# Real time map matching for low frequency GPS data based on machine learning technology

Miss Ornicha Sinthopvaragul



A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Computer Science

Department of Computer Engineering

FACULTY OF ENGINEERING

Chulalongkorn University

Academic Year 2022

# การประสานพิกัดแผนที่ของข้อมูลจีพีเอสความถี่ต่ำด้วยเทคโนโลยีแมชชีนเลิร์นนิง

น.ส.อรนิชา สินธพวรกุล



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

| | |
|---|---|
| Thesis Title | Real time map matching for low frequency GPS data based on machine learning technology |
| By | Miss Ornicha Sinthopvaragul |
| Field of Study | Computer Science |
| Thesis Advisor | Associate Professor VEERA MUANGSIN |

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

..... Dean of the FACULTY OF ENGINEERING

(Professor SUPOT TEACHAVORASINSKUN)

THESIS COMMITTEE

..... Chairman

(Associate Professor KRERK PIROMSOPA)

..... Thesis Advisor

(Associate Professor VEERA MUANGSIN)

..... Examiner

(Associate Professor SORAWIT NARUPITI)

..... External Examiner

(Associate Professor Teerayut Horanont)

อรนิชา สินธพวรกุล : การประสานพิกัดแผนที่ของข้อมูลจีพีเอสความถี่ต่ำด้วยเทคโนโลยีแมชชีนเลิร์นนิง. ( Real time map matching for low frequency GPS data based on machine learning technology) อ.ที่ปรึกษาหลัก : รศ. ดร.วีระ เหมืองสิน

ข้อมูล GPS ในพื้นที่ชุมชนเมืองอาจมีความคลาดเคลื่อนสูงเนื่องจากผลของความคลาดเคลื่อนของการสะท้อนของสัญญาณไปยังเครื่องรับสัญญาณ GPS ( Multipath Problem) ดังนั้นจึงต้องมีกระบวนการในการปรับแก้ข้อมูล GPS เพื่อลดถอนความคลาดเคลื่อนเชิงตำแหน่งได้แก่ กระบวนการประสานพิกัด(Map matching method) อย่างไรก็ตามกระบวนการนี้ต้องใช้ข้อมูลเชิงตำแหน่งที่หนาแน่นและจำนวนมากเพื่อใช้ในการประมวลผล รวมถึงเมื่อนำมาใช้ปรับแก้ในข้อมูล GPS ที่มีความหนาแน่นน้อย(GPS low-frequency Data) กระบวนการ Map matching นี้จะมีความแม่นยำและประสิทธิภาพ ดังนั้นในงานศึกษานี้จึงศึกษาเทคโนโลยีแมชชีนเลิร์นนิง(machine learning technology) และ กระบวนการจำแนกประเภท( classification methods)เพื่อทำการปรับแก้ข้อมูล GPS ความถี่ต่ำ รวมถึงเพื่อพัฒนาประสิทธิภาพในการประสานพิกัดแผนที่ งานวิจัยนี้ได้มีการนำคุณลักษณะของข้อมูล GPS บางประการมาใช้กับแมชชีนเลิร์นนิง โมเดล เพื่อเพิ่มความแม่นยำได้แก่ ความเร็วของรถยนต์ ทิศทางการเคลื่อนที่ของรถยนต์ และ ตำแหน่งของ GPS ก่อนที่จะทำการปรับแก้

จุดประสงค์ของงานวิจัยนี้คือพัฒนากระบวนการประสานพิกัดแผนที่ของข้อมูล GPS ความถี่ต่ำ และประสิทธิภาพในกระบวนการประมวลผลเพื่อลดระยะเวลาในการประมวลผลด้วยเทคโนโลยีแมชชีนเลิร์นนิง จากผลการศึกษาจะพบว่า Radom forest model ให้ผลความถูกต้องแม่นยำที่สุดทั้งในค่าความแม่นยำ(precision) ค่ารีคอล (recall) ค่าความถูกต้อง(accuracy) และค่า F1-score รวมถึงโมเดลนี้ยังมีค่าความอ่อนไหวต่ำ(sensitivity)เมื่อทำการเปลี่ยนความถี่ของข้อมูล อย่างไรก็ตามเมื่อเปรียบเทียบในเรื่องระยะเวลาในการปรับแก้ Decision tree เป็นโมเดลที่สามารถประมวลผลได้รวดเร็วที่สุด และใช้เวลาในการประมวลผลเพื่อสร้างโมเดล(Training model)รวดเร็วกว่า Random forest

ผลสรุปจากการศึกษาพบว่าเทคโนโลยีแมชชีนเลิร์นนิงสามารถพัฒนากระบวนการประสานพิกัดแผนที่ข้อมูล GPS ที่มีความถี่ต่ำ และ Random forest model เป็นโมเดลที่ให้ผลลัพธ์ในการปรับแก้เชิงตำแหน่งแม่นยำที่สุด ในขณะที่ในด้านความรวดเร็วในการประมวลผล Decision tree สามารถประมวลผลได้รวดเร็วที่สุด ดังนั้นการใช้โมเดลแมชชีนเลิร์นนิงที่ปรับแก้ข้อมูลได้รวดเร็วและแม่นยำจะสามารถนำมาประยุกต์ใช้ในหลากหลายแอพพลิเคชั่น



| | | |
|---|---|---|
| สาขาวิชา | วิทยาศาสตร์คอมพิวเตอร์ | ลายมือชื่อนิสิต ..... |
| ปีการศึกษา | 2565 | ลายมือชื่อ อ.ที่ปรึกษาหลัก ..... |

# # 6470304921 : MAJOR COMPUTER SCIENCE

KEYWORD: Map matching, GPS, Machine learning, Low frequency data, Random Forest

Ornicha Sinthopvaragul : Real time map matching for low frequency GPS data based on machine learning technology. Advisor: Assoc. Prof. VEERA MUANGSIN

GPS accuracy can be compromised in urban areas due to multipath issues, leading to low accuracy in GPS location. Therefore, map matching has been introduced as a method to reduce the GPS location error. However, current map matching methods require high-frequency GPS data, and may not perform well with low-frequency data. To address this issue, this study explores the use of machine learning technology and classification methods to adjust the low-frequency GPS location. The model developed in this research employs features such as speed, heading, and location of the previous GPS point for machine learning training, ultimately leading to more accurate map matching for low-frequency GPS data.

The objective of this study is to develop a more accurate and efficient map matching method for low-frequency GPS data using machine learning techniques. Experiment results show that the Random Forest machine learning model produced the highest precision, recall, and F1-Score. Additionally, this model is less sensitive to changes in data frequency. However, the Decision Tree model was the fastest in terms of prediction time and required less time for training than the Random Forest model.

In conclusion, this study shows that machine learning technology can be used to improve map matching methods for low-frequency GPS data. The Random Forest model appears to be the most effective in terms of accuracy, while the Decision Tree model is the fastest. These findings can be used to optimize map matching for a wide range of applications, including transportation and navigation.



| | | |
|---|---|---|
| Field of Study: | Computer Science | Student's Signature ..... |
| Academic Year: | 2022 | Advisor's Signature ..... |

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Associate Professor VEERA MUANGSIN , Associate Professor KRERK PIROMSOPA, Associate Professor SORAWIT NARUPITI and Associate Professor Teerayut Horanont who has give advice and suggestion to this thesis. Without their support, this accomplishment would not have been possible.

First and foremost, I would like to thank Associate Professor VEERA MUANGSIN my advisor for their guidance, encouragement, and unwavering support throughout this journey. Their expertise, insights, and feedback have been invaluable in shaping my study.

I would also like to thank my committee Associate Professor KRERK PIROMSOPA ,Associate Professor SORAWIT NARUPITI and Associate Professor Teerayut Horanont for their suggestion and commence for improve my study.

I am grateful to my advisor and the committee who generously shared their time and insights, without whom this research would not have been possible. Their willingness to participate in this study was crucial to its success.

Finally, I would like to acknowledge the support of my family. Their love, patience, and understanding have sustained me during this challenging period, and I could not have done this without them.

Once again, thank you to everyone who has helped and supported me in this endeavor.

Ornicha Sinthopvaragul

# TABLE OF CONTENTS

**Page**

# LIST OF FIGURES

**Page**

## Conference information

This study was published in the 8th International Conference on Business and Industrial Research (ICBIR2023), a multidisciplinary academic conference that includes research in science, engineering, informatics, modern business, social science, and linguistics. The conference has been organized by TNI and TPA since 2010.

This paper was presented on May 18, 2023, in the topic of Intelligent Transportation Systems and Assistive Technology for Vehicles and Transportation Systems at Thai-Nichi Institute of Technology. This study was published into IEEE Xplore.

# Introduction

GPS (Global Positioning System) is a type of positioning technology widely used in vehicles and mobile devices. Basically, a GPS receiver calculates its current geolocation based on signals from multiple GPS satellites. The main applications of GPS include finding the current location, tracking movement, and navigation. Moreover, GPS data collected from many receivers over time, so-called GPS probe data, can be used for analytical purposes. However, GPS is prone to position errors, especially in urban areas where buildings and other constructions can block and reflect satellite signals. When GPS is used for road navigation or traffic analysis, the GPS locations must be matched with the road where the vehicle is on. Moreover, the raw GPS location must be 'snapped' onto the best-estimated location on the road. This process is called 'map-matching'. To mitigate the effects of GPS error, most map-matching algorithms work on a subsequence of GPS locations that represent the trajectory of the vehicle. These algorithms are originally based on the Markov-chain method. Recently, machine learning methods have been applied to the problem. Map-matching algorithms usually perform more accurate when the input GPS locations are collected with a high frequency, e.g. one record every a few seconds. However, collecting probe data at a higher frequency requires more storage, communication, and power. Moreover, some applications of GPS probe data such as tracking the locations of vehicle fleets, i.e. trucks, buses, or taxis, do not require high-frequency data (frequency less than 30 sec [1], [2]). For example, Intelligent Traffic Information Center (iTIC) has collected GPS probe data of thousands of taxis at the frequency of one record per minute. Unfortunately, typical map matching algorithms do not perform well with such low-frequency data [3]. In addition, traffic analysis of historical probe data must perform map matching on a large dataset, which is a very time-consuming process. So this research tries to improve the map-matching method for adjusting GPS location by using low-frequency GPS data (frequency 1-5 min [3]) and a machine learning model to detect a pattern of data and reduce the error of GPS location. Hidden Markov Model has long been the basis of many match matching algorithms [4], [5]. However, the technique works accurately for up to 30 second sample period [4]. As a result, many attempts to improve the accuracy of map matching for lower-frequency data have been proposed. Some are based on resampling to obtain higher frequency data, e.g., the collaborative map matching method (CMM)

[6]. Some tried to minimize the number of candidate routes using a multi-criteria dynamic programming [7] or shortest path algorithm [1], [8]. Many research study Markov model for improve map matching method such as study Makov model and shortest path [9] to find top candidate and adjust GPS point. Some research study HMM-RCM [10] to map matching data. Hidden Makov model was used for map matching location point with topology data and other feature [11]. Recently, map matching algorithms based on machine learning have gained more popularity. Some approach the map-matching task as a regression problem and create a neural network model that reduces GPS errors [12]. Others approach it as a classification problem and use techniques like random forest [13]. Our work takes the latter approach.

## Objectives

The objectives of this thesis are:

1) To develop an accurate and fast map matching method from low-frequency GPS probe data based on machine learning

2) To apply the map matching method to traffic analysis tasks

## Scope of research

The objective of the study is to develop a map-matching method that can accurately adjust GPS locations for low-frequency data. The study uses historical raw vehicle GPS location data from Grab, a ride-hailing company, to train a machine learning model. To reduce the density of GPS points, the researchers select a subset of the data for training. The model is then used to predict and adjust GPS locations collected at a lower frequency (one record per minute) from the Intelligent Traffic Information Center (iTIC) in Bangkok, Thailand. OpenStreetMap is used as a reference for matching and adjusting the GPS location on the road. The study focuses on popular and high sample dataset roads in Bangkok, Thailand.

The focus of this study is the popular road in Bangkok, Thailand, which serves as a high-sampling dataset as below:

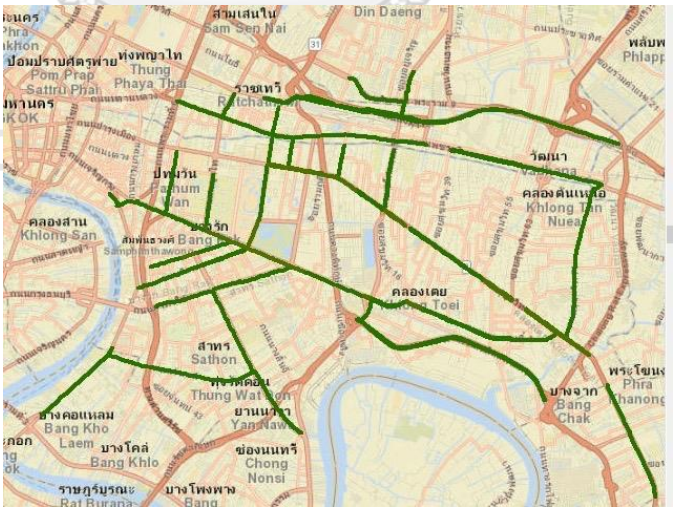



*Figure 1 Road data for training model.*

# Background

## The Global Positioning System (GPS)

The Global Positioning System (GPS) is a satellite-based global navigation system that provides information about a user's location, velocity, and time synchronization. GPS receivers are commonly integrated into mobile devices and vehicles for movement tracking and navigation purposes. In order to determine a location, GPS requires at least four satellites to provide accurate signals.

However, urban areas with high-rise buildings, trees, tunnels, highways, and other structures can interfere with and reflect satellite signals, resulting in signal weaknesses, delays, and multipath errors. This can negatively impact the accuracy of GPS location tracking.

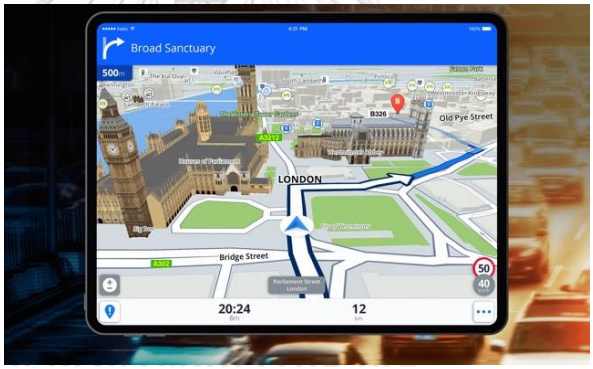



*Figure 2 Navigation System*

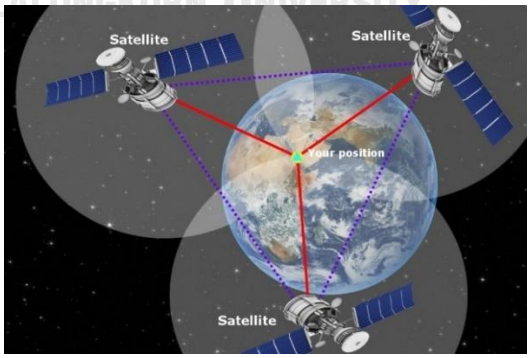



*Figure 3 Global Positioning System*

## Multipath Errors

Multipath errors are a common problem that affects the accuracy of GPS location. This error occurs when GPS signals bounce off objects such as buildings, water, and other reflective

surfaces before reaching the receiver. This causes a time delay in the signal and can result in errors in the calculated position. Multipath errors are more likely to occur in urban areas where there are tall buildings and other structures that can reflect and block the GPS signal, which can lead to inaccuracies in GPS data.

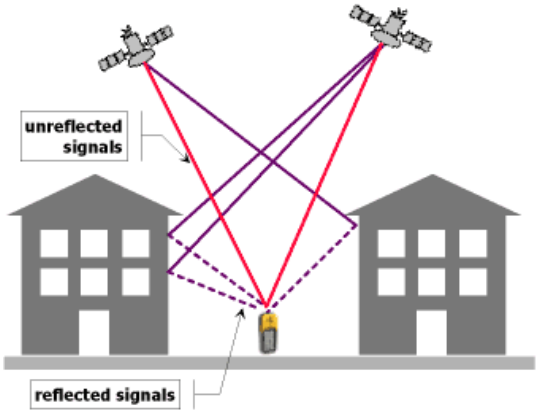



*Figure 4 Multipath Errors*

## OpenStreetMap (OSM)

OpenStreetMap is a freely accessible online map created and updated by a community of mappers who contribute and maintain data of roads, landmarks, and other geographic features. In OpenStreetMap, a road is comprised of multiple road segments, with each segment containing an ID, an ordered list of nodes, and other attributes like type and name. A node in the map refers to a single point on the map with a unique ID and geographic coordinates, usually latitude and longitude.



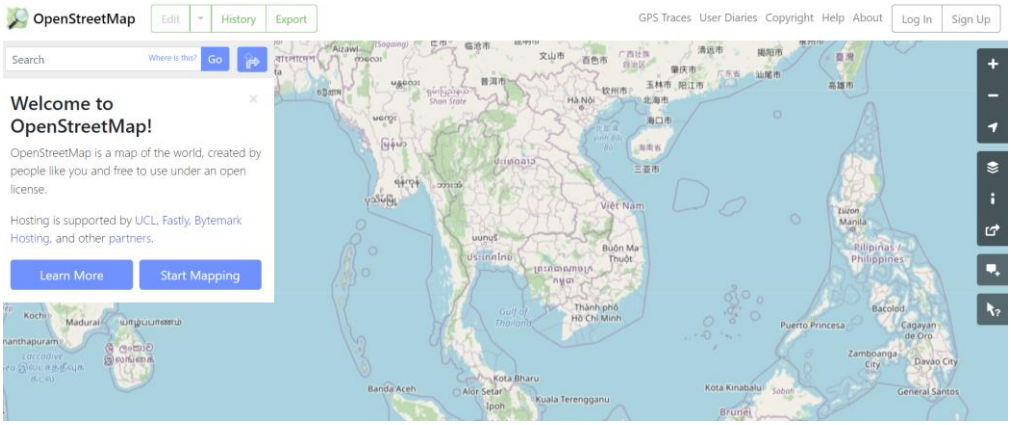



*Figure 5 OpenStreetMap*

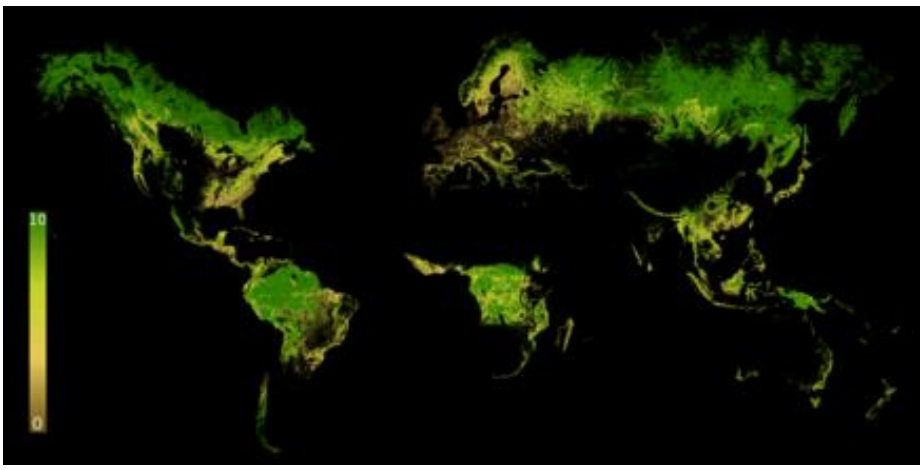



*Figure 6 Forest Landscape Integrity Index*

## Map matching method

As GPS errors can significantly impact analytical work, traffic analysis, and navigation systems, map matching is a crucial technique that matches recorded geographic coordinates to a logical model of the real world. This is typically achieved using some form of Geographic Information System (GIS). By implementing this method, the error of GPS location can be minimized, and GPS locations can be adjusted to align with the actual road network, thus improving the overall accuracy of the data.

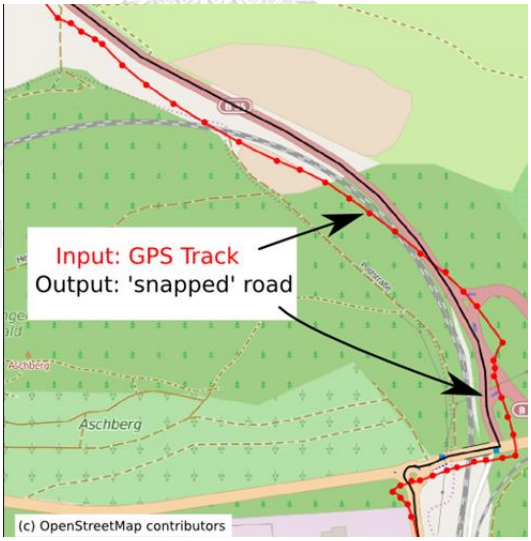



*Figure 7 Map matching method*

An example use case of map matching method is traffic analysis, where road attributes are defined and multiple GPS tracks are used to identify the restricted areas. This helps to identify traffic congestion, traffic flow, and other traffic-related problems, which are crucial for traffic management and planning. Map matching can also help in identifying the location of accidents,

incidents, and road closures, which can be critical information for emergency services and commuters. By using map matching, traffic analysis becomes more accurate and reliable, enabling efficient and effective traffic management.

