34] C. R. Rao, “Some Comments on the Minimum mean Square Error as a Criterion of Estimation.,” Pittsburgh Univ Pa Inst For Statistics And Applications, Oct. 1980. Accessed: Feb. 14, 2021. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA093824.

[35] B. J. Schroeder, “Part 2 - Transportation Planning,” in *Highway Engineering*, D. J. Findley, B. J. Schroeder, C. M. Cunningham, and T. H. Brown, Eds. Butterworth-Heinemann, 2016, pp. 17–88.

[36] V. Cherfaoui, T. Denoeux and Z. L. Cherfi, ”Distributed data fusion: application to confidence management in vehicular networks,” in *11th International Conference on Information Fusion*, Cologne, Germany, 2008, pp. 1-8.

[37] S. Mostafi, and K. Elgazzar, “An Open Source Tool to Extract Traffic Data from Google Maps,” in *5th IEEE International Conference on Fog and Edge Computing 2021*, Australia, 2021. submitted for publication.

[38] D. van Ravenzwaaij, P. Cassey, and S. D. Brown, “A simple introduction to Markov Chain Monte–Carlo sampling,” in *Psychonomic Bulletin & Review*, vol. 25, no. 1, pp. 143–154, Feb. 2018.

[39] M. Nishio and A. Arakawa, “Performance of Hamiltonian Monte Carlo and No-U-Turn Sampler for estimating genetic parameters and breeding values,” in *Genetics Selection Evolution*, vol. 51, no. 1, p. 73, Dec. 2019.

[40] M. Burda and J. M. Maheu, “Bayesian adaptively updated Hamiltonian Monte Carlo with an application to high-dimensional BEKK GARCH models,” in *Studies in Non-linear Dynamics and Econometrics*, vol. 17, no. 4, Jan. 2013.

[41] M. Betancourt, “The Convergence of Markov Chain Monte Carlo Methods: From the Metropolis Method to Hamiltonian Monte Carlo,” in *Annalen der Physik*, vol. 531, no. 3, p. 1700214, Mar. 2019.

[42] C. De Mol, E. De Vito, and L. Rosasco, “Elastic-net regularization in learning theory,” in *Journal of Complexity*, vol. 25, no. 2, pp. 201–230, Apr. 2009.

[43] Z. Wang, M. Broccardo, and J. Song, “Hamiltonian Monte Carlo methods for Subset Simulation in reliability analysis,” in *Structural Safety*, vol. 76, pp. 51–67, Jan. 2019.

[44] D. E. Farrar and R. R. Glauber, “Multicollinearity in Regression Analysis: The Problem Revisited,” in *The Review of Economics and Statistics*, vol. 49, no. 1, pp. 92–107, 1967.

[45] J. Barcelo, Ed., *Fundamentals of Traffic Simulation*. New York: Springer, 2010

[46] A. Pell, A. Meingast, and O. Schauer, “Trends in Real-time Traffic Simulation,” in *Transportation Research Procedia*, vol. 25, pp. 1477–1484, 2017.

[47] G. Kotusevski and K. A. Hawick, ”A review of traffic simulation software”. 2009, Accessed: Jul. 30, 2021. [Online]. Available: https://mro.massey.ac.nz/handle/10179/4506

[48] M. M. Mubasher and J. Syed Waqar ul Qounain, ”Systematic literature review of vehicular traffic flow simulators,” in *International Conference on Open Source Software Computing (OSSCOM)*, 2015, pp. 1-6.

[49] P. M. Ejercito, K. G. E. Nebrija, R. P. Feria and L. L. Lara-Figueroa, ”Traffic simulation software review”. in *8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2017, pp. 1-4.

[50] M. Fellendorf, and P. Vortisch, “Microscopic traffic flow simulator VISSIM,” in *Fundamentals of Traffic Simulation*, J. Barceló, Ed. New York, NY: Springer, 2010, pp. 63–93.

[51] M. Treiber and A. Kesting, ”An Open-Source Microscopic Traffic Simulator,” in *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 3, pp. 6-13, Fall 2010.

[52] H. Ramadhan and I. G. B. B. Nugraha, ”Web-based macroscopic road traffic simulator,” in *11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, 2017, pp. 1-6.

[53] J. Miller and E. Horowitz, ”FreeSim - a free real-time freeway traffic simulator,” 2007 IEEE Intelligent Transportation Systems Conference, 2007, pp. 18-23.

[54] P. A. Lopez et al., “Microscopic Traffic Simulation using SUMO,” in *21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, Nov. 2018, pp. 2575–2582.

[55] S. Haddouch, H. Hachimi and N. Hmina, ”Modeling the flow of road traffic with the SUMO simulator,” in *4th International Conference on Optimization and Applications (ICOA)*, 2018, pp. 1-5.

[56] R. Mena, F. Zumárraga, L. Urquiza and X. Calderón, ”Google Maps Route Color Mapping with SUMO Simulator,” in *International Conference on Information Systems and Software Technologies (ICI2ST)*, 2019, pp. 92-99.

[57] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, “Traffic Simulation with Aimsun,” in *Fundamentals of Traffic Simulation*, J. Barceló, Ed. New York, NY: Springer, 2010, pp. 173–232.

[58] M. Lindorfer, C. Backfrieder, C. F. Mecklenbräuker and G. Ostermayer, ”Modeling Isolated Traffic Control Strategies in TraffSim,” in *UKSim-AMSS 19th International Conference on Computer Modelling & Simulation (UKSim)*, 2017, pp. 143-148.

[59] P. Cai, Y. Lee, Y. Luo and D. Hsu, ”SUMMIT: A Simulator for Urban Driving in Massive Mixed Traffic,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4023-4029.

[60] D. K. Sorenson and J. Collins, “Practical Applications Of Traffic Simulation Using SimTraffic,” presented at the Compendium of Papers. Institute of Transportation Engineers 2000, District 6 Annual MeetingInstitute of Transportation Engineers (ITE), 2000. Accessed: Jul. 30, 2021. [Online]. Available: https://trid.trb.org/view/671329

[61] L. Wang, Z. Liu and Z. Liu, ”Research on correction method of traffic simulation model based on linear regression,” in *International Conference on Anti-Counterfeiting, Security and Identification*, 2010, pp. 192-194.

[62] O. K. Golovnin, K. V. Pupynin and A. S. Privalov, ”A Web-Oriented Approach for Urban Road Traffic Simulation,” in *International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, 2019, pp. 1-4.

[63] H. Mizuta, ”Evaluation of metropolitan traffic flow with agent-based traffic simulator and approximated vehicle behavior model near intersections,” in *Winter Simulation Conference (WSC)*, 2015, pp. 3925-3936.