ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Thesis

Vessel Future Location Prediction using Neural Networks and TRACLUS algorithm

Lazaros S. Sofikitis
AM P12143

supervisor:
Yannis Theodoridis

Department of Informatics
Piraeus, 2023

Abstract

In this thesis the use of machine learning methods is investigated to tackle the problem of vessel location forecasting. Particularly, the TRACLUS algorithm is applied to vessel trajectories to group similar trajectories together based on their location, and trajectory patterns. Afterwards, a neural network model is trained using the clustered data to predict future vessel locations. The results show that the use of the TRACLUS algorithm improves the accuracy of the model, indicating that clustering can help identify patterns and relationships within vessel trajectories. Overall, this thesis demonstrates the potential of combining neural networks and clustering techniques for improving vessel future location predictions and enhancing our understanding of vessel behaviour in maritime transportation systems.

To my supervisor Y. Theodoridis,
Postdoctoral researcher E. Chondrodima,
and my family.
For their unwavering support and help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Maritime transportation systems play a vital role in connecting people and facilitating global trade. The accurate prediction of vessels' future locations is a critical aspect of ensuring efficient operations and ensuring safety at sea. However, this task is not without its challenges. The dynamic nature of maritime traffic, constantly changing conditions, and the diverse range of mobility data all contribute to the complexity of predicting vessel locations. To overcome these challenges, experts in the field have been working diligently to develop innovative solutions. Utilizing advanced data analysis techniques and incorporating real-time monitoring systems to enhance the accuracy and reliability of vessels' future location prediction methods. By leveraging historical data and applying sophisticated algorithms, experts aim to identify patterns and trends in maritime traffic, enabling more precise predictions. Furthermore, experts in the field are actively engaged in ongoing research and development endeavors aimed at integrating cutting-edge technologies like artificial intelligence and big data analytics into vessel location prediction systems. These innovative advancements hold tremendous promise for transforming the precision and timeliness of predictions, leveraging the wealth of real-time and historical data at our disposal. The integration of machine learning methodologies has shown promise in improving vessel location prediction. By training models on vast datasets comprising information on vessel positions, courses, and other relevant factors, machine learning algorithms can learn to recognize and interpret complex patterns. This allows for more accurate predictions, aiding in route planning, resource allocation, and risk mitigation. By capitalizing on these advancements, stakeholders within the maritime domain can augment their decision-making processes, streamline operational efficiencies, and proactively adapt to evolving circumstances. This concerted effort towards embracing emerging technologies paves the way for a more robust and optimized maritime transportation industry, poised to navigate the challenges of the future with resilience and efficacy. In conclusion, the accurate prediction of vessels' future locations is a multifaceted task that demands a combination of advanced technologies, data analysis, and human expertise. By continually refining our understanding of maritime traffic patterns and leveraging the power of machine learning, we can strive to enhance the efficiency and safety of maritime transportation systems, ensuring the smooth flow of global trade and supporting sustainable economic growth.

The primary objective of this thesis is to employ machine learning methods and apply the TRACLUS algorithm in order to compare vessel location forecasting results with and without clustered data. By achieving this goal, we aim to contribute to the advancement of maritime transportation systems and optimize their operational efficiency.

Chapter 2

Literature Review

The Trajectory Prediction (TP) problem has received a lot of attention in recent years. This general problem can be further classified into two subfamilies: short-term prediction and long-term prediction. [1, 2] The focus of short term prediction models is on predicting the object's next location precisely, whereas long term prediction focuses on predicting possible next locations and routes. The proposed combating methods are mostly Machine Learning (ML) methods that are hybrid in nature, and their components fall into three categories: rule-based prediction approaches, clustering-based prediction techniques, and predictive analytics methods. [3, 1]

2.1 Trajectory and location prediction methods

Various methodologies exist to address the challenge of vessel trajectory and location prediction. This section aims to comprehensively analyze each approach, highlighting their individual strengths and practical applications. By conducting a thorough examination of these methods, we seek to gain insights into their effectiveness in addressing the complexities associated with trajectory and location prediction.

2.1.1 Neural Networks (NNs)

A plethora of research has demonstrated the effectiveness of neural networks (NNs) in trajectory and location prediction, leveraging their ability to learn patterns and relationships from vast datasets to enhance accuracy and gain insights into complex spatial dynamics. As proposed in [4, 5, 6] rigorous experiments have been conducted, applying different NN models. It is derived that using NN models provides us with high accuracy, especially when dealing with complex patterns and large datasets, due to the non-linear relationship modelled between features in NN models. Additionally, NNs tend to be robust and able to handle noisy and incomplete data, making them ideal for handling data without the need for manual feature engineering after they learn the relevant features from raw sensor data. Neural networks can be categorised as static or dynamic. Static neural networks have a fixed architecture, and the weights and biases of the network are pre-determined before the network is used for inference. In contrast, dynamic neural networks have an adaptive architecture where the network can change during inference based on the input data. In tasks with fixed input sizes, such as the one this thesis studied, static neural networks are commonly preferred. Unless it is specified beforehand, from this point on, when a neural network is referred to, it is implied as a static NN. We will expand on the tools used to create NNs for the Trajectory Prediction problem.

2.1.2 Rule-Based Methods

Using rule-based predictions is of great help in trajectory and location prediction problems, especially when the nature of the problem requires limitations such as the road network and aircrafts moving in airway space. Sadly, aquatic vessels sail with a 2-D flexibility that is derived from the non-fixed and sparser seaways. [3] Although rules are crucial for predictions, their application to aquatic vessels is limited due to the nature of maritime travel.

2.1.3 Clustering-Based Methods

Further facilitation of a NN to increase accuracy usually comes in the form of clustering. Especially in aquatic vessels where rule based methods are limited; clustering based methods have a greater role in the improvement of the model. Clustering based methods sacrifice time at the initial stages of building the NN models in order to properly feed the model with well clustered data that will raise the model's accuracy. Notable methods used for clustering is a scalable clustering and Markov chain-based hybrid framework (Traj-clusiVAT)-based Trajectory Prediction algorithm [1] where the technique was created and used to handle big data to create a scalable trajectory classification technique used for both short-term and long-term predictions. Density Based Spatial Clustering of Applications with Noise (DBSCAN) is also a commonly used technique that can be implemented on all data points to create multiple path corridors derived from representative trajectories. As made clear in [4] the knowledge of both trajectories and common sub-trajectories is used to extract crucial data that can further facilitate the NN models. Furthermore, a clustering method like TRACLUS [7] can create simplified representative trajectories from all the data points and find common sub-trajectories based on trajectory line density. In this study, the methodology will delve deeper into the TRACLUS method, which will be the primary area of focus.

2.1.4 Predictive analytic methods

Predictive analytics can be used to analyse big data streams and predict, based on historical data, the future behaviours of objects. In [8] the importance of feature selection and data sampling using decision Trees random forest and support vector machines is highlighted in order to improve the performance of models. In [3] , the predictive analytics framework based on Long Short-Term Memory (LSTM) NNs can be used in a variety of applications, such as maritime traffic management, vessel routing, and maritime safety. The authors highlight its ability to predict future locations of vessels based on learned patterns and compare it with other machine learning models to demonstrate its superiority in terms of efficiency and accuracy. Predictive analytics are also used in [4] in the form of tools such as heat maps and trajectory analysis, to make predictions from data derived from big data technologies, such as Hadoop and Spark, using machine learning algorithms, like random forest and gradient boosting to predict the likelihood of illegal fishing. The i4sea platform is a practical application of predictive analytics that can be also applied in many marine conservation and management domains.

2.1.5 Markov Models

Markov Models (MMs) are a type of probabilistic model that can be used for sequence prediction tasks such as speech recognition and natural language processing [9] . Existing methods for the TP problem are based mostly on data-driven methodologies, such as Hidden Markov Chains and Neural Networks (NN) [3] . An example of Markov models can be seen in [1] where a combination of MMs and NN models were used to predict the trajectories of objects such as vehicles, pedestrians, and animals. To address the limitations of existing approaches, they used a hierarchical approach that combines MMs and NNs. The MMs capture short-term dependencies in the trajectory data, while the NNs capture long-term dependencies, thus providing more accurate predictions overall. A similar approach can be seen in [10] where the study proposes a hybrid model that combines a feedforward neural network with a Markov model. The neural network is used to capture long-term dependencies in the trajectory data, while the Markov model is used to capture short-term dependencies.

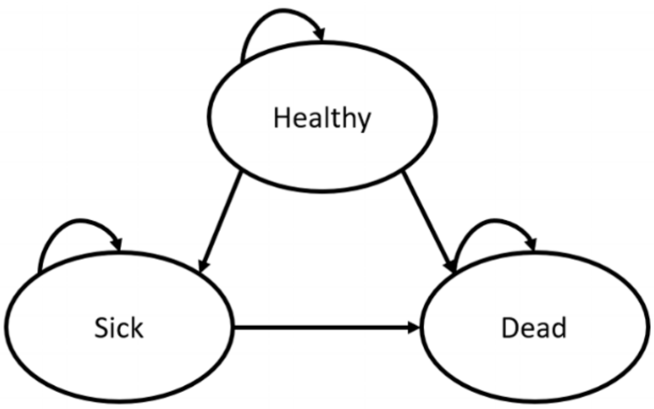


Figure 2.1: A three state Markov Model.
Published in: Generating Survival Curves from Study Data: An Application for Markov Models (Part 2 of 2), Mark Bounthavong March 15, 2018.

2.2 Limitations and Challenges

Although the methods referenced in Chapter 2.1. are used due to their competence and efficiency in solving the TP family of problems, this section will focus on, and analyse their general drawbacks.

