MODELING VESSEL BEHAVIOURS BY CLUSTERING AIS DATA USING OPTIMIZED DBSCAN

XUYANG HAN

A RESEARCH THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

FEBRUARY 2021

Abstract

Today, maritime transportation represents substantial international trade. Sustainable development of marine transportation requires systematic modeling and surveillance for maritime situational awareness. In this research thesis, we present an enhanced density-based spatial clustering (DBSCAN) method to model vessel behaviors. The proposed methodology enhances the DBSCAN clustering performance by integrating the Mahalanobis Distance metric that considers the correlations of the points representing the locations of the vessels. The clustering method is applied to historical Automatic Identification System (AIS) data and generates an action recommendation tool and a model for detecting vessel trajectory anomalies. Two case studies present outcomes from the openly available Gulf of Mexico AIS data, and Saint Lawrence Seaway and Great Lakes AIS licensed data acquired from ORBCOMM (a maritime AIS data provider). This research proposes a framework for modeling AIS data, an algorithm for generating a clustering model of the vessels' trajectories, and a model for detecting vessel trajectory anomalies such as unexpected stops, deviations from regulated routes, or inconsistent speed. This work's findings demonstrate the applicability and scalability of the proposed method for modeling more water regions, contributing to situational awareness, vessel collision prevention, safe navigation, route planning, and detection of vessel behavior anomalies for auto-vessels development.

Acknowledgments

My deepest gratitude to my advisors, Prof. Costas Armenakis and Prof. Mojgan Jadidi, who have consistently and diligently supported me to be a better student and researcher to succeed in finishing outstanding research in this journey. Without their guidance, it would be impossible to learn broad concepts in the Data Science research field focusing on spatial data. I want to express my gratitude for providing me the incredibly valuable opportunity to research the computer field while learning programming skills for my future career. Nevertheless, more than their guidance in my work, I want to thank them for their ability to constantly push me out of my comfort zone and continue the never-ending process of learning.

Also, gratitude to my committee member, Prof. Manos Papagelis, for being such a great professor and advisor, always patiently offering ideas and inspirations. I want to thank The Bureau of Ocean Energy Management (BOEM) and the National Oceanic and Atmospheric Administration (NOAA) for providing the open-sourced AIS dataset for this research. I want to thank ORBCOMM for collecting the maritime AIS data and thank York University for its legal support in requesting AIS data from ORBCOMM. I want to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and York University for their funding support. I also appreciate my family, who supported my education abroad in Canada. Finally, gratitude to all my friends, teammates, and my cat Lagu who comforted and supported me during the COVID-19 pandemic times.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AIS	Automatic Identification System
DBSCAN	Data Using Optimized Density-Based Spatial Clustering of Applications with Noise
Eps	Defined Radius to Determine Neighboring Points in DBSCAN
MinPts	Defined Minimum Number of Data Points to Determine Core Points in DBSCAN
SOG	Speed Over Ground
COG	Course Over Ground
ANN	Artificial Neural Networks
KNN	K Nearest Neighborhoods
SVM	Support Vector Machine
CNN	Convolutional Neural Network
MMSI	Maritime Mobile Service Identity

Chapter 1: Introduction

1.1 Research Motivation

Today maritime transportation represents 90% of international trade volume, and more than 50,000 vessels are sailing the ocean every day. Therefore, systematically modeling and surveillance should be of high priority in the maritime domain to reduce maritime transportation security risks. Statistically, between 75% and 96% of maritime accidents are caused by human error due to fatigue or misjudgment (Bernard Marr, 2019; Dana, 2019). Navigation safety contributes to a sustainable society by reducing marine transportation accidents, protecting the marine environment and creatures from exposure to hazardous chemicals leakage from vessel collisions. Besides, auto-vessels developments contribute to surveying and transportation efficiency, promoting and facilitating sustainable and cost-saving industries (Bernard Marr, 2019; Dana, 2019). The auto-vessels should be the most promising automatic vehicles to be implemented shortly, due to fewer barriers to adoption than unmanned vehicles driving on the road (Bernard Marr, 2019; Dana, 2019) and unmanned aerial vehicles with a more complex operational air domain (Vespe *et al.*, 2012). Auto-vessels equipped with autonomous and semi-autonomous systems can reduce human intervention reliance, making our oceans and maritime navigation safer. In December 2018, Rolls-Royce and Fin-ferries demonstrated the world's first fully autonomous ferry (Jallal, 2018). However, the ships were only deployed on simple inland where waters are calm, the route is simple, and there is not high traffic. Indeed, there is still

a long way to go in the design and development of auto-vessel-related research, with elements including route planning and trajectory anomalies detection, situational awareness, and intelligent responses toward changing environments. This research focuses on the first element, route planning, and anomalies detection, by proposing an algorithm for generating a clustering model of the vessels' trajectories and a model for detecting vessel trajectory anomalies such as unexpected stops, deviations from regulated routes, or inconsistent speed.

In this regard, reliable open-sourced data sources for studying vessel behaviors and generating nautical routes are the historical and real-time maritime Automatic Identification System (AIS) data (Silveira, Teixeira and Soares, 2013; Sheng and Yin, 2018). AIS is an automatic tracking system to identify and locate vessels by exchanging data with other nearby ships, AIS base stations, and satellites. According to the Safety of Life at Sea (SOLAS) convention, ships of 300 gross tonnages and upwards in international voyages, 500 and upwards for cargoes not in international waters, and passenger's vessels are obliged to be embedded with AIS equipment, making AIS data abundant globally (IMO, 2000). Furthermore, AIS becomes a worldwide data standard, and therefore this coherent source of information can be suitable for global marine transportation traffic modeling and analysis. In this research, we use open-sourced AIS data as the primary data source for the proposed algorithm testing and generate models based on big data from the Saint Lawrence Seaway Region. Since AIS data always contain inaccurate and uncertainty noise, outlier detection and filtration are required when organizing and modeling AIS data. Also, given a large

amount of AIS data, this is more feasible to adopt unsupervised learning in modeling and anomaly detection processes with a high degree of automation.

Thus in this research, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is proposed to be used as the foundation of the marine trajectory modeling. DBSCAN, an unsupervised method, is now available in many clustering libraries and is widely used in many real-world applications (Hall *et al.*, 2009; Pedregosa FABIANPEDREGOSA *et al.*, 2011; R Core Development Team, 2013; Schubert *et al.*, 2015). As DBSCAN relies on a density-based notion of clusters, this considers being an effective method to discover clusters of arbitrary shapes and identify outliers (Ester *et al.*, 1996). Thus, DBSCAN demonstrates vast potentials to be applied to marine trajectory clustering. However, applying the traditional DBSCAN clustering method has a considerable shortcoming with spatially unevenly distributed actual AIS data (Academy *et al.*, 2009; Esmaelnejad, Habibi and Yeganeh, 2010; Smiti and Eloudi, 2013; Fong *et al.*, 2014; Karami and Johansson, 2014; Sawant, 2014; Liu, 2015; Hou, Gao and Li, 2016; Schubert *et al.*, 2017; Han, Armenakis and Jadidi, 2020), making it an unreliable method to be applied to marine trajectory clustering without optimization. The traditional DBSCAN method requires two input parameters, *MinPts* (minimum number of data points) and *Eps* (radius to determine neighboring points), and the user needs to determine appropriate values for them. However, in real life, it is very difficult to find the optimal parameters when the data and scale cannot be well understood (Academy *et al.*, 2009; Esmaelnejad, Habibi and Yeganeh, 2010; Smiti and Eloudi, 2013; Fong *et al.*, 2014; Karami and Johansson, 2014; Sawant, 2014; Liu, 2015; Hou, Gao and Li, 2016;

Schubert *et al.*, 2017; Han, Armenakis and Jadidi, 2020). Furthermore, as the traditional DBSCAN is based on the Euclidean distance metric, this sometimes cannot handle clusters with complex shapes and distribution (Ren, Liu and Liu, 2012; Sangeetha, Padikkaramu and Chellan, 2018). Thus, innovative distance metrics need to be proposed to handle data with complicated spatial distributions, and to optimize the DBSCAN performance in terms of homogeneity, completeness, and other evaluation metrics.

1.2 Research Objectives

The main aim of this research project is to design a model that is capable of planning marine transportation routes and detecting abnormal vessel trajectories. This main aim is achieved by accomplishing the following objectives:

Objective 1: To study and optimize the DBSCAN clustering method and propose a method to auto-select the required parameters.

Objective 2: To propose a novel representation of marine trajectory data and propose a clustering framework to apply the enhanced DBSCAN to actual AIS big data.

Objective 3: To validate the proposed clustering algorithm and framework by comparing it to other commonly used machine learning techniques and apply them to case studies to generate the marine transportation models.

Figure 1–1 shows the general structure of the research work.

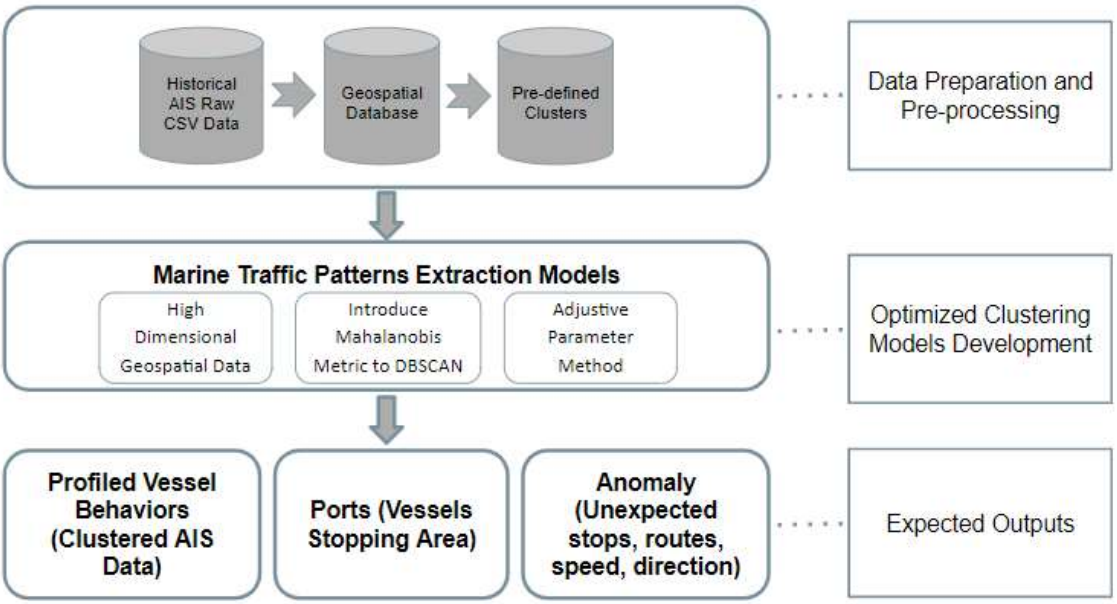


Figure 1–1. The workflow of the research

This model utilizes the Mahalanobis distance metric and the adaptive parameter method on DBSCAN and implemented high dimensional geospatial data. The main result of clustering is a set of generated highways on the ocean. The anomalous behaviors can be detected in real-time, considering the longitude, latitude, direction, and speed of the vessel. This approach can be applied to real-time AIS surveillance.

1.3 Scientific Contribution

The contributions of this research work are the following: First, the proposed clustering algorithm, specifically applying marine trajectory clustering, aims to contribute to marine transportation route planning and abnormal behavior detection. The research

proposes a framework to process massive and messy marine AIS data and generate a transportation model. Overall, the model developed in this research is based on enhancing the DBSCAN clustering method and is set to be applied to historical or real-time AIS data. Through organizing similar AIS data and clustering them together, the vessel behaviors can be profiled into labeled clusters, each of which represents a specific vessel behavior stage. Within each behavior stage, the vessel behaviors share maximum similarities and are different from other clusters. Marine transportation route planning can be done by selecting a series of stages provided by the model. The model can monitor vessels by detecting any anomaly behaviors by collecting new AIS data from vessels traveling in the already modeled region. For the waters which lack systematic route planning, the proposed unsupervised clustering method can be applied to plan “highway on the ocean.” The proposed algorithm can be applied for the waters with established routes to update these prior clusters from the new-collected AIS data. The model can also provide prospective routes and action recommendations based on autonomous vessels' location, facilitating and contributing crucial progress on Artificial Intelligence vessel s research (AI-vessels). Taking advantage of the proposed model, the autonomous vessels can stay on a safe route, with a safe speed and direction (heading) following the recommended route. In general, this research thesis provides a possible process for analyzing, clustering, and modeling AIS data, contributing to the sustainable marine transportation research community toward auto-vessel development. We believe the method and the results are very beneficial to marine transportation management and hydrographic research communities.

Second, a similar data analytic framework can also be applied to other data sources for more general analysis purposes. The proposed optimized DBSCAN clustering algorithm provides new understandings of the DBSCAN clustering method. The proposed point-based trajectory clustering algorithm and the framework to process unlabeled data beyond AIS data, prepare labeled training data, and generate classification AI models can be applied to various data mining domains. Using the proposed modified DBSCAN method, the algorithm improves the clustering performance on unevenly distributed data and solves the problem of finding parameters wisely based on the dataset characteristics, in which the traditional DBSCAN faces huge challenges. The machine learning research community can share this research progress and apply it to more general clustering tasks.

1.4 Thesis Outline

This research thesis presents an enhanced DBSCAN clustering method applied to historical or real-time AIS data; therefore, the vessel routes can be modeled, and the trajectories' anomalies can be detected.

The organization of the thesis is as follows. **Chapter 2** provides a comprehensive review of current state-of-the-art marine trajectory data clustering methods and the enhancements on the DBSCAN method.

Chapter 3 provides details about the proposed and developed method and corresponding algorithms. Furthermore, this chapter describes the framework of extracting

vessel behavior patterns and detecting outliers and the three stages of our proposed methodology. First, defines a novel representation of marine trajectory data by increasing the dimensions of the vessel's positioning data by considering additional attributes such as velocity and direction in the clustering process, along with the geospatial information. Second, the DBSCAN clustering method is enhanced by integrating the Mahalanobis Distance metric, taking into account the correlations of the position cluster points aiming to make a better identification process as well as reducing the computational cost. Third, we propose a method to select the parameter automatically based on the data itself.

Chapter 4 presents details about the data used in the experiment, how the synthetic data is generated for testing, how the clustering performance is evaluated, and the results of two case studies where the proposed algorithm has been applied using two big datasets.

Chapter 5 highlights lessons learned and concludes the thesis with future work.

Chapter 2: Literature Review

2.1 Background

According to (Kanevski *et al.*, 2009; Xu and Tian, 2015), clustering can be defined as partitioning the dataset and group them into subsets of typical entries making the data in each subset that share some common characteristics and where different subsets show obvious disparities. Clustering methods are usually unsupervised techniques that infer a function to describe internal relationships between unlabelled data, assisting data analysts to do exploratory data analysis and knowledge discovery (Xu and Tian, 2015). Unsupervised learning is a type of self-organized learning that helps find previously unknown data set patterns without massive pre-existing labeled training data. Through clustering similar data together, the clustering methods provide insight into the hidden structures and dependencies in the datasets. The data analysts gain intuition from the clustering results on how to furtherly design models to process the data. The clustering methods are widely used in many exploratory data analyses, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics, and machine learning.

The clustering algorithms are usually categorized into four types: partitioning methods, hierarchical methods, grid-based and density-based methods. One of the most common examples of the partition techniques is the K-Means algorithm, which divides ‘n’

data objects into ‘k’ numbers of clusters (Ahmed, Seraj and Islam, 2020). **Figure 2–1** shows a process of how the K-means clustering algorithm works when K=3. K-Means algorithm initializes k centroids randomly and then iteratively relocates the partitions until the defined centroid function is converged. As shown in **Figure 2–1a**, three centroids are randomly generated (Point C1 in blue, Point C2 in green, and Point C3 in red) and assign the points nearby to the nearest point initially. Then **Figure 2–1b** and **Figure 2–1c** shows that the K-Means algorithm recalculates each cluster's centroids to update Point C1, C2, and C3 until the cluster assignments no longer change. **Figure 2–1d** shows the result of K-Means clustering.

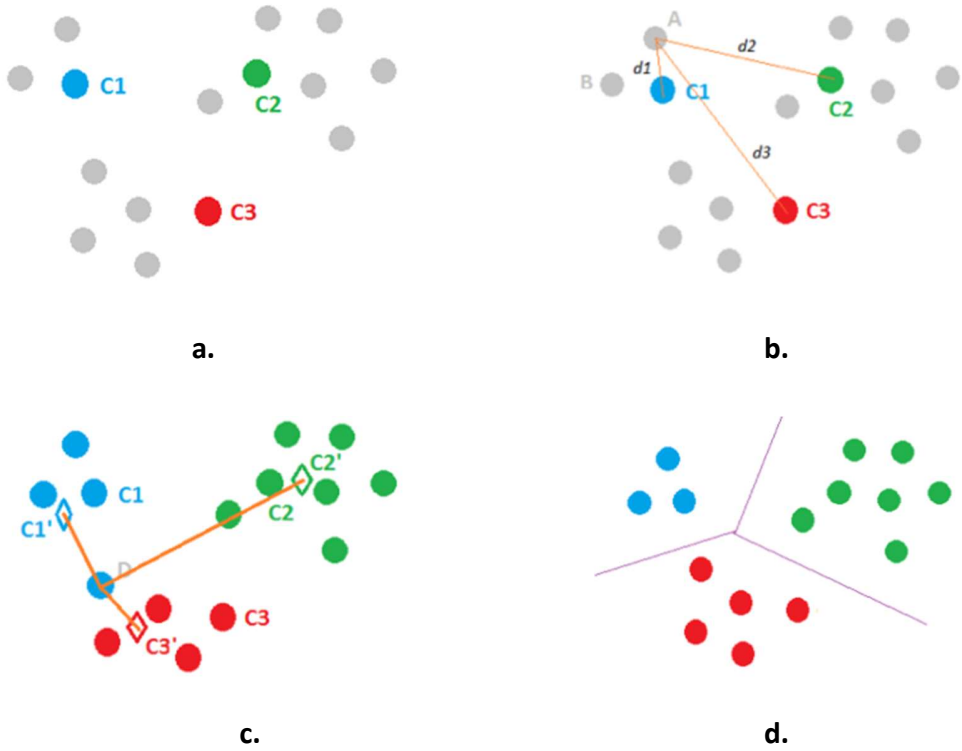


Figure 2–1. Principle of K-Means clustering algorithm (source: healthcare.ai)

The hierarchical clustering models use dendrograms to connect objects nearby (Reddy and Vinzamuri, 2019). The tree-structured models partition and group the objects by considering a specifically defined attribute at each layer. Agglomerative hierarchical clustering is a "bottom-up" approach that merges the clusters' pairs as one moves up the hierarchy. On the contrary, divisive hierarchical clustering is a "top-down" approach, which splits the cluster recursively as one moves down the hierarchy. The grid-based clustering methods create a grid structure by iteratively dividing data space into a finite number of cells until all cells' density is lower than the threshold density (Ilango and Mohan, 2010). Two common types of grid-based clustering methods are STING (STatistical INformation Grid approach) and CLIQUE (CLustering In QUEst) (Wang, Yang and Muntz, 1997; Zaki *et al.*, 1997).

Among various types of clustering approaches mentioned above, density-based clustering algorithms can be the most suitable method for this research because density-based clustering algorithms can find arbitrary shapes of clusters. Density-based clustering algorithms cluster data together by the concept of 'density reachability,' which means the points spatially connect to each other are to be clustered together (Ester *et al.*, 1996). Commonly used density-based clustering algorithms are Density-Based Spatial Clustering of Applications with Noises (DBSCAN), Ordering Points To Identify the Clustering Structure (OPTICS), and DENsity-based CLUstEring (DENCLUE) (Ester *et al.*, 1996; Ankerst *et al.*, 1999; Hinneburg and Keim, 2003). This research adopts DBSCAN as a foundation of the model for

clustering marine transportation trajectories. The details of the DBSCAN clustering method will be discussed in Section 2.3.

Classification algorithms can be considered supervised clustering methods, identifying the data into pre-defined clusters (Kanevski *et al.*, 2009). Supervised algorithms aim to train a model that can determine testing data labels after learning labeled training data. Therefore, supervised algorithms perform tasks based on an understanding of "ground truth", thus the accuracy is usually high. Commonly used classification methods are K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Artificial Neural Networks (ANN). KNN classifies the data by a voting system to determine the category of a new entity (Sun, Du and Shi, 2018). The object is assigned to the class based on the class to which the majority of the K-nearest neighbors belong to. **Figure 2–2** shows how the voting system of the KNN algorithm determines the label of a new observation (the green point). **Figure 2–2** shows assigning k=3 will classify the new observation into the red triangle class while assigning k=5 will classify the new observation into the blue square class.

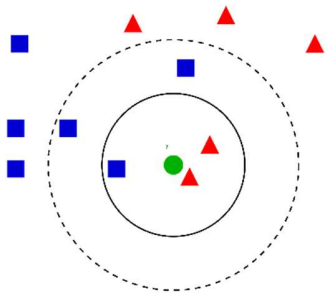
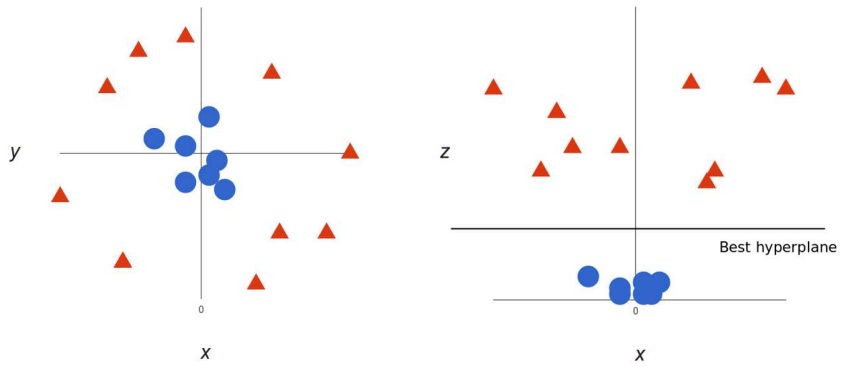


Figure 2–2. KNN algorithm determination of the label of a new observation by the voting system (k=3 in solid line vs. k=5 in dash line) (credit: Wikipedia.com User Antti Ajanki)

Support Vector Machine (SVM) is a classification method by training to find hyperplanes that can partition the data into subsets (Zhou, Zhang and Wang, 2016). **Figure 2–3** shows how SVM finds a hyperplane to separate the dataset by increasing the data dimension.



a. **b.**

Figure 2–3. SVM algorithm use of hyperplanes to partition the data (credit: Monkeylearn.com user Bruno Stecanella)

Artificial Neural Network is another widely used supervised classification algorithm. The network is constructed by a number of layers of connected neurons, each of which is represented by a regression model (Harvey and Harvey, 1998). **Figure 2–4** shows how a typical ANN is structured. Neural networks cluster the data by classifying observations into the pre-defined or pre-labeled clusters.

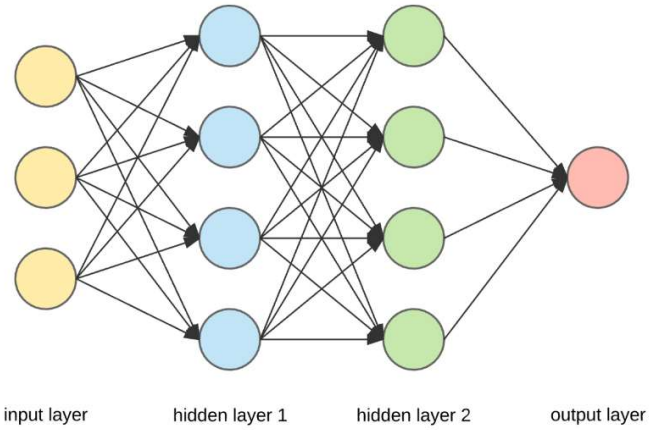


Figure 2–4. A typical Artificial Neural Network (ANN) structure (source: kdnuggets.com)

This research is inspired by both the unsupervised clustering methods and the supervised classification methods. Two components are designed in the proposed model based on the two methods, and the details are discussed in Chapter 3. The following Section 2.2 reviews how those mentioned clustering and classification methods are applied in trajectory clustering.

2.2 Trajectory Data Mining

Following the previous Section 2.1, which generally discussed commonly used clustering and classification methods, this section discusses how those methods are developed and applied in state-of-the-art marine trajectory clustering methods. Trajectory clustering has attracted growing attention, considering the critical role of trajectory data mining in modern intelligent systems for surveillance, security, abnormal behavior

detection, crowd behavior analysis, and traffic control (Bian *et al.*, 2018). The details of some existing trajectory clustering methods are presented in **Table 2–1**, categorized into three groups: 1) supervised, 2) unsupervised, and 3) semi-supervised algorithms.

Table 2–1. Marine Trajectory Data Clustering Related Works

Marine Trajectory Data Clustering	Details	Reference
Supervised Methods	**Nearest Neighbor algorithm** (e.g., KNN): finds a voting system to determine a new entity's category. In trajectory clustering, the algorithms calculate the distances from an inquiry trajectory to all labeled trajectory data and label the inquiry trajectory by most of its k nearest neighbors (Gao *et al.*, 2007).	(Gao etc. 2007)
	Support Vector Machine (SVM): generates a hypervolume, separate the outliers from the valid trajectories. However, as a binary classifier, SVM has challenges in grouping trajectories into sub-clusters (Piciarelli *et al.*, 2008).	(Piciarelli, 2008)
	Artificial Neural Network: constructs many layers of connected neurons, represented by a regression model. Since in most cases, Neural Network is used for data classification, neural networks cluster trajectories through classifying observations into the pre-defined or pre-labeled clusters (Cho and Chen, 2014).	(Cho, 2014)
Unsupervised Methods	**Hierarchical clustering models**: tree-structured models that consider different attributes at each level (Li *et al.*, 2006).	(Li, 2006)
	Spectral Clustering models: models representing trajectory data as affinity matrixes, then compute internal relationships by analyzing these affinity matrices (Xiang and Gong, 2008).	(Xiang, 2008)

	Densely clustering models: clusters trajectories by considering the spatial density with respect to distance metrics. A framework was proposed for partitioning and grouping trajectories close to each other (Lee, Han and Whang, 2007). K-means methods divide data into k clusters but are challenging to be implemented in actual data since the k value will never be well-known in real-world problems (Galluccio *et al.*, no date; Ferreira *et al.*, 2012). DBSCAN clusters point together, which are "density reachable."	(Lee, 2007) (Ferreira, 2013) (Galluccio, 2012)
Semi-Supervised Methods	Humans prepare a small number of pre-defined clusters, and the new observations are clustered to automatically update the classifier (Laxhammar and Falkman, 2014).	(Laxhammar, 2012)

As mentioned in the previous section 2.1, supervised algorithms learn from labeled training data to train a model, which can determine the labels of new observations (Kanevski *et al.*, 2009). Therefore, supervised algorithms perform tasks based on understanding the "ground truth." K-Nearest Neighbors (KNN) is one of the most commonly used supervised learning methods that can be applied in trajectory clustering. KNN determines the label of a new trajectory segment by finding the k nearest trajectory segments. In the implementation, trajectory data are usually represented by Minimum Bounding Rectangles (MBR) and indexed in a tree structure such as a Trajectory-Bundle Tree (TB Tree). **Figure 2–5** presents an example of MBR representation and TB tree structure (Gao *et al.*, 2007).

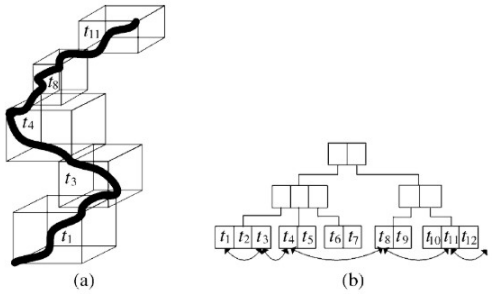


Figure 2–5. An example of MBR representation and Trajectory-Bundle Tree (TB Tree) structure for trajectory segment from (Gao *et al.*, 2007)

It was proposed how distances between trajectory segments to MBRs can be calculated (Gao *et al.*, 2007), and based on that, they proposed KNN algorithms based on the best-first traversal paradigm for retrieving historical trajectories. However, KNN and other Nearest Neighbor algorithms' drawback is that only the spatial relationships between a pair of trajectory data are considered, but local characters are ignored.

Another commonly used supervised learning method that can be applied in trajectory clustering is the Support Vector Machine (SVM). SVM is trained to generate the hypervolume, separating the outliers from the valid trajectories. For example, (Piciarelli *et al.*, 2008) proposed an approach based on single-class SVM clustering, where the novelty detection SVM capabilities are used for the identification of anomalous trajectories, even in the absence of a priori information on the distribution of outliers. However, as a binary classifier, SVM has difficulties explicitly labeling or grouping trajectories into sub-clusters, especially large datasets.

Artificial Neural Networks (ANN) are another widely used supervised type of algorithms. The network is constructed by many layers of connected neurons, represented by a regression model. Since, in most cases, ANN is used for data classification, neural networks usually cluster trajectories through classifying new observations into the pre-defined or pre-labeled clusters (Ilango and Mohan, 2010). In recent decades, with the advancement of ANN development, ANN starts to play crucial roles in trajectory mining. Convolutional Neural Network (CNN, or ConvNet) is a popular approach to use. CNN is a special artificial neural network that consists of convolutional and pooling layers, respectively. CNN is widely used in computer vision due to its efficiency in processing image data. **Figure 2–6** shows an example of a typical CNN structure on image classification.

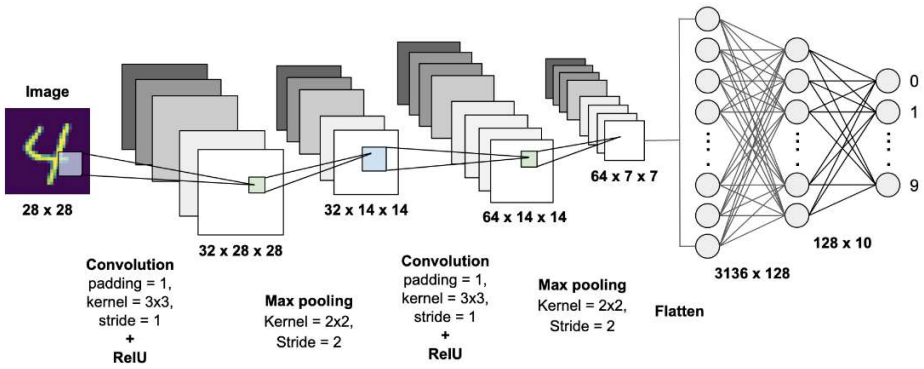


Figure 2–6. An example of a typical CNN structure on image classification (credit: medium.com user Meghna Asthana)

As CNN developed further, they have been applied to processing streaming video data and used for recognizing moving objects. For example, (Cho and Chen, 2014) studied

the problem of segmenting independently moving objects in a flow field. Those research works provide crucial supports for obtaining and preparing the trajectory segments from video sources.

As mentioned in Section 2.1, unsupervised learning is a type of self-organized learning that helps find previously unknown data set patterns without prior knowledge. Unsupervised algorithms can find hidden structures and unknown dependencies through clustering data with similarities. One advantage of the unsupervised algorithm is that extensive human effort in preparing the training data is not required, while unsupervised algorithms usually have lower accuracy without a training process. One of the most commonly used unsupervised algorithms is the K-Means algorithm, which divides 'n' data objects into 'k' number of clusters. The use of vector fields has been proposed in (Ferreira *et al.*, 2012) to represent trajectory datasets and applied a K-means algorithm to partition the vector fields into k clusters. K-means methods are difficult to be implemented in authentic data since the k value is difficult to be well-known on real-world problems. (Galluccio *et al.*, no date) proposed a graph-based method for estimating the number of clusters present and determining good centroid locations to initialize the K-means algorithm. The hierarchical clustering model is a tree-structured model that considers more attributes at each level (Li *et al.*, 2006).

The hierarchical clustering model is a tree-structured model that connects objects nearby or partition the large cluster into sub-clusters by applying different criteria in various layers. A coarse-to-fine strategy to cluster vehicle trajectory data was proposed (Li *et al.*,

2006). Applying the divisive approach, (Li *et al.*, 2006) keeps dividing the entire trajectory dataset into smaller and smaller sub-clusters until the intra-cluster tightness of each sub-cluster reaches a defined threshold.

Spectral Clustering models represent trajectory data as an affinity matrix, then compute internal relationships by analyzing these affinity matrices. It was proposed to use eigenvector selection to improve spectral clustering results by measuring the relevance of an eigenvector according to how well it can separate the data set into different clusters (Zaki *et al.*, 1997).

Densely clustering models classify trajectories by considering the spatial information calculated by distance metrics. Density-based clustering algorithms can be the most suitable method for this research because density-based clustering algorithms can find clusters' arbitrary shapes. A framework for partitioning and grouping trajectories close to each other was proposed (Ankerst *et al.*, 1999). It used a Minimum Description Length (MDL) principle to partition the trajectories and used a density-based line-segment clustering algorithm to find common sub-trajectories (Ankerst *et al.*, 1999). DBSCAN clustering algorithms cluster data together by the concept of 'density reachability,' which means the points spatially connect to each other are to be clustered together (Ester *et al.*, 1996). This research adopts DBSCAN as a foundation of the model for clustering marine transportation trajectories. The details of the DBSCAN clustering method will be discussed in Section 2.3.

Semi-supervised algorithms fall between unsupervised algorithms and supervised algorithms. Compared to supervised learning, semi-supervised algorithms require much

smaller human effort to prepare training data, while comparing to the unsupervised ones, semi-supervised models usually have better performance regarding accuracy. Some semi-supervised algorithms can be invented, starting from unsupervised or supervised algorithms. For example, the algorithms can only require users to prepare a small amount of labeled data to train the model then conduct the cluster tasks while updating the model with unlabeled data automatically (Laxhammar and Falkman, 2014). In this way, semi-supervised algorithms can be more efficient methods, combining the advantages of supervised and unsupervised algorithms. This thesis proposes a semi-supervised method to be applied to the trajectory clustering in real-world problems. This work starts with optimizing an unsupervised algorithm, DBSCAN, then modifies it into a semi-supervised model. The model can work in an unsupervised way and input labeled data to speed up, sending unlabeled observations to the model to update the model.

2.3 DBSCAN Enhancement

Since the proposed density-based clustering algorithm integrating with the Mahalanobis distance metric is closely related to DBSCAN, DBSCAN is briefly introduced in this section, including the development of DBSCAN optimizations. DBSCAN discovers clusters and outliers for a spatial dataset (Ester *et al.*, 1996). It defines clusters as maximum sets of density-connected data points, in which every core point in a cluster must have at least a minimum number of data points (*MinPts*) within a neighbor of a given radius (*Eps*). As shown in **Figure 2–7**, DBSCAN iterates through every point to grow the clusters until all

points are visited, and the unlabeled points left will be labeled as outliers. After DBSCAN clustering, all data points within one cluster can be reached from one to another by traversing a path of density-connected data points while the data points across different clusters cannot. DBSCAN can find arbitrarily shaped clusters, showing potentials for marine trajectory clustering. The complexity of traditional DBSCAN can be $O(n^2)$ without the use of any indexing to accelerate the computation. The overall average runtime complexity can be reduced to $O(n * \log(n))$ if an indexing structure is used for executing neighborhood queries.

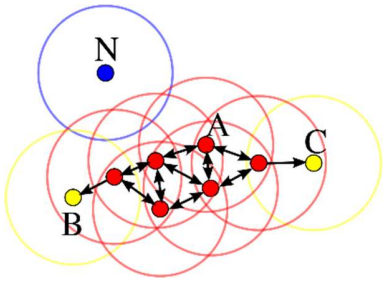


Figure 2–7. Example of the DBSCAN process. A is the core points; B, C are the border points; N is an outlier (credit: Wikipedia User Chire)

However, due to the drawbacks of the DBSCAN clustering method, optimizations are required before implementation. For example, traditional DBSCAN is very sensitive to the two parameters (*MinPts* and *Eps*) selected by the user. Even a slightly different set of them may lead to very different partitions of the dataset (Ren, Liu and Liu, 2012; Fong *et al.*, 2014; Schubert *et al.*, 2017). Usually, the users need to get the optimal parameters from a long and repetitive trial-and-error process. However, in real life, the optimal parameters are very

hard to find when the data and scale cannot be well understood (Academy *et al.*, 2009; Esmaelnejad, Habibi and Yeganeh, 2010; Smiti and Eloudi, 2013; Fong *et al.*, 2014; Karami and Johansson, 2014; Sawant, 2014; Liu, 2015; Hou, Gao and Li, 2016; Schubert *et al.*, 2017; Han, Armenakis and Jadidi, 2020). Besides, applying the traditional DBSCAN clustering method has a considerable shortcoming with unevenly distributed data, that some data are densely concentrated at several locations while other data are sparsely distributed. Unevenly distributed data are challenging to be clustered ideally with a single designated *Eps* (Academy *et al.*, 2009; Esmaelnejad, Habibi and Yeganeh, 2010; Smiti and Eloudi, 2013; Fong *et al.*, 2014; Karami and Johansson, 2014; Sawant, 2014; Liu, 2015; Hou, Gao and Li, 2016; Schubert *et al.*, 2017; Han, Armenakis and Jadidi, 2020), making real AIS data unreliable to be used for applying traditional DBSCAN to it without optimization. Furthermore, as the traditional DBSCAN is based on the Euclidean distance metric, this sometimes cannot handle data with complex shapes and distribution (Ren, Liu and Liu, 2012; Sangeetha, Padikkaramu and Chellan, 2018). Thus, novel distance metrics need to be proposed to optimize the DBSCAN performance.

As stated in **Table 2–2**, multiple optimizations have been proposed to enhance DBSCAN's performance from the research community. Solutions to the drawbacks of clustering unevenly distributed datasets with varied densities have been proposed (Xiaopeng Yu, Deyi Zhou and Yan Zhou, 2005; Uncu *et al.*, 2006; Borah and Bhattacharyya, 2007; Peng, Dong and Naijun, 2007; Ram *et al.*, 2009, 2010; Elbatta, 2012). Methods to find optimal parameters suitable for corresponding datasets have been proposed (Academy *et*

al., 2009; Esmaelnejad, Habibi and Yeganeh, 2010; Karami and Johansson, 2014; Sawant, 2014). Density clustering methods are proposed without requiring any parameters from the user (Fahim and Salem, 2006; Hou, Gao and Li, 2016). Various ways are proposed to increase the algorithm's computational efficiency when applying to large databases (Borah and Bhattacharyya, 2004; El-Sonbaty, Ismail and Farouk, 2004; Liu, 2006; Mahran and Mahar, 2008; Xiaoyun *et al.*, 2008). Various methods also bring new clustering conceptions to DBSCAN (Birant and Kut, 2007; Ren, Liu and Liu, 2012; Smiti and Eloudi, 2013; Sangeetha, Padikkaramu and Chellan, 2018).

Table 2–2. Current State-of-the-Art of the DBSCAN Enhancement Methods

DBSCAN Enhancement Features	Details	Author and Year
Clustering Uneven Dataset Efficiently Varied in Density	**VDBSCAN**: Varied Density-Based Spatial Clustering of Applications with Noise, by selecting several values of parameter *Eps* for different densities according to a k-dist plot	(Liu, 2007)
	A New Clustering Algorithm Based on Distance and Density, based on merging KNN and DBSCAN to enhance DBSCAN	(Yu, 2005)
	GRId Density-Based Spatial Clustering of Applications with Noise: selects appropriate grids, merges cells with similar densities, and identifies the most suitable values of *Eps* and *minPts* in each grid	(Uncu, 2006)
	Limited the amount of allowed local density variation to achieve better results **DVBSCAN**: A Density-based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases	(Ram, 2009), (Ram, 2010)

	Identification of Noise objects from a cluster with different densities	(Birant, 2007)
Help Finding the Optimal Parameters	**DMDBSCAN**: Selection of several *Eps* from the k-dist plot and then use of the dynamic method to find a suitable value	(Mohammad, 2012)
	A Novel Method to Find Appropriate ε for DBSCAN, remove the ε and replace it with another parameter named ϱ (Noise ratio of the data set)	(Esmaelnejad, 2010)
	Use Differential Evolution	(Karami, 2014)
	Adaptive Methods for Determining DBSCAN Parameters to determine *Eps* by the value of 'k.'	(Sawant, 2014)
	SA – DBSCAN, via analysis of the statistical characteristics of the dataset	(Xia, 2009)
Novel Clustering Conceptions	**ST-DBSCAN**: Discovers clusters concerning spatial, non-spatial, and temporal values of the objects. Discovering cluster on spatial-temporal data.	(Birant, 2007)
	DSets-DBSCAN: A Parameter-Free Clustering Algorithm	(Hou, 2016),
	DCBRD: Density Clustering based on radius of data, without parameters	(Fahim, 2006)
	Soft DBSCAN: Improving DBSCAN clustering method using fuzzy set theory	(Smiti, 2013)
Memory efficiency and I/O cost minimization	Introduction of Kernel function to make clustering more accurate and faster	(Liu, 2006)
	GMDBSCAN: Multi-Density DBSCAN Cluster Based on Grid	(Chen, 2008),
	An Improved Sampling-Based DBSCAN for Large Spatial Databases	(Borah, 2004),

	An Efficient Density-Based Clustering Algorithm for Large Databases, by partitioning and merging the datasets	(EI-Sonbaty, 2004)
Novel distance metrics	Incorporating Mahalanobis Distance by defining 'leaders' and 'followers' points	(Yan, 2012)
		(Sangeetha, 2018)

This research presents a method based on enhancing the DBSCAN clustering referencing to the literature review. The majority of the existing optimizations are designed for clustering 2-D spatial data (i.e., x, y). When the data dimension is growing and the Mahalanobis distance metric is used, the distribution of the dataset becomes different (Ren, Liu and Liu, 2012; Sangeetha, Padikkaramu and Chellan, 2018; Han, Armenakis and Jadidi, 2020). Therefore, the existing adaptive parameter method needs to be modified to be applied to the enhanced DBSCAN method. Thus, a suitable optimization is required to apply on high-dimensional DBSCAN clustering using an intuitive distance metric such as the Mahalanobis distance matrix. We integrate the Mahalanobis Distance metric into DBSCAN to enhance the DBSCAN clustering performance by considering the correlations. Besides, an automatic and data-driven approach is proposed to choose the required initial two parameters (*MinPts* and *Eps*) for enhanced DBSCAN.