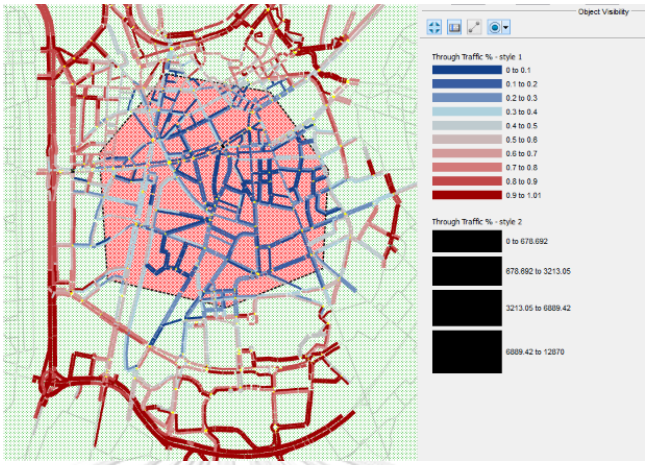



*Figure 8 Traffic analysis*

## Bearing

"Bearing" is a term used to describe the horizontal angle between an object and north, or another object and the direction of a vehicle. There are two types of bearings: absolute bearing and relative bearing.

Absolute bearing is the clockwise angle between magnetic north and true north. It gives a precise indication of direction and is commonly used in navigation.



Relative bearing, on the other hand, provides the angle between the heading of a vehicle and the location of another object. This type of bearing is commonly used in aviation, maritime navigation, and other applications where it is necessary to maintain a specific heading or course.

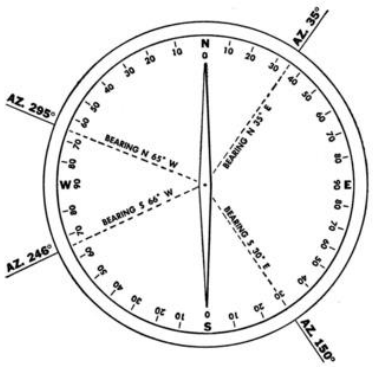



*Figure 9 Bearings*

## Decision Tree

A decision tree is a supervised machine learning method that is commonly used for creating models to predict outcomes for both classification and regression problems. It works by partitioning the data into smaller subsets based on the values of the attributes in the dataset, and then recursively repeating this process for each subset until a decision can be made. The main goal is to create a tree that is as small as possible while still accurately predicting outcomes.

In classification problems, decision trees can be used to classify data into different categories, such as spam or non-spam emails, based on a set of input attributes. The decision tree algorithm uses a measure of information gain to determine which attribute to split the data on at each node, in order to minimize the overall cost function and tree size. By repeatedly splitting the data based on the attribute that provides the most information gain, the decision tree algorithm can create a model that accurately classifies new data.

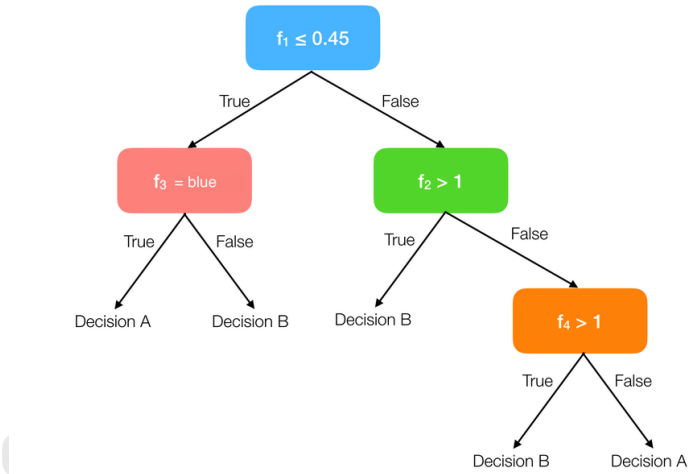



*Figure 10 Decision Tree*

## Random Forest

Random Forest is a type of supervised and ensemble learning method used for classification tasks in machine learning. It consists of multiple decision trees, each of which predicts a result. The final prediction is obtained by taking the majority vote of the output of all the decision trees in the forest

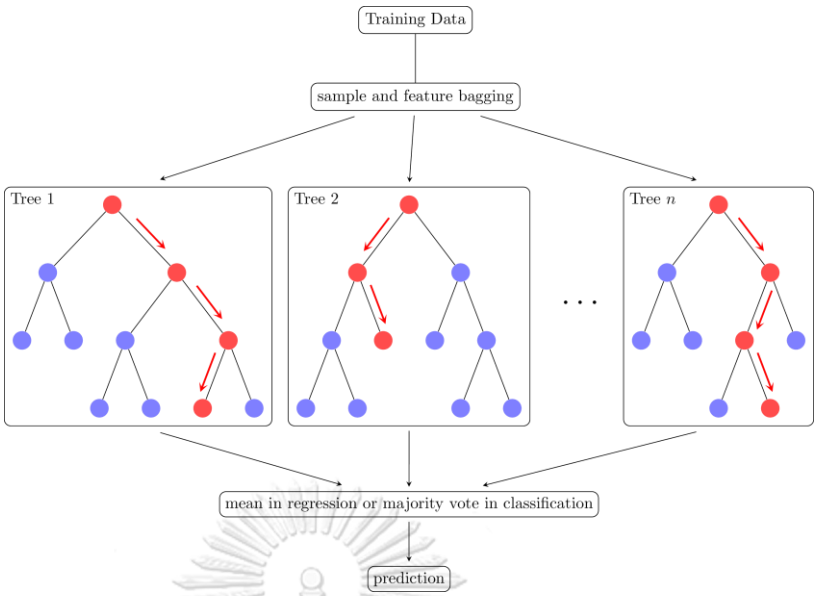



*Figure 11 Random Forest*

## K-Nearest Neighbor (K-NN)

K-Nearest Neighbor (K-NN) is a supervised machine learning classification model that uses distance to find the similarities between the input value and other classes in order to make predictions. The K-NN algorithm works by selecting the K closest data points from the training set to the new data point and determining the class with the highest frequency among these neighbors. K-NN can be used for both binary and multi-class classification problems, as well as regression problems by taking the average of the K-nearest neighbors.



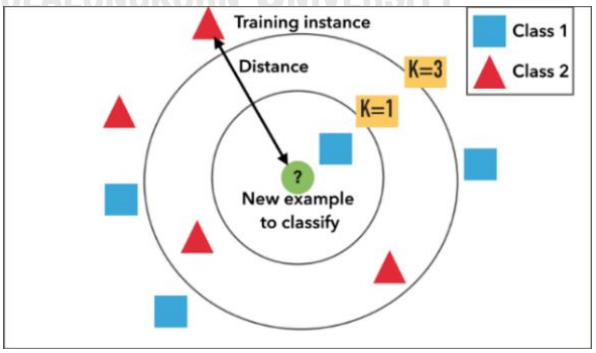



*Figure 12 K-NN*

## Long short-term memory (LSTM)

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) that excels at processing sequence data, such as speech and text. Unlike traditional RNNs, which have

difficulty retaining information over long periods of time, LSTMs use a gating mechanism to selectively remember or forget information, allowing them to better handle long sequences and avoid the vanishing gradient problem. As a result, LSTMs have become a state-of-the-art model for tasks such as language modeling, machine translation, and speech recognition.

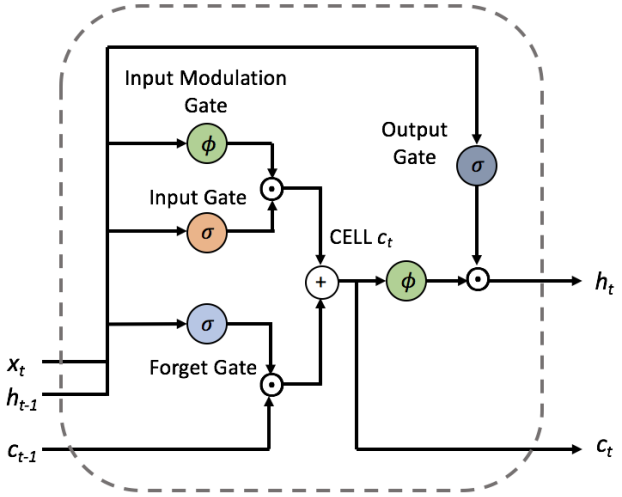



*Figure 13 LSTM*

## XGBoost

XGBoost is a supervised machine learning model that has been developed from the Gradient Boosting model to improve its performance. It can be used for both regression and classification problems. XGBoost is an ensemble learning method that learns from multiple learners. It works by creating a sequence of decision trees that are trained iteratively, with each new tree attempting to correct the errors of the previous ones. The final prediction is made by combining the predictions of all the trees. XGBoost is known for its speed and accuracy, and is widely used in machine learning competitions and industry applications.


*Figure 14 XGBoost*

### A multilayer perceptron (MLP)

A multilayer perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected nodes or neurons. Each neuron in one layer is connected to every neuron in the next layer. MLP is a supervised learning model that uses the backpropagation algorithm for training. The backpropagation algorithm involves a forward pass to compute the output and a backward pass to update the weights and parameters based on the error. MLPs are widely used in various applications, such as image and speech recognition, natural language processing, and predictive analytics.

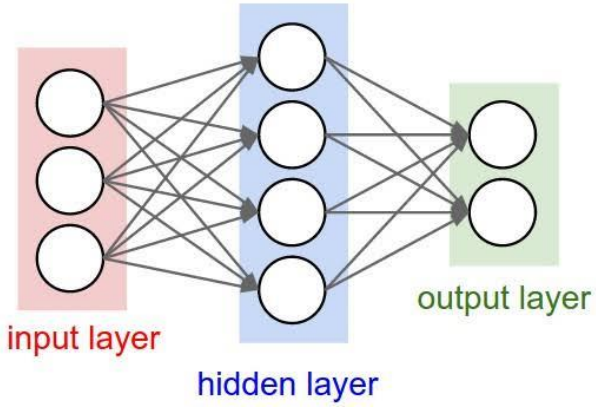



*Figure 15 Neural Network*

### K-means clustering

"K-means" is an unsupervised learning algorithm that does not require labeled data for training. The number of clusters or "K" is a required parameter of the K-means algorithm. The "elbow technique" is a method for finding the optimal value of "K". Many applications use K-means for customer segmentation.

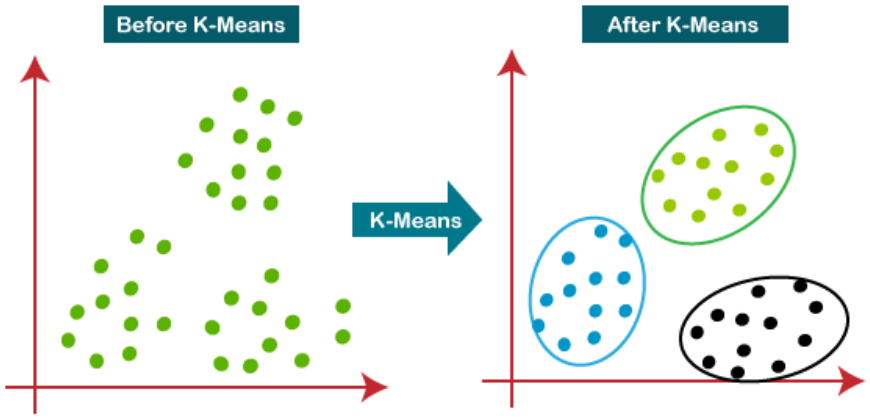



*Figure 16 K-Means Clustering*

### DBSCAN (Density-based spatial clustering of applications with noise)

DBSCAN is an unsupervised learning algorithm that stands for Density-based spatial clustering of applications with noise. It uses the density of data points to group them into clusters, while also identifying noise points. DBSCAN has two main parameters: epsilon (eps), which defines the radius around each data point, and MinPts, which is the minimum number of data points required to form a dense region. DBSCAN is particularly well-suited for spatial and location-based applications.

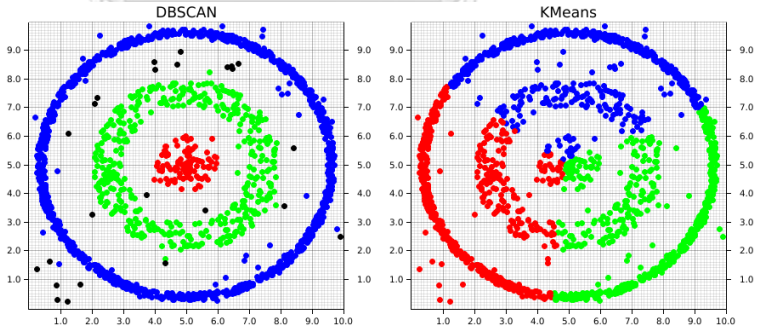



*Figure 17 Compare DBSCAN and KMeans*

### Hidden Markov Model(HMM)

A Hidden Markov Model (HMM) is a statistical model used to describe the probabilistic relationship between a sequence of observations and a sequence of hidden states. It is particularly useful when the underlying system or process that generates the observations is unknown or concealed, which is why it is called a "Hidden" Markov Model. HMMs are employed to predict

future observations or classify sequences based on the underlying hidden process that generates the data. An HMM consists of two types of variables: hidden states and observations.

The hidden states are the underlying variables responsible for generating the observed data, but they cannot be directly observed. On the other hand, the observations are the variables that are measured and recorded.

The relationship between the hidden states and the observations is modeled using probability distributions. In an HMM, this relationship is defined by two sets of probabilities: the transition probabilities and the emission probabilities.

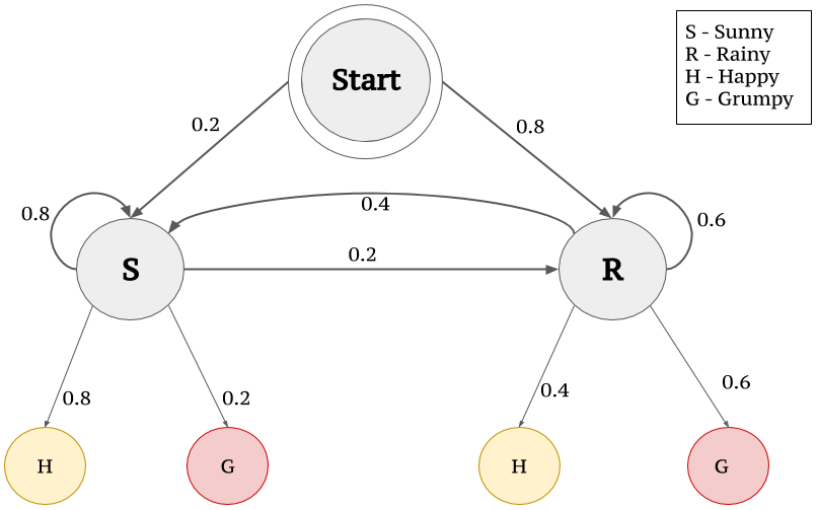





*Figure 18 Hidden Markov Model*

# Related Works

## DRFMM: a map-matching algorithm based on distributed random forest multi-classification[13]

This study aims to enhance the speed and accuracy of map matching methods for adjusting vehicle location using machine learning and Spark technology. To achieve this, the research utilizes past vehicle data to train machine learning models using the random forest algorithm. By leveraging Spark, the training model process can be accelerated and parallelized, resulting in faster processing times. Additionally, by dividing the road network into a grid system, the speed and parallel processing capabilities can be further improved. To calculate the grid ID, the following formula is used to calculate grid ID as below:

$$\boldsymbol{ID} = \text{int}\left(\frac{N \times (lon - lon_0)}{L}\right) + \text{int}\left(\frac{\text{M} \times (\text{lat} - \text{lat}_0)}{\text{H}}\right) \times \text{N}$$

*Figure 19 formular for calculate grid ID*

This study presents a distributed random forest map-matching algorithm (DRFMM) that can significantly enhance the accuracy of map machine process by up to 10%. The improvement in accuracy depends on the volume of data, particularly GPS coordinates collected in the past. The DRFMM method demonstrated superior performance compared to traditional methods by reducing data processing time by approximately 6 times. This was achieved through the use of Spark, an open-source platform for distributed processing of big data. The application of Spark significantly improved the machine learning process, thereby improving data processing speed.



This study developed and designed the map matching process in two modes. The first mode is the offline mode used in the process training model step with parallel processing to improve speed and reduce processing time. The second mode is the online mode used in the process prediction step.

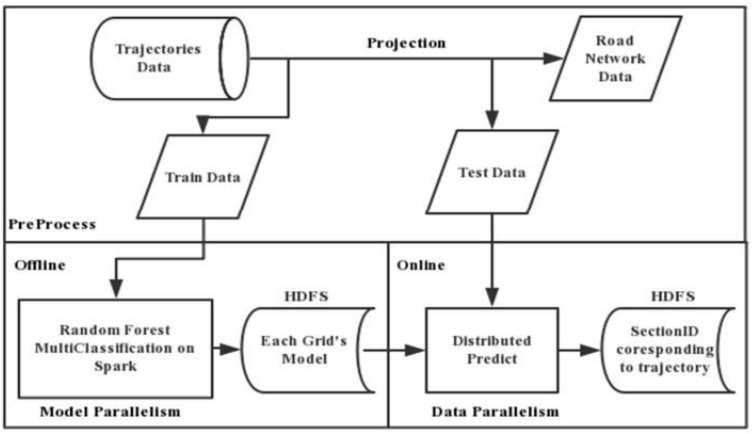



*Figure 20 Overall architecture of the DRFMM model*

A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories[6]

This study aims to improve the accuracy of map matching method using a low sampling GPS dataset. The map matching method is performed using a collaborative map matching method (CMM) and resampling data by a clustering technique, which is an unsupervised matching learning method for grouping similar GPS data. This is because the map matching method does not perform well with low sample data, and low accuracy from the adjustment by the map matching method can affect the use of navigation systems, traffic analysis, and other works. The study uses the longest common subsequence (LCSS) distance for the matching process. The paper outlines the development process in three steps as follows:



The first step of this study is the data preprocessing phase which involves preparing the GPS data prior to map matching. This step includes segmenting the road networks using R-tree and Path-Forest algorithms. Before processing the GPS data, outliers were removed and the data was cleansed to ensure accuracy in the subsequent steps.

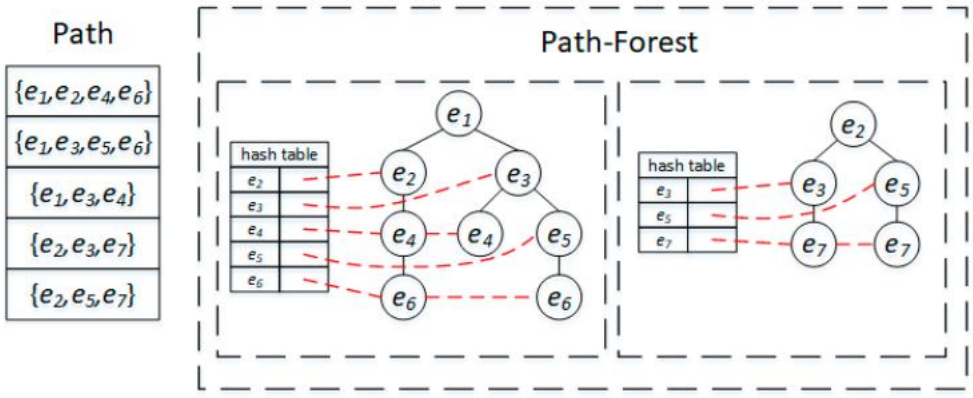



*Figure 21 The Path-Forest.*

The next step is the Trajectory Clustering & Resampling step, which involves using a clustering algorithm called DBSCAN to group similar data and a sliding window algorithm to increase the number of sample data (Resampling). This step aims to improve the accuracy of the map matching method by increasing the amount of available data.

The final step is Map matching, which involves adjusting the raw GPS locations to improve the accuracy of the location. This step utilizes the longest common subsequence (LCSS) distance for the matching process to improve the accuracy of the map matching method for low sampling GPS datasets.

