Machine Learning for Traffic Speed Prediction using Open Source Data

Widiatmoko Azis Fadilah

Main academic Supervisor: Prof., Christos, TJORTJIS, International Hellenic Univ.

A Master Thesis submitted for the Erasmus Mundus Joint Master Degree on Smart Cities and Communities (SMACCs)

June 2022

University of Mons, Heriot-Watt University, International Hellenic University, University of the Basque Country

Acknowledgments

I am incredibly grateful to the SMACCs consortium for funding my studies. Without it, I would not have had a wonderful experience studying in Europe and meeting some new friends. Special thanks to my supervisor, Prof. Christos Tjortjis. Completing this master thesis could not have been possible without guidance from him. I want to express my gratitude to the following people for helping me mentally and academically:

1. Aristeidis Mystakidis for giving me some advice and being open to my questions and discussions;
2. My family, especially my mom, for supporting me and giving some advice;
3. My friends, especially Fakhriy Ramadhan, Hammad Farooq, Aqeel Ahmed, and Komar Javanmardi, for accompanying me for most of my journey studying and living in Europe.

I would also like to thank Panagiotis Tzenos and the Centre for Research and Technology Hellas (CERTH) for providing the data for this study and giving me a detailed explanation of how the data was collected.

Abstract

With rapid population growth and development, traffic congestion has become a problem over the past decades, especially in the big cities. Traffic speed is one of the indicators to determine traffic conditions. With the current development of sensors and other data collection devices and the advancement of the Internet of Things (IoT), more traffic speed data has become available. At the same time, there is growing interest in Intelligent Transportation Systems (ITS) that may help solve the current traffic problem.

Traffic speed prediction is one of the components in ITS that has become one of the most developed research areas. However, most of the studies used Deep Learning (DL) algorithms which require complex data preprocessing and extensive computational power. Moreover, many studies did not consider the spatial features and only predicted the speed in a road corridor or the segment where a sensor is available. Therefore, this study is trying to compare the Machine Learning (ML) performance to an earlier study by considering the spatial features and finding the most effective way to use these models for network-wide prediction.

Floating Car Data (FCD) and OpenStreetMap (OSM) data were used and preprocessed by following the approach of the previous study. The data were trained and tested using several ML models. The result was evaluated by comparing it to the initial study's results. Moreover, several scenarios were tested to determine the most efficient way to predict the traffic speed on a network-wide scale. The Gradient Boosting (GB) algorithm is the best model in this study, with consistent performance in all scenarios and promising results.

Keywords: traffic speed prediction, data mining, machine learning, floating car data

Widiatmoko Azis Fadilah

June

Table of Contents

Table of Figures

Tables

1 Introduction

These days, population growth has been increasing fast and impacts our lives in many ways. The number of vehicles, especially in large cities, is also increasing along with the change. This rapid growth causes congestion problem that leads to health issues because of the increasing amount of CO_2 emissions in the air and noise pollution [1]. It also affects the economy since traffic congestion could cost much and decelerate development [1], [2]. It increases travel time, causing slow circulation of goods and people, which leads to inefficient work [2]. On top of that, there are several other effects of traffic congestion, namely, increasing traffic accidents, fuel consumption, and higher operating cost of vehicles [1].

Therefore, traffic congestion is not an easy problem to be solved, many efforts have been made to tackle the problem, yet it is still an issue today. One solution currently developing in this digitalization era is the Intelligent Transportation System (ITS). Supported by the amount of data generated and circulated in the cities and the growth of the IoT in recent years, ITS aims to provide a safer, coordinated, and "smarter" transportation system [3]–[5]. Hence, it will achieve traffic efficiency by minimizing traffic problems that are mainly traffic congestion [4], [6]. Moreover, it can provide benefits such as offering a range of affordable and convenient transport options and keeping track of incidents and congestion for road users and travelers [4].

One component that can support the development of ITS is traffic prediction. According to [7], the traffic prediction's final target is to make actual transportation intelligent. Therefore, building a unified ITS that can manage different data sources and efficiently handle traffic data analysis and mining processes in one environment is necessary. However, traffic prediction itself is not a simple problem. It consists of a broad spectrum that we can categorize into some concepts based on the perspective group of people, such as traffic status (crowds), traffic flow (governments), and travel demand prediction (related companies). Furthermore, from that variety of concepts, traffic problems can generally be divided into traffic classification, generation, and forecasting [7].

Moreover, according to [7], traffic prediction has some challenges. One of the challenges is *improving the current traffic prediction performance* in addressing the joint traffic prediction problem because of the possibility that traffic prediction problems will consider a different type of traffic data than any other complex factor and features. Additionally, according to [4], there are some gaps in traffic prediction research. Some lack the road types as input for traffic prediction, some *do not consider spatial and temporal relationships*, and some neural network techniques can only capture spatial or temporal data. Also, in the scale of prediction area, the network-wide traffic speed has become an important and challenging topic according to [8]

1.1 Objectives

In order to address the mentioned challenges, the research aims to compare the prediction performance from [9] using the approach in [3]. The model will also incorporate some spatial elements, such as road length and bus stops in the analyzed segment, to determine how it will affect performance. In addition, we will investigate the practical scenario for which to perform network-wide prediction to gain information sufficient for route planning analysis.

1.2 Research Questions and Boundaries

From the objectives above, the emerging research questions are:

a. How good is the prediction model's performance when the features in [3] are used to create a prediction model compared to the model in [9]?
b. How do spatial elements such as bus stops and road length affect traffic conditions?
c. What is the efficient scenario to predict network-wide traffic speed?

There are some research boundaries to limit the work, which are,

a. The study will be limited to Thessaloniki, Greece.
b. Only several traditional Machine Learning (ML) models will be used in this study.
c. The result for Network-Wide prediction will be in the form of scenario suggestions, not a prediction result

1.3 Contribution and Thesis Structure

In this section, the contributions from this study will be presented, followed by the structure of the thesis.

The Difference with Previous Research

Traffic prediction problems can be handled using either ML or DL algorithms. Some studies used DL models such as DL based multitask learning (MTL), CNN, RNN, and LSTM [8], [10], [11]. Indeed, DL algorithms may give more accurate results compared to ML algorithms. However, DL algorithms might be more time-consuming and computational power hungry when compared to ML algorithms. Thus in this study, traditional ML algorithms are used and compared to find a suitable model for the given data.

Classification has been used to deal with traffic prediction problems in several studies [3], [12], [13]. In [12], a prediction model based on weather conditions was developed and continued in [13] to study the impact of atypical conditions—in this case, COVID-19, on the result of the traffic prediction model. Those studies concluded that weather data could support the decision-making for traffic prediction while assessing the impact of a long, atypical condition. Moreover, some features can help the model, such as different intervals during the day—e.g., morning, afternoon, and evening. However, something that must be noted from [12] that the prediction model's result was based on weather data, which was also predicted. Thus, an accurate weather prediction model is expected, or bias to the traffic prediction will be introduced.

In a slightly similar fashion, the primary research objectives in [3] aimed to develop a framework with ML techniques—i.e., Classification, to tackle traffic congestion. With the data obtained from an API that gives the information about the congestion status in each road segment, a prediction model using the classification technique was developed—specifically, the model to predict the road segments with limited data. However, the road feature, such as bus stop or intersection, was not included in the features used as the predictor.

The importance of bus stops and intersections in affecting traffic was discussed in [14], which studied the factors contributing to traffic congestion at some critical traffic points. The research also analyzes how those factors influence the road network using travel time-delay data. It found that bus stops, T-junctions, and cross junctions are the bottlenecks to the vehicle flow and result in slow vehicle movement. Therefore, its result becomes one of the driving factors in performing this study.

Another way to deal with traffic prediction problems is to use a combination of parametric and non-parametric models; such was presented by Theodorou et al. in [15]. In his research, this combination method was developed to consider typical and atypical conditions in traffic prediction algorithms where the KNN regression model was used under atypical conditions. In contrast, the ARIMA model, on the other hand, was used when the typical situation was detected. Moreover, the initial research [3], [9] only analyzed one road or a few road segments. It is also supported in [8], as most traffic prediction studies were done in a corridor or around a sensor location. Thus the workflow and automation to explore the network-wide scale prediction are needed. Therefore, the contribution of this study can be summarised as follows:

1. A traffic speed prediction using the features proposed by [3] considers the time of the day and the behavior of stores. Additionally, spatial features will be included to see how they affect the prediction.
2. Several ML algorithms used in [9] with some more models will be used to train the models. The prediction result from this study will be compared with the result from [9] to determine if the model performs better than the previous study.
3. Some scenarios will be created to find an effective way to perform network-wide traffic speed prediction using ML algorithms.

Report Structure

The remainder of this thesis has the following structure. Chapter 2 explains the theoretical framework of this study. Chapter 3 presents the data description and the method conducted in this research, followed by Chapter 4, which presents the results. Continued with the discussion in Chapter 0, and finally, in Chapter 6, the conclusions and further studies will be explained.

2 Theoretical Framework

This chapter will explain the theoretical concept of traffic problems in our city, followed by the basic idea of data mining. Additionally, there will be some explanation about several ML algorithms used in this study. Moreover, how ML can tackle traffic problems also will be discussed.

2.1 Traffic Problems and Smart City

Congestion happens when the number of vehicles circulating throughout the city road network exceeds the maximum traffic flow. It also qualifies as congestion when the situation affects the traffic flow and causes increasing journey times [1]. Figure 1 [1] can easily explain the correlation between the number of vehicles on the street and the time to traverse along a particular street to visualize this phenomenon.

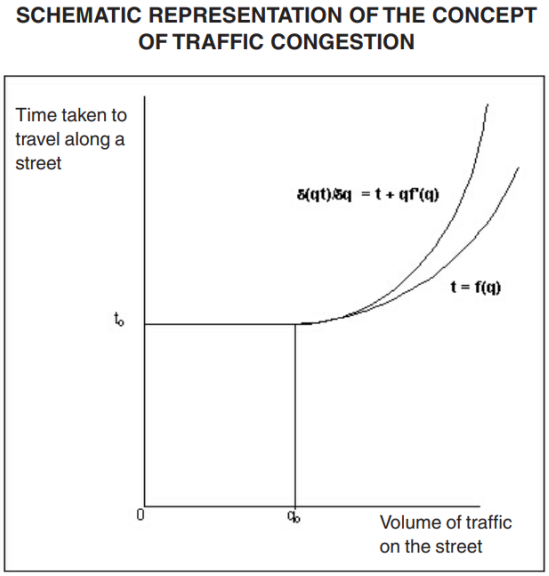

Figure 1: The concept of traffic congestion representation [1]

Therefore, according to Figure 1, there will be times when the travel time for all vehicles is the same when they can move at the usual speed. However, with increasing traffic volume, the time needed to traverse a street will be longer than when the volume is much lower. The increasing traffic volume has been a problem from the early 1990s until recent years and causes serious congestion problems, especially in large cities [1]. It is also caused by population growth, the increasing purchasing power of the middle-

income socioeconomic classes, the relative reduction in vehicle prices, and the availability of used vehicles [1], [3].

2.1.1 Factors Affecting Congestion

Other than the massive contribution from using private vehicles, several other factors can cause congestion. These factors are [1]:

1. The complexity of the public transport problem itself: for example, because the solution derives from the need to travel to the places that have different kinds of activities that take place in a different location; or because of the public transport that takes place on limited road spaces; and any other relevant example. This problem leads to congestion that occurs at various points.
2. The urban road network design and maintenance: the example of this factor, such as failure to mark traffic lanes, unexpected changes in the number of lanes, and the location of the bus stop, can cause disturbances to the traffic flow. Additionally, the condition of the road might increase the congestion as well.
3. Citizens' driving habits include forcing their way into intersections, blocking the passage of other drivers, buses that stop before an intersection, and other habits that show that some drivers do not respect other road users. This type of habit can cause congestion or, in the worst case, cause an accident.
4. The institutional problem may cause uncomprehensive planning. It may result in the development where it encourages people to use their private vehicles. Take, for example, uncoordinated action by the government official in deciding the city's development.

2.1.2 The Impacts of Congestion

Traffic congestion can cause many harms to many aspects of life. The most direct impact of this problem is the increasing time to travel from one point to another. It is not only caused by the reason already mentioned before but also because there will be a lot of public transport jams, resulting in slower travel time. The other effect of the congestion is the pollution exhausted from the congested vehicles, which also may lead to health problems among the citizens that live around the congested area [3]. Moreover, congestion can also cause an increase in vehicle energy consumption. Therefore, this problem may lead to an unsustainable city's lifestyle if it stays as it is for a long time [1].

However, with the advancement of sensor technology and IoT, a massive amount of data can be gathered throughout the cities. These data can be used to monitor and predict traffic. By monitoring the traffic and the help of interconnected devices, the operator or the machine can send the signal to the user for real-time information. While on the other hand, the predictive analysis using these data can help the planer and decision-maker to anticipate the problems and find out where and when these problems will occur. Additionally, by utilizing this development, the cities only have to invest in the technology, which has a lot less cost than building more roads or interchanges [3], [16].

2.2 Data Mining

Data mining is a process for extracting information from a massive amount of data to find new patterns or information for a particular problem. This technology is designed to utilize the stored data and extract new information using ML algorithms. The goal of this study is to predict continuous values of traffic speed. Thus, the algorithms used are the ones that can handle regression tasks. Some of the ML models used in this study are explained in the following sub-sections.

2.2.1 Linear Regression (LR)

LR or *Ordinary Least Squares* (OLS) is a classic and widely used model that uses the input feature's linear function to obtain the target value. The model consists of a dependent variable (DV) or target, an independent variable (IV) or features, coefficients showing the IV's proportion affects the DV, and the constant, as can be seen in the equation (1) below [17].

$$y = c_0 + c_1 * x \quad (1)$$

Through equation (1), we can find the "best line" that fits the data input with a minimum sum of squares difference between the model and the predicted value. The principal of finding this "best line" is fitting a linear model with a combination of coefficients with a minimal residual sum of squares between the actual target value and the predicted target value. When dealing with more than one feature, the model becomes a multiple linear regression function, as seen in the following equation (2) below [17], [18].

$$y = c_0 + c_1 * x_1 + c_2 * x_2 + \cdots + c_n * x_n \quad (2)$$

This model is quite fast to train and predict. Also, it works well with a vast dataset and sparse data. Another strength of this model is the ease of understanding how the model is created [17]. However, this model relies on the independency of its feature. Therefore, the more correlated a feature, the design matrix becomes closer to singular. As a result, it creates a model prone to random error in the actual target value and produces a significant variance [18].

2.2.2 K-Nearest Neighbors (KNN)

The KNN model is the most straightforward ML algorithm. It will predict by looking at the *k* nearest data points in the training dataset, which is why it is called "nearest neighbors." In this scenario, *k* is an integer value specified by the user. After selecting *k* nearest data points, the prediction will calculate the average or the most relevant neighbors, as seen in Figure 2 [17].

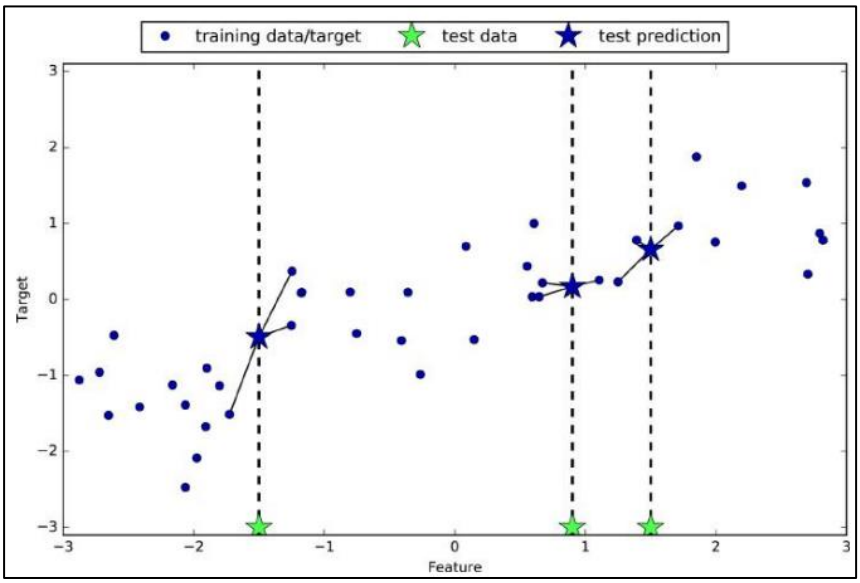

Figure 2: A visualization of prediction using KNN regressor taking three neighbor points [17]

As pictured in Figure 2, the KNN model strength is easy to understand. However, this model has some flaws, such as being unable to deal with a large amount of data and leading to a slower processing time. Thus, this model is not often used in practice [17].

The algorithm typically uses uniform weights, meaning that the local neighbor points contribute equally to the point that is being predicted. However, the weighting parameter can be changed to the distance to imply that the nearest points contribute more compared to the points that are far away [17], [19].

Several algorithms can be applied to optimize the model: *Brute Force*, *K-D Tree*, and *Ball Tree*. The difference between each algorithm is how efficient the distance calculation is performed. However, this does not mean that the most inefficient algorithm should not be used. For example, using the brute force algorithm in a small dataset is more efficient than any other tree-based approach [16]. Therefore, one thing to consider when selecting the algorithm is how big the data sets will be predicted.

2.2.3 Support Vector Regression (SVR)

SVR is a part of Kernelized Support Vector Machine (SVM) which allows complex models and is not defined simply by hyperplanes in the input spaces and implemented in the regression problems [17]. SVM constructs a hyper-plane or a set of hyper-planes in a high or infinite-dimensional space. SVM will have good separation when the hyper-plane has a significant distance to the nearest training data point of any class—the more significant the distance, the lower the generalization error of the classifier. Figure 3 below shows how SVM creates so-called "support vectors" with three samples on the margin boundaries [17], [20].

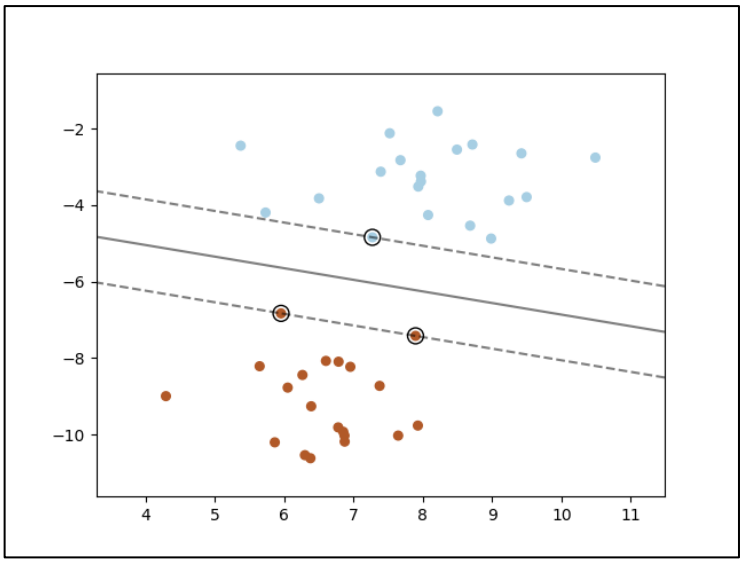

Figure 3: Decision function for linear problem creating "support vectors" [20]

The Gaussian kernel measures the distance between the data points with the following equation (3) [9], [17].

$$k_{rbf}(x_1, x_2) = \exp\left(\gamma \|x_1 - x_2\|^2\right) \qquad (3)$$

In equation (3), x_1 and x_2 are the data points, $\|x_1 - x_2\|$ is the Euclidean distance, and γ is the parameter that controls the width of the Gaussian kernel [17].

SVR is a powerful ML model used on various low- or high-dimensional datasets. It also allows complex decision boundaries, even if the data is only a few. However, its requirement for computing and storing will increase rapidly along with the increment number of training data. Thus, working with 10,000 samples of SVR might work well, but when the dataset has 100,000 samples or more, it will be more challenging in terms of runtime and memory usage. Moreover, parameters C and ɣ can be tuned to improve the performance of the SVR model [17], [20].

2.2.4 Decision Trees (DTs) Regression

DTs model is a non-parametric supervised learning method that can be used for classification or regression tasks. The model learns a hierarchy of if/else questions from its features, leading to a decision that predicts the target value. The following Figure 4 shows the DTs model result. It also shows that the depth of the trees affects model complexity, but the fitter the model [21].

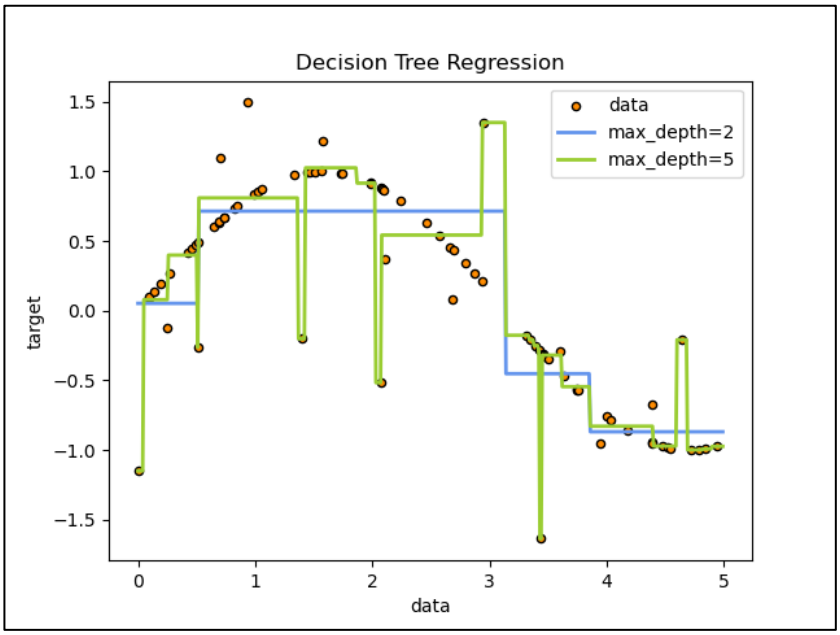

Figure 4: Example of DTs regressor result visualization [21]

From Figure 4, we can conclude that the resulting model is easy to visualize and understood by the nonexperts. Also, the other advantage of using this model is that it requires a bit of data preparation, as it is utterly invariant to the data scaling [17], [21]. However, DTs regressor and all other tree-based regression models cannot make predictions outside the training data range. It means that DTs might produce an overfitted model, which is very good at predicting the training dataset, but once leaving the train-

ing data range, the model will keep simply predicting the last known point. The problem can be minimized by controlling the complexity of the decision tree model using the pre-pruning parameters that stop the tree before it is fully developed [17], [22].

2.2.5 Random Forest (RF) Regression

RF is a part of an ensemble model that combines multiple ML models to create more powerful models. This model is intended to overcome the drawback of the DTs model, which tends to overfit the training data. Thus, the idea of RF is performing multiple DTs, where each tree is slightly different from the others and drawn from the training set. Therefore, by creating many trees, each tree will overfit in different ways, and the problem can be reduced by averaging their results [9], [17], [23]. Figure 5 visualize how RF predicts this thesis's speed value [9].

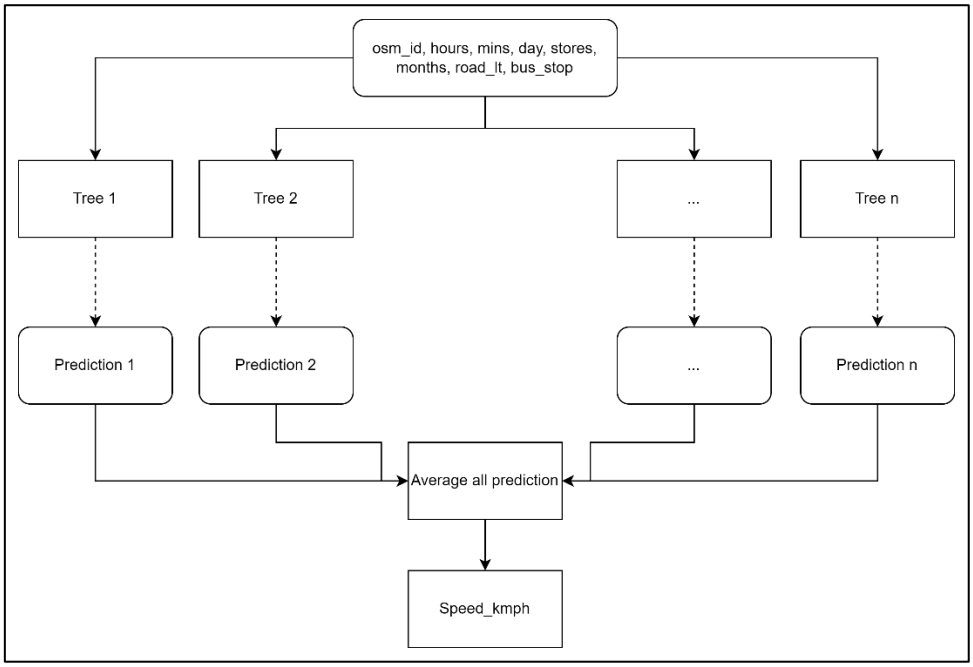

Figure 5: Visualization of how the RF prediction model works [9]

In splitting each node when constructing a tree, two options of parameters can be selected. Either from all input features or from a random subset with the size of "max_features" inserted by the user. These parameters can decrease the variance of the forest estimator, as some errors from DTs will cancel out each other, and thus, RF will become less prone to overfitting [17], [23]. As a result, RF becomes more robust and accurate compared to DTs [22]. However, RF is not working well on very high-dimensional and sparse data. Also, it requires much memory, resulting in a slower processing time [17].