2.4 Mahalanobis Distance

Since the proposed density-based clustering algorithm integrates the Mahalanobis distance metric with DBSCAN, in this section, Mahalanobis distance is described and compared to Euclidean distance. The Mahalanobis distance was first proposed by the Indian statistician P. C. Mahalanobis in 1936 (Mahalanobis, P.C., 1936). It calculates the distance between a multivariate vector data **x** and a distribution **C** or distance between two random vectors **x** and **y** of the same distribution. The Mahalanobis distance **DM(x, y)** from a point data **x**, to another point data **y**, which both are inside a cluster with the covariance matrix, **S**, is defined by Eqs. (1). Eq. (3) is the mean vector of **x** and Eq. (4) is the covariance matrix of the dimensions $\mathbf{x_i}$ and $\mathbf{x_j}$. The Mahalanobis distance **DM(x, C)** from a point data, **x**, to the cluster **C** with mean, $\boldsymbol{\mu}$, is defined by Eq (2). In the following equations, the dimensions of vectors **x**, **y** and $\boldsymbol{\mu}$ are 5×1, respectively. The dimensions of covariance matrix **S** and its inverse matrix $\mathbf{S^{-1}}$, are 5×5, respectively. The dimensions of Mahalanobis distances **DM(x, y)** and **DM(x, C)** are 1×1, respectively.

$$D_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{S}^{-1} (\boldsymbol{x} - \boldsymbol{y})} \quad (1)$$

$$D_M(\boldsymbol{x}, \boldsymbol{C}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{S}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})} \quad (2)$$

$$\boldsymbol{\mu} = \left[\mu_{latitude}, \mu_{longitude}, \mu_{SOG}, \mu_{COG}, \mu_{Heading}\right]^T \quad (3)$$

$$\boldsymbol{S_{ij}} = cov\left(x_i, x_j\right) = \langle (x_i - \mu_i)(x_j - \mu_j) \rangle \quad (4)$$

On the other hand, the Euclidean distance only calculates distance between two points in Euclidean space and does not consider the cluster's distribution. The difference between Euclidean distance and Mahalanobis distance is demonstrated by **Figure 2–8**. When calculating the Euclidean distance between the cluster centroid to Point 1 and Point 2, line segments' lengths (in pink and purple) are measured. Point 1 and Point 2 have the same Euclidean Distance to the centroid, but Point 2 has a much longer Mahalanobis Distance to this cluster because Point 2 is correlated to the cluster.

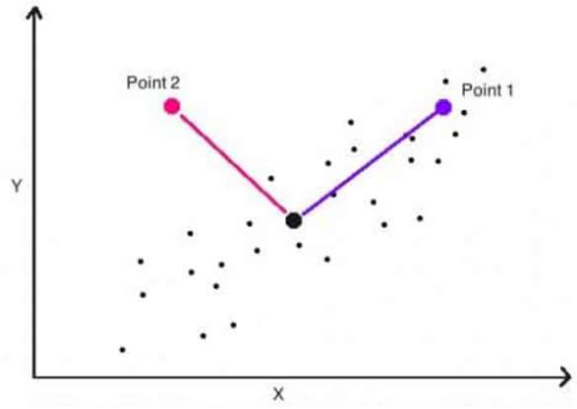


Figure 2–8. Differences between Euclidean distance and Mahalanobis distance – Even Point 1 and Point 2 have the same Euclidean Distance to the centroid, but Point 2 has much longer Mahalanobis Distance to this cluster (source: machinelearningplus.com user Selva Prabhakaran)

The Mahalanobis distance can be reduced to the same as Euclidean distance if the covariance matrix S is the identity matrix. In this case, all attributes in their specific dimensions are totally independent and uncorrelated. The corresponding equation is shown by Equ (5).

$$D_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{S}^{-1} (\boldsymbol{x} - \boldsymbol{y})} = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^2} = ED(\boldsymbol{x}, \boldsymbol{y}) \qquad (5)$$

The Mahalanobis distance, as one of the most common measures in multivariate statistics, is closely related to principal components analysis (PCA), another very common statistical procedure. Statistically, PCA finds lines and planes in the multi-dimensional space that best approximate the data regarding least-squares of residuals. The procedure is shown in **Figure 2–9**. PCA places the observations in the high-dimensional variable space and centers the mean to zero by subtracting the variable averages. The scores of the first principal component (PC1), t1, are calculated by finding the line that best approximates the data in the least-squares sense and projecting the data onto this line to get a coordinate value along the PC-line. A second principal component (PC2) is calculated by finding another PC-line orthogonal to the PC1. Then the scores of PC2, t_2, are also calculated by projecting the observations onto PC2-line and get coordinates. **Figure 2–9** graphically visualizes the data set structure by creating two-dimensional PCs to investigate data set in higher dimensions. PCA is commonly used in exploratory data analysis and for making predictive models.

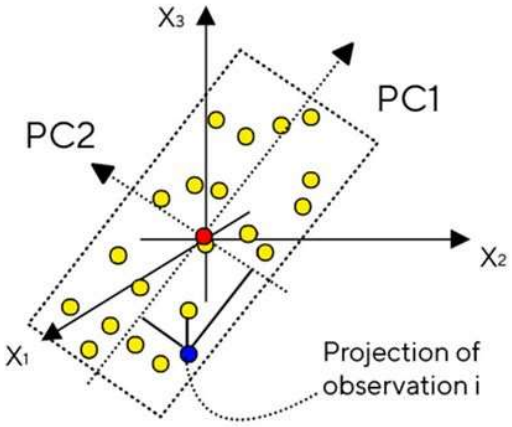


Figure 2–9. Principal components analysis (PCA) on forming two PCs as a window into the multidimensional space (source: sartorius.com)

PCA procedures share similarities with Mahalanobis distance, as the squared Mahalanobis distance is equal to the sum of squares of the scores of all non-zero standardized principal components. In the k dimension, the corresponding equation is shown below by Eq. (6), where t_i represents the score of the standardized principal component in dimension i.

$$D_M^2 = t_1^2 + t_2^2 \ldots + t_k^2 \qquad (6)$$

The most common use for the Mahalanobis distance is to find multivariate outliers, which indicates unusual combinations of variables in corresponding dimensions. **Figure 2–10** shows an example of Mahalanobis distance contour plot of 100 random draws from a bivariate normal distribution. The Mahalanobis distance can be used to determine whether

a sample is an outlier. For example, indicated by **Figure 2–10**, if *Eps* (distance threshold to be an outlier) is defined to be 0.02, both points in red square and the purple circle will be considered as outliers to this cluster.

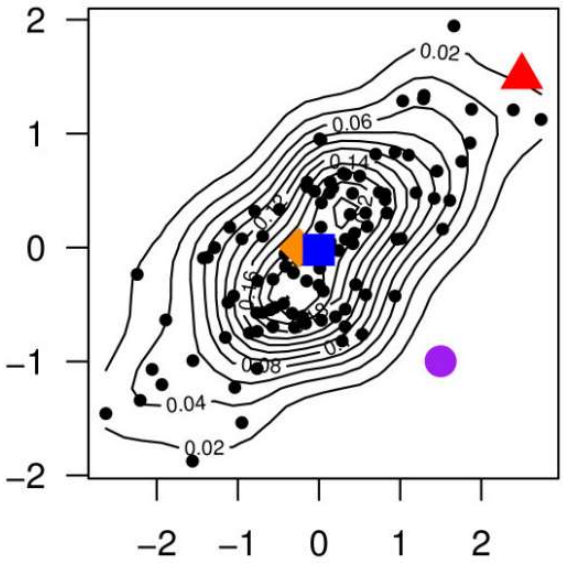


Figure 2–10. An example of Mahalanobis distance contour plot on 100 random points with mean zero, unit variance, and 50% correlation. A blue square notes the centroid defined by the marginal means. (source: statisticshowto.com)

Chapter 3: Methodology

This chapter provides details about the proposed algorithms and the clustering framework implementation in four steps. Section 3.1 describes a novel representation of marine trajectory data by increasing the dimensions of the vessel's positioning data by considering additional attributes such as velocity and direction in the clustering process, along with the geospatial information. Section 3.2 describes how the DBSCAN clustering method is enhanced by integrating the Mahalanobis Distance metric, taking into account the correlations of the position cluster points aiming to make a better identification process as well as reducing the computational cost. This section also gives detailed information about the definitions of the enhanced DBSCAN and pseudo-codes of the proposed algorithms. Section 3.3 describes a proposed method to select the parameter needed in the proposed algorithms automatically based on the data itself. Section 3.4 describes the designed frameworks implementing the algorithms on big data. This section presents three frameworks in three layers by which patterns can profile the vessel behaviors through finding clusters within historical data.

3.1 Novel Representation of Marine Trajectory Data

The traditional densely based clustering works with two-dimensional data (i.e., location data). Latitude and longitude are the only spatial components to be considered, and the 2-dimensional points are clustered together based on their spatial density.

Increasing the data dimensions can change the concept of "density reachability" and enhance the clustering model abilities to find more complex unknown similarities between the data rather than based on latitude and longitude.

In this research, it is proposed to extend each trajectory 2D point data record into a five-dimensional vector, as shown at Eq. (7), Latitude, Longitude, also taking into account Speed over Ground (SOG), Course over Ground (COG), and Heading. SOG is defined by the actual speed at which the GPS unit is moving over the ground. COG describes motion direction with respect to the ground that a vessel has moved relative to the magnetic north pole or geographic north pole. Heading describes the direction that a vessel is pointed at any time relative to the magnetic north pole or geographic north pole. The difference between Heading and COG is presented in **Figure 3–1.** Besides Latitude and Longitude describing the vessels' geographic location, Speed over Ground (SOG), Course over Ground (COG), and Heading are also essential parameters to describe vessels behaviors. Considering SOG, COG, and Heading when clustering AIS data can bring deeper insights on marine transportation from the clustering results.

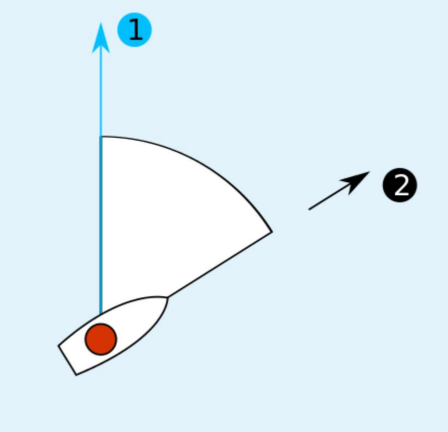


Figure 3–1. Difference between COG, the direction of motion with respect to the ground, ①, and Heading, the direction that a vessel is pointed at, ② (credit: Wikipedia user WolfgangW)

A marine transportation trajectory is defined as a finite sequence $\boldsymbol{T} = ((\boldsymbol{x_1}, t_1), (\boldsymbol{x_2}, t_2), \dots, (\boldsymbol{x_m}, t_m))$. Each data point $\boldsymbol{x_i}$ corresponds to a multi-dimensional feature vector representing the moving object by a set of $[latitude, longitude, SOG, COG, Heading]$ at time point t_i, where $t_i < t_{i+1}$ for i =1, ..., m-1.

$$\boldsymbol{x} = [latitude,\ longitude,\ SOG,\ COG,\ Heading]^T \quad (7)$$

$$\boldsymbol{T} = ((\boldsymbol{x_1}, t_1),\ (\boldsymbol{x_2}, t_2), \dots, (\boldsymbol{x_m}, t_m)) \quad (8)$$

For the sake of consistency and standardization, the data is normalized between [-1, 1] as required by most machine learning techniques, including DBSCAN. By obtaining the z-score of each attribute of each data, this normalization process is done by Eq. (9). After

normalization, all five attributes share the same mean value, $\boldsymbol{\mu}$, the same variance value, σ, and the same weight when clustering.

$$Z = \frac{\boldsymbol{x}-\boldsymbol{\mu}}{\sigma} \quad (9)$$

3.2 Integration of Mahalanobis Metric to DBSCAN

As mentioned in Section 2.3, traditional DBSCAN clustering iterates from point to point, calculate the distances among points, identify core points and cluster the surrounding points together. The traditional DBSCAN using Euclidean Distance has two main shortcomings: 1) high computation costs, and 2) only local characteristics are taken into account when identifying the cluster. The proposed clustering method, integrated with the Mahalanobis Distance metric, resolves the previously mentioned challenges by increasing the computational efficiency and considering the correlation between the point within the cluster.

When the traditional DBSCAN needs to identify if a point is an outlier, it finds the points around it and calculates the Euclidean distance to them; then it calculates the number of points within defined *Eps* and compares it to defined *MinPts* to determine if the point is a core point, a border point, or an outlier. The computational cost is high, even though with the help of spatial indexing. Furthermore, only some of the points around the target point play a role in defining the target point, but not the whole cluster. Integrating Mahalanobis distance enhances clustering performance on the unevenly distributed

dataset. As shown in **Figure 3–2**, when determining whether Point A and Point B belong to the cluster, the Mahalanobis distance is calculated and compared to defined *Eps*. Only one step of the calculation is required, and the correlation between the point to the whole cluster has been considered.

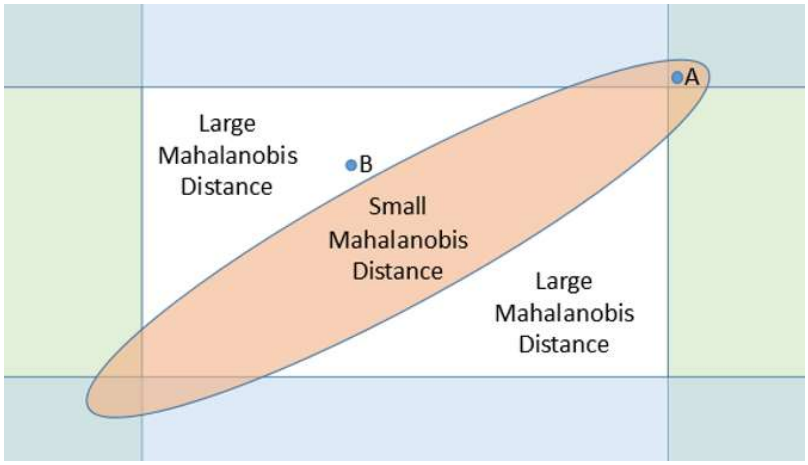


Figure 3–2. A point classification process using Mahalanobis Distance – even Point A has a longer Euclidean Distance to the centroid, but Point B has a much longer Mahalanobis Distance to this cluster (credit: Rick Wicklin on The DO Loop)

The corresponding definitions for implementing the proposed enhanced DBSCAN approach are given as follows.

Definition 1: Neighbourhoods is defined through 1.1 and 1.2 Definitions.

Definition 1.1: Neighborhoods of points

Given a group D of moving trajectory points, a point q is defined as a neighborhood of the point **p** when $\{\mathbf{q} \in D \mid D_M(\mathbf{p}, \mathbf{q}) \leq Eps\}$ is satisfied.

Definition 1.2: Neighborhoods of Pre-defined Clusters

Given a group D of moving trajectory points, the neighborhoods of a Pre-defined Cluster C, which at least has two points, are defined by {**q** ∈ D|DM (**q**, C) ≤ *Eps*}.

Definition 2: Core point

A core point contains at least a minimum number (*MinPts*) of neighborhoods.

Definition 3: Density-reachable is described through 3.1 and 3.2 Definitions.

Definition 3.1: Density-reachable to points

Given a database D of moving trajectory points, a point **p** is density-reachable from the point **q** with respect to *Eps* and *MinPts* if there is a chain of points **p1, …, pn**, **p1** = **p** and **pn** = **q** such that **pi+1** is within the *Eps*-neighborhood of **pi**, for 1 ≤ i ≤ n, **pi** ∈ D.

Definition 3.2: Density-Reachable to Pre-defined Clusters

A point **p** is directly density-reachable from a Pre-defined Cluster **C**, if **p** is within the *Eps*-neighborhood of **C**

Definition 4: Density-Based Cluster

A Density-Based Cluster **C** is a non-empty subset of D satisfying the following "maximality" and "connectivity" requirements:

∀**p**: if **p** is Density-Reachable from **C** with respect to *Eps*, then **p** ∈ **C**.

∀**p** ∈ **C**: **p** is Density-Reachable to **C** with respect to *Eps*

∀**q**, **p** ∈ **C**: **p** is Density-Reachable to **q**

Definition 5: Outlier

Given a database D of moving trajectory points and Clusters **C1, C2, …, Ck**, a point **p** is an outlier if **p** is not belonging to any cluster **Ci**

As discussed in Section 2.3, the key idea of DBSCAN is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Here in this thesis, we adopt this idea and consider three other attributes, SOG, COG, and

Heading. The intuition behind this is that the neighbors of a trajectory point should be geospatially near enough and with similar speed and traveling direction. Thus, we can modify the original definition of *Eps*-neighborhood in traditional DBSCAN to **Definition 1.1**. Note that **DM**(**p**, **q**) is the Mahalanobis distance between **p** and **q** instead of Euclidean distance because it is necessary to consider the correlation of the whole cluster in five dimensions to calculate distances.

We also innovated **Definition 1.2** to describe the relationship between points and clusters. Unlike the traditional DBSCAN, which defines Neighborhoods of Clusters by density-reachable to a core point, **Definition 1.2** gives a direct way to determine relationships between points and clusters by taking advantage of the Mahalanobis distance metric. In the following, we also give the formal definitions of other essential notions for our density-based algorithm.

From **Definition 2**, we can determine whether point **q** is a core trajectory point according to **Definition 1.1**. Because if the number of Neighborhoods of points is larger than *MinPts*, **q** is a core trajectory point.

By combining **Definition 1.1, Definition 2,** and **Definition 3.1**, we can start to grow the clusters using the unsupervised component. If the point **q** is a Neighborhood of the point **p**, and the point **p** is a core point that belongs to a cluster, then the point **q** can be identified to belong to the same cluster as the core point **p**. Then the number of neighborhoods of the point **q** is calculated to compare with *MinPts*. If the point **q** is a core trajectory point, the cluster growing process continues. If the point **q** is not a core trajectory point, the point **p**

is called Border Trajectory Point, and its neighborhoods cannot be grown into the cluster from this route. Also, we innovated the definition of **Definition 3.2** to grow the clusters.

Unlike **Definition 3.1**, which grows the clusters through Neighborhoods of the core points, **Definition 3.2** gives a direct way to grow the clusters by calculating the Mahalanobis distance metric between points and clusters. The *Eps*-Neighborhood of the cluster will be identified as part of the cluster.

So, summarizing the definitions 1-3, it is necessary to define Density-Based Cluster as **Definition 4** based on the relations above (*Eps*-Neighborhood and Density-reachable). We conclude that any points belong to a cluster will either be *Eps*-Neighborhoods to the cluster or one or more of its core points. Any points that belong to the cluster should either be *Eps*-Neighborhoods to each other or Density-reachable to each other. Any other points which do not belong to any clusters are considered outliers.

The corresponding algorithms for implementing the proposed enhanced DBSCAN approach are given as follows.

Algorithm 1 Proposed DBSCAN Unsupervised (D, *Eps*, *MinPts*):

```
1. Select the two parameters Eps and MinPts;
2. Mark all the points in the dataset as unclassified and set C = 0;
3. for each point p in the Dataset D:
4.         If the point's label is not 0:
5.                 continue to the next point;
6.         Find all of P's neighboring points, regionQuery(D, P, Eps);
7.         If the number of neighboring points is below MinPts:
8.                 this point is noise;
9.         Else:
10.                C += 1;
```

11. use this point as the seed for a new cluster, growCluster(D, labels, P, NeighborPts, **C**, *Eps*, *MinPts*);
12. Return labels

Algorithm 1.1 regionQuery(D, P, *Eps*):

1. neighbors = [];
2. For each point in the dataset:
3. If the Mahalanobis distance is below the threshold *Eps*:
4. add it to the neighbors list;
5. Return neighbors;

Algorithm 1.2 growCluster(D, labels, P, NeighborPts, C, *Eps*, *MinPts*):

1. Assign the cluster label, **C**, to the seed point, P;
2. for all neighbors of P, Pn:
3. If Pn labeled as noise:
4. change Pn label to C;
5. Elif Pn has no labels (0):
6. label Pn as C;
7. Find all the neighbors of Pn, PnNeighborPts = regionQuery(D, Pn, *Eps*);
8. If the number of neighboring points is above *MinPts*:
9. NeighborPts = NeighborPts + PnNeighborPts

Algorithm 2 Proposed DBSCAN Supervised (D, Training_D, *Eps*):

1. Select the parameter *Eps*;
2. for each pre-defined cluster in the Training Dataset:
3. Calculate Inverse of the Covariance Matrix of the pre-defined cluster;
4. Calculate Mean matrix of the pre-defined cluster;
5. for each point p in the Dataset D:
6. Calculate Mahalanobis Distance to each pre-defined cluster;
7. Find the one with the closest Mahalanobis Distance;
8. If the distance is above *Eps:*
9. this point belongs to the pre-defined cluster (share the same label);
10. Else:
11. this point is an outlier;
12. Return labels

Algorithm 1 presents the procedure of using the unsupervised component to discover clusters. It requires three parameters as input, including the Dataset **D** and two predefined parameters, *Eps* and *MinPts*. As mentioned in the definitions above, *Eps* sets the neighborhoods' threshold, and *MinPts* limits the number of the core points. The algorithm iterative through every point (line 3) and then find its *Eps*-neighborhoods by **Algorithm 1.1** (line 6). If the $\mathrm{number\ of}$ *Eps*-neighborhoods is above *MinPts* (line 7), by definition, the point is a core point and then grow the cluster from this point by **Algorithm 1.2** (line 11). Then the procedure iterates until all the points have been visited and labeled. By the end, all the points density connected are clustered together, and others are identified as outliers. The complexity of **Algorithm 1** in the proposed DBSCAN can be $O(n * \log(n))$ with the use of indexing to accelerate the computation.

Algorithm 2 presents the procedure of using the supervised component to classify new observations into pre-defined clusters. It requires three parameters as input, including the new observations Dataset **D**, the training Dataset **Training_D**, and a pre-defined parameter *Eps*. As mentioned in the definitions above, *Eps* sets the neighborhoods of clusters' neighborhoods and finds the outliers. The algorithm iterative through every pre-defined cluster in the training dataset (line 2) and then memorize associated the Inverse of the Covariance Matrix (line 3) and the Mean matrix of the pre-defined clusters (line 4). The algorithm iterative through every point in the new observations Dataset **D** (line 5) and classifies each point by calculating the Mahalanobis distance to each cluster (line 6 and 7). Then the procedure iterates until all the points have been visited and labeled. By the end,

all the points close to pre-defined clusters are classified, and others are identified as outliers. The complexity of **Algorithm 2** in the proposed DBSCAN is $O(n)$ with the use of indexing to accelerate the computation.

In summary, the proposed algorithm is composed of two parts: an unsupervised clustering method and a supervised one. The unsupervised algorithm finds the density reachability of the points in the defined high dimensional space. The unsupervised algorithm component integrates the Mahalanobis Distance metric considering the correlation to the whole dataset. In this way, density-based clusters are generated by grouping similar trajectory points. The supervised algorithm component takes advantage of the pre-defined clusters generated from the previous step, and the user input *Eps* parameter by an auto-selection method will be mentioned in the next section. The preliminary model reads each point to classify them. The Mahalanobis distances to each pre-defined clustered are computed, and the distance is compared with the user input *Eps* term. If the Mahalanobis distance is smaller than *Eps*, the point can be identified to belong to the cluster and then update. If the Mahalanobis distance is greater than *Eps*, then this point is an outlier to this cluster. This step can run iteratively until no outliers are closer than *Eps* to all clusters.

3.3 Method for the Auto-Selection of the Enhanced DBSCAN Parameters

As mentioned before, DBSCAN requires users to input two parameters (*MinPts* and *Eps*), and the clustering results can be very sensitive to the parameters selection. In addition, in this work, the map-reduce method for handling big data problems is adopted. Thus, it is required to have a universal way to select the parameters so that the clustering results in the first layer from each MMSI can remain consistent. This research proposes a simple and straightforward way to obtain a good initial selection for the two parameters of the proposed enhanced DBSCAN method.

The detailed algorithm is given as follows:

Algorithm 3 Find_Eps_MinPts (D):

```
Set MinPts = max(10, int(0.001 * len(D))) ;
Calculate Covariance matrix of D;
Calculate Inverse of the Covariance matrix;
K = MinPts – 1;
for each point p in the Dataset D:
        get K nearest Mahalanobis Distance;
        save the value;
// Find “the last spike”
Eps = round(q3(kn_dists)+1.5*iqr(kn_dists), 3);
Eps = min(Eps, max(D))
Return Minpts, Eps
```

The proposed method finds *MinPts* by selecting 0.1% of the sample size. By this method, *MinPts* are usually around 30, concerning that most of the datasets of each MMSI

have around 30,000 trajectory points. The *MinPts* are required to be at least ten since some dataset's data sizes are too small to have a valid *MinPts* parameter. The *Eps* are calculated by the distribution of the k-nearest-neighbor distances of each data point. The method is popular for understanding how the data distributed before setting clustering parameters. Ideally speaking, multiple *Eps* should be selected corresponding to each distance level to form clusters.

Nevertheless, due to the limitation of the DBSCAN, we only choose the *Eps* by prioritizing to filter outliers. The upper limits of the KNN distribution are selected as *Eps*. Moreover, the upper limit is defined by the upper quartile sum and 1.5 times the Interquartile Range (IQR). The complexity of the proposed parameter auto-selection method can be $O(n * \log(n))$ with the use of indexing to accelerate the computation.

3.4 Implementation of the Clustering Framework

This section describes the designed frameworks that are capable of profiling vessel behaviors and detecting abnormal vessel trajectories. The frameworks are developed by applying the proposed clustering algorithm described in previous sections 3.1 – 3.3. **Figure 3–4** shows the overview to process historical AIS data to generate the model (represented by a pink square). By this framework, the vessel behaviors can be profiled by behavior patterns through finding clusters within historical data and generate the model.

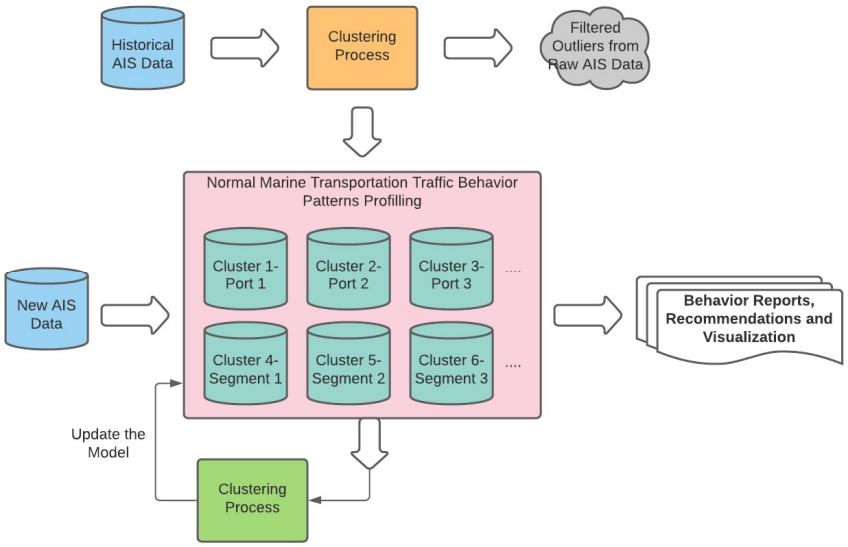


Figure 3–4. The Framework of extracting behavior patterns from actual AIS Data and applying the model to new observations

Behavior patterns of marine transportation traffic are profiled based on the clustering results. These clusters are used to model the AIS data within a certain region and monitor vessels installed with AIS equipment. The model then can be applied to new AIS observations to provide the desired outcome, including vessel behavior reports, action recommendations, and even behavior prediction. The model has monitoring purposes for crewed vessels and determining if the vessel has some anomaly behaviors. Autonomous vessels can also take advantage of the model for planning the route from selecting sequential clusters and getting recommendations for actions.

In order to apply the algorithms to big data, in this research, we adopt a map-reduce framework to cluster smaller pieces of AIS data divided from the raw data based on

Maritime Mobile Service Identity (MMSI), then merge them and generate the final classification model be used for reading the new observations. The method is based on the assumption that data under each MMSI, which has been used for identifying individual vessel trajectories, should have its specific behavior patterns.

The proposed first layer clustering algorithm groups similar trajectory points within each MMSI and define this specific trajectory stage. The second layer clustering merges the clusters from each MMSI data pieces, combining similar clusters and generating the final clusters. Each cluster from the first layer clustering has been profiled as a behavior vector to represent the cluster. The second layer cluster combines similar clusters by clustering similar behavior vectors. The same clustering algorithm proposed in section 3.2 is utilized in this step. The selections of the parameters differ from the method proposed in section 3.3. Instead of prioritizing filtering outliers, the parameter setting in this step prioritizes the merging of the most similar clusters. Therefore, the *Minpts* and *Eps* are manually adjusted to be much smaller than the recommendation value from the proposed parameter auto-selection method.

In this way, the algorithms run efficiently on processing big data. The details of the proposed hierarchy clustering structure are shown in **Figure 3–5**, representing the clustering process (orange square) in **Figure 3–4.**

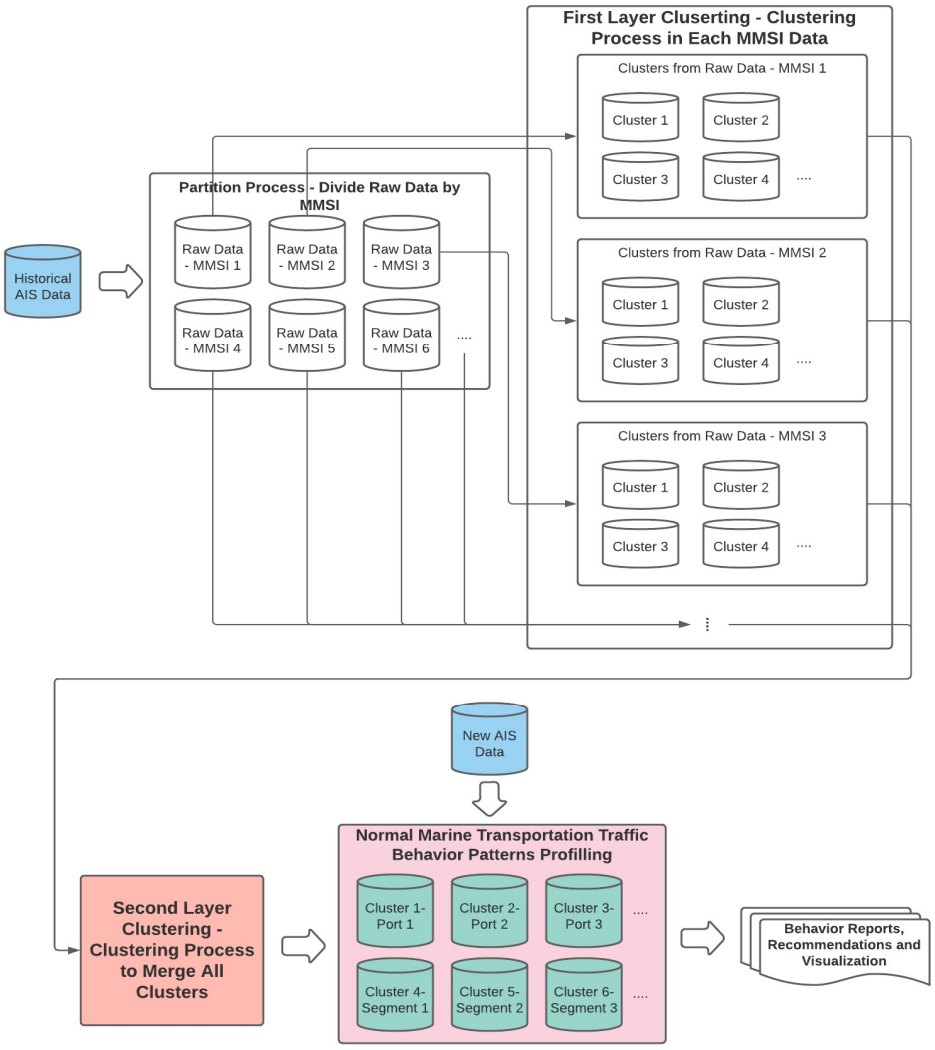


Figure 3–5. A Schematic overview of the clustering hierarchy – Segmentation of actual AIS data into smaller pieces and merging of the Clustering Results

Figure 3–6 presents the framework of clustering a dataset after integrating the Mahalanobis distance into DBSCAN, representing the first layer clustering in **Figure 3–5**. The raw data is firstly randomly split into two portions ensuring both datasets (Portion 1 and Portion 2) share the same distributions by setting the *stratify* parameter. Portion 1 is

required to contain at least 10,000 trajectory points so that the prepared training data is not biased due to overfitting. Usually, when the raw data has tremendous size, 5% of the raw data will have larger than 10,000 trajectory points, which is large enough to be selected to be used for preparing labeled training data.

The proposed clustering algorithm is a semi-supervised algorithm composed of an unsupervised clustering component and a supervised one. The algorithm can generate labeled data first in an unsupervised way with a smaller portion of the data. Then the pre-defined model from the last step reads the rest inquiry data and keeps updating itself. **Figure 3–6** shows that the unsupervised algorithm component is implemented on Portion 1 (in orange) to create pre-defined clusters. The supervised component then read Portion 2 (in green) to update the model into a final model. By this method, only a small amount of effort is allocated in the unsupervised step, which has highest runtime complexity in the whole clustering process. The two clustering steps are consistent with a similar approach, implementing the Mahalanobis Distance metric in the clustering process.

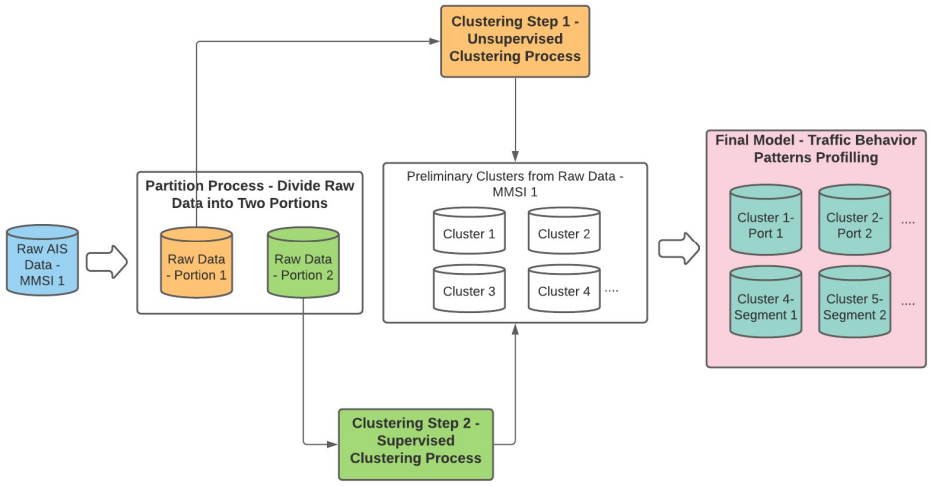


Figure 3–6. Details of the semi-supervised clustering process - a combination of an unsupervised clustering component and a supervised component

Figure 3–6 is related to the definitions and algorithms described in Section 3.2. **Definition 1.1** and **Definition 2** apply to clustering step 1 in **Figure 3–6** when implementing the unsupervised component. **Definition 1.2** describes the relationship between points and clusters, and Definition 3.2 which grows the clusters, applies to clustering step 2 in **Figure 3–6** when implementing the supervised component.

Chapter 4: Data and Experiments

This section presents details about the experiments implementing the proposed algorithms into actual AIS data and validations of the results. Section 4.1 describes the AIS data characteristics and the data sources, and how the synthetic data is processed and generated for testing. Section 4.2 describes the process of evaluating and validating the proposed algorithms. This section describes what evaluation metrics are selected, how four separated tests are designed and displays the results to validate the enhanced DBSCAN clustering performance. A sensitivity analysis is designed for validating the parameter auto-selecting method, which is also described in the section. Section 4.3 describes the results of two case studies where the proposed algorithm has been applied using two big datasets and shows examples of the application of the proposed algorithms and frameworks.

4.1 Data and Data Pre-Processing

Reliable open-sourced data sources for studying vessel behaviors and generating nautical routes such as the historical and real-time Automatic Identification System (AIS) data (Silveira, Teixeira and Soares, 2013; Sheng and Yin, 2018) have ben used for the implementation, testing and validation of the proposed approach. AIS is an automatic tracking system to identify and locate vessels by exchanging data with other nearby ships, AIS base stations, and satellites. According to the Safety of Life at Sea (SOLAS) convention, ships of 300 gross tonnages and upwards in international voyages, 500 and upwards for

cargoes not in international waters, and passenger's vessels are obliged to be embedded with AIS equipment, making AIS data abundant globally (IMO, 2000). Furthermore, AIS becomes a worldwide data standard, and therefore this coherent source of information can be suitable for global marine transportation traffic modeling and analysis.

AIS data is considered the raw data source of marine transportation, as AIS data is abundant and coherent globally. AIS contains 27 message types defined in ITU (International Telecommunication Union) recommendation M.1371-4, and two classes of shipboard equipment: class A (used mainly by commercial vessels) and class B (used mainly by fishing vessels and pleasure craft). Among 27 message types and two classes, data of Class A and Message type 1, 2, and 3 is suitable for this research because it contains the desired attributes including date/time; Maritime Mobile Service Identity (MMSI); speed over ground (SOG); latitude, longitude; course over ground (COG) and heading. **Figure 4–1** shows the overview of parsed AIS data.

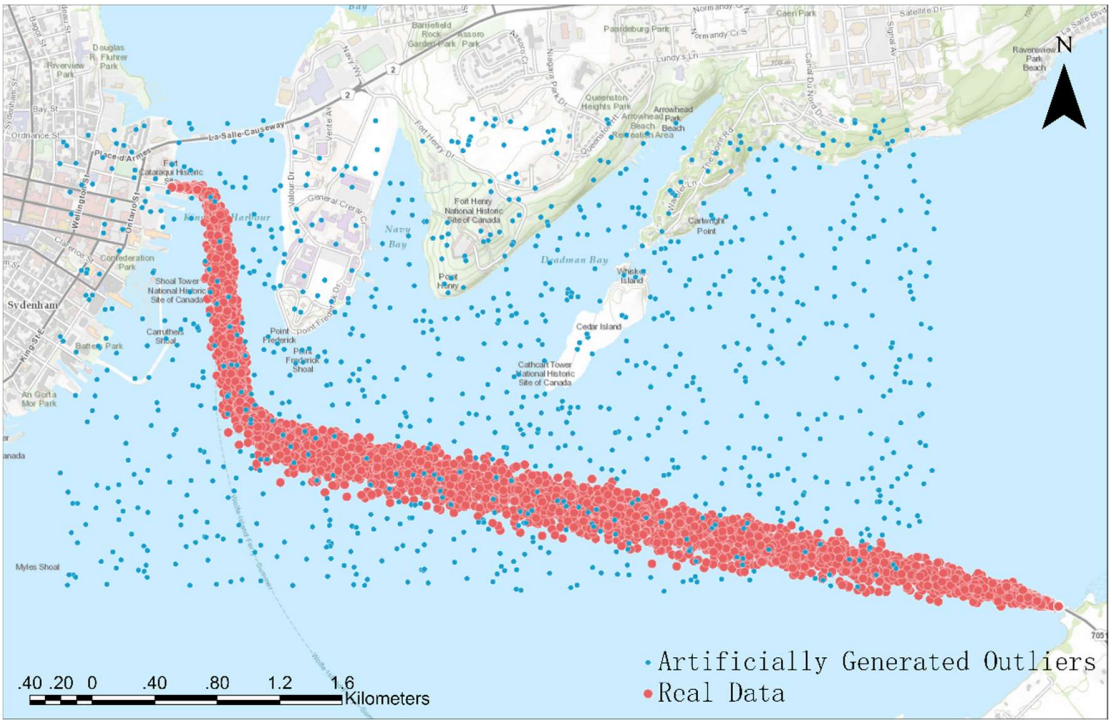
MMSI	BaseDate	LAT	LON	SOG	COG	Heading	VesselNa	IMO	CallSign	VesselTyp	Status	Length	Width	Draft	Cargo
366940480	2017-01-0	52.4873	-174.023	10	-140.7	267	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.48718	-174.028	10	-141.6	266	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.48705	-174.036	10	-142.3	267	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.41575	-174.6	9.1	-154	251	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.41311	-174.617	9.1	-157.3	251	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.40527	-174.662	9	-154	252	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.39625	-174.715	9.1	-159	249	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.39278	-174.733	9.1	-157	250	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.38917	-174.752	9.3	-157.9	252	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.36916	-174.851	9.2	-157.1	251	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31
366940480	2017-01-0	52.36112	-174.892	9.3	-152.8	255	EARLY DA	IMO78211	WDB7319	1001	undefinec	32.95	8.82	4	31

Figure 4–1. Overview of AIS data characteristics

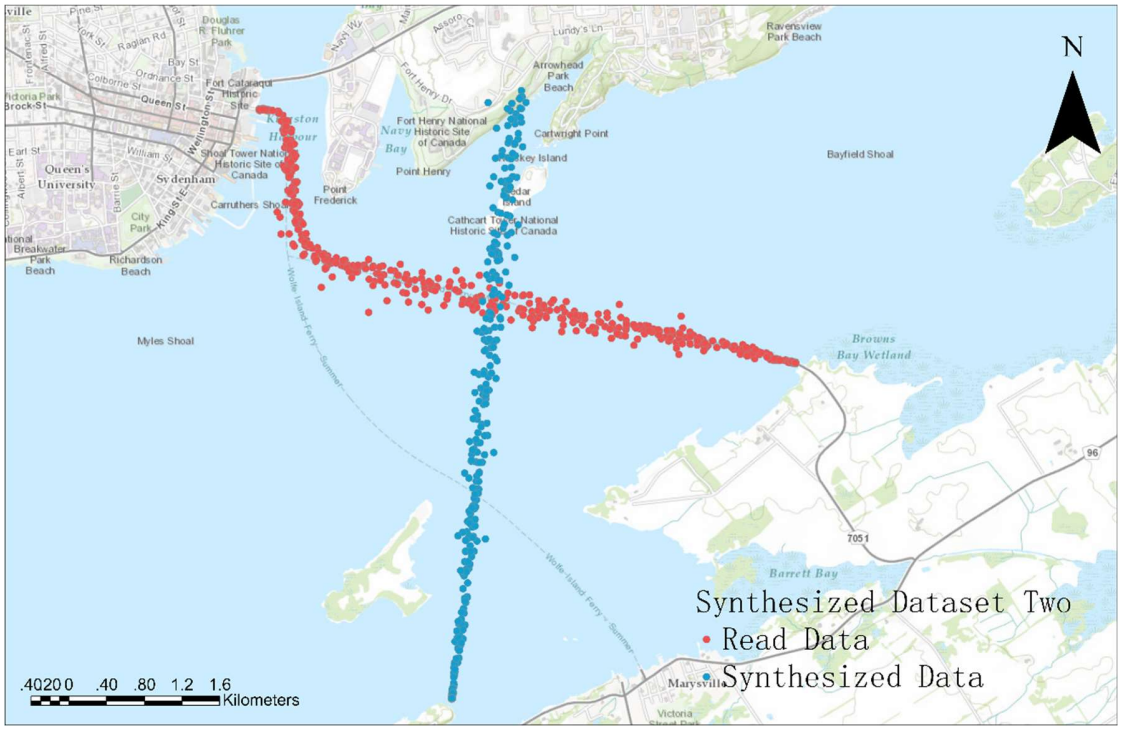
Though, in this research, we use open-sourced AIS data as the primary data source for the proposed algorithm testing. Three scales of data sets are used for studying, and they

are stored in MongoDB data management system. A smaller sample dataset is used for the algorithm testing purposes, and two larger AIS datasets are chosen for final result validation. The selected small dataset for algorithm testing is Data of Wolfe Island Ferry in January 2017.