2.2.1 Limitations of Neural Network models

NN models may encounter limitations when the data they rely on is of low quality or insufficient in quantity, for this reason, researchers exercise caution to avoid using unsuitable data for NN models. Sometimes, the resulting models can be challenging to interpret, making it difficult for researchers to comprehend the rationale behind specific predictions. This becomes especially problematic in domains where understanding the reasoning behind predictions is crucial. During the training process, special attention is devoted to preventing overfitting, which can cause the model to perform poorly on test data and struggle to generalize to new data. Throughout the model design phase, extensive checks are conducted on the training data to mitigate bias and address fairness concerns. This ensures that the resulting model is developed in a manner that is fair and unbiased. Additionally, selecting the appropriate hyperparameters can be a daunting task as different hyperparameters possess distinct strengths and weaknesses. Making a poor choice in hyperparameters selection can lead to unsatisfactory outcomes. Therefore, careful consideration is required when navigating the selection process to achieve the desired results.

2.2.2 Limitations of Clustering Techniques

Clustering-based techniques are often based on distance measures and are sensitive to noise and outliers, especially if those are not handled correctly. Additionally, their simplistic nature makes them unable to capture complex relationships between variables, which can cause underfitting. They usually require a posteriori knowledge of the optimal number of clusters [3] , which may not be known a priori, and an incorrect choice of cluster number can lead to inaccurate predictions. In regards to clusters, points in the same cluster are treated as similar between them and dissimilar with points in other clusters, which, in practice, may not be the case since clusters can be overlapping or have non-uniform densities. Lastly, they are not well suited for time-series problems due to their lack of understanding of temporal dependencies between data points, leading to inaccurate predictions when the nature of the data contains complex patterns or trends.

2.2.3 Limitations of Predictive analytics methods

Predictive analytics methods, such as Markov Models and hidden Markov Models, have proven to be very useful in making predictions, but their predictive ability is confined to the data that is used to train the model, giving them a limited scope [10] . If the data is not representative of the general population, the predictions will not be accurate. Furthermore, it is important to note that regression models can only identify correlations between variables, not causation. Thus, predictions by the model should be interpreted with caution, as there may be underlying variables that are not captured in the model. Finally, variables with nonlinear relationships are not properly understood by models using predictive analytics in view of the fact that they assume linearity between input variables.

2.2.4 Limitations of Markov Models

Specifically Markov models, assume that the probability of an outcome depends on the previous state, disregarding more complex relationships of higher order between variables. This in turn shows the model's dependency on the initial conditions; if the initial conditions are not accurately known, the predictions are unreliable. Even with those factors accounted for, Markov models do not have the ability to update the probability distributions of their states over time, and they assume that the future is independent of the past, being unable to take into account the history of the process beyond its current state [3] . Lastly, their dependability on large sample sizes to estimate the transition possibilities between states makes them unreliable for smaller sample sizes.

2.2.5 Limitations of rule-based algorithms

In general, applying pre-defined rules to the data can make the algorithm inflexible and unable to adapt to changes in the data or new scenarios. Also, the complexity of the rules can make it challenging to interpret the predictions or identify the rules that contribute the most to the predictions. Lastly, applying rule based algorithms implies that the rules applied capture all the relevant features of the data, which can be difficult to estimate in complex systems and can create an inability to handle noise and outliers, which sequentially impacts the performance of the model.

Chapter 3

Theoretical background

3.1 Machine Learning and Neural Networks.

Machine learning (ML) is a field that falls under the umbrella of artificial intelligence (AI) and computer science. Its primary focus is on leveraging algorithms and data to replicate human learning processes and gradually enhance accuracy over time. Neural networks, a popular subset of machine learning, are computational systems inspired by the structure and function of biological brains. Neural Networks (NNs) consist of interconnected nodes, known as artificial neurons, which mimic the behavior of neurons in the human brain. These neurons process and transmit information through weighted connections, enabling the network to identify patterns, make predictions, and even generate new data. Machine learning utilizes algorithms and statistical models to analyze data and enhance performance in specific tasks. The core objective is to develop algorithms that can make accurate predictions without explicit programming. This approach facilitates a deeper understanding of problems, revealing insights into causation and correlation among variables. Fundamentally, machine learning focuses on building models that can generalise and adapt to examples in a dataset, with the goal of making accurate predictions and decisions about new data. The life cycle of machine learning models includes several steps: problem definition, data collection, data preprocessing, model training, model evaluation, model deployment, and model maintenance. In general, the creation and maintenance of a machine learning model is an iterative process with the focus on constant and continuous improvements to achieve optimal performance on the task at hand. There are several different types of machine learning algorithms; the most common types include reinforcement learning, unsupervised learning, and supervised learning. Reinforcement algorithms use a trial-and-error technique and aim to maximise a reward based on their actions. Unsupervised techniques learn from unlabeled data and aim to find patterns and structure in the data; on the other hand, supervised techniques use labelled examples where the input data is associated with the output target data. Machine learning has many practical applications in various domains, including healthcare, finance, marketing, natural language processing, and automotive. Most notably, it has been used for tasks such as image recognition, speech recognition, fraud detection, and driving autonomous cars. In this study, the focus will be on supervised training and its application to sequential problems. Sequential problems involve using an input sequence to output a prediction. An example of that could be predicting the future values of a time series based on past observations.

3.2 Neurons | Perceptrons

Neurons, also called perceptrons, are the building blocks of a neural network; in operation, they mimic the neurons of the brain. Within a neural network, neurons are organised into layers. The input layer receives input data, such as an image, number, or text, and passes it to the first hidden layer of neurons. Each subsequent layer processes the output of the previous layer and passes it to the next layer, until the final output layer produces a prediction or classification. Specifically, the neuron is a set of inputs, a set of weights, and an activation function. These inputs can either be raw input features or the output of neurons from an earlier layer. The neuron translates these inputs into a single output, which is then passed through an activation function and picked up as input for another layer of neurons. Each neuron has a weight vector for every input to that neuron. The weights and biases for each neuron are adjusted based on the error between the predicted and actual output during the training stage, such that the final network output is biassed toward some preferred value.

(a) The structure of a neuron cell

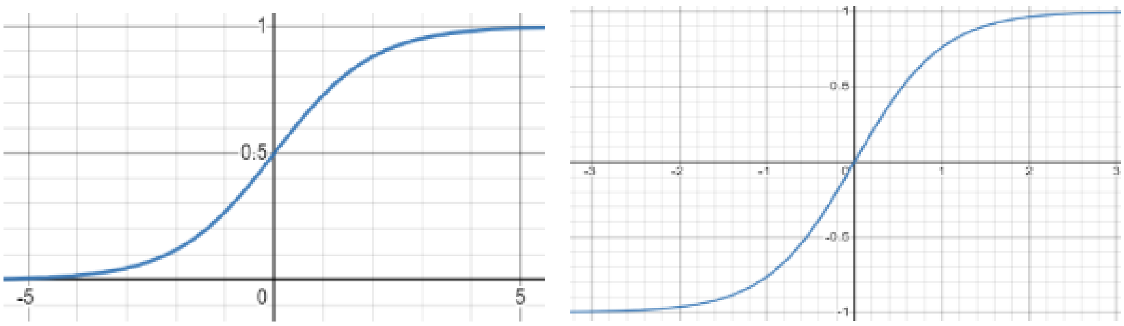
(b) The structure of a perceptron

Figure 3.1: The similarities between biological neurons and perceptrons.

3.2.1 Activation Functions

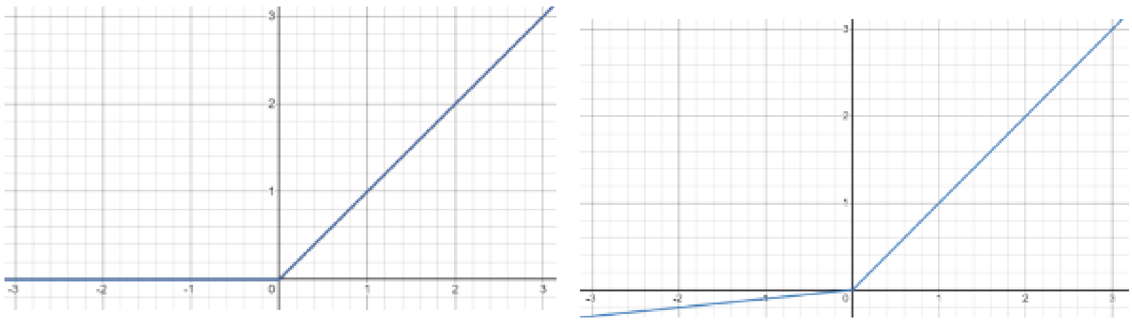
The output of each neuron passes through an activation function before continuing to the next layer of neurons. The purpose of an activation function is to introduce non-linearity into the output of a neuron, allowing the network to model complex relationships between inputs and outputs. Without an activation function, a neural network would simply be a linear regression model. Typically, activation functions are nonlinear and monotonically increasing. Some commonly used activation functions include the sigmoid function, the Hyperbolic Tangent (Tanh) function, the Rectified Linear Unit (ReLU) function, and its variants such as leaky ReLU and Exponential Linear Unit (ELU). The activation functions can either be differentiable or non differentiable. The choice of activation function can have a significant impact on the performance of a neural network, and appropriate consideration is given.

Supervised problems are categorised as classification or regression problems. Activation functions that are commonly used for classification problems belong to the sigmoid family. A



(a) Sigmoid function (b) Hyperbolic tangent function

Figure 3.2: (a) the sigmoid function and (b) the hyperbolic function tanh.
Both the sigmoid and the tanh function are examples of differentiable, monotonically increasing functions.



(a) ReLU function (b) Leaky ReLU function

Figure 3.3: (a) the rectified linear unit function and (b) the leaky rectified linear unit function.

The ReLU and Leaky ReLU functions are examplesof non-differentiable, monotonically increasing functions. Non-differentiable functions can create problems with learning, as numerical gradients calculated near a non-differentiable point can be incorrect.

sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point and exactly one inflection point. Common activation functions in this family are the sigmoid, hyperbolic tangent, and softmax functions. The sigmoid function compresses the output into the range [0, 1] most commonly it is used for binary classification, where the output can be interpreted as the probability of belonging to the positive class. This function has a bias towards 0 and 1.The tanh function compresses the output into the range [-1,1] most commonly it is used for classification of zero-centred data or data that also contains negative values. This function has a bias towards -1 and 1.