2.2.6 Gradient Boosting (GB) Regression

Like RF, GB is also part of an ensemble model that combines multiple DTs and can be used for classification or regression tasks. However, GB work by serially building trees. Thus, it will correct the mistakes from the previous trees. It often uses very shallow trees with a depth from one to five, which makes the model smaller in memory and leads to faster processing. With this combination of shallow trees, each of them will provide a good prediction on the part of the data. After iteratively adding this shallow tree, the model's performance will eventually be improved [17], [23].

Several parameters can be set to optimize the GB model: pre-pruning the trees, adjusting the number of trees, and controlling the learning rate. Increasing the number of trees will increase the model complexity and might be able to correct mistakes on the training sets. At the same time, increasing the learning rate will also allow for a more complex model as each tree can make more substantial corrections. However, increasing the model complexity might also lead to overfitting [17].

2.2.7 Multilayer Perceptrons (MLPs)

MLPs are known as the feed-forward neural networks (NN), part of the “deep learning” family algorithm. It can be seen as a generalization of linear models that perform multiple processing stages to find the decision. Similar to the linear problem, predicting the output works to find the weight of the input features. However, in MLP, the process of computing the weight is done multiple times. Figure 6 below shows the process of the NN in predicting the target value.

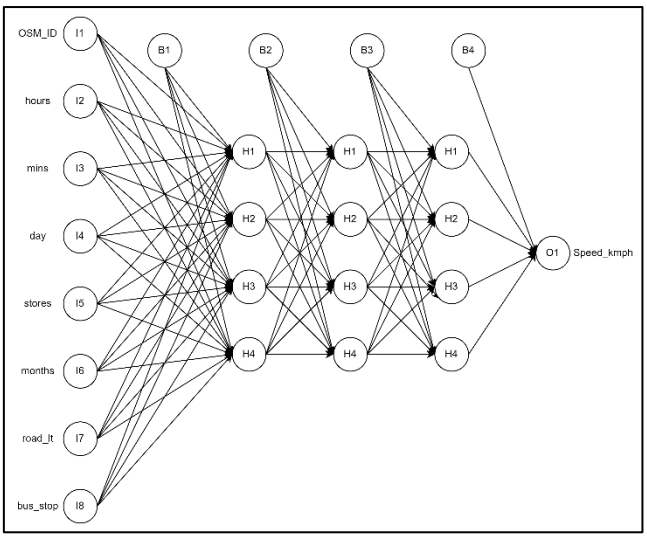

Figure 6: Visualization of how the NN model works in predicting the traffic speed [9]

In Figure 6, each node on the left represents an input feature, while the node on the far right side is the output, which in the figure below is the predicted mean speed. MLP has many more calculated weights or coefficients represented by the connecting lines from the input feature node to the hidden layer, from the first hidden layer to the next hidden layer, and so on, until it reaches the output node. In addition, after computing the weighted sum for each hidden layer, a non-linear function—such as a hyperbolic function, is performed to calculate the weighted sum used to compute the output [17], [24].

In this study, MLPRegressor is used, which implements MLP that trains using backpropagation with no activation in the output layer; thus, it uses the square error as the loss function, and the output set is a set of continuous values. This model can learn non-linear models and works in real-time, which are the advantages of using this model. Thus, this model can create an incredibly complex model. However, as MLP has a non-convex loss function, applying weight initialization is necessary to increase the validation accuracy. Therefore, the drawback of using MLP is the requirement to tune several hyperparameters such as hidden neurons, layers, and iteration. Moreover, MLP takes a long time to train, and it is sensitive to feature scaling, which leads to the need for careful data preprocessing [9], [17], [24].

2.3 Intelligent Transportation System (ITS)

Currently, because the capabilities of sensors or other data collection systems have increased significantly, the amount of data with a variety of quality and types are vastly available. Simultaneously, this factor becomes an opportunity and challenge for developing an Intelligent Transportation System (ITS) [8], [11]. It is developed to provide a safer, coordinated, and “smarter” transportation system to minimize traffic congestion and achieve efficient traffic. It has the advantage of providing affordable and convenient transport options and keeping track of incidents and congestion for road users and travelers [3]–[6]. ITS has become a solution that attracted much interest, as it can collect and process data from different sources using current computer infrastructure and the latest algorithms to create efficient traffic [25].

Because the ITS relies on the quality of traffic information, the study of traffic prediction has become one of the most developed research areas [8], [9], [11]. According to [7], the traffic prediction’s final target is to make actual transportation intelligent.

Therefore, building a unified ITS that can manage different data sources and efficiently handle traffic data analysis and mining processes in one environment is necessary.

2.4 Traffic Prediction using Machine Learning

A mathematical model can describe the traffic phenomenon and perform a prediction through it. Therefore, we can analyze the issues if we have the information to optimize the traffic parameters [26]. Moreover, many methods were proposed to forecast the traffic speed, volume, density, and travel time in the last three decades, aiming to improve the prediction accuracy [4], [8], [9]. According to [8], because the traffic conditions are becoming more complex, with the addition of sensors and computational power development, rather than using classical statistic methods, most studies are shifting towards computational intelligence (CI) in traffic prediction.

There are several scenarios for traffic prediction: short-term (e.g., 10-15 minutes ahead), long-term (e.g., next day), congestion prediction, and travel times analysis (by analyzing the shortest route and forecasting the traffic). As traffic problem is a much more complex phenomenon, it is often disturbed by sudden incidents such as accidents and extreme weather. Thus linear regression model is insufficient [11]. Many studies use Recurrent Neural Networks/LSTM to tackle this problem [8], [26]. However, in this study, these sophisticated models were not used due to the process's complexity.

Numerous applications utilize the result of traffic prediction. For example, in real-time traffic management, we can generate the typical traffic profiles from real-world traffic data (i.e., FCD data). Then, these profiles can act as an input to the NN model, together with the real-time traffic data, to recalibrate the traffic parameters if an atypical condition occurs in the road network. The other applications are to find the optimal location and capacities for parking and charging stations and the optimal tolling policies [26]. Some of these applications require a network-wide scale traffic prediction, which has become an important and challenging topic [8].

3 Methodology

This chapter will discuss the data used in this study, continued by how it is preprocessed and processed to obtain the prediction result. This section will also explain how the model will be evaluated.

3.1 Methodology

This study aims to improve the model performance from [9] using the idea from [3] and to find the best scenario for predicting the traffic speed on a network-wide scale. Thus, the prediction process can be simpler and faster.

Therefore, the general idea for the methodology is first to prepare the data in a way similar to [3]. At the same time, for the case when multiple road segments are analyzed, additional spatial features from OSM Data are added to the input features. After the input data is ready, the input dataset is then selected and masked accordingly, so the selected data satisfy each scenario. Before the assessment, after training the data input, each ML model will be evaluated and hyperparameter tuned to reach the optimum result. Eventually, the prediction result will be used to assess the ML models' performance and find the best scenario for network-wide prediction.

The overall process in this study is conducted using Python 3.8.8 programming with the scikit learn 1.0.2 library [27]. The IDE used for this process is Jupyter Notebook in Microsoft Visual Studio. The traffic speed data is stored and preprocessed using the Postgre SQL database. Additionally, some spatial data preprocessing is done using QGIS software to obtain spatial-related features.

3.2 Data

3.2.1 Floating Car Data (FCD)

The data used in this study are some subsets of an FCD produced by a taxi fleet belonging to the taxi association “Taxiway,” which consists of vehicles operating in the region of Thessaloniki [9]. According to personal interaction with (Tzenos, 2022), to produce the network-speed dataset, the data was processed as follows:

1. Initially, the record regarding personal information from the voluntary vehicle or driver identification is discarded in the beginning process of all the available FCD records to consider the privacy aspect;
2. Then, the remaining records are filtered to remove any erroneous entries with extraordinary speeds or unaligned coordinates, or entries generated by faulty GPS receivers;
3. Using a specific algorithm that takes into account the network topology (mono-graphs, types of roads, and other network topology related.), each record (GPS entry) is mapped to the part of the road network to which it belongs with the highest degree of certainty;
4. Finally, for each link (osm_id) of the road network, proper statistical analysis is performed to provide a safe estimation of the average speed on that link.

This process produced the "speed_kmph" values in kilometers per hour, and it denotes the average speed on each link with steps that occur every 15 minutes. The data also has a "unique_entries" value that indicates the number of different vehicles (per 15 minutes) that produced the GPS data to make the average speed on each link. This data can be used to determine each link's congestion status (low, medium, and high), using a set of rules created for each link type. These rules consider the "Free flow speed" value, which during the conversion of speed values into congestion status, the ratio of "observed_speed/free_flow_speed" is evaluated (Tzenos, 2022). The following Table 1 shows the sample of the data downloaded from [28],

Table 1 Data sample from the source

osm_id	link_dir	date_time	speed_kmph	unique_entries
181919282	1	2018-04-09 12:15:00	75	1
175724175	1	2018-02-17 14:15:00	87	2
18623648	2	2018-02-13 00:30:00	40	1
14886450	1	2018-01-17 13:15:00	25	1
401315597	1	2018-04-03 08:00:00	25	1

The obtained data is a collection of already processed FCD data grouped for each month from 2018 to 2022. It has roughly 37,469,181 records each year, which means the data has around 149,876,724 entries.

3.2.2 OpenStreetMap (OSM) Data

OSM is an open-source map data built, contributed, and maintained by a community of mappers. The data contains worldwide information on roads, trails, cafés, buildings, railway stations, and other geospatial features. The data is produced using aerial imagery, GPS devices, and low-tech field maps to verify the accuracy of the information by emphasizing its contributors' local knowledge [29]. From OSM, the data on road networks and bus stops are obtained, as shown in Figure 7 below. It is displayed using QGIS 3.16, open-source software that can process geospatial data.

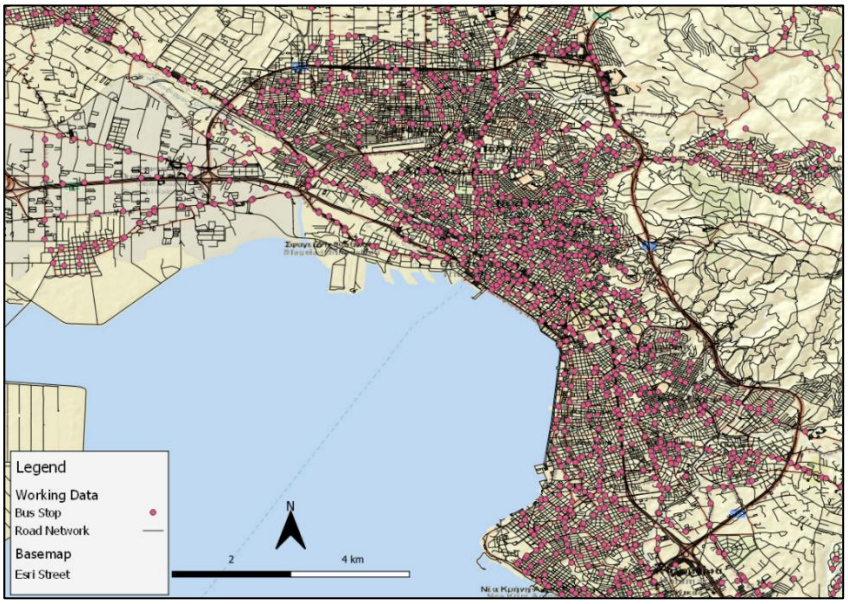

Figure 7: Road network and bus stop distribution in Thessaloniki

The road network data contain osm_id, railway, name, and many more features. However, some unnecessary features are dropped, leaving only osm_id, oneway, name, and type. On the other hand, bus stops data has features such as osm_id, public_tra, amenity, name, type, and more. Moreover, the used features are only osm_id and name for this data. Something to note here is that the osm_id in the road network feature is matched with the osm_id in FCD data and will be joined in the preprocessing step. The following Table 2 is the description of the features that are used in this study.

Table 2 Feature description from OSM data

Feature	Description
Osm_id (road network)	A unique id related to each segment in the road network links the OSM data with FCD data.
Name (road network)	The name of the road
Oneway	It shows the information whether the segment is a one-way road or not.

Table 2 Feature description from OSM data (Continued)

Feature	Description
Type	The road type, according to OSM data.
Osm_id (bus stops)	A unique id related to the bus stop
Name (bus stops)	The name of the station

In the preprocessing step, these data are merged by the nearest feature to get the information on whether the road segments have a bus stop or not. So then, later, the data can be joined with FCD data to produce the data input for creating a prediction model.

3.3 Data Preprocessing

The preprocessing stage deals with preparing FCD and OSM data and continues by combining these data. The first stage consists of the preprocessing part of FCD data to extract information from the current data and obtain new information. The preprocessing stage of OSM data continued them to gain necessary features from geospatial-related objects. The preprocessing step ends when FCD and OSM data are joined, and some road segments are selected to be ready as an input to create the prediction model.

3.3.1 The Store Feature

The FCD data comes in a text file for each month from January 2018 until March 2022, so the data were concatenated and converted into a SQL table for further processing. A python program in Appendix A was used to concatenate the files into a single table and store them in a database. In the PostgreSQL software, the data was preprocessed using some code fragments in Appendix B to derive information such as the year, month, day, hour, and minutes from the "date_time" column. An additional feature, "store," is added to the current data by using the rules used in [3]. The rules are defined as follows:

1. The stores status is 'open' from 09:00 to 20:59 on Tuesday, Thursday, and Friday; while on Monday, Wednesday, and Saturday, the stores are open from 09:00 to 17:59;
2. The stores status is 'opening' from 07:30 to 08:59 on all days except Sunday;
3. The stores status is 'closing' from 21:00 to 22:00 on Tuesday, Thursday, and Friday; while on Monday, Wednesday, and Saturday, the stores are closing from 18:00 to 19:00;
4. Other than the previous cases, the store status is closed.

Thus, the effect of store behavior on the road speed network can be incorporated into building the prediction model with this additional feature. This preprocessing result can be seen in Table 3. Note that the table only shows the new column extracted from the original data. Practically, the following features co-exist with the original data.

Table 3 Sample result of preprocessing on FCD data

osm_id	n_time	hours	mins	n_day	n_month	stores
332282307	11:45	11	45	SUNDAY	MAY	CLOSED
534308047	1:15	1	15	SUNDAY	MARCH	CLOSED
388902759	13:00	13	0	THURSDAY	SEPTEMBER	OPEN
14301548	21:45	21	45	SATURDAY	APRIL	CLOSED
163255795	19:30	19	30	FRIDAY	AUGUST	OPEN

3.3.2 The Road Feature

The downloaded OSM data of Thessaloniki comes in three separate shapefile formats according to its geometry: point, line, and polygon. For this study, the analysis only used the points and lines shapefile.

Feature Extraction

The first step of preprocessing the OSM data start with feature extraction. Because the points and lines shapefiles contain not only the information needed for the analysis, it is necessary to apply feature selection to select only road network and bus stop features.

Road Network

Figure 8 below shows the sample result of road network selection. From the figure, the line shapefile contains all geographical features such as rail and road networks.

Figure 8: Line shapefile data with all features (left) and selected road network data (right)

Road network feature comes from line shapefile that contains various information such as road network itself, train line, proposed metro line, and some other features that can be depicted as line geographically. Therefore, a spatial query was performed in the QGIS application to extract the road network feature, as shown in Figure 8. After the feature selection, the road network is further preprocessed to filter only the necessary columns (features) for analysis. Table 4 below shows a random sample of the selected column from the road network data.

Table 4 Sample result of preprocessing on the line shapefile from OSM data

osm_id	oneway	name	highway	Road_lt_m
1000447998	yes	null	secondary_link	13
1000447999	no	Θεσσαλονίκης - Κιλκίς	secondary	58
1020353755	yes	Κωνσταντίνου Καραμανλή	primary	48
1020644502	yes	null	primary_link	26
1020673255	yes	Μοναστηρίου	primary	9

From Table 4, the road length feature was calculated, showing the length of a road segment that constructs an entire road. For example, in Figure 9 below, the Egnatia street is one of the longest roads in Thessaloniki, as it has 66 road segments.

Figure 9: Egnatia street

Bus Station

The exact process was applied to the points shapefile data to obtain bus stop features. In the beginning, the data contains information regarding bus stops and all other geograph-

ical features in point forms, i.e., rest areas, give way signs, traffic signals, crossings, and other features. Therefore, a spatial query was done to obtain only the bus stop feature. Figure 10 shows the result of the feature selection.

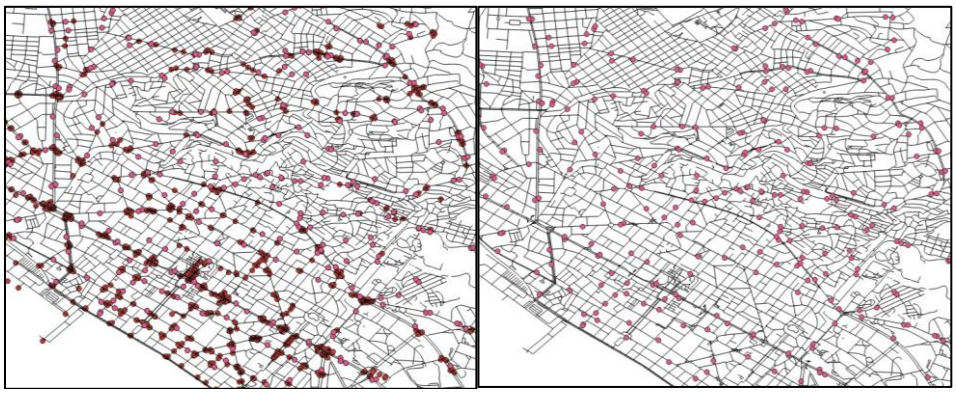

Figure 10: The subset of point shapefile (left) and feature selection result of a bus stop (right)

Moreover, some unnecessary columns were dropped, leaving only the id and name column. These columns are needed to validate the following process of joining bus stops and road network features.

Feature Join

A feature join process was applied after the road network, and bus stop features were ready. This process determines whether a bus stop occurs in the segments. Therefore, a tool that analyzes the closest point from a line is needed. Using Distance to Nearest Hub (points), one of QGIS 3.16 processing tools, a new column feature containing the distance between the closest point and a line was created. Moreover, the distance was checked numerically and visually to validate the process. Table 5 shows the result of this process.

Table 5 Sample result of joining road network and bus stop in OSM data

osm_id	oneway	name	highway	road_lt_m	nrst_bus	bus_stop	stop_dist
112282624	yes	Εγνατία	primary	233	5340987236	yes	3.416
469586088	yes	Γρηγορίου Κολωνιάρη	secondary	197	3115198174	yes	2.983
47019446	yes		cycleway	380	5405630104	yes	2.195
13769290	yes		secondary_link	41		no	
14159818	yes	Καπετάν Άγρα	tertiary	337		no	

Note that for further processing, the ‘stop_dist’ feature from Table 5, which tells the distance between the bus stop to the closest road segment, was not included in creating the prediction model.

3.3.3 Joining FCD and OSM Data

At this stage of preprocessing, OSM data which is now only a road network shapefile containing the features as depicted in Table 5, is then joined to preprocessed FCD data to give it more features for creating the prediction model. The process was done in PostgreSQL software with PostGIS extension to import OSM data into the database. Joining these two datasets was done using the code fragment in Appendix B, and the sample result can be seen in Table 6.

Table 6 Sample result of joining preprocessed FCD and OSM data

date_time	osm_id	speed_ kmph	hours	mins	n_day	stores	Road_lt _m	Bus_ stop
2018-01-01 00:00:00	14904476	33	0	0	MONDAY	CLOSED	96	No
2018-01-01 00:00:00	174486699	26	0	0	MONDAY	CLOSED	146	No
2018-01-01 00:00:00	302472924	28	0	0	MONDAY	CLOSED	88	No
2018-01-01 00:15:00	14904476	20	0	15	MONDAY	CLOSED	96	No
2018-01-01 00:15:00	35355049	41	0	15	MONDAY	CLOSED	165	No

Note that some columns do not appear in Table 6 for visualization purposes. These columns are ‘link_dir,’ ‘months,’ and ‘highway.’ Therefore, there are 14 features in the input data. However, some of these columns are eventually not included in creating the prediction model, which is discussed more in the following subsection. For pilot processing, some road segments were selected according to the results from the shortest route analysis using network analysis tools in QGIS. The results of this road selection can be seen in Figure 11 and Figure 12.

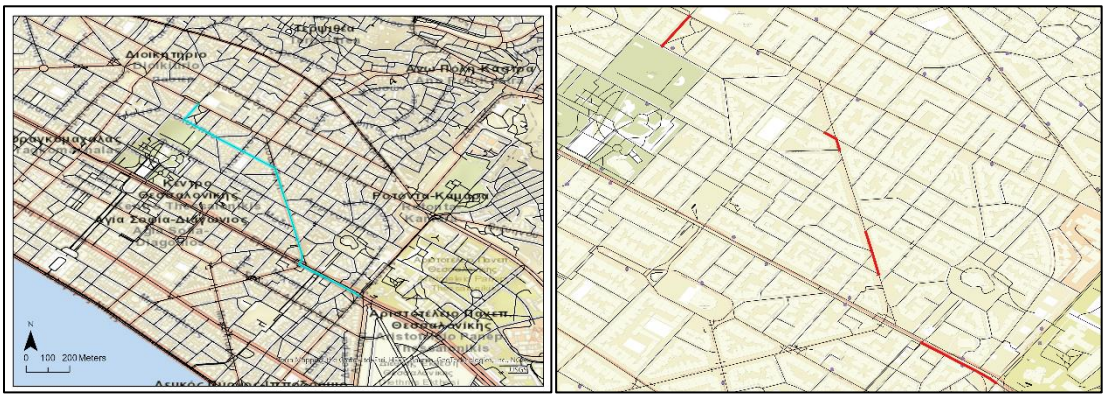

Figure 11: Selected Road based on the shortest route from Egnatia Street to Aristotelous Street (left) and the only available FCD data on the selected route shown in red lines (right)

However, after inspecting these preprocessing results, it is found that not all road segments in OSM data have the corresponding FCD data. Therefore, some problems arise when selecting some road segments, as depicted in Figure 11 and Figure 12. Because the data gap in Figure 11 was quite big, another road was selected for analysis: the Tsimiski street, as pictured in Figure 13.

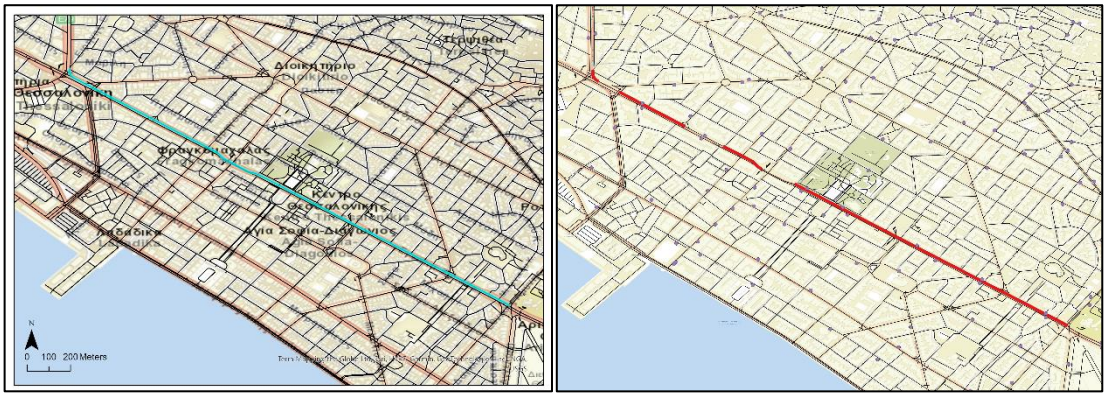

Figure 12: Selected Road based on the shortest route from Egnatia Street near East Thessaloniki Byzantine Walls to the Junction that leads to Monastiriou or Lagkada (left) and the only available FCD data on the selected route shown in red lines (right)

Figure 13: Selected Road segments on Tsimiski street

Hence, the subset of joined data for analysis was established in some parts of the Egnatia and Tsimiski street. The selected data were varied, consisting of up to four years of data to see how the model is affected by the number of data fed as an input. It also varied based on the number of segments selected with and without a gap between segments to study how it affects the model accuracy. The selected data are visualized in Figure 12 and Figure 13. Note that all preprocessed data were saved into CSV files for further processing.