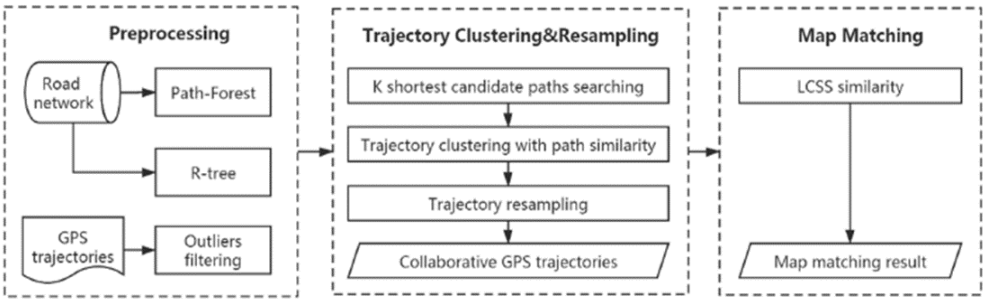



*Figure 22 The framework of CMM algorithm.*

### Map Matching Integration Algorithm Based on Historical Trajectory Data[14]

This research study employs a Hidden Markov Model to adjust GPS locations and the map matching process. Additionally, the study utilizes the DBSCAN algorithm, an unsupervised

machine learning method, to cluster data and adjust segmentation. The flowchart for the map matching process is shown below.

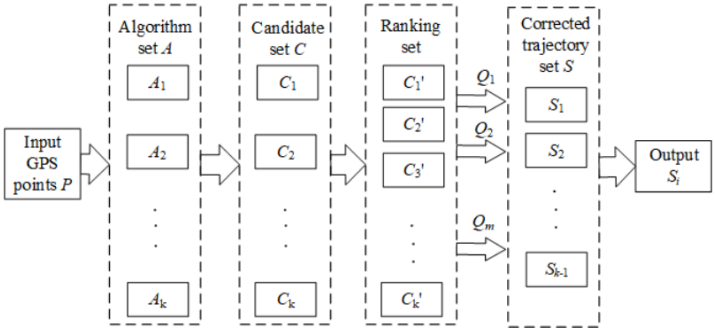



*Figure 23 Algorithm flow chart.*

This study demonstrates that using a combination of algorithms for map matching methods can enhance the performance and accuracy of GPS location adjustment. Conversely, relying solely on a single map matching algorithm has limitations and may not perform effectively in some situations due to the uncertainty of GPS location data.

(a) Projection algorithm different segment result graph.

(b) HMM algorithm different segment result graph.

*Figure 24 Different segment result graph.*



A Machine Learning Approach to Improve the Accuracy of GPS-Based Map Matching Algorithms[12]

This research aims to enhance the accuracy of the map matching method by utilizing an artificial neural network (ANN) to adjust GPS points and reduce the horizontal error. The ANN model is trained using various GPS data, including GPS points, speed, HDOP, and horizontal error. Once the accuracy of the neural network model meets the acceptable criteria, it is used to adjust the GPS points and identify road segments. This adjustment helps in projecting points for training the neural network model, ultimately improving the overall algorithm performance.

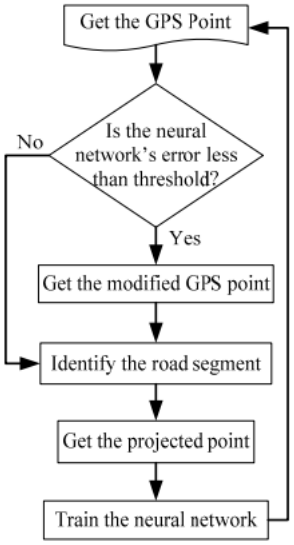



*Figure 25 The proposed integration methodology.*

Map-matching for low-sampling-rate GPS trajectories[1]

This research uses ST-Matching, a map matching algorithm that utilizes the road network, topology, and speed of the car to improve the accuracy of the map matching process. The study consists of three main steps for the map matching process.

The first step is candidate preparation, which involves using GPS data and road data to generate a candidate set. The candidate set is a set of potential road segments that the vehicle could have traveled on based on its GPS data.

The next step is Spatial and Temporal Analysis, which considers both the GPS point and the candidate set. This step also utilizes the speed of the vehicle to analyze and select the most likely candidate for the vehicle's location at that point in time.

The final step is to use the candidate graph to create a result matching. This step involves selecting the most likely path for the vehicle to have traveled based on the candidates generated in the previous steps. By utilizing both spatial and temporal data along with the road network and speed data, the ST-Matching algorithm can provide a more accurate map matching result.

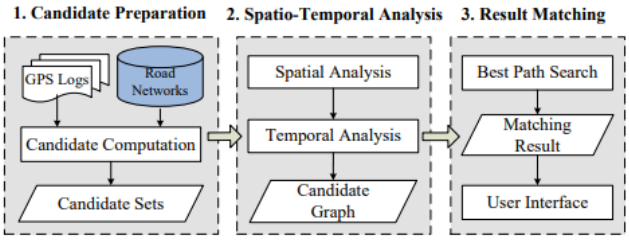



*Figure 26 Overview of system architecture*

Map-Matching on Low Sampling Rate Trajectories through Frequent Pattern Mining[3]

This study is divided into two phases: the offline phase and the online phase. In the offline phase, FP-forest is utilized to process historical data. Additionally, dynamic programming is used to select candidates for the map matching process. In the online phase, the map matching process is applied to GPS data.

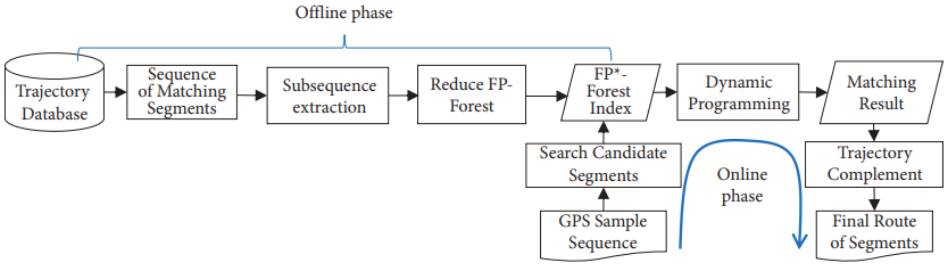



*Figure 27 Overview of our solution.*

## Compare related works

Numerous research studies have focused on improving the accuracy and performance of map matching methods, using various techniques such as traditional map matching, machine learning, and deep learning models. To compare these related works, we have created a table, which is presented below.

*Table 1 Table compare related works*

| Title | Authors | Year | Source | Motivate | Proposed method/ Technology | Evaluation | Contributor |
|---|---|---|---|---|---|---|---|
| a map-matching algorithm based on distributed random forest multi-classification | GuangLin Zhou and Feng Chen | 2018 | 2018 IOP Conference Series Earth and Environmental Science | Improve accuracy and speed in map matching method for traffic analysis , reduce error in vehicle positioning and the Intelligent Transportation System (ITS). | Use The spark platform(distributed computing platform) to preparing feature to train randomforest model ,prediction and training model. The dataset for training model contains feature such as distance from point to road id and label is road id that seperated by grid to improve performance | The result was verify by using model to predict test dataset that split from dataset and not include in dataset for training model and check accuracy from test dataset | This method can speed to matching point in the road and apply to online map matching .Moreover,This method can imporve accuracy compared with the traditional point-line matching algorithm |
| Map Matching Integration Algorithm Based on Historical Trajectory Data | Xin LAI, Jianhua CHEN ,Jingjing CAO and Fei XIA | 2019 | 2019 IEEE Symposium on Product Compliance Engineering - Asia (ISPCE-CN) | Improve the accuracy of GPS trajectory data by using historical trajectory data | The first proposed algorithm integrates the projection distance method and Hidden Markov Model( HMM ) . The second approch is clusters the historical trajectory data by using DBSCAN clustered , calculated similarity and select road path | Calcuarate the accracy of result by compare the result and the historical trajectory data. | The algorithm proposed in this paper adjust the driving position of the vehicle and suitable for complex urban road networks |
| A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories | Wentao Bian , Ge Cui and Xin Wang | 2020 | 2020 MDPLA Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories. | Develop map matching method for low-sampling-rate GPS trajectories and improve accuracy. | Use clsutering group of GPS trajectories by simirality and resampling the missing data for increas the frequency of data and use The Longest Common Subsequence (LCSS) for map matching process | Investigation precision, recall and running time of the process and compare result and real data | This method can improve accuracy and performance of map matching for low-sampling-rate GPS data and uncertainty GPS data. |
| A Machine Learning Approach to Improve the Accuracy of GPS-Based Map Matching Algo | Mahdi Hashemi, Hassan A. Karimi | 2016 | 2016 IEEE 17th International Conference on Information Reuse and Integration | Reduce error of GPS horizontal location and improve map-matching method | Use artificial neural network (ANN) to reduce horizontal error of GPS data | Calcuarate the accracy of result by compare the result that predict from model and map-matching point | This study improve map-matching method and accuracy of horizantal location in GPS data |
| Map-Matching on Low Sampling Rate Trajectories through Frequent Pattern Mining | Lei Yu , Zhiqiang Zhang, and Rongtao Ding | 2022 | 2022 Hindawi Scientific Programming | Improve map matching method in Low sampling data | Study dynamic programming for reduce historycal data | By calculate accuracy by number and length , Minimum Fr´echet Distance and Route Mismatch Fraction | This method can improve map-matching method in low-sampling-rate GPS data so it can save cost in collect data process and processing data process |

Table compare related work in traditional map matching method as below:

*Table 2 Compare related works in traditional map matching method.*

| Title | Authors | Year | Source | Motivate | Proposed method/ Technology | Evaluation | Contributor | Data |
|---|---|---|---|---|---|---|---|---|
| Hidden Markov map matching through noise and sparseness | P. Newson and J. Krumm | 2009 | Proc. 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA | Reduce error in GPS location and improve accuracy of map matching method | Use Hidden Markov Model (HMM) to map matching method and improve accuracy of map matching method. | Compare error from result that adjust by map matching method and GPS data that collect in real world. | Improve accuracy of map matching method and reduce error in GPS location. | GPS data point in the Seattle, Washingt on and USA area. |
| A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories | Wentao Bian , Ge Cui and Xin Wang | 2020 | 2020 MDPI.A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories. | Develop map matching method for low-sampling-rate GPS trajectories and improve accuracy. | Use clustering group of GPS trajectories by simirality and resampling the missing data for increas the frequency of data and use The Longest Common Subsequence (LCSS) for map matching process | Investigation precision, recall and running time of the process and compare result and real data | This method can improve accuracy and performance of map matching for low-sampling-rate GPS data and uncertainty GPS data. | The real taxi GPS Trajectory dataset in Shenzhen, China. The format |
| Enhanced Map-Matching Algorithm with a Hidden Markov Model for Mobile Phone Positioning | An Luo ,Shenghua Chen and Bin Xv | 2017 | ISPRS Int. J. Geo-Inf. 2017 | Use map matching method for adjust GPS location that low sample rate | Hidden Markov model is map matching method that study use for improve map matching method | Errror between point that adjust from map matching method and real location | Improve map matching method for adjust GPS low frequncy data and improve accuracy of data. | the GPS data and mobile phone data in the Do-iT project spatial road network data |
| Shortest path and vehicle trajectory aided map-matching for low frequency GPS data | Mohammed Quddus , Si mon Washington | 2015 | Transp. Res. Part C Emerg. Technol | Improve accuracy and preformance of map matching method by shotest path | The shortest path that use A+ search algorithm for map matching method and use stMM algorithm for adjust GPS point. | Calculate accuracy of GPS location that adjust by algorithm and location in real world. | Improve map matching method by shotest path algorithm | real-time Intelligent Transport System (ITS) |
| A Map-matching Algorithm with Extraction of Multi-group Information for Low-frequency Data | Jie Fang, Xiongwei Wu, Dianchao Lin, Mengyun Xu, Huahua Wu, Xuesong Wu, Ting Bi | 2022 | IEEE Intelligent Transportation Systems Magazine | Reduce error in GPS location point and improve map matching method | Use Markov neutral network for map matching method and a modified top-K shortest-path method to search the candidate paths. | Compare error from result that adjust from map matching method and baselline data. | Increase accracy in GPS data in urban area and improve map matching method data. | Road network in the urban area of Zhangzhou, China. GPS Data point |
| An Enhanced Hidden Markov Map Matching Model for Floating Car Data | Mingliang Che, Yingli Wang,* Chi Zhang, and Xinliang Cao | 2018 | Inf. Sci | Improve Map matching method for adjust location data | Enhanced hidden Markov map matching (EHMM) is algorithm that research study for map matching method. | Calculate different between adjust point by map matching method and groud truth location | This method can reduce error in GPS location and improve map matching method | The trajectory data Road network |
| Map Matching Integration Algorithm Based on Historical Trajectory Data | Xin LAI, Jianhua CHEN ,Jingjing CAO and Fei XIA | 2019 | 2019 IEEE Symposium on Product Compliance Engineering - Asia (ISPCE-CN) | Improve the accuracy of GPS trajectory data by using historical trajectory data | The first proposed algorithm integrates the projection distance method and Hidden Markov Model( HMM ) . The second approch is clusters the historical trajectory data by using DBSCAN clustered , calculated similarity and select road path | Calcuarate the accracy of result by compare the result and the historical trajectory data. | The algorithm proposed in this paper adjust the driving position of the vehicle and suitable for complex urban road networks | 1.The data of truck GPS trajectory data. (records of 50 trucks in 7 days) 2 Open street map ( road network) |
| Map-Matching on Low Sampling Rate Trajectories through Frequent Pattern Mining | Lei Yu , Zhiqiang Zhang, and Rongtao Ding | 2022 | 2022 Hindawi Scientific Programming | Improve map matching method in Low sampling data | Study dynamic programming for reduce historycal data | By calculate accuracy by number and length , Minimum Fr´echet Distance and Route Mismatch Fraction | This method can improve map-matching method in low-sampling-rate GPS data so it can save cost in collect data process and processing data process | low-sampling-rate GPS data |

Table compare related work that study Machine learning and Deep learning for improve map matching method as below:

*Table 3 Compare related work that study Machine learning and Deep learning for improve map matching method*

| Title | Authors | Year | Source | Motivate | Proposed method/ Technology | Evaluation | Contributor | Data |
|---|---|---|---|---|---|---|---|---|
| A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories | Wentao Bian , Ge Cui and Xin Wang | 2020 | 2020 MDPI.A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories. | Develop map matching method for low-sampling-rate GPS trajectories and improve accuracy. | Use clustering group of GPS trajectories by simirality and resampling the missing data for increas the frequency of data and use The Longest Common Subsequence (LCSS) for map matching process | Investigation precision, recall and running time of the process and compare result and real data | This method can improve accuracy and performance of map matching for low-sampling-rate GPS data and uncertainty GPS data. | The real taxi GPS Trajectory dataset in Shenzhen, China. The format |
| Map Matching Integration Algorithm Based on Historical Trajectory Data | Xin LAI, Jianhua CHEN ,Jingjing CAO and Fei XIA | 2019 | 2019 IEEE Symposium on Product Compliance Engineering - Asia (ISPCE-CN) | Improve the accuracy of GPS trajectory data by using historical trajectory data | The first proposed algorithm integrates the projection distance method and Hidden Markov Model( HMM ) . The second approch is clusters the historical trajectory data by using DBSCAN clustered , calculated similarity and select road path | Calcuarate the accracy of result by compare the result and the historical trajectory data. | The algorithm proposed in this paper adjust the driving position of the vehicle and suitable for complex urban road networks | 1.The data of truck GPS trajectory data. (records of 50 trucks in 7 days) 2 Open street map ( road network) |
| a map-matching algorithm based on distributed random forest multi-classification | GuangLin Zhou and Feng Chen | 2018 | 2018 IOP Conference Series Earth and Environmental Science | Improve accuracy and speed in map matching method for traffic analysis , reduce error in vehicle positioning and the Intelligent Transportation System (ITS). | Use The spark platform(distributed computing platform) to preparing feature to train randomforest model ,prediction and training model. The dataset for training model contains feature such as distance from point to road id and label is road id that seperated by grid to improve | The result was verify by using model to predict test dataset that split from dataset and not include in dataset for training model and check accuracy from test dataset | This method can speed to matching point in the road and apply to online map matching .Moreover,This method can imporve accuracy compared with the traditional point-line matching algorithm | 1 the full-day taxi driving data of Hefei City in March,7th,2017. 2 Open street map ( road network) |
| A Machine Learning Approach to Improve the Accuracy of GPS-Based Map Matching Algo | Mahdi Hashemi, Hassan A. Karimi | 2016 | 2016 IEEE 17th International Conference on Information Reuse and Integration | Reduce error of GPS horizontal location and improve map-matching method | Use artificial neural network (ANN) to reduce horizontal error of GPS data | Calcuarate the accracy of result by compare the result that predict from model and map-matching point | This study improve map-matching method and accuracy of horizantal location in GPS data | Raw GPS point and Map-matched points for training Model |

## Discussion and summary of related works

Traditional map matching methods typically require a large amount of data for processing and take a significant amount of time to adjust GPS data. As a result, these methods are not well-suited for low-frequency GPS data and real-time applications. However, machine learning technology offers a promising alternative by leveraging historical data patterns and utilizing attributes to adjust GPS locations. This approach can significantly enhance the speed and performance of map matching methods.

In this study, a machine learning model is employed, which incorporates advanced features such as previous point location, bearing, and speed to improve the map matching process for low-frequency GPS data. The research focuses on utilizing the journey points of GPS and the relationship between previous and current points as key factors in adjusting GPS points.

## Methods

This research uses machine learning techniques in the classification method to reduce the density of Grab data from every 4 seconds to 1 minute, to simulate low-frequency data and to predict low-frequency data (iTic). The research aims to compare the accuracy of classification models such as Random Forest classification, Decision tree, Multi-layer Perceptron Classifier (MLPClassifier), XGBoost, K-Nearest Neighbors (KNN), and Long Short-Term Memory (LSTM) for classifying points in road ID and adjusting raw GPS data in the study. The features used for the classification method in this study are listed below:

Rawlng: raw data of longitude

Rawlat: raw data of latitude

Rawlat_lag1: raw data latitude of previous point 1 point

Rawlng_lag1: raw data longitude of previous point 1 point

Rawlat_lag2: raw data latitude of previous point 2 point

Rawlng_lag2: raw data longitude of previous point 2 point

Rawlat_lag3: raw data latitude of previous point 3 point

Rawlng_lag3: raw data longitude of previous point 3 point

Rawlat_next1: raw data latitude of next point 1 point

Rawlng_next1: raw data longitude of next point 1 point

Rawlat_next2: raw data latitude of next point 2 point

Rawlng_next2: raw data longitude of next point 2 point

Rawlat_next3: raw data latitude of next point 3 point

Rawlng_next3: raw data longitude of next point 3 point

Bearing: Direction of vehicle

Speed: Speed of vehicle

Month: Month record of point

Day: Weekday record of point

Hr: Hour record of point

Diff_time: different time between the previous point and current point

Diff_speed: difference speed between the previous point and current point

Acceleration: different speed between the previous point and current point divided by different time between the previous point and current point.

*Table 4 List of features*

| **Table List of Feature** | |
|---|---|
| ***Feature*** | ***Description*** |
| Raw location | raw data of longitude and latitude |
| Raw location previous1-3 point | raw data latitude and longitude of previous point 1-3 point |
| Raw location next 1-3 point | raw data latitude and longitude of next point 1-3 point |
| Bearing | Direction of vehicle |
| Speed | Speed of vehicle |
| Month,Day and Hour of record | Month record of point, Weekday record of point and Hour record of point |
| Diff_time | Different time between the previous point and current point |
| Acceleration | Different speed between the previous point and current point divided by different time between the previous point and the current point. |

Score of feature importance shows feature impacts on the classification model by feature important technic.