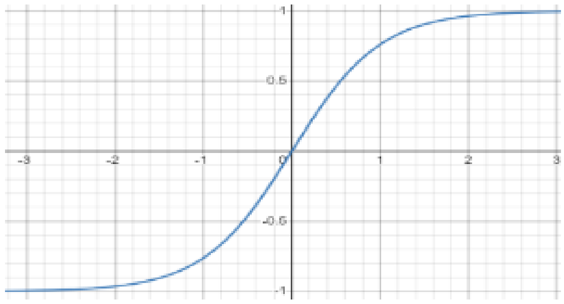


Figure 3.4: The softmax function.
Regarding classification problems, the softmax function takes as input a vector of real-valued scores, which can be seen as the evidence for each possible class, and outputs a probability distribution over the classes that sums to 1. The output of the softmax function can be interpreted as the model's confidence in the prediction for each class.

For regression problems, the goal is to predict a continuous numerical output value. Typically, activation functions for regression problems have piecewise linearity, which further helps calculate a continuous numerical output. Common activation functions for regression problems are the linear activation function, ReLU, Leaky ReLU, and Softplus. Note that the Tanh function is nonlinear but is symmetric around 0 and can handle negative values, making it useful for regression problems.

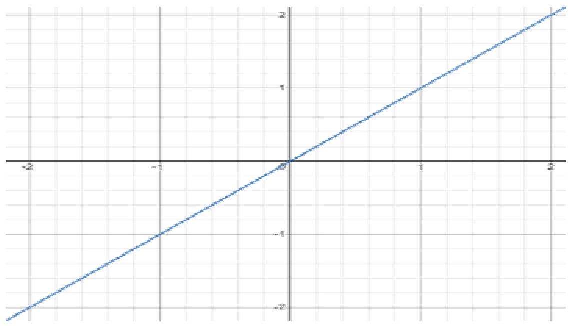


Figure 3.5: The output of the linear function.
The output of the linear function is the input value multiplied by a constant factor and possibly shifted by another constant factor (bias), without any non-linear transformation.

$$output = (input \times weight) + bias$$

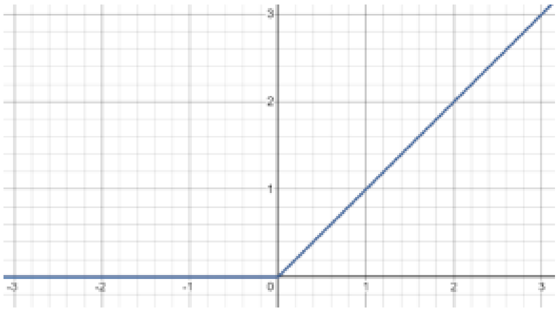


Figure 3.6: The ReLU function.
The ReLU function returns zero for all negative input values and the input value itself for all non-negative values. This can help introduce non-linearity in the network.

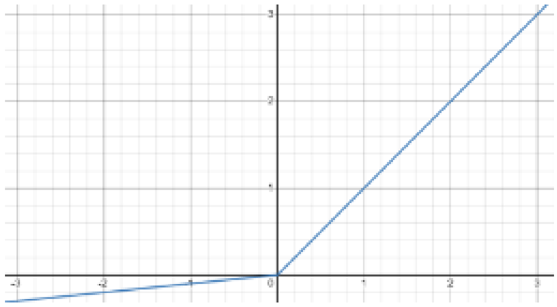


Figure 3.7: The Leaky ReLU function.
The Leaky ReLU is a variant of ReLU, with the key difference being that for negative input, it doesn't output 0, but outputs a small constant multiplied by the input value. This prevents neurons from getting stuck with zero output and no gradient being propagated through them during backpropagation, leading to no updates of their weights. (The dying ReLU problem).

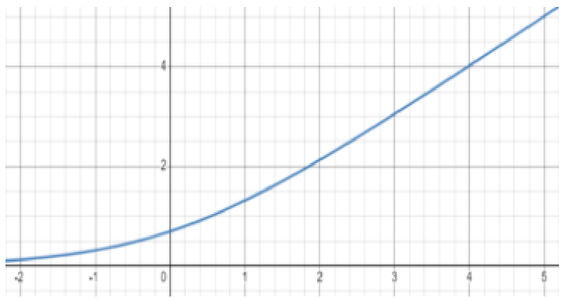


Figure 3.8: The Softplus activation function.
The Softplus activation function is similar to the ReLU activation function, but it is a smooth, differentiable function $f(x) = ln(1 + e^x)$

3.3 Cost functions

In machine learning, cost functions hold significant importance as they serve as a metric to evaluate a model's accuracy. Their purpose is to guide the model towards finding the best combination of parameters by minimizing the discrepancy between the actual and predicted outputs during the training phase. The cost function quantifies the error or cost associated with the model's predictions by considering the predicted output and the true output, yielding a value that the model strives to minimize. When it comes to choosing a cost function, it is crucial to consider the specific task and the type of model being utilized. For regression problems, which involve predicting continuous values, the Mean Squared Error (MSE) function is frequently used. This function calculates the average of the squared differences between the predicted and actual values. On the other hand, in classification problems where the objective is to predict categorical labels, the cross-entropy loss is a popular choice. This loss function measures the dissimilarity between the predicted probability distribution and the true distribution of labels. The selection of the appropriate cost function depends on the nature of the problem and the desired outcome, ensuring that the model's performance is appropriately evaluated and optimized. In the realm of cost functions, there exist various options including Mean Absolute Error (MAE), hinge loss, and l1 loss, among others. The selection of the appropriate cost function is contingent upon the specific demands of the task at hand and the characteristics of the model being utilized. During the development of a model, it is customary to experiment with multiple cost functions to identify the one that produces the most effective outcomes. It is important to note that the choice of the most suitable cost function may evolve over time as researchers continue to make advancements in the field, enhancing our understanding of different tasks and models.

Mean Squared Error	$MSE = \frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{n}$
Root Mean Square Deviation	$RMSD = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{n}}$
Mean Absolute Error	$MAE = \frac{\left\lvert \sum_{i=1}^{n} (y_i - \widehat{y}_i) \right\rvert}{n}$
Mean Absolute Percentage Error	$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left\lvert \frac{y_i - \widehat{y}_i}{y_i} \right\rvert$

Figure 3.9: Commonly used cost functions
Notations: (a) y_i = observed value, (b) $\hat{y}_i$ = estimated value, (c) n= number of data points, (d) i = iterative variable i.

3.4 Backpropagation

Backpropagation is a fundamental algorithm utilized to minimize the scalar value of the cost function. It plays a crucial role in computing the gradients of the loss function with respect to the weights in a neural network. Through methods such as gradient descent, the network can iteratively approach an optimized state. The backpropagation algorithm draws upon the chain rule of calculus, enabling the calculation of the derivative of a composite function. In the context of a neural network, where the inputs and outputs are represented as multi-dimensional arrays of numerical data (tensors), denoted as $T_i[m, n]$ and $T_o[j, k]$ respectively, each layer of the network can be regarded as a function that transforms $T_i[m, n]$ to $T_o[j, k]$. This implies that the derivative of the cost function with respect to the weights can be computed by sequentially applying the chain rule in a backward manner, propagating the gradients through the layers of the network.

Firstly, the input data is fed through the layers of the model, and the output for each layer is computed based on the current weights. Then, using the selected cost function, the error between the predicted output and the actual output is calculated. Next, the derivative of the loss function with respect to the weights of each layer is determined by applying the chain rule. This derivative calculation begins from the output layer and progresses backward through the layers of the network. The gradients provide information about how each weight contributes to the overall error. Finally, the weights of the network are updated using the computed gradients. This update process typically employs optimization techniques such as gradient descent to adjust the weights in a way that minimizes the loss function. The entire backpropagation process is iteratively repeated until the network reaches a state where the loss is minimized or converges to a satisfactory level. By continuously updating the weights based on the calculated gradients, the model gradually learns to make better predictions and improve its overall performance.

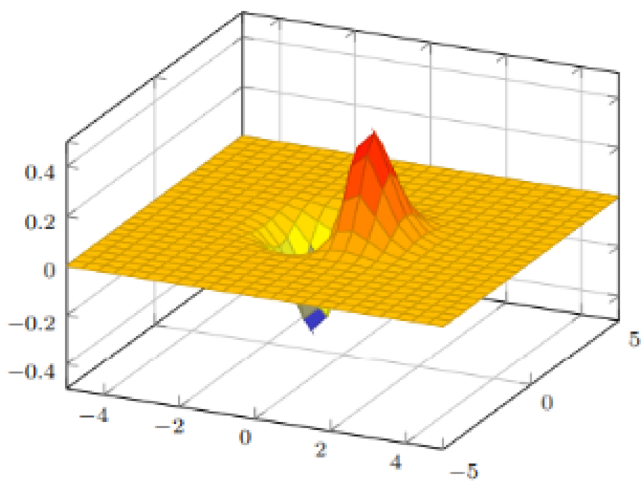


Figure 3.10: A 3d representation of the cost function.
The goal of backpropagation is to guide the model to reach the lowest point with regards to cost.

3.5 Feedforward Neural Networks

Feedforward Neural Networks, also known as MultiLayer Perceptron (MLPs), are a type of artificial neural network where the layers of the model are organised in a serial manner and each neuron is connected to all neurons in the previous and subsequent layers. In other words, the output from one layer is fed as input to the next layer, and so on, while the order of the layers stays fixed. As a neural network, it consists of an input layer, hidden layers, and an output layer. The information travels in one direction, from the input layer to the output layer. Each layer processes the information from the previous layer and passes it forward sequentially. As models, they are easy to design and interpret and can model complex nonlinear relationships between the input and output data, making them useful in applications such as speech and image recognition, natural language processing, predictive modelling, and time series analysis.