3.4 Data Processing

This stage explains how the selected data was processed and evaluated. Several ML algorithms were tested to see which algorithms were most appropriate for the data. Some evaluation parameters were also applied to find which algorithm gives the lowest error while comparing the results achieved in [9].

3.4.1 Applying Machine Learning Algorithms

As already discussed in the Theoretical Framework section, because the feature that wants to be predicted in this study is a speed value, which is continuous, regression ML algorithms were used. This process was done using Python 3.8.8 programming with the scikit learn 1.0.2 library [27]. The IDE used for this process is Jupyter Notebook in Microsoft Visual Studio.

Initially, the selected data were read using Pandas' library and converted into a data frame. Then, the input data was inspected to drop the missing values. After that, the prediction target column—speed_kmph, was selected.

```
 # Read the data
df = pd.read_csv('selected_route_18.csv', sep=',')
df.dropna(axis=0, how='any', inplace=True) ...
 # Split df into X and y
 # Selecting the prediction target (label)
 y = df.speed_kmph
```

Then, the features that act as the predictor in creating the model are selected. For this analysis, the selected features are osm_id, hours, mins, day, month, year, stores, road_lt_m, and bus_stop. Note that link_dir and highway were also added as features in the beginning. However, because these features only contain one unique value, they will not affect the prediction model; hence these features were not included. Because some of the features were still in categorical values, label and one hot encoding were performed to transform these features into numerical values.

```
# Selecting the 'features'
 data_features = ['osm_id', 'hours', 'mins', 'n_day_n', 'n_month_n',
                  'stores_n','road_lt_m', 'bus_stop_n']
 X = df[data_features]
```

Label Encoder

This encoder can be applied to numerical and non-numerical labels as long as it is comparable to numerical labels [17], [18]. It normalized labels so the feature will contain only values between 0 and n_classes-1. Table 7 shows the result of this encoding.

```
# Convert all the features with label encoder
df['n_time_n'] = LabelEncoder().fit_transform(df['n_time'])
df['n_day_n'] = LabelEncoder().fit_transform(df['n_day'])
df['n_month_n'] = LabelEncoder().fit_transform(df['n_month'])
df['stores_n'] = LabelEncoder().fit_transform(df['stores'])
df['highway_n'] = LabelEncoder().fit_transform(df['highway'])
df['bus_stop_n'] = LabelEncoder().fit_transform(df['bus_stop'])
```

Table 7 Value comparison before and after applying label encoder to categorical features

Before Label Encoder	After Label Encoder
time values: ['00:00' '00:15' '00:30' '00:45' '01:00' '01:15'... '23:45'] day values: ['MONDAY ' 'TUESDAY ' 'WEDNESDAY' 'THURSDAY ' 'FRIDAY ' 'SATURDAY ' 'SUNDAY '] month values: ['JANUARY ' 'FEBRUARY ' 'MARCH ' 'APRIL ' 'MAY ' 'JUNE ' 'JULY ' 'AUGUST ' 'SEPTEMBER' 'OCTOBER ' 'NOVEMBER ' 'DECEMBER '] stores values: ['CLOSED' 'OPENING' 'OPEN' 'CLOSING'] bus_stop values: ['no' 'yes']	time values: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 16 18 19 ... 95] day values: [1 5 6 4 0 2 3] month values: [4 3 7 0 8 6 5 1 11 10 9 2] stores values: [0 3 2 1] bus_stop values: [0 1]

One Hot Encoder (OHE)

This encoder will convert categorical features as a one-hot numeric array [30]. The result of this process can be seen in Table 8 below. The code using OHE can be seen below.

```
# Convert features with one hot encoder
# Emit highway and bus_stop for one segment process ['n_day',
'stores', 'n_month', 'highway', 'bus_stop']
column_trans = make_column_transformer(
    (OneHotEncoder(), ['n_day', 'stores', 'n_month', 'bus_stop']),
    remainder='passthrough'
X = column_trans.fit_transform(X)
```

Table 8 Value comparison before and after applying one hot encoder to categorical features

Before One Hot Encoder	After One Hot Encoder
time values: ['00:00' '00:15' '00:30' '00:45' '01:00' '01:15'... '23:45'] day values: ['MONDAY ' 'TUESDAY ' 'WEDNESDAY' 'THURSDAY ' 'FRIDAY ' 'SATURDAY ' 'SUNDAY '] month values: ['JANUARY ' 'FEBRUARY ' 'MARCH ' 'APRIL ' 'MAY ' 'JUNE ' 'JULY ' 'AUGUST ' 'SEPTEMBER' 'OCTOBER ' 'NOVEMBER ' 'DECEMBER '] stores values: ['CLOSED' 'OPENING' 'OPEN' 'CLOSING'] bus_stop values: ['no' 'yes']	array ([[0., 1., 0., ..., 0., 0., 72.], [0., 0., 0., ..., 14., 45., 72.], [0., 1., 0., ..., 0., 0., 42.], ..., [0., 0., 0., ..., 0., 0., 42.], [0., 0., 0., ..., 0., 0., 72.], [0., 0., 0., ..., 0., 0., 542.]])

However, some sources state that Label Encoder should not be used for categorical features with more than two values; it should also be used only for target values [31]–[33]. Nevertheless, the feature encoding process still uses both ways to see how it affects the final prediction. After encoding categorical values and selecting both features and target column, the data was split into training and test dataset.

```
# Train-test split
train_X, val_X, train_y, val_y = train_test_split(X, y,
                                train_size=0.7, shuffle=False,
                                   random_state=1)
# Scale X
scaler = StandardScaler()
scaler.fit(train_X)
train_X = pd.DataFrame(scaler.transform(train_X), index=train_X.index,
          columns=train_X.columns)
val_X = pd.DataFrame(scaler.transform(val_X), index=val_X.index,
          columns=val_X.columns)
```

Because the data input for creating the model is still not standardized, the StandardScaler utility from the Scikit-learn processing module was applied to training and test features. This process was done because many ML algorithms require individual features to be standard-normally distributed data [34]. Table 9 shows the results after applying standard scaling to the dataset.

Table 9 Value comparison before and after applying standard scaler to the features

Before Standard Scaler	After Standard Scaler
Variance before scaler: osm_id 1.929929e+16	Variance after scaler: osm_id 1.000007

Table 9 Value comparison before and after applying standard scaler to the features (Continued)

Before Standard Scaler	After Standard Scaler
hours 4.783168e+01 mins 2.815786e+02 n_day_n 4.018732e+00 n_month_n 1.012323e+01 stores_n 1.115074e+00 road_lt_m 1.164300e+04 bus_stop_n 1.306943e-01	hours 1.000007 mins 1.000007 n_day_n 1.000007 n_month_n 1.000007 stores_n 1.000007 road_lt_m 1.000007 bus_stop_n 1.000007
Variance before scaler: 920443936122033.5	Variance after scaler: 0 1.000045 1 1.000045 2 1.000045 3 1.000045 4 1.000045 5 1.000045 6 1.000045 ...

After those processes, ML algorithms can be applied to the training dataset to produce the models. Several ML algorithms were selected for this study to find suitable algorithms for predicting the speed value.

```
models = {
    "                      Linear Regression": LinearRegression(),
    "                    K-Nearest Neighbors": KNeighborsRegressor(),
    "                         Neural Network": MLPRegressor(),
    "Support Vector Machine (Linear Kernel)": LinearSVR(),
    "   Support Vector Machine (RBF Kernel)": SVR(),
    "                           Decision Tree": DecisionTreeRegressor(),
    "                           Random Forest": RandomForestRegressor(),
    "                       Gradient Boosting": GradientBoostingRegressor()
}
for name, model in models.items():
    model.fit(train_X, train_y)
    print(name + " trained.")
```

3.4.2 Evaluating Applied Algorithms

After the prediction models were created, a performance evaluation was conducted. For the regression method, several evaluation methods can be used. Thus, for this study, the R^2 score, root mean square error (RMSE), Mean Absolute Error (MAE), and Cross-Validation method were applied to measure model performance.

```
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error
```

```
for name, model in models.items():
    val_predict = model.predict(val_X)
    print(name + " R^2 Score: {:.5f}".format(r2_score(val_y, val_predict)))
    print(name + " RMSE: {:.5f}".format(np.sqrt(mean_squared_error(val_y,
          val_predict))))
    print(name + " MAE: {:.5f}".format(mean_absolute_error(val_y, val_pre
          dict)))
```

R^2 Score

R^2 score metric shows the variation in the features that is predictable from the target feature. Therefore, it can tell how good the prediction result is, based on the proportion of total variation of results produced by the model. The metric is calculated using equation (4) below [35], [36].

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (4)$$

The value of the R^2 score ranges from 0 to 1, and this metric can be intuitively more informative than the later metrics [35].

Root Mean Square Error (RMSE)

RMSE measures the root average of the squares of residuals between two data sets, the actual and predicted values. This evaluation metric is calculated using equation (5) [36]–[38].

$$\text{RMSE}\ (y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{n_{samples}-1}(y_i - \hat{y}_i)^2}{n_{samples}}} \quad (5)$$

This metric calculates the accuracy in comparing prediction errors between different models for a particular dataset because it is scale-dependent. Thus, this cannot be used to compare the accuracy between datasets [38].

Mean Absolute Error (MAE)

MAE measures errors between two sets of data that express the same phenomenon. For this study, the comparison between labels' actual value and predicted value which calculated using equation (6) [36], [39].

$$\text{MAE}\ (y, \hat{y}) = \frac{\sum_{i=0}^{n_{samples}-1}|y_i - \hat{y}_i|}{n_{samples}} \quad (6)$$

For equations (4), (5), and (6), $\hat{y}_i$ is the predicted value and y_i is the actual speed value. This evaluation methods except R^2 score have the same unit as the predicted value. Thus, the error measurement unit for this study is km/h. Therefore, the lower the value error of RMSE and MAE (closer to 0), the model is a perfect fit for the data. On the other hand, for the R^2 score value, the closer the value to 0 means the model is not fit with the data [35], [36], [38], [39].

Cross-Validation

This technique is applied to avoid overfitting the prediction model. The process will cross-validate the model to find the best parameters through the workflow shown in [40], depicted in Figure 14 below.

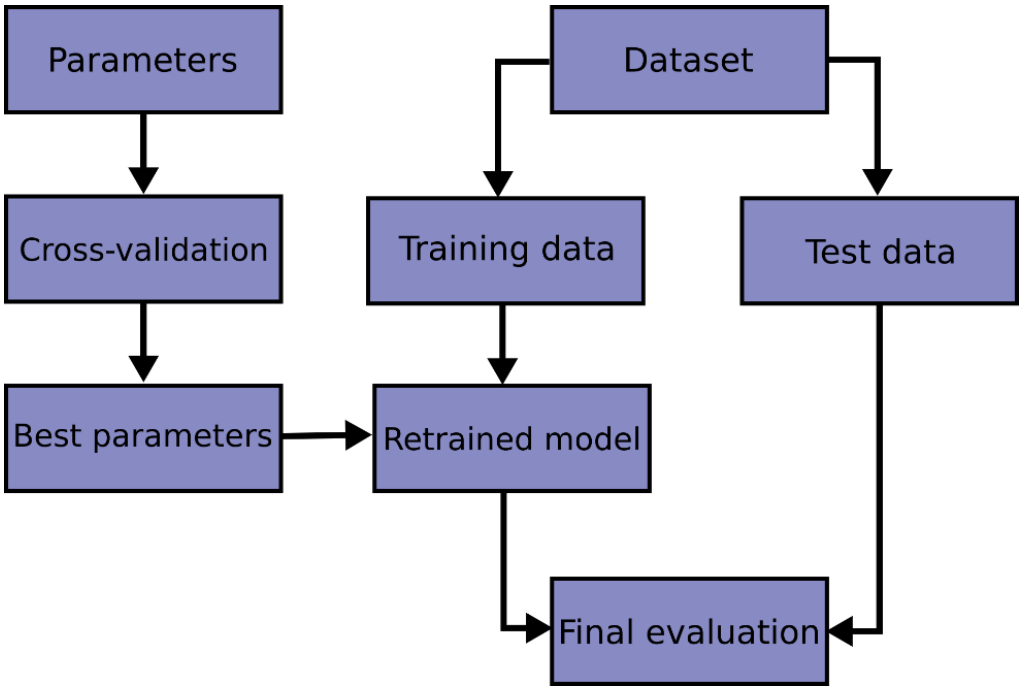

Figure 14: Cross-validation process [40]

Cross-validation in this study is used to check whether the model is overfitted and to see if the model score is correct. It is done using the k-fold, where the k value represents the number of split groups of the smaller training set. Then, the model will be trained using k-1 of the folds, and the resulting model is validated on the remaining part of the data. Therefore, overfitting can be avoided as the process of tweaking the settings (hyperparameters) for the prediction model is already put in the test set out for final validation. Other than avoiding overfitting, the advantage of using this technique is that the process does not waste too much data. However, this process can be computationally expensive [36], [37]. The following code shows how cross-validation is implemented.

```
# Implementing Cross Validation techniques
from sklearn.model_selection import cross_val_score

scores = cross_val_score(data_model, train_X, train_y, cv=10)
scores
```

4 Results

This chapter presents the results from the processing step explained in the previous section. Firstly, this part assesses how the model results compared to the previous study. After that, it compares the results between the selected road segments and the results between the number of data selected. It also covers the problem regarding the availability of FCD data throughout the OSM data and how the result might affect the network-wide prediction.

4.1 Assessment of Prediction Results

This section will present the prediction result from several scenarios applied to the data. First, the prediction results were assessed using the scenario mentioned in [9], which are:

1. Speed prediction at random dates and times on randomly selected roads,
2. Speed prediction at eight consecutive times on random dates and randomly selected roads, and
3. Speed prediction for 24 hours on random dates and randomly selected roads.

These scenarios were tested only on Tsimiski street and only at one road segment to simplify the processing. It was selected as the previous study did not mention the exact road segment that was selected, and through this study, the Tsimiski road segment data is found to have good quality data. Moreover, the data on Tsimiski was selected between January to March 2019. It was done to select the data within the time frame of the data used in [9]. The selected data on road segment 17665188 has 14,381 records, and after interpolation was applied to fill the gap between the data, the total records became 14,495 records. The following Table 10 shows the R^2 score for each model.

Table 10 R^2 score for each model

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
R^2 Score	Tsimiski Street (Test Data)	0.27	0.58	0.65	0.27	0.63	0.63	0.64	0.64
	Tsimiski Street (Training Data)	0.27	0.71	0.69	0.27	0.63	0.66	0.66	0.65

The model was trained on the data two weeks prior to the data for assessment. In this case, the data from 1st January until 14th February 2019 was selected for training data. For these scenarios, the input features that were used are hours, minutes, days, and stores feature. Several features were dropped because there was only one unique value when only one road segment was selected. From Table 10, the training and test data scores were compared to ensure that the model does not overfit the training sample. Note that these results were accomplished after tuning the hyperparameter for each model. The code for these comparisons can be found in Appendix D.

4.1.1 Random Dates and Times

For this scenario, 35 records with random times and dates were selected between 15th and 28th February 2019. Figure 15 shows the RMSE comparison for all models.

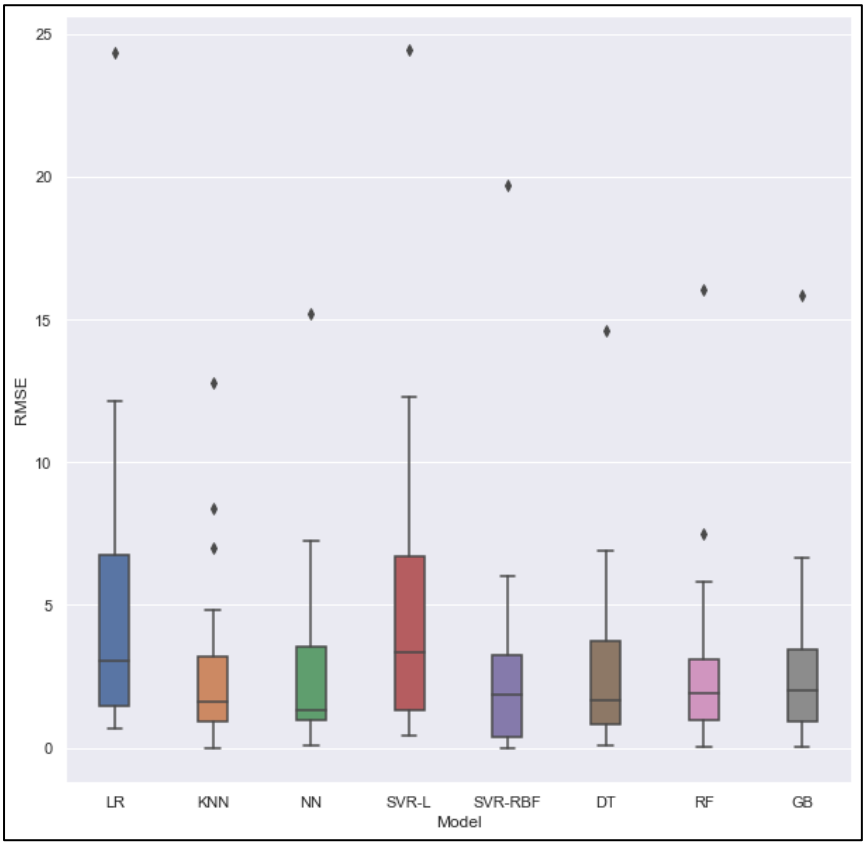

Figure 15: RMSE comparison between models when assessing the scenario on random dates and times

Moreover, Table 11 shows the MAE in km/h, which expresses the average model prediction error from testing 35 random selection of times and dates.

Table 11 Average MAE score for all models at 35 random dates and times

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	4.74	2.41	2.48	4.68	2.56	2.60	2.62	2.67

Figure 15 and Table 11 show that LR and SVR linear models can be ruled out of the competition since those models have the highest MAE and RMSE score. In contrast, the model other than those two is not significantly different because the difference between these models is in the fraction of 0.07 to 0.26 km/h. From this assessment, KNN is the best model according to the MAE score alone. However, if the results from Table 10 are considered, the KNN model was overfitted as the model score on the training data is way higher than the test score. Therefore, the NN model is the best, followed by DT, RF, and GB models.

4.1.2 Eight Consecutive Times on Random Dates

Two sets of data were selected, one is on 16th February (weekday), and the other is on 22nd February (weekend). The selected data were two hours long, which eventually contained eight-step (eight consecutive times). These periods were selected to represent the activity of people and stores. It means the selected periods will represent the street when the activity is high and when a good prediction might be necessary. This selection idea was based on [9]; however, the selected date differed. The following Figure 16 and Figure 17 show the comparison between each model in each selected period.

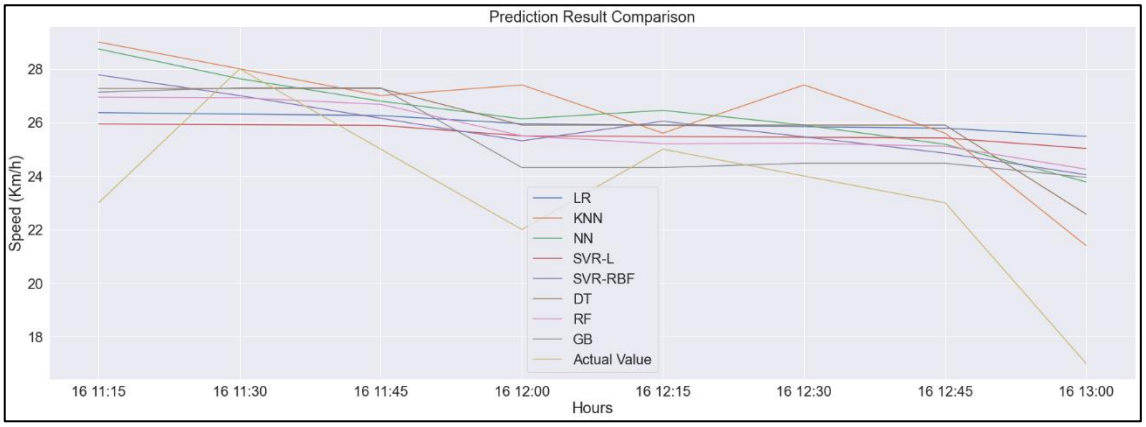

Figure 16: Result comparison between the actual and predicted value for each model on 16th February from 11:00 to 13:00

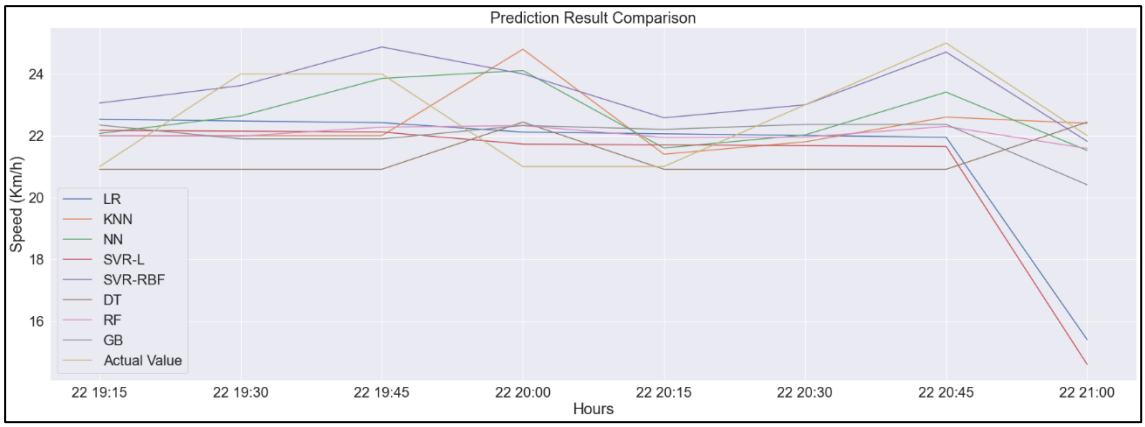

Figure 17: Result comparison between the actual and predicted value for each model on 22nd February from 19:00 to 21:00

From Figure 16, it appears that both linear models (LR and SVR linear) can predict the actual value. However, Figure 17 shows the other way around, where the model could predict the first and the following values before the last point. This trend can also be seen in Figure 16, where these models could have a decent prediction initially and fail afterward. Therefore we can assume that these models are not consistent and suitable for traffic prediction problems. Additionally, the overall results for the other models can be identified through the overall MAE and RMSE scores in Table 16.

Table 12 Overall MAE and RMSE values for each model from two tests

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	16th February	3.03	3.05	3.04	2.72	2.71	2.80	2.62	2.38
	22nd February	2.18	1.65	1.17	2.30	1.05	1.80	1.40	1.62
RMSE	16th February	3.80	3.66	3.70	3.51	3.40	3.22	3.36	3.15
	22nd February	2.82	1.97	1.45	3.11	1.45	2.30	1.55	1.72

Through Table 12, the other ML model, except the linear model, performs very well and consistently when the previous scenario is considered. However, consider that the KNN model was overfitted during this assessment. Therefore, the validity of the result from it is still questionable. The table also shows that the models that perform better among the good models are the tree models (DT and RF) and the GB model.

4.1.3 24 Hours Time Window on Random Dates

The assessment for this scenario was conducted on two random dates. The first test was selected on the 24th of February, while the second test data was on the 28th of February. The following Figure 18 and Figure 19 show the results of these two tests, respectively.