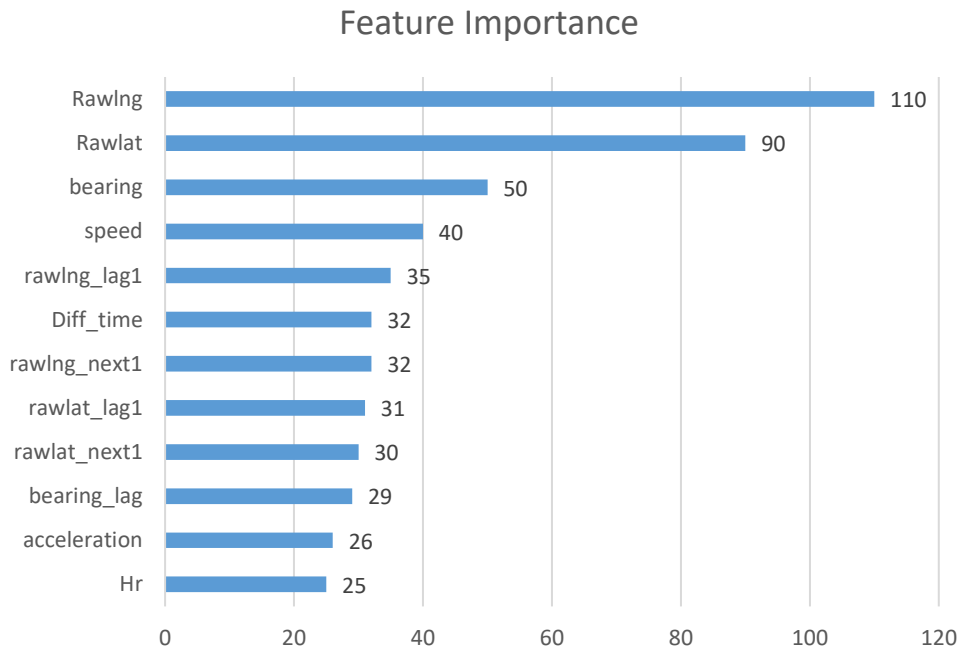



*Figure 28 Feature important*

The overview workflow for creating machine learning models for a map matching method is as below:

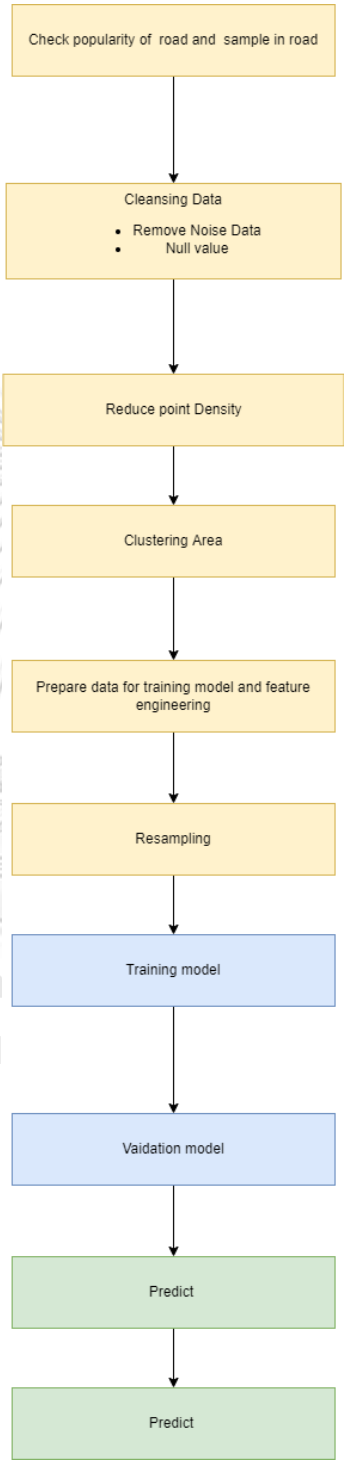



*Figure 29 Flow chart* of machine learning model

In the workflow mentioned above, the initial step for developing a machine learning model is to verify whether the number of samples in each road ID is adequate for training the model. If the number of sample points is insufficient, it can decrease the accuracy of the model and lead to class imbalance issues. Therefore, this step checks the popularity of the road and selects only the main and popular roads.

In the next step of the workflow, the data cleansing process is performed to improve the quality of data and remove any outlier data. This process involves using a machine learning technique called DBSCAN (Density-Based Spatial Clustering) to detect outlier GPS points in the spatial data. The DBSCAN algorithm is an unsupervised learning technique that clusters data points based on their density in the feature space. In this research, the raw latitude and longitude data of GPS points are used to cluster data points into groups of outliers and non-outliers, as shown in the example provided.

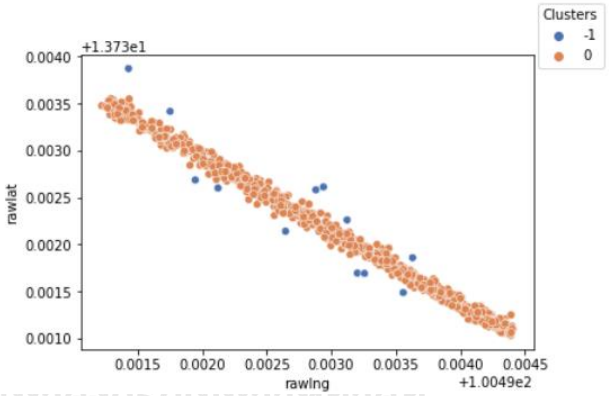



*Figure 30 DBSCAN Clustering*

After applying the DBSCAN clustering algorithm, the GPS points are grouped into two clusters: Cluster 0, which represents normal GPS points, and Cluster -1, which represents outlier data. The points in Cluster -1 can be further analyzed to determine whether they are noise or not. If a point is determined to be noise, it can be removed from the dataset to improve the accuracy of the model.

The second method to detect outlier is to detect outlier by grouping the points by waypointid and driverid and counting the number of samples after grouping. In case the number of samples after grouping by waypoint and driverid is very low, this point will be checked and explored for validation and removal in case this point is noise. The step and workflow to check the outlier point will be as below:

1. Group GPS points by waypointid and driverid.
2. Count the number of samples in each group.
3. Check if the number of samples is lower than a threshold value.
4. If the number of samples is lower than the threshold value, explore and validate the points in the group.
5. Remove the points identified as noise in step 4.
6. Repeat steps 1-5 for all groups of GPS points.

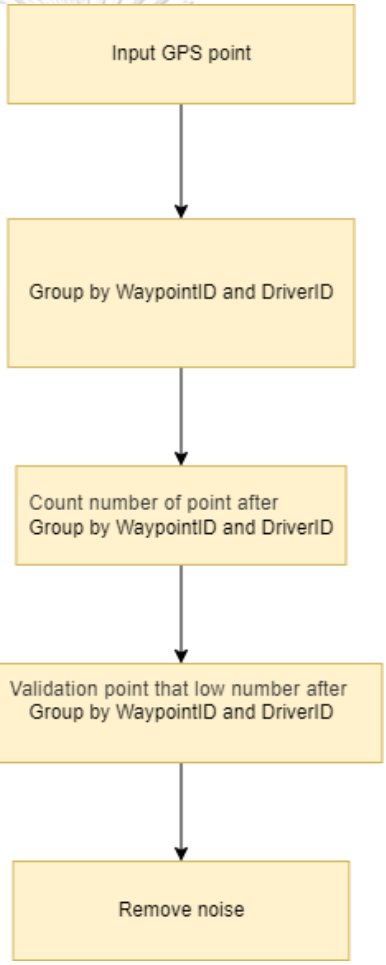



*Figure 31 Work flow for check outlier point*

| | driverid | wayid | count |
|---|---|---|---|
| 0 | 00022b685bd41c250ced3ba6d803ce9e2e4988197f110b... | 10266636 | 2 |
| 1 | 0010381eefb36a40a5b4726e17f315655db1f9af1cb531... | 39943967 | 3 |
| 2 | 0010381eefb36a40a5b4726e17f315655db1f9af1cb531... | 154774001 | 4 |
| 3 | 00112912bc9836971f5c30f8d163143915628e1b3f763c... | 154774001 | 1 |
| 4 | 00112912bc9836971f5c30f8d163143915628e1b3f763c... | 462260865 | 1 |
| ... | ... | ... | ... |
| 17622 | ffdccfe869a534ece4df78e50bd448d21f931dec31623d... | 206846596 | 3 |
| 17623 | ffdccfe869a534ece4df78e50bd448d21f931dec31623d... | 221227406 | 3 |
| 17624 | ffede656ccb9a85a44904794f8246925b5c85ce651fb87... | 453884722 | 1 |
| 17625 | fff26a3df5e3b3ad02ee06416d0c45481afa771b476984... | 322197502 | 1 |
| 17626 | fff9b94d92f3ffbaea7417ed76184dc01b4ea9c1d9dd1d... | 632450279 | 1 |

*Figure 32 Sample count GPS point group by driverid and wayid*


*Figure 33 Sample noise GPS point*



If the number of samples after grouping by waypointid and driverid is not very low or high, this point is checked by a Python script that loops through and detects changes in wayid. The workflow for this is shown below:

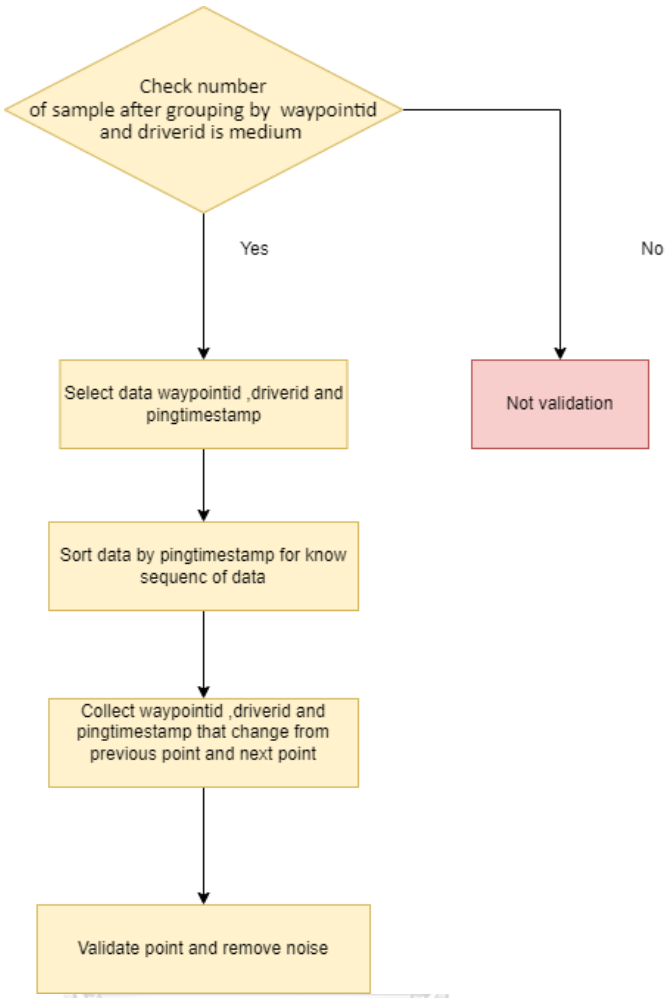



*Figure 34 Work flow for check outlier point for medium points*

Detect change Point and suspect outlier point

| projectedlat | projectedlng | rawlat | rawlng | segmentstartnode | segmentendnode | wayid | mapid | adjust_lat | adjust_lng | predict_class |
|---|---|---|---|---|---|---|---|---|---|---|
| 13.720992 | 100.526919 | 13.720846 | 100.526967 | 5956131210 | 1883602210 | 177961597 | BKK_4W | 13.720992 | 100.526919 | 177961597 |
| 13.721089 | 100.527238 | 13.720914 | 100.527295 | 5956131210 | 1883602210 | 177961597 | BKK_4W | 13.721089 | 100.527238 | 177961597 |
| 13.721151 | 100.527446 | 13.720984 | 100.527500 | 5956131210 | 1883602210 | 177961597 | BKK_4W | 13.721151 | 100.527446 | 177961597 |
| 13.721207 | 100.527628 | 13.721044 | 100.527681 | 5956131210 | 1883602210 | 177961597 | BKK_4W | 13.721207 | 100.527628 | 177961597 |
| 13.728296 | 100.572800 | 13.728190 | 100.572721 | 272453727 | 281732227 | 459784945 | BKK_4W | 13.728296 | 100.572800 | 459784945 |
| 13.728274 | 100.572831 | 13.728165 | 100.572750 | 272453727 | 281732227 | 459784945 | BKK_4W | 13.728274 | 100.572831 | 459784945 |
| 13.728274 | 100.572831 | 13.728157 | 100.572764 | 272453727 | 281732227 | 459784945 | BKK_4W | 13.728274 | 100.572831 | 459784945 |
| 13.728274 | 100.572831 | 13.728160 | 100.572763 | 272453727 | 281732227 | 459784945 | BKK_4W | 13.728274 | 100.572831 | 459784945 |
| 13.729294 | 100.571260 | 13.729268 | 100.571240 | 901886735 | 272206252 | 710591527 | BKK_4W | 13.729294 | 100.571260 | 710591527 |
| 13.729427 | 100.571074 | 13.729394 | 100.571049 | 272206252 | 1867150494 | 452179357 | BKK_4W | 13.729427 | 100.571074 | 452179357 |
| 13.729569 | 100.570874 | 13.729539 | 100.570852 | 272206252 | 1867150494 | 452179357 | BKK_4W | 13.729569 | 100.570874 | 452179357 |
| 13.729735 | 100.570640 | 13.729697 | 100.570612 | 1867150494 | 4484624701 | 452179357 | BKK_4W | 13.729735 | 100.570640 | 452179357 |
| 13.735118 | 100.562972 | 13.735116 | 100.562970 | 4484651863 | 272487195 | 258200369 | BKK_4W | 13.735118 | 100.562972 | 258200369 |
| 13.735319 | 100.562684 | 13.735269 | 100.562647 | 272487195 | 4460800322 | 258200369 | BKK_4W | 13.735319 | 100.562684 | 258200369 |
| 13.735413 | 100.562548 | 13.735375 | 100.562520 | 272487195 | 4460800322 | 258200369 | BKK_4W | 13.735413 | 100.562548 | 258200369 |
| 13.735522 | 100.562391 | 13.735492 | 100.562369 | 272487195 | 4460800322 | 258200369 | BKK_4W | 13.735522 | 100.562391 | 258200369 |

*Figure 35 Sample GPS point grouped by driverid and wayid and sorted with pinftimestamp*


*Figure 36 Sample noise GPS point*

Figure 36 shows the sample GPS points that are grouped by driver ID and way ID, and then sorted by pingtimestamp. This figure also illustrates how the Python scripts detect the change points.

During the cleansing step, one approach is to remove the nearest points based on the distance between points and the difference in time. This method helps identify and eliminate points that are likely to be duplicates.

The step to reduce point density involves converting high-frequency GPS data from Grab to low-frequency data that simulates iTic data. This is done to change the distribution of Grab data to that of iTic data, which enables the map matching model to predict low sample data. Reducing the number of samples also saves time and cost that would be required to collect additional data.



The step to cluster the area involves dividing the study area into smaller segments to increase processing speed and improve model accuracy. This is achieved by clustering the area using K-means clustering, which calculates the sum of the length of roads in each district divided by the district area and the average count of sampling per length of road in each district. The result of clustering by K-means is shown in the figure.

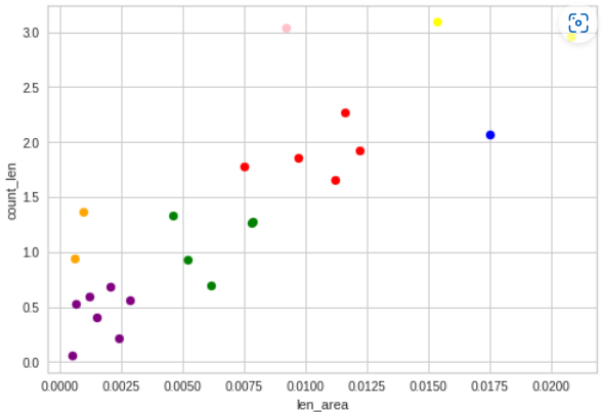



*Figure 37 Clustering Area*

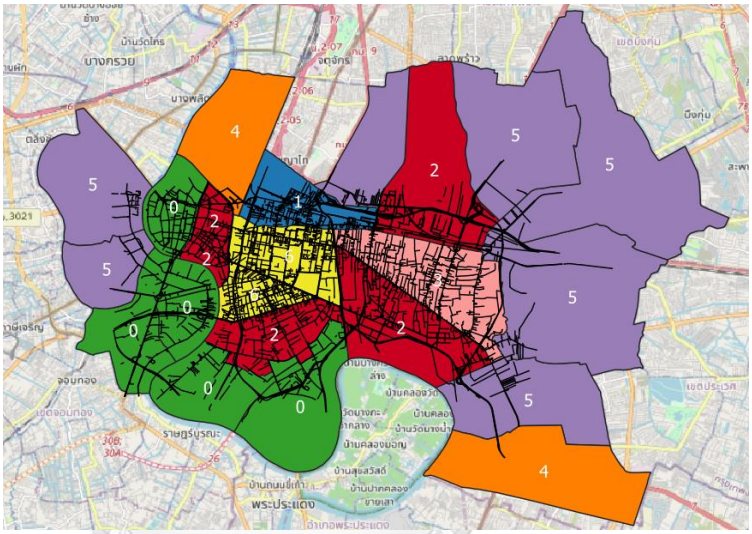



*Figure 38 Area clustering in Map*

From Figure 37 and Figure 38, 7 groups of area are as below:

- Group0 is the area with medium sample and medium road density such as Khlong San, Thon Buri and Bang Kho Laem.
- Group1 is the area with high density of road and quite high sample of data but lower than Group6 and Group3 such as Rat Thewi.
- Group2 is the area with quite high density of road and quite high sample of data such as Huai Khwang, Pom Prap Sattruphai, Samphanthawong.
- Group3 is the area with very high sample of data and medium density of road such as Watthana
- Group4 is the area with very low density of road and low sample of data but higher than Group5 such as Dusit and Bang Na

- Group5 is the area with very low density of road and very low sample of data such as Bangkok Noi and Bangkok Yai
- Group6 is the area with very high density of road and very high sample of data such as Pathum Wan, Bang Rak

After defining the clustered groups of areas, we removed the ones with low data samples from the study area (specifically, groups 5 and 4) since they were not suitable for training the map matching model. To increase the sample size of another area, we buffered it by 1 km and created a separate machine learning model for each area during the training step. Additionally, we developed a model for the overlapping area to improve overall accuracy.

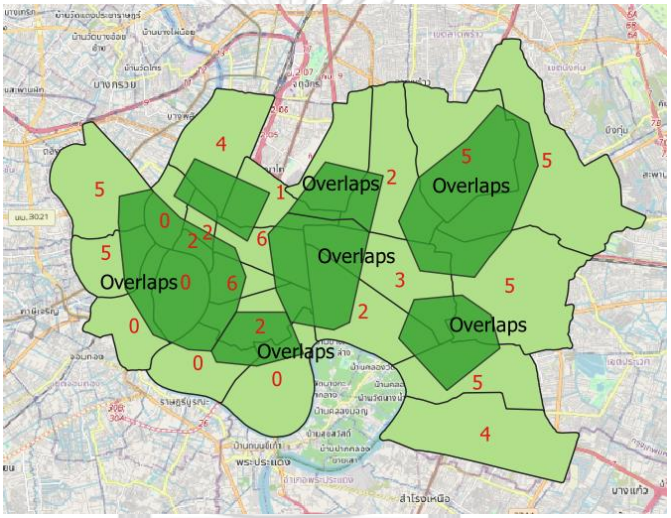



*Figure 39 Overlap area and buffer*



The next step involves preparing the data for the training model by creating a Python function to transform and calculate features such as the time difference between the previous and current point, and the ID of the previous point. Once the features are generated, they will be evaluated for their importance and quality for the training model. Any features that are deemed unqualified or unimportant will be removed from the feature list to ensure the best quality training model.

Resampling data is a crucial step in this study because imbalanced classes can pose a significant problem during training, validation, and prediction of the classification method. To address this issue and improve model accuracy, we utilized SMOTE, an over-sampling method designed to address the problem of imbalanced classes.

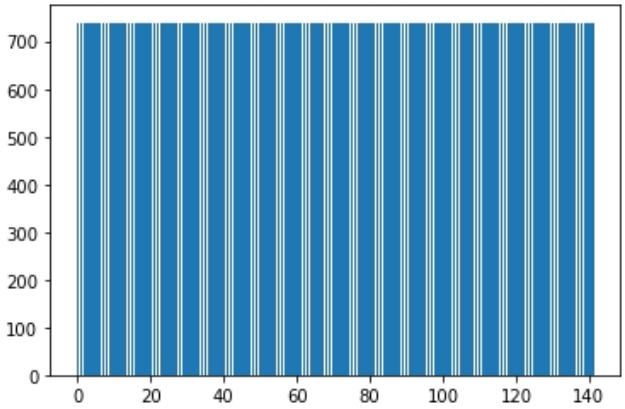



*Figure 40 SMOTE*

The training model step is a crucial component of the study, involving several processes to create an accurate machine learning model, such as grid search, fine-tuning parameters, and selecting appropriate machine learning algorithms. In this study, we utilized classification techniques to predict road ID and adjust GPS points on the road as predicted by the machine learning model.

The validation step involves evaluating the model's performance using a test dataset that was not included in the training phase of the machine learning step. Its purpose is to verify that the model is accurate enough to be used in the map matching process.