As shown in **Figure 4–2**, this dataset describes the Wolfe Island Ferry traveling between Kingston to port at Wolfe Island. Synthetic data are generated as additional supportive datasets to test the clustering algorithm's performance under two scenarios. For instance, the optimized DBSCAN algorithm should identify outliers and noises from the main trajectories. Also, the algorithm should distinguish different paths from intersections. So, two synthetic datasets are created based on the two synthetic datasets of Wolfe Island Ferry AIS data in January 2017, traveling between Kingston and Wolfe Island, for testing the two mentioned scenarios, as shown in **Figure 4–2**. The real data are shown in red color, and the synthetic outliers and crossing data in blue. One thousand noisy points were randomly generated around the main trajectory in **Figure 4–2a**, while 2000 points were rotated by 90 degrees in **Figure 4–2b**.



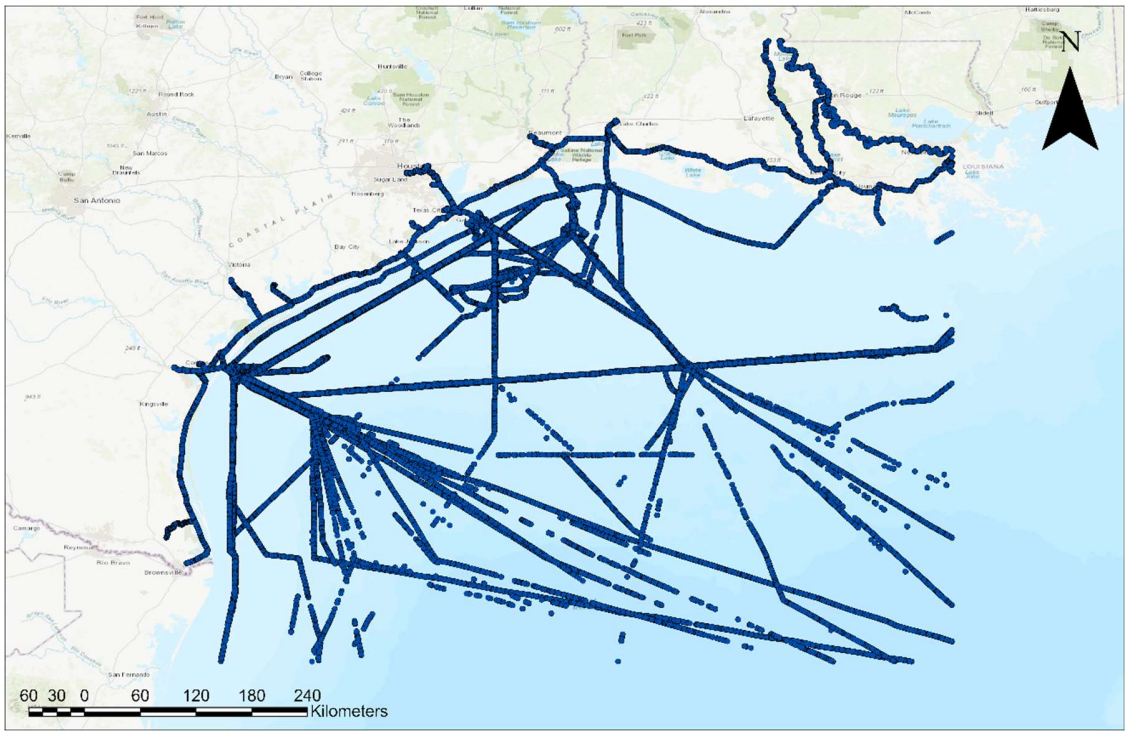
4-2a. Synthesized Dataset One to test outlier detection performance



4-2b. Synthesized Dataset Two to test the performance of distinguishing intersections

Figure 4–2. Two synthesized datasets for algorithm testing and performance evaluation

The proposed clustering algorithm has been applied on two big datasets as case studies, and models to monitor vessels in those regions have been generated. The first one is open-sourced data in the Gulf of Mexico Region (*MarineCadastre.gov | Vessel Traffic Data*, no date), and the second one is AIS data purchased from ORBCOMM (Saint Lawrence Seaway, from the Gulf of St. Lawrence to Lake Superior. **Figure 4–3** presents all the raw AIS point data of the two datasets.



4-3a. Raw AIS point data in the Gulf of Mexico Region

4-3b. Raw AIS Point Data of the Saint Lawrence Seaway Region

Figure 4–3. Two Raw AIS Big Data

The Gulf of Mexico Dataset is around 200 MB describing the whole month's vessel movement history of January 2017. The data contains around 1.2 million trajectory points to be clustered. The Saint Lawrence Seaway Region Dataset is around 17 MB describing the vessel movement history from 1st June 2017 to 3rd June 2017. The data contains around 135 thousand trajectory points to be clustered.

4.2 Testing and Evaluation Using Synthetic Data

The evaluation methods can be divided into two categories: internal evaluation and external evaluation. The differences between the two evaluation categories are whether external benchmarks or labels are referred to in the evaluation process. Internal evaluation methods evaluate a clustering performance based on the clustering results themselves. External evaluation methods evaluate clustering results based on external pre-defined labels and benchmarks as ground truth. Besides, the estimated number of clusters and noises are selected for evaluating the algorithms. The metrics selected in this research are: the estimated number of clusters and noises, Entropy (Homogeneity), Purity (Completeness), V-Measure, Adjusted Rand Index, F-measure (F1 Score), Davies-Bouldin Index, Silhouette Coefficient, and Calinski-Harabasz Index. Since each selected criterion indicates one aspect of the clustering performance, all of those indicators are utilized for holistic evaluation.

4.2.1 Internal Evaluation

The selected internal evaluation methods are the Davies-Bouldin Index (Davies and Bouldin, 1979), Silhouette Coefficient (Rousseeuw, 1987), and Calinski-Harabasz Index (Caliñski and Harabasz, 1974). **Table 4–1** states the detailed definitions of these internal evaluation metrics. All of them assign a ratio describing the average similarity within a cluster to the difference between clusters. Davies-Bouldin Index measures the similarities within clusters as the average Euclidean distance of all data points to the cluster centroid

and measures the difference between clusters as the distance between cluster centroids. The lower the Davies-Bouldin Index is, the better the clustering performance is. Silhouette Coefficient measures the similarities within clusters as the mean intra-cluster distance and measures the difference as the mean nearest-cluster distance. Silhouette Coefficient ranges from +1 (the best) to -1 (the worst). Both the Davies-Bouldin Index and Silhouette Coefficient indicate that a good clustering result should group all closed points while clusters are distant from one another. The Calinski-Harabasz Index is also known as the Variance Ratio Criterion. The score is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion. A better clustering result has a higher Calinski-Harabasz Index value.

Table 4–1. A comprehensive list of internal performance metrics for evaluating clustering results

Performance Metric	Equation	Parameters
Davies-Bouldin Index	$DB = \frac{1}{n}\sum_{i=1}^{n} \max_{j \neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$	• $\boldsymbol{n}$ is the number of clusters, • $\boldsymbol{c_i}$ is the centroid of cluster $\boldsymbol{i}$, • $\boldsymbol{\sigma_i}$ is the average distance of all elements in cluster $\boldsymbol{i}$ to the centroid $\boldsymbol{c_i}$, • $\boldsymbol{d(c_i, c_j)}$ is the distance between centroids $\boldsymbol{c_i}$ and $\boldsymbol{c_j}$
Silhouette Coefficient	$S = \frac{\sum\left\{\frac{b(i) - a(i)}{max(a(i), b(i))}\right\}}{n}$	• a is the mean intra-cluster distance, • b is the mean nearest-cluster distance, is the distance between a sample and the nearest cluster that the sample is not a part of, • n is the total number of points

Calinski-Harabasz Index	$$s(k) = \frac{Tr(B_k)}{Tr(W_k)} * \frac{N-k}{k-1}$$ Where: • $\boldsymbol{B_k}$ is the between-group dispersion matrix $$\boldsymbol{B_k} = \sum_{q} \boldsymbol{n_q}(\boldsymbol{c_q} - \boldsymbol{c})(\boldsymbol{c_q} - \boldsymbol{c})^{\boldsymbol{T}}$$ • $\boldsymbol{W_k}$ is the within-cluster dispersion matrix. $$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$	• $\boldsymbol{N}$ be the number of points in our data, • $\boldsymbol{C_q}$ be the set of points in cluster q, • $\boldsymbol{c_q}$ be the center of cluster q, • $\boldsymbol{c}$ be the center of $\boldsymbol{E}$, • $\boldsymbol{n_q}$ be the number of points in cluster $\boldsymbol{q}$.

However, the drawback of the internal criteria is obvious. High scores on an internal measure do not necessarily imply a more effective clustering method. Some of the clustering methods, take K-means as an example, optimize the clustering result using a similar model. Thus, internal criteria will be biased towards them and naturally assign them with higher scores. Therefore, the internal evaluation metrics only provide a reference to understand the structure of the clusters. The clustering performance evaluation cannot entirely depend on them.

4.2.2 External Evaluation

The external evaluation methods measure how close the clustering result is to the predetermined ground truth. **Table 4–2** states the detailed definitions of these external evaluation metrics. The selected external evaluation methods are Homogeneity, Completeness, V-measure (Rosenberg and Hirschberg, 2007), Adjusted Rand Index (Rand,

1971), and F1 Score/F-Measure (Sasaki and Fellow, 2007). Homogeneity, completeness, and V-Measure scores are the metrics to evaluate the clustering performance based on normalized conditional entropy measures of the clustering labeling. Entropy is a measure of the amount of disorder in a vector. Homogeneity score (h) maximizes when all of its clusters contain only data points from a single class. Completeness score (c) maximizes when all the data points from a given class are elements of the same cluster. The V-measure is the harmonic mean between homogeneity and completeness. All three metric scores range from 0 to 1. The greater values indicate better clustering performance. Adjusted Rand Index and F1 Score measures the clustering results' overall accuracy compared to the ground truth. The Rand Index measures the percentage of correct decisions, which is simply accuracy. The Rand index gives equal weight to false positives and false negatives. However, separating similar documents (FN) is usually worse than putting pairs of different documents in the same cluster (FP). To solve this problem, F-measure penalizes FN more strongly than FP by selecting a value $\beta > 1$, thus giving more weight to recall. The greater is the F-measure, the better is the clustering results.

Table 4–2. A comprehensive list of external performance metrics for evaluating clustering results

Performance Metric	Equation	Parameters
Homogeneity	$$c = 1 - \frac{H(K\|C)}{H(K)}$$ Where: $$\boldsymbol{H(C) = -\sum_{c=1}^{\|C\|} \frac{n_c}{n} * \log\left(\frac{n_c}{n}\right)}$$ $$H(C\|K) = -\sum_{c=1}^{\|C\|}\sum_{k=1}^{\|K\|} \frac{n_c k}{n} * \log\left(\frac{n_{c,k}}{n_k}\right)$$	• $\boldsymbol{n}$ is the total number of samples, • $\boldsymbol{n_c}$ and $\boldsymbol{n_k}$ belong are the number of samples of class C and class K respectively, • $\boldsymbol{n_{c,k}}$ are the number of samples divided from class C to class K.
Completeness	$$\boldsymbol{h = 1 - \frac{H(C\|K)}{H(C)}}$$	Same as Homogeneity
V-measure	$$v = \frac{(1+\beta) * h * c}{\beta * h + c}$$ By default, when beta equals 1, v-measure is defined by: $$\boldsymbol{v = 2 * \frac{h * c}{h + c}}$$	Same as Homogeneity

Adjusted Rand Index and F1 Score (F-Measure)	$$\boldsymbol{RI} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{FP} + \boldsymbol{FN} + \boldsymbol{TN}}$$ $$\boldsymbol{F_\beta} = \frac{(\boldsymbol{\beta^2} + \mathbf{1})\boldsymbol{Pr}}{\boldsymbol{\beta^2 Pr} + \boldsymbol{Rc}}$$ When by default, β = 1: $$F = \frac{2Pr * Rc}{Pr + Rc}$$	• TP: True Positive, assigns two similar documents to the same cluster • FP: False Positive, assigns two different documents to different clusters • TN: True Negative, assigns two dissimilar documents to the same cluster • FN: False Negative, assigns two similar documents to different clusters • Precision: $\boldsymbol{Pr} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}$ • Recall: $\boldsymbol{Rc} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}}$

However, the external evaluations require the assumption that a factual ground truth exists for any real or synthetic dataset, and human experts can generate them as training data. Besides the challenges of creating accurate ground-truth data, the core concept of clustering methods does not entirely fit the assumption. Various clustering methods can group the data while discovering complex but unknown similarities between the data. The given ground truth will exclude any other possibilities to cluster the dataset. Thus, the external evaluation uses subjective ground truth and evaluates how close the clustering results to it. The ground truth labels are manually pre-defined for the two synthesized datasets.

4.2.3 Enhanced DBSCAN Algorithm Performance Evaluation

In this section, experiments are performed to evaluate the proposed approach's effectiveness by comparing it with various commonly used machine learning algorithms.

The clustering results from those algorithms are used as a reference on the testing datasets. Since the proposed algorithm has two components: unsupervised components for generating clusters in the first place and the supervised component that uses the previous result as training data to label the rest, this section evaluates two parts of the algorithms separately. Thus, the selected algorithms are also divided into two categories. The selected unsupervised clustering methods are K-Means and traditional DBSCAN methods. The selected supervised clustering methods are K Nearest Neighbors (KNN), Support Vector Machines (SVM), and Artificial Neural Network (ANN).

K-Means method requires the user to input the k value as a pre-determined parameter. K was input into both experiments on two datasets based on the ground truth. Traditional DBSCAN also requires the user to input Eps and MinPts values as a pre-determined parameter. The parameters were obtained accordingly by the parameter auto-selection method proposed in Section 3.4, though with some adjustment. KNN requires the user to input the k value as a pre-determined parameter, and k=5 was used in this experiment.

The following **Figure 4–4** presents the designed artificial neural network structure (two hidden layers are omitted). The detailed structure is described as follows.

- An Input Layer: five neurons
- A Dense Layer: 128 neurons, with ReLU activation and L1 (Lasso Regression) and L2 (Ridge Regression) regularization whose parameter was set to be 0.01.
- A Dense Layer: 64 neurons with ReLU activation

- A Dense Layer: 32 neurons with ReLU activation,
- A Dense Layer: 16 neurons with ReLU activation,
- A Dense Layer: 8 neurons with sigmoid activation.
- An Output Layer: The number of neurons depends on how many pre-defined clustered in the training data.

The input layer has five neurons, which are represented by five attributes of the trajectory point. Those L1 and L2 regularizers add penalty as model complexity increases, to avoid overfitting on noises in the training data. ReLU gives output zero for all negative inputs, while returns any positive value back. The ReLU activation is selected in the intermediate layers because it is simple and it consists of no heavy computation, making the training process faster. The sigmoid activation function takes input and maps the resulting values in between 0 to 1. Since what the last layer does is a binary classification task to determine if the point belongs to corresponding class, the sigmoid activation is selected. The number of neurons in the last dense layer depends on how many pre-defined clustered in the training data. For example, if the data is in Class One out of five classes, the output layer can be represented as $[1, 0, 0, 0, 0]$.

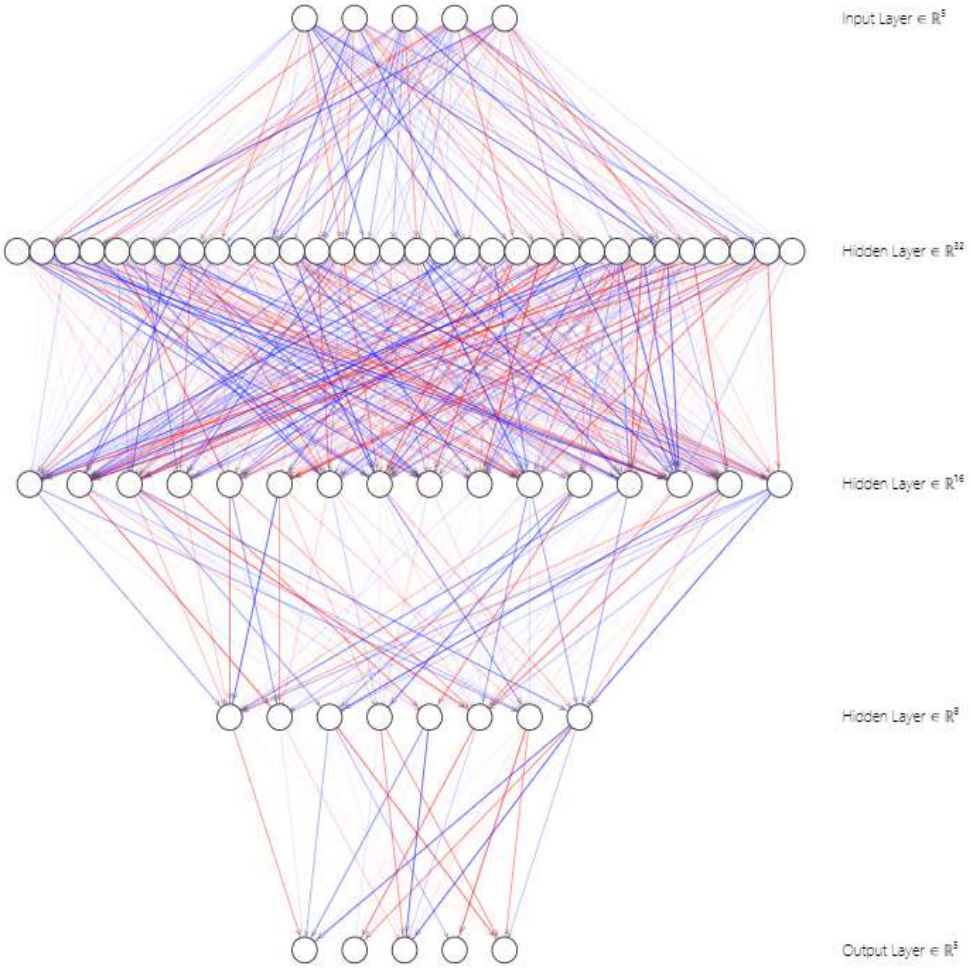


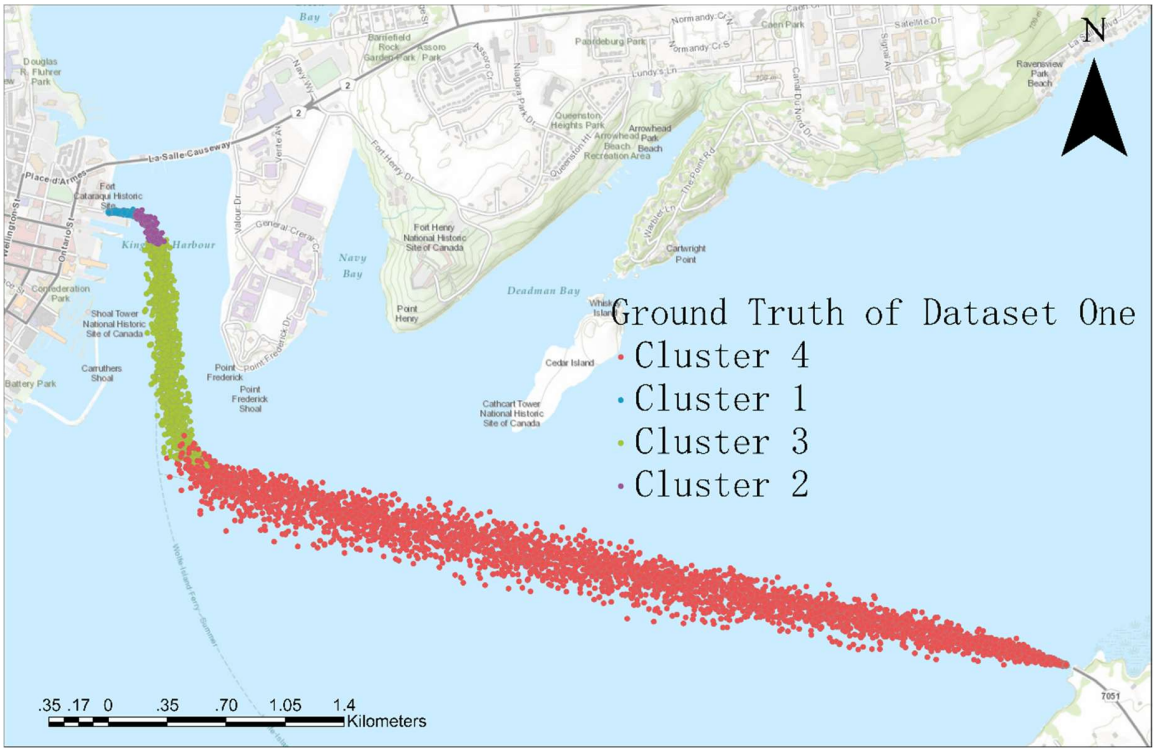
Figure 4–4. Designed structure of the Artificial Neural Network to be used in algorithm comparison (Two Hidden Layers Are Omitted)

Finding clusters and detecting outliers are the two performances to be evaluated in this section, and both external evaluation and internal evaluation criteria are used in the evaluation. Overall, this section presents four comparison groups categorized by the types of clustering algorithms and clustering purposes. The four groups are clustering

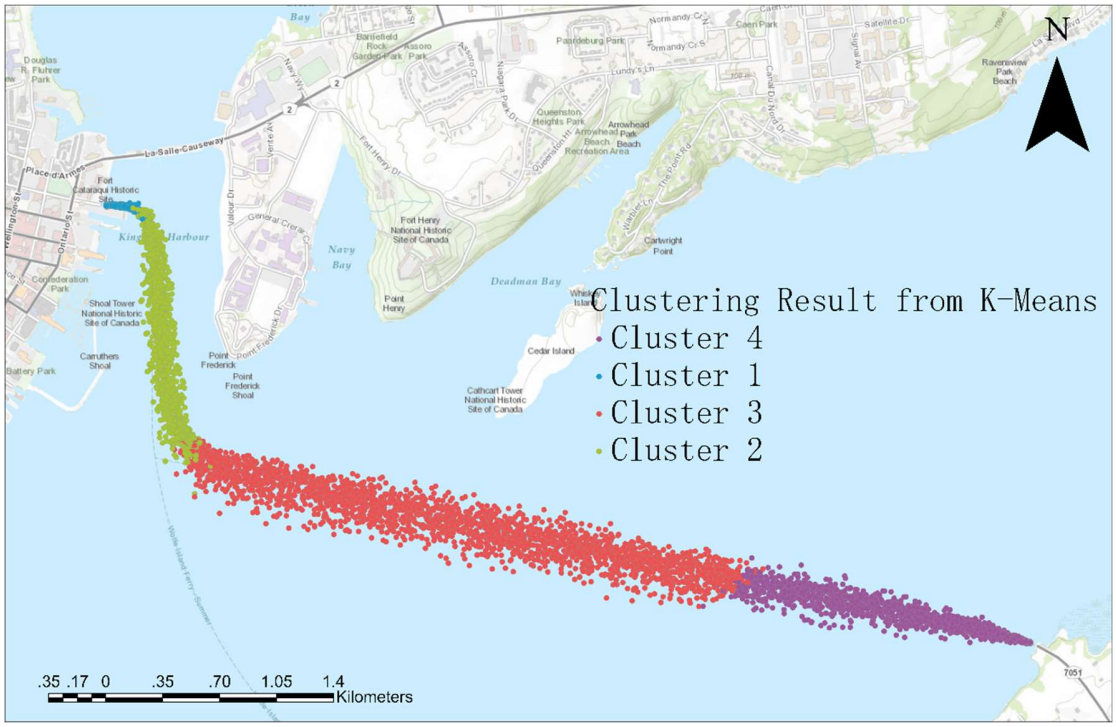
performance of using unsupervised learning to discover clusters (Section 4.2.3.1), clustering performance of using unsupervised learning to detect outliers (Section 4.2.3.2), the performance of using supervised learning to classify new observations (Section 4.2.3.3), the performance of using supervised learning to detect outliers (Section 4.2.3.4).

4.2.3.1 Clustering Performance of Using Unsupervised Learning to Discover Clusters

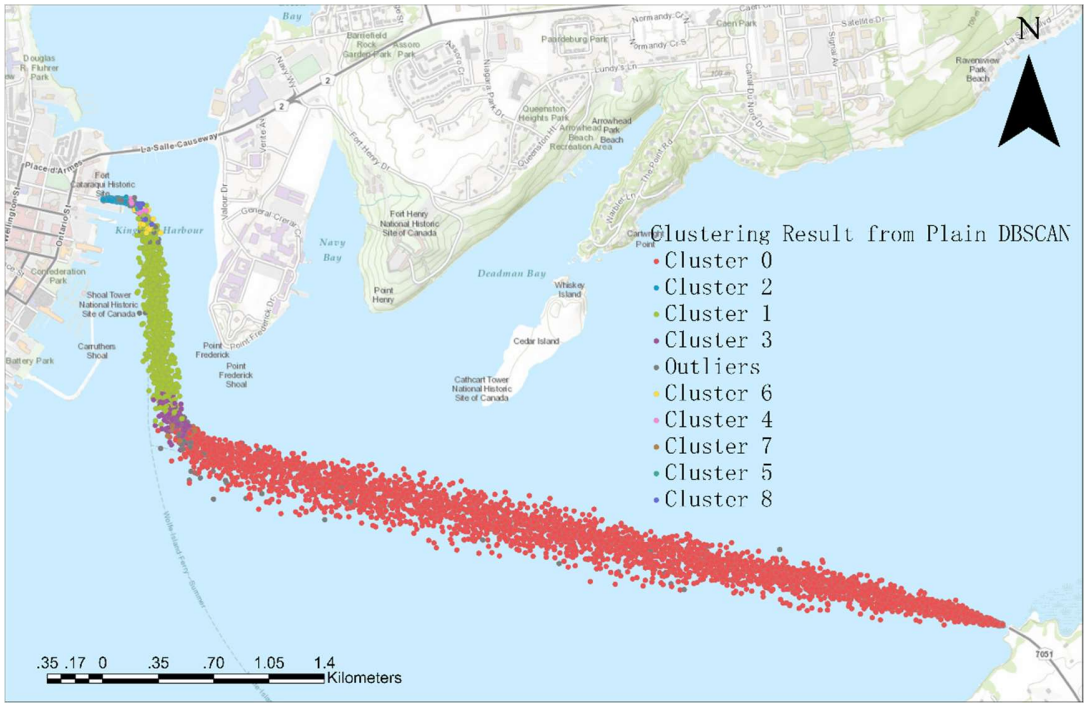
This section evaluates the proposed clustering method's clustering performance on discovering clusters by comparing it with other unsupervised learning algorithms. **Figure 4–5** shows comparison results among unsupervised learning algorithms implementing on Dataset One. **Figure 4–5a** presents the synthesized ground truth for this comparison set. As previously mentioned in Section 3.1, the proposed clustering framework utilizes the unsupervised component of the raw data to discover the clusters first. The experiment only uses 30% of the Dataset One to test the clustering performance by the same framework. Since this section focuses on evaluating the performance of discovering clusters, the synthesized outliers are filtered before implementing the algorithms. **Figure 4–6** shows comparison results among unsupervised learning algorithms implementing on Dataset Two. Due to a smaller data size of Dataset Two, 40% of the Dataset Two are used for testing, and **Figure 4–6a** presents the synthesized ground truth of Dataset Two.



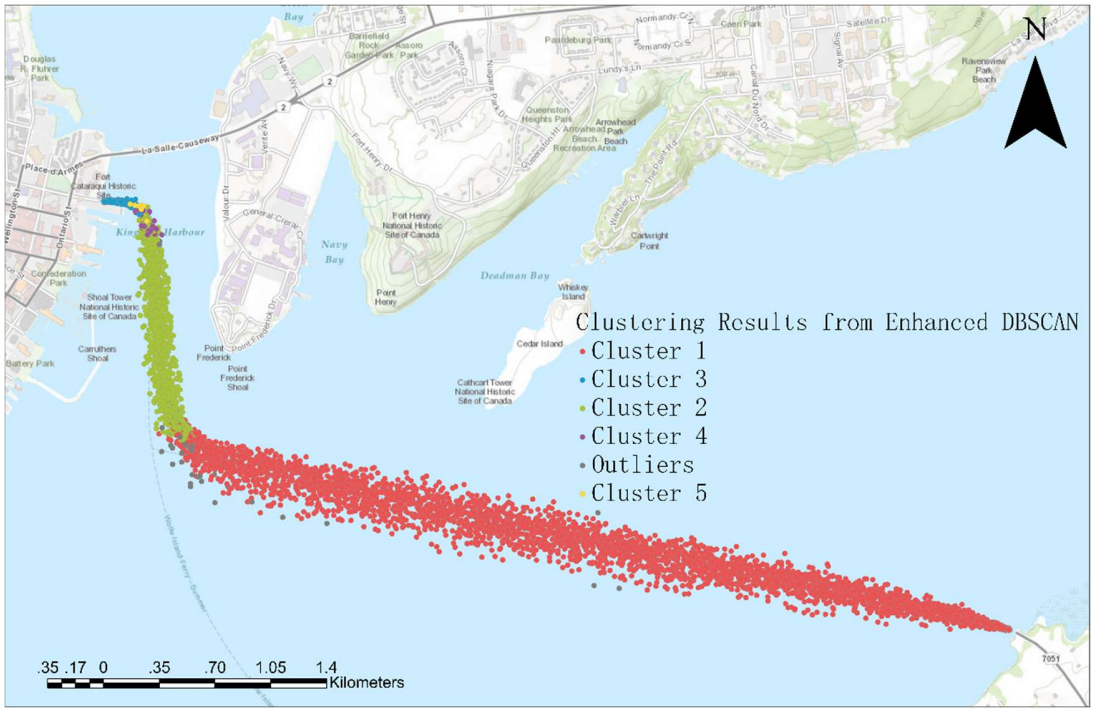
4-5a. Ground Truth of Dataset One



4-5b. Clustering Result from K-Means

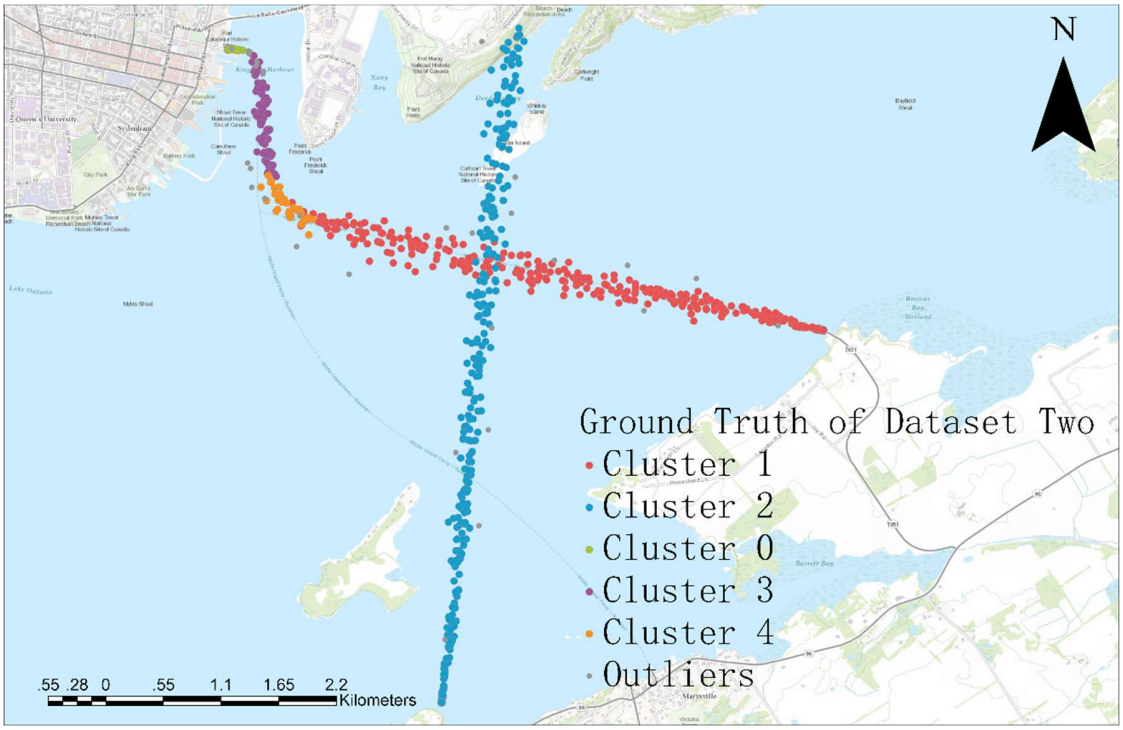


4-5c. Clustering Result from Plain DBSCAN

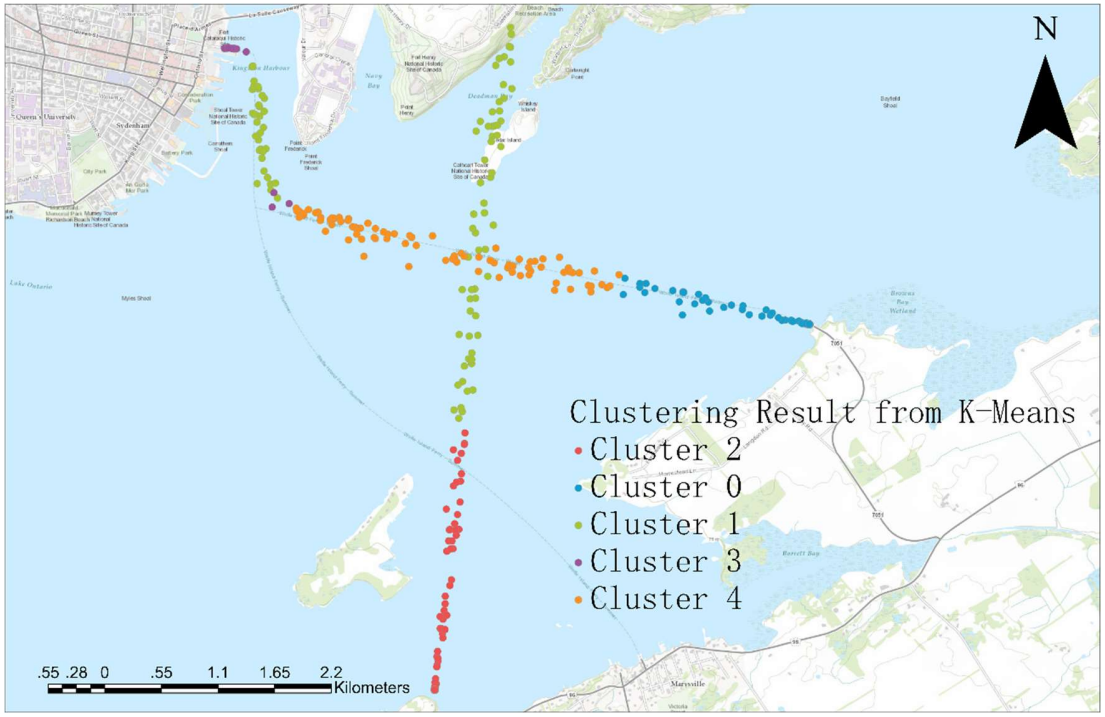


4-5d. Clustering Results from Enhanced DBSCAN

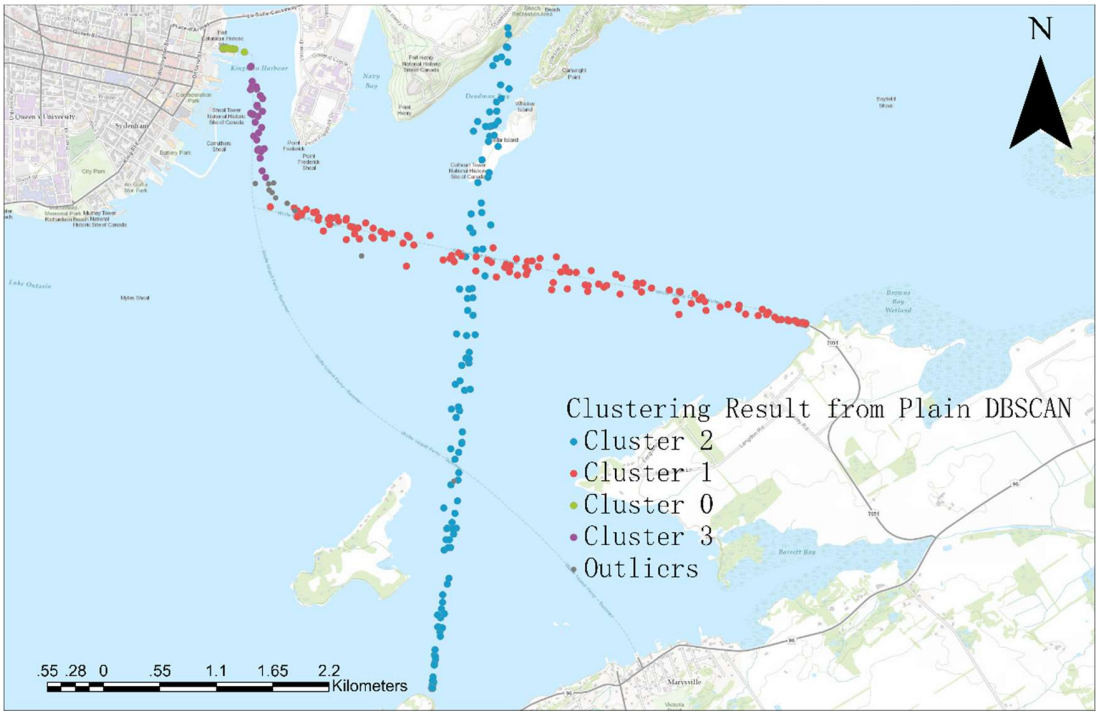
Figure 4–5. Comparing Enhanced DBSCAN's performance on discovering clusters from synthetic dataset one to ground truth and other unsupervised clustering methods



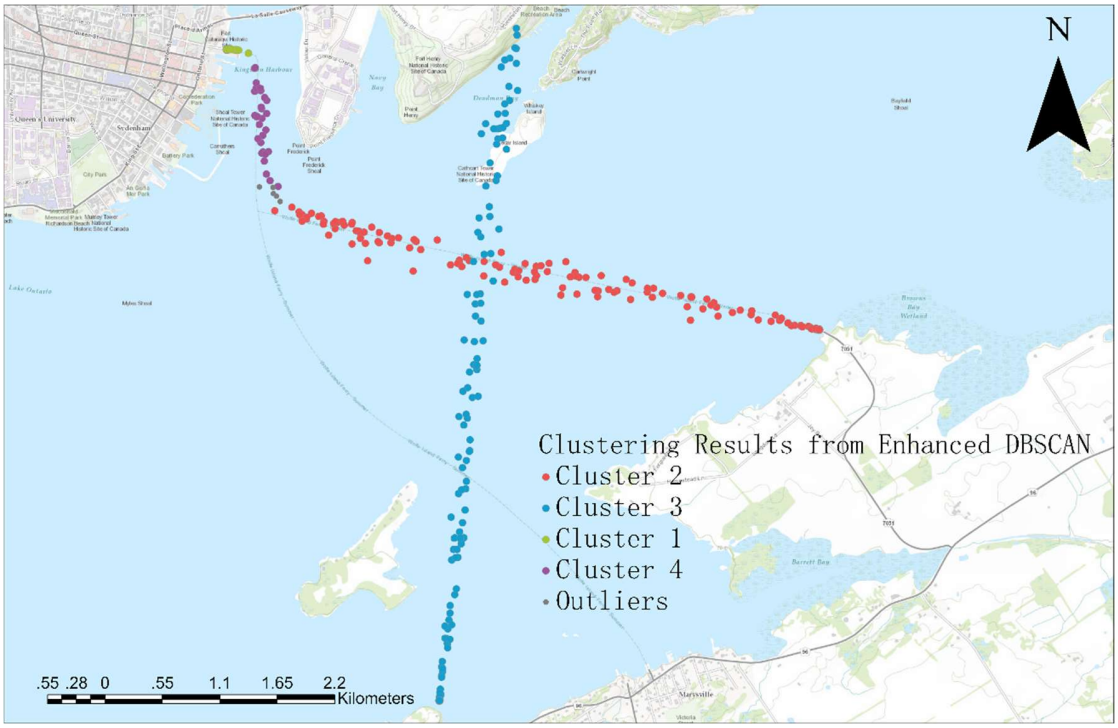
4-6a. Ground Truth of Dataset Two



4-6b. Clustering Result from K-Means



4-6c. Clustering Result from Plain DBSCAN



4-6d. Clustering Results from Enhanced DBSCAN

Figure 4–6. Comparing Enhanced DBSCAN's performance on distinguishing intersections from synthetic dataset two to ground truth and other unsupervised clustering methods

By visually evaluating the performance on discovering the clusters from **Figure 4–5** and **Figure 4–6** first, results from Plain DBSCAN and Enhanced DBSCAN is more similar to the ground truth than the results from K-means. K-means requires the user to input k as a pre-determined parameter, and 'k=4' and 'k=5' were input into both experiments on two datasets based on the ground truth. K-means did not function well even though with the pre-determined parameters. For example, as **Figure 4–5b** shows, Cluster 4 in Ground Truth (the cluster in red in **Figure 4–5a**) was divided into two clusters by K-means (the cluster in red and green in **Figure 4–5b**). The ground truth indicates that since the vessels keep the same heading and speed within Cluster 4, making it simply the constant cluster is not recommended to be divided. **Figure 4–5b** shows a wrong clustering result from K-means. As shown in **Figure 4–5c** and **Figure 4–5d**, both Plain DBSCAN and Enhanced DBSCAN still find outliers even though the outliers in the ground truth were filtered in the first place. The results from both DBSCAN methods can be justified by the nature of DBSCAN algorithms and the proposed parameter selection method mentioned in Section 3.4, that the most deviated data within the dataset will be considered outliers. Both two DBSCAN methods find more than four clusters, which is different from the ground truth.