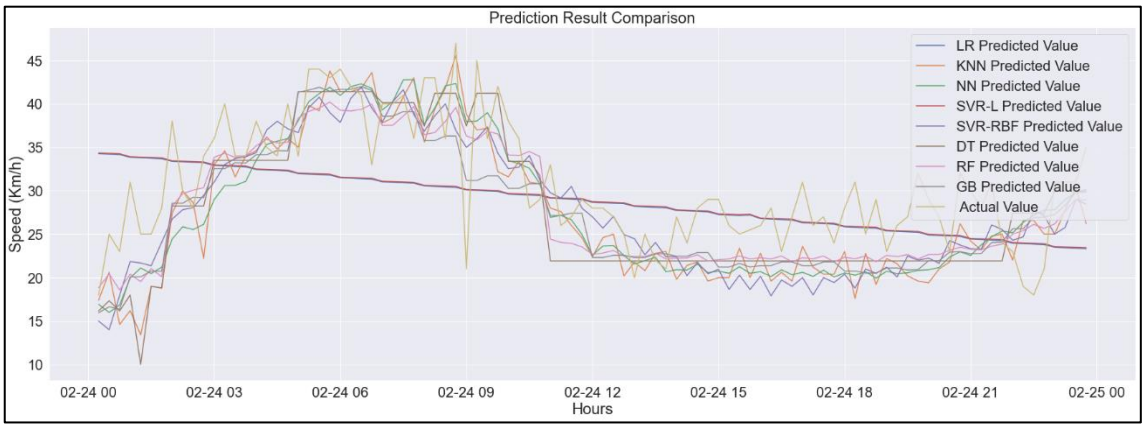

Figure 18: Prediction result comparison between each model and the actual value on the 24th of February

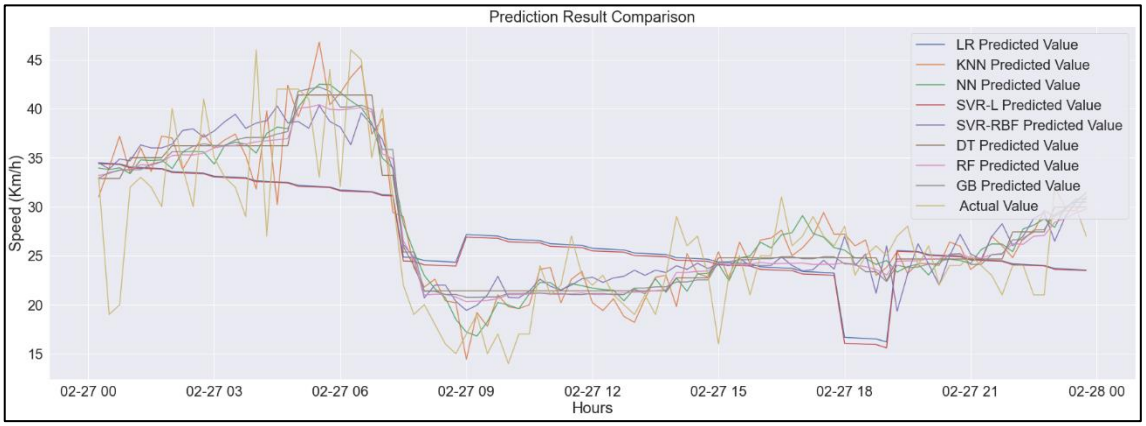

Figure 19: Prediction result comparison between each model and the actual value on the 28th of February

Note that the results for this scenario and the scenario for eight consecutive periods were made through simultaneous prediction, meaning that the prediction was not performed iteratively through each row of the test data frame. In contrast, the previous study by [9] performs the prediction iteratively per step (15 minutes). This study did not follow this since it will cause performance issues when more extensive datasets, i.e., more than 1000 rows, are predicted [41].

Again, from Figure 18 and Figure 19, the linear model does not work well as those models predict a complex problem with linear approaches. However, the other models other than those two perform very well by following the trend of the actual value.

Moreover, KNN and NN models can outperform SVR, GB, DT, and RF, as from those figures, both models can predict closer to the actual value. Therefore, to better understand how well these models predict the value, both MAE and RMSE values should be considered. The following Table 13 shows the MAE and RMSE values from each model.

Table 13 Overall MAE, RMSE, and R^2 score values for each model from two tests on 24 hours prediction scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	24th February	5.14	4.91	4.92	5.13	4.84	4.75	4.38	4.67
	28th February	5.04	3.47	3.35	5.02	3.86	3.45	3.43	3.47
RMSE	24th February	6.58	6.12	5.85	6.55	5.74	5.81	5.17	5.61
	28th February	6.41	4.95	4.47	6.40	4.89	4.46	4.38	4.48
R^2 Score	24th February	0.10	0.22	0.29	0.11	0.32	0.30	0.44	0.35
	28th February	0.29	0.57	0.67	0.29	0.59	0.65	0.67	0.65

From Table 13, in contrast to the visual analysis of previous figures, both KNN and NN were outperformed by SVR, tree models, and GB models. According to MAE, RMSE, and R2 score values, the best model is the RF model, followed by the GB model. Moreover, assuming the R^2 score can detect the quality of the input data, the data on 24th February is considered to have bad quality. Therefore, by looking at MAE and RMSE behavior on the good models, NN can perform better than the other models when the data quality is good. On the other hand, the other models, except KNN, have a pretty stable performance regardless of the quality of the input.

4.2 Assessment for Network-Wide Analysis

After assessing the model, several scenarios were tested to study the effect of bus stop features and find the best scenario for predicting the whole city road network. The scenarios are as follows:

1. One month of data with only one random road segment,
2. One month of data with multiple road segments analyze at once,
3. Multiple months of data with only one random road segment, and
4. Multiple months of data with multiple road segments analyze at once.

These scenarios were applied to both Tsimiski and Egnatia street. Figure 20 shows the speed values on each street from 2018 until 2022.

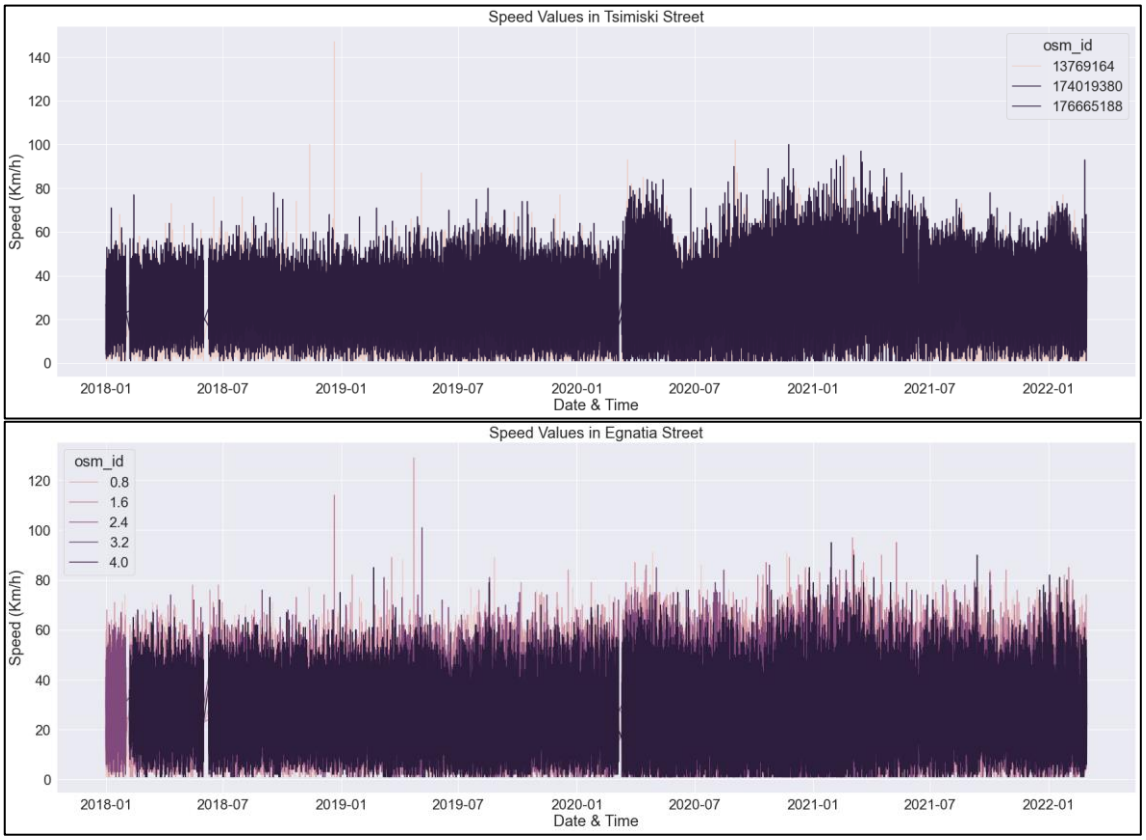

Figure 20: Speed value in Tsimiski Street (top) and Egnatia Street (bottom) from January 2018 to March 2022

Respectively, each dataset has a total of 390,922 and 907,021 records. From those records, the data were sliced according to the scenario needed. The number of selected road segments caused the difference in the number of records between these two datasets. On Egnatia data, seven road segments out of 19 were selected. On the other hand, from Figure 20, it is shown that there are only three segments selected on Tsimiski street. The reason is that several road segments do not have speed data, as already mentioned in section 3.3.3. This problem might affect the following process and be further discussed in section 5.2.

4.2.1 One Month Data – One Road Segments

This scenario aims to become a benchmark comparison for the result when multiple segments are selected. Figure 21 below shows the speed trend on each segment from each road in one month.

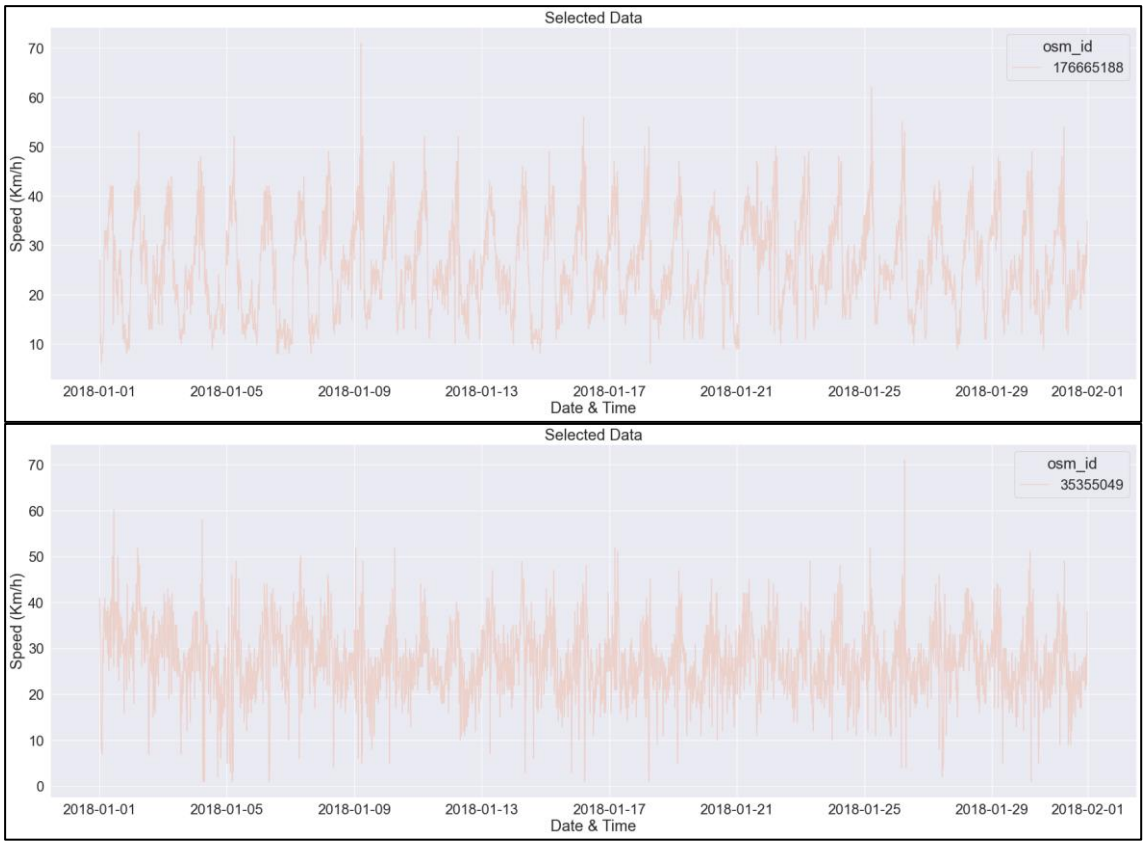

Figure 21: Speed profile of road segment 176665186 on Tsimiski street (top) Speed profile of road segment 35355049 on Egnatia street (bottom)

Daily speed patterns are depicted in those figures, so the related features will be included when creating the model. Moreover, a heatmap correlation was created for each road to determine the correlation between features and target features, as shown in Figure 22.

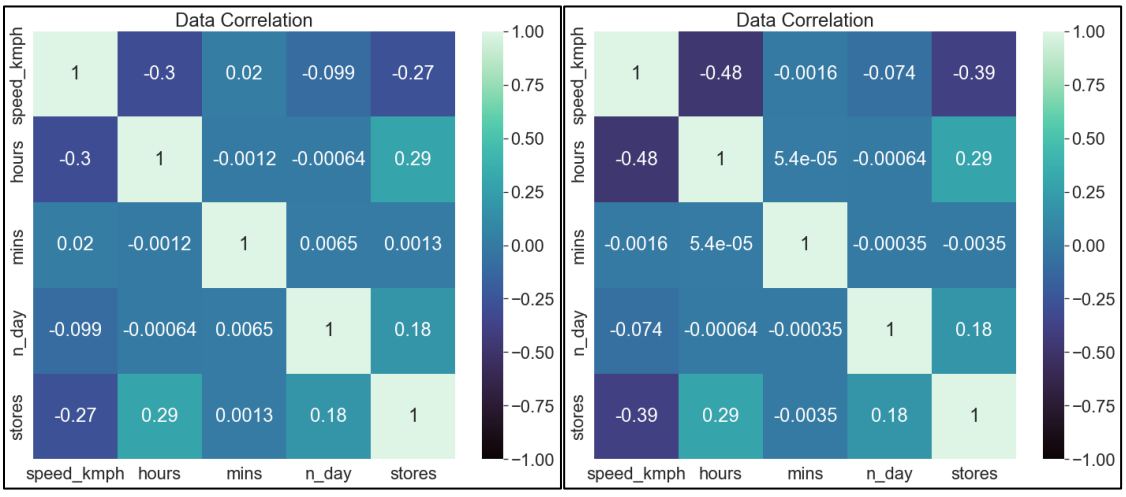

Figure 22: Tsimiski's data (left) and Egnatia's data (right) feature correlation for the first scenario

There were about 14 features from the actual data, as mentioned in section 3.3.3. Nevertheless, most of the features were dropped as several features in this scenario only have one unique value. Moreover, Figure 22 shows that both ‘hours’ and ‘stores’ features correlate enough to the speed value, while ‘mins’ and ‘n_day’ features have a very low correlation toward the speed value. However, these two features were not dropped as it will make the prediction process more challenging.

After selecting the target and input features, the train-test-split process was applied to the input feature. The training and test data's hourly trend is presented in Figure 23.

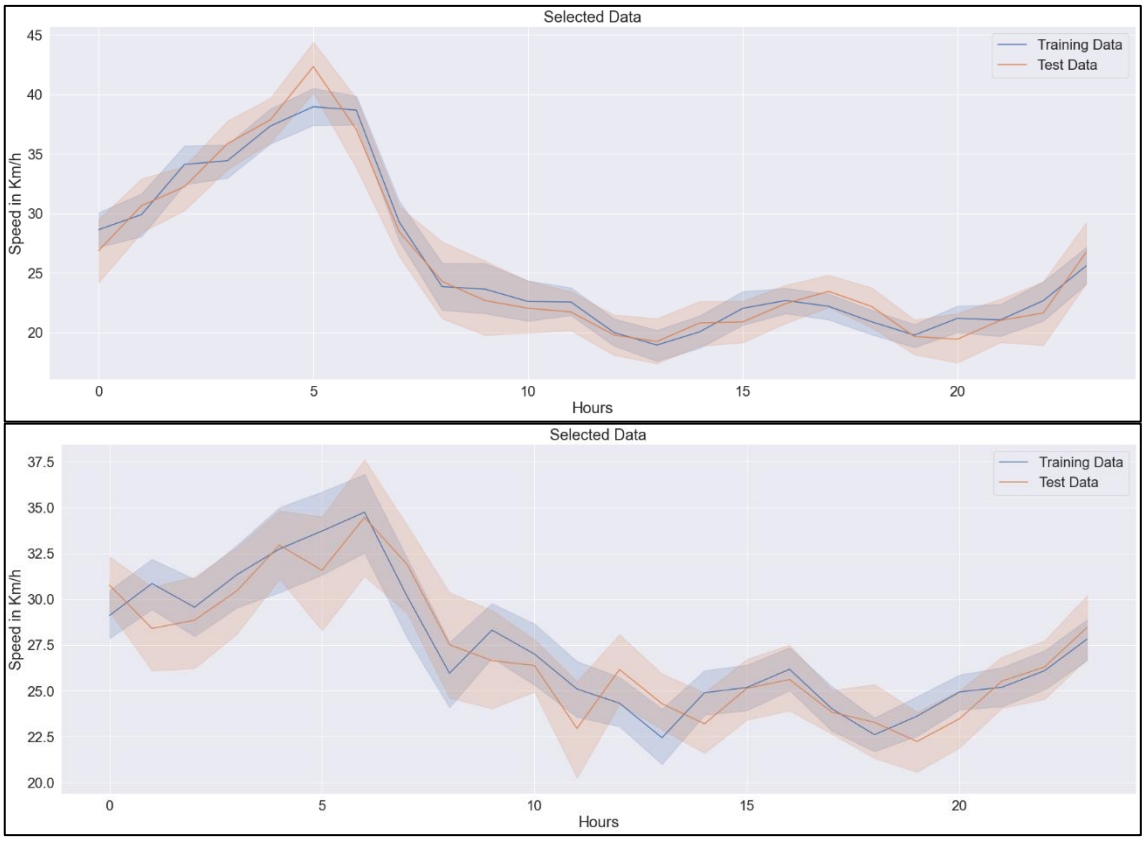

Figure 23: Hourly trend of training and test data on Tsimiski Street (top) and Egnatia Street (bottom)

Those figures show that the speed reaches its free-flow speed on each road segment around 05:00 to 06:00 before the road starts congested. The roads become free again from around 13:00 to 17:00 before congestion begins. The figure also shows sufficient samples for training and test sets for each time step. Moreover, several ML models were applied to the data after selecting the training and test data. Table 14 presents the prediction performance of this scenario.

Table 14 Prediction performance on the selected data on both streets in January 2018

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.26	4.65	5.58	6.22	4.99	4.96	4.68	4.06
	Egnatia Street	5.19	5.50	5.05	5.19	4.80	6.08	5.73	4.71
RMSE	Tsimiski Street	7.89	6.27	7.15	7.92	6.50	6.97	6.50	5.46
	Egnatia Street	6.81	7.30	6.74	6.82	6.51	8.20	7.65	6.32
R^2 Score	Tsimiski Street (Test Data)	0.29	0.55	0.42	0.28	0.52	0.44	0.52	0.66
	Tsimiski Street (Training Data)	0.30	0.70	0.42	0.29	0.52	0.76	0.76	0.67
	Egnatia Street (Test Data)	0.12	-0.01	0.13	0.12	0.20	-0.27	-0.11	0.24
	Egnatia Street (Training Data)	0.13	0.39	0.16	0.13	0.20	0.49	0.48	0.27

Result Optimization on the First Scenario

From the preceding result in this scenario, model optimization must be applied to improve each model's performance. Those models were improved by tweaking their hyperparameters until each model reached the optimal model score. The idea of this hyperparameter tweaking is to adjust the model so the model will be less complex and avoid overfitting. Thus, the following codes in Appendix C are the final tuning of its hyperparameters for this scenario. After these settings were applied to the model, the performance of each model was improved, as shown in Table 15.

Table 15 Optimized prediction performance on the first scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.26	4.61	4.28	6.24	4.29	4.13	4.15	4.06
	Egnatia Street	5.19	5.38	4.76	5.19	4.72	4.71	4.70	4.71
RMSE	Tsimiski Street	7.89	6.17	5.72	7.89	5.69	5.63	5.56	5.45
	Egnatia Street	6.81	7.20	6.40	6.81	6.46	6.33	6.35	6.32
R^2 Score	Tsimiski Street (Test Data)	0.29	0.56	0.63	0.29	0.63	0.64	0.66	0.66
	Tsimiski Street (Training Data)	0.30	0.71	0.69	0.30	0.65	0.66	0.67	0.67
	Egnatia Street (Test Data)	0.12	0.02	0.22	0.12	0.21	0.24	0.24	0.24
	Egnatia Street (Training Data)	0.13	0.39	0.24	0.13	0.24	0.24	0.26	0.28

By comparing the results from Table 14 and Table 15, it can be concluded that the GB model can produce the best model without tuning its hyperparameter. In contrast, the other model should be tuned to reach the score achieved by the GB model without hyperparameter tuning. Therefore, the most accurate model must be selected if this scenario is deployed. Therefore, Figure 24 below shows the actual and predicted values' hourly and daily comparison results on Tsimiski street, while Figure 25 and Figure 26 are the results on Egnatia street.

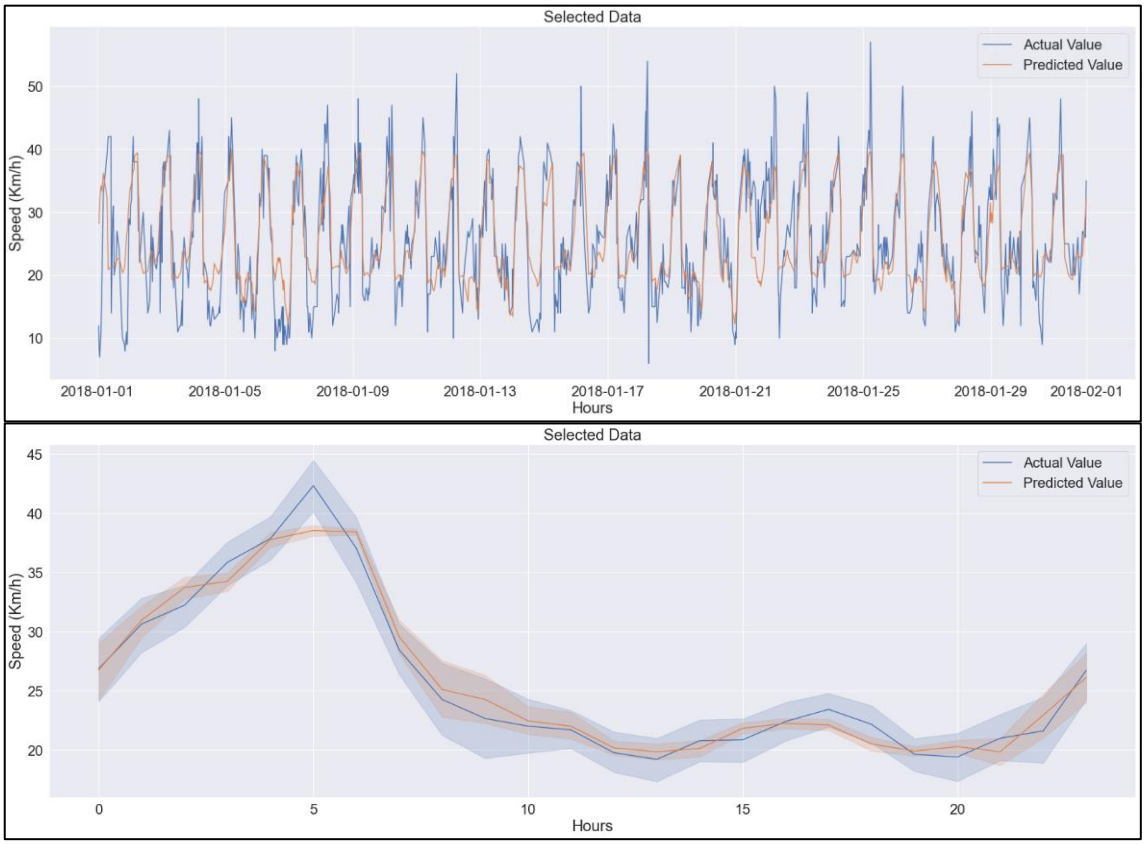

Figure 24: Daily comparison (top) and hourly comparison (bottom) between the actual and predicted value on Tsimiski street

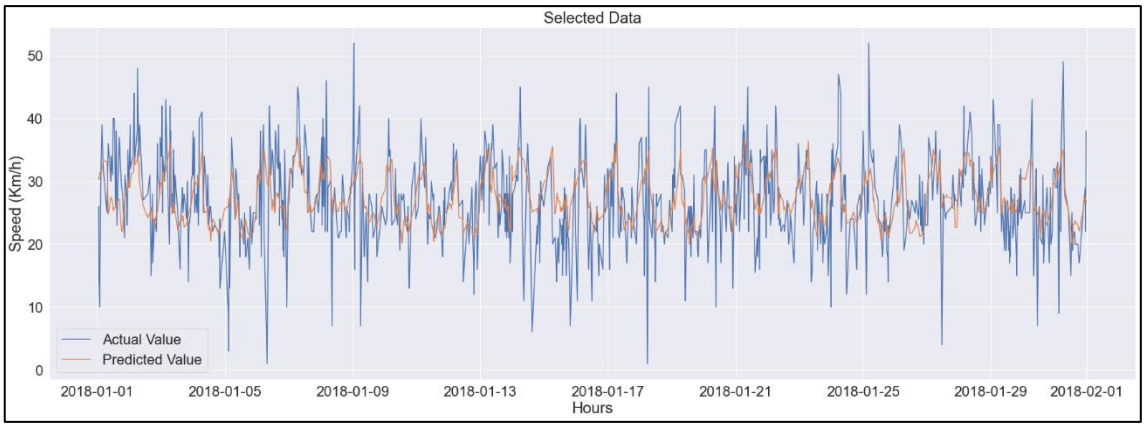

Figure 25: Daily comparison between the actual and predicted value on Egnatia street

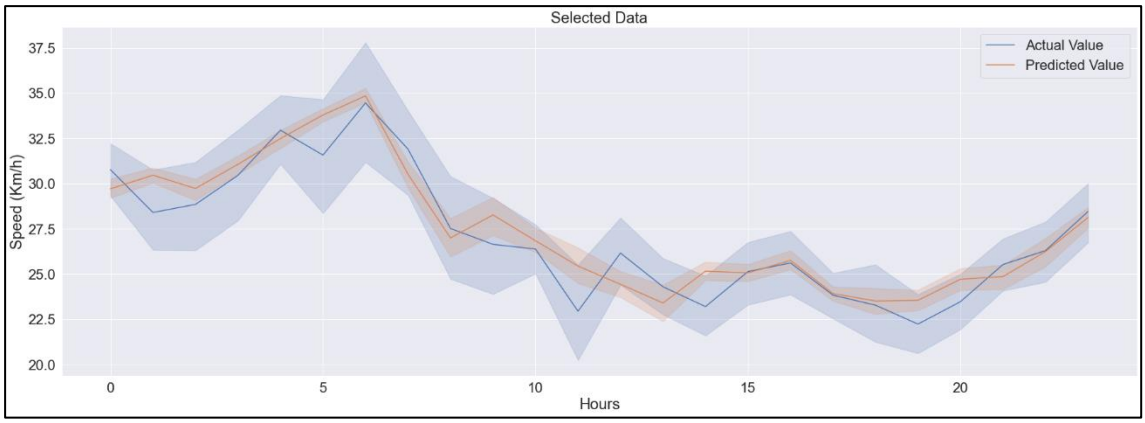

Figure 26: Hourly comparison between the actual and predicted value on Egnatia street

4.2.2 One Month Data – Multiple Road Segments

This scenario assesses the prediction performance when multiple road segments are selected. The following Figure 27 shows the data selected for this scenario.