The predict step is performed after the training model and validation phase, and its purpose is to adjust the GPS location of a new dataset using the model generated in the previous steps. To accomplish this, the new dataset is inputted into the machine learning model, which generates a prediction of the class or road ID.



| pingtimestamp | speed | bearing | rawlat | rawlng | predict |
|---|---|---|---|---|---|
| 1514764754 | 38 | 307 | 13.71175 | 100.59596 | 156261400 |
| 1514764762 | 0 | 290 | 13.73129 | 100.53217 | 675969812 |
| 1514764760 | 24 | 305 | 13.73594 | 100.56158 | 258200369 |
| 1514764761 | 13 | 256 | 13.74429 | 100.54272 | 156985388 |
| 1514764781 | 26 | 129 | 13.72337 | 100.57993 | 452182940 |
| ... | ... | ... | ... | ... | ... |
| 1514851101 | 37 | 15 | 13.75633 | 100.56501 | 25744948 |
| 1514851140 | 67 | 257 | 13.71173 | 100.56720 | 210884217 |
| 1514851100 | 43 | 11 | 13.75641 | 100.56498 | 25744948 |
| 1514851142 | 42 | 287 | 13.74204 | 100.59655 | 178872781 |

*Figure 41 Sample output prediction*

The final step is to adjust the raw latitude and longitude of the GPS points in the road ID that was predicted by the machine learning model. This adjustment is made based on the predicted road ID and is done to ensure the accuracy of the GPS location.

{'Lat': 100.59581074347099, 'Lon': 13.711886348176401}
{'Lat': 100.531737452521, 'Lon': 13.731482857048302}
{'Lat': 100.56174489830799, 'Lon': 13.7359745686715}
{'Lat': 100.542722858373, 'Lon': 13.744219593659402}
{'Lat': 100.57993011818, 'Lon': 13.723349480194399}
{'Lat': 100.590475432695, 'Lon': 13.7018822501657}
{'Lat': 100.59068416084499, 'Lon': 13.7155245177423}
{'Lat': 100.591756139235, 'Lon': 13.714934360132599}
{'Lat': 100.558754543687, 'Lon': 13.720235422027098}
{'Lat': 100.555844617235, 'Lon': 13.7401957228544}
{'Lat': 100.536840710331, 'Lon': 13.730262153275799}
{'Lat': 100.545158528244, 'Lon': 13.725855496833802}
{'Lat': 100.552967046592, 'Lon': 13.7423703294272}
{'Lat': 100.545664383405, 'Lon': 13.7435693010824}
{'Lat': 100.536583301878, 'Lon': 13.729687747291699}
{'Lat': 100.536799757903, 'Lon': 13.7301689187606}
{'Lat': 100.552523598086, 'Lon': 13.7423636645515}

*Figure 42 Sample adjust GPS location*

Workflow: How to use machine learning adjust point

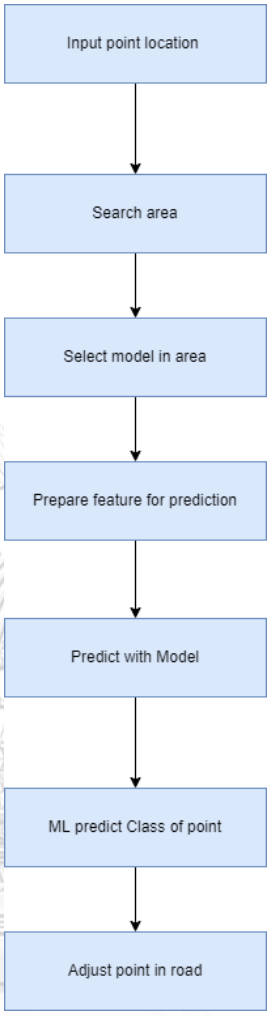



*Figure 43 How to use ML adjust location point*



When a location point that requires adjustment is inputted, the system will select the corresponding area that contains the point. It will then select the appropriate machine learning model for that area to predict the road ID of the point and adjust its location on the corresponding road.

## Algorithms

This study utilizes geospatial technologies to prepare and explore data, including techniques such as reducing point density based on data frequency. The data preparation phase also includes the use of over-sampling techniques such as SMOTE to address class imbalance issues. Machine learning technology is then used to create a classification model and adjust raw GPS data on the corresponding road ID.

## Tool

The training of machine learning models, data cleansing, feature engineering, and data exploration in this research were conducted using Python version 3.8. The code was run on a Google Colab notebook, which provides a cloud-based Python development environment with GPU support. The study utilized various Python libraries, including:

Data Preparation:

- Pandas
- NumPy
- SMOTE
- Scikit-learn
- StandardScaler
- Principal Component Analysis (PCA)

Visualization and Exploratory Data Analysis (EDA):

- Matplotlib
- Seaborn
- Plotly
- Graphviz

Machine Learning Model Training:

- TensorFlow
- Keras
- LSTM (Long Short-Term Memory)
- XGBoost
- MLPClassifier (Multi-Layer Perceptron Classifier)
- KNeighborsClassifier

- DecisionTreeClassifier
- RandomForestClassifier
- Scikit-learn

Model Evaluation:

- Confusion Matrix
- Cross-validation
- Scikit-learn's evaluation metrics (accuracy_score,precision_score, recall_score and f1_score)

The study also utilized QGIS that is a free and open-source geographic information system (GIS) software that allows users to visualize, manage, and analyze geospatial data. It is commonly used for tasks such as creating and editing maps, geocoding addresses, and performing spatial analysis. QGIS supports a wide range of file formats and provides a variety of tools for geospatial analysis and data processing to perform spatial functions on the prepared dataset for training the model and exploring the dataset.

## Dataset

The training dataset used in this study consists of GPS location data from Grab's vehicles in June 2019, collected every 4 seconds. To ensure that the dataset is suitable for training the machine learning model, the density of the dataset was checked to avoid class imbalance, and only popular roads with a significant number of samples were selected for use in traffic analytics. For prediction, the dataset used is vehicle location data from the Intelligent Traffic Information Center (iTIC), collected every minute. Grab data was chosen for training the model due to its similarity to the iTIC data and the fact that GPS points were adjusted using a map matching technique, which is suitable for creating a machine learning model. The Grab data includes various information, such as raw latitude and longitude, vehicle speed, and direction, as well as a label for the road ID, which was adjusted using the traditional map matching method and Markov chain model. Therefore, the label for road ID in the Grab data after adjustment is highly accurate and reliable, making it suitable for representing the map matching method for this study.

This study aims to use historical raw GPS location data from Grab's vehicles in Bangkok, Thailand, which were collected every 4 seconds, to train a machine learning model for map matching. The data was preprocessed by reducing point density to avoid imbalance class issues

and ensure the appropriate number of data for the training model. Only popular roads or those with a high number of samples were used for classification to enable efficient traffic analytics. For prediction purposes, low-frequency GPS data from an Intelligent Traffic Information Center (iTIC) collected every minute was used. The map matching technique was applied using OpenStreetMap to adjust the GPS location of the vehicles to the actual road network.

The Grab dataset includes various information such as raw latitude and longitude before being adjusted, vehicle speed, vehicle direction, and road ID labels that are adjusted using traditional map matching methods and Markov chain models. The adjusted road ID label in the Grab dataset is highly accurate and reliable due to the map matching method used. Additionally, this map matching method and dataset are utilized in the Grab application, making it suitable to represent the map matching method for real-world application purposes.

The area of this study is the road that is popular and high sample dataset in Bangkok Thailand as below:

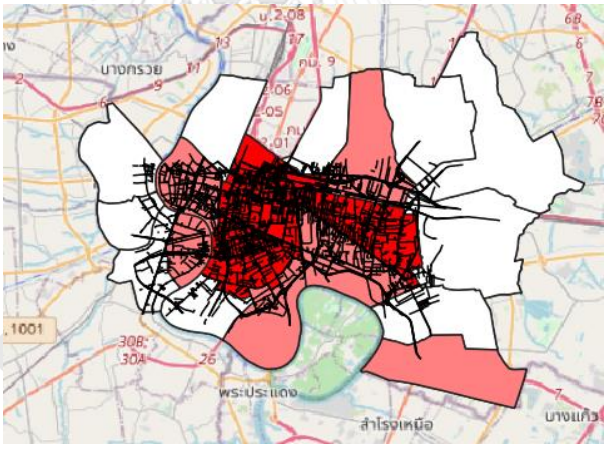



*Figure 44 Heatmap*

The Dataset in study area contain error and accuracy as table below:

*Table 5 Error in GPS Data*

| Cluster | Average of Error Mean(m) | Average of Error Max(m) | Average of Error Min(m) |
|---|---|---|---|
| 0 | 8.1474478 | 117.48 | 0.94 |
| 1 | 9.369096 | 127.449 | 1 |
| 2 | 9.125858 | 126.2364 | 1.02 |
| 3 | 10.421158 | 128 | 0.9 |
| 4 | 7.024712 | 51 | 1 |
| 5 | 8.416711333 | 88.40633333 | 1.374166667 |
| 6 | 10.769797 | 128 | 0.9 |

The study reports a mean average error of 9 m across all areas, with a maximum error of 128 m. It was observed that areas with high data samples and medium density of roads, such as Watthana, and areas with high density of roads and data samples, such as Pathum Wan and Bang Rak, had relatively high errors due to the high density of roads, tall buildings, and the presence of the BTS Skytrain. Consequently, GPS performance was hindered in such areas. In contrast, cluster 5 comprised areas with low road density and data samples, such as Bangkok Noi and Bangkok Yai, resulting in relatively low errors compared to other areas.

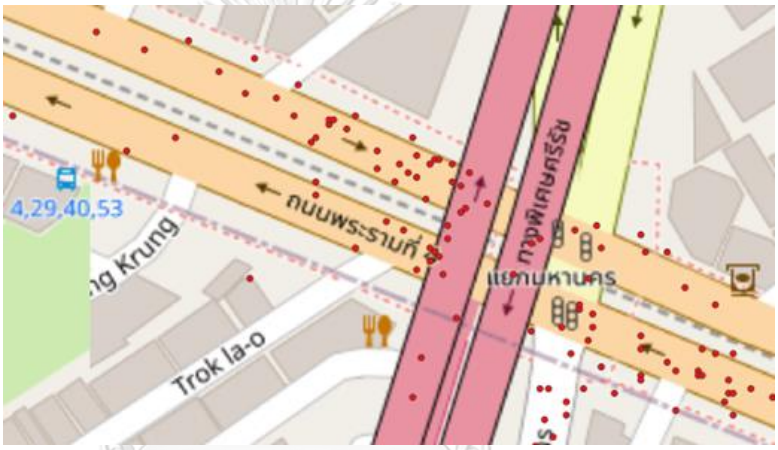



*Figure 45 Sample GPS Point Data*

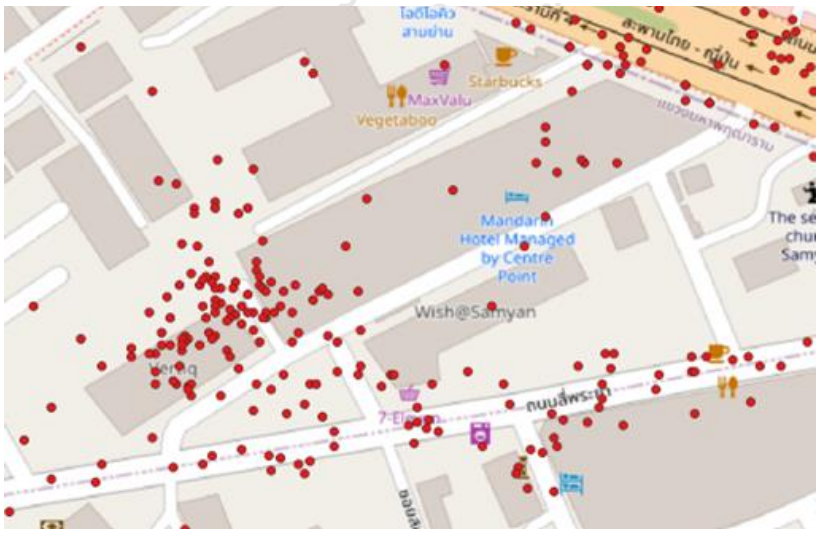



*Figure 46 Sample GPS Data in Urban area*

The GPS data from the sample picture indicates that the accuracy of GPS points is better in areas with fewer high-rise buildings and less complex road networks. Conversely, the accuracy of GPS data in areas with high-rise buildings is lower.

## Evaluation

To further improve the accuracy and reliability of the machine learning model, it is important to split the dataset into a training set and a validation set. The validation set should not be used in the training process and should only be used to evaluate the performance of the trained model.

Once the training process is complete, the validation dataset will be used to predict the road IDs using the trained model. The predicted labels will then be compared with the actual labels of the validation dataset using a Confusion Matrix. The Confusion Matrix will allow us to calculate metrics such as accuracy, precision, recall, and F1-score, which will help in selecting the best model. The study employed the F1-score as an evaluation metric for the selected machine learning model and its parameters.

By splitting the dataset into a training set and a validation set, we can ensure that the model is not overfitting to the training data and can generalize well to new data. This will result in a more accurate and reliable model that can be used for traffic analytics and other related applications.

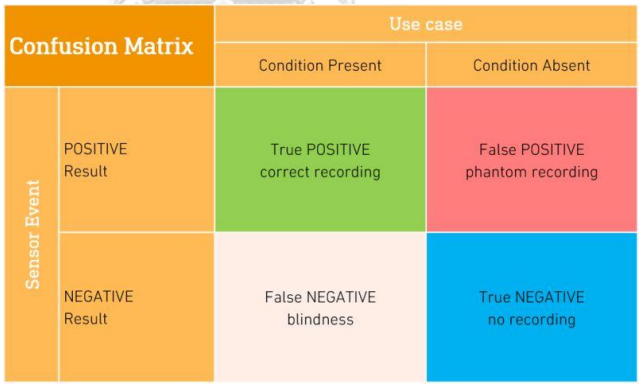



*Figure 47 Confusion Metrix*

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

*Figure 48 Equation of Accuracy, Precision ,Recall and F1-Score*

The evaluation of model performance will use a matrix of processing transformation data time to prediction time. The model should be able to return prediction output for not less than 100 points in 1 second and not less than 6000 points in 1 hour. This is because, based on historical iTic data, the maximum number of points in 1 second is 93 points, and in 1 hour is 5612 points. Therefore, this study proposes handling the maximum case of iTic data to ensure the model's effectiveness and efficiency.

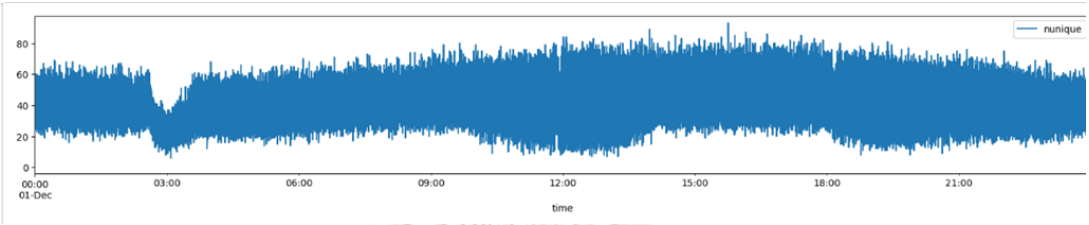



*Figure 49 Trend of iTic data by Hour*

| time | nunique |
|---|---|
| 2019-03-07/16 | 1 |
| 2019-03-20/12 | 1 |
| 2019-07-25/13 | 1 |
| 2019-09-08/3 | 1 |
| 2019-10-14/14 | 1 |
| ... | ... |
| 2019-12-01/5 | 5566 |
| 2019-12-01/6 | 5572 |
| 2019-12-01/7 | 5586 |
| 2019-12-01/8 | 5585 |
| 2019-12-01/9 | 5591 |

```
max(df2['nunique'])
```

```
5612
```

*Figure 50 Result count number of car in iTic Data by Hour*

## Result

This research project aims to compare the performance of different classification models in map matching methods. The models to be used include Random Forest Classification, Decision Tree, Multi-layer Perceptron Classifier (MLPClassifier), XGBoost, K-Nearest Neighbors (KNN), and Long Short-Term Memory (LSTM).

The study consists of three experiments, which are as follows:

- Comparison of accuracy and time model Random Forest classification, Decision Tree, Multi-layer Perceptron Classifier (MLPClassifier), XGBoost, K-Nearest Neighbors (KNN) and Long Short-Term Memory (LSTM) by using the training model of low-frequency data( Grab GPS data that reduces density)
- Comparison of model accuracy and time Random Forest Classification, Decision Tree and Long Short-Term Memory (LSTM) by using the training model of low-frequency data( Grab GPS data that reduces density) and by using the training model of variant frequency data
- Comparison of map matching results from the machine learning model with the python library for map matching methods

### Case1 Comparison of accuracy and time model Random Forest classification, Decision Tree, Multi-layer Perceptron Classifier (MLPClassifier), XGBoost, K-Nearest Neighbors (KNN) and Long Short-Term Memory (LSTM) by using the training model of low-frequency data( Grab GPS data that reduces density)

In this study, various machine learning models will be compared, including:

· Random Forest classification use all feature

· Random Forest classification use all feature exclude raw location

· Decision Tree use all feature

· Decision Tree use all feature exclude raw location

· Multi-layer Perceptron Classifier (MLPClassifier) use all feature

· XGBoost use all feature

· XGBoost use all feature exclude raw location

· K-Nearest Neighbors (KNN) use all feature

· K-Nearest Neighbors (KNN) use all feature exclude raw location

· Long Short-Term Memory (LSTM)  use all feature

To compare the speed and accuracy of the machine learning models trained and tested on low-frequency data (Grab GPS data with reduced density) in all areas, the study evaluated the models based on their accuracy, precision, recall, and F1 score. The results of the comparison are presented below:

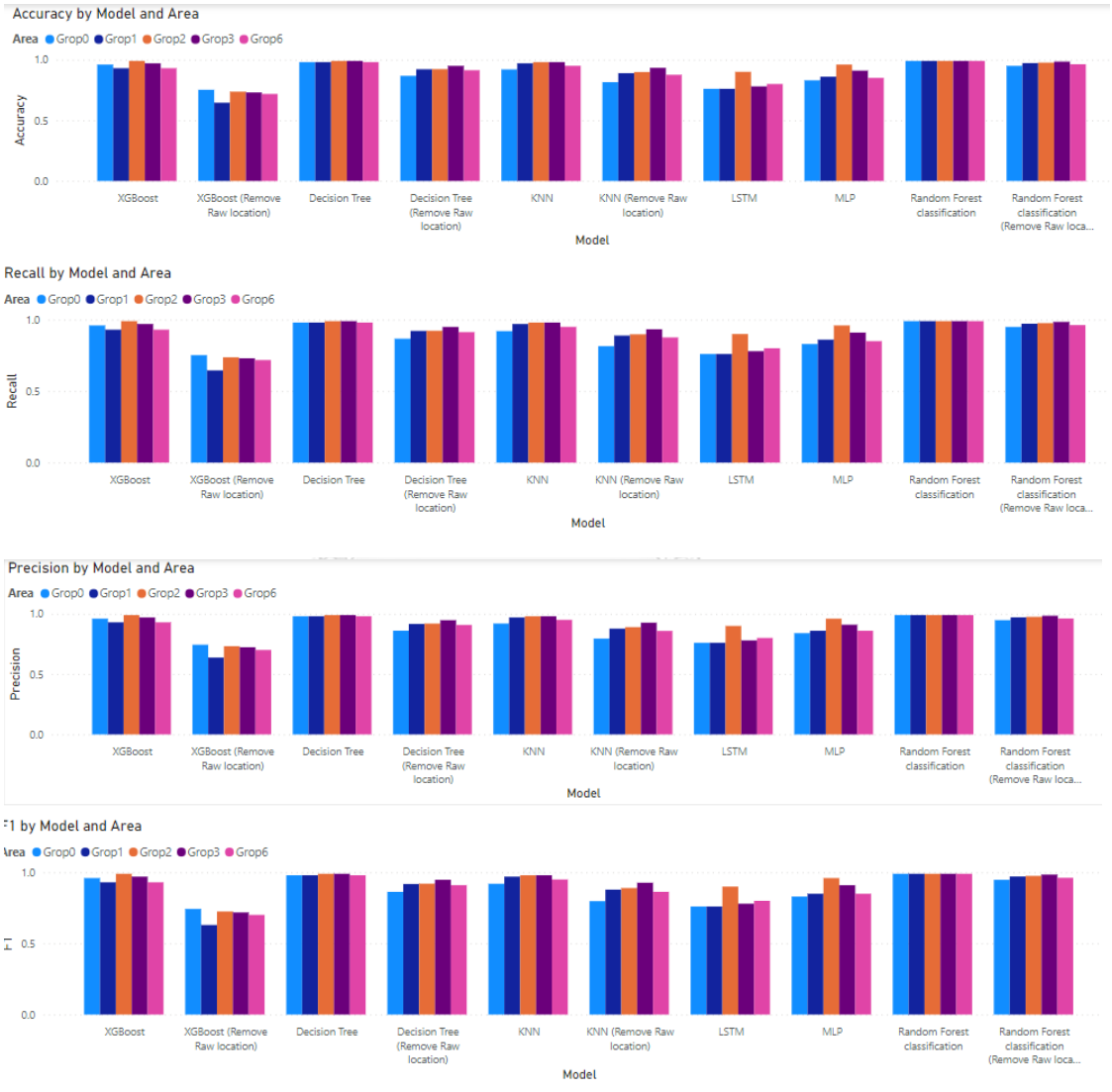



*Figure  51 Comparison of accuracy, precision, recall and f1 score in machine learning model with low frequency data in all areas*

To compare the speed and accuracy of different machine learning models using training and test datasets with low-frequency data (Grab GPS data with reduced density) in all areas, the results for accuracy, precision, recall, and F1 score are as follows:

The XGBoost model and the XGBoost model without raw latitude and longitude in the feature exhibit quite similar trends in all values, but the latter has slightly lower values. The Decision Tree model and the Decision Tree model without raw latitude and longitude in the feature do not show significant differences in values. The KNN model and the KNN model without raw latitude and longitude in the feature show low accuracy, precision, recall, and F1 score in all areas, but Area 3 (an area with high sample size) has the highest value in the KNN model. The LSTM and MLP models perform well in Area 2 (an area with medium road density and sample size), but other areas have low accuracy, precision, recall, and F1 score. The Random Forest model and the Random Forest model without raw latitude and longitude in the feature exhibit insignificant differences in all values. Moreover, both Random Forest models (with and without raw latitude and longitude in the feature) show high values. Area 2 (an area with medium road density and sample size) contains high values in all models, while Area 0 (an area with low sample size data and low road density) contains low values in all models except the XGBoost model. Area 1 (an area with high road density and medium sample size) contains the lowest value in the XGBoost model.