Besides, as mentioned in Section 4.2.1 and Section 4.2.2, the proposed clustering algorithm's performance is assessed using the selected metrics, external evaluation, and internal evaluation metrics. Before evaluating the results by the selected metrics, some divided clusters were combined to ensure evaluation is more valid, but the outliers were

not edited into any clusters. The action may favor K-means. **Table 4–3** states the values of the clustering performance metrics of all clustering algorithms.

Table 4–3. Clustering performance evaluation of various unsupervised methods on discovering clusters and distinguishing intersections

Data Set	Algorithms	Estimated number of clusters	Entropy / Homogeneity	Purity / Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Information	F1 Score	Silhouette Coefficient	Davies-Bouldin Index	Calinski-Harabasz Index
	Ground Truth	4	1	1	1	1	1	1	0.671	0.687	14512.862
Data Set One	K-Means	4	0.906	0.959	0.932	0.974	0.932	0.976	0.706	0.371	22519.854
	Plain DBSCAN	5	0.879	0.879	0.879	0.956	0.879	0.97	0.65	0.781	12416.903
	Enhanced DBSCAN	5	0.934	0.967	0.95	0.985	0.95	0.986	0.666	0.49	14429.481
	Ground Truth	5	1	1	1	1	1	1	0.557	0.586	528.491
Data Set Two	K-Means	3	0.796	0.977	0.877	0.877	0.887	0.811	0.55	0.633	681.302
	Plain DBSCAN	5	0.955	0.967	0.961	0.979	0.96	0.987	0.548	0.735	499.87
	Enhanced DBSCAN	5	0.949	0.981	0.965	0.979	0.965	0.986	0.552	0.562	492.229

Figure 4–7 to **Figure 4–10** visualize the values of each clustering performance metric. Various colors are used to represent different clustering methods. The x-axis represents external evaluation attributes, and the y-axis represents the corresponding score.

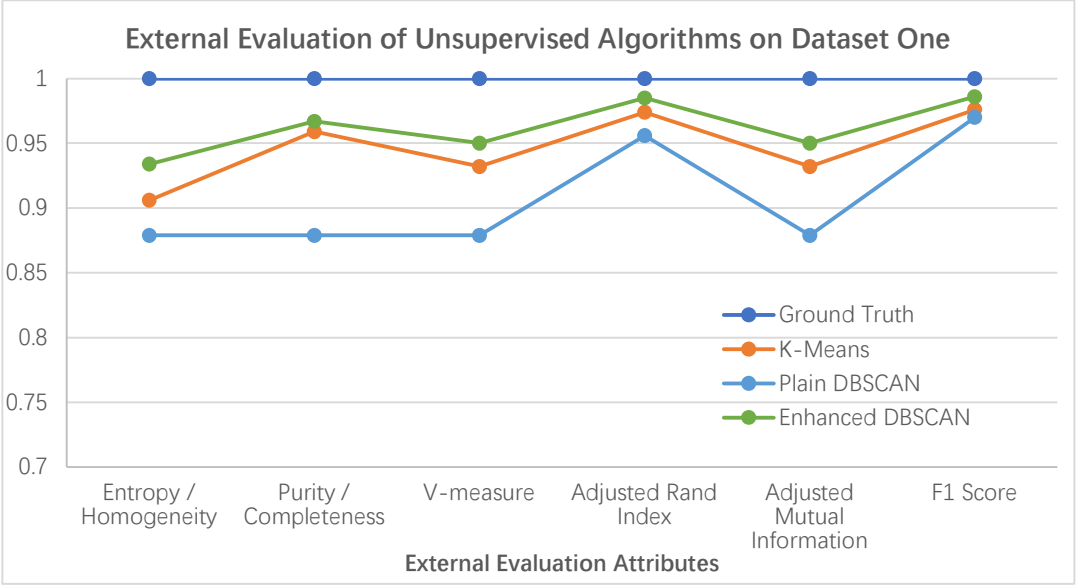


Figure 4–7. Performance evaluation of discovering clusters on unsupervised algorithms using external evaluation metrics on dataset one

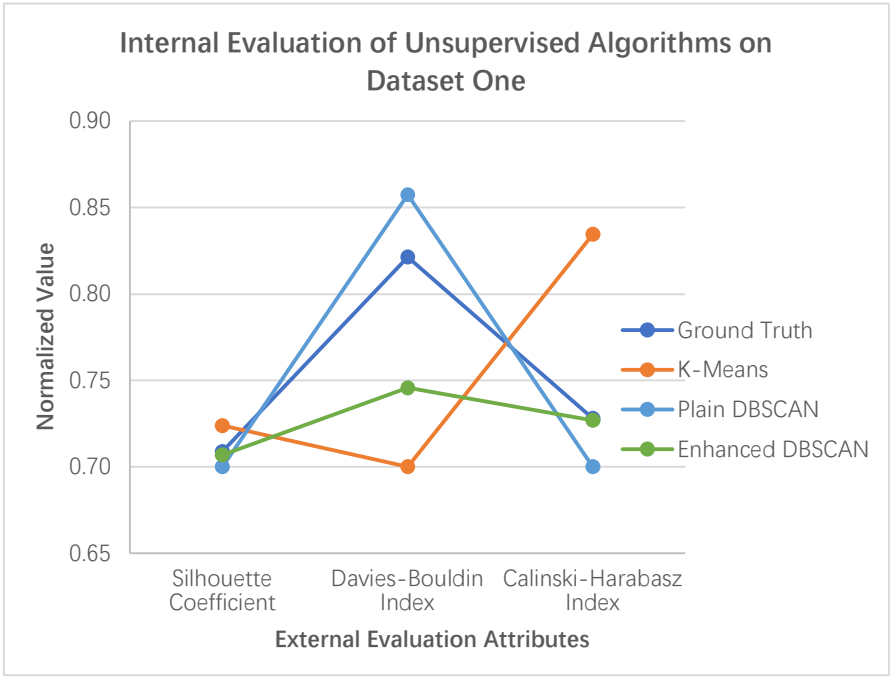


Figure 4–8. Performance evaluation of discovering clusters on unsupervised algorithms using internal evaluation metrics on dataset one

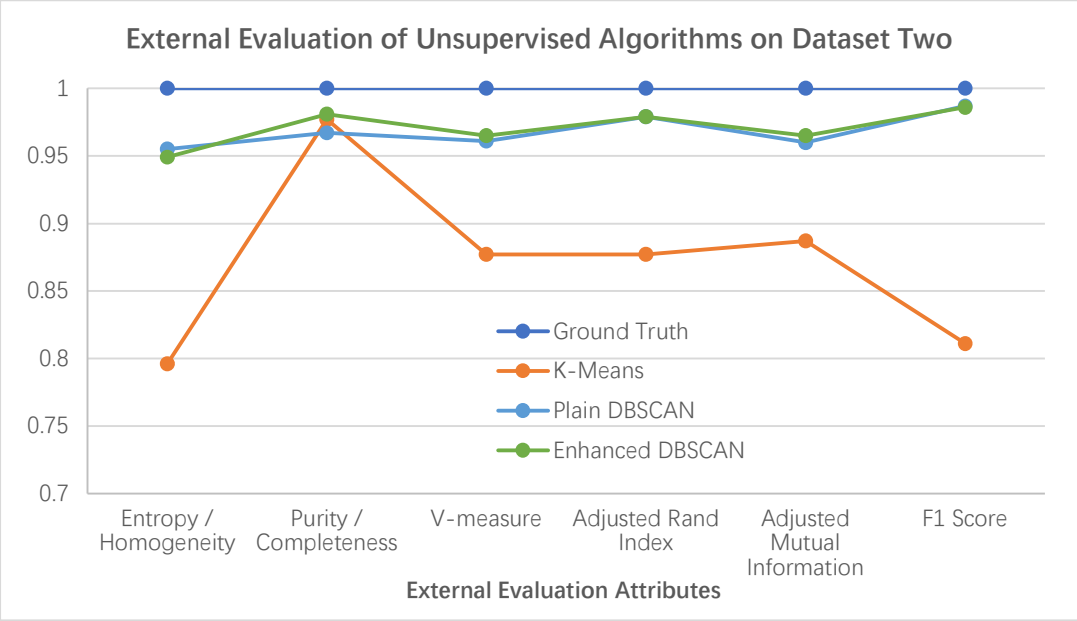


Figure 4–9. Performance evaluation of distinguishing intersections and discovering clusters on unsupervised algorithms using external evaluation metrics on dataset two

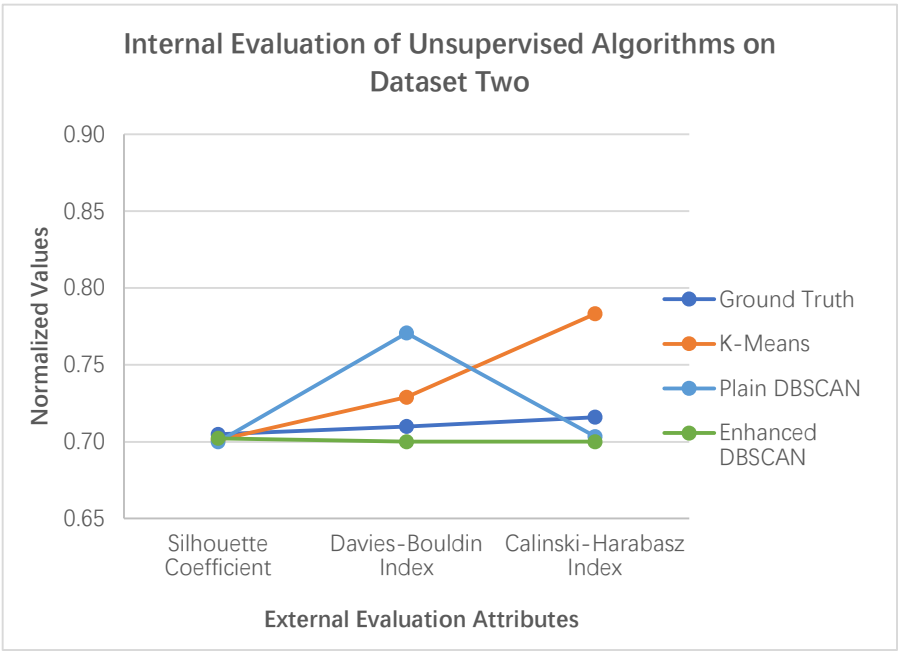


Figure 4–10. Performance evaluation of distinguishing intersections and discovering clusters on unsupervised algorithms using internal evaluation metrics on dataset two

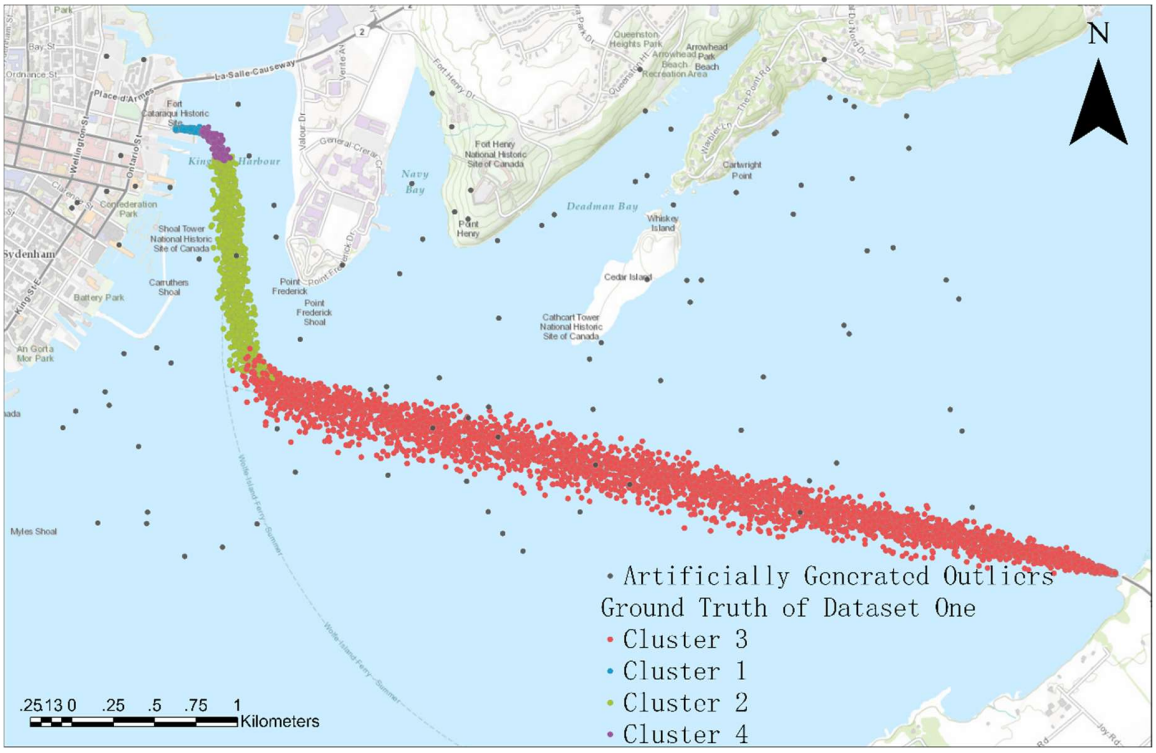
The results indicate that the proposed approach has the best performance regarding external evaluation by analyzing both **Figure 4–7** to **Figure 4–10** and **Table 4–3**. The ground truth is one on every external evaluation attribute, and the enhanced DBSCAN has the closest values to it compared to other algorithms. On analysis of Dataset One, the entropy value of enhanced DBSCAN is 0.934. The purity value and F1 Score of enhanced DBSCAN is 0.934 and 0.986 respectively. It is found that the enhanced DBSCAN has the highest score on these metrics, demonstrating the proposed algorithm has good accuracy, homogeneity, and completeness.

Internal evaluation metrics' values were normalized into 0.5 – 1 so that all three internal evaluation metrics can share the same scale. Since the internal evaluation does not use ground truth as a reference, the internal evaluation values cannot determine which algorithm gives a better clustering result. So, this experiment uses internal evaluation methods to evaluate the ground truth and the values used as indicators for describing the ground truth. The results of internal evaluation methods on other algorithms are compared with the ground truth. The closer the values to the ground truth, the result is more similar to the ground truth, and it has better clustering performance.

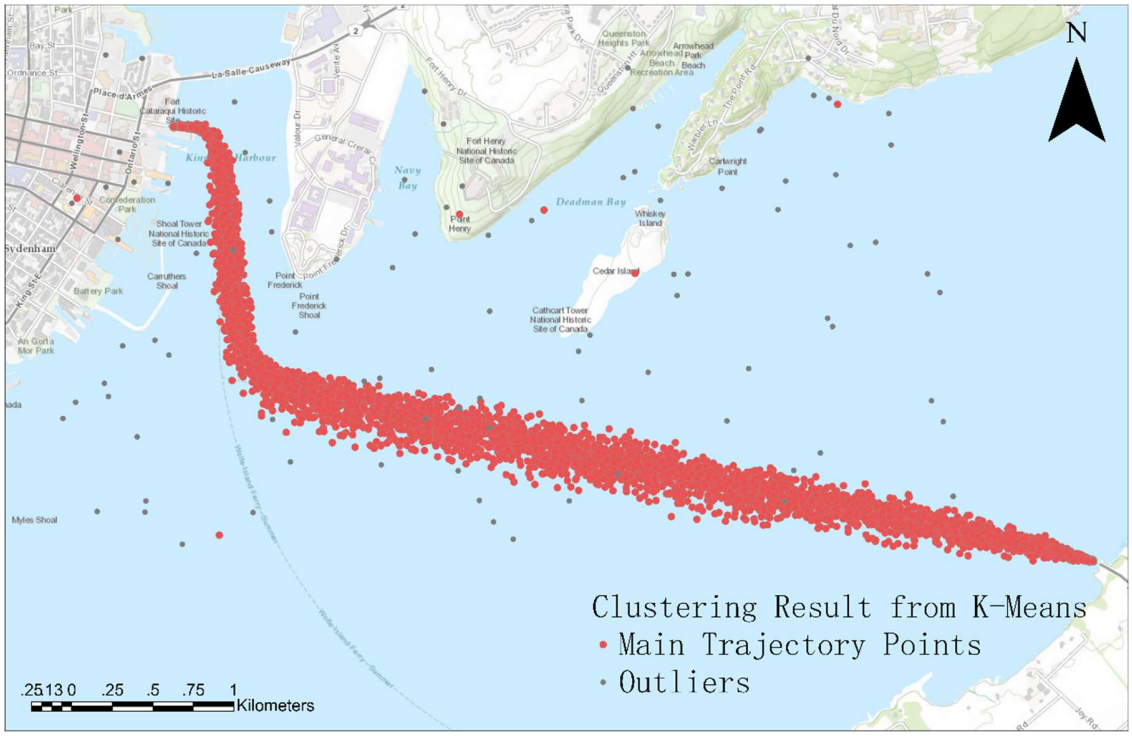
Overall, the enhanced DBSCAN results satisfy the expectation that it can keep a high level of performance quality in terms of external evaluation metrics, compared to other unsupervised algorithms. The enhanced algorithm results are also very similar to ground truth concerning internal evaluation metrics with less than 0.1 difference.

4.2.3.2 Clustering Performance of Using Unsupervised Learning to Detect Outliers

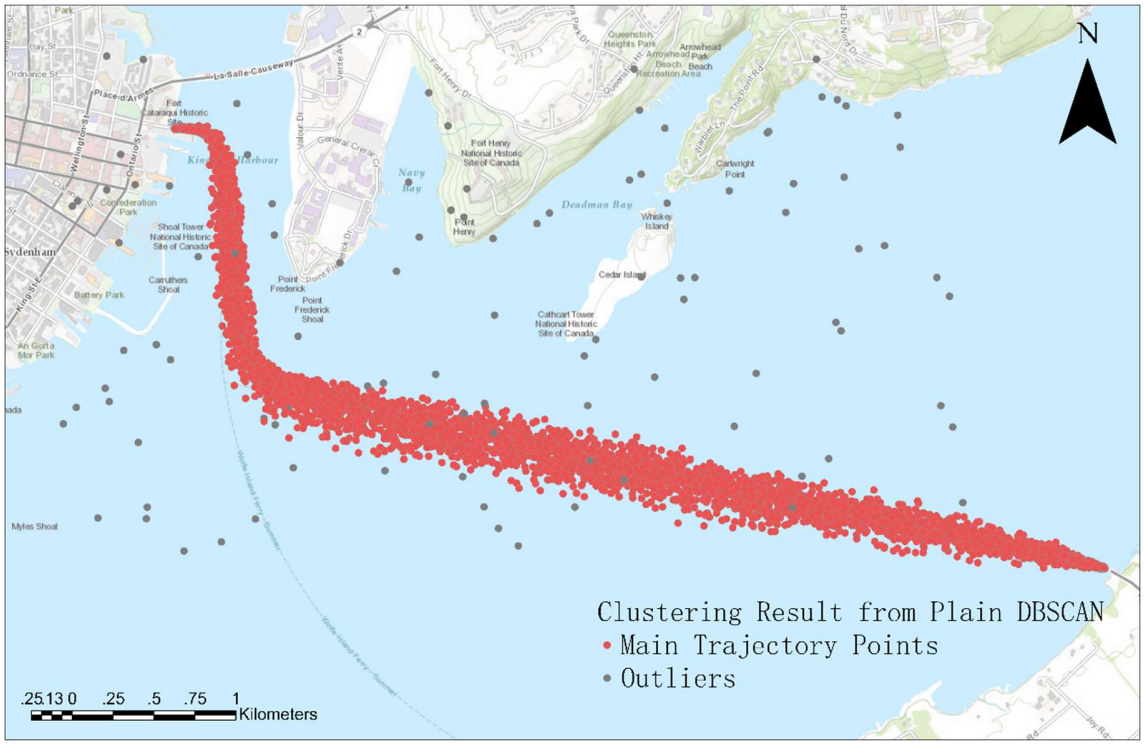
This section evaluates the proposed clustering method's clustering performance in detecting outliers by comparing it with other unsupervised learning algorithms. **Figure 4–11** shows comparison results among unsupervised learning algorithms implementing on Dataset One. **Figure 4–11a** presents the synthesized ground truth for this comparison set. Same to Section 4.2.3.1, the experiment only uses 30% of the Dataset One to test the clustering performance. One hundred multivariate outliers were artificially synthesized around the main trajectory. The experiment tests binary classification performance to compare the results from various unsupervised clustering algorithms on the same level.



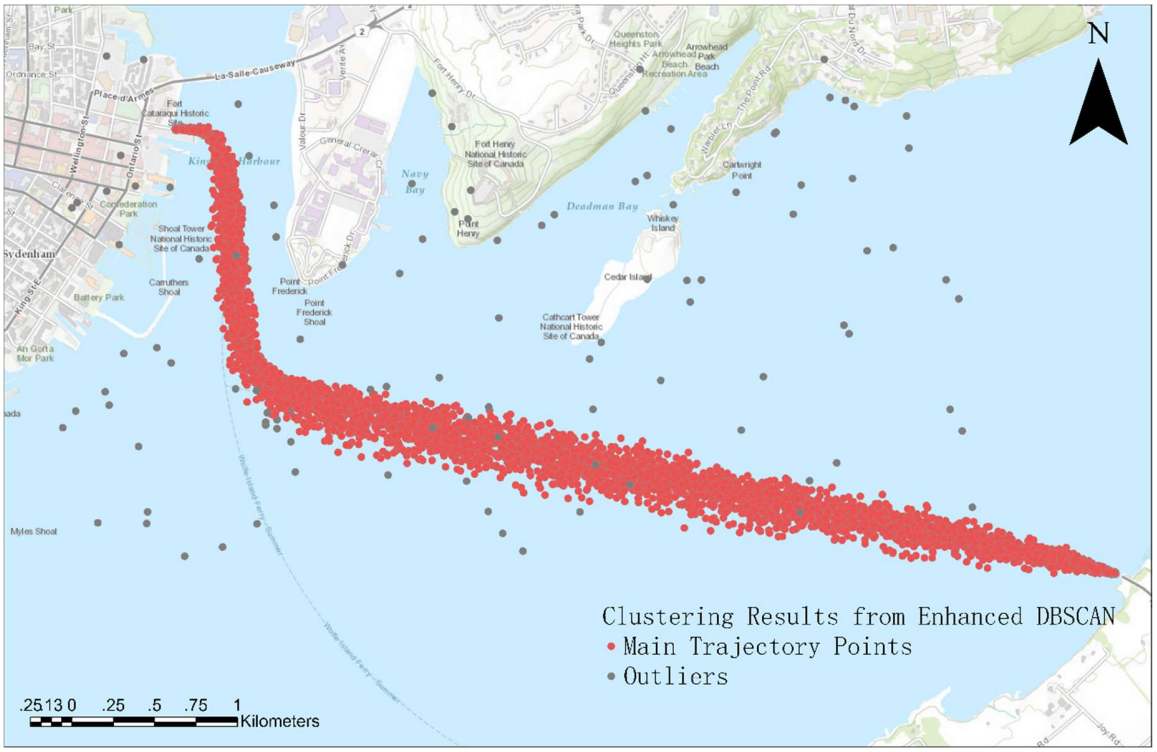
4-11a. Ground Truth of Dataset One



4-11b. Clustering Result from K-Means



4-11c. Clustering Result from Plain DBSCAN



4-11d. Clustering Results from Enhanced DBSCAN

Figure 4–11. Comparing Enhanced DBSCAN's performance on detecting outliers from synthetic dataset one to ground truth and other unsupervised clustering methods

By visually evaluating the performance on discovering the clusters from **Figure 4–11** first, Plain DBSCAN and Enhanced DBSCAN have better performance on detecting outliers than K-means. “K = 2” was input into K-Means to differentiate outliers from the main trajectory. As shown in **Figure 4–11b**, some outliers are identified as main trajectory points, and they are wrong. DBSCAN methods, by their natures, can find outliers based on the “density connectivity” of each cluster. To modify the method for doing binary classification tasks, parameters are adjusted from the results given by the proposed parameter selection method mentioned in Section 3.4 for this purpose. As shown in **Figure 4–11c** and **Figure 4–**

11d, both Plain DBSCAN and Enhanced DBSCAN successfully identify synthesized outliers but still find more outliers as they are also relatively deviated from the main route.

Besides, the proposed clustering algorithm's performance is assessed by the selected metrics, both external evaluation, and internal evaluation metrics. **Table 4–4** states the values of the clustering performance metrics of all clustering algorithms.

Table 4–4. Clustering performance evaluation of various unsupervised methods on detecting outliers

Data Set	Algorithms	Estimated number of noise points	Entropy / Homogeneity	Purity / Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Information	F1 Score	Silhouette Coefficient	Davies-Bouldin Index	Calinski-Harabasz Index
	Ground Truth	100	1	1	1	1	1	1	0.056	1.492	1487.002
Data Set One	K-Means	93	0.897	0.952	0.923	0.963	0.923	0.999	0.864	0.586	5493.952
	Plain DBSCAN	107	0.976	0.952	0.964	0.985	0.964	1	0.854	0.661	5117.953
	Enhanced DBSCAN	109	0.944	0.88	0.911	0.956	0.911	0.999	0.846	0.725	4708.045

Figure 4–12 and **Figure 4–13** visualize the values of each clustering performance metric. Various colors are used to represent different clustering methods. The x-axis represents external evaluation attributes, and the y-axis represents the corresponding score.

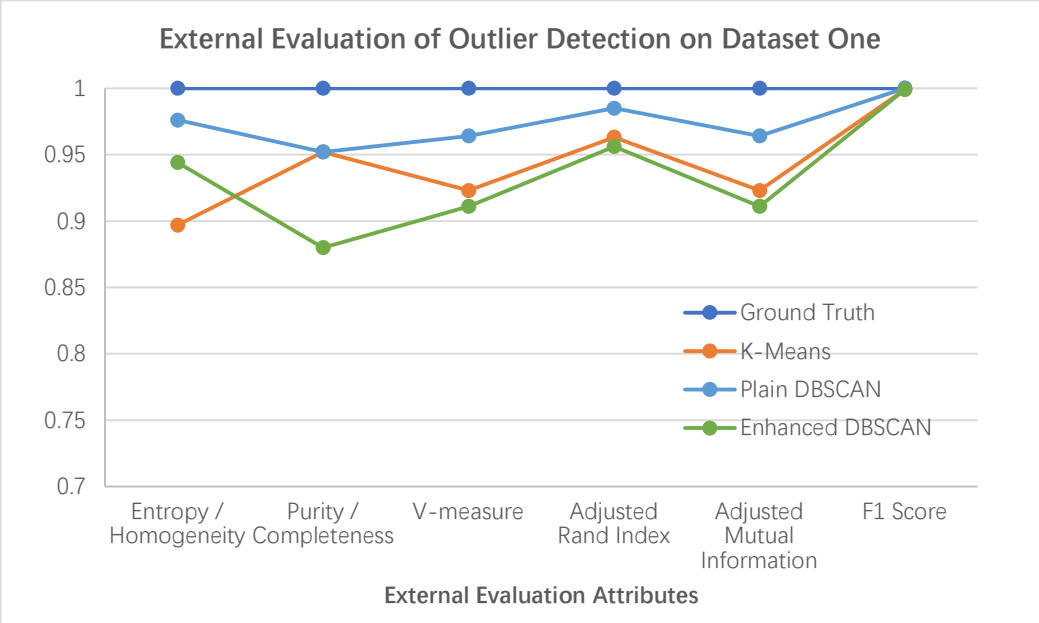


Figure 4–12. Performance evaluation of outlier detection on unsupervised algorithms using external evaluation metrics on dataset one

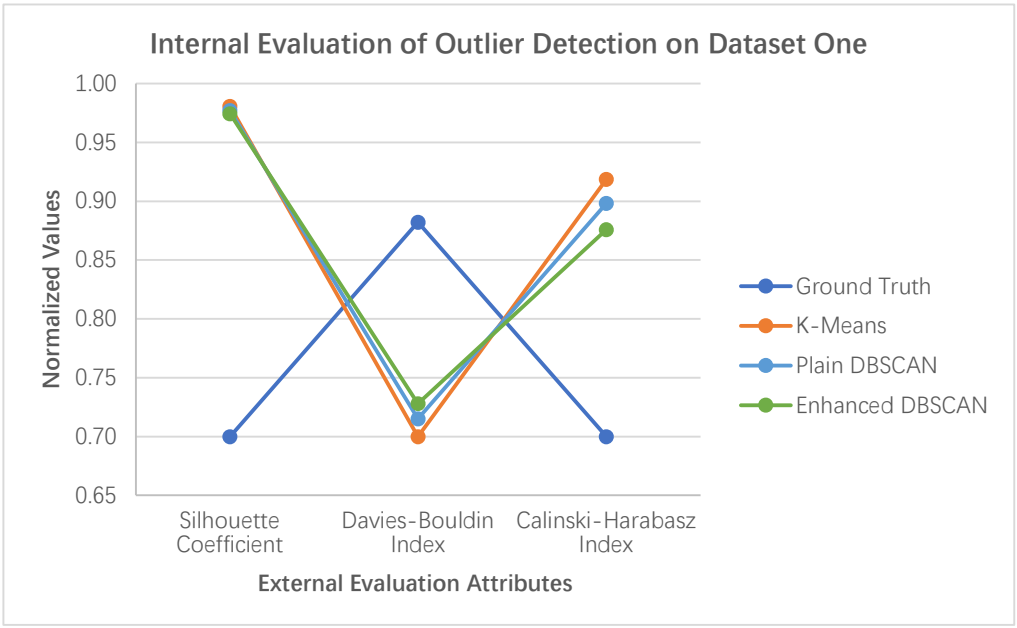


Figure 4–13. Performance evaluation of outlier detection on unsupervised algorithms using internal evaluation metrics on dataset one

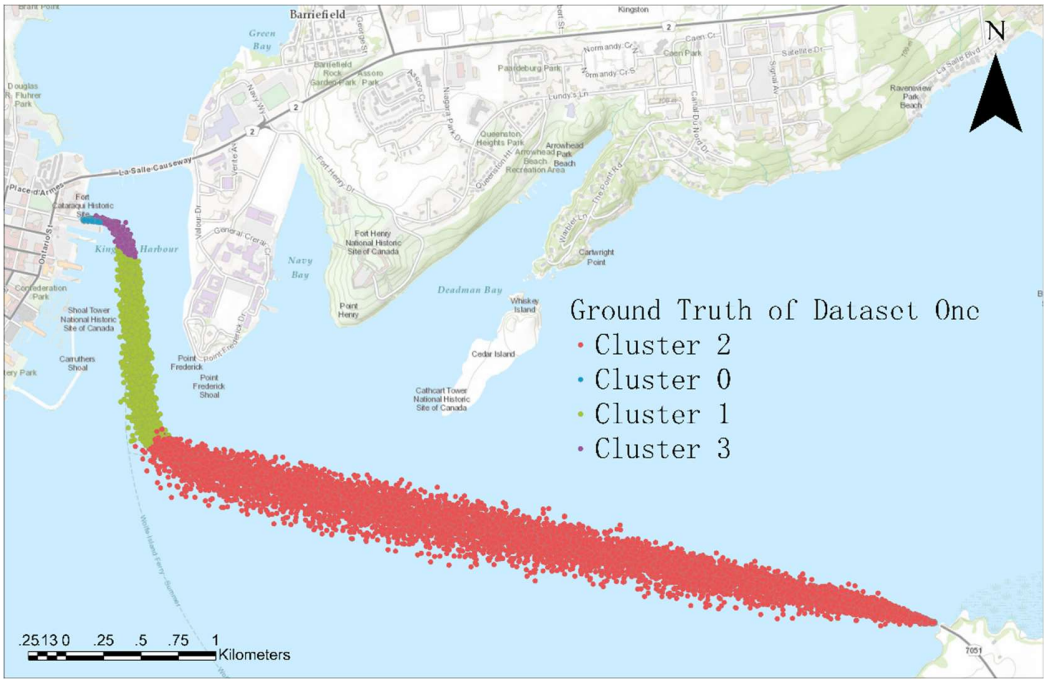
The results indicate that the proposed approach has high performance in detecting outliers based on the external evaluation and internal evaluation by analyzing **Figure 4–12**, **Figure 4–13**, and **Table 4–4**. According to the figures and tables, the plain DBSCAN has the highest external evaluations score with higher than 0.95. Even though the enhanced DBSCAN has a high F-1 score close to 1, indicating good accuracy, entropy, and purity scores, even still at a high level, is lower than plain DBSCAN (higher than 0.9). This can be explained by that enhanced DBSCAN finds more outliers than expected.

Same to Section 4.2.3.1, the values of internal evaluation metrics were normalized into 0.5 – 1 so that all three internal evaluation metrics can share the same scale. The results of internal evaluation methods on other algorithms are compared with the ground truth. According to **Figure 4–13**, the ground truth is different from the clustering method's results. This can be explained by that the ground truth may not be entirely objectively prepared by the analyzer. Some synthesized outliers that fall on the main trajectory can be part of the clusters. Thus, indicated by this experiment, the result of the enhanced DBSCAN satisfies the expectation that it keeps a high level of performance quality on detecting outliers among other unsupervised algorithms.

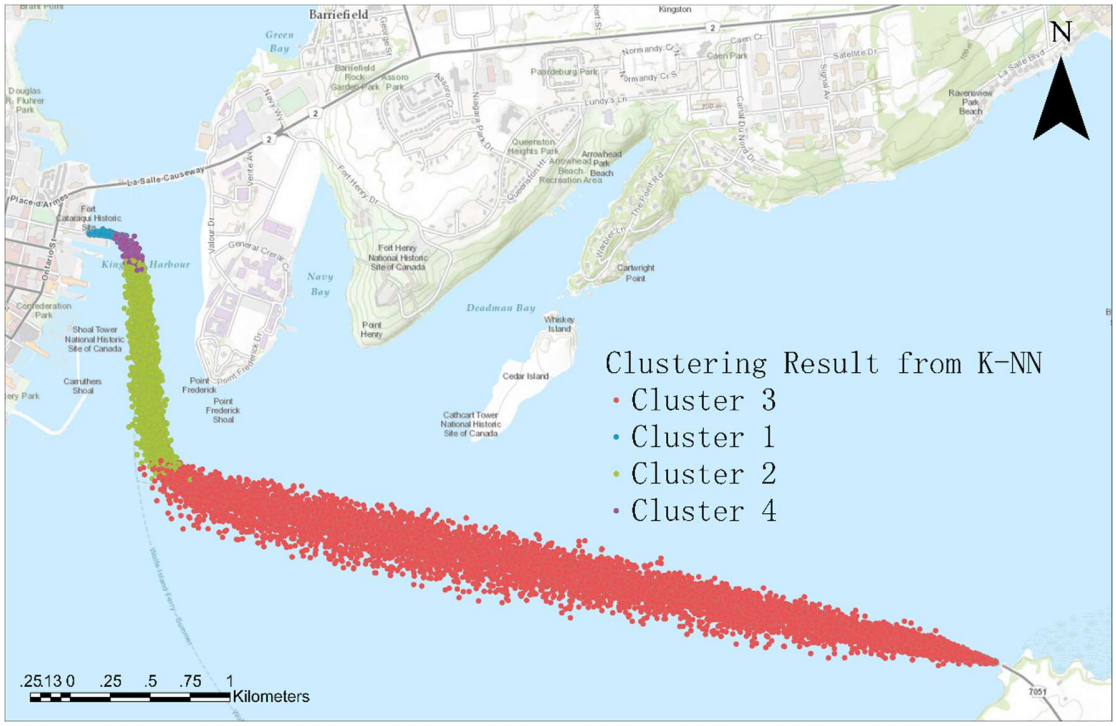
4.2.3.3 Performance of Using Supervised Learning to Classify New Observations

This section evaluates the proposed clustering method’s clustering performance on discovering clusters by comparing it with other supervised learning algorithms. **Figure 4–14** shows comparison results among supervised learning algorithms implementing on Dataset

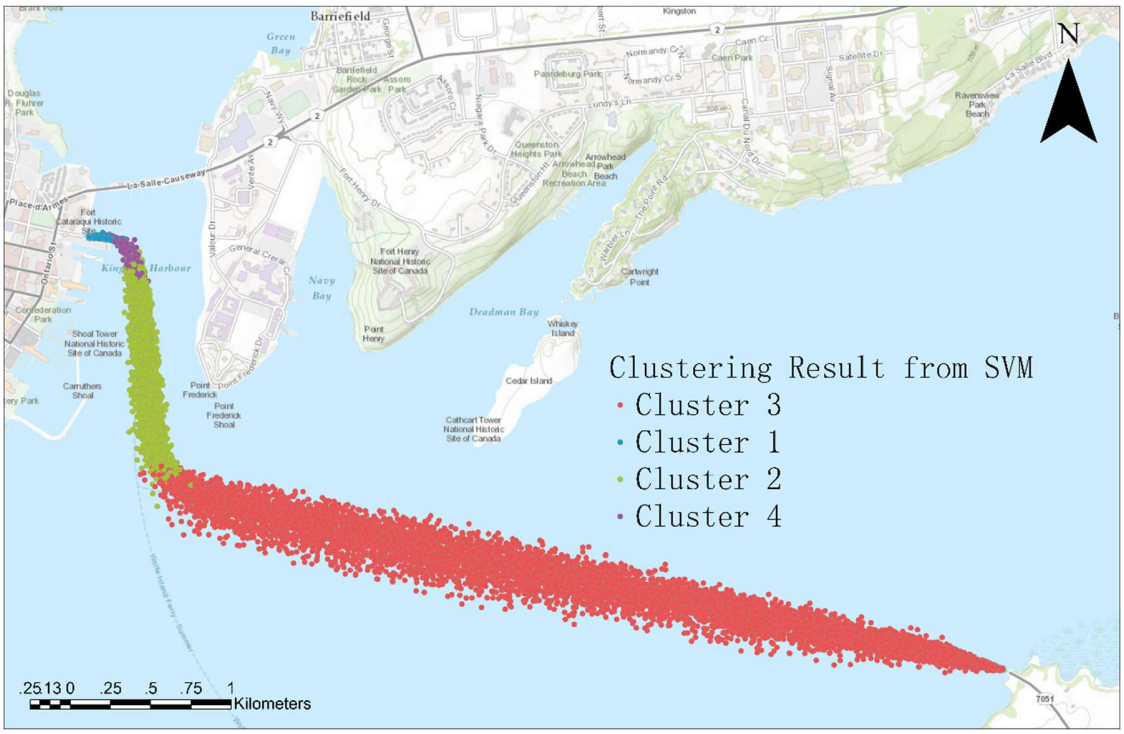
One. **Figure 4–14a** presents the synthesized ground truth for this comparison set. As previously mentioned in Section 3.1, the proposed clustering framework utilizes the unsupervised component on a portion of the raw data to discover the clusters first and then apply the supervised component to the rest of the data. By the same framework, the experiment uses 30% of the Dataset One as training data to train each supervised model and then test the clustering performance on the rest 70% data. Same to Section 4.2.3.1, this section focuses on evaluating the performance of discovering clusters, and the synthesized outliers are filtered before implementing the algorithms. **Figure 4–15** shows comparison results among supervised learning algorithms implementing on Dataset Two. Due to the smaller data size of Dataset Two, 40% of Dataset Two are used for training, and 60% of the data for testing. **Figure 4–15a** presents the synthesized ground truth of Dataset Two.