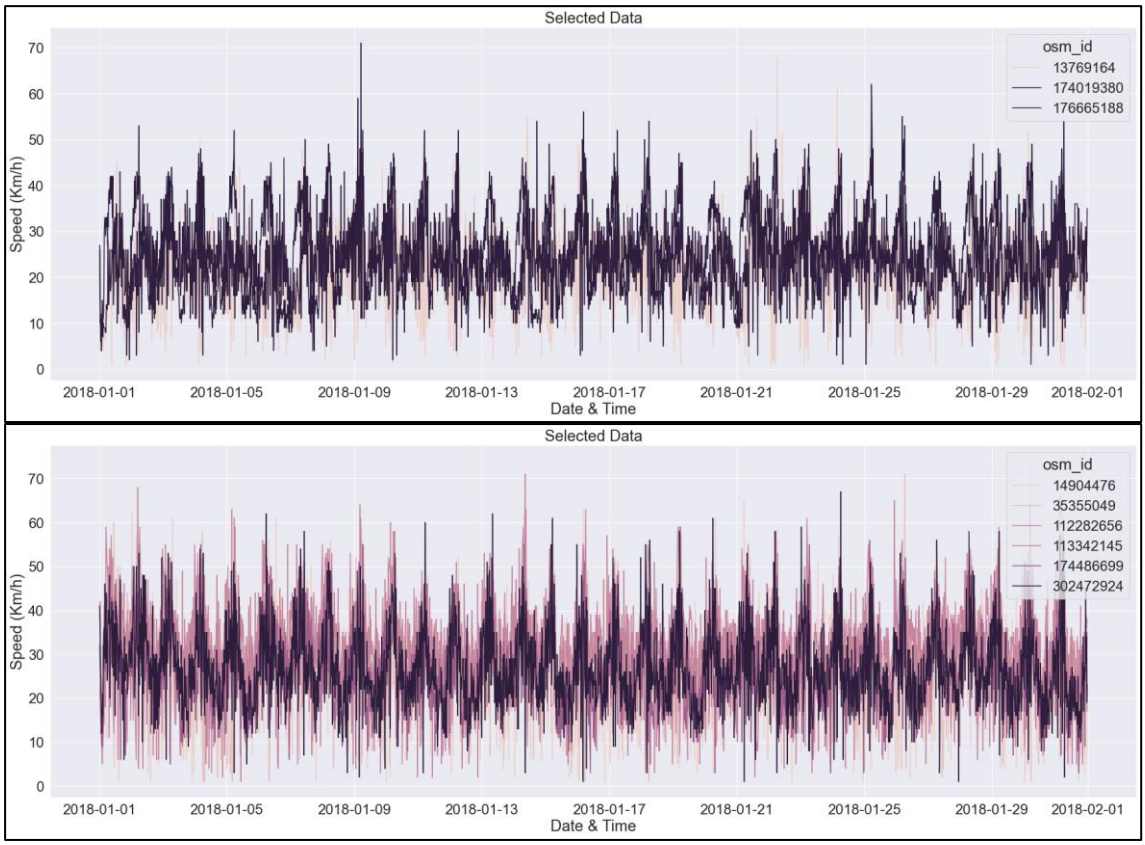

Figure 27: The selected data on Tsimiski Street (top) and Egnatia Street (bottom) for the second scenario

The idea of this scenario is that traffic speed prediction on each segment can be performed faster if the process can be performed for the whole network-wide simultaneously. The process will be faster than predicting each segment one at a time. Therefore,

the viability of this scenario could be assessed, and the factor that reduces prediction accuracy can be investigated. This scenario also tries to assess the prediction results when new features such as the road length and bus stop are added as input features. Figure 28 shows the feature correlation between the speed value and input features.

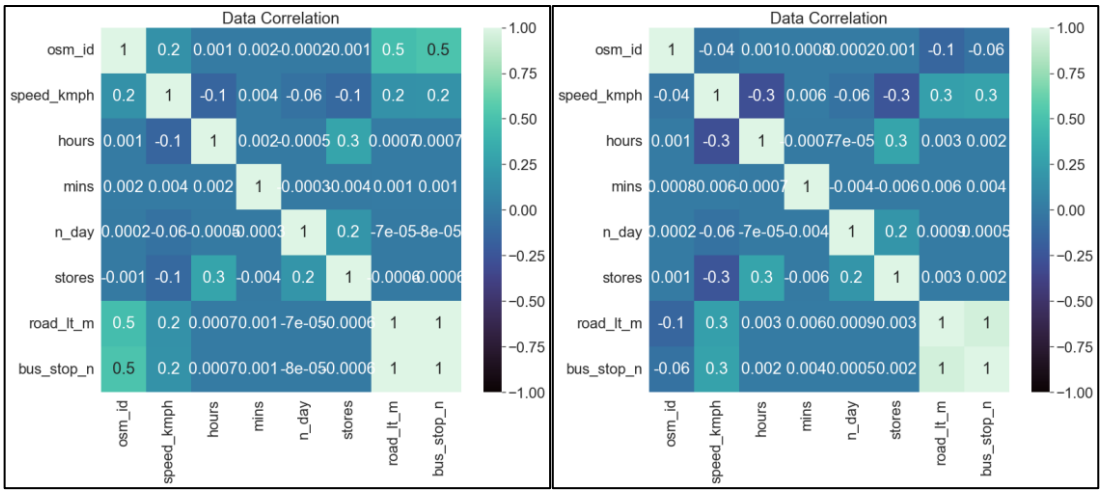

Figure 28: Tsimiski's data (left) and Egnatia's data (right) feature correlation for the second scenario

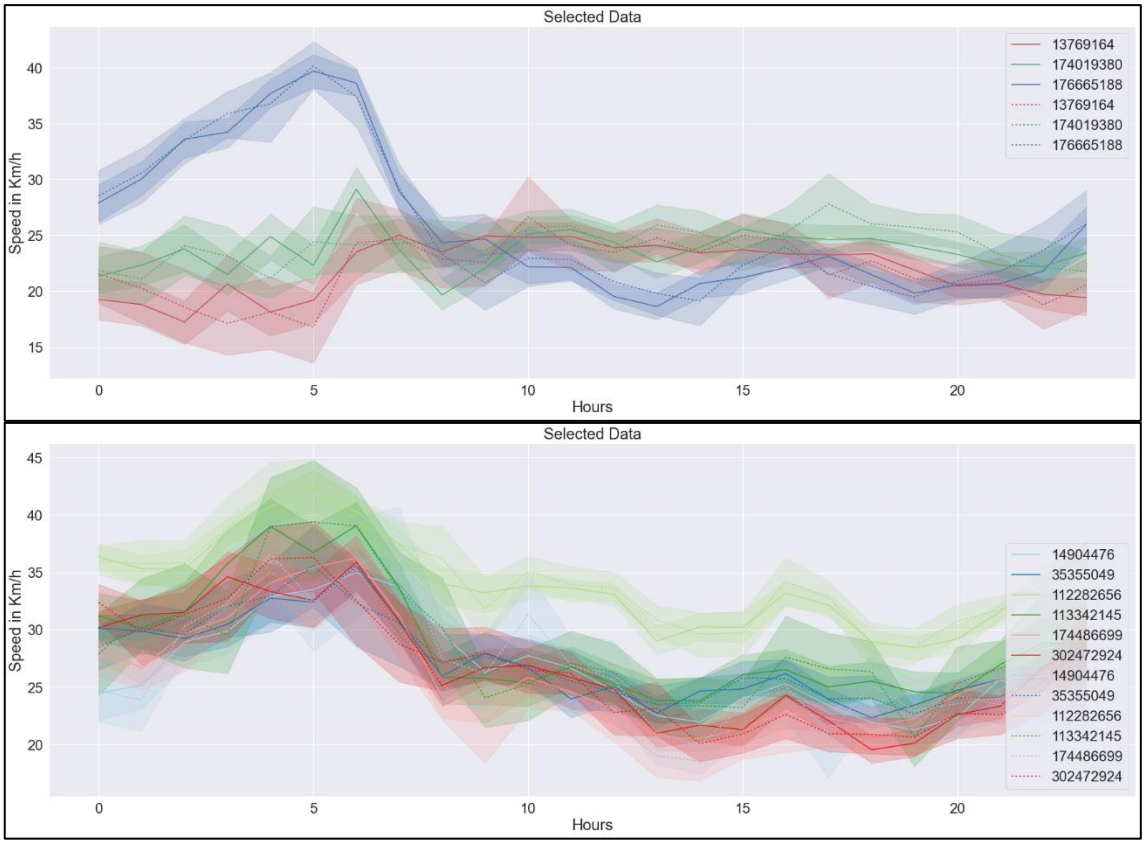

Figure 29: Hourly trend of training and test data on Tsimiski Street (top) and Egnatia Street (bottom) for each road segment

From Figure 28, road length and bus stop positively correlate with the speed value, while hours and store features negatively correlate with the speed value. On the other hand, the 'mins' and 'n_day' features have a little correlation with the target value. However, these features were still included as the input since only a few input features were in the dataset. Moreover, the training and test dataset was then selected after investigating the correlation between the target and input features, which the selected data is shown in Figure 29.

The selected data were then trained and tested on eight ML models similar to the previous scenario. The model performance can be seen in Table 16.

Table 16 Prediction performance on the selected data on both streets in the year 2018 on the second scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.34	5.47	5.34	6.34	5.42	6.10	5.78	5.13
	Egnatia Street	6.49	6.38	5.98	6.48	6.06	7.01	6.63	5.87
RMSE	Tsimiski Street	8.13	7.32	7.02	8.14	7.16	8.23	7.75	6.83
	Egnatia Street	8.59	8.56	8.05	8.60	8.16	9.47	8.91	7.93
R^2 Score	Tsimiski Street (Test Data)	0.06	0.24	0.30	0.06	0.27	0.04	0.15	0.34
	Tsimiski Street (Training Data)	0.07	0.52	0.34	0.06	0.30	0.60	0.59	0.39
	Egnatia Street (Test Data)	0.20	0.21	0.30	0.20	0.28	0.03	0.14	0.32
	Egnatia Street (Training Data)	0.20	0.47	0.31	0.20	0.29	0.59	0.57	0.33

Compared to the previous scenario, the very distinct change is the value of the R^2 score, where on Egnatia Street, the value was increased, and in contrast, on Tsimiski Street, the value was decreased. The reduction on Tsimiski Street might be related to Figure 29, where the hourly trend is quite different between one segment and another, even though the selected train and test data represent each segment. It is shown in the figure where the segment with id 174019380 and 13769164 have different speed trends in the morning compared to the road segment that was analyzed in the preceding scenario. Therefore, it may be because of the data quality issue, which will reduce the overall R^2 value. However, looking at the value of R2 on Egnatia Street may indicate that predicting multiple segments is viable, and adding more features for the training will increase the R^2 score value as long as the quality of the data input is reliable.

Aside from the change in the R^2 value, according to Table 16, the NN and GB models outperform the other models by having lower error (MAE and RMSE) values and higher R^2 scores. Note that these models can achieve this result without hyperparameter tuning. However, by comparing Table 15 and Table 16, this scenario does not perform better than the previous one as the error values increase.

Result Optimization on the Second Scenario

These models were optimized to see whether there would be any significant improvement if proper hyperparameters were set. Table 17 shows the optimized results from each model in this scenario.

Table 17 Optimized prediction performance on the second scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.35	5.53	5.13	6.33	5.16	5.16	5.21	5.13
	Egnatia Street	6.49	6.42	5.91	6.49	5.97	6.01	6.05	5.87
RMSE	Tsimiski Street	8.14	7.37	6.86	8.12	6.88	6.87	6.89	6.83
	Egnatia Street	8.59	8.58	7.98	8.60	8.08	8.09	8.13	7.93
R^2 Score	Tsimiski Street (Test Data)	0.06	0.23	0.33	0.06	0.33	0.33	0.33	0.34
	Tsimiski Street (Training Data)	0.07	0.51	0.43	0.06	0.40	0.39	0.37	0.39
	Egnatia Street (Test Data)	0.20	0.20	0.31	0.20	0.29	0.29	0.28	0.32
	Egnatia Street (Training Data)	0.20	0.47	0.32	0.20	0.34	0.30	0.29	0.33

By looking at Table 16 and Table 17, the GB model achieved optimal results without hyperparameter tuning. At the same time, the NN model improved insignificantly after hyperparameter tuning. The SVR (RBF) and the tree models (DT and RF) also improved after tuning their hyperparameters. Both tree models do not have overfitted models, as presented in the previous table. However, these results cannot be improved if hyperparameter tuning is not implemented. Moreover, tuning hyperparameters will be challenging if many road segments are being predicted.

To study the behavior of the prediction model, the GB model, as assumed as the best model for this scenario, was selected. The following Figure 30, Figure 31, and Figure 32 show the prediction behavior of the GB model both on Tsimiski and Egnatia street.

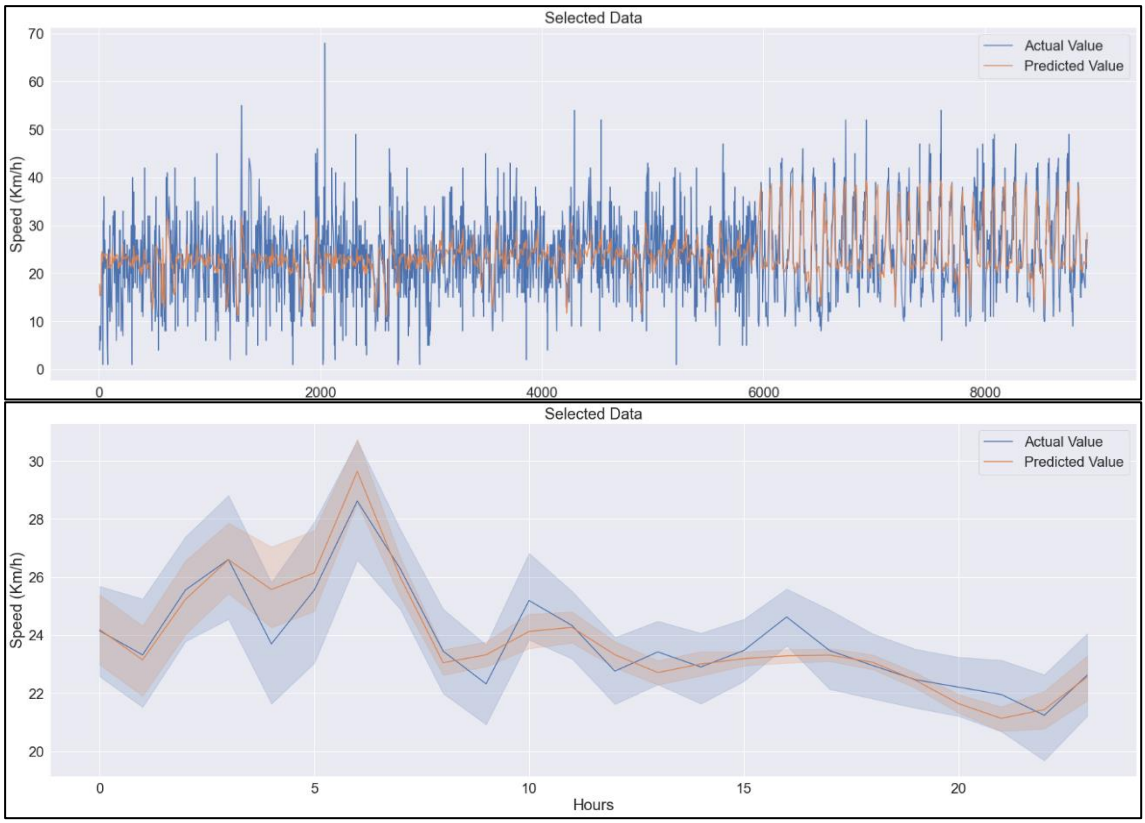

Figure 30: Daily comparison (top) and hourly comparison (bottom) between the actual and predicted value on Tsimiski street for the second scenario

From Figure 30 above, the model has trouble predicting the value as it 'stretched' by the segments with the inconsistent trend. This effect can be seen in the hourly trend figure, where the speed prediction from hour four until seven was trying to balance the input trend, as depicted in Figure 29. Moreover, according to the daily trend figure, the model can predict the result very well after step 6000 (day 21st), which may be because the problem segments have had decent data quality since then.

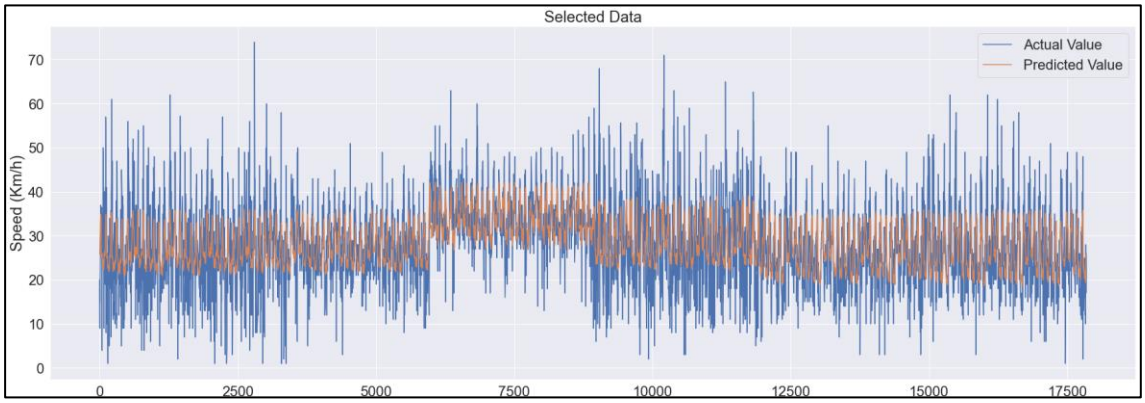

Figure 31: Daily comparison between the actual and predicted value on Egnatia street for the second scenario

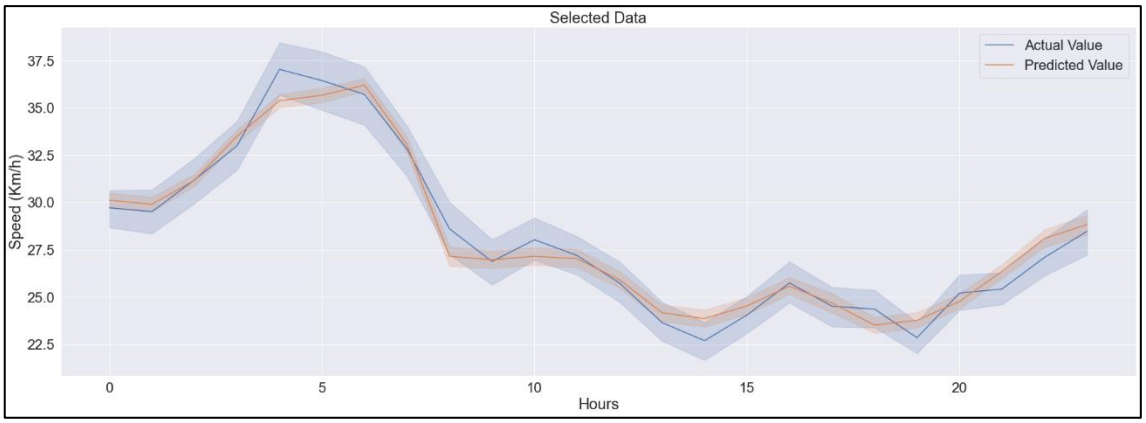

Figure 32: Hourly comparison between the actual and predicted value on Egnatia street for the second scenario

From both Figure 31 and Figure 32, the GB model does not have any trouble predicting the speed value as it shows that the prediction result is consistent, whether daily or hourly. However, even though it is consistent, the data input quality for Egnatia street is unreliable, which results in a low score value. Moreover, a prediction using the GB model with and without additional features was performed to see how the road length and bus stop features affect the prediction result. Table 18 shows the comparison results of this experiment—C1 shows where both features are included, while C2 shows where those features are excluded. This result shows that adding these features does not significantly affect the model performance.

Table 18 Prediction comparison for road length and bus stop features using the GB model

	Tsimiski St.		**Egnatia St.**	
Metric	**C1**	**C2**	**C1**	**C2**
R^2 Score	0.34	0.33	0.32	0.32
MAE	5.13	5.15	5.87	5.88
RMSE	6.83	6.84	7.93	7.93

4.2.3 Multiple Month Data – One Road Segments

The idea of this scenario is to study the behavior of the analysis when one road segment is analyzed with multiple months included. With multiple months included, more features will be included; in this case, the month column would be the additional feature. Also, adding multiple months in one or even from different years would add more traffic patterns to be learned by each model. Figure 33 shows the correlation between each feature and the target feature for this scenario.