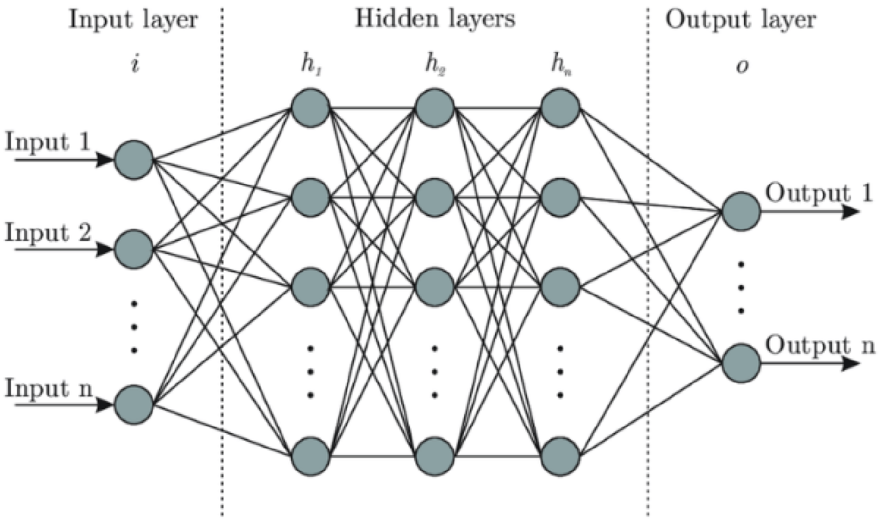


Figure 3.11: A Simple feed forward fully connected Neural Network.
Published in: Python in Plain English, Apr 30, 2020

3.6 Trajectory Data

Trajectories are a key concept in machine learning. Understanding the trajectories of objects can help in predicting the behaviour of complex systems and optimising the performance of machine models. Spatial trajectories can inform policy and decision-making in areas such as urban planning, transportation, and emergency response. Mostly, they are influenced by a range of factors, including geography and climate. In order to study spatial trajectories, extremely large datasets (Big Data) need to be analysed and various tools and techniques are applied, namely visualisation tools such as Geographic Information System (GIS) in conjunction with spatiotemporal datasets from the Automatic Identification System (AIS).

3.6.1 Big Data

Big Data are too complex or varied to be analysed using traditional data processing tools. They can be generated from various sources, such as social media, online transactions, and sensor networks. The main features when mentioning big data are the three Vs (volume, velocity, and variety). Volume refers to the size of the data; velocity is the speed at which these data are generated and need to be processed; and variety refers to the diverse range of data types inside the dataset. Big data has become important in recent years due to the rise of new technologies that allow the creation, manipulation, analysis, and processing of large datasets. The challenge of processing and analysing such datasets is the epicentre of many studies in the field of computer science.

3.6.2 Automatic Identification System (AIS)

The Automatic Identification System (AIS) is a technology used in the maritime industry that uses transceivers, radio signals, marine radars, and sometimes satellite signatures to track and identify vessels in the surrounding area and share this information with nearby vessels and authorities. The vessels using AIS broadcast real time information such as position, course, speed, and vessel identity, which is collected and sent to a central database to be accessed and used by authorised users. Its main uses are collision prevention, fishing fleet monitoring, accident investigation, and search and rescue missions. Furthermore, it can be used to monitor vessel traffic and improve shipping routes through the use of computer algorithms. Some potential limitations of AIS fall in the fields of data privacy, cyber attacks, signal jamming, and the possibility of data emission errors. However, AIS is an important technology that continues to develop in order to improve safety, efficiency, and sustainability. In conjunction with GIS, it is a powerful tool that can help create machine learning models for trajectory prediction, autonomous shipping, and dynamic collision avoidance.

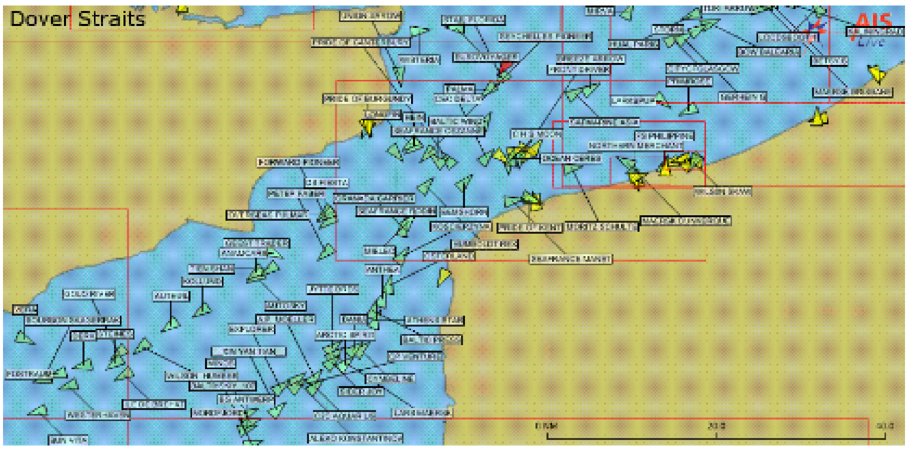


Figure 3.12: A graphical display of AIS data on board a vessel.

Chapter 4

Maritime Data Preparation

Data is an essential ingredient in the realm of machine learning, playing a pivotal role in training models and enabling accurate predictions. It serves as the backbone, empowering algorithms to learn patterns and relationships. Carefully selecting, preparing, and utilizing data is critical for building robust models. From training data that represents real-world scenarios to preprocessing techniques that clean and enhance data quality, every step contributes to the model's accuracy and reliability. This chapter describes in detail the methods and tools used to transform, prepare and utilise the data.

4.1 Dataset Description

The data used is from passenger vessels for the month of January 2018 with the use of AIS. The focus will be on the Aegean Sea. Identifying the longitude and latitude values that will be the boundaries of the region in question will help clean the data from outliers. The boundaries of the Aegean Sea are not fixed and can vary depending on the source of information. Arbitrarily, the rectangle created from the coordinates 35.660669° N, 39.517331° N, 23.088642° E and 26.896934° E encompasses a large part of the Aegean Sea and fragments of the surrounding landmass. A larger window encompassing the whole Aegean Sea is technically feasible but not expected to provide additional meaningful insights beyond what the chosen window already offers. Considering computational efficiency is important when dealing with large datasets, the selected boundaries offer appropriate balance between computational efficiency and expected insights.

Figure 4.1: The user defined boundaries of the Aegean sea for the problem at hand.

4.2 Data Preparation

Data preparation, also known as data Wrangling, refers to the process of preparing data for training and testing machine learning models. This includes tasks such as data cleaning, data transformation, data encoding, and data splitting. The goal of data preparation for machine learning is to ensure that the data used to train the machine learning model is of high quality and can be effectively used to learn patterns and relationships in the data. Data preparation is considered one of the most time-consuming and crucial steps in the life cycle of a ML project. Given a raw dataset, data preparation involves the process of removing errors,checking for inconsistencies, and in general, ensuring reliability, accuracy, and consistency across the dataset.

Since a regional window was already chosen, the first step will be to delete all rows from the dataset that are beyond the window that is allowed. Regarding the homogeneity and standardisation of the data, care is given to every column, and duplicate rows are deleted. e.g., timestamps are converted to unix integers with a datetime pandas conversion, and longitude and latitude degrees are turned to World Geodetic System 1984 (WGS84) decimals. Columns that are considered unnecessary or do not inspire confidence are deleted, e.g., the speed column. Lastly, columns that

Table 4.1: Sample Preprocessed Data.

[10.283.171 rows x 8 columns].

mmsi	timestamp	lon	lat	shiptype	speed	course	heading
355931000	2018-10-02 20:11:12	23.870230	37.402481	60	174	341	341.0
237233600	2018-10-02 20:11:14	25.740101	35.007332	65	0	359	511.0
237294700	2018-10-02 20:11:16	23.542440	37.959049	60	71	86	511.0
271041325	2018-10-02 20:11:16	27.429489	37.034210	60	0	0	511.0
271002606	2018-10-02 20:11:16	27.954029	40.355400	60	0	0	511.0
...	...	...	...	...	...	...	...
237641000	2018-10-02 20:10:56	23.869181	37.349380	69	200	157	151.0
249727000	2018-10-02 20:11:01	24.457350	38.598270	60	141	316	316.0
237032000	2018-10-02 20:11:07	24.736830	35.937832	60	206	327	328.0
271044139	2018-10-02 20:11:07	28.730591	40.890369	60	108	323	511.0
239575000	2018-10-02 20:11:07	23.834829	37.381168	60	188	158	158.0

will be helpful later on are added, e.g., the column dt, which denotes how much time has passed from one signal to the next, and the new speed column.

After completing the basic data wrangling mentioned above, further preparation is applied to the data. The trajectories of the vessels that are denoted by their MMSI identification are split based on logical assumptions. A trajectory cannot be less than 10 timesteps or more than 1.000 timesteps, and if there is a gap between signals of more than 30 minutes, the trajectory is split at that point. Furthermore, a trajectory cannot last more than 24 hours. Regarding the vessels' speed, if the speed is less than 0.1 nautical knots, it is considered stopped; the trajectory is then split ata that point. Finally, if the speed is greater than 25.7 nautical knots, it is considered unrealistic, and the trajectory is split. This process is iterative and continues until no further changes are made by the program. For the final step, basic data analysis is applied, the dataset is shuffled to avoid overfitting when training, and it is split into three parts for training, validation, and testing. Overall, data cleaning is a critical step in the data analysis process that ensures that the data is reliable, accurate, and consistent and can be used for further analysis or modelling. Data cleaning requires careful attention to detail and a solid understanding of the data and the problem being solved.

mmsi	tr1_val2_test3	id	WGS84lon	WGS84lat	t	lon	lat	dist_m	dlon	dlat	dt	speed
237018400	1	1	23.530130	39.166431	1527491036	200205.617017	4.340983e+06	1086.690420	869.729229	651.511426	171.0	6.354915
237018400	1	1	23.538771	39.171829	1527491194	200975.271999	4.341554e+06	958.158259	769.654983	570.699970	158.0	6.064293
237018400	1	1	23.549900	39.178848	1527491404	201966.648647	4.342296e+06	1238.622488	991.376647	742.534856	210.0	5.898202
237018400	1	1	23.558029	39.183849	1527491555	202690.132658	4.342825e+06	895.938895	723.484012	528.467018	151.0	5.933370
237018400	1	1	23.569630	39.191471	1527491774	203724.494176	4.343633e+06	1312.623793	1034.361518	808.132089	219.0	5.993716
...	...	...	...	...	...	...	...	...	...	...	...	...
240080500	3	53715	23.543800	37.959721	1524390974	196343.777405	4.206983e+06	403.029610	-400.377580	-46.159074	123.0	3.276663
240080500	3	53715	23.538919	37.959450	1524391102	195913.693696	4.206969e+06	430.315890	-430.083709	-14.133955	128.0	3.361843
240080500	3	53715	23.534700	37.960499	1524391227	195547.244059	4.207099e+06	388.906798	-366.449637	130.242700	125.0	3.111254
240080500	3	53715	23.531450	37.964001	1524391362	195276.115323	4.207499e+06	482.708402	-271.128736	399.370267	135.0	3.575618
240080500	3	53715	23.530300	37.965599	1524391501	195181.669898	4.207680e+06	204.291116	-94.445425	181.148894	139.0	1.469720

Table 4.2: The data after processing

4.2.1 Converting timeseries to a supervised learning problem

Given a timeseries with 3 variables X,Y,Dt where X is the longitude, Y is the latitude, and Dt is the time difference from one timestep to the next, the goal is to create $n \times 3$ new columns, where n is the number of lagged timesteps we want to create. Every new column will be the same as its predecessor but shifted by one step.