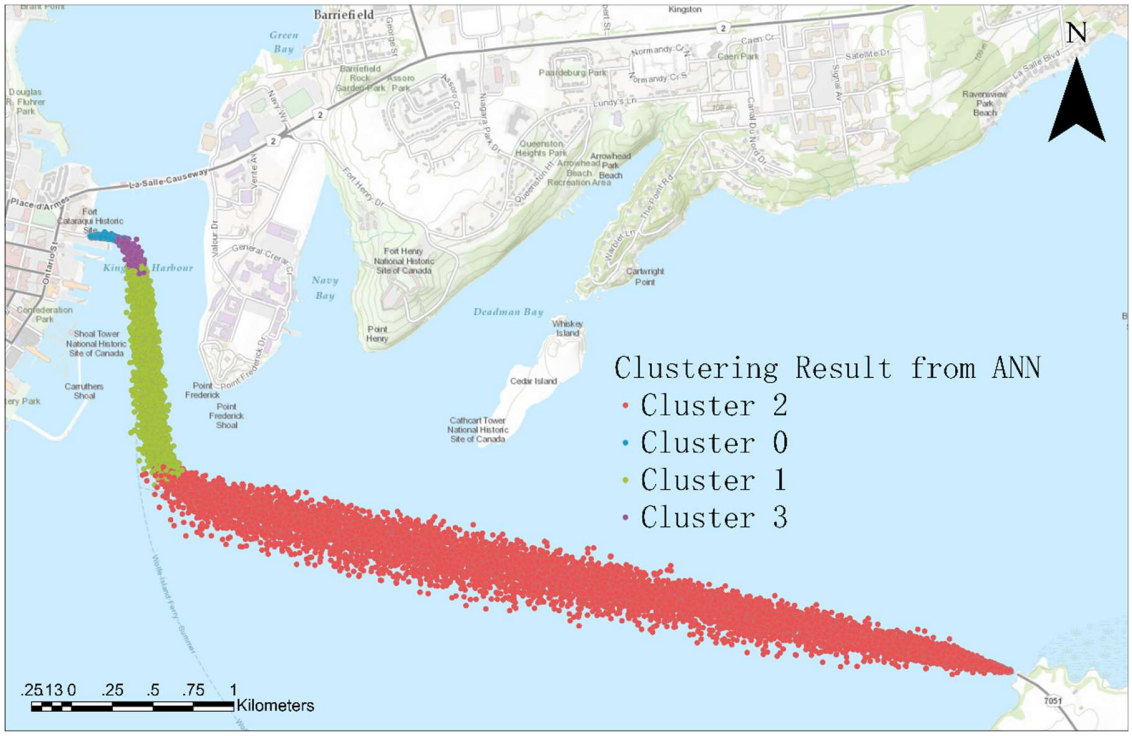
4-14a. Ground Truth of Dataset One



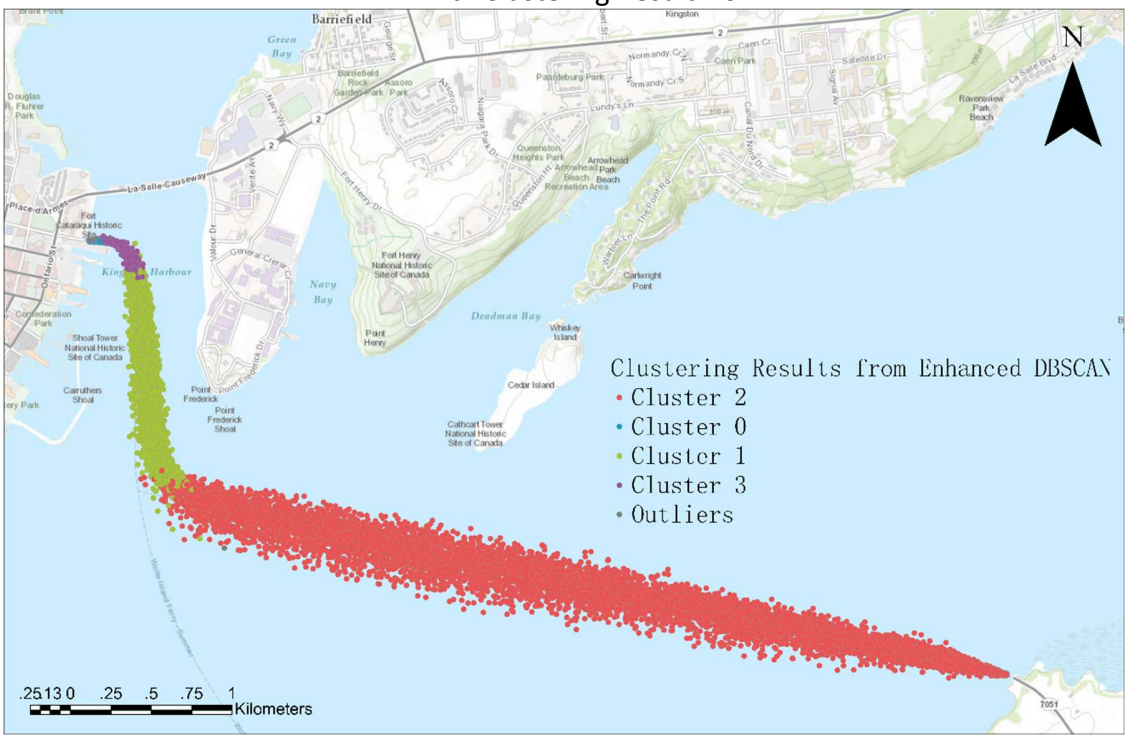
4-14b. Clustering Result from K-NN



4-14c. Clustering Result from SVM

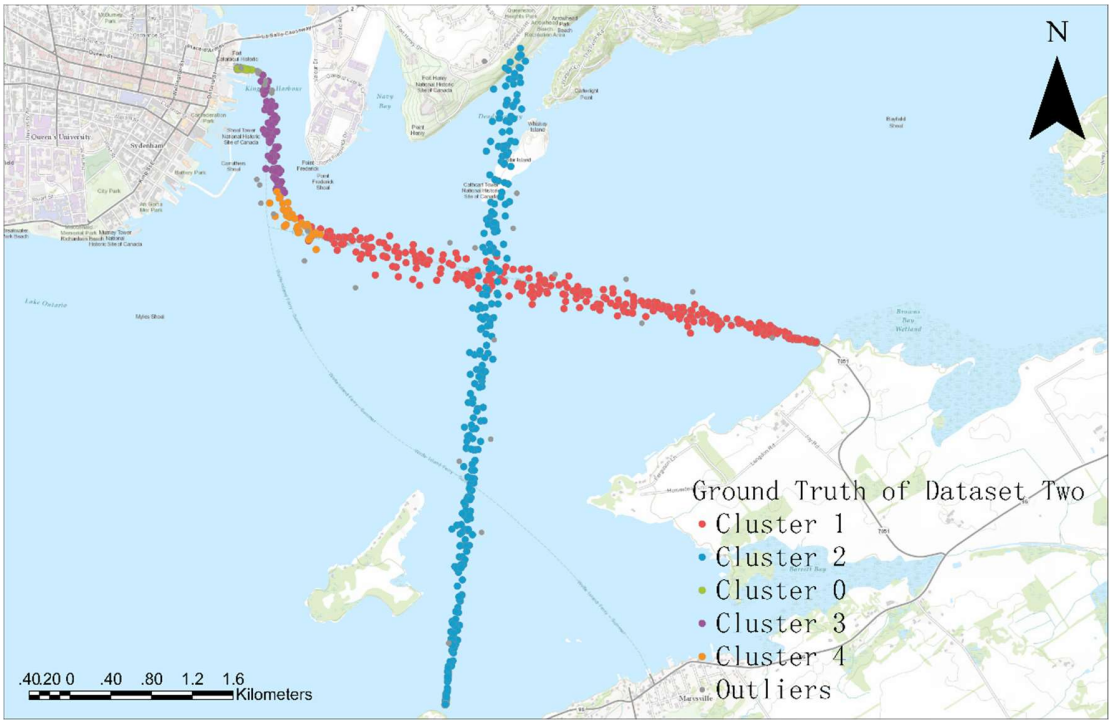


4-14d. Clustering Result from ANN

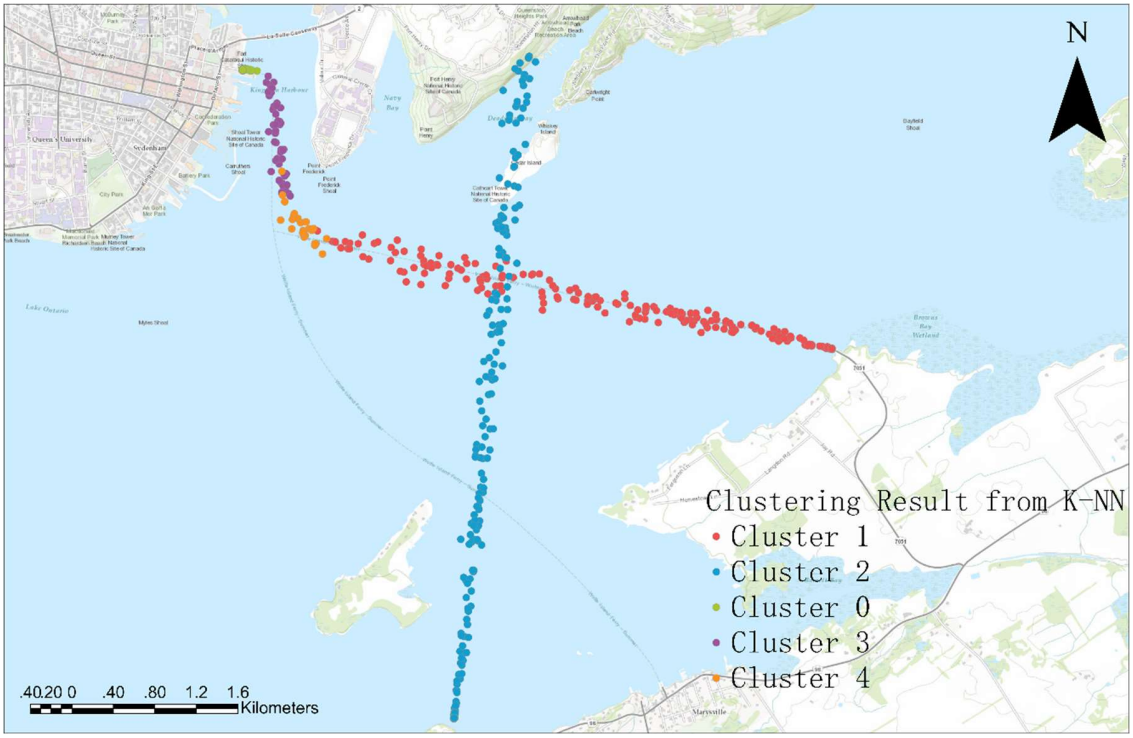


4-14e. Clustering Results from Enhanced DBSCAN

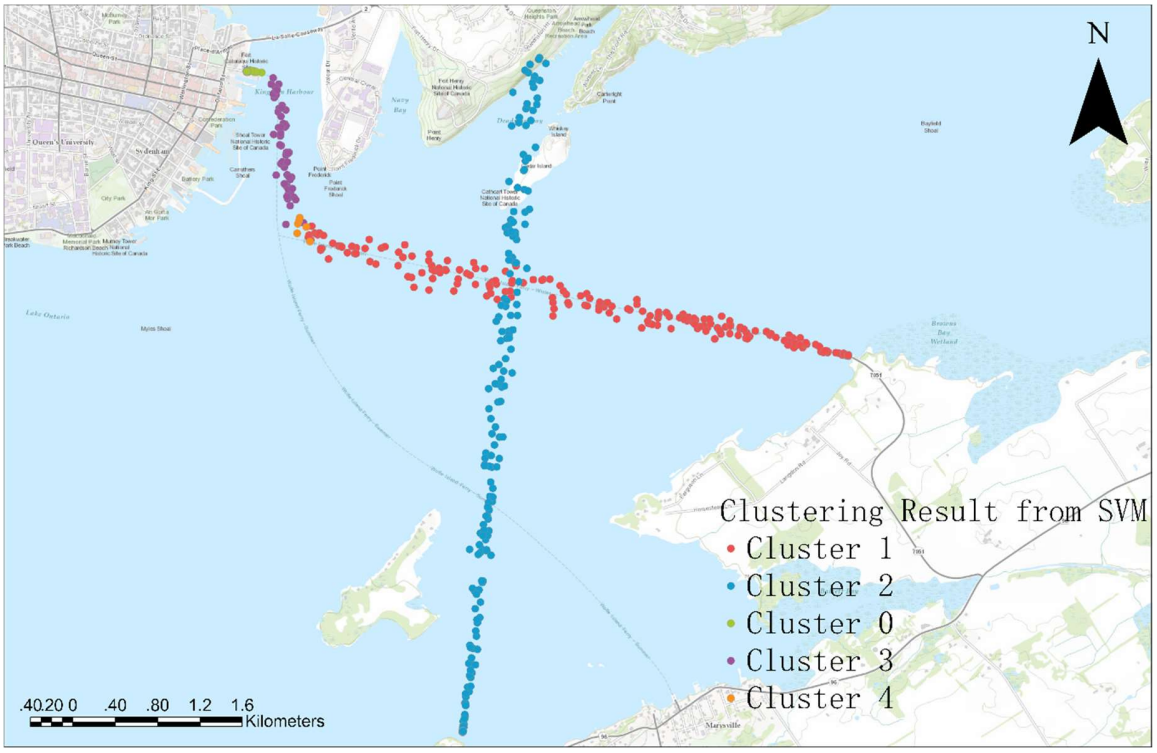
Figure 4–14. Comparing Enhanced DBSCAN's performance on discovering clusters from synthetic dataset one to ground truth and other supervised clustering methods



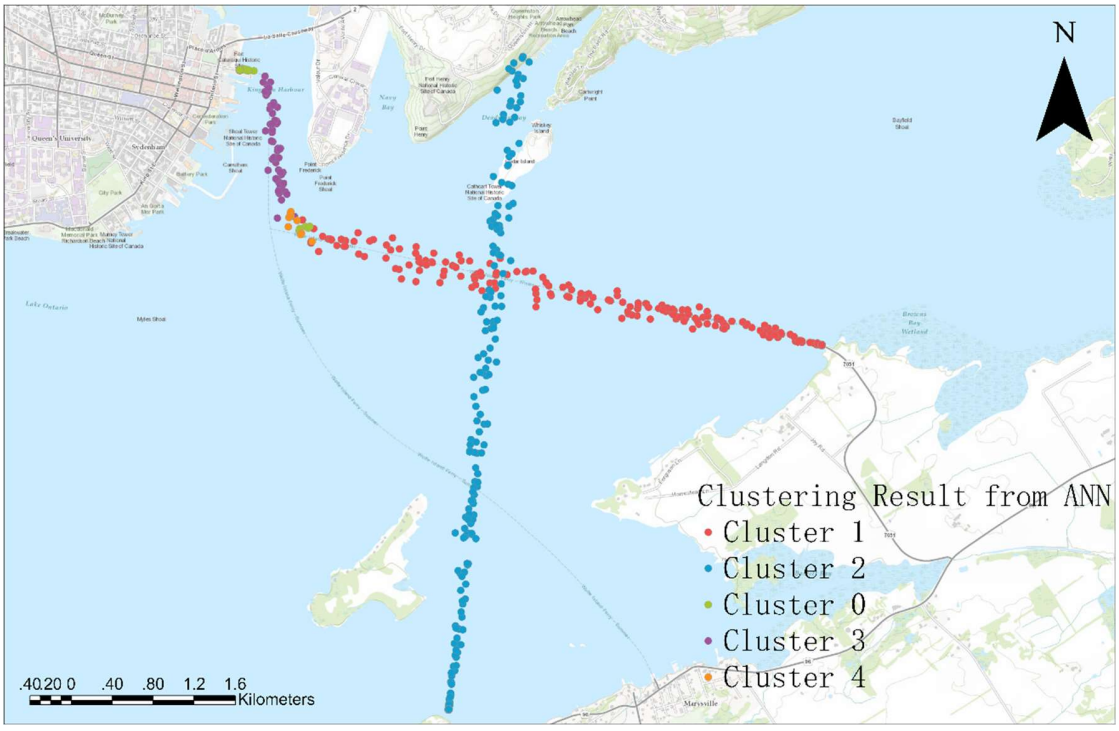
4-15a. Ground Truth of Dataset Two



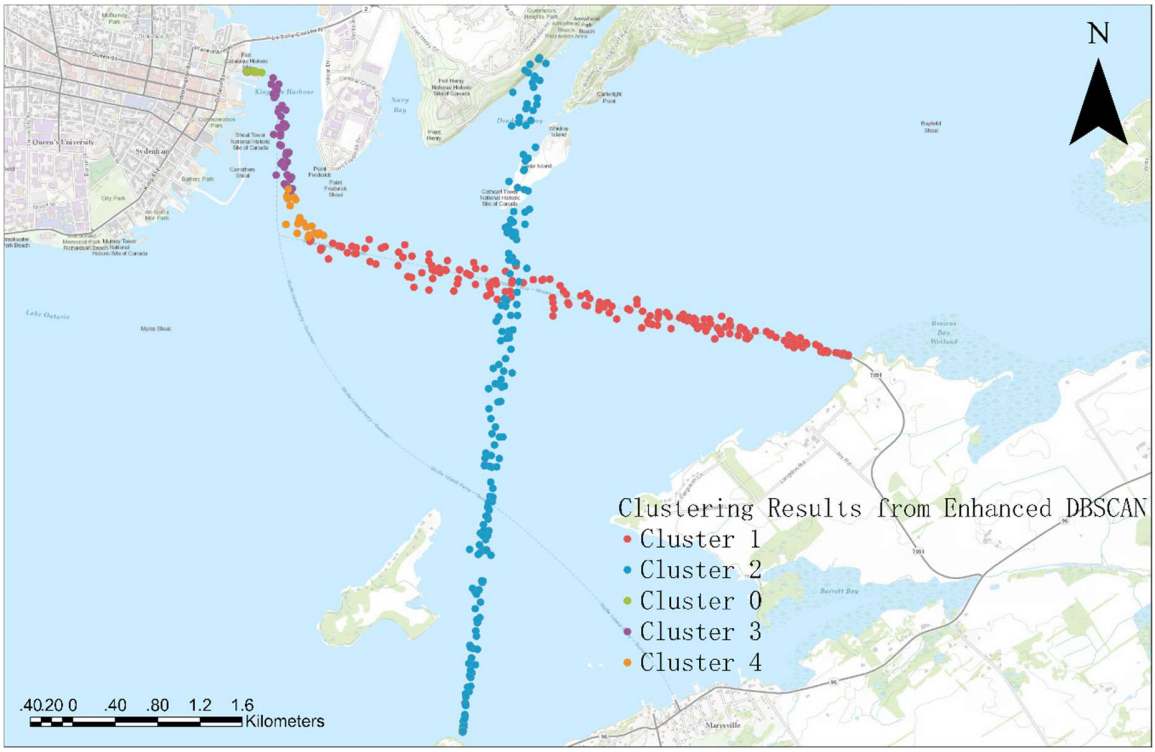
4-15b. Clustering Result from K-NN



4-15c. Clustering Result from SVM



4-15d. Clustering Result from ANN



4-15e. Clustering Results from Enhanced DBSCAN

Figure 4–15. Comparing Enhanced DBSCAN's performance on distinguishing intersections and discovering clusters from synthetic dataset two to ground truth and other supervised clustering methods

By visually evaluating the performance on classifying the observations into pre-defined clusters from **Figure 4–14** and **Figure 4–15** first, all supervised algorithms can get a similar outcome to the ground truth with training data. As shown in **Figure 4–14e** and **Figure 4–15e**, Enhanced DBSCAN still find outliers even though the outliers in the ground truth were filtered in the first place. The results can be justified by the nature of the supervised component in the enhanced DBSCAN algorithms. Based on the proposed parameter selection method mentioned in Section 3.4, the most deviated data beyond 1.5 times of IQR within the dataset will be considered outliers.

Besides, as mentioned in Section 4.2.1 and Section 4.2.2, the proposed clustering algorithm's performance is assessed using the selected metrics, external evaluation, and internal evaluation metrics. **Table 4–5** states the values of the clustering performance metrics of all clustering algorithms.

Table 4–5. Clustering performance evaluation of various supervised methods on distinguishing intersections and discovering clusters

Data Set	Algorithms	Estimated number of clusters	Entropy / Homogeneity	Purity / Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Information	F1 Score	Silhouette Coefficient	Davies-Bouldin Index	Calinski-Harabasz Index
	Ground Truth	4	1	1	1	1	1	1	0.659	0.791	32966.388
Data Set One	KNN	4	0.921	0.952	0.936	0.975	0.936	0.984	0.673	0.646	34590.951
	SVM	4	0.907	0.944	0.925	0.971	0.925	0.98	0.679	0.542	35517.235
	Enhanced DBSCAN	4	0.977	0.973	0.975	0.992	0.975	0.996	0.66	0.816	33429.567
	ANN	4	0.918	0.952	0.935	0.972	0.935	0.983	0.673	0.628	34638.614
	Ground Truth	4	1	1	1	1	1	1	0.581	0.527	892.718
Data Set Two	KNN	4	0.991	0.985	0.988	0.996	0.988	0.997	0.583	0.537	906.921
	SVM	4	0.947	0.981	0.963	0.979	0.963	0.984	0.566	0.498	824.776
	Enhanced DBSCAN	4	0.96	0.967	0.964	0.983	0.963	0.987	0.577	0.636	859.67
	ANN	4	0.955	0.977	0.966	0.989	0.965	0.987	0.569	0.497	868.113

Figure 4–16 to **Figure 4–19** visualize the values of each clustering performance metric. Various colors are used to represent different clustering methods. The x-axis represents external evaluation attributes, and the y-axis represents the corresponding score.

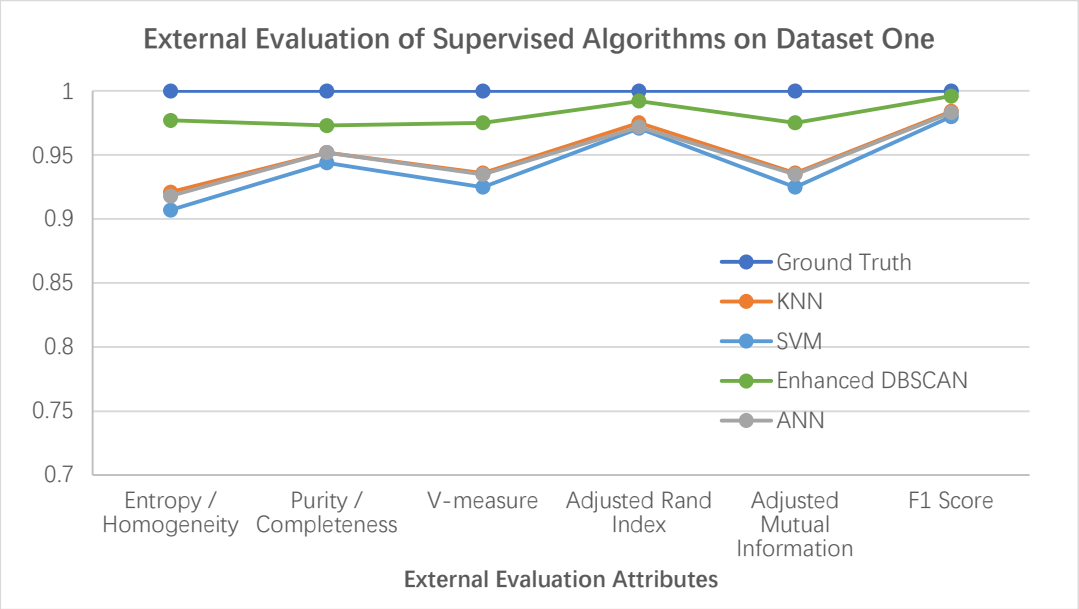


Figure 4–16. Performance evaluation of discovering clusters on supervised algorithms using external evaluation metrics on dataset one

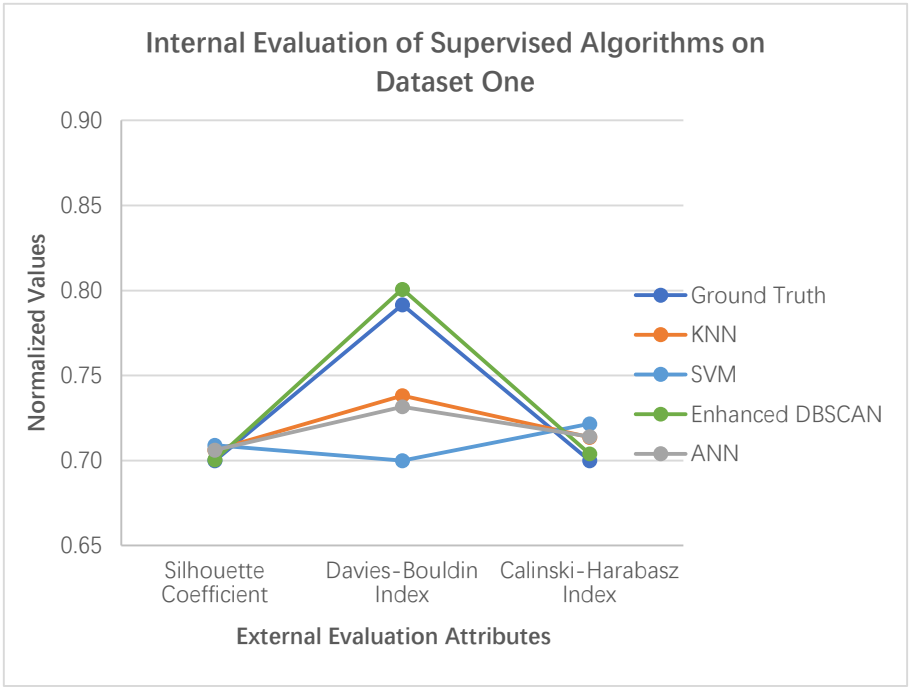


Figure 4–17. Performance evaluation of discovering clusters on supervised algorithms using internal evaluation metrics on dataset one

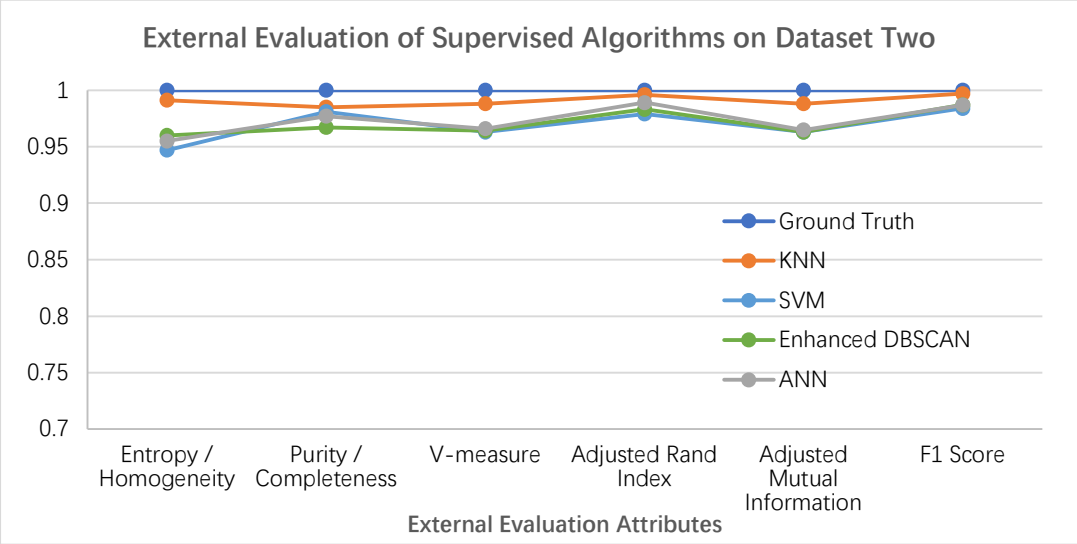


Figure 4–18. Performance evaluation of distinguishing intersections on supervised algorithms using external evaluation metrics on dataset two

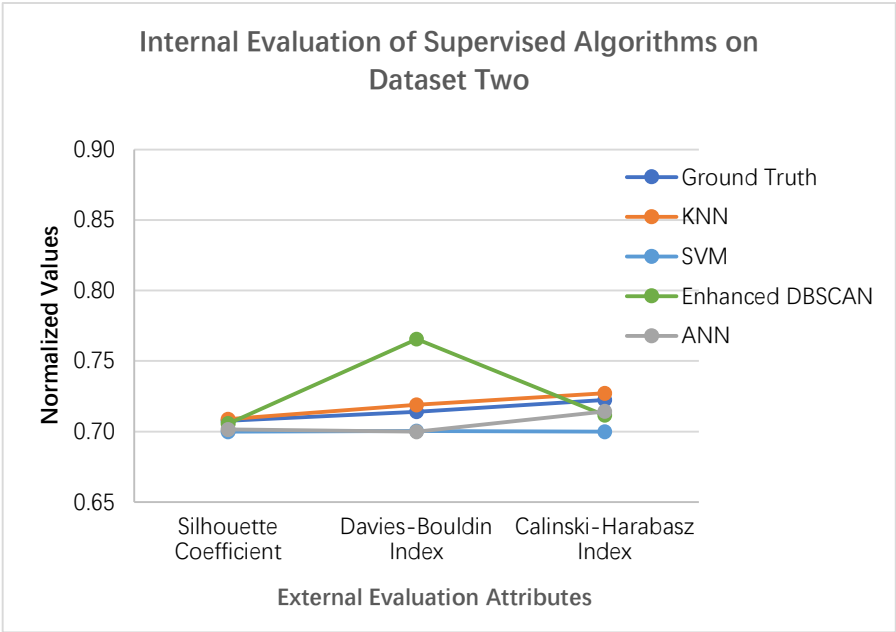


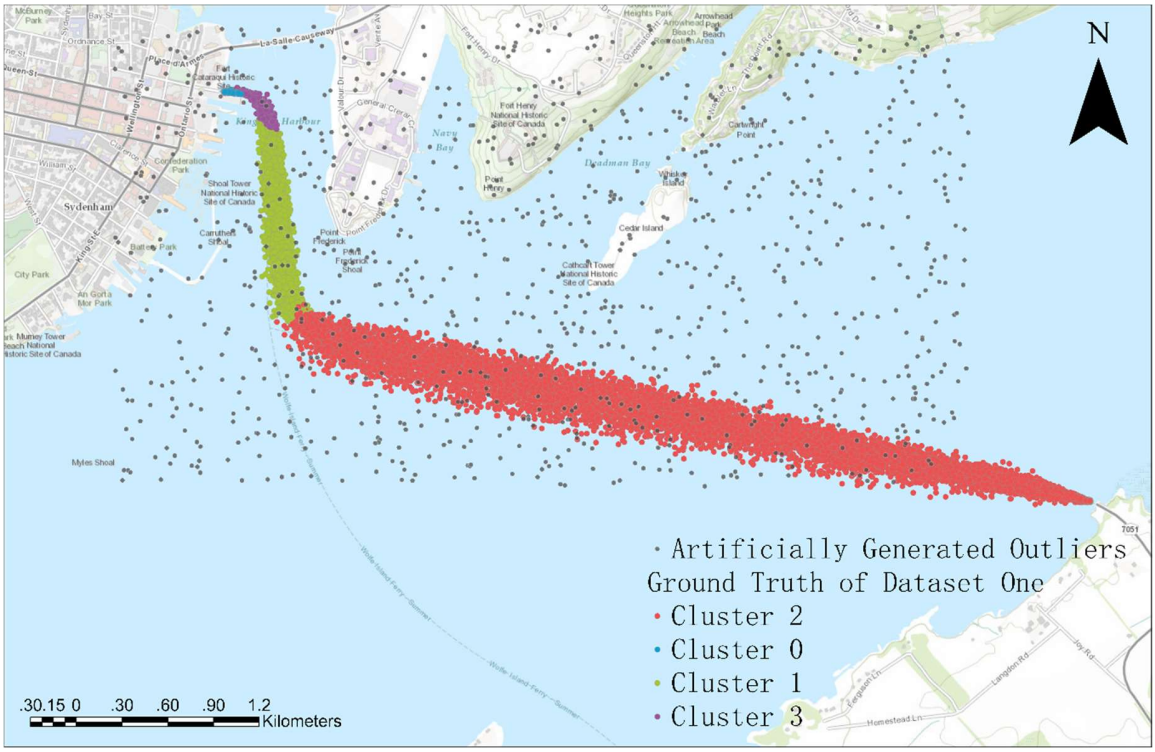
Figure 4–19. Performance evaluation of distinguishing intersections on supervised algorithms using internal evaluation metrics on dataset two

The results indicate that the proposed approach keeps high performance by analyzing both **Figure 4–16** to **Figure 4–19** and **Table 4–5**. According to **Figure 4–16,** the blue line represents the ground truth, and the green line represents the enhanced DBSCAN with a score very close to 1. The external evaluation results indicate that the proposed algorithm has good accuracy, homogeneity, and completeness since the scores are all higher than 0.95. Internal evaluation metrics' values were normalized into 0.5 – 1 so that all three internal evaluation metrics can share the same scale. According to **Figure 4–17**, the result from evaluating enhanced DBSCAN is very close to the ground truth with less than 0.1 difference on those metrics, suggesting that the result is similar to the ground truth and has good clustering performance. According to **Figure 4–18** and **Figure 4–19**, enhanced DBSCAN is not the best algorithm when implementing on Dataset Two. This can be explained by that enhanced DBSCAN finds outliers beyond filtered outliers. Since Dataset Two has a smaller data size, only a small variation may make a huge difference in the evaluation result even though enhanced DBSCAN gets higher than 0.96 for each external evaluation attribute.

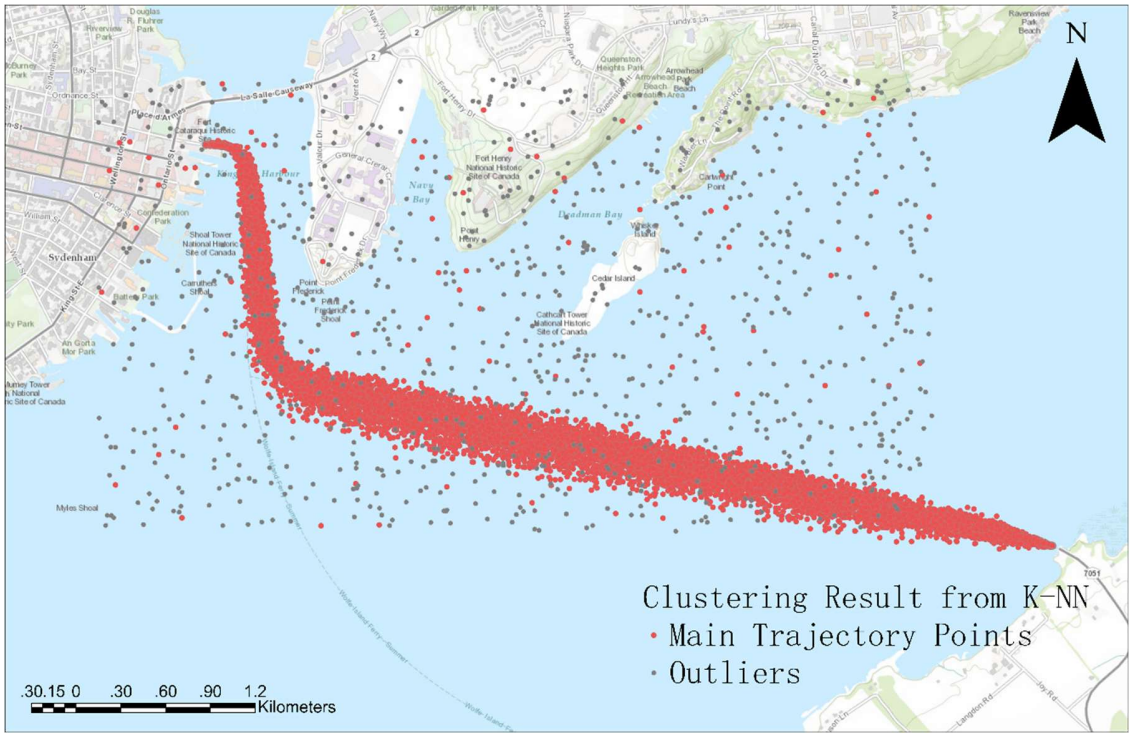
Overall, results from the enhanced DBSCAN satisfy the expectation that it keeps a high level of performance quality in terms of external evaluation metrics, compared to other supervised algorithms. The results from the enhanced algorithm also stay in a satisfying range concerning internal evaluation metrics.

4.2.3.4 Performance of Using Supervised Learning to Detect Outliers

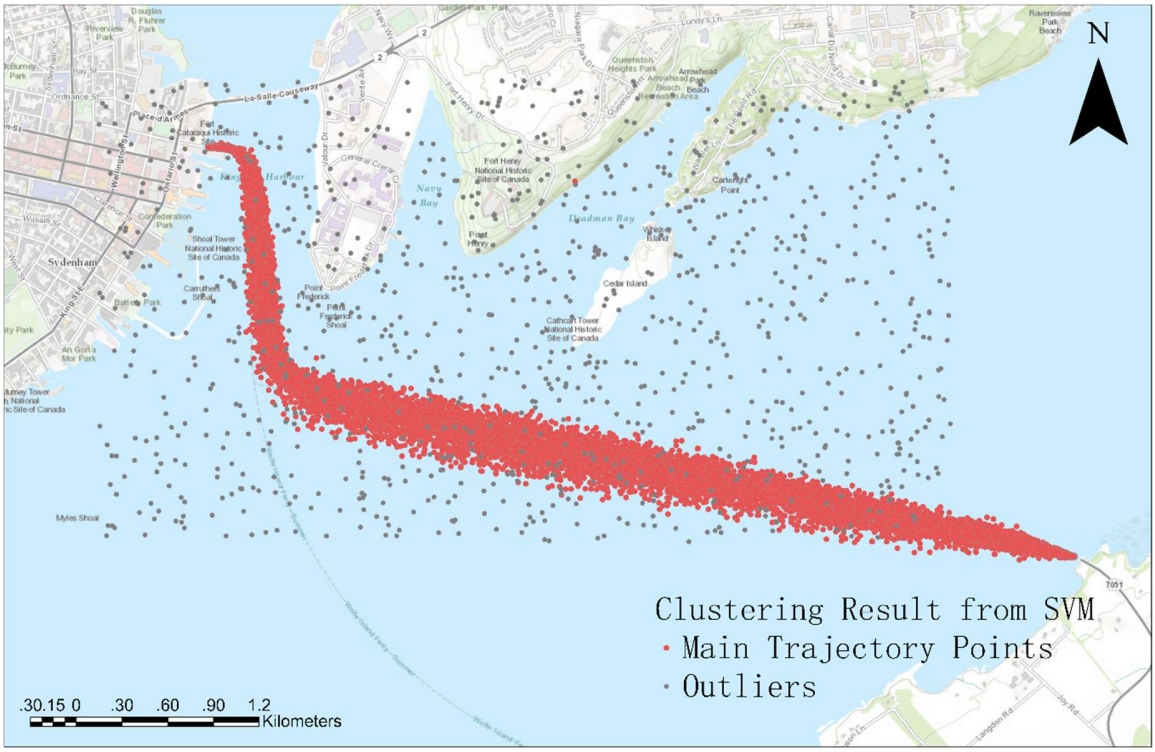
This section evaluates the proposed clustering method's clustering performance in detecting outliers by comparing it with other supervised learning algorithms. **Figure 4–20** shows comparison results among supervised learning algorithms implementing on Dataset One. **Figure 4–20a** presents the synthesized ground truth for this comparison set. Same to Section 4.2.3.3, the experiment uses 30% of the Dataset One as training data and test the clustering performance on the rest 70% of the data. One thousand multivariate outliers were artificially synthesized around the main trajectory. To compare the results from various supervised clustering algorithms on the same level, the experiment tests binary classification performance. Due to enhanced DBSCAN's ability to detect outliers is based on each cluster, in this experiment, the enhanced DBSCAN is doing two tasks simultaneously, which are discovering clusters and detecting outliers. So, the training data for each algorithm is different in this experiment. Training data for KNN, SVM, and ANN only have two labels, 1 and 0, representing the main trajectory and outliers. Enhanced DBSCAN in this experiment uses the same training data as Section 4.2.3.1 and does not use any synthetic outliers as training data.



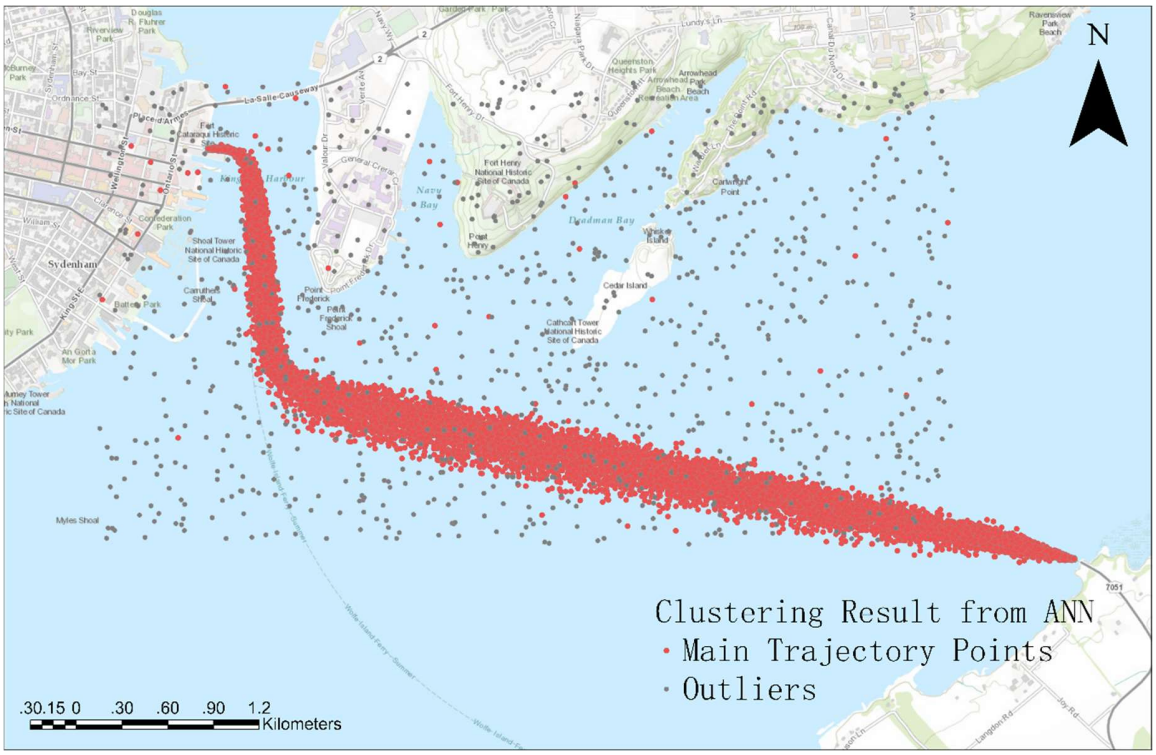
4-20a. Ground Truth of Dataset One



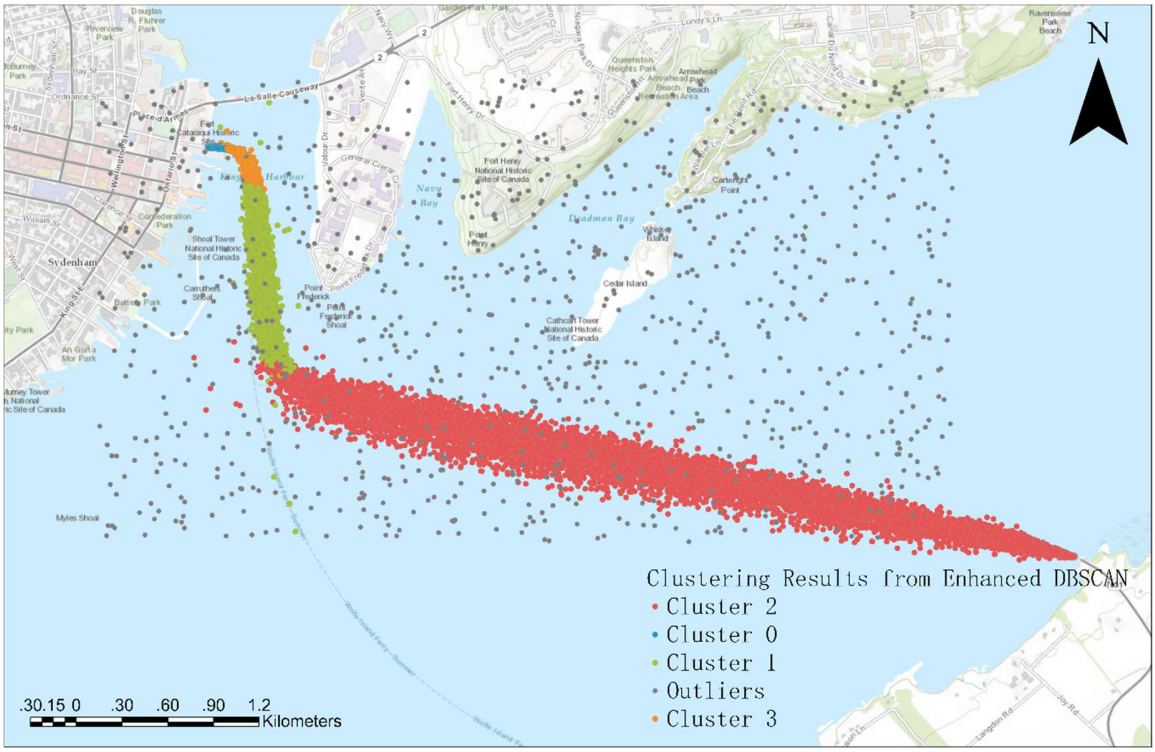
4-20b. Clustering Result from K-NN



4-20c. Clustering Result from SVM



4-20d. Clustering Result from ANN



4-20e. Clustering Results from Enhanced DBSCAN

Figure 4–20. Comparing Enhanced DBSCAN's performance on outlier detection from synthetic dataset one to ground truth and other supervised clustering methods

By first visually evaluating the performance on discovering the clusters from **Figure 4–20**, SVM results have the best performance to differentiate outliers from the main trajectory. Both KNN and ANN identified some outliers as main trajectory points by mistake. Enhanced DBSCAN uses a Mahalanobis distance metric to measure the distance between points and clusters. Mahalanobis distance considers correlation within the cluster. As shown in **Figure 4–20e**, some outliers at the same line with pre-defined clusters are identified as part of those clusters.