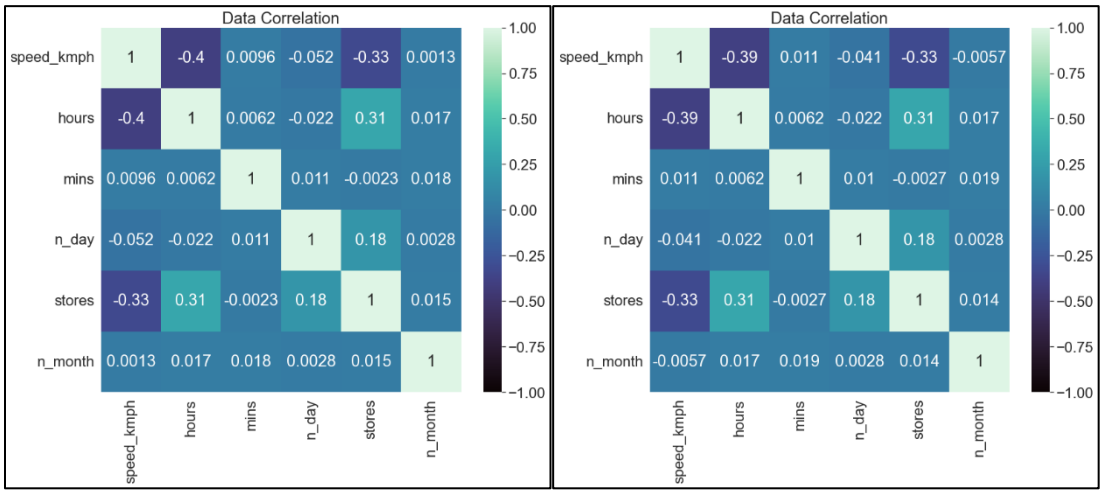

Figure 33: Tsimiski's data (left) and Egnatia's data (right) feature correlation for the third scenario

From Figure 33, both 'hours' and 'stores' correlate relatively similar to the speed feature experienced in the first scenario. At the same time, the rest features were still added, so the models have some more features to learn to create a better model. Moreover, after selecting the target and input features, the dataset was split into training and test datasets shown in Figure 34.

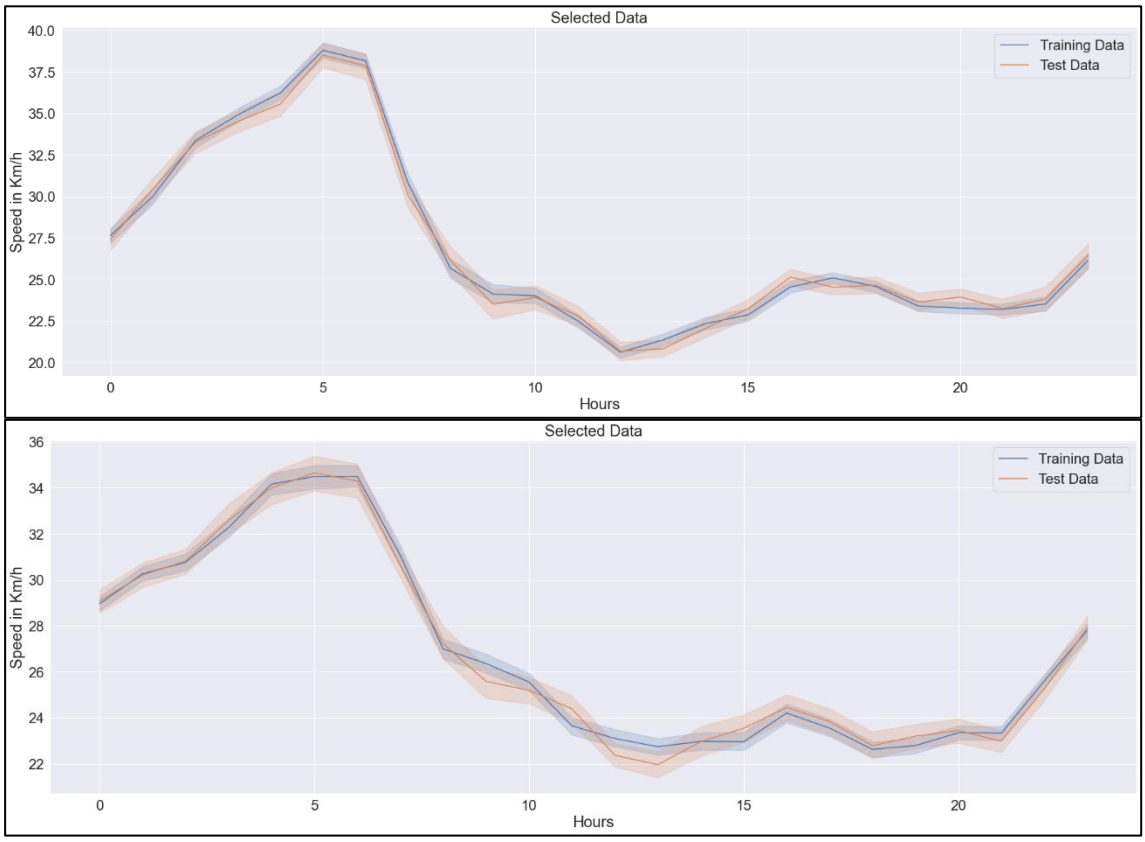

Figure 34: Hourly trend comparison between the selected training and test data on Tsimiski (Top) and Egnatia (bottom) street

The same trend with the first scenario is shown in both figures above. The traffic will reach its peak speed in the morning before it starts congesting. Then again, it will reach the maximum speed in the afternoon before it starts congesting in the evening. After this process, training and test datasets were fed to the model, and Table 19 shows the prediction result.

Table 19 Prediction performance on the selected data on both streets in the year 2018 on the third scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.28	3.97	4.32	4.66	4.66	4.38	4.13	4.09
	Egnatia Street	5.29	4.51	4.42	4.48	4.48	5.04	4.77	4.24
RMSE	Tsimiski Street	7.83	5.43	5.61	7.87	6.16	6.08	5.71	5.35
	Egnatia Street	6.79	6.09	5.84	6.79	5.96	6.82	6.42	5.67
R^2 Score	Tsimiski Street (Test Data)	0.20	0.61	0.59	0.19	0.50	0.52	0.57	0.62
	Tsimiski Street (Training Data)	0.22	0.74	0.60	0.21	0.52	0.79	0.79	0.63
	Egnatia Street (Test Data)	0.20	0.35	0.40	0.20	0.38	0.19	0.28	0.44
	Egnatia Street (Training Data)	0.20	0.56	0.41	0.20	0.38	0.64	0.64	0.44

Compared to the first scenario on Egnatia street, the result for this scenario is relatively better. In this scenario, the R^2 score on this street reaches up to 40%, whereas in the first scenario, it can only reach 25%. However, there is no change in the result on Tsimiski street, as it has a constant R^2 score of around 60% in both scenarios. The same goes for the MAE and RMSE value, where the error difference in both scenarios reach up to 0.4 km/h, with this scenario performing better than the first scenario. The result of this scenario might be affected by the additional data added from other months, which gives Egnatia data that was assumed to have insufficient quality data in January some more patterns to learn and improve the performance.

Result Optimization on Third Scenario

In this scenario, from Table 19 alone, we can already conclude that the GB model is still performing relatively better than any other model even when hyperparameter tuning was not applied to the model yet. Again with the tuned hyperparameter shown in Appendix C, the performance of each model can be improved, and the result of this optimization can be seen in Table 20.

Table 20 Optimized prediction performance on the third scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.28	3.99	3.94	6.27	4.22	4.22	4.41	4.08
	Egnatia Street	5.29	4.53	4.26	5.30	4.56	4.56	4.49	4.24
RMSE	Tsimiski Street	7.83	5.45	5.24	7.83	5.30	5.61	5.71	5.35
	Egnatia Street	6.79	6.11	5.71	6.78	5.73	6.00	5.91	5.67
R^2 Score	Tsimiski Street (Test Data)	0.20	0.61	0.64	0.19	0.63	0.59	0.57	0.63
	Tsimiski Street (Training Data)	0.22	0.74	0.66	0.21	0.65	0.60	0.59	0.63
	Egnatia Street (Test Data)	0.20	0.35	0.43	0.20	0.43	0.37	0.39	0.44
	Egnatia Street (Training Data)	0.20	0.56	0.45	0.20	0.45	0.38	0.40	0.45

From the table above, both GB and NN models perform better than the other models. Additionally, from the MAE result, KNN performs on par with those two models, but the R^2 value for the KNN model should also be considered. It overfitted the training data even though the model hyperparameters were already tweaked to get the best results. Therefore, the acceptable R^2 value should be defined if this model is selected for deployment. Aside from this, the GB model should be selected to analyze the result further, considering it is also the best model in this scenario. Respectively, Figure 35 and Figure 36 below show the average hourly comparison between the actual and the predicted value on Tsimiski street and Egnatia street.

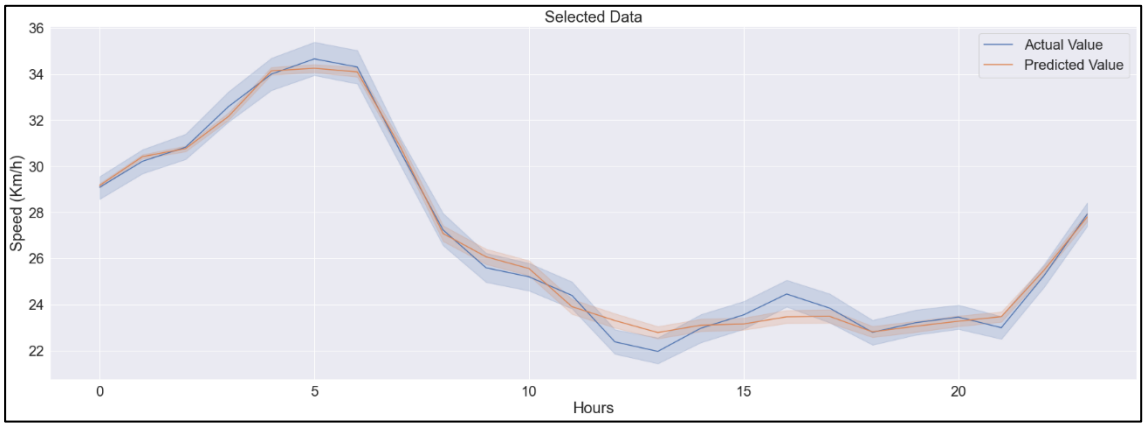

Figure 35: Hourly comparison between the average actual and predicted value on Tsimiski street for the third scenario

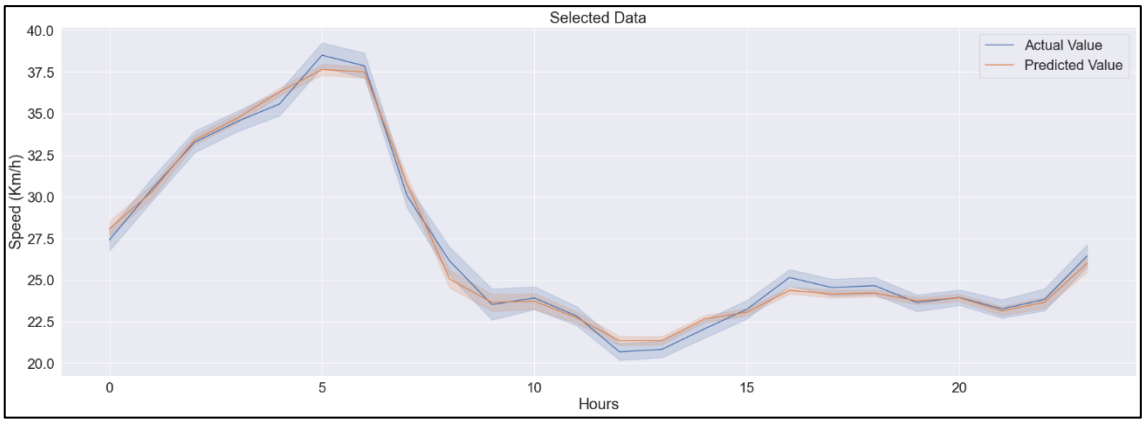

Figure 36: Hourly comparison between the average actual and predicted value on Egnatia street for the third scenario

Moreover, additional processing was conducted to test how the multiple-year data affected the prediction result, shown in Table 21. Only GB results are compared for these experiments because it represents the best model for this scenario.

Table 21 Multiple-year comparison results using the GB model

	Tsimiski St.				**Egnatia St.**			
Metric	**1 Y**	**2 Y**	**3 Y**	**4 Y**	**1 Y**	**2 Y**	**3 Y**	**4 Y**
R^2 Score	0.62	0.60	0.54	0.56	0.44	0.40	0.36	0.37
MAE	4.09	4.37	5.01	5.24	4.24	4.55	5.12	5.16
RMSE	5.36	5.73	6.68	7.01	5.67	6.09	6.90	6.97

Additionally, Table 22 compares the prediction results between years to see how consistent the prediction is for each year.

Table 22 Each year's comparison results using the GB model

	Tsimiski St.				**Egnatia St.**			
Metric	**2018**	**2019**	**2020**	**2021**	**2018**	**2019**	**2020**	**2021**
R^2 Score	0.62	0.60	0.52	0.61	0.44	0.38	0.29	0.40
MAE	4.09	4.47	5.48	5.22	4.24	4.80	5.93	4.96
RMSE	5.36	5.95	7.53	7.14	5.67	6.39	8.03	6.80

From both tables above, we can conclude that adding more years might increase or decrease the model's performance. In this case, both selected streets are experiencing a change in the year 2020. This change is most likely due to COVID-19 that happened throughout that year. Therefore, while predicting the speed, the overall score will be reduced by the score in 2020.

4.2.4 Multiple Month Data – Multiple Road Segments

This scenario aims to test the model's capabilities if the second scenario has an acceptable result. Therefore, adding more patterns from other months might improve the prediction result. Figure 37 below shows the correlation between the speed value, which acts as the target value, and the input features in this study.

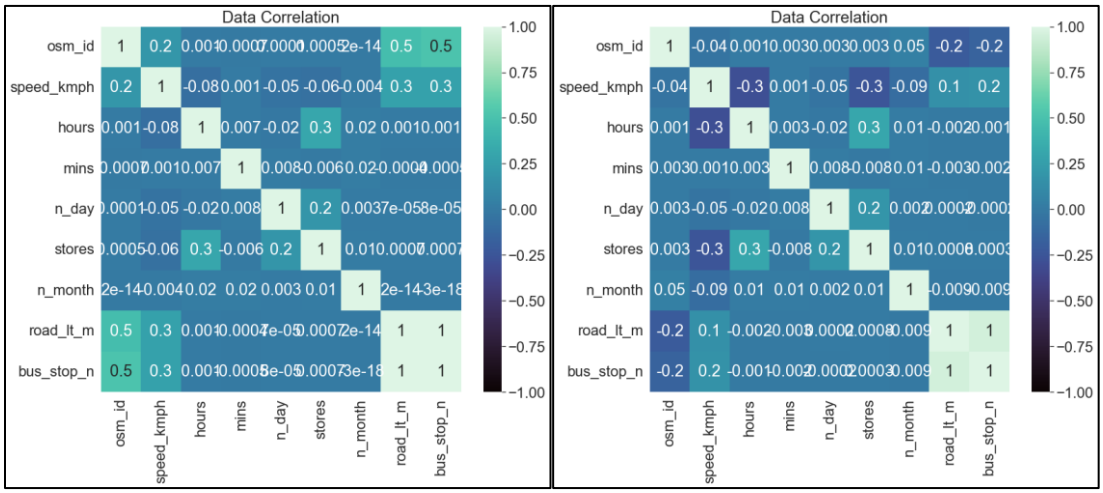

Figure 37: Tsimiski's data (left) and Egnatia's data (right) feature correlation for the fourth scenario

Similar to the second scenario in 4.2.2, the additional input features, i.e., the bus stop and road length, positively correlate to the target feature. At the same time, the other input features other than those two features act similarly as in other scenarios. After checking the correlation between target and input features, the training and test set were selected, as seen in the following Figure 38 and Figure 39.

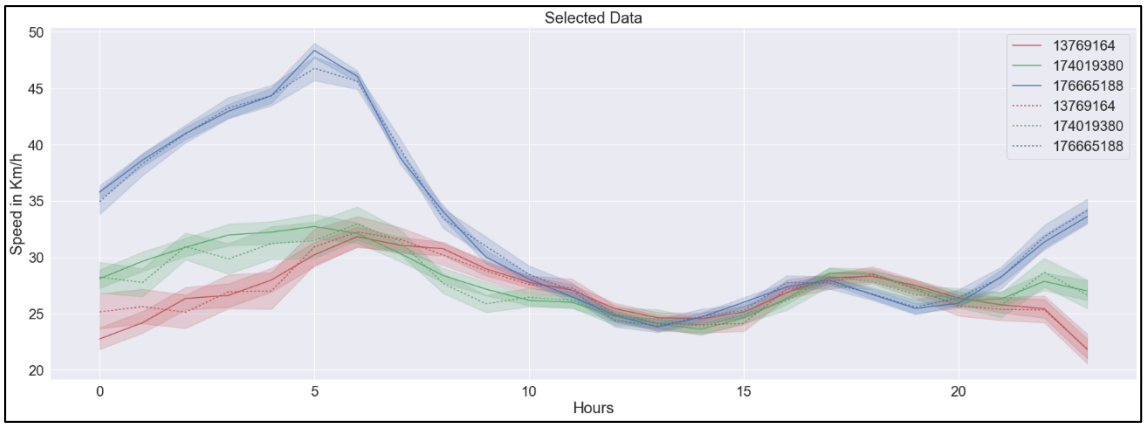

Figure 38: Hourly trend of training and test data on Tsimiski Street for each road segment

Figure 38 above shows a different pattern between each segment, which indicates the actual behavior in that road, or both segments 174019380 and 13769164 have bad

quality data recording. In contrast, Figure 39 shows that the Egnatia dataset segments have similar hourly average speed patterns.

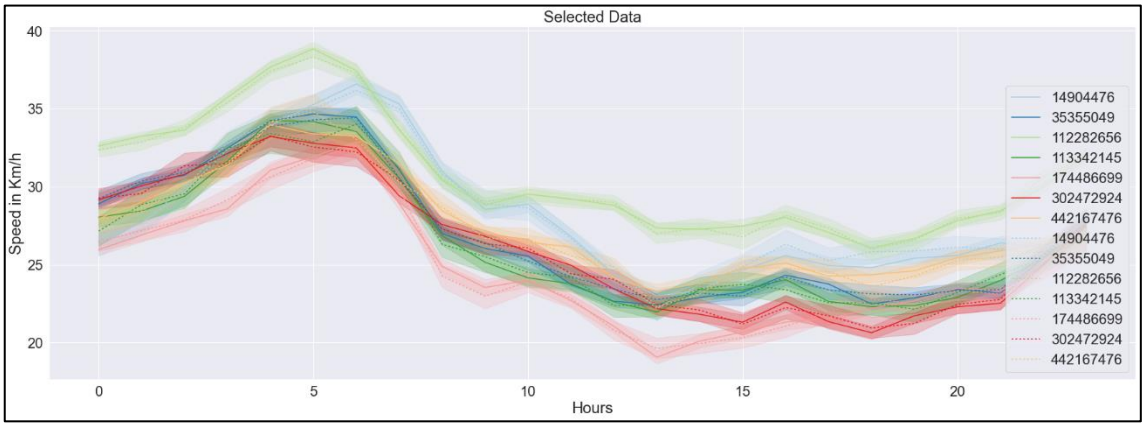

Figure 39: Hourly trend of training and test data on Egnatia Street for each road segment

After the training and test data had been selected, the data were then fed to the ML models. For this scenario, the total records for the Tsimiski and Egnatia input data are 105,117 and 241,764. These numbers differ because the total selected segments are different between these roads. Therefore, it makes the training and test process for Egnatia data more challenging when using the SVR (RBF) model, as seen in Table 23.

Table 23 Prediction performance on the selected data on both streets in the year 2018 on the third scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.53	5.15	5.11	6.51	5.35	5.66	5.36	5.27
	Egnatia Street	6.34	5.72	5.52	6.33	-	6.38	5.99	5.58
RMSE	Tsimiski Street	8,32	7.08	6.84	8.33	7.16	7.82	7.39	6.95
	Egnatia Street	8.17	7.74	7.34	8.18	-	8.70	8.12	7.38
R^2 Score	Tsimiski Street (Test Data)	0.08	0.34	0.38	0.08	0.32	0.19	0.28	0.36
	Tsimiski Street (Training Data)	0.09	0.55	0.39	0.09	0.32	0.64	0.63	0.36
	Egnatia Street (Test Data)	0.15	0.24	0.31	0.15	-	0.04	0.16	0.31
	Egnatia Street (Training Data)	0.14	0.49	0.32	0.15	-	0.60	0.59	0.31

As discussed in section 2.2.3, the SVR may require more memory and computation power to process a dataset with more than 100,000 records. Therefore, with the total record that Egnatia street has, SVR using the RBF kernel cannot be accomplished since

it needs much time to compute. Aside from this matter, for this scenario, both the NN and GB is still the best model. These models can achieve the optimal score without the need for hyperparameter tweaking.

Result Optimization on Fourth Scenario

In order to compare the model at the best performance, hyperparameter tuning was done to all models until each model reached its maximum result. Table 24 shows the optimized performance of each model except the SVR (RBF) model. From the table, even after hyperparameter tweaking, the NN and GB are still the best models among all ML models used in this study.

Table 24 Optimized prediction performance on the third scenario

	Scenario	LR	KNN	NN	SVR (Linear)	SVR (RBF)	DT	RF	GB
MAE	Tsimiski Street	6.53	5.15	5.08	6.52	5.09	5.29	5.37	5.27
	Egnatia Street	6.34	5.72	5.40	6.34	-	5.87	5.79	5.58
RMSE	Tsimiski Street	8.32	7.07	6.80	8.32	6.80	7.03	7.06	6.95
	Egnatia Street	8.17	7.75	7.24	8.17	-	7.67	7.58	7.38
R^2 Score	Tsimiski Street (Test Data)	0.08	0.34	0.39	0.08	0.39	0.35	0.34	0.36
	Tsimiski Street (Training Data)	0.09	0.55	0.40	0.09	0.40	0.35	0.34	0.36
	Egnatia Street (Test Data)	0.15	0.24	0.33	0.15	-	0.25	0.27	0.31
	Egnatia Street (Training Data)	0.15	0.49	0.35	0.15	-	0.26	0.27	0.31

Moreover, additional processing was conducted to test how the multiple-year data affected the prediction result. This test result can be seen in the following Table 25. Similar to the experiment in Table 21 and Table 22, only GB results are compared for these experiments because it represents the best model for this scenario.

Table 25 Multiple-year comparison results using the GB model

	Tsimiski St.				Egnatia St.			
Metric	**1 Y**	**2 Y**	**3 Y**	**4 Y**	**1 Y**	**2 Y**	**3 Y**	**4 Y**
R^2 Score	0.36	0.35	0.36	0.36	0.31	0.31	0.30	0.31
MAE	5.27	5.52	6.30	6.66	5.58	5.93	6.51	6.62
RMSE	6.95	7.28	8.39	8.84	7.38	7.79	8.59	8.73

Table 26 compares the prediction results between years to see the prediction consistency. When the result between Table 22 and the following table, the result between years seems to be consistent. It may indicate that the previous result was not affected by the fact that COVID-19 happened in 2020, or probably as for this scenario, multiple segments are analyzed, which makes the effect of an event such as COVID-19 not distinctly appears in the result.

Table 26 Each year's comparison results using the GB model

	Tsimiski St.				**Egnatia St.**			
Metric	**2018**	**2019**	**2020**	**2021**	**2018**	**2019**	**2020**	**2021**
R^2 Score	0.36	0.36	0.37	0.46	0.31	0.30	0.30	0.38
MAE	5.27	5.66	7.30	6.51	5.58	6.15	7.30	6.60
RMSE	6.95	7.44	9.74	8.75	7.38	8.08	9.61	8.78

5 Discussion

This section will discuss the results from the previous section, followed by the viability of performing network-wide prediction. The results from section 4.1 will be compared with those from the previous study as the benchmark. After discussing the prediction performance, the results from section 4.2 will be used for city-wide network prediction.

5.1 Prediction Performance

The discussion for the prediction outcomes will use [9] results as a benchmark. The same dataset was used in their study: Thessaloniki's FCD data from the same source. It was five months of traffic data in 2019, containing features such as minimum, maximum, standard deviation, mean skewness, and kurtosis of speed at the 15 minutes time window, the date and time, the road entries, and the unique entries. These features were used as the training input to several algorithms: RF, SVR, NN, and LR models. Therefore, the models were trained using the data from the previous two weeks to the desired date-time and assessed using 10-fold cross-validation techniques [9].

Several scenarios were applied to the experiments. The scenarios are the speed prediction at random dates and times, the speed prediction at eight consecutive dates and times, and for a specific 24-hour time window; all these scenarios were applied on randomly selected roads. Therefore, these were expected to analyze the algorithms' forecasting abilities [9]. In section 4.1, these scenarios were recreated to test model performance in this study when different input features were used. The result and the comparison between the current and previous studies can be seen in the following Figure 40 and Figure 41.