Table 4.3: Example mock Timeseries problem.

Longitude X	Latitude Y	Time Dt
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9

Table 4.4: Example mock Timeseries problem converted to supervised problem of lag timesteps 2.

Longitude X_{-1}	Latitude Y_{-1}	Time Dt_{-1}	Longitude X	Latitude Y	Time Dt
1	1	1	2	2	2
2	2	2	3	3	3
3	3	3	4	4	4
4	4	4	5	5	5
5	5	5	6	6	6
6	6	6	7	7	7
7	7	7	8	8	8
8	8	8	9	9	9
9	9	9	Nan	Nan	Nan

For the purpose of this research, a step of $n = 10$ is used, resulting in 156 columns. At this point, the dataset is checked to avoid rows that contain trajectory identities that do not match. After further cleaning and removal of redundant columns, the dataset left contains, in total, 66 columns. This technique is commonly used for machine learning; converting the dataset to a tabular form helps create a table where every row contains information about the object for the last n observations. The rows of the dataset that contain missing values are removed, the data wranlging program is applied again and the dataset created is split into training, validation, and testing sets to further accommodate solving the supervised problem.

Table 4.5: The input of the algorithm.

dlon(t-10)	dlat(t-10)	speed(t-10)	dt(t-9)	dlon(t-9)	...	dt(t)
678.048594	-265.691638	4.887556	141.0	616.915065	...	131.0
616.915065	-296.483109	4.854331	220.0	616.915065	...	131.0
955.288911	-440.407878	4.781455	131.0	577.215446	...	60.0
577.215446	-233.752623	4.753819	140.0	615.886767	...	220.0
615.886767	-211.968419	4.652447	149.0	606.261439	...	180.0
...	...	...	...	...	...	...

Table 4.6: The goal output of the algorithm.

dlon(t)	dlat(t)
676.550407	-198.792543
633.754767	-178.405028
592.817516	-164.923137
582.374405	-194.422268
616.910006	-206.865550
...	...

Lastly, the resulting tables are split into two tables, one of observations $(t-10)...(t-9)$ and one of observations (t). The first will be used as the input of our algorithm, while the latter will be the goal output of the supervised problem. This tabular conversion is a powerful tool that helps make accurate predictions based on past observations. The training set will be the set on which the model will train its parameters, while the validation dataset will be the control set that judges accuracy and helps avoid overfitting. Lastly, the testing set will be the dataset that will judge the general accuracy of the model with data it has not seen yet. Of the original set, 55% will be used as the training set, 15% will be used as the validation set, and 30% will be used as the testing dataset.

Chapter 5

Methodology

In this study, we developed a method to predict the future locations of maritime vessels by applying a clustering algorithm that clusters similar trajectories together and then training a feed forward neural network that predicts the future positions of vessels based on the cluster they belong to. To evaluate the results, we compare the method's accuracy with the results of a feed forward neural network trained without clustered data. The following sections describe in detail the tools, and techniques used in this study to carry out location prediction with the use of NNs and trajectory clustering with the TRACLUS algorithm [7].

5.1 Definition of the trajectory prediction problem.

Given a) a trajectory of a moving object: $\{p_0, t_0, p_1, t_0 + \Delta t_1, ..., p_i, t_{i-1} + \Delta t_i\}$, consisting of i transitions of the object, and b) a time interval Δt_{i+1}, the goal is to predict the expected position p_{i+1} of the object at the time $t_{i+1} = t_i + \Delta t_{i+1}$. Practically, given the "when" component, the goal is to predict the corresponding "where" component, e.g., where the vessel will be in the next few minutes.

5.2 The Neural Network model architecture

When attempting to create a neural network model, various parameters have to be considered. A Feed Forward NN was chosen due to its simplicity, training speed, and flexibility. The architecture of the model consists of an input layer with 42 variables, 4 dense hidden layers with populations of 32, 16, 8, and 4 neurons, respectively, and an output of 2. Various layer compositions were tried. This decreasing layer approach was chosen to extract increasingly abstract features from the input data, making it effective in capturing important feature points while minimising the risk of overfitting. The data were scaled before passing through the model, depending on the activation function used each time. The best results were given when using a standard scaling and the ReLU activation function for every layer. Regarding the hyperparameters of the model, the chosen cost function was the mean absolute error (MAE) and the mean squared error (MSE) while the optimising method used was an extended version of stochastic gradient descent called Adam optimiser. Although the Adam optimizer requires more computational power, it has some benefits, making it a better choice for the problem at hand. Some benefits of the Adam optimizer against stochastic gradient descent are its ability to use adaptive learning rates for each weight and its ability to use both gradient and momentum to update the weight parameters. Finally, regarding the iterations of the training process, two methods were used. A check mechanism that compares the accuracy of the validation data of the current epoch with the validation accuracy of the previous epoch; if the new weights are more accurate, they are saved; and a patience parameter of size 10 that checks if any meaningful improvements have been made; if for 10 consecutive epochs no meaningful improvements are made, the iteration stops. The model showed best results in the range of 15 to 25 iterations.

5.3 Clustering trajectories

In order to understand the process of trajectory clustering, an understanding of general clustering is needed first. There are three main factors to take into account. Firstly, the distance metric; when clustering data points, a metric is needed to judge if points should be clustered together. Secondly, the clustering algorithm; in this study, an algorithm based on density, Traclus, was chosen [7]. Lastly, the number of clusters that best suit the problem at hand.

5.4 Representative trajectories

Regarding trajectories, the first action needed for computational speed is to create representative trajectories. In this Chapter, the dataset was created, and every trajectory was given a characteristic identity (id). Given the spatiotemporal points of a trajectory, a simplified representative trajectory can be extracted without sacrificing much in terms of accuracy. The goal is to partition every trajectory of n points into its characteristic points to create a representative trajectory with fewer points. This process requires the use of the minimum description length principle (MDL). The program traverses the trajectories sequentially, and at every point it computes the difference between the real trajectory and a linear trajectory from the origin point to where it is currently located. If the cost of partitioning is greater than not partitioning, then the point prior to the one the program is currently checking is considered a characteristic point. The number of traversed points is set to 0, and the new characteristic point is now considered the origin. The program continues until there are no more trajectories to check and creates a new dataset of characteristic trajectories. To accomplish the goal mentioned, the creation of a distance function is needed. The distance function was made up of three components. the perpendicular distance, the parallel distance, and the angular distance. The perpendicular distance is provided by the Lehmer mean; the parallel distance is the minimum distance between the origin of the line and the first projection of the line segment on it, and the second projection up to the end of the line. Lastly, the angular distance is defined as the sine of the angle between the real line and the representative line times the length of the real line.

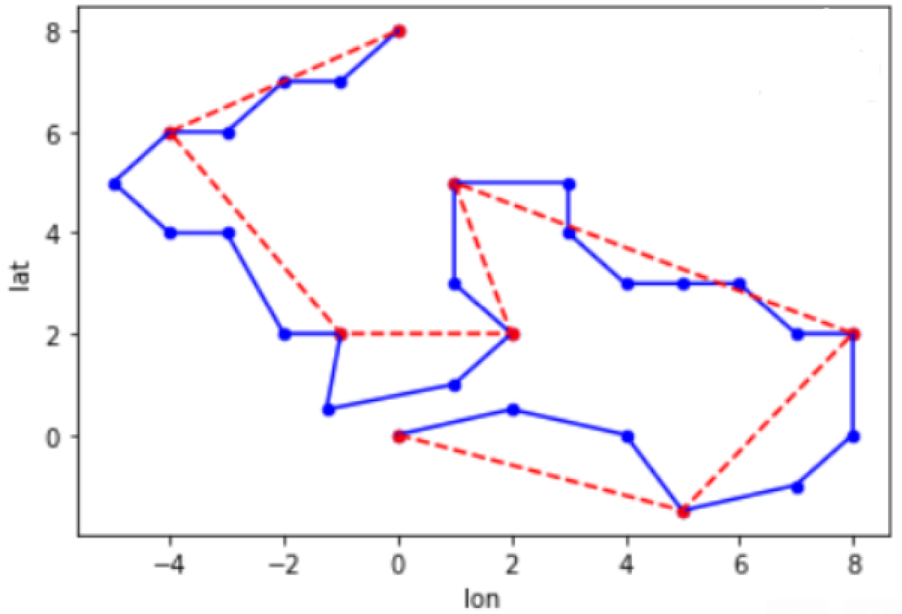


Figure 5.1: An example of trajectory partitioning
blue: The example trajectory of n points.
red: The representative trajectory made from j characteristic points