After testing the processing time of each model, it was found that Decision Tree had the fastest processing time, followed by XGBoost and Random Forest classification. MLP had a slightly slower processing time compared to the previous models, while KNN and LSTM had the slowest processing time. However, it is worth noting that the processing time may vary depending on the size of the dataset and the complexity of the model.

In conclusion, the Random Forest model, which uses all features, achieved the highest F1-Score. However, the XGBoost model, which excludes the raw location feature, achieved the lowest F1-Score. The Decision Tree model performed slightly worse than the Random Forest but had faster prediction and training times.

*Table 6 Comparison of average f1-score in machine learning model*

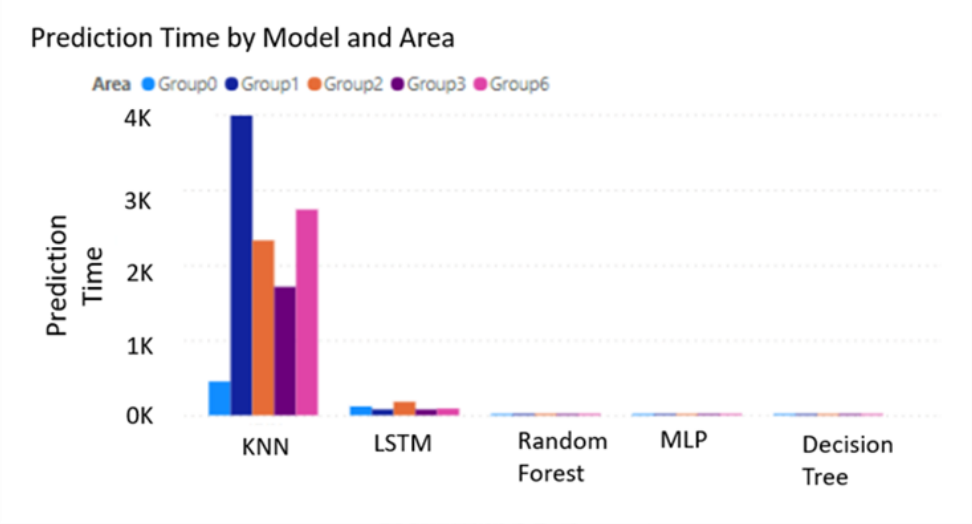
| Model | Use all Feature | Remove raw location |
|---|---|---|
| XGBoost | 0.96 | 0.70 |
| Decision Tree | 0.97 | 0.91 |
| KNN | 0.96 | 0.87 |
| LSTM | 0.80 | |
| MLP | 0.88 | |
| Random Forest | 0.98 | 0.97 |




*Figure 52 Comparison of speed in processing time*

The LSTM model has the longest processing time in all areas, with Area 1 (high road density and medium sample size) taking the longest time for training in all areas, except for the LSTM model, where Area 6 (high road density and sample size) has the longest processing time. On the other hand, KNN has the shortest processing time in all areas. Decision Tree, XGBoost, and MLP do not take much time in processing in all areas. Area 2 (medium road density and sample size) has the shortest processing time in almost every model.

In conclusion, the Random Forest model performs well in all areas, but when compared in terms of processing time, Decision Tree has the lowest processing time compared to Random Forest and has high values in accuracy, precision, recall, and F1-score.




| Model | Group0 | Group1 | Group2 | Group6 | Group4 |
|---|---|---|---|---|---|
| KNN | 450.00 | 3988.00 | 2327.00 | 1710.00 | 2738.00 |
| LSTM | 119.00 | 80.00 | 180.00 | 79.00 | 90.00 |
| RF | 8.00 | 19.40 | 19.70 | 10.00 | 19.20 |
| MLP | 0.29 | 1.15 | 1.34 | 0.85 | 0.95 |
| DC | 0.06 | 0.14 | 0.13 | 0.07 | 0.12 |

*Figure 53* Comparison of *prediction time(sec)*

The KNN model has the longest prediction time, while the Decision Tree model has the shortest time, taking less than 1 second in all cases. Area0 requires relatively less processing time when dealing with 10,000 records. To apply machine learning models to iTic data, it is important to ensure that the prediction time does not exceed 100 seconds for a dataset with 10,000 points. However, models such as KNN (K-Nearest Neighbors) and LSTM (Long Short-Term Memory) may exceed this time limit due to their computational requirements or the complexity of their training and prediction processes.

Compare case study1 machine learning model in F1-Score,precision ,accuracy ,Training processing time and prediction time

XGBoost model

- The feature 'raw location' is important for the XGBoost model because the accuracy, precision, recall, and F1-Score drop significantly when this feature is removed in all areas.
- 'Raw location' is a highly impactful feature for XGBoost because it appears at the top of the feature importance list.
- XGBoost models that exclude 'raw location' feature contain lower values in all areas because 'raw location' is a top feature in the importance list.
- The trend of accuracy, precision, recall, and F1-Score is similar in both XGBoost models that use all features and XGBoost models that exclude 'raw location'.
- Area0 has the lowest accuracy in the XGBoost model when compared to other areas because it has low sample size data and low road density.
- Area2 performs well in the XGBoost model because it has medium road density and sample size.

Decision tree model



- The Decision Tree model that uses all features and the one that removes raw location as a feature do not show significant differences in accuracy, precision, recall, and F1-Score in all areas.
- In Area0 and some other areas, the performance of the decision tree model slightly decreases when the raw location feature is removed from the model. This might be due to the low frequency of data in these areas. Therefore, some features such as the previous point may have a negative impact on the model's performance when the data is sparse.
- The Decision Tree model requires a relatively short time for training in all areas.

- Decision tree is the model that takes the least time for prediction in all areas because it is a simple machine learning model.

Random Forest model

- Removing raw location from Random Forest model does not significantly affect its accuracy, precision, recall, and F1-score in all areas. This is because Random Forest is an ensemble learning model that creates multiple decision trees for prediction, so it can handle different types of data and features well.
- It appears that using all features in the Random Forest model results in high accuracy in all areas without significant differences in values among each area. This may be due to the fact that Random Forest is an ensemble learning model that creates multiple decision trees for prediction, which can handle data and various features well. Additionally, including all features may help capture more information and patterns in the data, leading to higher accuracy in the model.
- Random Forest is a more complex model compared to Decision Tree, as it involves building multiple decision trees and combining their predictions. Therefore, it may take longer to train a Random Forest model compared to a Decision Tree model. However, Random Forest is still faster than more complex models like LSTM, which involves recurrent neural networks and sequential processing.
- Decision tree models typically have faster prediction times compared to random forest models, since random forest models require multiple decision trees to be created and evaluated for each prediction. However, random forest models may have higher accuracy due to the ensemble approach, where the combination of multiple decision trees can reduce the impact of individual decision trees' weaknesses.

K-Nearest Neighbors (KNN)

- The KNN model performs similarly in terms of accuracy whether all features are used or the raw location feature is excluded.

- KNN is a distance-based algorithm and works by calculating the distance between the data points. Therefore, it requires a sufficient amount of data to identify the nearest neighbors accurately. As a result, Area3 with high sample size is likely to perform well in the KNN model. However, it's also important to keep in mind that KNN is sensitive to the choice of the number of neighbors (k) and the distance metric used. So, these factors should also be carefully considered when applying the KNN algorithm.
- KNN model with all features and KNN model without raw location feature have similar performance in terms of accuracy, precision, recall, and F1-score. However, in area3, where the sample size is high, KNN performs well as it can handle high-density data by calculating the similarity and prediction based on the nearest neighbors. In this area, KNN with both feature sets performs similarly as the model can effectively use all features to predict the outcomes.
- Area0 has the lowest accuracy in the KNN model, and the raw location feature has a significant impact in this area. This is because KNN does not perform well in low-density areas, and low-sample areas can reduce the quality of the features and contain a low number of nearest neighbor points for prediction.
- KNN is known as a lazy learning algorithm because it doesn't learn a model during the training phase, but rather stores the entire training dataset in memory. Therefore, the training time for KNN is generally very fast compared to other algorithms.
- KNN is a lazy learning algorithm, meaning that it defers the processing of training data until a new query is encountered, it is not always the lowest mode in prediction time.

  The prediction time for KNN depends on the number of neighbors used for prediction, and the number of data points in the training set. As the size of the training set increases, KNN's prediction time can become significantly slower, since it has to calculate the distances between the query point and all points in the training set.

  In contrast, some other machine learning models, such as decision trees and logistic regression, have fixed prediction times regardless of the size of the training set, since their prediction only involves traversing a fixed tree or calculating a linear function. So,

in summary, KNN can be the lowest mode in prediction time for small training sets or when using a small number of neighbors, but it can become significantly slower than other models for larger training sets or larger numbers of neighbors.

LSTM

- LSTM models are known to be effective in handling time-series data and sequence data, which is why they might have performed better in Area2, where the data may have had a more balanced distribution and higher frequency. On the other hand, in areas with lower frequency data, the performance of LSTM models may not be significantly different from other models such as decision trees or random forests.
- LSTM is a type of deep learning model that is known for its ability to process sequential data. It has a complex architecture with several layers of recurrent neural networks, which requires a significant amount of computation during training. As a result, it can take much longer to train than other types of machine learning models.

MLP

- The MLP model has the lowest accuracy in Area0, which can be attributed to the low data density in this area. MLP is a neural network model that requires a high density of data to create an accurate model.
- The MLP is neural network model that require a lot of data to perform well. In areas with high sample size, such as Area2, the MLP model can learn patterns in the data better and make more accurate predictions. Additionally, since Area2 does not have high road density, the MLP model is less likely to be affected by noise in the data caused by road intersections and other similar factors.

Compare case study in term of area

- Area 0 is the region with the lowest values in almost all of the models, likely due to the limited amount of sample data available in this area.
- Area 2 contains the highest value in all models. This is because this area has a high number of sample points and a low density of roads.

- Area 1 takes a longer time for training the model in almost all models because this area has a high density of sample points.
- The area that contains the lowest training time is Area 0 because this area has a low number of samples.
- Area 1 takes a longer time for prediction processing compared to other areas due to the high density of sample points.
- Area 0 has the lowest training time in almost all models due to its low sample size.

Case2 compares accuracy and time model Random Forest classification, Decision Tree, K-Nearest Neighbors (KNN) and Long Short-Term Memory (LSTM) by using the training model of low-frequency data (Grab GPS data that reduces density) and by using the training model of variant frequency data in 30 classes.

After processing the data, the model trained on high-frequency data is not necessarily faster than the model trained on low-frequency data, but the accuracy, precision, recall, and F1 score tend to be higher on high-frequency data. This is because high-frequency data usually contains more information and details, allowing the model to better capture the patterns and relationships within the data, resulting in higher accuracy. However, the trade-off is that high-frequency data can be more complex and require more resources and time for processing and training.



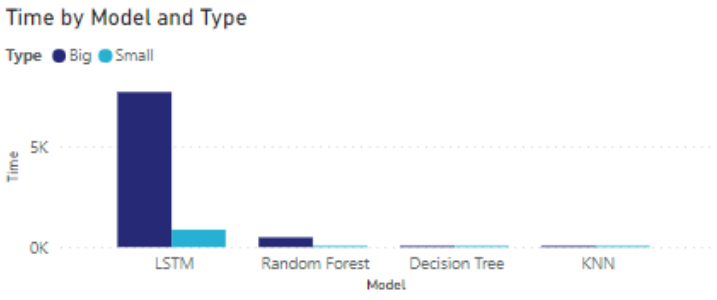



*Figure 54 Comparison of time to process with low and high frequency data*

Figure 54 presents a comparison of the time taken to process low and high-frequency data. It can be observed that for LSTM and Random Forest models, the low-frequency data takes

less time to train than the high-frequency data. However, for KNN and Decision Tree models, there is not a significant difference in the training time between low and high-frequency data.

Regarding the accuracy, precision, recall, and f1 score comparison between machine learning models trained with low and high-frequency data, the results indicate that the models trained with high-frequency data outperform the ones trained with low-frequency data in terms of all these evaluation metrics. This suggests that high-frequency data can lead to better-performing machine learning models.

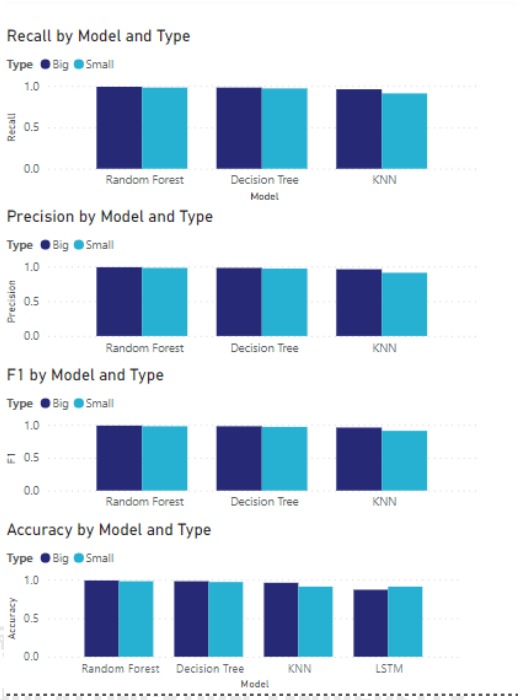





*Figure 55 Comparison of accuracy, precision, recall and f1 score to process with low and high frequency data*

Figure 55 compares the accuracy, precision, recall, and F1 score between models trained on low and high-frequency data. The majority of the models trained on high-frequency data perform better in terms of accuracy, precision, recall, and F1 score than those trained on low-frequency data.

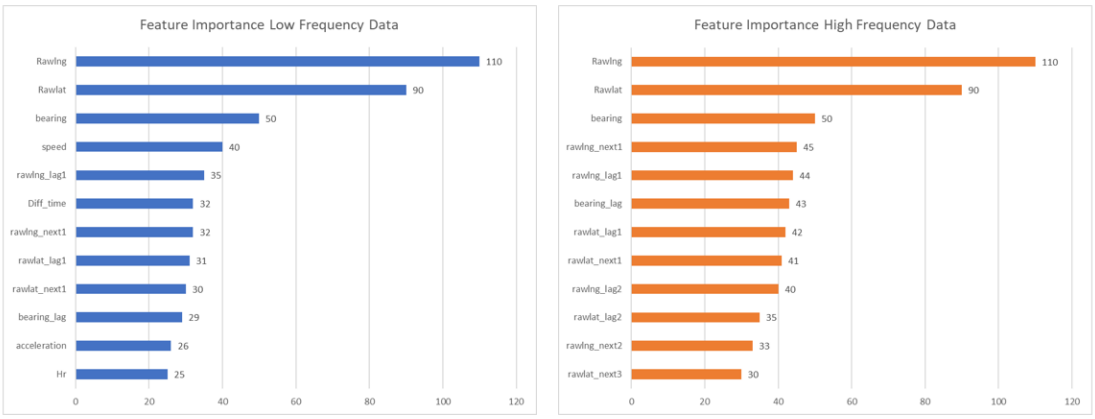



*Figure 56 Comparison of feature importance between model from low and high frequency data*

After training with machine learning models using high-frequency data, the importance of location-related features, such as previous and next points, has slightly increased compared to low-frequency data. Furthermore, features related to other points, such as the score of feature importance, have increased, as high-frequency data can better capture changes and relationships between points compared to low-frequency data.


*Figure 57 Comparison of accuracy, F1-Score, Precision and recall in data 0-2400 sec*

From Figure 57, when comparing the accuracy, F1-Score, Precision, and Recall in data ranging from 0-2400 sec, the Random Forest and Decision Tree models that use all features show a slight drop in accuracy, F1-Score, Precision, and Recall when the frequency of data is reduced. However, the XGBoost model that uses all features remains quite stable when the frequency of data is changed. On the other hand, the KNN model that uses all features shows high sensitivity

but low accuracy, F1-Score, Precision, and Recall compared to other models that use all features for prediction and training. In data with a frequency of 2400 sec, the KNN model drops rapidly in all values. The Random Forest model that uses all features except raw latitude and longitude shows the highest value when compared to other models that use all features except raw latitude and longitude. The Decision Tree that uses all features except raw latitude and longitude has a high value, but it is lower than the Random Forest that uses all features except raw latitude and longitude. The KNN model with all features and the model that uses all features except raw latitude and longitude are not significantly different in value in data with a frequency of 1200-2800 sec, but before 1200 sec, the KNN model that uses all features is slightly higher when compared to the KNN model that uses all features except raw latitude and longitude. The XGBoost model that uses all features except raw latitude and longitude slightly increases in all values in the range of 0-2000 sec and all values drop in the range of 2000-2800 sec.

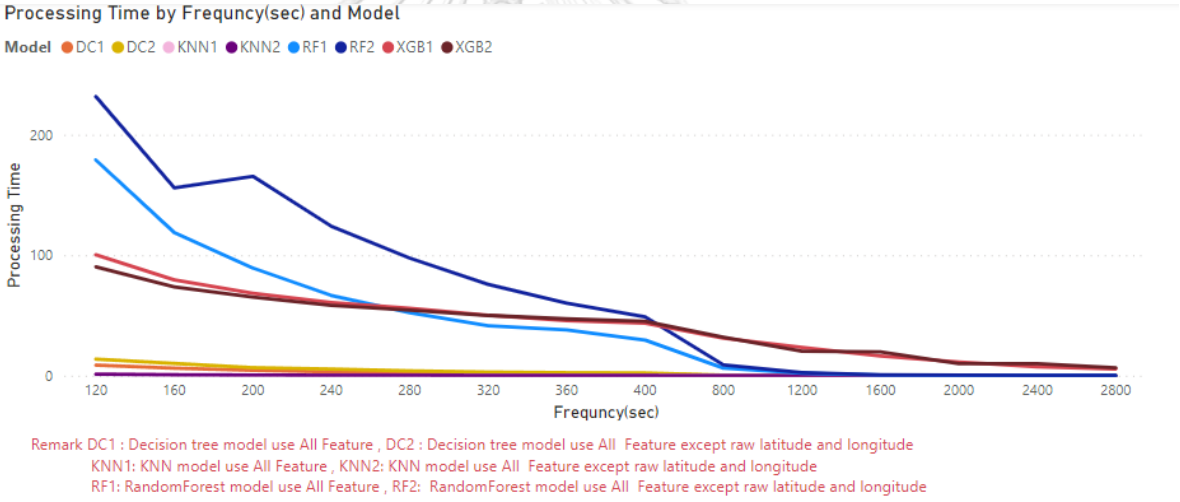



*Figure 58 Comparison Time to process in data 0-2400 sec*

From Figure 58, we can see that the processing time of XGBoost, Decision Tree, and KNN models that use all features and those that use all features except raw latitude and longitude do not show a significant decrease when the density of data is reduced. However, the processing time of the Random Forest model that uses all features and the one that uses all features except raw latitude and longitude decreases rapidly in data frequency 0-800. In data frequency 1200-2800 sec, the processing time becomes more stable for the Random Forest model that uses all features and the one that uses all features except raw latitude and longitude.