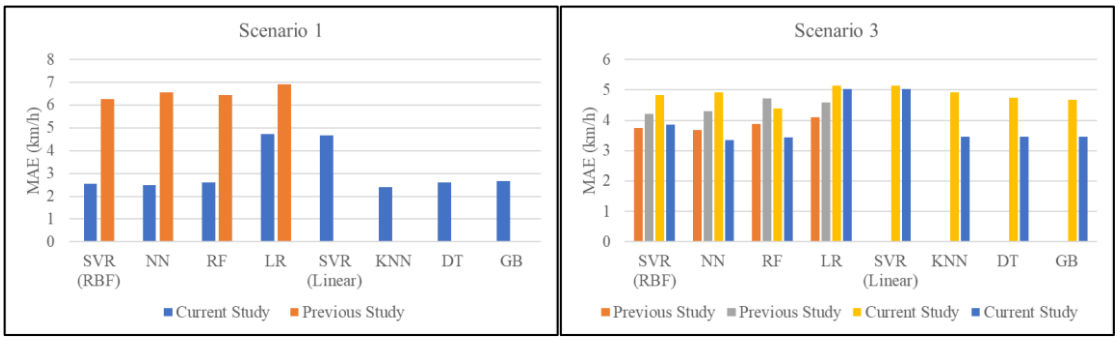

Figure 40: MAE comparison between the current and previous studies on the first and third scenarios

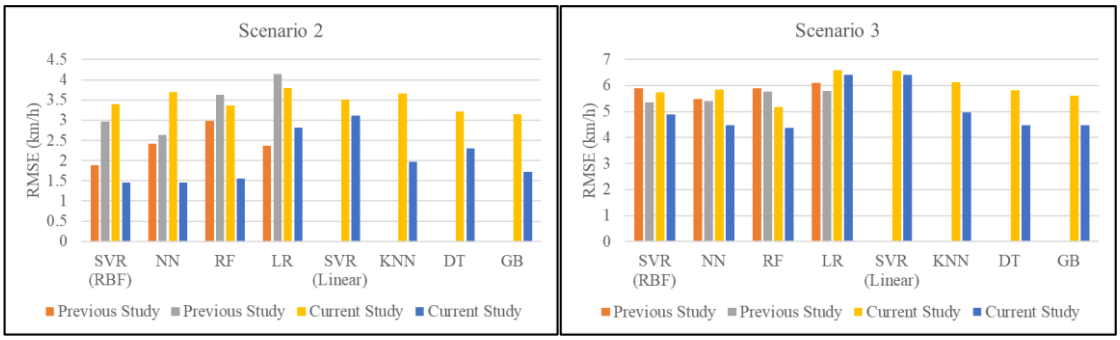

Figure 41: RMSE comparison between the current and previous studies on the second and third scenarios

From all the figures above, it also can be concluded that the linear models (LR and SVR linear) have the worst performance among the other ML models. Therefore, for further discussion, *these ML models will be excluded*. It is aligned with what was already mentioned in section 2.4; the linear model is insufficient to predict the traffic speed because it is a complex phenomenon that might get disturbed by typical and atypical conditions such as accidents and extreme weather. However, the result from section 4.1.2 might imply that the linear model can be used when the prediction is for more than one time step but no more than eight time steps, as the prediction performance is poor when the 24 hours scenario was applied for these linear models.

Overall, from Figure 40, the results have outperformed the preceding study on the first scenario. Additionally, this study found more ML models that can perform on par with SVR, NN, and RF models. Furthermore, throughout all the figures, the ML models that perform very well in this study are the NN, GB, RF, DT, SVR (RBF), and KNN. For the third scenario, the MAE is quite varied, where a sample in this study can perform very poorly, but there is also a sample that performs exceptionally well compared to all the tests. Therefore, the RMSE from Figure 41 should also be considered to analyze the results for the third scenario. The results from the current study have a relatively small difference between MAE and RMSE values, which means that the individual error from this study has a smaller variance than the previous study. Moreover, in Figure 41, the second scenario also has a similar problem where the RMSE results are varied. Nevertheless, as the previous study did not calculate the MAE for the second scenario, the results for the second scenario can not be deeply analyzed.

In the first scenario, as mentioned in 4.1.1, the KNN model top the performance, followed by NN, SVR, trees, and GB models with a fraction less than 0.5 km/h difference. It also has the lowest maximum error, which is capped at 5 km/h, while on the

other hand, the linear models have a maximum RMSE of around 7 km/h. However, the R^2 score for the KNN model should be considered because even though hyperparameter tuning is already applied to the model, the resulting model still overfits the training data, making the result from this model seem unreliable. Looking at Figure 15, the NN model performs very well when predicting the speed at random times, as the median is closer to the lower quartile of RMSE results. Nevertheless, if the lower maximum error is considered, the SVR (RBF), RF, and GB models can be selected as the best model for this scenario. Aside from the maximum error from these models, the distributions of RMSE for each model are gathered around the lowest error, and the box size shows that the RF model precisely predicts the speed, followed by GB and SVR (RBF) model.

When predicting multiple times, such as in the second and third scenarios, the overall MAE and RMSE will become more significant. Because, during the period of data being predicted, the prediction error will be more varied. This study did not perform the prediction in a loop to predict for each timestep. Instead, it will simultaneously predict the whole data input for prediction. It was done to avoid performance issues when predicting the speed value for a more extended period, as mentioned in section 4.1.3. Moreover, the next section will discuss which model is more suitable and efficient for a more extended period of city-wide prediction.

5.2 Network-Wide Prediction

The network-wide traffic speed prediction is needed for route planning and traffic intervention in advance to improve traffic efficiency. Therefore, the development of ITS will be supported by being able to predict the traffic speed at the macro-level accurately and efficiently. For this study, the network-wide prediction is still in the phase of concept, which will be discussed using the results from section 4.2. Traditional ML algorithms are used in this study because the preparation of input data for more sophisticated models is quite complex and time-consuming. Thus, using the traditional model is expected to give a lot more simple approaches and less time-consuming process, but could give a good result with acceptable performance.

According to section 4.2, several ML models can be picked when the most efficient scenario is selected for deployment. Overall, the models that perform very well are similar to the results from the previous section: the NN, GB, trees, and SVR (RBF). For most scenarios, linear models show the sign that traffic speed prediction is a complex

problem that cannot be approached using a linear model. While on the other hand, the KNN model experience overfitting, which was challenging to tune the hyperparameters. Therefore, only those five models are still in consideration. However, from this study, after comparing these models in four different scenarios, it is found that the GB might be the most suitable model to be picked in any scenario that will be selected. This ML reaches optimum results from all the scenarios without hyperparameter tuning. In comparison, the rest of the models need to be tweaked *to reach or slightly surpass* the result of the GB model.

Moreover, when comparing NN and GB models, in terms of training and prediction time, GB models could work much faster than NN, especially when the input data have many records. The statement in section 2.2.7 supports it; NN (for this study, MLP algorithm is used) takes a long time to train and requires careful data preprocessing. It also needs hyperparameter tuning, such as its hidden neurons, the layers, and the number of iterations. On the other hand, according to section 2.2.6, the GB model works by creating shallow trees that lead to a faster processing time.

In terms of the scenario, only the scenarios that predict one road segment at a time are recommended. Even though simultaneously predicting multiple road segments might speed up the process, the overall prediction performance might suffer from the road segment with low-quality data or a very different pattern. Moreover, analyzing multiple road segments one at a time might require much more computational power and RAM. Additionally, adding spatial features such as bus stop and road length to the input training data do not help improve the result, as seen in section 4.2.2, where there is no difference between the result when these features are included and excluded in the training process. It is also supported by the fact that through the correlation matrix, even though these spatial features correlate to the traffic speed, the correlation is not high enough to affect the prediction result.

If the spatial features want to be added when predicting one road segment, it is suggested to try putting the adjacent road segment and its data as input features. Therefore, the effect of adjacent segments on the predicted road segment can be incorporated during the model creation. It is because the traffic on one road segment might propagate to other segments—this traffic also might be affected by the features that occur on the other road segments. However, if the spatial features are not considered, the prediction

process for a network-wide case can be done through looping for each road segment at a time.

Predicting in a looping fashion for one road segment at a time should be used to obtain network-wide prediction results. Nevertheless, the question is, how much input data are needed to predict efficiently. According to the results from the third scenario, adding more data might increase or decrease the R^2 score. For the case of Egnatia data, adding more data to one year improve the overall result compared to the first scenario. In contrast, in the Tsimiski case, adding more data will slightly decrease the overall results. One thing that has not yet been tried is how far the model can predict the traffic speed from the period of the training data.

Threats to the Validity

Overall, the method in this study can be applied to another traffic prediction case, as long as the data related to the traffic speed is available. The method is flexible with other datasets because it uses date-time features to derive some information such as hours, minutes, days, months, and even the store status. However, when applying the method, the appropriate store schedule should be adjusted to the usual working hours in the city where this method will be applied.

Several factors must be considered as they might introduce problems to the results. For example, in this study, FCD comes from some taxi fleets passing a road segment, and each car's speed will be averaged to obtain the traffic speed in that particular segment. However, many road segments get their value from only one vehicle. Thus, the certainty of the speed value cannot be guaranteed. Also, as the speed value will be mapped to each appropriate road segment using an algorithm, there might be an error during that process. Therefore, these pieces of information regarding data quality are necessary. However, as this study did not cover the processing step of GPS data into FCD, a more in-depth study is needed to clarify this.

Another problem that could arise when predicting the traffic speed on the network-wide scale is the availability of the data itself. As mentioned in 3.3.3 (see both Figure 11 and Figure 12), there was a problem regarding the availability of FCD data. Therefore, it is further inspected to know the distribution of unavailable FCD data throughout OSM data. The following Figure 42 shows the result of the inspection.

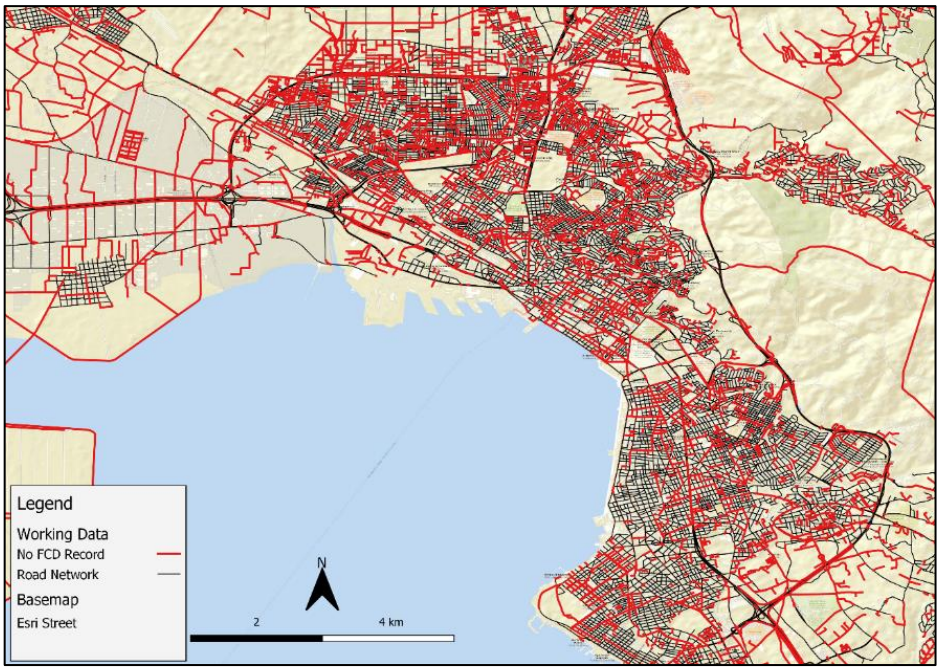

Figure 42: Road network overlayed with unavailable FCD data

From the figure above, the red lines indicate the road segment that does not have traffic speed data from the FCD data. Moreover, the number of a road segment that does not have the traffic speed data can be seen in Table 27 below.

Table 27 The number of missing FCD data on each road type

Road Type	Count
corridor	10
living_street	384
motorway	20
motorway_link	28
primary	200

Road Type	Count
primary_link	55
proposed	134
residential	3231
secondary	795
secondary_link	113

Road Type	Count
tertiary	758
tertiary_link	111
trunk	98
trunk_link	73
unclassified	661

From the table above, the road segment that does not have traffic speed data are dominated by the residential type road, which is less likely to be populated by taxi car. However, the residential road can be excluded from the analysis, as route planning should avoid this road type. Nevertheless, the number of unavailable data on the primary and the other essential road types is pretty high and cannot be ignored. Therefore there should be a study to predict the traffic speed in the segments with no data. Furthermore, analyzing network-wide traffic speed should consider the influence of other features and adjacent segments because the traffic jam in one segment may propagate to other segments. Therefore, the study that uses this data for network-wide prediction will become more challenging because of the gap between segments within the data.

6 Conclusion and Future Work

The approach from [3] can give comparable results to the result in [9]. Overall, the results in this study perform better than the prior study. However, the comparison between the current and previous studies was not on the same date and times or the same road. It is because the initial study did not give this information; thus, this study tried to imitate the scenario as closely as possible to get the result that could be compared. This study also processes data simultaneously, while the previous study's results seem to predict the speed per time steps. According to the community discussion, processing the data in that manner may be time-consuming and need a lot of computational resources.

In terms of the best model from this study, the NN, GB, SVR (RBF), and tree models are strong contenders among the tested ML algorithms. However, the SVR might suffer computational power issues when dealing with a dataset with more than 100,000 records. Meanwhile, tree models (RF and DT) are pretty good models. However, these models need hyperparameter tuning to adjust the trees and make it less complex to overcome overfitting. Tuning hyperparameters might be a problem when predicting speed on a network-wide scale, which requires some automation to find the best parameters for each road prediction. Therefore, the NN and GB models are the best among those models. Nevertheless, this study will pick the GB model as the best as it works faster than NN. As presented in the respective section, it also does not need hyperparameter tuning, where this model already reaches optimal results without tweaking.

Several scenarios were tested to determine the most efficient way to predict the traffic speed on a network-wide scale. From these analyses, deploying the scenario where multiple road segments are selected is not recommended as it may reduce the overall accuracy of the prediction and drag the prediction accuracy in other road segments. In this scenario, the effect of adding spatial features, which are the bus stop and road length, were studied. It was found that adding these features was not significantly affect the prediction result. Therefore, the most efficient way to predict traffic speed using this data is to predict the speed per segment. Adding more months to the input data is recommended as it may increase the prediction score. However, there is still a need to

study the sustainability of the prediction model, as it is still unknown how far the created model can predict the traffic speed accurately.

Future Works

For future work, it is recommended to perform network-wide prediction using the recommendation from this study and see how fast and accurate this approach is compared to the traffic prediction using DL algorithms as performed in [8] or other similar studies. To extend the usability of network-wide prediction, the effect of route planning using historical and predicted data can be compared to see whether it is feasible to use traffic speed prediction results for route planning.

Additionally, some experiments mentioned in the discussion may become the future works from this study. One example is performing one road segment prediction while adding adjacent roads as input features to see whether the prediction performance might improve or not. Alternatively, using small time steps on different days for years as the input features to predict that particular time step might be interesting to be tested. The other possibility of future work might be performing prediction on the road segment with no speed data based on adjacent road segment might be useful to improve the capability of open data that does not always have good quality. Moreover, adding weather in the input features may also be the future work from this study. Furthermore, analyzing whether the method in this study is sensitive to typical or atypical conditions can also be the other expansion of this study.

Bibliography

[1] A. Bull, United Nations, and Deutsche Gesellschaft für Technische Zusammenarbeit, Eds., *Traffic congestion: the problem and how to deal with it*. Santiago, Chile: United Nations, Economic Commission for Latin America and the Caribbean, 2003.

[2] G. P. Rocha Filho *et al.*, 'Enhancing intelligence in traffic management systems to aid in vehicle traffic congestion problems in smart cities', *Ad Hoc Networks*, vol. 107, p. 102265, Oct. 2020, doi: 10.1016/j.adhoc.2020.102265.

[3] A. Mystakidis and C. Tjortjis, 'Big Data Mining for Smart Cities: Predicting Traffic Congestion using Classification', in *2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA*, Piraeus, Greece, Jul. 2020, pp. 1–8. doi: 10.1109/IISA50023.2020.9284399.

[4] N. A. M. Razali, N. Shamsaimon, K. K. Ishak, S. Ramli, M. F. M. Amran, and S. Sukardi, 'Gap, techniques and evaluation: traffic flow prediction using machine learning and deep learning', *Journal of Big Data*, vol. 8, no. 1, p. 152, Dec. 2021, doi: 10.1186/s40537-021-00542-7.

[5] 'Intelligent transportation system', *Wikipedia*. Mar. 09, 2022. Accessed: Mar. 15, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Intelligent_transportation_system&oldid=1076143573

[6] M. Choudhary, 'What is Intelligent Transport System and how it works?', *Geospatial World*, Jan. 14, 2019. https://www.geospatialworld.net/blogs/what-is-intelligent-transport-system-and-how-it-works/ (accessed Mar. 15, 2022).

[7] H. Yuan and G. Li, 'A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation', *Data Sci. Eng.*, vol. 6, no. 1, pp. 63–85, Mar. 2021, doi: 10.1007/s41019-020-00151-z.

[8] Z. Cui, R. Ke, Z. Pu, and Y. Wang, 'Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction', arXiv, arXiv:1801.02143, Nov. 2019. doi: 10.48550/arXiv.1801.02143.

[9] C. Bratsas, K. Koupidis, J.-M. Salanova, K. Giannakopoulos, A. Kaloudis, and G. Aifadopoulou, 'A Comparison of Machine Learning Methods for the Prediction of Traffic Speed in Urban Places', *Sustainability*, vol. 12, no. 1, Art. no. 1, Jan. 2020, doi: 10.3390/su12010142.

[10] K. Zhang, L. Zheng, Z. Liu, and N. Jia, 'A deep learning based multitask model for network-wide traffic speed prediction', *Neurocomputing*, vol. 396, pp. 438–450, Jul. 2020, doi: 10.1016/j.neucom.2018.10.097.

[11] S. Zhang, L. Zhou, X. (Michael) Chen, L. Zhang, L. Li, and M. Li, 'Network-wide traffic speed forecasting: 3D convolutional neural network with ensemble empirical mode decomposition', *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 10, pp. 1132–1147, 2020, doi: 10.1111/mice.12575.

[12] K. Christantonis, C. Tjortjis, A. Manos, D. E. Filippidou, E. Mougiakou, and E. Christelis, 'Using Classification for Traffic Prediction in Smart Cities', in *Artificial Intelligence Applications and Innovations*, vol. 583, I. Maglogiannis, L. Iliadis, and E. Pimenidis, Eds. Cham: Springer International Publishing, 2020, pp. 52–61. doi: 10.1007/978-3-030-49161-1_5.

[13] S. Liapis, K. Christantonis, V. Chazan-Pantzalis, A. Manos, D. Elizabeth Filippidou, and C. Tjortjis, ‘A methodology using classification for traffic prediction: Featuring the impact of COVID-19’, *Integrated Computer-Aided Engineering*, vol. 28, no. 4, pp. 417–435, Jan. 2021, doi: 10.3233/ICA-210663.
[14] J. Mahona, C. Mhilu, J. Kihedu, and H. Bwire, ‘FACTORS CONTRIBUTING TO TRAFFIC FLOW CONGESTION IN HETEROGENOUS TRAFFIC CONDITIONS’, *IJTTE*, vol. 9, no. 2, pp. 238–254, 2019, doi: 10.7708/ijtte.2019.9(2).09.
[15] T.-I. Theodorou, A. Salamanis, D. D. Kehagias, D. Tzovaras, and C. Tjortjis, ‘Short-Term Traffic Prediction under Both Typical and Atypical Traffic Conditions using a Pattern Transition Model’:, in *Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems*, Porto, Portugal, 2017, pp. 79–89. doi: 10.5220/0006293400790089.
[16] ‘Smart Cities Council | How cities can fix traffic congestion (and why one solution is rarely enough)’. https://rg.smartcitiescouncil.com/readiness-guide/article/congestion-reduction-how-cities-can-fix-traffic-congestion-and-why-one-solution-rarely-enough (accessed May 16, 2022).
[17] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python*, First Edition. United States of America: O’Reilly Media, Inc., 2017.
[18] ‘1.1. Linear Models’, *scikit-learn*. https://scikit-learn/stable/modules/linear_model.html (accessed May 22, 2022).
[19] ‘1.6. Nearest Neighbors’, *scikit-learn*. https://scikit-learn/stable/modules/neighbors.html (accessed May 22, 2022).
[20] ‘1.4. Support Vector Machines’, *scikit-learn*. https://scikit-learn/stable/modules/svm.html (accessed May 24, 2022).
[21] ‘1.10. Decision Trees’, *scikit-learn*. https://scikit-learn/stable/modules/tree.html (accessed May 22, 2022).
[22] D. Varghese, ‘Comparative study on Classic Machine learning Algorithms’, *Medium*, May 10, 2019. https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222 (accessed May 23, 2022).
[23] ‘1.11. Ensemble methods’, *scikit-learn*. https://scikit-learn/stable/modules/ensemble.html (accessed May 22, 2022).
[24] ‘1.17. Neural network models (supervised)’, *scikit-learn*. https://scikit-learn/stable/modules/neural_networks_supervised.html (accessed May 24, 2022).
[25] Z. Zhou, Z. Yang, Y. Zhang, Y. Huang, H. Chen, and Z. Yu, ‘A comprehensive study of speed prediction in transportation system: From vehicle to traffic’, *iScience*, vol. 25, no. 3, p. 103909, Feb. 2022, doi: 10.1016/j.isci.2022.103909.
[26] PyData, *TensorTraffic - traffic prediction using machine learning - Pawel Gora*, (Nov. 13, 2017). Accessed: Apr. 28, 2022. [Online Video]. Available: https://www.youtube.com/watch?v=XqxwnbKHEEs
[27] F. Pedregosa *et al.*, ‘Scikit-learn: Machine Learning in Python’, *MACHINE LEARNING IN PYTHON*, p. 6.
[28] ‘network-speed-historical - CERTH-HIT OpenData Hub Greece’. https://opendata.imet.gr/dataset/network-speed-historical (accessed May 04, 2022).
[29] ‘OpenStreetMap’, *OpenStreetMap*. https://www.openstreetmap.org/about (accessed May 04, 2022).
[30] ‘sklearn.preprocessing.OneHotEncoder’, *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html (accessed May 09, 2022).

[31] ‘sklearn.preprocessing.LabelEncoder’, *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (accessed May 09, 2022).
[32] À. E. Nixon, ‘A common mistake to avoid when encoding categorical features’, *Medium*, Dec. 16, 2020. https://towardsdatascience.com/a-common-mistake-to-avoid-when-encoding-ordinal-features-79e402796ab4 (accessed May 09, 2022).
[33] ‘When to use LabelEncoder - Python Example’, *Data Analytics*, Aug. 10, 2020. https://vitalflux.com/when-use-labelencoder-python-example/ (accessed May 09, 2022).
[34] ‘6.3. Preprocessing data’, *scikit-learn*. https://scikit-learn/stable/modules/preprocessing.html (accessed May 10, 2022).
[35] ‘Coefficient of determination’, *Wikipedia*. Apr. 20, 2022. Accessed: May 10, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Coefficient_of_determination&oldid=1083670552
[36] ‘3.3. Metrics and scoring: quantifying the quality of predictions’, *scikit-learn*. https://scikit-learn/stable/modules/model_evaluation.html (accessed May 10, 2022).
[37] ‘Mean squared error’, *Wikipedia*. Apr. 26, 2022. Accessed: May 10, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=1084745972
[38] ‘Root-mean-square deviation’, *Wikipedia*. Aug. 06, 2021. Accessed: May 10, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Root-mean-square_deviation&oldid=1037360077
[39] ‘Mean absolute error’, *Wikipedia*. Nov. 03, 2021. Accessed: May 10, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mean_absolute_error&oldid=1053388699
[40] ‘3.1. Cross-validation: evaluating estimator performance’, *scikit-learn*. https://scikit-learn/stable/modules/cross_validation.html (accessed May 10, 2022).
[41] cs95, ‘Answer to “How to iterate over rows in a DataFrame in Pandas”’, *Stack Overflow*, Apr. 07, 2019. https://stackoverflow.com/a/55557758 (accessed Jun. 07, 2022).