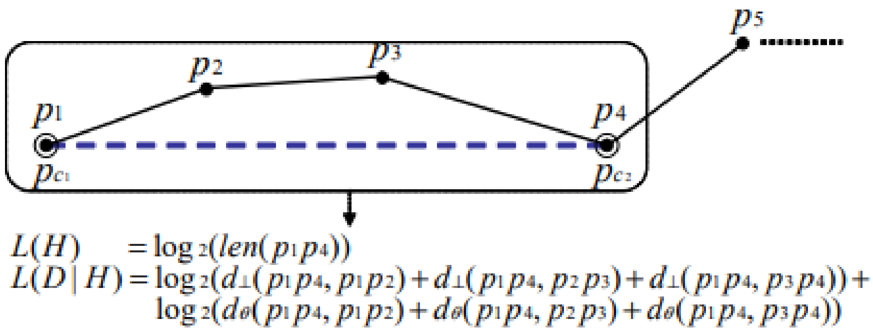
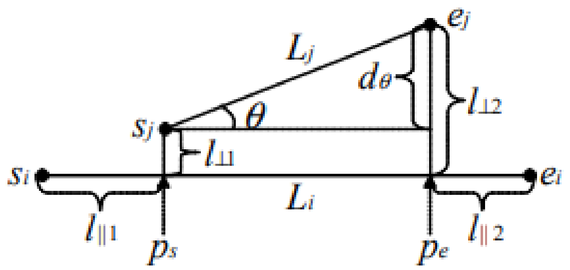


Figure 5.2:
Formula of the MDL cost.
$L(H)$ is the hypothesized partition. $L(D|H)$ is the sum of the difference between a trajectory and a set of its trajectory partitions p_1, p_2, p_3, p_4. The best hypothesis minimises the sum $L(H)(D|H)$
$d_{||}$ = parallel distance, $d_\perp$ = perpendicular distance, d_θ = angular distance
Published in ACM SIGMOD Conference 2007
Jae-Gil Lee, Jiawei Han, K. Whang

The sum of the three distances is considered the total distance between two lines .

$$dist(L_i, L_j) = w_1 \times d_{||}(L_i, L_j) + w_2 \times d_\perp(L_i, L_j) + w_3 \times d_\theta(L_i, L_j) \tag{5.1}$$

In this study, when applying the (5.1) formula for trajectory partitioning, the weights of the perpendicular and angular distances are considered equal to one, while the weight of the parallel distance is considered equal to zero. This decision was made with regards to fairness and equality, since the parallel distance is zero for the lines that share the same origin as the characteristic points when traversing a trajectory. Lastly, when using the distance formula for clustering all weights were equal to one, since there where no doubts in regards to fairness and equality.



$$d_\perp = \frac{l_{\perp 1}^2 + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}}$$

$$d_\parallel = \mathrm{MIN}(l_{\parallel 1}, l_{\parallel 2})$$

$$d_\theta = \|L_j\| \times \sin(\theta)$$

Figure 5.3: The three distance metrics.
If $\theta > 90$ we consider $sin(\theta) = 1$.
Published in ACM SIGMOD Conference 2007
Jae-Gil Lee, Jiawei Han, K. Whang

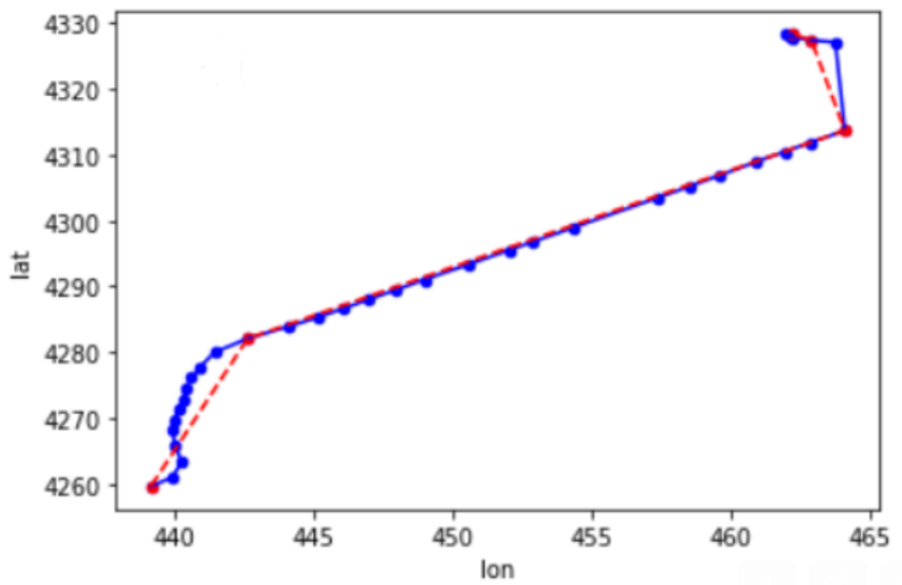


Figure 5.4:
Result of a real trajectory partitioning.

5.4.1 clusters of representative trajectories

After performing the actions mentioned above, the resulting product is a dataset containing each characteristic line. In order to continue and perform the clustering on the trajectories, a new dataset is created containing the distances used in Figure 5.3 between all characteristic lines. This dataset will be crucial for applying the altered DBSCAN algorithm to the line segments and will help to calculate the heuristic parameters of ϵ and Minimum neigbours (MinLns). The ϵ parameter is the distance threshold that is used to define a radius around each line in the dataset; in this case, the radius results in transformed ellipses. Lastly, MinLns is the parameter that specifies the minimum number of lines that must be within the epsilon radius of a core line in order for that line to be considered a part of a cluster.

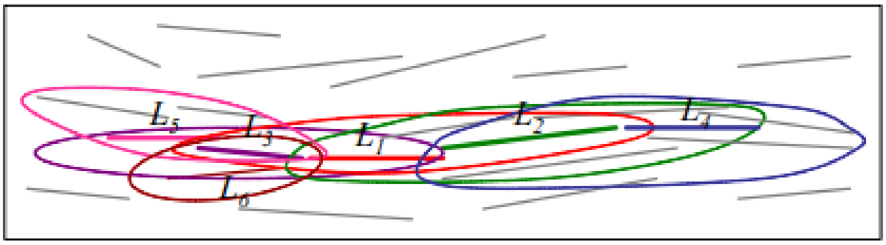


Figure 5.5: Example of the DBSCAN algorithm on line segments.
Published in ACM SIGMOD Conference 2007
Jae-Gil Lee, Jiawei Han, K. Whang

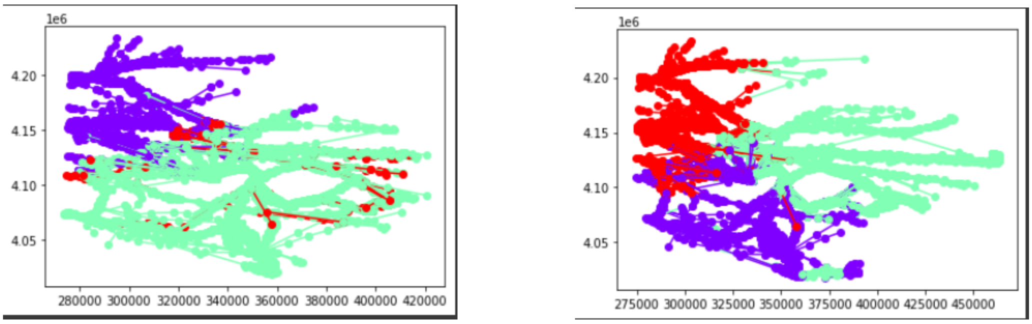
5.4.2 calculating the hyperparameters

As mentioned, the ϵ parameter is the distance threshold that is used to define the radius around each line. In order to find the optimal ϵ value, the entropy formula is used. Entropy is a measure of the disorder or randomness in a system, and it is used to quantify the amount of uncertainty or unpredictability in a set of data. The goal is to choose an ϵ value that minimises the entropy of the system. To achieve that, the entropy of the dataset is calculated for a range of ϵ values, and a value that results in the minimum entropy is used. It is important to note that the ϵ value is not the one that gives the least entropy, but rather a value around the region of least entropy that gives the most meaningful clusters. After calculating the optimal ϵ the average line neighbours of that ϵ value are calculated, and the MinLns value is chosen. Since the clusters that are to be created have to be meaningful, it is a logical assumption that MinLns > AvgNgb. In order for the MinLns chosen to be impactful, just like the ϵ value, it is chosen based on the clusters it creates. The clusters created are judged on their population and their cardinality. A healthy cluster must contain a healthy population of trajectories of different identifications; otherwise, it is deleted as not helpful. In this case, clusters with identification cardinality $(id) < 50$ were deleted. The final result is 3 clusters with identification cardinalities ≥ 230.

$$H(X) = \sum_{i=1}^{n} p(x_i) \log_2 \frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i),$$

$$\text{where } p(x_i) = \frac{|N_\varepsilon(x_i)|}{\sum_{j=1}^{n} |N_\varepsilon(x_j)|} \text{ and } n = num_{ln}$$

Figure 5.6: The entropy formula.
Published in ACM SIGMOD Conference 2007, Jae-Gil Lee, Jiawei Han, K. Whang



(a) Example of bad clustering.

(b) Example of good clustering

Figure 5.7: An intuitive example of good and bad quality clustering.

5.4.3 Representative Trajectories of the Clusters

Having acquired the clusters, the next step is to create their representative trajectories. With a set of vectors, the first step is to calculate the average direction vector, rotate the plane axes, and use a sweeping vertical algorithm that calculates the mean of the vectors it penetrates and their population. The average direction vector is given by the formula:

$$\vec{V} = \frac{\vec{u_1} + \vec{u_2} + \vec{u_3} + ... + \vec{u_n}}{|N_{u_n}|} \tag{5.2}$$

Having acquired the average vector, the X, Y axes are rotated until the average vector becomes parallel to the X axis. The angle ϕ of rotation between the average vector and the X axis can be extracted by the inner product of $\vec{V}$ and the unit vector $\vec{x}$ and the new coordinate system is given by the rotation matrix formula:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{5.3}$$

Having rotated the X, Y axes, the representations of all the vectors in the X', Y' are calculated in order to be used for the line sweep algorithm.