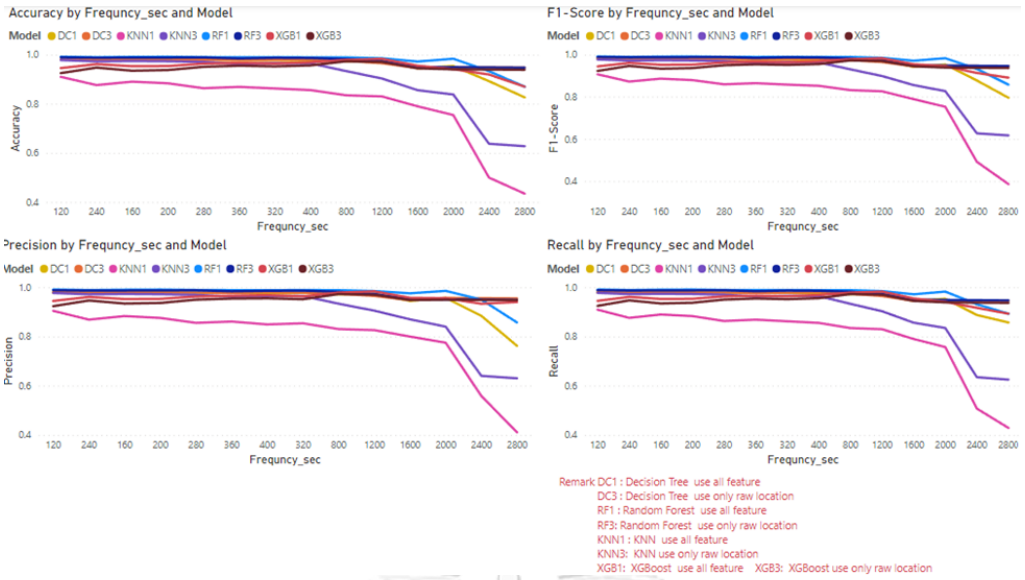



*Figure 59 Comparison of accuracy, F1-Score, Precision and recall in data 0-2800 sec with variant features*

*Table 7 Comparison of Average F1-Score in data 0-2800 sec with variant features*

| Model | Use all Feature (Data frequncy 0-1900 sec) | Use only Raw location and bearing (Data frequncy 0-1900 sec) | Use all Feature (Data frequncy > 1900 sec) | Use only Raw location and bearing (Data frequncy > 1900 sec) |
|---|---|---|---|---|
| XGBoost | 0.96 | 0.68 | 0.91 | 0.93 |
| Decision Tree | 0.97 | 0.83 | 0.87 | 0.90 |
| KNN | 0.72 | 0.65 | 0.48 | 0.51 |
| Random Forest | 0.98 | 0.91 | 0.92 | 0.95 |



In the range of 0-1900 sec, Decision Tree, Random Forest, and XGBoost models that use all features have slightly higher values than those that only use raw location and bearing data. However, in the range of 2000-2800 sec, the models that use only raw location and bearing data have higher values than the models that use all features. Meanwhile, KNN models that use only raw location and bearing data have higher values than those that use all features across all frequencies.

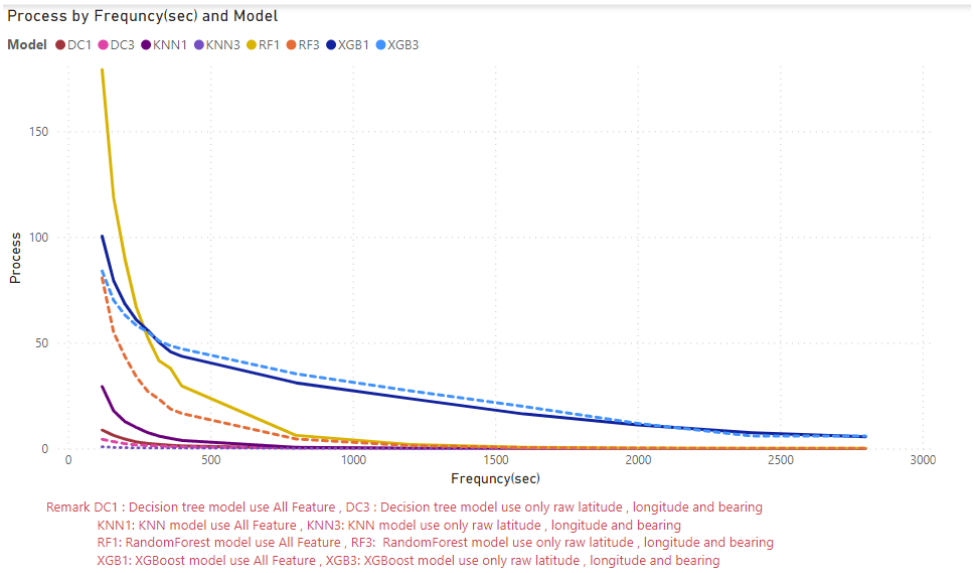



*Figure 60 Comparison Time to process in data 0-2800 sec with variant features*

The KNN and Decision Tree models that use all features and only raw location and bearing have low processing time and their values are not significantly different. In contrast, the Random Forest model with all features rapidly drops in data frequency between 120-400 sec and then gradually drops again after 800 sec. However, the Random Forest model that only uses raw location and bearing does not drop rapidly like the Random Forest model that uses all features, but it still drops slightly after 800 sec.

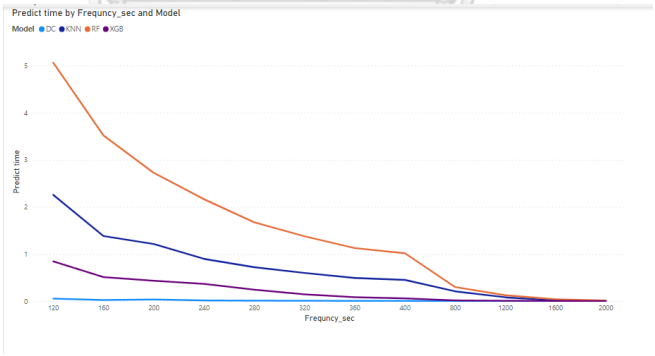



*Figure 61 Compare prediction time with multi frequency*

Random Forest is the model that takes the longest time for prediction, while Decision Tree has the lowest prediction time. In all models, the prediction time decreases when the frequency of the data is reduced.

Compare case study2 machine learning model in F1-Score,precision ,accuracy ,Training processing time and prediction time

Decision Tree

- The Decision Tree model that uses all features, including raw location, has higher accuracy, precision, recall, and f1 score than the Decision Tree model that excludes raw location in all frequency of data, because raw location has a high score in feature importance.
- Accuracy and other performance metrics drop significantly when the data frequency is lower than 2000 sec. This is because the data contains very low frequency, which makes it difficult for the model to capture enough samples and relationships. In particular, models that rely on time-sensitive features, such as changes in location or velocity, are affected the most. With higher frequency data, the model can capture more detailed information and better capture the dynamics of the system. Therefore, increasing the data frequency can improve the performance of the model in such cases.
- The Decision tree model that uses all features is better than the Decision tree model that uses only raw location and bearing in data frequency from 0 to 2000 seconds because some features can improve the machine learning model in certain cases. For example, including the previous or next point location can improve the model's accuracy in terms of route and environment.
- The Decision tree model that uses only raw location and bearing can sometimes outperform the model that uses all features, especially when the data frequency is lower than 2000 sec. This is because some features, such as previous point and next point location, may not be reliable or accurate enough at low frequencies, which can lead to a decrease in model performance. In such cases, relying only on the raw location and bearing information can still provide useful information for the model and result in better accuracy.
- Decision tree is model that low sensitivity.
- Decision tree is a relatively simple model and has a fast training time compared to other models like Random Forest or XGBoost. This is because the decision tree algorithm only

needs to make a series of binary decisions based on the values of the input features to classify the data. However, the processing time for prediction can still be affected by the size of the data and the number of features used. When the number of features or the size of the data is large, the decision tree may become more complex and require more computation time to make predictions. Additionally, if the data contains missing values or categorical variables, preprocessing steps may be needed, which can also increase processing time.

- The Decision tree is considered to be the fastest model in terms of prediction time among the models evaluated, regardless of the frequency of the data. This is due to the fact that decision trees are simple models that do not require a lot of computational resources to make predictions. Therefore, if speed is an important factor in the analysis, decision trees can be a good option. However, it is important to keep in mind that decision trees may not always provide the highest accuracy compared to more complex models.

Random Forest

- Random Forest models that utilize all features have higher accuracy and better performance in terms of all metrics compared to models that exclude raw location features in all data frequencies. This is likely due to the importance of raw location data in accurately predicting the outcome variable.
- The accuracy and all value metrics of machine learning models drop significantly when the data frequency is lower than 2000 sec. This is because a very low amount of data can lead to lower quality features and a higher bias in the model. With fewer data points, the model has a limited understanding of the patterns and relationships in the data, leading to lower accuracy and overall performance. Therefore, it is important to have a sufficient amount of data to build a reliable model that can capture the dynamics of the system.
- Random Forest that uses all features achieves the best accuracy and value in all frequency of data, as it is an ensemble machine learning model that can handle overfitting problems and improve the quality of the model.

- Random Forest that uses all features performs better than Random Forest that only uses raw location and bearing in data frequency 0-2000 sec. This is because some features, such as previous point or next point location, can improve the machine learning model in terms of capturing the route and environmental information. Using all features can provide the model with more information and improve its performance.
- When the data frequency is lower than 2000 sec, Random Forest models that only use raw location and bearing have higher accuracy than those that use all features. This is because the low frequency of the data can reduce the quality of some features, such as previous or next point location, which are important for improving the model's performance.
- Random Forest is a model with low sensitivity because it can handle high variance and overfitting problems by utilizing multiple decision trees and ensemble algorithms.
- The training time of Random Forest models is influenced by the frequency of the data. High frequency data requires more computational power for the Random Forest to train, as it is a more complex model compared to other models like Decision Trees.
- A Random Forest model that excludes raw location may take longer time for training processing, as this model would have to rely on other features to capture the patterns and relationships in the data, which may require more computational resources and time. However, it ultimately depends on the specific dataset and the complexity of the features included in the model. It's important to note that Random Forest is generally known for its relatively fast training time compared to other complex models, such as deep neural networks.
- Random Forest takes more time for prediction compared to simpler models like Decision Trees. This is because Random Forest is an ensemble model that combines multiple decision trees, which can increase the complexity of the model and the time needed to make predictions.

  Moreover, the prediction time in Random Forest may increase with high-frequency data because the model needs to process more data points, which can lead to longer

processing times. However, the impact of data frequency on prediction time may vary depending on other factors such as the number of trees in the forest, the complexity of the model, and the hardware used for training and prediction.

KNN

- The KNN model is highly sensitive to the frequency and density of data points in the input dataset. Since KNN relies on finding the nearest neighbors of a given point, the density of the data and the distance between the data points can greatly impact the accuracy and performance of the model. In areas where there are few data points or where the data is sparse, KNN may struggle to accurately predict the output. On the other hand, in areas where there are many data points, KNN may perform well, as there are more neighbors to consider for each point. Therefore, the sensitivity of the KNN model is highly dependent on the characteristics of the input data.
- KNN tends to have the lowest accuracy and all value in all frequency of data because the model is not well-suited for low-density data and complex areas, and its performance heavily depends on the frequency and density of data due to its reliance on nearest neighbor points for calculation.
- KNN that uses only raw location and bearing generally has higher accuracy and all values than KNN that uses all features, regardless of the frequency of the data, because density of data is an important factor for the KNN model. When the frequency of data drops, it can affect the quality of features and reduce the model's performance. Therefore, using only the essential features such as location and bearing can improve the accuracy and all values of the KNN model.
- KNN is a relatively fast learning model compared to some other machine learning algorithms like neural networks, random forests, or support vector machines. This is because KNN does not involve a time-consuming training process. Instead, it simply stores the entire training dataset in memory and uses it to make predictions for new data points based on their similarity to the existing data points. However, the prediction time

of KNN can be affected by the number of data points in the training dataset and the dimensionality of the feature space. When the training dataset is large, the prediction time can increase due to the computational cost of searching for the k-nearest neighbors. Similarly, in high-dimensional feature spaces, it can be challenging to find relevant neighbors, which can also lead to slower prediction times. So, while KNN is generally a fast model, the processing time can still be influenced by some factors.

- KNN is a lazy learning algorithm that requires high computational effort for prediction, leading to longer processing times. However, KNN is a fast learning model, and the frequency of data does not significantly affect the training time.

XGBoost

- Raw location can be an important feature in XGBoost models, especially in tasks related to spatial analysis or movement prediction. However, it's worth noting that the importance of features in XGBoost can vary depending on the specific task and dataset. It's always recommended to perform feature selection or feature importance analysis to identify the most relevant features for the task at hand.
- Raw location is feature that importance in XGBoost model because this feature can improve accuracy of model. Raw location is an important feature in XGBoost because it provides important contextual information about the location of the data points, which can improve the accuracy of the model.
- The accuracy of XGBoost model drops significantly in data frequency lower than 2400 sec in all cases because this low frequency data may not contain enough samples to represent the pattern of the data. As a result, the model may not be able to learn from the data and generalize well to new data. However, the impact of data frequency on the accuracy of the XGBoost model may also depend on other factors such as the complexity of the model, the quality of the data, and the hyperparameters used for training the model.

- The difference in accuracy and all value between XGBoost that uses all feature and XGBoost that uses only raw location and bearing is not significant, especially if raw location is the most important feature in the data. However, this may depend on the specific dataset and the complexity of the relationships between the features. It is generally recommended to perform feature selection and evaluate the performance of different models using different subsets of features to determine the best approach for a given problem.
- While raw location may have a high feature importance score in XGBoost, it doesn't necessarily mean that it will always improve the performance of the model in all areas and data frequencies. Feature importance scores are relative to the other features in the dataset and can change depending on the specific dataset being used. Additionally, while raw location may be important in some scenarios, it may not be as important in others. Therefore, it's important to carefully evaluate the impact of each feature on the model performance in the specific context of the problem being addressed.
- The training time of XGBoost depends on several factors such as the size of the dataset, the number of features, the complexity of the model, the number of trees in the ensemble, and the hardware used for training. While it is true that XGBoost is optimized for performance and can be faster than some other algorithms, it can still take a considerable amount of time to train on large datasets with many features. In general, as the size and complexity of the dataset increase, the training time for XGBoost will also increase.

### Case3 compares the map matching result from the machine learning model with the python library for the map matching method

This In this case, we compare the performance of two methods for map matching - the python library called NoisePlanet and a machine learning model. NoisePlanet is a python package that provides two modes for matching points to the road network. The first mode is based on distance and direction and the second mode uses a Hidden Markov Model (HMM) to model the

movement of a vehicle. On the other hand, the machine learning model used in this study is a Random Forest algorithm trained on a dataset of GPS points.

We evaluated the performance of both methods by calculating the accuracy and all value metrics for each method using a dataset of GPS points. Our analysis revealed that the Random Forest model outperformed the NoisePlanet library in terms of accuracy and all value. The Random Forest model achieved higher accuracy and all value scores across all data frequencies compared to the NoisePlanet library.

Furthermore, we also analyzed the impact of different features on the performance of the Random Forest model. Our findings showed that the model performed better when using all features compared to using only raw location and bearing. However, the impact of features on the performance of the model varied depending on the data frequency.

The NoisePlanet python library for map matching method called noiseplanet is a python package to adjust point location and contains 2 modes for matching points in the road as below:

- matching to the nearest edge
- hmm based matching.

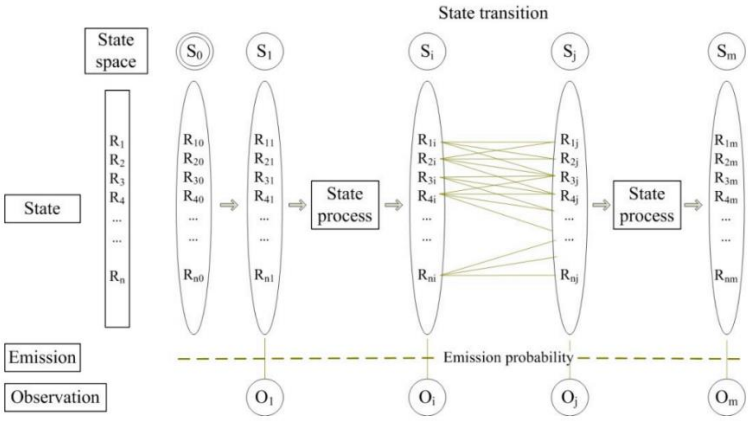



*Figure 62 HMM*

Figure 62 shows the Hidden Markov Model process used in the map matching method of noiseplanet, which resulted in good precision. In this case, we compare the results obtained using the HMM mode in noiseplanet for adjusting points, with the results obtained using a machine learning model.

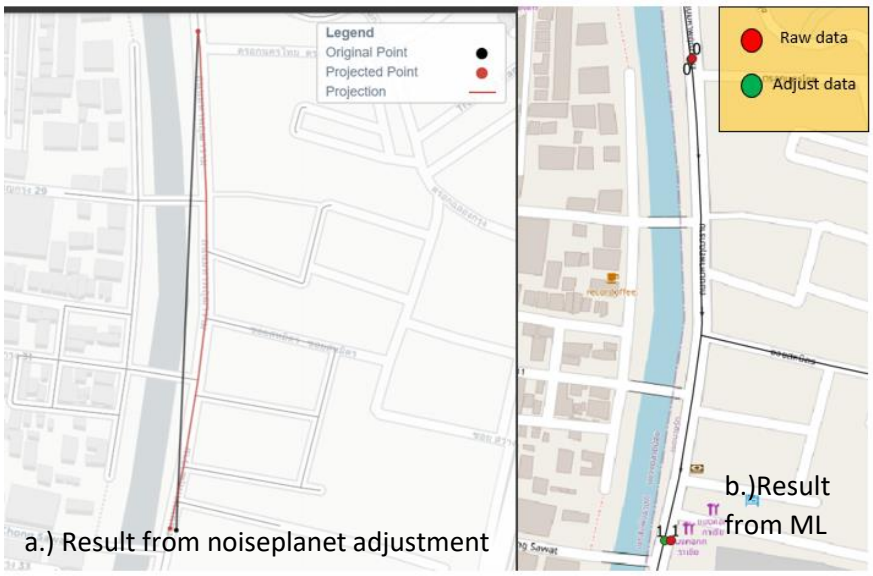



*Figure 63 Comparison of results from a.) Result from noiseplanet adjustment and b.)Result from ML*

After comparing the results of map matching using the python library noiseplanet and the machine learning model, we can observe that the two methods produce similar results with points accurately adjusted to the road. However, the processing time for the noiseplanet library is slightly longer compared to the machine learning model. This is because the noiseplanet library requires a data preparation process for graph-based adjustment, which takes time, followed by the matching process. On the other hand, the machine learning model requires time for data preparation and prediction, which is less than the noiseplanet library.

*Table 8 comparison of Noiseplanet and ML*

| Step_Number | Noiseplanet | ML |
|---|---|---|
| Step1 | Create graph network:1.074 mins | Prepare data for prediction:0.300 mins |
| Step2 | Mapmatching method : 2.602 mins | Predict class wayid : 0.037 mins |
| Step3 | - | Adjust point in road : 0.057 mins |
| Summary | 3.676 | 0.394 |

## Compare map matching method with machine learning with another research

Comparing the machine learning model used in this research with other studies, it can be observed that many studies have used traditional map matching methods such as Markov chain and Hidden Markov Method (HMM), which are suitable for GPS high frequency data but do not perform well in GPS low frequency data and take a lot of time to process the data. This research aims to overcome the limitations of traditional map matching methods by using advanced features

and machine learning algorithms. The study explores the use of new machine learning algorithms and deep learning algorithms to improve the map matching method. Additionally, the study tries to address the challenges of adjusting GPS location in urban areas by using features other than just the raw location of GPS points. This is important because GPS location in urban areas is often less accurate due to multipath problems and environmental factors.

## Example Result and Discussion

After training the classification models and applying them for prediction on the test dataset, the performance of the models was evaluated using the confusion matrix. This metric is commonly used in classification tasks to evaluate the accuracy of the model's predictions. The confusion matrix displays the number of true positives, true negatives, false positives, and false negatives, which are then used to calculate various performance metrics such as precision, recall, and F1-score. By analyzing the values in the confusion matrix and calculating these performance metrics, we can determine how well the model is performing and make necessary adjustments to improve its accuracy.