Besides, the proposed clustering algorithm's performance is assessed by the selected metrics, both external evaluation, and internal evaluation metrics. **Table 4–6** states the values of the clustering performance metrics of all clustering algorithms.

Table 4–6. Clustering performance evaluation of various supervised methods on outlier detection

Data Set	Algorithms	Estimated number of noise points	Entropy / Homogeneity	Purity / Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Information	F1 Score	Silhouette Coefficient	Davies-Bouldin Index	Calinski-Harabasz Index
	Ground Truth	1000	1	1	1	1	1	1	0.74	0.87	13213.074
Data Set One	KNN	909	0.857	0.921	0.888	0.946	0.888	0.996	0.747	0.823	13107.855
	SVM	998	0.995	0.997	0.996	0.999	0.996	1	0.741	0.869	13227.934
	Enhanced DBSCAN	975	0.825	0.84	0.932	0.925	0.832	0.994	0.725	0.945	10899.795
	ANN	956	0.923	0.955	0.939	0.975	0.939	0.998	0.742	0.843	13144.003

Figure 4–21 and **Figure 4–22** visualize the values of each clustering performance metric. Various colors are used to represent different clustering methods. The x-axis represents external evaluation attributes, and the y-axis represents the corresponding score.

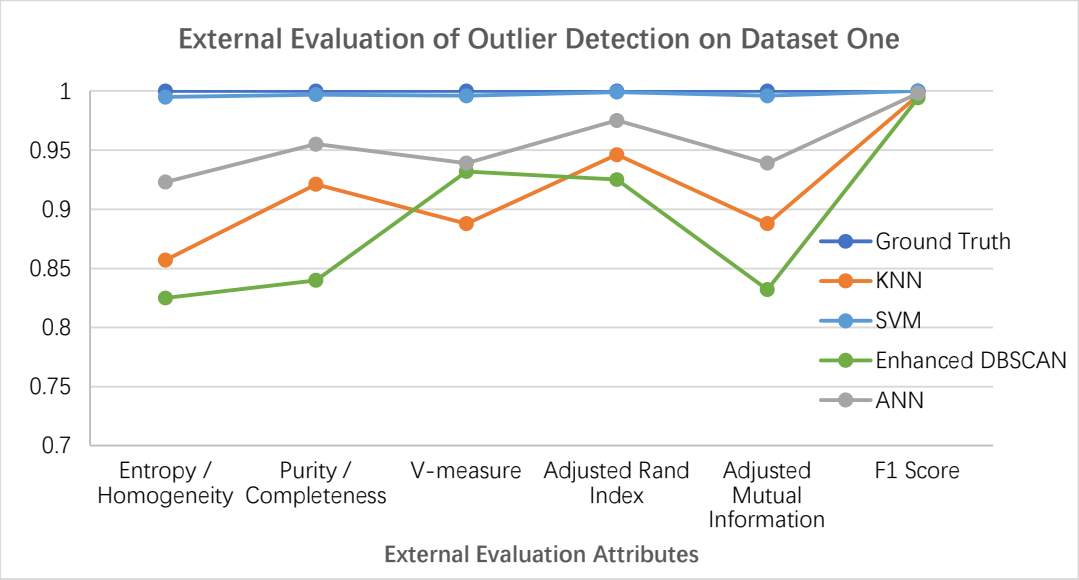


Figure 4–21. Performance evaluation of outlier detection on supervised algorithms using external evaluation metrics on dataset one

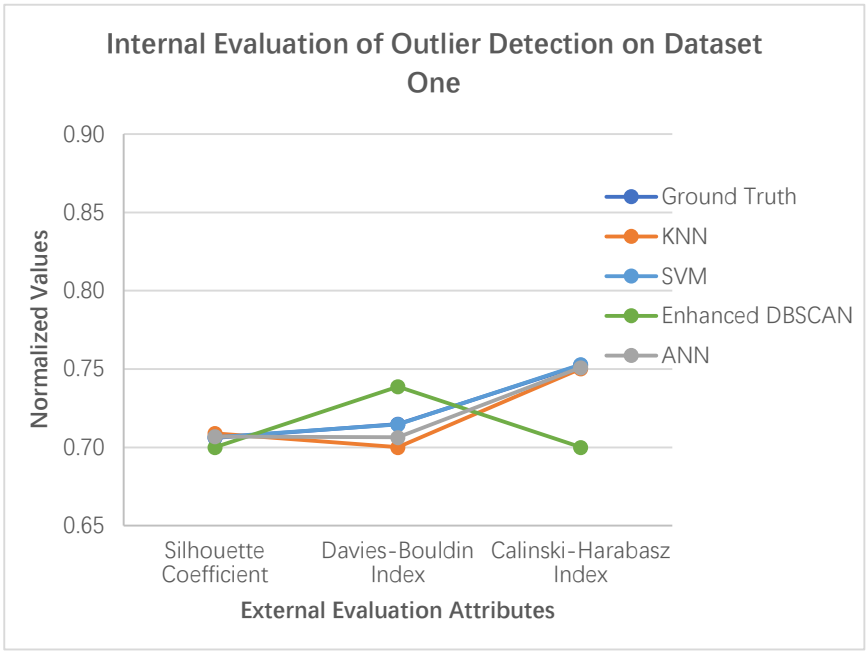


Figure 4–22. Performance evaluation of outlier detection on supervised algorithms using internal evaluation metrics on dataset one

The results indicate that the proposed approach has good performance in detecting outliers based on the external evaluation by analyzing **Figure 4–21**, **Figure 4–22**, and **Table 4–6**. According to the figures and tables, the SVM has the highest score. Even though the enhanced DBSCAN has a high F-1 score, indicating good accuracy, entropy and purity scores, even still at a high level, are relatively lower. This can be explained by that enhanced DBSCAN finds more outliers than expected. Besides, enhanced DBSCAN is doing two tasks simultaneously. Thus, the evaluation favors other algorithms. Same to Section 4.2.3.1, the values of internal evaluation metrics were normalized into 0.5 – 1 so that all three internal evaluation metrics can share the same scale. The results of internal evaluation methods on other algorithms are compared with the ground truth. According to **Figure 4–22**, the enhanced DBSCAN is different from the results from other clustering methods. This can also be explained by that enhanced DBSCAN finds more outliers than expected. Thus, indicated by this experiment, the enhanced DBSCAN keeps a good performance quality level on detecting outliers among other supervised algorithms, satisfying the expectation.

4.2.4 Sensitivity Analysis and Validation

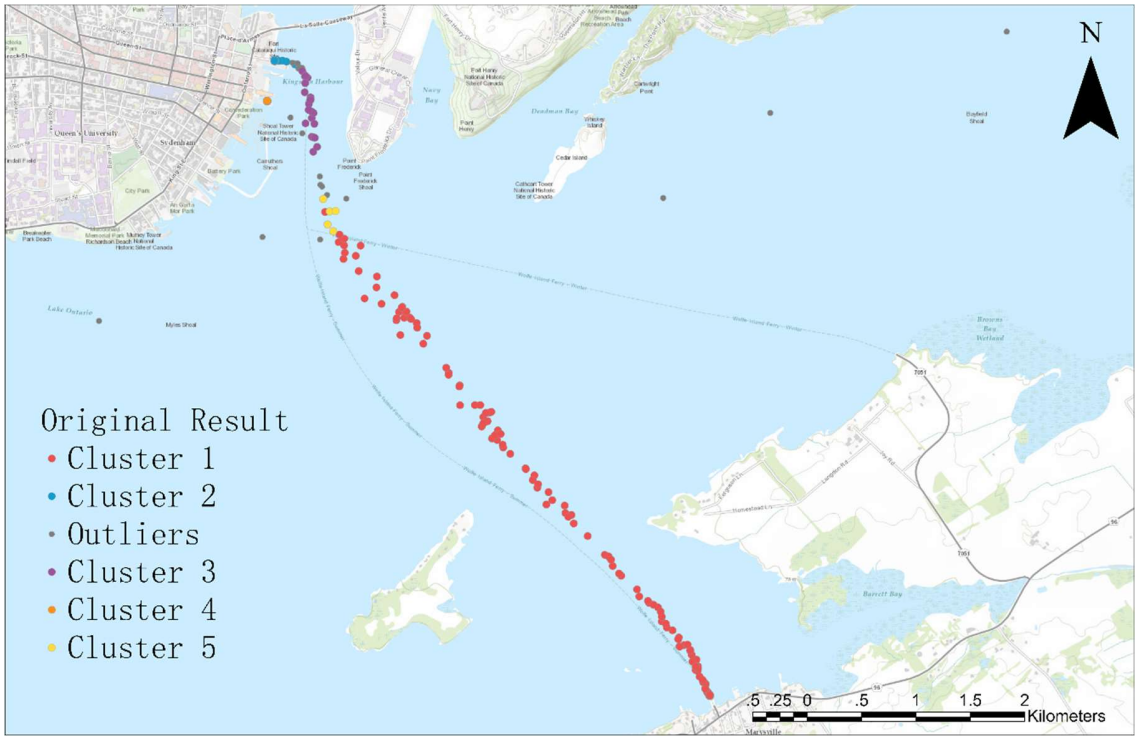
This section evaluates the validity of the proposed method for the Auto-Selection Parameters of the Enhanced DBSCAN. This experiment variates the parameters to obtain a set of results with different sets of parameters. Through analyzing the results, how those parameters can influence the results are analyzed. The purposed of this sensitivity analysis is to test whether the parameters recommended by auto-selection method is a good initial

setting, and whether the clustering result will be still valid when parameters are variated. This experiment's raw dataset is three-day data in regions between Kingston to Port at Wolfe Island, as shown in **Figure 4–23a**. The proposed parameters auto-selection method was implemented on this dataset and the following **Table 4–7** presents the parameters recommended by the method.

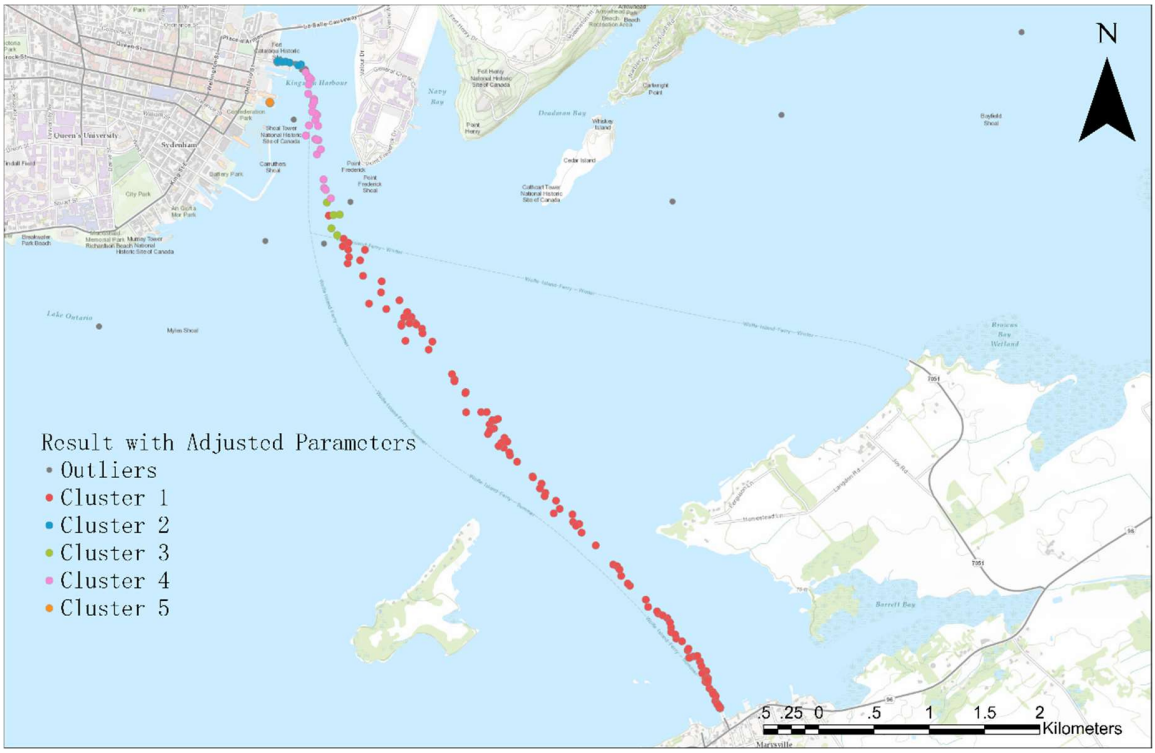
Table 4–7. The recommended values of the required parameters to be used in the Enhanced DBSCAN unsupervised component

Parameters	Recommended Values
MinPts	**10**
Eps	**0.287333**

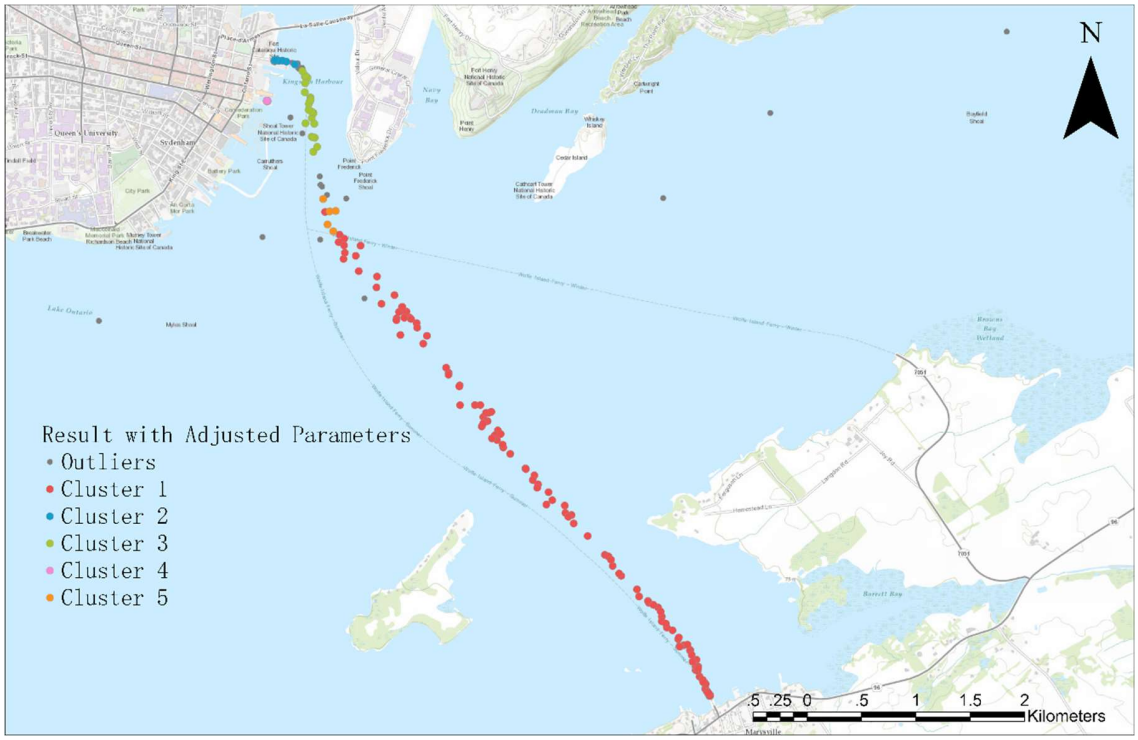
The experiment adjusts the parameters by 20% up and down separately and then applies the adjusted parameters to the enhanced DBSCAN. The comparison results are then shown as followed in **Figure 4–23**.



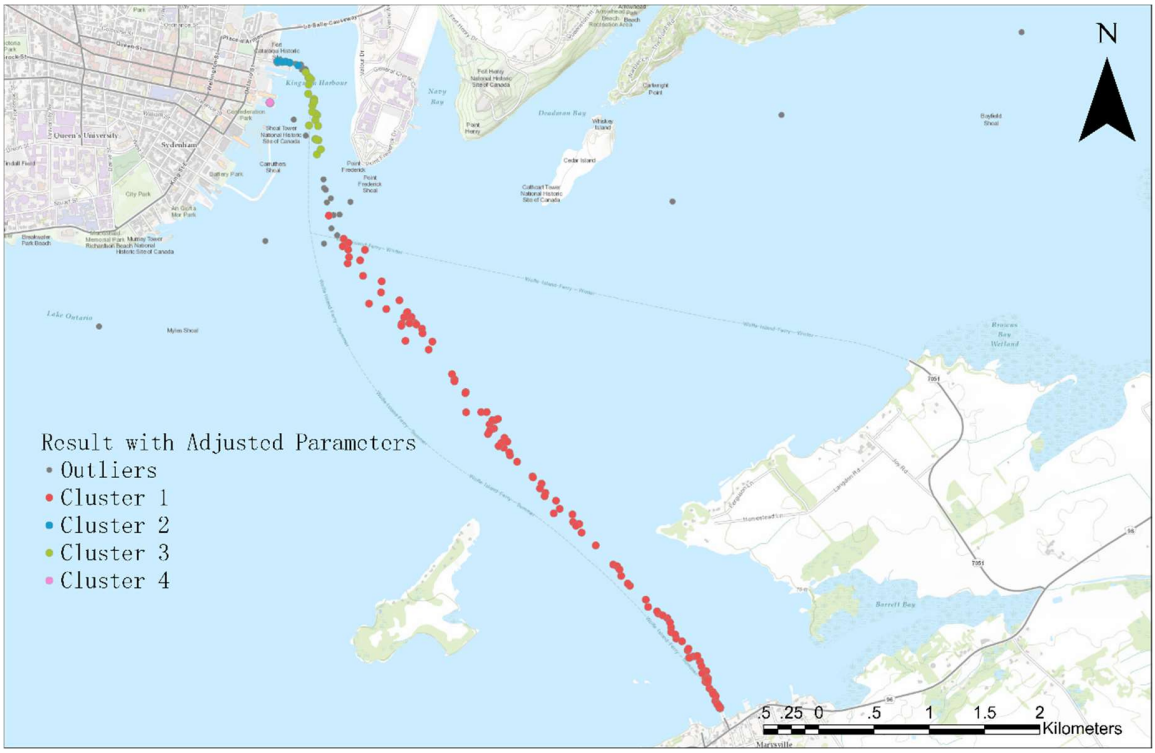
4-23a. Original Result (MinPts = 10, Eps = 0.2873)



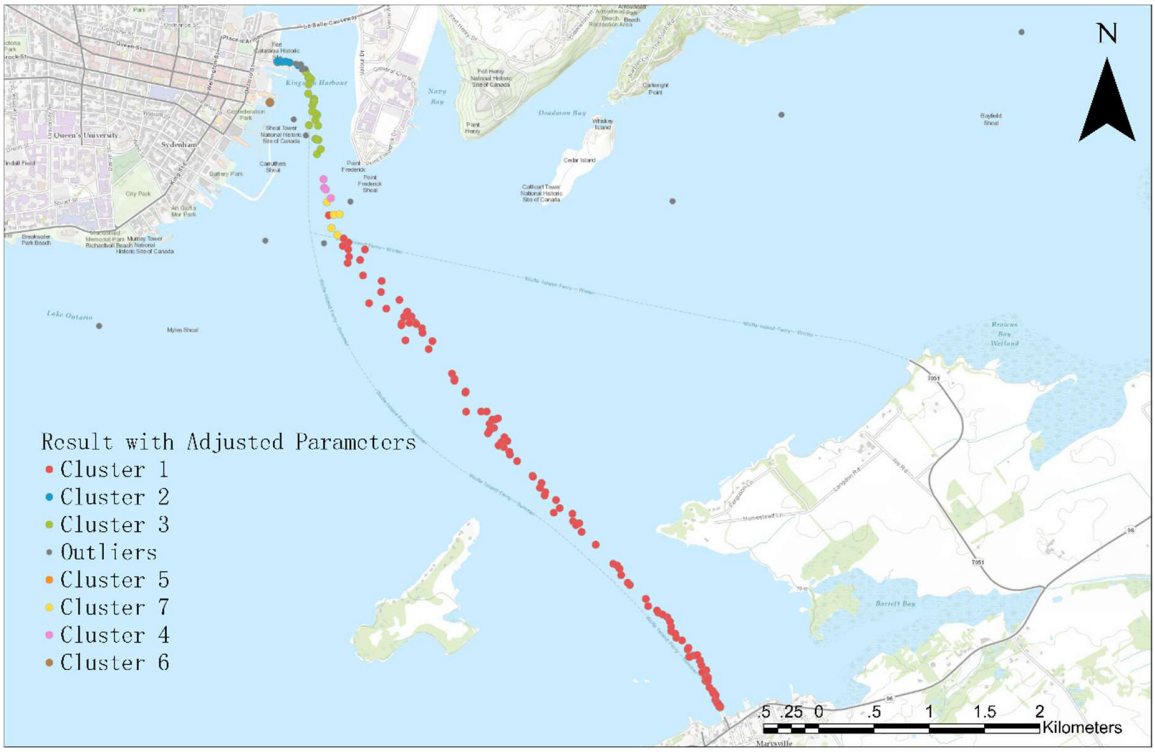
4-23b. Result with Adjusted Parameters (MinPts = 10, Eps = 0.3448)



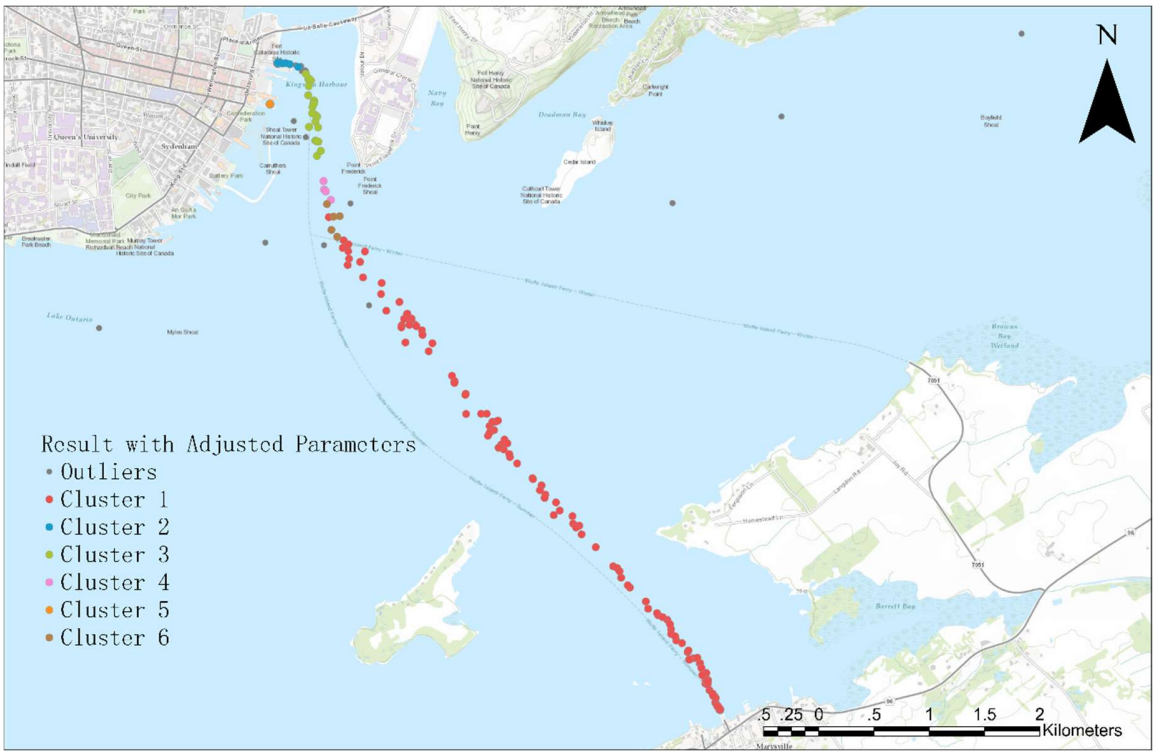
4-23c. Result with Adjusted parameters (MinPts = 10, Eps = 0.2394)



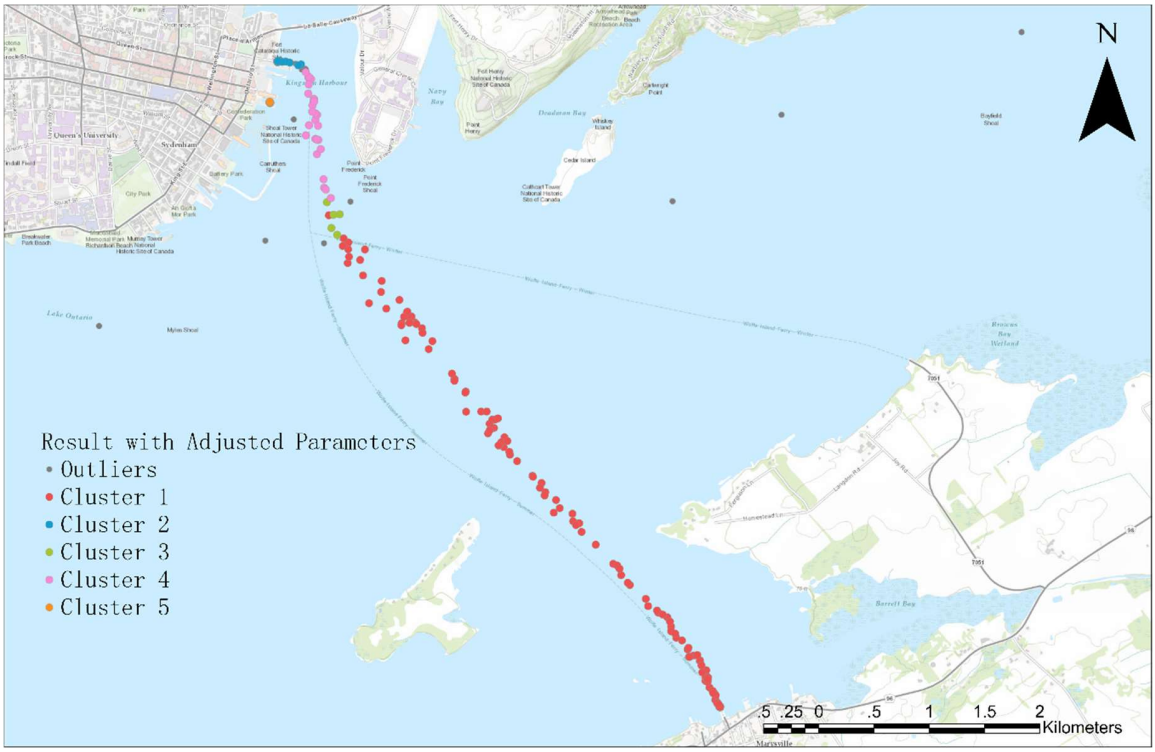
4-23d. Result with Adjusted parameters (MinPts = 12, Eps = 0.2873)



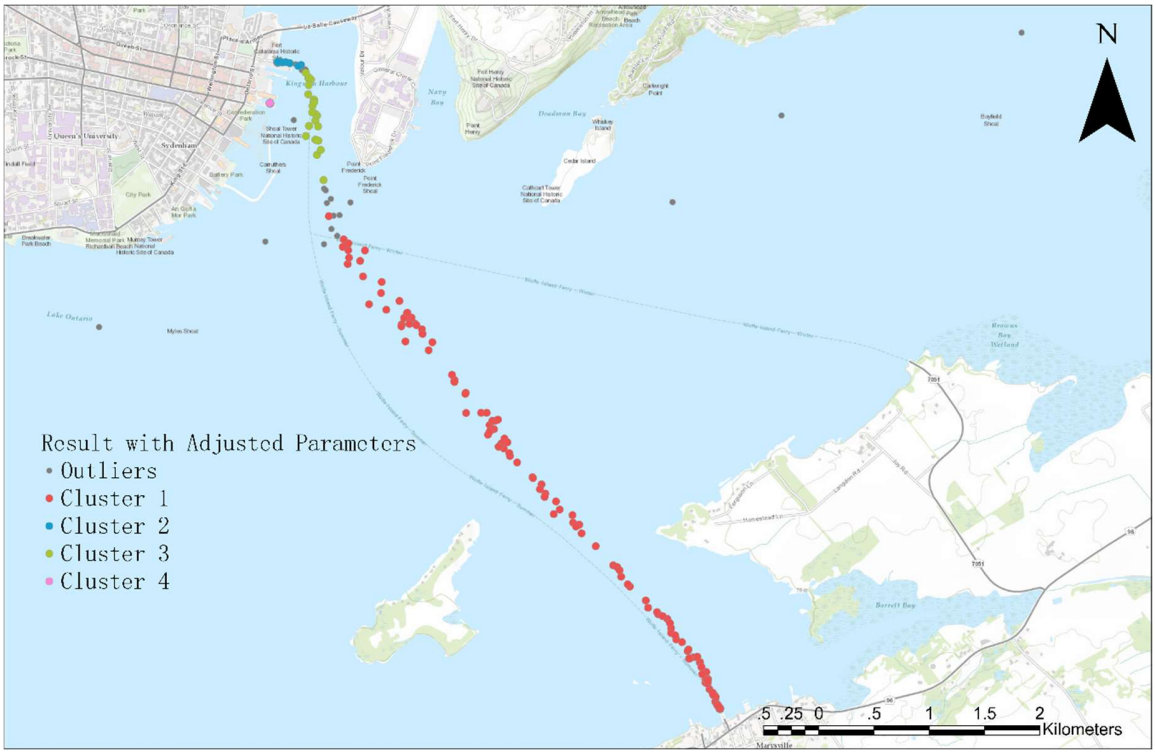
4-23e. Result with Adjusted Parameters (MinPts = 8, Eps = 0.2873)



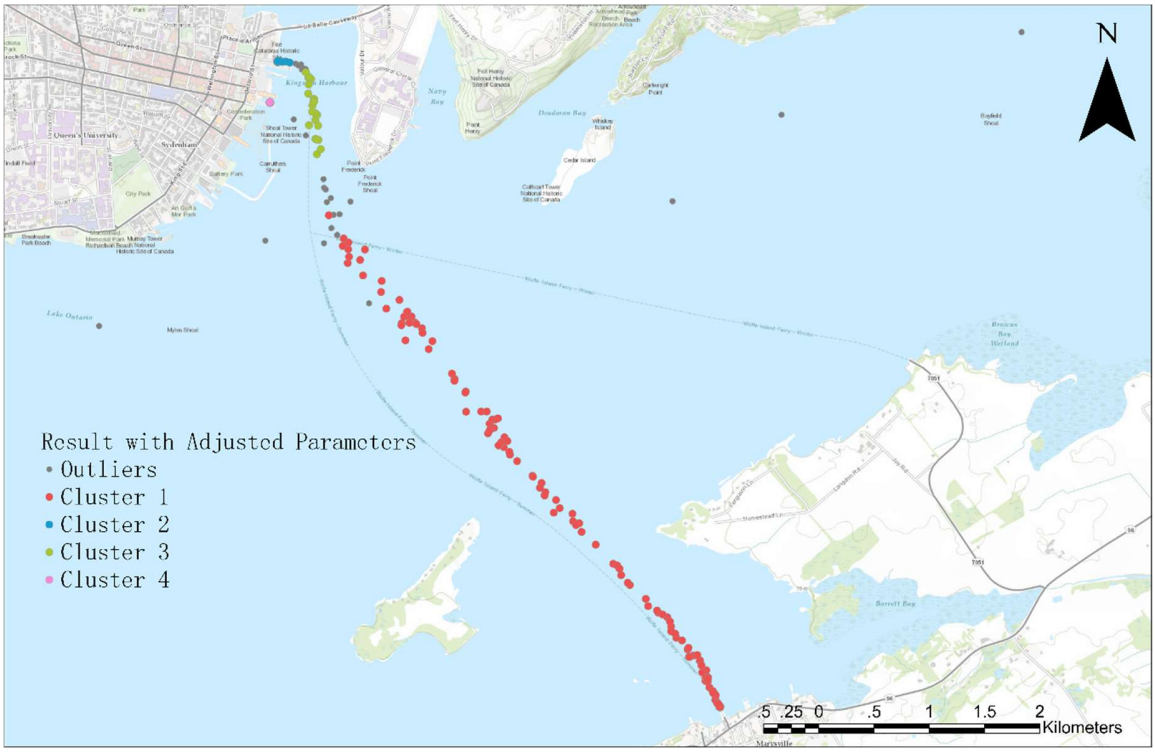
4-23f. Result with Adjusted parameters (MinPts = 8, Eps = 0.2394)



4-23g. Result with Adjusted parameters (MinPts = 8, Eps = 0.3448)



4-23h. Result with Adjusted parameters (MinPts = 12, Eps = 0.3448)



4-23i. Result with Adjusted parameters (MinPts = 12, Eps = 0.2394)

Figure 4–23. Comparison results by variating the parameters by 20%

MinPts is the parameter to limit the number of the core points used to grow the clusters. The larger the *MinPts* is, the smaller the number of the core points is, and the size of the clusters will be smaller, making more points to be identified as outliers. *Eps* is the parameter to determine neighborhoods. The smaller the *Eps* is, the fewer neighborhoods can be clustered together, making more clusters discovered. So **Figure 4–23f,h,i,** and **g** are the four extreme results: **Figure 4–23f** is the result with the most clusters; **Figure 4–23h** is the result with the least clusters; **Figure 4–23i** is the result with the most outliers; **Figure 4–23g** is the results with the least outliers.

Through first visually analyzing the comparison group, including those four extreme results, it can be concluded that the parameters recommended by the auto-selection method are accurate, and the algorithm is robust from variating the parameters. Basically, all results successfully, at least partially, in discovering the corresponding clusters and detecting outliers. All results have detected the port area. Besides, as shown in **Figure 4–24** and **Table 4–8**, at least four clusters (stages) have been defined by those results. **Figure 4–24** shows that even the worst parameters selection still has a higher than 0.9 average score, showing the results' quality and the algorithm's robustness.

Table 4–8. Clustering performance evaluation of various parameters

Data Set	Parameters	Estimated number of clusters	Estimated number of noise points	Entropy / Homogeneity	Purity / Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Information	F1 score
	MinPts = 10, Eps = 0.2873	5	56	1	1	1	1	1	1
Data Set between Kingston to Port at Wolfe Island	MinPts = 10, Eps = 0.3448	5	32	0.91	0.931	0.92	0.973	0.92	0.961
	MinPts = 10, Eps = 0.2394	5	60	0.974	0.968	0.971	0.988	0.971	0.994
	MinPts = 12, Eps = 0.2873	4	66	0.958	0.957	0.958	0.988	0.957	0.984
	MinPts = 8, Eps = 0.2873	7	40	0.939	0.949	0.944	0.982	0.943	0.974
	MinPts = 8, Eps = 0.2394	6	52	0.937	0.933	0.935	0.979	0.935	0.981
	MinPts = 8, Eps = 0.3448	5	32	0.91	0.931	0.92	0.973	0.92	0.961
	MinPts = 12, Eps = 0.3448	4	60	0.922	0.923	0.922	0.98	0.922	0.974
	MinPts = 12, Eps = 0.2394	4	72	0.925	0.919	0.922	0.972	0.922	0.974

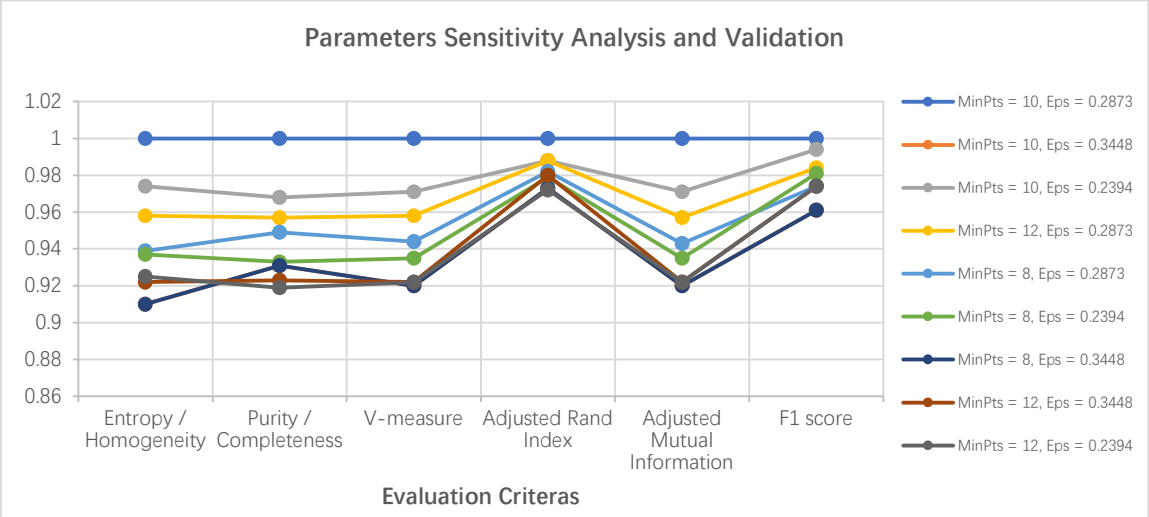


Figure 4–24. Clustering performance evaluation of various parameters using external evaluation metrics for sensitivity analysis

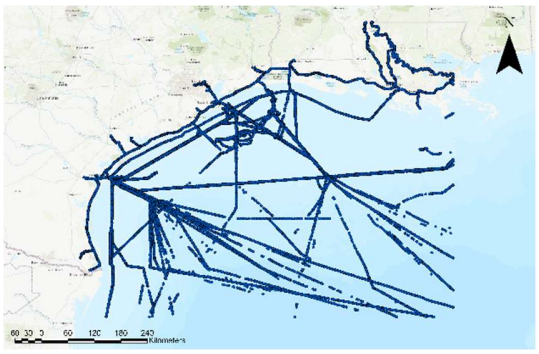
In summary, this section proves that the auto-selection method gives good recommendations, and they have a certain level of tolerance of variation and uncertainty. The proposed parameter auto-selection method can be used in large scale big data.

4.3 Case Studies and Results

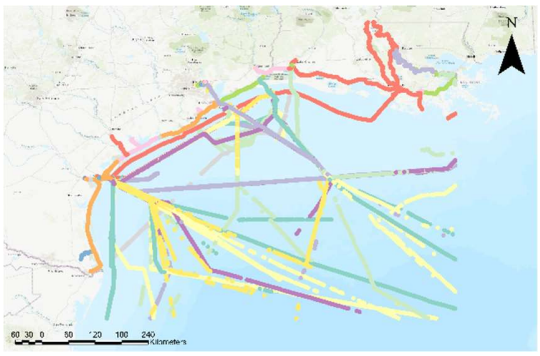
4.3.1 Gulf of Mexico AIS Big Data

The proposed clustering algorithm has been applied to big data in the Gulf of Mexico Region, and a model to monitor vessels in that region has been generated. In this research thesis, we accessed around 200 MB of open-sourced data in the Gulf of Mexico Region

(*MarineCadastre.gov | Vessel Traffic Data*, no date). The data is from one month of January 2017 and contains data of 70 MMSI. The data contains around 1.2 million trajectory points to be clustered. The raw data has been visualized in **Figure 4–25**. **Figure 4–25a** presents all the raw AIS point data, and **Figure 4–25b** visualized the data in various colors for corresponding MMSI.



4-25a. Raw AIS Point Data in the Gulf of Mexico Region



4-25b. Raw Trajectory Data by MMSI in the Gulf of Mexico Region

Figure 4–25. Raw AIS data and raw trajectory data in the Gulf of Mexico Region

As mentioned in Section 3.1, the enhanced DBSCAN clustering method has been implemented on the data of each MMSI. By this step, 2653 clusters are generated as the preliminary results. Each cluster represents a kind of profiled vessel behavior concerning the corresponding MMSI. Then the similar clusters from different data of MMSI have been merged. Behavior vectors are created by averaging the five attributes in the same cluster, and the behavior vectors are used for second layer clustering. The second layer, clustering, used the same enhanced DBSCAN method to keep the result consistent. Parameters are manually modified to fit this step's goal since the priory in the second layer clustering was

no longer filter the outliers. After this step, 1279 third layer clusters are generated from merging the 2653 clusters by the last step. **Figure 4–26** shows the final AIS clusters on the Gulf of Mexico, each representing one vessel behavior pattern in the region. Since there are way too many clusters presented in **Figure 4–26**, some colors are used repetitively to present different clusters.