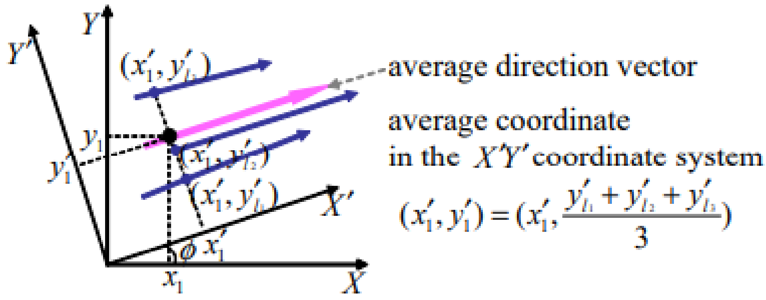
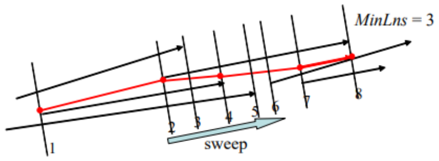


Figure 5.8: Example of the average vector $\vec{V}$ and the rotation of the axes.
Published in ACM SIGMOD Conference 2007, Jae-Gil Lee, Jiawei Han, K. Whang.

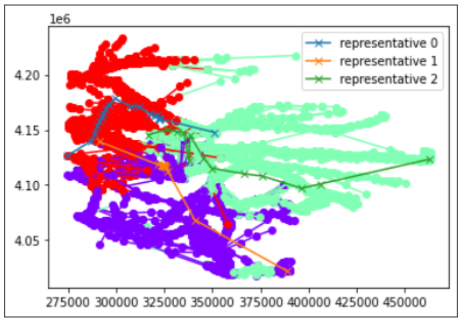
5.4.4 Applying the line sweep algorithm

The line sweep algorithm will be applied to the transformed vectors of the X', Y' plane. Since the plane was calculated in order for the X' axis to be parallel to the average vector, the line sweep will be applied with a line parallel to the Y' axis. For every step the algorithm moves the line in the X' axis, the population of the vectors it intercepts is counted; if the population is greater than a scalar value, the mean value Y' of the vectors and the algorithm's line X' value are assigned as a characteristic point. At the end, the trajectory created from the characteristic points is rotated back to normal, and the characteristic trajectory is extracted.



(a) Example of the sweep algorithm creating a characteristic trajectory.

Published in ACM SIGMOD Conference 2007, Jae-Gil Lee, Jiawei Han, K. Whang



(b) The representative trajectories of the clustered data.

Figure 5.9: Extracting representative trajectories from data with the line sweep algorithm.

5.5 Training the Neural Network model with clustered data

Up until now, the sequence of actions has been as follows. Firstly, the representative trajectories of the real data were created using the MDL principle and distance formula, then the hyperparameters ϵ and MinLns were calculated using the entropy formula discussed in 5.5..2, and a density based algorithm for line segments, Traclus, was applied. Lastly, having created the clusters, the representative trajectory of each cluster was calculated using a sweep line algorithm. To train the neural network with clustered data, every row of the dataset has to be clustered accordingly. This can be accomplished in numerous ways. For this study, the representative trajectory points were interpolated, and the data were clustered based on the Euclidean distance between the points of the dataset and the cluster representative trajectories; if it was needed, extrapolation instead of interpolation was used. The dataset rows were assigned to the closest cluster. This method can also be used when deploying the model for that region. Given the last 10 timestamps of a vessel, the vessel can be assigned to a cluster and the corresponding model can give the prediction. The training process was the same, as mentioned in Chapter 5.3. The results will be shown in Chapter 6.

Chapter 6

Results

Considering the results of the unclustered data, a difference in accuracy can be observed when the model is trained with a cost function of MAE compared to MSE. In general, the model trained with the MAE cost function yielded better results by up to one hundred metres. In the cases where the MAE function underperformed, it can be explained by the lack of instances. In regards to the clustered data, better results are clearly observed. Regarding the choice between a model with MAE cost function compared to MSE, there is no significant difference in results, but it should be noted that in training, the MAE model weights were adjusted one to five epochs faster compared to the MSE model, making it a better overall choice. Lastly, By utilizing the Douglas-Peucker algorithm, we partition larger trajectories with large time windows into smaller trajectories and seamlessly integrate them into the dataset. This technique allows us to uncover valuable insights from previously unoccupied time windows without compromising the reliability of the data. The results unequivocally demonstrate that adopting this algorithm has no detrimental impact on the model's predictive capabilities.

a) In tables 6.1. and 6.2 the R^2 values where calculated for both the latitude and longitude values in the test data, the high R^2 values mean that the model fits the data well and the independent variables are good predictors of the dependent variables in all the cases examined. In all cases the Easting error is greater than the Northing error, this is primarily caused by the trajectories aligning parallel to the x-axis as shown in figure 5.9.(b).

b) The table 6.3 contains the results of the model trained with unclustered data. It is the basis upon which we will compare the results of the models trained with clustered data. It is observed that for smaller time windows the model performs with higher accuracy, that can be explained by the population of observations in the smaller time windows and the physical limitations of travelling speed. This pattern is observed on all tables and is to be expected.

c) The tables 6.4, 6.5, and 6.6 contain the results of the models trained with cluster 1, cluster 2 and, cluster 3 respectively. In all cases,the results show smaller errors compared to table 6.3 by a great margin. This shows that clustering improves accuracy.

d) The tables 6.7, 6.8, and 6.9 contain the results of the models trained with clustered data after utilizing the Douglas-Peucker algorithm. The algorithm has no detrimental impact on the model's predictive capabilities, as no major difference can be observed.

Table 6.1: Easting and Northing R2, MAEs, and MSEs values from the model trained with MAE cost function.

	unclustered	cluster 1	cluster 2	cluster 3
Test Northing R2:	0.99	0.99	0.99	0.99
Test Easting R2:	0.99	0.99	0.99	0.99
Test Northing $\sqrt{MSE}$:	684.76	791.68	723.67	854.49
Test Easting $\sqrt{MSE}$:	924.72	1,147.73	745.32	901.47
Test Northing MAE:	428.32	583.12	490.99	619.84
Test Easting MAE:	589.35	744.80	574.98	673.04

Table 6.2: Easting and Northing R2, MAEs, and MSEs values from the model trained with the MSE cost function.

	unclustered	cluster 1	cluster 2	cluster 3
Test Northing R2:	0.99	0.99	0.99	0.99
Test Easting R2:	0.99	0.99	0.99	0.99
Test Northing $\sqrt{MSE}$:	689.43	603.07	724.53	867.10
Test Easting $\sqrt{MSE}$:	932.49	1,533.25	1,314.51	919.93
Test northing MAE:	442.68	371.56	487.69	654.31
Test easting MAE:	625.27	419.35	552.49	717.60

Table 6.3: Distances of the unclustered data with MAE and MSE cost function.

The mean euclidean distance between predicted and real positions of the unclustered data in time windows of 1 minute for the model trained with the cost function MAE compared to MSE.

Time window in minutes	distance in km (MAE)	distance in km (MSE)	number of observations
[1,2)	0.48	0.52	159
[2,3)	0.76	0.78	15024
[3,4)	1.14	1.17	5535
[4,5)	0.66	0.70	228
[5,6)	1.26	1.28	96
[6,7)	1.32	1.35	48
[7,8)	1.90	1.95	42
[8,9)	1.22	1.25	59
[9,10)	1.46	1.57	45
[10,11)	1.52	1.55	42
[11,12)	1.31	1.24	39
[12,13)	1.30	1.39	35
[13,14)	1.18	1.21	27
[14,15)	1.42	1.51	41
[15,16)	1.56	1.60	18
[16,17)	2.42	2.43	24
[17,18)	2.53	2.52	16
[18,19)	1.09	1.09	9
[19,20)	1.12	1.12	11
[20,21)	2.61	3.09	9
[21,22)	3.47	3.48	12
[22,23)	4.31	4.34	11
[23,24)	1.92	1.88	14
[24,25)	1.66	1.62	8
[25,26)	1.40	1.54	5
[26,27)	1.24	1.24	9
[27,28)	2.15	1.64	6
[28,29)	3.67	3.52	7
[29,30]	3.40	3.46	6

Table 6.4: Distances of the cluster 1 data with MAE and MSE cost function.

The mean euclidean distance between predicted and real positions of the cluster 1 data in time windows of 1 minute for the model trained with the cost function MAE compared to MSE.

Time window in minutes	distance in km (MAE)	distance in km (MSE)	number of observations
[1,2)	0.11	0.11	19
[2,3)	0.13	0.13	761
[3,4)	0.19	0.19	317
[4,5)	0.25	0.25	13
[5,6)	0.32	0.32	8
[6,7)	0.38	0.38	4
[7,8)	0.44	0.44	4
[8,9)	0.51	0.51	3
[9,10)	0.57	0.57	6
[10,11)	0.64	0.64	5
[11,12)	0.69	0.69	1
[12,13)	0.73	0.73	3
[13,14)	0.81	0.81	3
[14,15)	0.85	0.85	2
[15,16)	0.94	0.94	1
[16,17)	0.99	0.99	1
[17,18)	Nan	Nan	Nan
[18,19)	1.10	1.10	1
[19,20)	Nan	Nan	Nan
[20,21)	Nan	Nan	Nan
[21,22)	1.27	1.27	3
[22,23)	Nan	Nan	Nan
[23,24)	1.43	1.43	1
[24,25)	1.46	1.46	1
[25,26)	Nan	Nan	Nan
[26,27)	Nan	Nan	Nan
[27,28)	Nan	Nan	Nan
[28,29)	1.73	1.73	1
[29,30]	Nan	Nan	Nan

Table 6.5: Distances of the cluster 2 data with MAE and MSE cost function.

The mean euclidean distance between predicted and real positions of the cluster 2 data in time windows of 1 minute for the model trained with the cost function MAE compared to MSE.