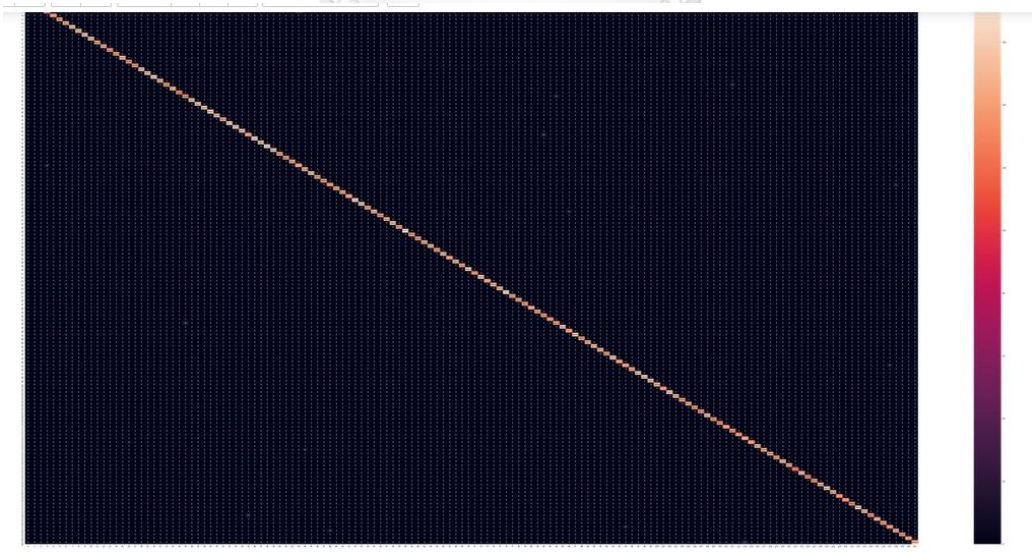


*Figure 64 Confusion Matrix result*

The result after applying the machine learning model to the iTic data is a CSV file that includes several columns. The first column is the driverID, which indicates the unique ID of the

driver. The second column is the pingtimestamp, which is the timestamp of the GPS point. The third column is the predicted class of road based on the machine learning model. The fourth and fifth columns are the latitude and longitude values after adjustment by the map matching method. The adjusted location values are expected to be more accurate and closer to the actual location on the road than the raw GPS location values. This CSV file can be further analyzed and processed for various applications, such as traffic analysis, route optimization, and driver behavior analysis.

| | driverid | pingtimestamp | adjust_lat | adjust_lng | predict_class |
|---|---|---|---|---|---|
| 0 | 94108d01e3a591a2bf0c9f41352072dda75d877847d5b9... | 1574405102 | 13.753678 | 100.597556 | 26575879 |
| 1 | 5313322a87bde575960c891f4cbc96a59e6cd88e8fe587... | 1574405090 | 13.746808 | 100.575236 | 178872792 |
| ... | ... | ... | ... | ... | ... |
| 1169656 | d4610fca730e217fe861eb9123663a9ee99799e800eaa5... | 1574392358 | 13.743836 | 100.562324 | 332206412 |
| 1169657 | 4b34ac7a1e47e80e3f8c2f095f21e79d2df238837665d5... | 1574389040 | 13.731601 | 100.582178 | 690606633 |
| 1169658 | 4b34ac7a1e47e80e3f8c2f095f21e79d2df238837665d5... | 1574389044 | 13.731575 | 100.582260 | 690606633 |
| 1169659 | 4b34ac7a1e47e80e3f8c2f095f21e79d2df238837665d5... | 1574389048 | 13.731555 | 100.582323 | 690606633 |
| 1169660 | 44d21cc5bf8e478c83443b67484bdb8a53d3fdd121fbdf... | 1574392347 | 13.747254 | 100.563004 | 722659603 |

*Figure 65 Example of raw data in iTic data before adjustment and after adjustment with model.*

After applying machine learning model to adjust GPS raw data , the result is as below:

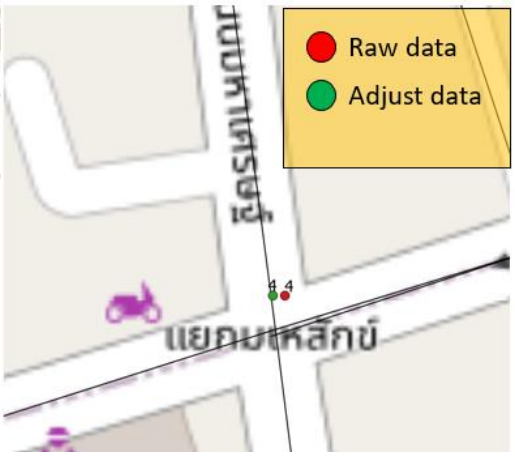



*Figure 66 Example point after adjust*


*Figure 67 Example point after adjust*

The study showed that in urban areas where GPS data cannot perform well and GPS points contain high errors due to multipath problems, the machine learning model that uses bearing, previous GPS points, and raw location features can improve the accuracy of the GPS points. The machine learning model adjusts the GPS points by considering the direction of the car and the route. This is done by using previous points, bearing, and raw location features to adjust the GPS points. By doing so, the model can overcome the limitations of traditional map matching methods such as Markov chain and Hidden Markov Method (HMM), which are suitable for high-frequency GPS data but perform poorly in low-frequency GPS data and take a lot of time to process the data. The results of the machine learning model were evaluated using confusion metrics and compared to traditional map matching methods, and the model was found to perform well in improving the accuracy of GPS points in urban areas with high multipath errors.




*Figure 68 Example1 Map matching method result in High building area*


*Figure 69 Example2 Map matching method result in High building area*

# Conclusion

## Conclusion of Case1 Comparison of accuracy and time model Random Forest classification, Decision Tree, Multi-layer Perceptron Classifier (MLPClassifier), XGBoost, K-Nearest Neighbors (KNN) and Long Short-Term Memory (LSTM) by using the training model of low-frequency data( Grab GPS data that reduces density)

The results of case study1 show that the performance of machine learning models depends on the area and features used. Random Forest and Decision Tree models performed well in all areas, but Decision Tree model had faster training and prediction times. XGBoost model performed well in some areas, especially in Area2 where the raw location feature had a high impact on the accuracy of the results. LSTM and MLP, which are neural network models, performed well in Area2 that had a high sample density, but not in Area0 which had a low sample density. KNN performed well in Area3 that had a high density of samples and roads. Since KNN is a lazy learning algorithm, it takes less time for training but more time for prediction. Area0 with low sample density was challenging for many models to perform well, but these models had shorter prediction and training times. In contrast, Area1 with a high sample density required a longer time for training and prediction. Area2 was the most balanced area where most models could perform well due to the balanced number of samples and road density.



*Table 9 Compare result in case study1*

| Algo | Suitable_Area | Not_perform_well_Area | Training time | Prediction time |
|---|---|---|---|---|
| **RF All Feature** | All area | - | Medium | Low |
| **RE exclude raw location** | All area | - | Medium | Low |
| **DC All Feature** | All area | - | Low | Low |
| **DC exclude raw location** | All area | - | Low | Low |
| **XGB All Feature** | Area2 | Area0 | Low | Medium |

| Algo | Suitable_Area | Not_perform _well_Area | Training time | Prediction time |
|---|---|---|---|---|
| **XGB exclude raw location** | - | All area | Low | Medium |
| **MLP** | Area2 | Area0 | High | High |
| **LSTM** | Area2 | Area0 | High | High |
| **KNN All Feature** | Area3 | Area0 | Low | High |
| **KNN exclude raw location** | Area3 | Area0 | Low | High |

Conclusion of Case2 compares accuracy and time model Random Forest classification, Decision Tree, K-Nearest Neighbors (KNN) and Long Short-Term Memory (LSTM) by using the training model of low-frequency data (Grab GPS data that reduces density) and by using the training model of variant frequency data in 30 classes.

The results of case study 2 show that the performance of machine learning models depends on both the frequency of data and the features used. Models that use all available features perform better than those that use only raw location and bearing data when the data frequency is between 2000-4000 sec. This is because features such as previous and next point locations can improve the model's accuracy in terms of route and environment. However, models that use only raw location and bearing data have higher accuracy when the data frequency is lower than 2000 sec because other features such as previous and next point locations may not be reliable in this case.

All machine learning algorithms experience a significant drop in accuracy when the data frequency is lower than 2000 sec. Among the models that use all features, Random Forest performs the best. In contrast, KNN that uses all features has the lowest accuracy across all data

frequencies. Decision Tree is the fastest model in terms of prediction time and training time across all data frequencies.

*Table 10 Compare result in case study2*

| Algo | Accuracy,Precision,Recall and F1-Score | Training time | Prediction time |
|---|---|---|---|
| **DC use All Feature** | Perform well with data frequency 0-2000 sec | Not depend on data frequency | Not depend on data frequency |
| **DC use only raw location and bearing** | Suitable for data frequency lower than 2000 sec | Not depend on data frequency | Not depend on data frequency |
| **DC excludes raw location** | Not Perform well with data frequency lower than 2000 sec | Not depend on data frequency | Not depend on data frequency |
| **RF use All Feature** | Perform well with data frequency 0-2000 sec | Depend on data frequency | Depend on data frequency |
| **RF use only raw location and bearing** | Suitable for data frequency lower than 2000 sec | Depend on data frequency | Depend on data frequency |
| **RF excludes raw location** | Perform well with data frequency 0-2000 sec | Depend on data frequency | Depend on data frequency |
| **KNN use All Feature** | Perform well in high frequency of data | Not depend on data frequency | Depend on data frequency |
| **KNN use only raw location and bearing** | Perform well with data frequency 0-2000 sec | Not depend on data frequency | Depend on data frequency |
| **KNN excludes raw location** | Perform well in high frequency of data | Not depend on data frequency | Depend on data frequency |
| **XGB use All Feature** | Perform well with data frequency 0-2400 sec | Depend on data frequency | Depend on data frequency |
| **XGB use only raw location and bearing** | Perform well with data frequency 0-2400 sec | Depend on data frequency | Depend on data frequency |
| **XGB excludes raw location** | Not perform well in all frequency data | Depend on data frequency | Depend on data frequency |

Conclusion of Case3 compares the map matching result from the machine learning model with the python library for the map matching method

The results from case study 3 demonstrate that the machine learning model is faster than the Python package. While the machine learning model takes more time in processing data, the

Python package takes longer to create the graph network. When comparing accuracy, there is not a significant difference between the Python package and the machine learning model in areas without a complex environment. However, the machine learning model can handle GPS data in complex and multipath areas better than the Python package.

## Conclusion of Study

After creating machine learning for map matching, we found that the most importance features are raw location , bearing and speed. Random Forest Model is the model that contains the highest precision, recall,F1-Score and accuracy but takes prediction time and training time more than Decision Tree. In Case Study1,Area2(medium road density and sample size) contain high accuracy in all models. On the other hand, Area0(low sample size data and low road density ) contains low value in most models. Random forest can preform well in all areas. In Case Study 2, all model values drop depend on frequency of data. In data frequency 0-1900 sec that uses all features can improve accuracy but when data frequency is more than 1900 sec, it uses only bearing and raw location features is better than use all features. Random Forest Model and Decision Tree has low sensitivity but KNN is the model with high sensitivity and accuracy depending on frequency of data. In conclusion, Random Forest is the model that can preform well in all areas and data frequency but take much time for prediction and training model but Decision Tree is a little bit lower in accuracy,precision,recall and F1-Score when compared with Random forest but is faster than Random Forest both in terms of prediction time and training time. Random Forest is a reliable model for map matching but it can be computationally expensive. Decision Tree, on the other hand, is faster but may sacrifice some accuracy. In this study, we recommend utilizing both Random Forest and Decision Tree models for the process map matching method with low-frequency GPS data. It is advisable to carefully consider the trade-off between accuracy and processing time when choosing a model. It's also interesting to note that the importance of features varies depending on the data and the area. Overall, it's important to carefully evaluate the performance of different machine learning models to find the best fit for the specific problem at hand.

Random Forest and Decision Tree algorithms are indeed suitable for map matching of GPS data with a frequency of 1 minute, particularly for traffic analysis in a specific study area. However, it's important to note that when using these algorithms for different applications or in

different areas, you should consider adjusting the frequency of the data to be suitable for the specific application and area.The frequency of the GPS data collection should be determined based on the requirements of the application and the dynamics of the area being studied. For example, in a high-traffic urban environment, a higher frequency of GPS data collection, such as every few seconds, might be necessary to capture fine-grained traffic patterns and provide accurate results.Conversely, in a less dynamic or rural area with lower traffic density, a lower frequency of GPS data collection may be sufficient, such as every few minutes or even longer intervals.It's essential to align the data collection frequency with the goals of the analysis and the characteristics of the area under study to ensure meaningful and accurate results.

*Table 11 Comparison of model performance*

| Task | RF | DC | MLP | LSTM | XGBoost | KNN |
|---|---|---|---|---|---|---|
| F1-Score,Precision,Accuracy and Recall | High | High | Low | Low | Medium | Medium |
| Prediction time | Low | Low | Low | High | Medium | High |
| Training time | Medium | Low | High | High | Low | Low |
| Sensitivity of frequency | Low | Low | High | High | Medium | High |

## Future Work

Using the Spark platform is a great way to improve the performance of data processing and machine learning model training. Spark provides a distributed computing framework that can handle large-scale data processing and machine learning workloads across a cluster of machines. Spark can improve processing time by allowing parallel processing of data preparation and training model steps across multiple nodes in a cluster. This can lead to significant reductions in processing time, especially for large datasets.

In addition, Spark can improve prediction time by enabling distributed processing of GPS data points. With Spark, it is possible to handle the processing of multiple GPS data points in parallel, which can significantly reduce prediction time. This is especially important for real-time applications, where quick response times are critical.

Overall, using the Spark platform can help improve the performance of the map matching algorithm by reducing processing time and improving prediction time, which can lead to better accuracy and more efficient handling of large-scale GPS data.

Using MLflow in the data science pipeline can indeed be helpful. MLflow is a platform-agnostic tool for managing the end-to-end machine learning lifecycle. It provides various functionalities such as tracking experiments, packaging code into reproducible runs, and sharing and deploying models.



Using MLflow can help with model selection by allowing you to keep track of multiple experiments and their results in a centralized location. You can compare different models and their hyperparameters, and identify which one works best for your use case. MLflow also provides tools for tracking and logging data, so you can keep track of the inputs and outputs of each experiment.

Additionally, using MLflow can help with data preparation by allowing you to version data, track changes, and reproduce previous data transformations. This can help you ensure data quality and consistency across different runs of the pipeline.

In summary, using MLflow can improve the data science pipeline by providing a centralized location for tracking experiments, model selection, and data preparation. It can help ensure reproducibility, consistency, and quality in the pipeline.

Reinforcement learning can be a useful technique to improve the accuracy of machine learning models in situations where the data is complex or the environment is constantly changing. By using trial-and-error learning, reinforcement learning algorithms can adapt and improve their performance over time.

To implement reinforcement learning in the context of map matching, the model could be trained to learn from its mistakes and adjust its predictions based on the feedback it receives. For example, if the model predicts a location that is significantly different from the true location, it could receive a penalty and adjust its parameters to make a better prediction in the future. Additionally, increasing the size of the dataset can improve the quality and performance of the model. By having more data, the model can learn from a wider range of examples and improve its ability to generalize to new situations. However, it's important to ensure that the additional data is representative of the problem space and doesn't introduce bias or noise into the model.



To integrate a machine learning model with a server and create a batch processing pipeline for data preparation, training, and retraining in case of accuracy drop or data drift, as well as handle online processing of raw location and attribute data for prediction with the machine learning model, and return adjusted locations, It can follow these steps:

1. Data Collection: Gather the raw location and attribute data from the server or external sources.
2. Preprocessing: Perform data cleansing, transformation, and feature engineering to prepare the raw data for training the machine learning model such as cleansing missing values ,remove noise and  normalization.

3. Batch Processing Pipeline: Design a batch processing pipeline that automates the data preparation steps and model training. This pipeline can be scheduled to run periodically or triggered based on specific conditions.
4. Initial Model Training: Train the machine learning model using the prepared data. Select an appropriate algorithm, split the data into training and validation sets, and tune hyperparameters. Evaluate the model's performance using appropriate metrics.
5. Model Monitoring: Continuously monitor the model's performance and accuracy metrics. Set up mechanisms to detect accuracy drops or data drift. This can involve monitoring validation metrics, comparing them with predefined thresholds, or using statistical techniques.
6. Retraining: If accuracy drops or data drift is detected, initiate the retraining process. Incorporate new data into the training set and retrain the model. This step ensures that the model adapts to changes in the data distribution and maintains optimal performance.
7. Online Prediction: Implement an online process to handle incoming raw location and attribute data in real-time. Utilize the trained machine learning model to predict adjusted locations based on the input data.
8. Output Adjustment: Return the adjusted locations to the server or client application for further processing or display.

# REFERENCES


[1] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, *Map-matching for low-sampling-rate GPS trajectories*. 2009, p. 361. doi: 10.1145/1653771.1653820.

[2] S. Saki and T. Hagen, "A Practical Guide to an Open-Source Map-Matching Approach for Big GPS Data," *SN Comput. Sci.*, vol. 3, no. 5, p. 415, Aug. 2022, doi: 10.1007/s42979-022-01340-5.

[3] L. Yu, Z. Zhang, and R. Ding, "Map-Matching on Low Sampling Rate Trajectories through Frequent Pattern Mining," *Sci. Program.*, vol. 2022, p. 3107779, Mar. 2022, doi: 10.1155/2022/3107779.

[4] P. Newson and J. Krumm, "Hidden Markov Map Matching Through Noise and Sparseness," in *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2009), November 4-6, Seattle, WA*, Nov. 2009, pp. 336–343. [Online]. Available: https://www.microsoft.com/en-us/research/publication/hidden-markov-map-matching-noise-sparseness/

[5] A. Salhi, G. Fahmi, and A. Fakhfakh, "Estimation for Motion in Tracking and Detection Objects with Kalman Filter," 2020. doi: 10.5772/intechopen.92863.

[6] W. Bian, G. Cui, and X. Wang, "A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories," *Sensors*, vol. 20, no. 7, 2020, doi: 10.3390/s20072057.

[7] B. Y. Chen, H. Yuan, Q. Li, W. Lam, and S.-L. Shaw, "Map matching algorithm for large-scale low-frequency floating car data," *Int. J. Geogr. Inf. Sci.*, vol. 28, pp. 22–38, Jan. 2014, doi: 10.1080/13658816.2013.816427.

[8] M. Quddus and S. Washington, "Shortest path and vehicle trajectory aided map-matching for low frequency GPS data," *Eng. Appl. Sci. Optim. OPT- - Profr. Matthew G Karlaftis Meml. Issue*, vol. 55, pp. 328–339, Jun. 2015, doi: 10.1016/j.trc.2015.02.017.

[9] J. Fang *et al.*, *A Map-matching Algorithm with Extraction of Multi-group Information for Low-frequency Data*. 2022.

[10] Y.-L. Hsueh and H.-C. Chen, "Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions," *Inf. Sci.*, vol. 433–434, pp. 55–69, Apr. 2018, doi:

10.1016/j.ins.2017.12.031.

[11] A. Luo, S. Chen, and B. Xv, “Enhanced Map-Matching Algorithm with a Hidden Markov Model for Mobile Phone Positioning,” *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 11, 2017, doi: 10.3390/ijgi6110327.

[12] M. Hashemi and H. A. Karimi, “A Machine Learning Approach to Improve the Accuracy of GPS-Based Map-Matching Algorithms (Invited Paper),” in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, 2016, pp. 77–86. doi: 10.1109/IRI.2016.18.

[13] G. Zhou and F. Chen, “DRFMM: a map-matching algorithm based on distributed random forest multi-classification,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 189, no. 5, p. 052014, Nov. 2018, doi: 10.1088/1755-1315/189/5/052014.

[14] X. LAI, J. CHEN, J. CAO, and F. XIA, “Map Matching Integration Algorithm Based on Historical Trajectory Data,” in *2019 IEEE Symposium on Product Compliance Engineering - Asia (ISPCE-CN)*, 2019, pp. 1–6. doi: 10.1109/ISPCE-CN48734.2019.8958632.

# VITA

| | |
|---|---|
| **NAME** | Ornicha Sinthopvaragul |
| **DATE OF BIRTH** | 14 June 1995 |
| **PLACE OF BIRTH** | Bangkok |
| **INSTITUTIONS ATTENDED** | 2014-2018 Bachelor in Survey Engineering, Chulalongkorn University |
| **HOME ADDRESS** | 33/222 Bang Khae , Bang Phli District ,Province Samut Prakan |