Appendix A

Below is the python code to convert FCD text files into the PostgreSQL database.

```
import pandas as pd
import glob
from sqlalchemy import create_engine

# Importing and merging the files
data_path = glob.glob('C:/Users/Widiatmoko Azis F/Documents/_WAF Docs/SMACCs
Study/SMACCs Thesis/2. PROCESSING/Working_Data/*.txt')
df = pd.concat(pd.read_csv(f, sep = "\t", header = None) for f in data_path)
df.columns = ['osm_id', 'link_dir', 'date_time', 'speed_kmph', 'unique_entries']

# Converting date_time format
df['date_time'] = df['date_time'].astype('datetime64[ns]')

# Drop missing values (Na = not available data)
df = df.dropna(axis = 0)

# Connecting postgreSQL to python
# Engine configuration for postgreSQL,
# see for more detail: https://docs.sqlalchemy.org/en/14/core/engines.html
conn_string = 'postgresql+psycopg2://postgres:1234@localhost/thesis'

# Perform to_sql to convert df to SQL
db = create_engine(conn_string)
conn = db.connect()

print('Your data will be converted, Please wait...')
# Change the name of the table every run!
df.to_sql('network_speed_2021-22', con=conn, if_exists='replace', index=False)

print('Your data is in the database!')
```

Appendix B

Below is the SQL code to process and query the data input.

```
CREATE EXTENSION postgis;

-- Table: public.spatial_ref_sys

-- DROP TABLE public.spatial_ref_sys;

CREATE TABLE public.spatial_ref_sys
(
    srid integer NOT NULL,
    auth_name character varying(256) COLLATE pg_catalog."default",
    auth_srid integer,
    srtext character varying(2048) COLLATE pg_catalog."default",
    proj4text character varying(2048) COLLATE pg_catalog."default",
    CONSTRAINT spatial_ref_sys_pkey PRIMARY KEY (srid),
    CONSTRAINT spatial_ref_sys_srid_check CHECK (srid > 0 AND srid <= 998999)
)

TABLESPACE pg_default;

ALTER TABLE public.spatial_ref_sys
    OWNER to postgres;

GRANT ALL ON TABLE public.spatial_ref_sys TO postgres;

GRANT SELECT ON TABLE public.spatial_ref_sys TO PUBLIC;

-- create a new column to store time, day, and stores
-- for future processes, this should be done in a temporary way to save memory
ALTER TABLE network_speed
ADD column n_time VARCHAR,
add column hours Varchar,
add column mins Varchar,
ADD column n_day VARCHAR,
ADD column n_month VARCHAR,
ADD column stores VARCHAR;

-- fill the data with time extracted from the date_time column
-- for future processes, this should be done in a temporary way to save memory
UPDATE network_speed
SET n_time = TO_CHAR(network_speed.date_time, 'HH24:MI');

Update network_speed
set hours = to_char(network_speed.date_time, 'HH24');

Update network_speed
set mins = to_char(network_speed.date_time, 'MI');

-- fill the data with the day extracted from the date_time column
-- for future processes, this should be done in a temporary way to save memory
UPDATE network_speed
SET n_day = TO_CHAR(network_speed.date_time, 'DAY');

-- fill the data with the month extracted from the date_time column
-- for future processes, this should be done in a temporary way to save memory
UPDATE network_speed
SET n_month = TO_CHAR(network_speed.date_time, 'MONTH');

-- fill the data with store extracted from conditional case
```

```
-- for future processes, this should be done in a temporary way to save memory
UPDATE network_speed
SET stores = CASE
WHEN (TO_CHAR(network_speed.date_time, 'HH24:MI') BETWEEN '09:00' AND '20:59') AND
(TO_CHAR(network_speed.date_time, 'DAY')
IN ('TUESDAY  ','FRIDAY   ','THURSDAY ')) THEN 'OPEN'
WHEN (TO_CHAR(network_speed.date_time, 'HH24:MI') BETWEEN '09:00' AND '17:59') AND
(TO_CHAR(network_speed.date_time, 'DAY')
IN ('MONDAY   ','WEDNESDAY','SATURDAY ')) THEN 'OPEN'
WHEN (TO_CHAR(network_speed.date_time, 'HH24:MI') BETWEEN '07:30' AND '08:59') AND
(TO_CHAR(network_speed.date_time, 'DAY')
NOT IN ('SUNDAY   ')) THEN 'OPENING'
WHEN (TO_CHAR(network_speed.date_time, 'HH24:MI') BETWEEN '21:00' AND '22:00') AND
(TO_CHAR(network_speed.date_time, 'DAY')
IN ('TUESDAY  ','FRIDAY   ','THURSDAY ')) THEN 'CLOSING'
WHEN (TO_CHAR(network_speed.date_time, 'HH24:MI') BETWEEN '18:00' AND '19:00') AND
(TO_CHAR(network_speed.date_time, 'DAY')
IN ('MONDAY   ','WEDNESDAY','SATURDAY ')) THEN 'CLOSING'
ELSE 'CLOSED'
END;

-- Add network_speed_2018
-- Add network_speed_2019
-- Add network_speed_2020
-- Add network_speed_2021_22

-- for pilot processing, some road segments will be selected by following
-- the road segment that was previously selected
-- in the future, need to find a way to automate this process (probably by using the
segment coordinates)
select network_speed_2018.osm_id, network_speed_2018.date_time, net-
work_speed_2018.link_dir, network_speed_2018.speed_kmph,
network_speed_2018.n_time, network_speed_2018.hours, network_speed_2018.mins, net-
work_speed_2018.n_day, network_speed_2018.stores,
network_speed_2018.n_month, thessaloniki_road_network.highway, thessaloni-
ki_road_network.road_lt_m, thessaloniki_road_network.bus_stop
from network_speed_2018
left join thessaloniki_road_network
on network_speed_2018.osm_id = thessaloniki_road_network.osm_id
where network_speed_2018.osm_id in (
    112282638, 113342134, 176665208, 176665230, 176665233,
207312702, 357014247, 681636672, 681859683, 681859684,
681859685, 681859686, 681859687, 724288053, 724288054,
724298955, 724298956, 724298957, 724298958, 724298959,
724298960
)
union
select network_speed_2019.osm_id, network_speed_2019.date_time, net-
work_speed_2019.link_dir, network_speed_2019.speed_kmph,
network_speed_2019.n_time, network_speed_2019.hours, network_speed_2019.mins, net-
work_speed_2019.n_day, network_speed_2019.stores,
network_speed_2019.n_month,
thessaloniki_road_network.highway, thessaloniki_road_network.road_lt_m, thessaloni-
ki_road_network.bus_stop
from network_speed_2019
left join thessaloniki_road_network
on network_speed_2019.osm_id = thessaloniki_road_network.osm_id
where network_speed_2019.osm_id in (
    112282638, 113342134, 176665208, 176665230, 176665233,
207312702, 357014247, 681636672, 681859683, 681859684,
681859685, 681859686, 681859687, 724288053, 724288054,
724298955, 724298956, 724298957, 724298958, 724298959,
724298960
)
union
```

```
select network_speed_2020.osm_id, network_speed_2020.date_time, net-
work_speed_2020.link_dir, network_speed_2020.speed_kmph,
network_speed_2020.n_time, network_speed_2020.hours, network_speed_2020.mins, net-
work_speed_2020.n_day, network_speed_2020.stores,
network_speed_2020.n_month,
thessaloniki_road_network.highway, thessaloniki_road_network.road_lt_m, thessaloni-
ki_road_network.bus_stop
from network_speed_2020
left join thessaloniki_road_network
on network_speed_2020.osm_id = thessaloniki_road_network.osm_id
where network_speed_2020.osm_id in (
    112282638, 113342134, 176665208, 176665230, 176665233,
207312702, 357014247, 681636672, 681859683, 681859684,
681859685, 681859686, 681859687, 724288053, 724288054,
724298955, 724298956, 724298957, 724298958, 724298959,
724298960
)
union
select network_speed_2021_22.osm_id, network_speed_2021_22.date_time, net-
work_speed_2021_22.link_dir, network_speed_2021_22.speed_kmph,
network_speed_2021_22.n_time, network_speed_2021_22.hours, net-
work_speed_2021_22.mins, network_speed_2021_22.n_day, network_speed_2021_22.stores,
network_speed_2021_22.n_month,
thessaloniki_road_network.highway, thessaloniki_road_network.road_lt_m, thessaloni-
ki_road_network.bus_stop
from network_speed_2021_22
left join thessaloniki_road_network
on network_speed_2021_22.osm_id = thessaloniki_road_network.osm_id
where network_speed_2021_22.osm_id in (
    112282638, 113342134, 176665208, 176665230, 176665233,
207312702, 357014247, 681636672, 681859683, 681859684,
681859685, 681859686, 681859687, 724288053, 724288054,
724298955, 724298956, 724298957, 724298958, 724298959,
724298960
)
```

Appendix C

Below is the code used to predict the traffic speed for several scenarios for network-wide prediction.

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import LinearSVR, SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

import warnings
warnings.filterwarnings(action='ignore')

!python -V
# print(sklearn.__version__)

o_df = pd.read_csv('Tsimiski_st.csv', sep=',') # parse_dates = ['date_time'], in-
dex_col = ['date_time']

# Converting date_time format
o_df['date_time'] = o_df['date_time'].astype('datetime64[ns]')
o_df['year'] = o_df['date_time'].dt.year

o_df.dropna(axis=0, how='any', inplace=True)

sns.set(rc={'figure.figsize':(30,10)}, font_scale=2)
sns.lineplot(x=o_df['date_time'], y=o_df['speed_kmph'], hue=o_df['osm_id'])
plt.ylabel('Speed (Km/h)')
plt.xlabel('Date & Time')
plt.title('Speed Values in Tsimiski Street')

# filter the data for processing
df = o_df.loc[o_df['year'] == 2019]
df = df.loc[df['osm_id'] == 176665188] # Change to df = df.loc[df['osm_id'] ==
176665188] for multiple year analysis

#greater than the start date and smaller than the end date
mask = (df['date_time'] > '2019-01-01 00:00:00') & (df['date_time'] <= '2019-05-31
23:45:00') # selection format 2018-01-01 10:15:00
df = df.loc[mask]

# loop through id to create temp. df
osm = df['osm_id'].unique()

segments = {}
for i in range(len(osm)):
```

```
    segments['segment{}'.format(i+1)]=df[df['osm_id']==osm[i]]

ndf = pd.DataFrame()
# resample for each id
for segment in segments:
    dfs = segments[segment]
    dfs = dfs.set_index('date_time')
    dfs = dfs.resample('15T').interpolate()
    dfs = dfs.bfill()
# convert the type for some of the column
    dfs = dfs.astype({"osm_id":'int', "hours":'int', "mins":'int', "stores":'int',
"n_day":'int', "n_month":'int', "year":'int'})

# append all temp. into one df
    ndf = ndf.append(dfs)
    df = ndf


# Use only when multiple road segments are selected
df = df.reset_index()
df


# Split df into X and y
# selecting the prediction target (label)
y = df.speed_kmph

# convert all the features with label encoder
df['highway_n'] = LabelEncoder().fit_transform(df['highway'])
df['bus_stop_n'] = LabelEncoder().fit_transform(df['bus_stop'])
# selecting the 'features', depending on the data
data_features = ['hours', 'mins', 'n_day', 'stores']
X = df[data_features]


for col in df.columns:
    if len(df[col].unique()) == 1:
        df.drop(col,inplace=True,axis=1)

# Visualizing Correlation
sns.set(rc={'figure.figsize':(12,10)}, font_scale=2)
sns.heatmap(df.corr(), annot=True, vmin=-1.0, cmap='mako')
plt.title('Data Correlation')
plt.show()


# Train-test split
train_X, val_X, train_y, val_y = train_test_split(X, y, train_size=0.7, shuffle=True,
random_state=1)

# Set your custom color palette
customPalette = sns.set_palette(sns.color_palette("hls", 8))

ax = sns.set(rc={'figure.figsize':(30,10)}, font_scale=2)
ax = sns.lineplot(x=train_X['hours'], y=train_y, hue=train_X['osm_id']) # , la-
bel='Training Data', hue=train_X['osm_id']
ax = sns.lineplot(x=val_X['hours'], y=val_y, hue=val_X['osm_id']) # , label='Test
Data', hue=val_X['osm_id'], palette=customPalette

# Customize the axes and title
ax.set_title("Selected Data")
ax.set_xlabel("Hours")
ax.set_ylabel("Speed in Km/h")


# Scale X
print('Variance before scaler:', train_X.var(), sep='\n')
print('\n')
scaler = StandardScaler()
```

```
scaler.fit(train_X)
train_X = pd.DataFrame(scaler.transform(train_X), index=train_X.index, col-
umns=train_X.columns)
val_X = pd.DataFrame(scaler.transform(val_X), index=val_X.index, col-
umns=val_X.columns)

# Comparing the models
models = {
    "                      Linear Regression": LinearRegression(normalize=True), #
normalize=True
    "                    K-Nearest Neighbors": KNeighborsRegressor(n_neighbors=5,
weights='uniform', algorithm='brute', p=1), # n_neighbors=5, weights='uniform', algo-
rithm='brute', p=1
    "                          Neural Network": MLPRegressor(max_iter=500, ran-
dom_state=0, solver='lbfgs'), # max_iter=500, random_state=0, solver='lbfgs'
    "Support Vector Machine (Linear Kernel)": LinearSVR(C=1, epsilon=5), # C=1, epsi-
lon=5
    "   Support Vector Machine (RBF Kernel)": SVR(C=500, epsilon=5), # C=500, epsi-
lon=5
    "                           Decision Tree": DecisionTreeRegressor(max_depth=6), #
max_depth=6
    "                           Random Forest": RandomForestRegressor(max_depth=6,
max_features="log2", random_state=0), # max_depth=6, max_features="log2", ran-
dom_state=0
    "                       Gradient Boosting": GradientBoostingRegres-
sor(n_estimators=100, learning_rate=0.1, max_depth=3) # n_estimators=100, learn-
ing_rate=0.1, max_depth=3
}

for name, model in models.items():
    model.fit(train_X, train_y)
    print(name + " trained.")

for name, model in models.items():
    print(name + " R^2 Score: {:.5f}".format(model.score(val_X, val_y)))
    print(name + " R^2 Score: {:.5f}".format(model.score(train_X, train_y)))

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

for name, model in models.items():
    val_predict = model.predict(val_X)
    print(name + " R^2 Score: {:.5f}".format(r2_score(val_y, val_predict)))
    print(name + " RMSE: {:.5f}".format(np.sqrt(mean_squared_error(val_y,
val_predict))))
    print(name + " MAE: {:.5f}".format(mean_absolute_error(val_y, val_predict)))

# Specifying and creating the MODEL
# Change the model accordingly!
data_model = GradientBoostingRegressor()
data_model.fit(train_X, train_y)

# PREDICTING and VALIDATING the values
val_predictions = data_model.predict(val_X)

print('R^2 Score =', r2_score(val_y, val_predictions))
print('Mean Absolute Error =', mean_absolute_error(val_y, val_predictions))
print('Root Mean Squared Error =', np.sqrt(mean_squared_error(val_y,
val_predictions)))

# Implementing Cross Validation techniques
from sklearn.model_selection import cross_val_score
scores = cross_val_score(data_model, train_X, train_y, cv=10)
scores
```

```
# Inverse transform the X data for plot

print('Before Inverse Scaler:')
print("Training Data", train_X)
print("Test Data", val_X)

train_X = pd.DataFrame(scaler.inverse_transform(train_X), index=train_X.index, col-
umns=train_X.columns)
val_X = pd.DataFrame(scaler.inverse_transform(val_X), index=val_X.index, col-
umns=val_X.columns)

print("After Inverse Scaler:")
print("Training Data", train_X)
print("Test Data", val_X)


# Set your custom color palette
customPalette = sns.set_palette(sns.color_palette("hls", 8))

ax = sns.set(rc={'figure.figsize':(30,10)}, font_scale=2)
ax = sns.lineplot(x=val_X['hours'], y=val_y, label='Actual Value') # ,
hue=train_X['osm_id']
ax = sns.lineplot(x=val_X['hours'], y=val_predict, label='Predicted Value') # ,
hue=val_X['osm_id'], palette=customPalette

# Customize the axes and title
ax.set_title("Selected Data")
ax.set_xlabel("Hours")
ax.set_ylabel("Speed (Km/h)")


# Set your custom color palette
customPalette = sns.set_palette(sns.color_palette("hls", 8))

ax = sns.set(rc={'figure.figsize':(30,10)}, font_scale=2)
ax = sns.lineplot(x=val_X.index, y=val_y, label='Actual Value') # ,
hue=train_X['osm_id']
ax = sns.lineplot(x=val_X.index, y=val_predict, label='Predicted Value') # ,
hue=val_X['osm_id'], palette=customPalette

# Customize the axes and title
ax.set_title("Selected Data")
ax.set_xlabel("Hours")
ax.set_ylabel("Speed (Km/h)")


# Set your custom color palette
customPalette = sns.set_palette(sns.color_palette("hls", 8))

ax = sns.set(rc={'figure.figsize':(30,10)}, font_scale=2)
ax = sns.lineplot(x=val_X['hours'], y=val_y, hue=val_X['osm_id'],
style=val_X['osm_id']) # , label='Actual Value', hue=train_X['osm_id']
ax = sns.lineplot(x=val_X['hours'], y=val_predict, hue=val_X['osm_id'],
style=val_X['osm_id']) # , label='Predicted Value', hue=val_X['osm_id'], pal-
ette=customPalette

# Customize the axes and title
ax.set_title("Selected Data")
ax.set_xlabel("Hours")
ax.set_ylabel("Speed (Km/h)")
```

Appendix D

Below is the code for assessing the models for comparison with the preceding study.

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

df = df.reset_index()
train_mask = (df['date_time'] > '2019-01-01 00:00:00') & (df['date_time'] <= '2019-
02-14 23:45:00') # selection format 2018-01-01 10:15:00
wdf = df.loc[train_mask]

y = wdf.speed_kmph
data_features = ['hours', 'mins', 'n_day', 'stores']
X = wdf[data_features]

# Train-test split
train_X, val_X, train_y, val_y = train_test_split(X, y, train_size=0.7, shuffle=True,
random_state=1)

scaler = StandardScaler()
scaler.fit(train_X)
train_X = pd.DataFrame(scaler.transform(train_X), index=train_X.index, col-
umns=train_X.columns)
val_X = pd.DataFrame(scaler.transform(val_X), index=val_X.index, col-
umns=val_X.columns)

# Comparing the models
models = {
    "LR": LinearRegression(normalize=True), # normalize=True
    "KNN": KNeighborsRegressor(n_neighbors=5, weights='uniform', algorithm='brute',
p=1), # n_neighbors=5, weights='uniform', algorithm='brute', p=1
    "NN": MLPRegressor(max_iter=500, random_state=0, solver='lbfgs'), # max_iter=500,
random_state=0, solver='lbfgs'
    "SVR-L": LinearSVR(C=1, epsilon=5), # C=1, epsilon=5
    "SVR-RBF": SVR(C=500, epsilon=5), # C=500, epsilon=5
    "DT": DecisionTreeRegressor(max_depth=6), # max_depth=6
    "RF": RandomForestRegressor(max_depth=6, max_features="log2", random_state=0), #
max_depth=6, max_features="log2", random_state=0
    "GB": GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3)
# n_estimators=100, learning_rate=0.1, max_depth=3
}

for name, model in models.items():
    model.fit(train_X, train_y)
    print(name + " trained.")
    print(name + " R^2 Score: {:.5f}".format(model.score(val_X, val_y)))
    print(name + " R^2 Score: {:.5f}".format(model.score(train_X, train_y)))


# Scenario 1 test data (One Random Time on Random Days, Random Road)
test_mask = (df['date_time'] > '2019-02-15 00:00:00') & (df['date_time'] <= '2019-02-
28 23:45:00') # selection format 2018-01-01 10:15:00
wdf = df.loc[test_mask]
wdf = wdf.set_index('date_time')

result = pd.DataFrame()

result['Model'] = ''
result['MAE'] = ''
result['RMSE'] = ''

result = result.astype({"MAE":'int', "RMSE":'int'})
```

```
for i in range(35):
    ldf = wdf.sample()

    val_y = ldf.speed_kmph

    data_features = ['hours', 'mins', 'n_day', 'stores']
    val_X = ldf[data_features]

    val_X = pd.DataFrame(scaler.transform(val_X), index=val_X.index, col-
umns=val_X.columns)

    for name, model in models.items():
        val_predict = model.predict(val_X)

        result = result.append({'Model': name, 'MAE' : mean_absolute_error(val_y,
val_predict), 'RMSE': np.sqrt(mean_squared_error(val_y, val_predict))}, ig-
nore_index=True)

result.groupby('Model')['MAE'].describe().applymap(lambda x: f"{x:0.2f}")
result.groupby('Model')['RMSE'].describe().applymap(lambda x: f"{x:0.2f}")
sns.set(rc={'figure.figsize':(10,10)}, font_scale=1)
sns.boxplot(y=result['RMSE'], x=result['Model'], width=0.3)
plt.show()

# Scenario 2 test data (Eight Consecutive Times on Random Days, Random Road )
test_mask1 = (df['date_time'] > '2019-02-16 11:00:00') & (df['date_time'] <= '2019-
02-16 13:00:00') # selection format 2018-01-01 10:15:00
wdf = df.loc[test_mask1]
wdf = wdf.set_index('date_time')

val_y = wdf.speed_kmph

data_features = ['hours', 'mins', 'n_day', 'stores']
val_X = wdf[data_features]

val_X = pd.DataFrame(scaler.transform(val_X), index=val_X.index, col-
umns=val_X.columns)

print("Test 1")

result = pd.DataFrame()

result['Model'] = ''
result['MAE'] = ''
result['RMSE'] = ''
result['R2'] = ''

result = result.astype({"MAE":'int', "RMSE":'int', "R2":'int'})

for name, model in models.items():
    val_predict = model.predict(val_X)

    result = result.append({'Model': name, 'MAE' : mean_absolute_error(val_y,
val_predict), 'RMSE': np.sqrt(mean_squared_error(val_y, val_predict)),
'R2':r2_score(val_y, val_predict)}, ignore_index=True)

    # Set your custom color palette
    customPalette = sns.set_palette(sns.color_palette("hls", 8))

    ax = sns.set(rc={'figure.figsize':(30,10)}, font_scale=2)
    ax = sns.lineplot(x=val_X.index, y=val_predict, label=name) # ,
hue=val_X['osm_id'], palette=customPalette
```

```
ax = sns.lineplot(x=val_X.index, y=val_y, label='Actual Value', dashes=True) # ,
hue=train_X['osm_id']

# Customize the axes and title
ax.set_title("Prediction Result Comparison")
ax.set_xlabel("Hours")
ax.set_ylabel("Speed (Km/h)")
plt.legend(loc='lower right')

result.groupby('Model')['MAE'].describe().applymap(lambda x: f"{x:0.2f}")
result.groupby('Model')['RMSE'].describe().applymap(lambda x: f"{x:0.2f}")
result.groupby('Model')['R2'].describe().applymap(lambda x: f"{x:0.2f}")

# Scenario 3 test data (Random Days, Random Roads)
test_mask1 = (df['date_time'] > '2019-02-24 00:00:00') & (df['date_time'] <= '2019-
02-24 23:45:00') # selection format 2018-01-01 10:15:00
wdf = df.loc[test_mask1]
wdf = wdf.set_index('date_time')

val_y = wdf.speed_kmph

data_features = ['hours', 'mins', 'n_day', 'stores']
val_X = wdf[data_features]

val_X = pd.DataFrame(scaler.transform(val_X), index=val_X.index, col-
umns=val_X.columns)

print("Test 1")

result = pd.DataFrame()

result['Model'] = ''
result['MAE'] = ''
result['RMSE'] = ''
result['R2'] = ''

result = result.astype({"MAE":'int', "RMSE":'int', "R2":'int'})

for name, model in models.items():
    val_predict = model.predict(val_X)

    result = result.append({'Model': name, 'MAE' : mean_absolute_error(val_y,
val_predict), 'RMSE': np.sqrt(mean_squared_error(val_y, val_predict)),
'R2':r2_score(val_y, val_predict)}, ignore_index=True)

    # Set your custom color palette
    # plt.figure()
    ax = sns.set(rc={'figure.figsize':(30,10)}, font_scale=2)
    ax = sns.lineplot(x=val_X.index, y=val_predict, label=name+' Predicted Value') #
, hue=val_X['osm_id'], palette=customPalette

ax = sns.lineplot(x=val_X.index, y=val_y, label=' Actual Value') # ,
hue=train_X['osm_id']

# Customize the axes and title
ax.set_title("Prediction Result Comparison")
ax.set_xlabel("Hours")
ax.set_ylabel("Speed (Km/h)")
plt.legend(loc='upper right')

result.groupby('Model')['MAE'].describe().applymap(lambda x: f"{x:0.2f}")
result.groupby('Model')['RMSE'].describe().applymap(lambda x: f"{x:0.2f}")
result.groupby('Model')['R2'].describe().applymap(lambda x: f"{x:0.2f}")
```