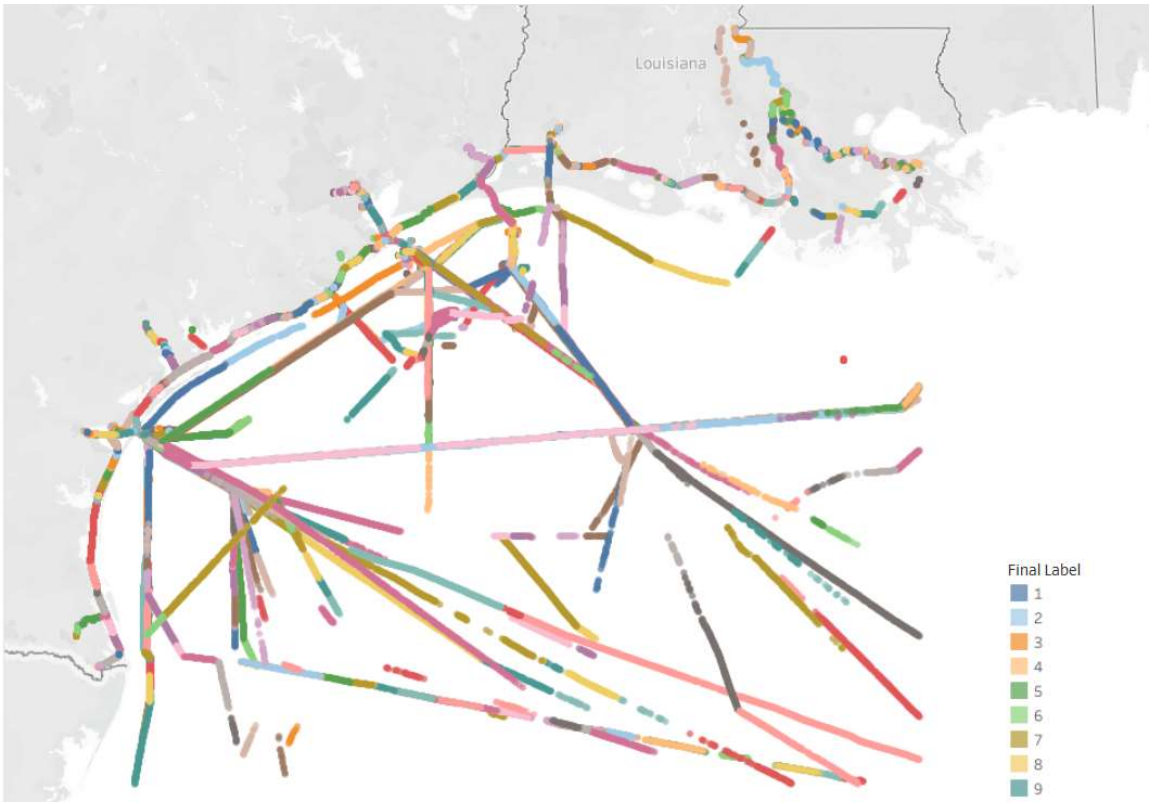
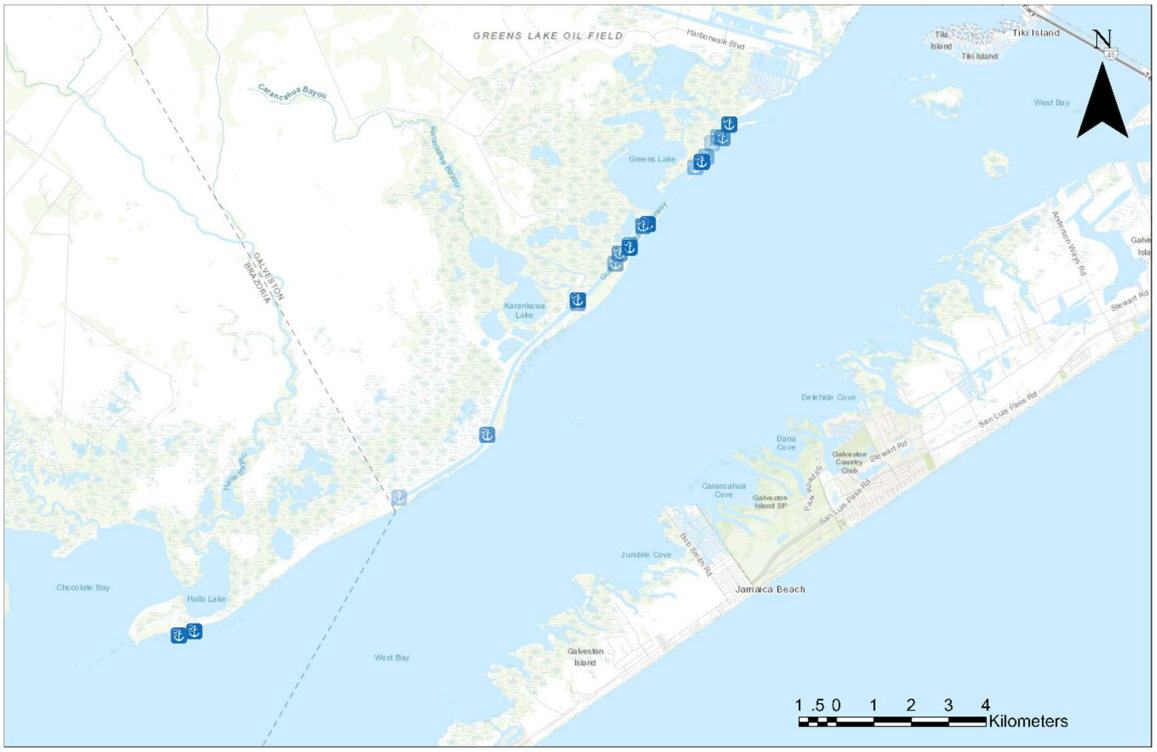


Figure 4–26. Final AIS Clusters resulted from the proposed clustering framework on the Gulf of Mexico, with each color representing one vessel behavior pattern

One of the applications of analyzing the clustering results is to detect port areas. The places where all the vessels are anchored in the Gulf of Mexico Region are presented in **Figure 4–27a. Figure 4–27b** presents a zoomed-in figure showing the details of those ports.

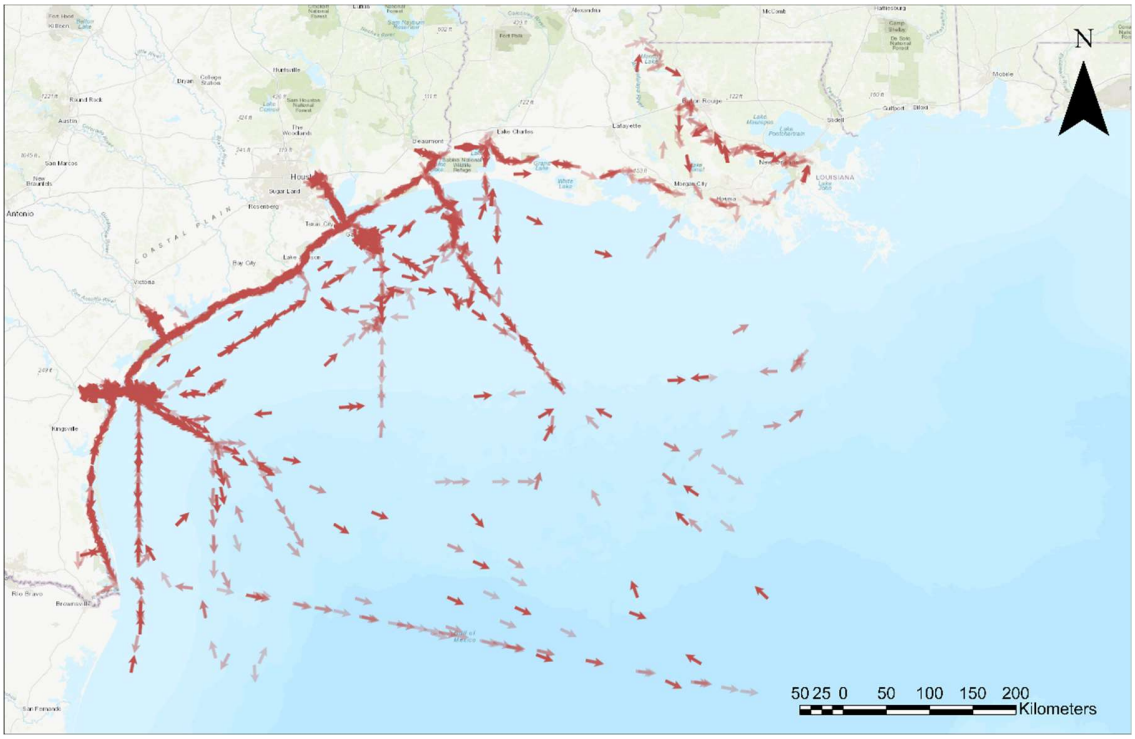
4-27a. All Ports Detected by the Algorithm



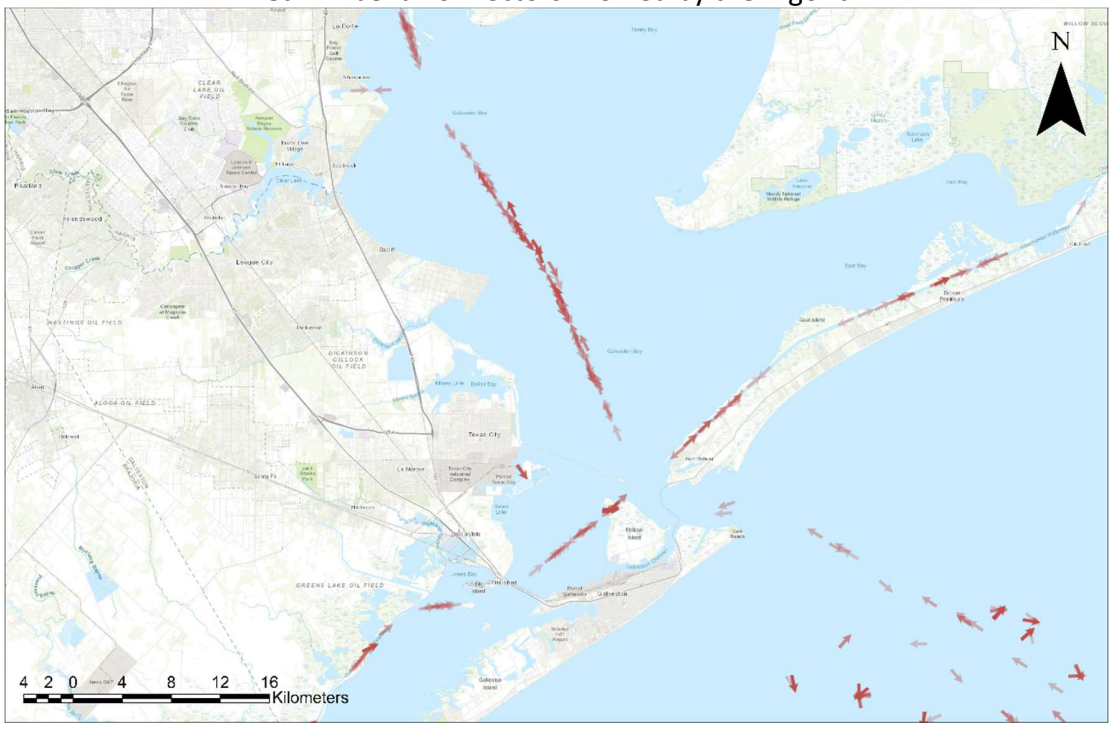
4-27b. Zoomed-In Figure Showing Details Around West Bay

Figure 4–27. Ports and locations where vessels are mooring detected by the proposed clustering framework in Gulf of Mexico Area

New behavior vectors are created by averaging the attributes and considering data size as weights. In the following **Figure 4–28**, those behavior vectors are represented as arrows. The data size is presented as darkness level, and the directions of the arrows present the heading. **Figure 4–28b** presents a zoomed-in figure showing the details of those behavior vectors.



4-28a. All behavior vectors Profiled by the Algorithm



4-28b. Zoomed-in figure showing details around Galveston Bay and Trinity Bay

Figure 4–28. Profiled behavior vectors on the Gulf of Mexico from proposed clustering framework, represented by the arrows

Taking advantage of the final clustering results, vessel behavior recommendation and anomaly detection model have been developed. Given the vessel location, the model will recommend what the vessel should do based on the well-organized training data. The example is shown in **Figure 4–29**. For a vessel located at (25.49, -93.3906), the recommendation model calculates Euclidean Distance to all profiled behavior vectors, and finds the two closest clusters (Cluster 1775 and Cluster 1115). The corresponding probabilities is found by the ratio of the inverse of the distance, making the closer cluster has higher weights to provide possible vessel actions to the vessel at the location. The recommendation model, at the current stage, only recommends speed and heading. When the model becomes more comprehensive for future works, more advanced information can be provided, such as destination and routes associated with the specific clusters.

	Point			Cluster 1775			Cluster 1115
LAT	25.4900		**LAT**	25.4399		**LAT**	25.4676
LONG	-93.3906		**LON**	-92.9318		**LON**	-93.1219
			SOG	12.8459		**SOG**	12.4737
			COG	100.7888		**COG**	107.2874
			Heading	100.2267		**Heading**	108.1444
			Probability	80.29%		**Probability**	19.71%

Figure 4–29. Application of the model for the Gulf of Mexico: vessel behavior recommendations based on given location

The anomaly detection model has been developed from the final clustering result as well. The algorithm is the same as the supervised component of the proposed enhanced

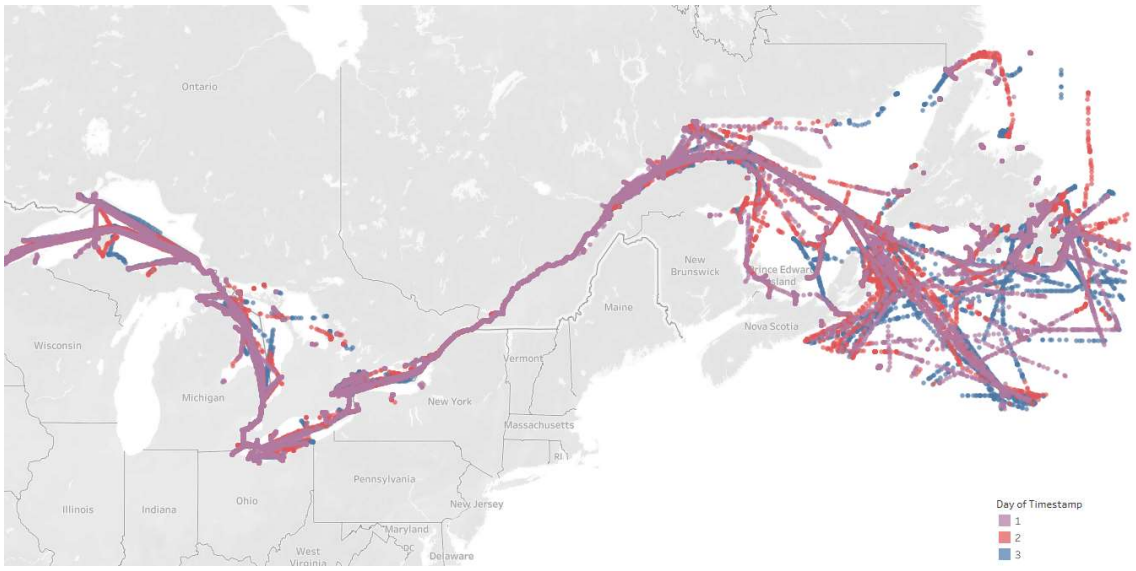
DBSCAN. Based on the new observations, the model calculates the Mahalanobis distance to the two closest clusters and determines which cluster the data belongs. The model can also provide probabilities that the vessel has anomaly behaviors. The example is shown in **Figure 4–30**. The model reads the AIS signal from the monitored vessel and finds matchings with the well-organized clusters (Cluster 111 and Cluster 1710) with corresponding probabilities. For an AIS signal from the monitored vessel, the anomaly detection model calculates Mahalanobis Distance to all profiled AIS clusters vectors, and finds the two closest clusters (Cluster 111 and Cluster 1710). The corresponding probabilities are found by the ratio of the inverse of the distance, making the closer cluster has higher probabilities to be matched with the monitored vessel. The anomaly detection model, at the current stage, detects anomaly behaviors with respect to all clusters. When the model becomes more comprehensive for future works, more advanced information can be used for detecting anomaly behaviors associated with the specific routes.

	New Observation			Cluster 111			Cluster 1710			Anomaly
LAT	27.8456		LAT	27.8456		LAT	27.8174			
LON	-97.2262		LON	-97.2261		LON	-97.3759			
SOG	0.0000		SOG	0.0415		SOG	2.6565			
COG	-105.7000		COG	-141.3424		COG	-45.8701			
Heading	230.0000		Heading	229.2495		Heading	185.7938			
			Probabality	74.54%		Probabality	16.87%		Probabality	8.58%

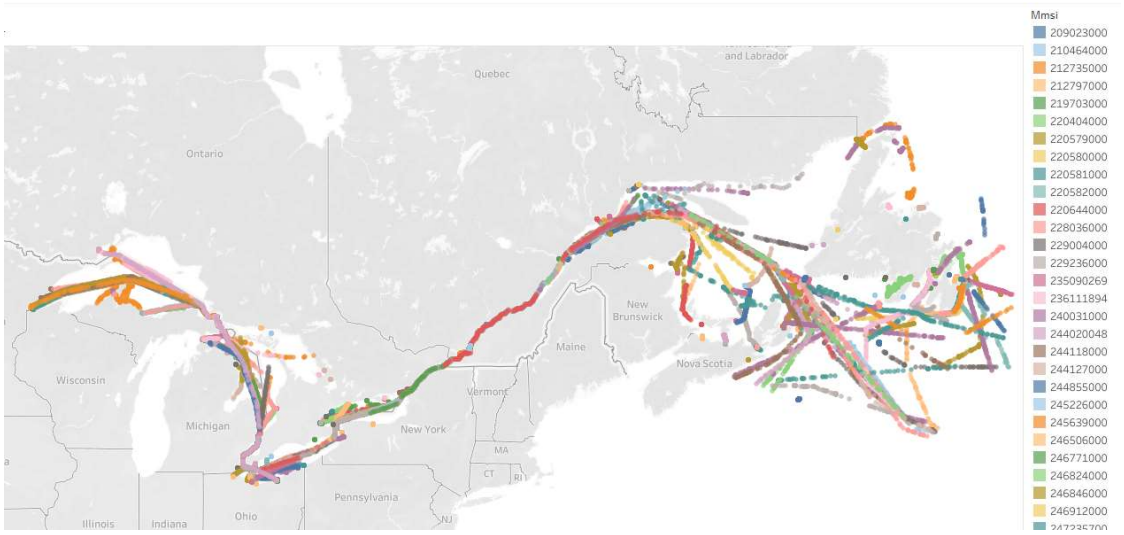
Figure 4–30. Application of the model for the Gulf of Mexico: vessel behavior monitoring and anomaly detection on new observations

4.3.2 Saint Lawrence Seaway AIS Data

Another dataset that was implemented by the proposed clustering algorithm is the Saint Lawrence Seaway region AIS data. Models to monitor vessels in that region have been generated for the case study. We purchased 3-day data in June 2017 from OBCOMM. The data is around 17 MB describing the vessel's movement history and contains around 135 thousand trajectory points clustered. The raw data has been visualized in **Figure 4–31**. **Figure 4–31a** presents all the raw AIS point data, and **Figure 4–31b** visualized the data in various colors for corresponding MMSI.



4-31a. Raw AIS Point Data in Saint Lawrence Seaway Region



4-31b. Raw Trajectory Data by MMSI in Saint Lawrence Seaway Region

Figure 4–31. Raw AIS data and raw trajectory data in Saint Lawrence Seaway and Great Lakes Region

Same to Section 4.3.1, the hierarchical clustering framework has been implemented on the dataset. Firstly, the enhanced DBSCAN clustering method has been implemented on the data of each MMSI in the first layer. By this step, 3095 clusters are generated as the preliminary results. Each cluster represents a kind of profiled vessel behavior concerning the corresponding MMSI. Then the similar clusters from different data of MMSI have been merged. Behavior vectors are created by averaging the five attributes in the same cluster, and the behavior vectors are used for second layer clustering. The second layer, clustering, used the same enhanced DBSCAN method to keep the result consistent. Parameters are manually modified to fit this step's goal since the priory in the second layer clustering was no longer filter the outliers. After this step, 2888 third layer clusters are generated from merging the 3095 clusters by the last step. **Figure 4–32** shows the final AIS clusters on the

Saint Lawrence Seaway, each representing one vessel behavior pattern in the region. Since there are way too many clusters presented in **Figure 4–32**, some colors are used repetitively to present different clusters.

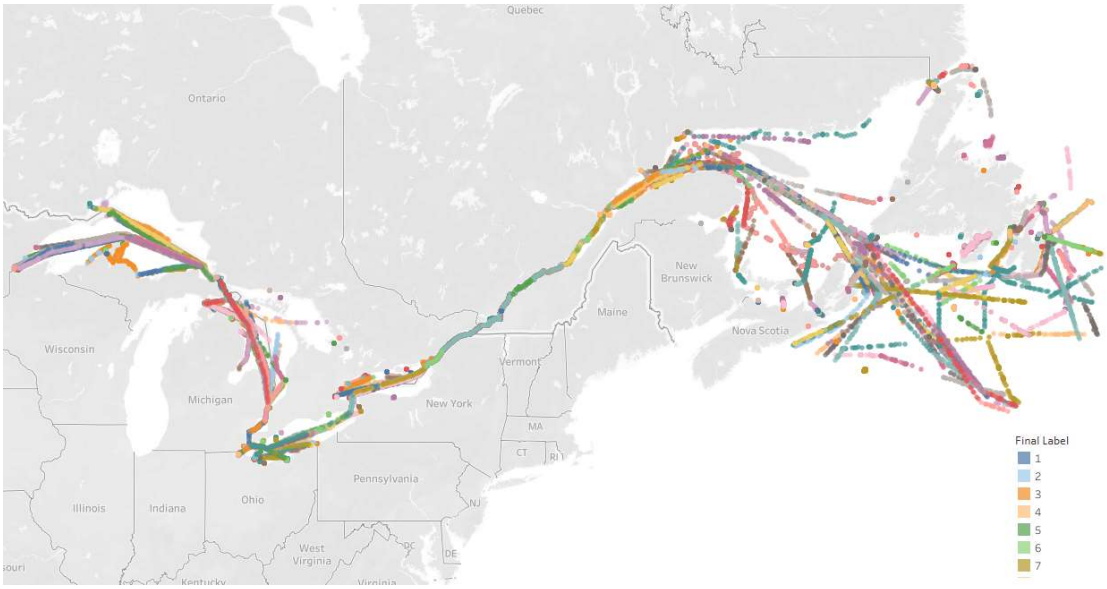


Figure 4–32. Final AIS Clusters resulted from the proposed clustering framework on the Saint Lawrence Seaway and Great Lakes Region, with each color representing one vessel behavior pattern

One of the applications of analyzing the clustering results is to detect port areas. The places where all the vessels are anchored in the Saint Lawrence Seaway Region are presented in **Figure 4–33**. **Figure 4–34** presents zoomed-in figures showing the details of those ports.

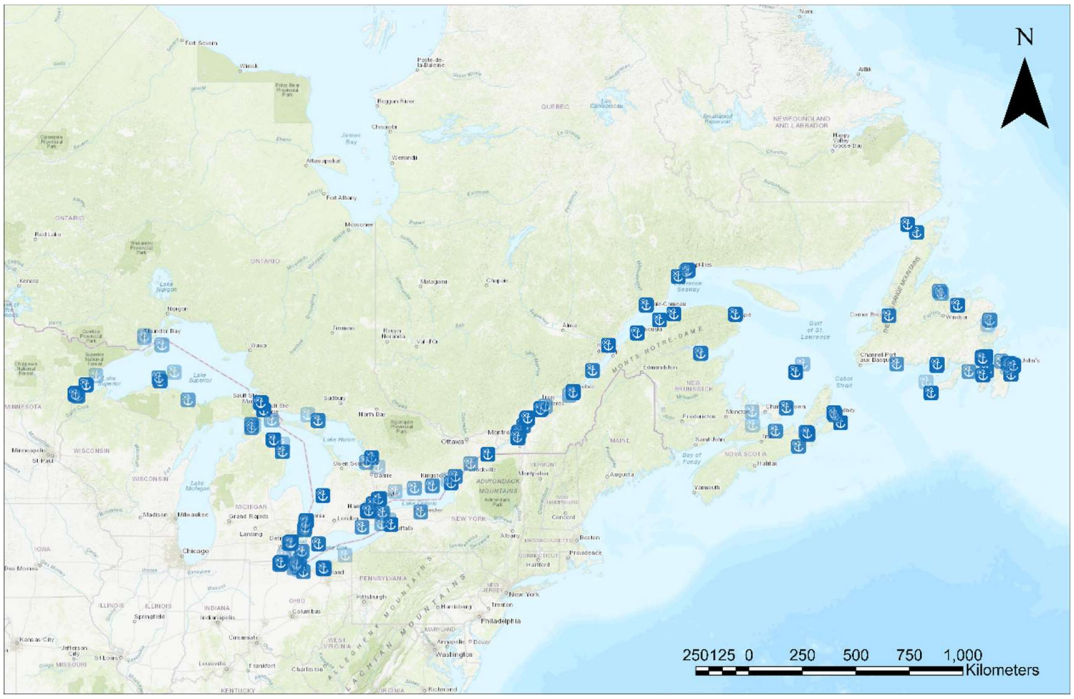
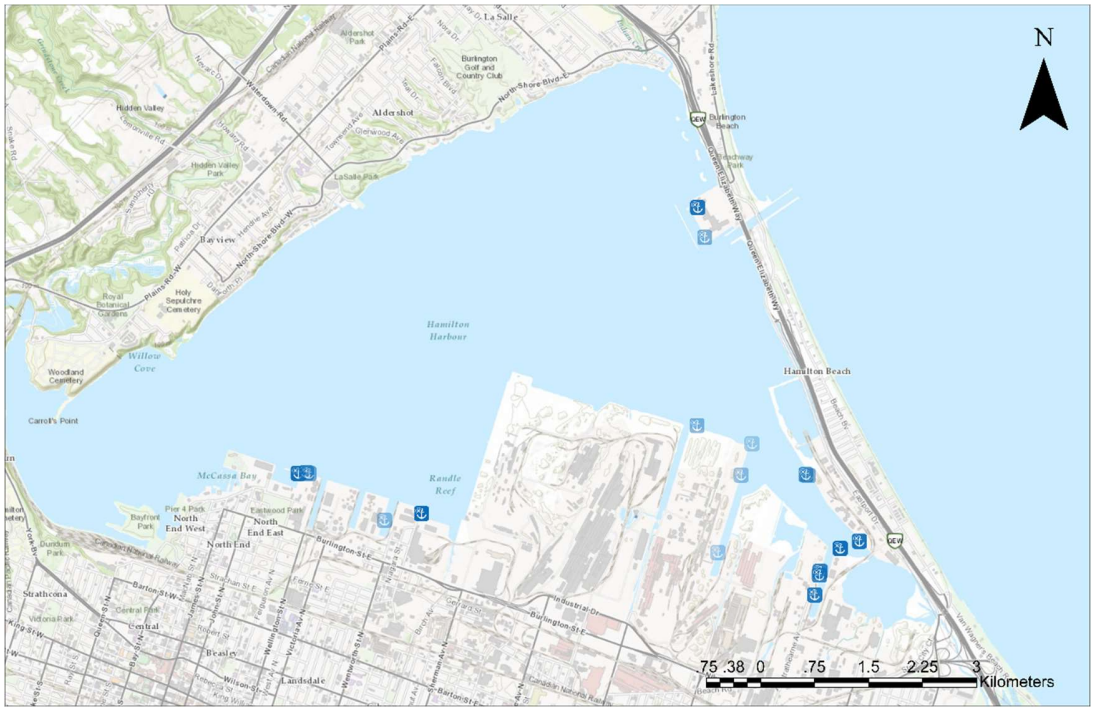
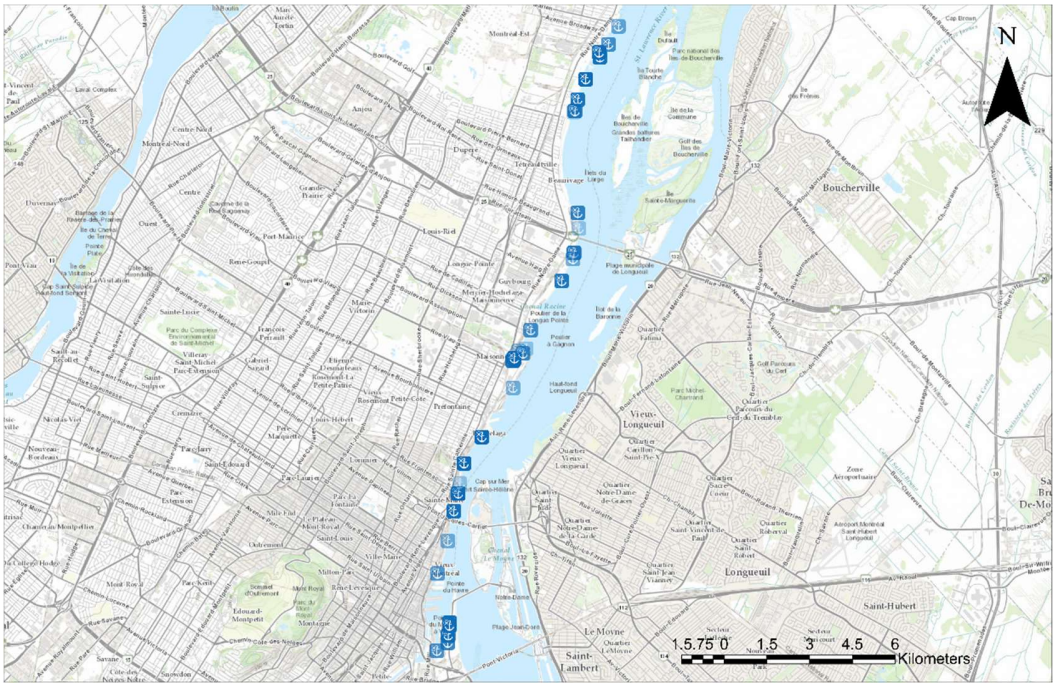


Figure 4–33. Ports and locations where vessels are mooring detected by the proposed clustering framework in Saint Lawrence Seaway and Great Lakes Region

4-34a. Locations of the Vessels mooring in Toronto harbor



4-34b. Locations of the Vessels mooring at Hamilton and Burlington



4-34c. Vessels moored in Montreal harbor

Figure 4–34. Zoomed-in figures showing the details of those Ports and locations where vessels are mooring detected by the proposed clustering framework in Saint Lawrence Seaway and Great Lakes Region

New behavior vectors are created by averaging the attributes and considering data size as weights. In the following **Figure 4–35**, those behavior vectors are represented as arrows. The data size is presented as darkness level, and the directions of the arrows present the heading. **Figure 4–36** presents zoomed-in figures showing the details of those behavior vectors. The results help visual analytics in identifying port regions and busy routes.

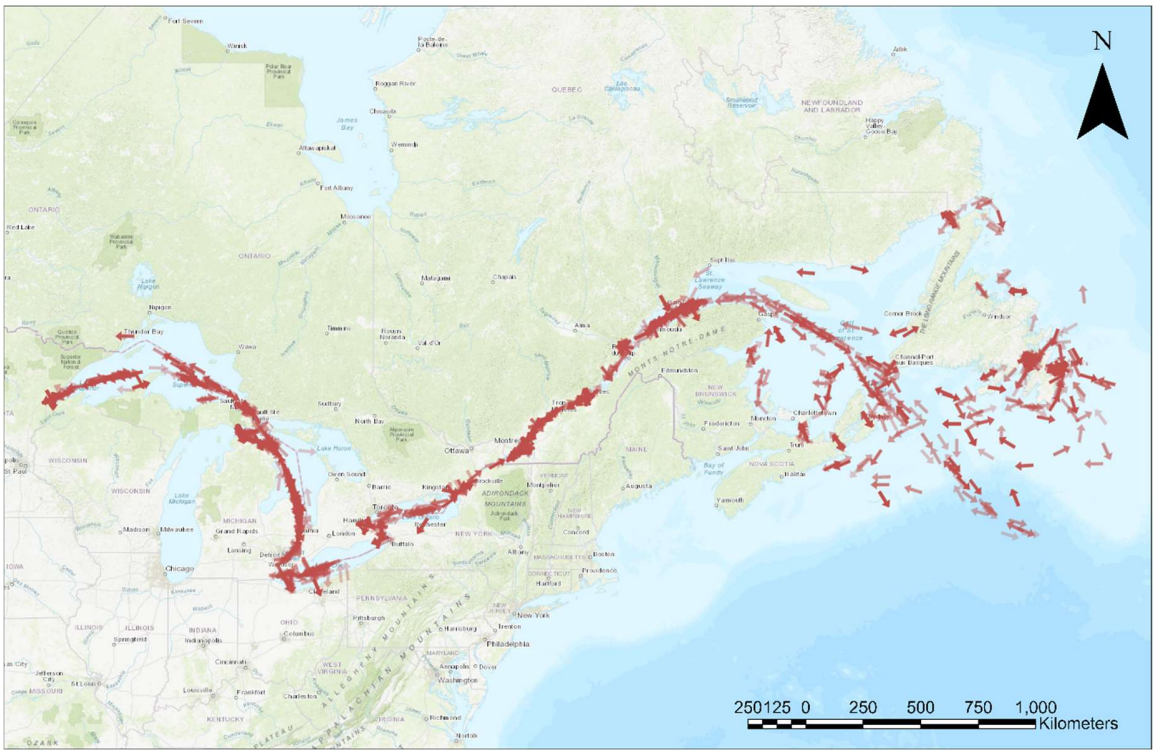
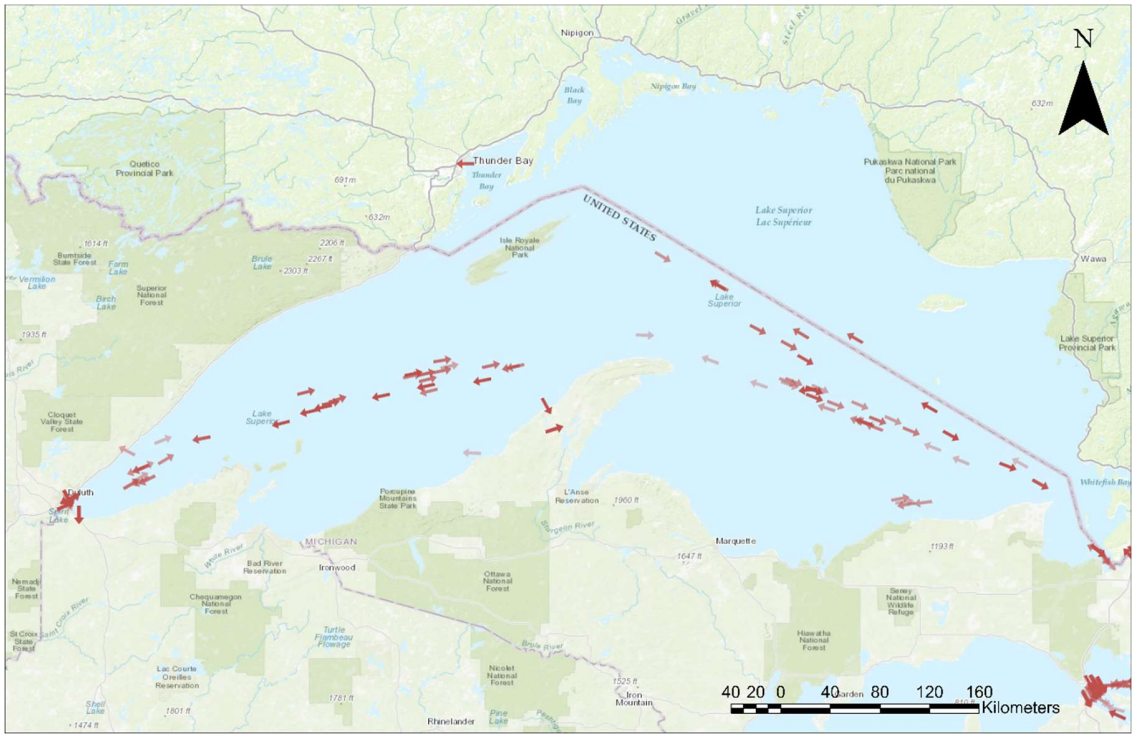
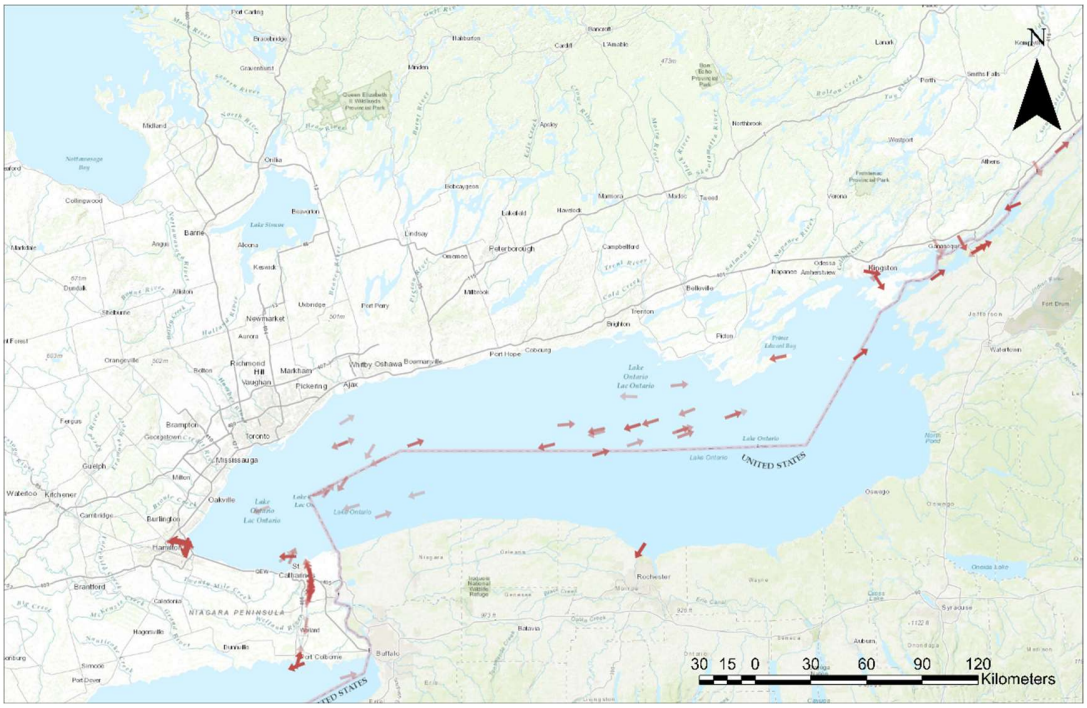


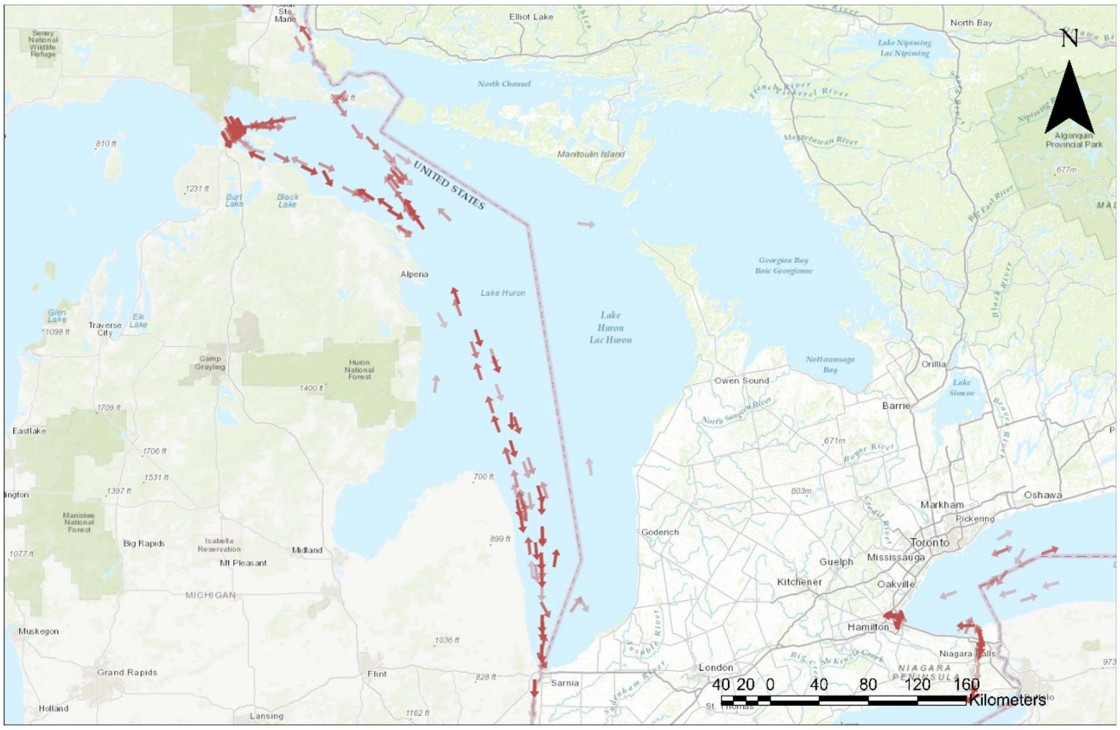
Figure 4–35. Profiled behavior vectors on the Saint Lawrence Seaway and Great Lakes Region from proposed clustering framework, represented by the arrows



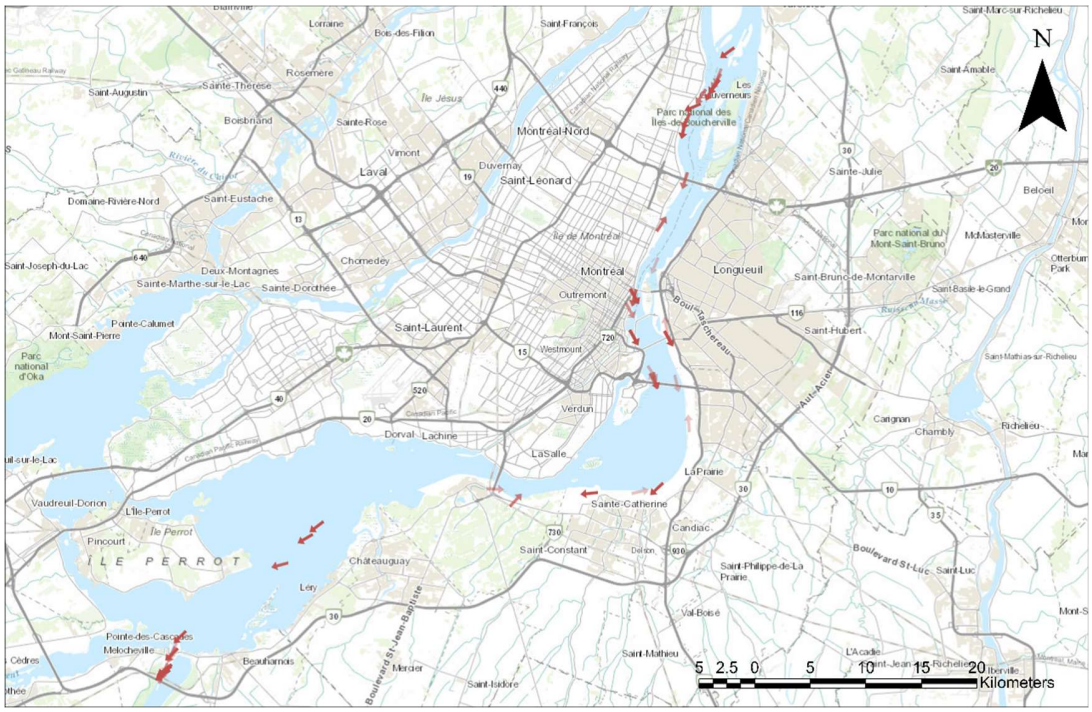
4-36a. Vessels Vectors in Lake Superior



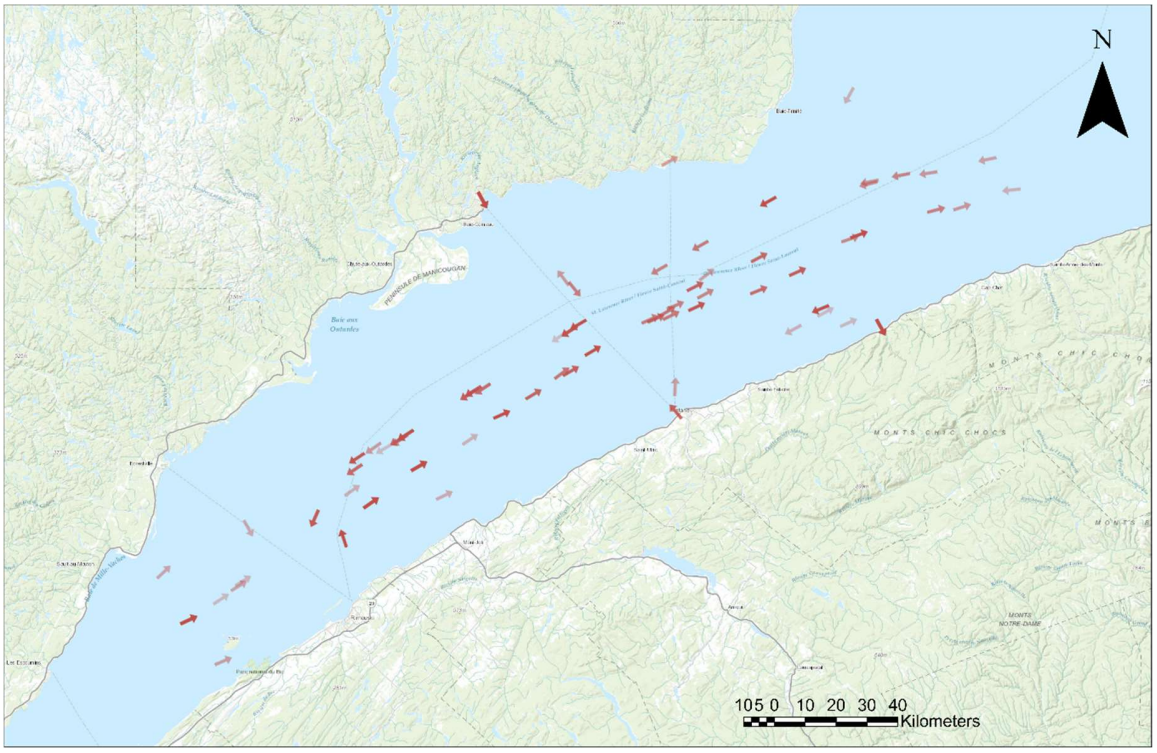
4-36b. Vessels Vectors in Lake Ontario



4-36c. Vessels Vectors in Lake Huron



4-36d. Vessels Vectors in the Montreal area



4-36e. Vessels vectors in St. Lawrence Seaway

Figure 4–36. Zoomed-in Figures Showing the Details of Those Profiled behavior vectors on the Saint Lawrence Seaway and Great Lakes Region from proposed clustering framework, represented by the arrows

In **Figure 4–37** and **Figure 4–38**, some final clusters are presented in the zoomed-in regions. **Figure 4–37** covers the same region as the dataset for algorithm validation. The result shows consistency to the ground truth, such as **Figure 4–5a**. Besides the four clusters (stages) describing the vessels traveling across the river, one more cluster to sailing along the river is profiled (the cluster in orange). This cluster has less data density, but the sparse points are still clustered together, taking advantage of the Mahalanobis distance metric, which considers the correlation between the point in the dataset.