Time window in minutes	distance in km (MAE)	distance in km (MSE)	number of observations
[1,2)	0.11	0.11	9
[2,3)	0.13	0.13	2732
[3,4)	0.19	0.19	1098
[4,5)	0.25	0.25	15
[5,6)	0.32	0.32	7
[6,7)	0.40	0.40	4
[7,8)	0.46	0.46	5
[8,9)	0.52	0.52	19
[9,10)	0.57	0.57	7
[10,11)	0.62	0.62	12
[11,12)	0.69	0.69	5
[12,13)	0.74	0.74	2
[13,14)	0.81	0.81	2
[14,15)	0.86	0.86	5
[15,16)	0.93	0.93	3
[16,17)	0.98	0.98	3
[17,18)	1.04	1.04	2
[18,19)	Nan	Nan	Nan
[19,20)	1.18	1.18	1
[20,21)	Nan	Nan	Nan
[21,22)	1.30	1.30	2
[22,23)	Nan	Nan	Nan
[23,24)	1.39	1.39	2
[24,25)	1.45	1.45	1
[25,26)	1.54	1.54	1
[26,27)	1.60	1.60	2
[27,28)	1.65	1.65	2
[28,29)	1.68	1.68	1
[29,30]	Nan	Nan	Nan

Table 6.6: Distances of the cluster 3 data with MAE and MSE cost function.

The mean euclidean distance between predicted and real positions of the cluster 3 data in time windows of 1 minute for the model trained with the cost function MAE compared to MSE.

Time window in minutes	distance in km (MAE)	distance in km (MSE)	number of observations
[1,2)	0.10	0.10	42
[2,3)	0.13	0.13	3473
[3,4)	0.19	0.19	1255
[4,5)	0.26	0.26	61
[5,6)	0.32	0.32	25
[6,7)	0.38	0.38	14
[7,8)	0.44	0.44	11
[8,9)	0.50	0.50	13
[9,10)	0.56	0.56	15
[10,11)	0.62	0.62	5
[11,12)	0.69	0.69	15
[12,13)	0.74	0.74	8
[13,14)	0.80	0.80	6
[14,15)	0.86	0.86	12
[15,16)	0.92	0.92	2
[16,17)	0.97	0.97	5
[17,18)	1.04	1.04	6
[18,19)	1.10	1.10	3
[19,20)	1.17	1.17	2
[20,21)	1.22	1.22	5
[21,22)	1.27	1.27	2
[22,23)	1.35	1.35	8
[23,24)	1.40	1.40	3
[24,25)	1.46	1.46	2
[25,26)	1.53	1.53	4
[26,27)	1.58	1.58	3
[27,28)	1.65	1.65	1
[28,29)	1.71	1.71	3
[29,30]	1.75	1.75	2

Table 6.7: Distances of the cluster 1 data after applying the Douglas-Peucker algorithm.

The mean euclidean distance between predicted and real positions of the cluster 1 data in time windows of 1 minute for the model trained with the cost function MAE after applying the Douglas-Peucker algorithm.

Time window in minutes	distance in km (MAE)	number of observations
[1,2)	0.10	556
[2,3)	0.13	25098
[3,4)	0.19	12434
[4,5)	0.26	693
[5,6)	0.32	400
[6,7)	0.38	163
[7,8)	0.45	155
[8,9)	0.50	160
[9,10)	0.57	150
[10,11)	0.63	168
[11,12)	0.68	81
[12,13)	0.74	140
[13,14)	0.81	99
[14,15)	0.86	160
[15,16)	0.92	81
[16,17)	0.99	102
[17,18)	1.04	36
[18,19)	1.10	36
[19,20)	1.16	54
[20,21)	1.20	9
[21,22)	1.28	63
[22,23)	1.34	9
[23,24)	1.40	45
[24,25)	1.46	27
[25,26)	Nan	Nan
[26,27)	1.60	9
[27,28)	1.64	9
[28,29)	1.73	9
[29,30)	Nan	Nan

Table 6.8: Distances of the cluster 2 data after applying the Douglas-Peucker algorithm.

The mean euclidean distance between predicted and real positions of the cluster 2 data in time windows of 1 minute for the model trained with the cost function MAE after applying the Douglas- Peucker partitioning algorithm.

Time window in minutes	distance in km (MAE)	number of observations
[1,2)	0.11	1046
[2,3)	0.13	15330
[3,4)	0.19	9391
[4,5)	0.25	201
[5,6)	0.32	81
[6,7)	0.40	45
[7,8)	0.46	54
[8,9)	0.52	217
[9,10)	0.56	99
[10,11)	0.62	113
[11,12)	0.68	117
[12,13)	0.74	58
[13,14)	0.80	50
[14,15)	0.86	47
[15,16)	0.93	32
[16,17)	0.98	41
[17,18)	1.04	36
[18,19)	1.13	9
[19,20)	1.17	21
[20,21)	Nan	Nan
[21,22)	1.28	27
[22,23)	1.37	9
[23,24)	1.42	15
[24,25)	1.45	12
[25,26)	1.54	9
[26,27)	1.60	16
[27,28)	1.65	27
[28,29)	1.68	5
[29,30)	Nan	Nan

Table 6.9: Distances of the cluster 3 data after applying the Douglas-Peucker algorithm.

The mean euclidean distance between predicted and real positions of the cluster 3 data in time windows of 1 minute for the model trained with the cost function MAE after applying the Douglas- Peucker partitioning algorithm.

Time window in minutes	distance in km (MAE)	number of observations
[0,2)	0.11	1419
[2,3)	0.13	20524
[3,4)	0.19	10463
[4,5)	0.26	518
[5,6)	0.32	206
[6,7)	0.39	166
[7,8)	0.45	96
[8,9)	0.50	117
[9,10)	0.56	134
[10,11)	0.61	54
[11,12)	0.69	126
[12,13)	0.74	95
[13,14)	0.80	77
[14,15)	0.86	113
[15,16)	0.93	38
[16,17)	0.97	55
[17,18)	1.04	46
[18,19)	1.11	33
[19,20)	1.17	18
[20,21)	1.22	70
[21,22)	1.27	15
[22,23)	1.35	71
[23,24)	1.40	45
[24,25)	1.46	21
[25,26)	1.53	36
[26,27)	1.58	50
[27,28)	1.64	18
[28,29)	1.72	45
[29,30)	1.77	25

Chapter 7

Conclusion

This research study delved into the problem of predicting future locations in maritime transportation. It aimed to assess the effectiveness of two approaches: using unclustered data for Neural Networks and applying trajectory clustering algorithms to the data prior to training. The findings indicate that both methods yield accurate predictions, and employing clustered data enhances the models' accuracy even further. The improvement achieved through clustering is attributed to reducing the data's dimensionality while enabling the neural network to extract valuable insights from the data. Consequently, this study demonstrates the feasibility of enhancing the trajectory prediction of neural networks in maritime transportation by clustering trajectories. However, it is important to acknowledge that trajectory clustering necessitates additional preprocessing steps that are time-consuming and computationally demanding. To advance this field, future research can explore alternative distance formulas or modifications to the existing formula. Additionally, incorporating weather data, such as temperature and wind speed, holds potential for further enhancing prediction outcomes. Furthermore, exploring the application of different types of Neural Networks, such as Long Short-Term Memory, Convolutional Neural Networks, and Ensemble Networks, could offer valuable insights. To summarize, this study establishes the potential of trajectory clustering and Neural Networks for location prediction and emphasizes the need for continued research to identify optimal approaches for predicting maritime vessels' locations.

Glossary

AI artificial intelligence. 8

AIS Automatic Identification System. 15

Big Data extremely large datasets. 15

DBSCAN Density Based Spatial Clustering of Applications with Noise. 4

ELU Exponential Linear Unit. 9

GIS Geographic Information System. 15

LSTM Long Short-Term Memory. 4

MAE Mean Absolute Error. 13

ML Machine learning. 8

MLPs MultiLayer Perceptron. 15

MMs Markov Models. 5

MSE Mean Squared Error. 13

NNs Neural Networks. 8

ReLU Rectified Linear Unit. 9

Tanh Hyperbolic Tangent. 9

TP Trajectory Prediction. 3

Traj-clusiVAT a scalable clustering and Markov chain-based hybrid framework. 4

WGS84 World Geodetic System 1984. 18

Bibliography

[1] Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, and James C. Bezdek. A scalable framework for trajectory prediction. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3860–3874, 2019.

[2] Alexandros TroupiotisKapeliaris, Nicolas Zygouras, Manolis Kaliorakis, Spiros Mouzakitis, Giannis Tsapelas, Alexander Artikis, Eva Chondrodima, Yannis Theodoridis, and Dimitris Zissis. volume 1, pages 744–751. 1st edition, 2022.

[3] Eva Chondrodima, Nikos Pelekis, Aggelos Pikrakis, and Yannis Theodoridis. An efficient lstm neural network-based framework for vessel location forecasting. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–17, 2023.

[4] Panagiotis Tampakis, Eva Chondrodima, Aggelos Pikrakis, Yannis Theodoridis, Kostis Pristouris, Harry Nakos, Eleni Petra, Theodore Dalamagas, Andreas Kandiros, Georgios Markakis, Irida Maina, and Stefanos Kavadas. Sea area monitoring and analysis of fishing vessels activity: The i4sea big data platform. In *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, pages 275–280, 2020.

[5] Eva Chondrodima, Petros Mandalis, Nikos Pelekis, and Yannis Theodoridis. Machine learning models for vessel route forecasting: An experimental comparison. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, pages 262–269, 2022.

[6] Petros Mandalis, Eva Chondrodima, Yannis Kontoulis, Nikos Pelekis, and Yannis Theodoridis. Machine learning models for vessel traffic flow forecasting: An experimental comparison. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, pages 431–436, 2022.

[7] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. SIGMOD '07, page 593–604, New York, NY, USA, 2007. Association for Computing Machinery.

[8] Angelos Valsamis, Konstantinos Tserpes, Dimitrios Zissis, Dimosthenis Anagnostopoulos, and Theodora Varvarigou. Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction. *Journal of Systems and Software*, 127:249–257, 2017.

[9] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

[10] Yuan Wang, Dongxiang Zhang, Ying Liu, and Kian-Lee Tan. Trajectory forecasting with neural networks: An empirical evaluation and a new hybrid model. *IEEE Transactions on Intelligent Transportation Systems*, 21(10):4400–4409, 2020.