Figure 4–37. Final AIS Clusters resulted from the proposed clustering framework on the location between Kingston to Port at Wolfe Island, with each color representing one vessel behavior pattern

Figure 4–38 shows another example of a series of clustered vessels in Lake Ontario and St. Lawrence River. The result shows that some port areas are identified on the Southwestern Lake Ontario and a vessel sailing behavior from the lake into the river. The stages traveling along the river are mainly divided by the speed difference since the direction almost remains the same. A new finding by this result is that some stages of separated geospatially can still be clustered together. Once the vessel slows down in more crowded water, a new stage is defined. However, after the vessel resumes the normal speed, taking advantage of the Mahalanobis distance metric, these vessel behaviors are highly similar to those before slowing down. Thus, they are clustered together. The cluster in green

in **Figure 4–38** is a case in point. Even after two stages (in pink and purple) divide this cluster, the three separated stages are still considered one cluster (in green).

Figure 4–38. Final AIS Clusters resulted from the proposed clustering framework in Lake Ontario and St. Lawrence River, with each color representing one vessel behavior pattern

Taking advantage of the final clustering results, vessel behavior recommendation and anomaly detection model have been developed. Given the vessel location, the model will recommend what the vessel should do based on the well-organized training data. The example is shown in **Figure 4–39**. For a vessel located at Lat: 46°, Long: -73°, the recommendation model calculates Euclidean Distance to all profiled behavior vectors, and finds the four closest clusters (Cluster 2169, Cluster 2840, Cluster 1867, and Cluster 2119).

The corresponding probabilities is found by the ratio of the inverse of the distance, making the closer cluster has higher weights to provide possible vessel actions to the vessel at the location. The recommendation model, at the current stage, only recommends speed and heading. When the model becomes more comprehensive for future works, more advanced information can be provided, such as destination and routes associated with the specific clusters.

	Point			Cluster 2169			Cluster 2840			Cluster 1867			Cluster 2119
LAT	46.0000		LAT	46.6253		LAT	46.4463		LAT	46.0816		LAT	46.6144
LONG	-73.0000		LON	-71.7936		LON	-72.1909		LON	-72.9443		LON	-71.7876
			SOG	90.1609		SOG	125.3478		SOG	95.7679		SOG	93.3478
			COG	2296.4598		COG	639.6957		COG	2194.1964		COG	2386.4203
			Heading	230.5977		Heading	63.3043		Heading	219.5536		Heading	238.4638
			Probabality	34.18%		Probabality	23.64%		Probabality	22.11%		Probabality	20.07%

Figure 4–39. Application of the Model for Canadian Great Lakes Region: Vessel Behavior Recommendations Based on Given Location

The anomaly detection model has been developed from the final clustering result as well. The algorithm is the same as the supervised component of the proposed enhanced DBSCAN. The model will calculate the Mahalanobis distance to the two closest clusters based on the new observations and determine which cluster the data belongs. The model can also provide probabilities that the vessel has anomaly behaviors. The example is shown in **Figure 4–40**. The model reads the AIS signal from the monitored vessel and finds matchings with the well-organized clusters (Cluster 734 and Cluster 838) with corresponding probabilities. For an AIS signal from the monitored vessel, the anomaly

detection model calculates Mahalanobis Distance to all profiled AIS clusters vectors, and finds the two closest clusters (Cluster 734 and Cluster 838). The corresponding probabilities are found by the ratio of the inverse of the distance, making the closer cluster has higher probabilities to be matched with the monitored vessel. The anomaly detection model, at the current stage, detects anomaly behaviors with respect to all clusters. When the model becomes more comprehensive for future works, more advanced information can be used for detecting anomaly behaviors associated with the specific routes.

	New Observation			Cluster 734			Cluster 838			Anomaly
LAT	45.6245		LAT	43.6366		LAT	43.7469			
LON	-73.4976		LON	-79.3915		LON	-81.7179			
SOG	0.0000		SOG	0.0000		SOG	0.0000			
COG	2652.0000		COG	2685.6322		COG	2311.8750			
Heading	252.0000		Heading	511.0000		Heading	511.0000			
			Probabality	73.16%		Probabality	26.41%		Probabality	0.42%

Figure 4–40. Application of the Model for Canadian Great Lakes Region: Vessel Behavior Monitoring and Anomaly Detection on New Observations

Chapter 5: Conclusions

5.1 Summary and Contribution

The research can be summarized into three following aspects. First, the research proposed enhanced DBSCAN by optimizing clustering performance in terms of homogeneity, completeness, and other evaluation metrics. Second, the research proposed a clustering framework to be implemented on big data, generating the clustering results in two case studies. Third, the research developed the models for vessel action recommendation and detecting vessel behavior outliers in the case study regions.

Firstly, a clustering method has been proposed to enhance the DBSCAN clustering method by incorporating the Mahalanobis distance metric. The proposed clustering method outperforms traditional DBSCAN by considering correlations among the points and reduce computational cost. The enhanced DBSCAN using Mahalanobis distance can deal with scale and correlation issues better than traditional DBSCAN using Euclidean distance. The thesis presents a straightforward way to find the required parameters required by the enhanced DBSCAN, making the method consistent when applying to big data. The proposed algorithm has also been thoroughly compared with other commonly used clustering algorithms in four designed validation experiments. It has been evaluated by both internal and external clustering evaluation metrics, and the results indicate that the proposed algorithm's performance is high.

Secondly, based on the proposed algorithm, the thesis proposed a clustering framework that can be efficiently applied to big data. The framework takes speed and heading into account when clustering the vessels in addition to geospatial information. By defining the point data as a novel five-dimensional vector, the clustering algorithm can find insights and discoveries in a space with a more complex concept of 'density reachability.' The hierarchical clustering framework comprises layers of clustering, utilizing both the unsupervised component and supervised component of the proposed clustering algorithm. The proposed clustering framework has been implemented on novel high-dimensional data to represent historical AIS data for modeling vessel behaviors.

Thirdly, the clustering results generate maritime traffic patterns extraction and vessel behavior anomaly detection models. Two big datasets are accessed and used for the case study. The first study area is the Gulf of Mexico, and the second is the Canadian Great Lakes regions. The thesis presents how the models work on giving action recommendations based on the information from the vessel and detecting behavior anomalies of the vessel. The results indicate that the proposed framework can effectively model vessel behaviors in those two waters and show its potentials to work in other regions.

The contributions of the research can also be summarized into three aspects. Firstly, the machine learning community benefits from the DBSCAN clustering optimization. The enhanced DBSCAN brings new possibilities and understanding of clustering. The proposed parameter auto-selecting method facilitates clustering tasks and spare efforts on trial-and-error methods to find suitable parameters.

Secondly, the enhanced DBSCAN clustering method and proposed framework implementing on historical AIS data contributes to modeling marine transportation and autonomous vessels research. This research proposed a way to monitor crewed vessels, provides foundations for vessel route planning and vessel behavior anomalies detection.

Thirdly, the proposed clustering algorithm and framework can use applied to more general data analytics tasks. Beyond contributions to marine transportation modeling, a similar clustering framework can also be applied to similar tasks on modeling data from other moving objects such as Automatic Dependent Surveillance-Broadcast (ADS-B) data, data from pedestrians, data from vehicles, and data from UAVs. The proposed clustering algorithm and framework can also be applied to social media and video platform user analysis. Through profiling user behaviors and organizing contents, Ads/contents promotion algorithm can be designed. Besides providing a possible process for analyzing, clustering, and modeling AIS data, the enhanced DBSCAN and hierarchy clustering framework can be applied to organizing the other raw unlabeled data and facilitating preparing labeled training data by descriptor data clustering. The framework also provides a foundation for active learning. The framework can be furtherly modified into an interactive process taking advantage of the designed semi-supervised process. The machine learning community will be tremendously benefited as it can help spare huge efforts on preparing large training data when generating AI models.

5.2 Future Work and Perspectives

The proposed method is based on assumptions that marine AIS data contains valuable insights into mined and vessel behavior patterns discovered from systematically studying them. To guarantee the assumption to be valid, it requires that the raw unlabeled AIS trajectories data should have been quality controlled. Even though the proposed framework provides the ability to filter outliers from the trajectory, it is not powerful enough, yet the whole trajectory has inferior quality. A pre-processing procedure to filter the anomaly AIS trajectory should be implemented in the future to ensure the quality of the training data.

Working with heavily unevenly distributed data has been challenges. The work proposed an intuitive approach to find parameters for enhanced DBSCAN. However, since the way parameters are defined is to prioritize differentiating the outliers, the results are not optimal for some datasets with more outliers than valuable data. For example, in some datasets, the majority of the points data is concentrated at port areas. The proposed algorithm will mistakenly detect main trajectories as outliers concerning the way to define the parameters. Furthermore, some datasets in which stages are not distinct enough to be separated from each other. Thus, the proposed parameter selection method has an unreliable result to obtain clusters composed of consistent points. The parameter setting still requires adjusting from the user, and a more dynamic modification is expected from the future.

Besides solving the limitations listed above, two extra aspects of the research can be expanded in the future. Firstly, the currently proposed clustering method works on five-dimensional point data at a specific time and profile behaviors based on the clustering results but does not consider the trajectory. A long short-term memory concept can be applied to the clustering process by integrating another time dimension. For example, the vector data can be influenced by its behavior history, with larger weights on closer previous vectors. In this case, data can be more accurately profiled, and the model generated can make better behavior predictions. Secondly, the framework can be furtherly modified into an active learning model with an interactive labeling process. The clustering framework can generate accurate clustering results with small human efforts on preparing large training data and computational cost by iteratively returning the least reliable clustering results from the unsupervised component and manually modifying the labels.

References

[1] Bernard Marr. 2019. “The Incredible Autonomous Ships of The Future: Run by Artificial Intelligence Rather Than A Crew.” Forbes. 2019. https://www.forbes.com/sites/bernardmarr/2019/06/05/the-incredible-autonomous-ships-of-the-future-run-by-artificial-intelligence-rather-than-a-crew/?sh=6c5f29806fbf.

[2] Dana, Merkel. 2019. “Autonomous Ships, Opportunities & Challenges.” 2019. https://www.marinelink.com/news/autonomous-ships-opportunities-challenges-471609.

[3] Vespe, Michele, Ingrid Visentini, Karna Bryan, and Paolo Braca. 2012. “Unsupervised Learning of Maritime Traffic Patterns for Anomaly Detection.” *IET Conference Publications* 2012 (595 CP): 14. https://doi.org/10.1049/cp.2012.0414.

[4] Jallal, C. 2018. “Rolls-Royce and Finferries Demonstrate World’s First Fully Autonomous Ferry.” *Maritime Digitalisation & Communications*, December. https://www.rolls-royce.com/media/press-releases/2018/03-12-2018-rr-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry.aspx.

[5] Sheng, Pan, and Jingbo Yin. 2018. “Extracting Shipping Route Patterns by Trajectory Clustering Model Based on Automatic Identification System Data.” *Sustainability (Switzerland)* 10 (7). https://doi.org/10.3390/su10072327.

[6] Silveira, P. A.M., A. P. Teixeira, and C. Guedes Soares. 2013. “Use of AIS Data to Characterise Marine Traffic Patterns and Ship Collision Risk off the Coast of Portugal.” *Journal of Navigation* 66 (6): 879–98. https://doi.org/10.1017/S0373463313000519.

[7] IMO. 2000. “SOLAS CHAPTER V SAFETY OF NAVIGATION REGULATION 1-Application.” IMO. 2000. www.imo.org.

[8] Ester, Martin, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. 1996. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.” www.aaai.org.

[9] Schubert, Erich, Alexander Koos, Tobias Emrich, Andreas Züfle, Klaus Arthur Schmid, and Arthur Zimek. 2015. “A Framework for Clustering Uncertain Data.” In *Proceedings of the VLDB Endowment*, 8:1976–79. https://doi.org/10.14778/2824032.2824115.

[10] Pedregosa FABIANPEDREGOSA, Fabian, Vincent Michel, Olivier Grisel OLIVIERGRISEL, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Jake Vanderplas, et al. 2011. “Scikit-Learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot.” *Journal of Machine Learning Research*. Vol. 12. https://doi.org/10.5555/1953048.2078195.

[11] R Core Development Team. 2013. “R: A Language and Environment for Statistical Computing, Reference Index Version 3.0.2.” https://www.gnu.org/copyleft/gpl.html.

[12] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. “The WEKA Data Mining Software: An Update.” *ACM SIGKDD Explorations Newsletter* 11 (1): 10–18. https://doi.org/10.1145/1656274.1656278.

[13] Han, X, C Armenakis, and M Jadidi. 2020. “DBscan Optimization for Improving Marine Trajectory Clustering and Anomaly Detection.” In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 43:455–61. https://doi.org/10.5194/isprs-archives-XLIII-B4-2020-455-2020.

[14] Hou, Jian, Huijun Gao, and Xuelong Li. 2016. “DSets-DBSCAN: A Parameter-Free Clustering Algorithm.” *IEEE Transactions on Image Processing* 25 (7): 3182–93. https://doi.org/10.1109/TIP.2016.2559803.

[15] Sawant, Kedar. 2014. “Adaptive Methods for Determining DBSCAN Parameters.” *IJISET-International Journal of Innovative Science, Engineering & Technology*. Vol. 1. www.ijiset.com.

[16] Karami, Amin, and Ronnie Johansson. 2014. “Choosing DBSCAN Parameters Automatically Using Differential Evolution.” *International Journal of Computer Applications* 91 (7): 1–11. https://doi.org/10.5120/15890-5059.

[17] Xia Ln, and Jing Jw. 2009. “SA - DBSCAN: A self-adaptive density-based clustering algorithm” *Journal of the Graduate School of the Chinese Academy of Sciences*

[18] Esmaelnejad, Jamshid, Jafar Habibi, and Soheil Hassas Yeganeh. 2010. “A Novel Method to Find Appropriate ε for DBSCAN.” In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5990 LNAI:93–102. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-12145-6_10.

[19] Smiti, Abir, and Zied Eloudi. 2013. “Soft DBSCAN: Improving DBSCAN Clustering Method Using Fuzzy Set Theory.” In *2013 6th International Conference on Human System Interactions, HSI 2013*, 380–85. https://doi.org/10.1109/HSI.2013.6577851.

[20] Liu, Bo. 2015. “Maritime Traffic Anomaly Detection from AIS Satellite Data in near Port Regions,” no. August. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwi4_IW9hcDhAhXi_XMBHS0qCsUQFjAAegQIARAC&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.856.5052%26rep%3Drep1%26type%3Dpdf&usg=AOvVaw09P6dR.

[21] Fong, Simon, Saif Ur Rehman, Kamran Aziz, and Information Science. 2014. “DBSCAN: Past, Present and Future,” 232–38.

[22] Schubert, Erich, Jörg Sander, Martin Ester, Hans-Peter Kriegel, Xiaowei Xu, and H.-P Kriegel. 2017. “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN.” *DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. ACM Trans. Database Syst* 42 (3). https://doi.org/10.1145/3068335.

[23] Ren, Yan, Xiaodong Liu, and Wanquan Liu. 2012. “DBCAMM: A Novel Density Based Clustering Algorithm via Using the Mahalanobis Metric.” *Applied Soft Computing Journal* 12 (5): 1542–54. https://doi.org/10.1016/j.asoc.2011.12.015.

[24] Sangeetha, Margaret, Velumani Padikkaramu, and Rajakumar Thankappan Chellan. 2018. “A Novel Density Based Clustering Algorithm by Incorporating Mahalanobis Distance.” *International Journal of Intelligent Engineering and Systems* 11 (3). https://doi.org/10.22266/ijies2018.0630.13.

[25] Kanevski, Mikhail, Loris Foresti, Christian Kaiser, Alexel Pozdnoukhov, Vadim Timonin, and Devis Tuia. 2009. “Machine Learning Models for Geospatial Data.” *Handbook of Theoretical and Quantitative Geography*, no. April 2018: 175–227.

[26] Xu, Dongkuan, and Yingjie Tian. 2015. “A Comprehensive Survey of Clustering Algorithms.” *Annals of Data Science* 2 (2): 165–93. https://doi.org/10.1007/s40745-015-0040-1.

[27] Ahmed, Mohiuddin, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. “The K-Means Algorithm: A Comprehensive Survey and Performance Evaluation.” *Electronics (Switzerland)* 9 (8): 1–12. https://doi.org/10.3390/electronics9081295.

[28] Reddy, Chandan K., and Bhanukiran Vinzamuri. 2019. “A Survey of Partitional and Hierarchical Clustering Algorithms.” In *Data Clustering*, 87–110. Chapman and Hall/CRC. https://doi.org/10.1201/9781315373515-4.

[29] Ilango, Mr, and V Mohan. 2010. “A Survey of Grid Based Clustering Algorithms.” *International Journal of Engineering Science and Technology* 2 (8): 3441–46.

[30] Wang, Wei, Jiong Yang, and Richard Muntz. 1997. “STING: A Statistical Information Grid Approach to Spatial Data Mining.” In *Proceedings of the 23rd International Conference on Very Large Databases, VLDB 1997*, 186–95.

[31] Zaki, Mohammed J, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. 1997. “Parallel Algorithms for Discovery of Association Rules.” *Data Mining and Knowledge Discovery* 1 (4): 343–73. https://doi.org/10.1023/A:1009773317876.

[32] Ankerst, Mihael, Markus M Breunig, Hans-peter Kriegel, and Jörg Sander. 1999. “OPTICS: Ordering Points to Identify the Clustering Structure.” *ACM SIGMOD Record* 28 (2): 49–60.

[33] Hinneburg, Alexander, and Daniel A Keim. 2003. “A General Approach to Clustering in Large Databases with Noise.” *Knowledge and Information Systems* 5 (4): 387–415. https://doi.org/10.1007/s10115-003-0086-9.

[34] Sun, Jingwen, Weixing Du, and Niancai Shi. 2018. “A Survey of KNN Algorithm.” *Information Engineering and Applied Computing*, 1–10.

[35] Zhou, Xujun, Xianxia Zhang, and Bing Wang. 2016. “Online Support Vector Machine: A Survey.” In *Advances in Intelligent Systems and Computing*, 382:269–78. Springer Verlag. https://doi.org/10.1007/978-3-662-47926-1_26.

[36] Harvey, Shane, and Reg Harvey. 1998. “An Introduction to Artificial Neural Network.” *Appita Journal* 51 (1): 26–30. https://doi.org/10.2514/6.1994-294.

[37] Bian, Jiang, Dayong Tian, Yuanyan Tang, and Dacheng Tao. 2018. “A Survey on Trajectory Clustering Analysis,” February. http://arxiv.org/abs/1802.06971.

[38] Gao, Yun Jun, Chun Li, Gen Cai Chen, Ling Chen, Xian Ta Jiang, and Chun Chen. 2007. “Efficient K-Nearest-Neighbor Search Algorithms for Historical Moving Object Trajectories.” *Journal of Computer Science and Technology* 22 (2): 232–44. https://doi.org/10.1007/s11390-007-9030-x.

[39] Piciarelli, Claudio, Claudio Piciarelli, Christian Micheloni, Gian Luca Foresti, and Senior Member. 2008. “Trajectory-Based Anomalous Event Detection.” https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.149.6985.

[40] Cho, Kyunghyun, and Xi Chen. 2014. “Classifying and Visualizing Motion Capture Sequences Using Deep Neural Networks.” In *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, 2:122–30. SciTePress. https://doi.org/10.5220/0004718301220130.

[41] Li, Xi, Xi Li, Weiming Hu, and Wei Hu. 2006. “A Coarse-to-Fine Strategy for Vehicle Motion Trajectory Clustering.” *IN PROC. INT’L CONF. PATTERN RECOGNITION*. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.488.1975.

[42] Xiang, Tao, and Shaogang Gong. 2008. “Spectral Clustering with Eigenvector Selection.” *Pattern Recognition* 41 (3): 1012–29. https://doi.org/10.1016/j.patcog.2007.07.023.

[43] Lee, Jae-Gil, Jiawei Han, and Kyu-Young Whang. 2007. *Trajectory Clustering: A Partition-and-Group Framework **.

[44] Ferreira, Nivan, Claudio Silva, James T Klosowski, and Carlos Scheidegger. 2012. “Vector Field K-Means: Clustering Trajectories by Fitting Multiple Vector Fields.” *Computer Graphics Forum*, Vol. 32, Wiley Online Library, 2013, pp. 201–210

[45] Galluccio, Laurent, Olivier Michel, Pierre Comon, and Alfred O Hero III. 2012. “Graph Based K-Means Clustering.” *Signal Processing* 92 (9) (2012) 1970–1984.

[46] Laxhammar, Rikard, and Goran Falkman. 2014. “Online Learning and Sequential Anomaly Detection in Trajectories.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (6): 1158–73. https://doi.org/10.1109/TPAMI.2013.172.

[47] Peng, Liu, Zhou Dong, and Wu Naijun. 2007. “VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise.” In *Proceedings - ICSSSM’07: 2007 International*

Conference on Service Systems and Service Management. https://doi.org/10.1109/ICSSSM.2007.4280175.

[48] Uncu, Ozge, William A. Gruver, Dilip B. Kotak, Dorian Sabaz, Zafeer Alibhai, and Colin Ng. 2006. "GRIDBSCAN: GRId Density-Based Spatial Clustering of Applications with Noise." In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 4:2976–81. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICSMC.2006.384571.

[49] Ram, Anant, Sunita Jalal, Anand S. Jalal, and Manoj Kumar. 2010. "A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases." *International Journal of Computer Applications* 3 (6): 1–4. https://doi.org/10.5120/739-1038.

[50] Ram, Anant, Ashish Sharma, Anand S. Jalal, Raghuraj Singh, and Ankur Agrawal. 2009. "An Enhanced Density Based Spatial Clustering of Applications with Noise." In *2009 IEEE International Advance Computing Conference, IACC 2009*, 1475–78. https://doi.org/10.1109/IADCC.2009.4809235.

[51] Borah, B., and D. K. Bhattacharyya. 2007. "A Clustering Technique Using Density Difference." In *Proceedings of ICSCN 2007: International Conference on Signal Processing Communications and Networking*, 585–88. https://doi.org/10.1109/ICSCN.2007.350675.

[52] Xiaopeng Yu, Deyi Zhou, and Yan Zhou. 2005. "A New Clustering Algorithm Based on Distance and Density." In *Proceedings of ICSSSM '05. 2005 International Conference on Services Systems and Services Management, 2005.*, 2:1016-1021 Vol. 2. IEEE. https://doi.org/10.1109/ICSSSM.2005.1500146.

[53] Elbatta, Mohammad N T. 2012. "An Improvement for DBSCAN Algorithm for Best Results in Varied Densities."

[54] Fahim, Am, and Am Salem. 2006. "Density Clustering Based on Radius of Data (DCBRD)." *Enformatika* 16: 344–49.

[55] Liu, Bing. 2006. "A Fast Density-Based Clustering Algorithm for Large Databases." In *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, 2006:996–1000. https://doi.org/10.1109/ICMLC.2006.258531.

[56] Xiaoyun, Chen, Min Yufang, Zhao Yan, and Wang Ping. 2008. "GMDBSCAN: Multi-Density DBSCAN Cluster Based on Grid." In *IEEE International Conference on E-Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, DKEEE'08*, 780–83. https://doi.org/10.1109/ICEBE.2008.54.

[57] Borah, B., and D. K. Bhattacharyya. 2004. "An Improved Sampling-Based DBSCAN for Large Spatial Databases." In *Proceedings of International Conference on Intelligent Sensing and Information Processing, ICISIP 2004*, 92–96. https://doi.org/10.1109/icisip.2004.1287631.

[58] El-Sonbaty, Yasser, M. A. Ismail, and Mohamed Farouk. 2004. “An Efficient Density Based Clustering Algorithm for Large Databases.” In *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 673–77. https://doi.org/10.1109/ICTAI.2004.27.

[59] Mahran, Shaaban, and Khaled Mahar. 2008. “Using Grid for Accelerating Density-Based Clustering.” In *Proceedings - 2008 IEEE 8th International Conference on Computer and Information Technology, CIT 2008*, 35–40. https://doi.org/10.1109/CIT.2008.4594646.

[60] Birant, Derya, and Alp Kut. 2007. “ST-DBSCAN: An Algorithm for Clustering Spatial-Temporal Data.” *Data and Knowledge Engineering* 60 (1): 208–221. https://doi.org/10.1016/j.datak.2006.01.013.

[61] “MarineCadastre.Gov | Vessel Traffic Data.” n.d. Accessed December 14, 2020. https://marinecadastre.gov/ais/.

[62] ‘Reprint of: Mahalanobis, P.C. (1936) “On the Generalised Distance in Statistics.”’ (2018) Sankhya A. Springer Science and Business Media LLC, 80(S1), pp. 1–7. doi: 10.1007/s13171-019-00164-5.

[63] Davies, D. L. and Bouldin, D. W. (1979) ‘A Cluster Separation Measure’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), pp. 224–227. doi: 10.1109/TPAMI.1979.4766909.

[64] Rousseeuw, P. J. (1987) ‘Silhouettes: A graphical aid to the interpretation and validation of cluster analysis’, *Journal of Computational and Applied Mathematics*. North-Holland, 20(C), pp. 53–65. doi: 10.1016/0377-0427(87)90125-7.

[65] Caliński, T. and Harabasz, J. (1974) ‘A Dendrite Method Foe Cluster Analysis’, *Communications in Statistics*, 3(1), pp. 1–27. doi: 10.1080/03610927408827101.

[66] Rosenberg, A. and Hirschberg, J. (2007) *V-Measure: A conditional entropy-based external cluster evaluation measure*.

[67] Sasaki, Y. and Fellow, R. (2007) *The truth of the F-measure*.

[68] Rand, W. M. (1971) ‘Objective criteria for the evaluation of clustering methods’, *Journal of the American Statistical Association*, 66(336), pp. 846–850. doi: 10.1080/01621459.1971.10482356.

Appendix

Python Source Codes

Algorithm 1 Proposed_DBSCAN_Unsupervised (D, Eps, MinPts)

Algorithm 1 is the function of using the unsupervised component of the proposed clustering algorithm to discover clusters in the first step. It requires three parameters as input, including the Dataset **D** and two predefined parameters, *Eps* and *MinPts*. *Eps* sets the threshold of the neighborhoods, and *MinPts* limits the number of the core points. The procedure iterates until all the points have been visited and labeled. By the end, all the points density connected are clustered together, and others are identified as outliers. The function returns a list of cluster labels.

```
def Proposed_DBSCAN_Unsupervised(D, eps, MinPts):
    """
    Cluster the dataset `D` using the proposed enhanced DBSCAN algorithm.

    The proposed enhanced DBSCAN algorithm takes a dataset `D` (a list of
vectors),
    a threshold distance `eps,`
    and a required number of points `MinPts`.

    It will return a list of cluster labels. The label -1 means noise,
and then
    the clusters are numbered starting from 1.
    """

    # This list will hold the final cluster assignment for each point in
D.
    # There are two reserved values:
```

```
    #    -1 - Indicates a noise point
    #     0 - Means the point hasn't been considered yet.
    # Initially all labels are 0.
    labels = [0] * len(D)

    # C is the ID of the current cluster.
    C = 0

    # This outer loop is just responsible for picking new seed points
    # --a point from which to grow a new cluster.
    # Once a valid seed point is found, a new cluster is created,
    # and the cluster growth is all handled by the 'expandCluster'
routine.

    # For each point P in the Dataset D...
    # ('P' is the index of the datapoint, rather than the datapoint
itself.)
    print('start to Cluster in Step 1:')
    for P in range(0, len(D)):
        # print the progress
        # if P % 1000 == 0:
        #     print(str(P / len(D) * 100) + '%')

        # Only points that have not already been claimed can be picked as
new seed points.
        # If the point's label is not 0, continue to the next point.
        if not (labels[P] == 0):
            continue

        # Find all of P's neighboring points.
        NeighborPts = regionQuery(D, P, eps)

        # If the number is below MinPts, this point is noise.
        # This is the only condition under which a point is labeled NOISE
        # -- when it's not a valid seed point.
        # A NOISE point may later be picked up by another cluster as a
border point
        # (this is the only condition under which a cluster label can
change
        # --from NOISE to something else).
        if len(NeighborPts) < MinPts:
            labels[P] = -1

        # Otherwise, if there are at least MinPts nearby,
        # use this point as the core point / seed for a new cluster.
        else:
            C += 1
            growCluster(D, labels, P, NeighborPts, C, eps, MinPts)
```

```
    # All data has been clustered!
    return labels




def growCluster(D, labels, P, NeighborPts, C, eps, MinPts):
    """
    Grow a new cluster with label `C` from the core point / seed point
`P`.

    This function searches through the dataset to find all points that
belong
    to this new cluster. When this function returns, cluster `C` is
complete.

    Parameters:
      `D`      - The dataset (a list of vectors)
      `labels` - List storing the cluster labels for all dataset points
      `P`      - Index of the core point / seed point for this new
cluster
      `NeighborPts` - All of the neighbors of `P`
      `C`      - The label for this new cluster.
      `eps`    - Threshold distance
      `MinPts` - Minimum required number of neighbors
    """

    # Assign the cluster label to the seed point.
    labels[P] = C

    # Look at each neighbor of P (neighbors are referred to as Pn).
    # NeighborPts will be used as a FIFO queue of points to search
    # --that is, it will grow as we discover new core points for the
cluster.
    # The FIFO behavior is accomplished by using a while-loop rather than
a for-loop.
    # In NeighborPts, the points are represented by their index in the
original dataset.
    i = 0
    while i < len(NeighborPts):

        # Get the next point from the queue.
        Pn = NeighborPts[i]

        # If Pn was labelled NOISE during the seed search,
```

```
        # then we know it's not a core point (it doesn't have enough
neighbors),
        # so make it a border point of cluster C and move on.
        if labels[Pn] == -1:
            labels[Pn] = C

        # Otherwise, if Pn isn't already claimed, claim it as part of C.
        elif labels[Pn] == 0:
            # Add Pn to cluster C (Assign cluster label C).
            labels[Pn] = C

            # Find all the neighbors of Pn
            PnNeighborPts = regionQuery(D, Pn, eps)

            # If Pn has at least MinPts neighbors, it's a core point!
            # Add all of its neighbors to the FIFO queue to be searched.
            if len(PnNeighborPts) >= MinPts:
                NeighborPts = NeighborPts + PnNeighborPts

        # Advance to the next point in the FIFO queue.
        i += 1

        # We've finished growing cluster C!




def regionQuery(D, P, eps):
    """
    Find all points in dataset `D` within distance `eps` of point `P`.

    This function calculates the Mahalanobis distance between a point P
and every other
    point in the dataset, and then returns only those points which are
within a
    threshold distance `eps`.
    """
    neighbors = []

    # Calculate the covariance matrix
    D_covMatrix = np.cov(D.T, bias=True)

    # If the covariance matrix is singular and do not have inverse
matrix,
    # Use the (Moore-Penrose) pseudo-inverse of a matrix as D_I for next
step
    try:
```

```
        D_I = np.matrix(D_covMatrix).I
    except:
        D_I = np.matrix(np.linalg.pinv(D_covMatrix))

    # For each point in the dataset...
    for Pn in range(0, len(D)):

        # If the distance is below the threshold, add it to the neighbors
list.
        dis = np.dot(np.dot((D[P] - D[Pn]), D_I), (D[P] - D[Pn]).T)[0, 0]
** 0.5
        if dis < eps:
            neighbors.append(Pn)

    return neighbors

    # We've finished Finding all neighbour points in the dataset!
```

Algorithm 2 Proposed DBSCAN Supervised (D, Training_D, Eps)

Algorithm 2 is the function of using the supervised component to classify new observations into pre-defined clusters. It requires three parameters as input, including the new observations Dataset **X_test**, the training Dataset **Training_D** (or split into X_train and Y_train), and a pre-defined parameter *Eps*. *Eps* sets the threshold of the neighborhoods of clusters and find the outliers. The procedure iterates until all the points have been visited and labeled. By the end, all the points close to pre-defined clusters are classified, and others are identified as outliers. The function returns a list of cluster labels.

```
def Proposed_DBSCAN_Supervised(X_train, Y_train, X_test, Eps):
    """
    Cluster the dataset `X_test` using the proposed enhanced DBSCAN
algorithm.

    The supervised component takes a dataset `X_train` (a list of
vectors) and
    a list of labels,`X_train`, to train the decision machine.

    Then the trained model is implemented on the target dataset,
'X_test',
    using a threshold distance `eps`,
    to classify the points.

    It will return a list of cluster labels. The label -1 means noise,
and then
    the clusters are numbered starting from 1.
    """

    # Create a dataframe to hold the training data
    df_train = pd.DataFrame(X_train)
    df_train['label'] = Y_train
```

```
    I_df_train = []
    mean_df_train = []
    labels = []

    # iterate every cluster in the trainig data
    for label in np.unique(Y_train):
        # get the label
        df_train_i = df_train[df_train['label'] == label]

        # get the five attributes of X
        df_X_train_i = df_train_i[[0, 1, 2, 3, 4]]

        # Calculate the covariance matrix
        covMatrix_df_X_train_i = np.cov(df_X_train_i.T, bias=True)

        # If the covariance matrix is singular and do not have inverse
matrix,
        # Use the (Moore-Penrose) pseudo-inverse of a matrix as D_I for
next step
        try:
            D_I = np.matrix(covMatrix_df_X_train_i).I
        except:
            D_I = np.matrix(np.linalg.pinv(covMatrix_df_X_train_i))

        # get center point of the cluster
        mean_df_X_train_i = df_X_train_i.mean(axis=0)

        # store the values in the memory for the use in next step
        I_df_train.append(D_I)
        mean_df_train.append(mean_df_X_train_i)

    # print('start to Cluster in Step 2:')
    # iterate every point in the test dataset
    for i in range(X_test.shape[0]):
        # print the progess
        # if i % 1000 == 0:
        #     print(str(i / X_test.shape[0] * 100) + '%')

        # point p
        p = X_test[i]
        dist_list = []

        # iterate through every pre-defined cluster
        for j in range(len(np.unique(Y_train))):
            # calculate the Mahalanobis distance from the point to the
cluster
            I_j = I_df_train[j]
```

```
            mean_j = mean_df_train[j]
            dist_j = np.dot(np.dot((p - mean_j), I_j), (p - mean_j).T)[0,
0] ** 0.5

            # store the distance
            dist_list.append(dist_j)

        # find the closest cluster
        min_idx = dist_list.index(min(dist_list))

        # compare the distance to the defined Eps
        # if smaller than Eps, the classify the point into this cluster
        if min(dist_list) < Eps:
            labels.append(min_idx)
        # if not, the point is an outlier
        else:
            labels.append(-1)

    # All data has been clustered!
    return labels
```

Algorithm 3 Find_Eps_MinPts_for_Unsupervised (D)

Algorithm 3 is the function of finding the two recommended parameters (*Eps* and *MinPts*) to be used in the unsupervised component of the proposed clustering algorithm to discover clusters in the first step. It requires only one parameter as input: the targeted Dataset **X.** And the two parameters, *Eps* and *MinPts,* will be recommended as returned output.

```
def Find_Eps_MinPts_for_Unsupervised(X):
    """
    Based on the dataset `X` itself
    to recommend a pair of parameters (MinPts and Eps)
    to be used in the unsupervised component in the proposed enhanced
DBSCAN algorithm.

    The algorithm only takes a dataset `X` (a list of vectors)

    It will return two values.
    The first one is MinPts, and the second one is Eps.
    """

    # the recommended MinPts is 0.1% of the total size of Dataset X
    # the recommended MinPts also should be at least 10
    # it can be varied based on user's configuration
    Minpts = max(10, int(0.001 * len(X)))

    # Calculate the covariance matrix
    D_covMatrix = np.cov(X.T, bias=True)

    # If the covariance matrix is singular and do not have inverse
matrix,
    # Use the (Moore-Penrose) pseudo-inverse of a matrix as D_I for next
step
    try:
        D_I = np.matrix(D_covMatrix).I
    except:
        D_I = np.matrix(np.linalg.pinv(D_covMatrix))
```

```
    # start to calculate k nearest neighbours distance
    k = Minpts - 1
    kn_distance = []

    # print('start to find MinPts and Eps in Step 1:')
    # iterative through every point
    for i in range(len(X)):
        MM_dist = []

        # iterate every other point to calculate the Mahalanobis distance
        for j in range(len(X)):
            dis = np.dot(np.dot((X[i] - X[j]), D_I), (X[i] - X[j]).T)[0,
0] ** 0.5
            MM_dist.append(dis)

        # find the k nearest neighbours distance
        MM_dist.sort()
        kn_distance.append(MM_dist[k])
        # print the progress
        # if i % 1000 == 0:
        #     print(str(i / len(X) * 100) + '%')

    # set the Eps to be the upper limit
    # defined by the sum of the upper quartile and 1.5 times the Inter-
quartile Range (IQR)
    # this is prioritized for differentiating the outliers
    Eps = round(np.percentile(kn_distance, 75) + 1.5 * iqr(kn_distance),
3)

    # We find the required parameters!
    return Minpts, Eps
```

Algorithm 4 Find_Eps_for_Supervised (Training_D, D)

Algorithm 4 is the function of finding the recommended parameter *Eps* used in the supervised component of the proposed clustering algorithm to classify the new observations. It requires t2o parameters as input, including the new observations Dataset **X_test**, the training Dataset **Training_D** (or split into X_train and Y_train). *Eps* will be recommended as returned output.

```
def Find_Eps_for_Supervised(X_train, Y_train, X_test):
    """
    Based on the targeted dataset `X_test` itself,
    and the training dataset Training_D (or split into X_train and
Y_train)
    to recommend the parameter (Eps)
    to be used in the Supervised component in the proposed enhanced
DBSCAN algorithm.

    The algorithm takes parameters of the targeted dataset `X_test` (a
list of vectors),
    and the training dataset Training_D.

    It will return one value, Eps.
    """

    # Create a dataframe to hold the training data
    df_train = pd.DataFrame(X_train)
    df_train['label'] = Y_train

    I_df_train = []
    mean_df_train = []

    # iterate every cluster in the trainig data
    for label in np.unique(Y_train):
        # get the label
        df_train_i = df_train[df_train['label'] == label]
```

```
        # get the five attributes of X
        df_X_train_i = df_train_i[[0, 1, 2, 3, 4]]

        # Calculate the covariance matrix
        covMatrix_df_X_train_i = np.cov(df_X_train_i.T, bias=True)

        # If the covariance matrix is singular and do not have inverse
matrix,
        # Use the (Moore-Penrose) pseudo-inverse of a matrix as D_I for
next step
        try:
            D_I = np.matrix(covMatrix_df_X_train_i).I
        except:
            D_I = np.matrix(np.linalg.pinv(covMatrix_df_X_train_i))

        # get center point of the cluster
        mean_df_X_train_i = df_X_train_i.mean(axis=0)

        # store the values in the memory for the use in next step
        I_df_train.append(D_I)
        mean_df_train.append(mean_df_X_train_i)

    # create a output list
    kn_distance = []

    # iterate every point in the test dataset
    for i in range(X_test.shape[0]):
        # only 1/3 of the points are used for getting Eps
        if i % 3 == 0:

            # Point p
            p = X_test[i]

            dist_list = []

            # calculate Mahalanobis distance to each pre-defined clusters
            # iterate every cluster
            for j in range(len(np.unique(Y_train))):
                I_j = I_df_train[j]
                mean_j = mean_df_train[j]

                # calculate Mahalanobis distance
                dist_j = np.dot(np.dot((p - mean_j), I_j), (p -
mean_j).T)[0, 0] ** 0.5

                dist_list.append(dist_j)

            # save the smallest dist
```

```
            kn_distance.append(dist_list.index(min(dist_list)))

    # set the Eps to be the upper limit
    # defined by the sum of the upper quartile and 1.5 times the Inter-
quartile Range (IQR)
    # this is prioritized for differentiating the outliers
    Eps = round(np.percentile(kn_distance, 75) + 1 * iqr(kn_distance), 3)

    # We find the required parameter!
    return Eps
```