Faculty of Engineering and
Informatics

UNIVERSITY of BRADFORD

Vehicle State Predictors for Predicting the Kinematic State of Vehicles

Author: Oliver David Coates

Student Number: 19004918

Project Supervisor: Dr. Marco Profili, Dr. Viktor Doychinov

Project Shadow: Prof. Fun Hu, Dr. Viktor Doychinov

Course: MSc Satellite Systems Engineering

Submitted in partial fulfilment of the requirements for the degree of

MSc Satellite System Engineering

From the

University of Bradford

Table of Contents

List of Figures

Section 1

Section 2

Section 3

Section 4

List of Tables

Section 3

Section 4

List of Acronyms

AI - Artificial Intelligence.
AIS - Automatic Identification System
CNN - Convolutional Neural Network
CSV - Comma Separated Values
DoP - Depth of Precision
EKF - Extended Kalman Filter
GGA - Global Positioning System Fix Data
GLL - Geographic Position, Latitude and Longitude (and time)
GGBG - GPS, Galileo, BEIDOU & GLONASS
GNSS - Global Navigation Satellite System
GPS - Global Positioning System
IMU - Inertial Measurement Unit
LSRM - Long Short Term Memory
MAE - Mean Absolute Error
ML - Machine Learning
MMSI - Maritime Mobile Service Identity
MSE - Mean Squared Error
NN - Neural Network
RMSE - Root Mean Squared Error
SOG - Speed Over Ground
UID - Unique Identifier
UTC - Coordinated Universal Time
VHF - Very High Frequency

List of Equations

0. Abstract

This thesis aims to address questions related to predicting the kinematic state of vehicles using satellite products. This study investigates the feasibility of predicting a vehicle's kinematic trajectory from satellite products and satellite-enabled data then assessing the results of various models. The study of traditional statistical approaches compared to machine learning methods was conducted, comparing the performance of traditional methods such as linear regression and random forest regression to traditional neural networks.

A preliminary study was also conducted comparing the usefulness of various GNSS constellations, accessing accuracy and precision of each GNSS constellation.

In order to fulfil the objectives of this thesis publicly available data was taken from reputable and official data sources including the European Commision and the French Naval Academy. The availability of maritime data meant that the implementation phase of this study focused on maritime vehicles. This data is then appropriately preprocessed ready for the implementation of three prediction models: linear regression, random forest regressor and a traditional neural network.

Results were analysed based on various evaluation metrics, numerical analysis and residual graph interpretations. Findings indicated that there was varied performance and quality from each model. The random forest approach yielded results implying it was the most effective predictor model for maritime data based on analysis of evaluation metrics and the interpretation of residual graphs. By contrast, the neural network produced results showing it struggled to fit the data, particularly with the production of a negative R2-score.

The conclusion of this thesis demonstrates the usefulness of satellite products in conjunction with other data sources in order to demonstrate the validity and feasibility of performing trajectory predictions for the kinematic state of a vehicle. Providing insight into which models may be best appropriate as well as producing interpretations regarding how good a model may fit data.

1. Introduction

1.1. Overview

This thesis aims to produce an answer to the following problem: “Can we predict a vehicle´s movement using satellite products?” That is the core question of this thesis.

In order to get to the conclusion of this question, we need to understand multiple aspects related to the question we are trying to answer: Why do we need to predict a vehicle´s movement? What is the current state of research in vehicle state predictors? In the case of methodology, do we employ a traditional statistical approach or use an ever increasingly popular AI/Machine Learning approach? Or even adopt a hybrid approach for our final model? In terms of data, what kind of vehicle data is required to conduct an accurate prediction? In a broader sense, how does this research impact you or other people?

These questions are essentially to help pose an introduction, a mutual understanding and question to me and to you, the reader: why should we predict a vehicle´s movement from satellite products and how do we get there? Current similar research into autonomous driving [2] [3] [6], land based vehicle detection from satellite image products [4] [7], naval studies on AIS data [5] and other papers researching trajectory prediction [9] [10] [16] suggests that the possibility of predicting vehicle (even human) trajectories based on parameters associated to the vehicle, e.g. speed, longitude, latitude and other related parameters is certainly a realistic possibility.

Each aspect of research, no matter the subject, will almost certainly have pitfalls, challenges, no definitive answers, or numerous other setbacks, the thesis must be resilient when encountering any setback. With this in mind, in terms of this thesis, some of the challenges I anticipate this study will encounter, due to its nature, are:

- Data gathering and processing - collecting relevant data from vehicles may be time-consuming to collect and may be irrelevant or challenging to procure from other external sources.
- Alternate data sources stemming from data unavailability.
- Data procurement times - especially for original data.
- Data pre-processing - data taken from external sources may require pre-processing in order to be compatible or even usable with later prediction models.
- Model implementation and optimisation.
- Prediction and overall model quality.
- Producing a model with a good fit to data.
- Data overfitting and underfitting.

Finally, considering the above questions and recognising the potential challenges associated with the project of this thesis, we can now turn our attention to the background associated with the theme of this thesis, providing some additional context to our questions.

1.2. Background

Using satellite products primarily for vehicle state prediction is nothing new in terms of research, however, with the most viewed papers on ResearchGate, Google Scholar, IEEEXplore and many

other related research websites and publishers relating vehicle state prediction and it's related research/papers/journals to autonomous vehicles [2][3], a key research area in the field of vehicle state prediction. Satellite products in many papers, as well as those papers covered in this thesis, are used as a complement or addition to existing data from things such as automatic identification systems (AIS), inertial measurement units (IMU) and ego-vehicle (the vehicle containing the sensors) sensor data for both land and maritime based vehicles, satellite products are rarely the main source of data in many cases.

However, most products utilising vehicle state prediction in autonomous vehicles [3] and AIS-utilising maritime vessels (i.e. maritime vehicles such as boats and ships) [4] will utilise satellite products in their operations (e.g. navigational information from GNSS) in conjunction with other navigational data parameters such as speed and bearing. Meaning we could also use them for our own prediction purposes, as well as taking into account other information a vehicle may transmit.

In terms of why we would potentially want to utilise satellite images for vehicle state prediction purposes, a number of scenarios and problems could be solved by answering the original question (or parts of the original question) of this thesis.

For instance, consider a practical scenario where we need to implement satellite products to solve a problem and produce some kind of output. In this case, we could look at a maritime related problem: there may be a busy trade route that is frequented by a large amount of maritime vessels, this is a huge logistical and commercial operation, if a sudden or unexpected event were to happen to disrupt normal operations - such as a vessel deviating from a trade route due to adverse/extreme weather, terrorism or mutiny, the case where we would be able to predict where that vehicle is going to a reasonable degree using a prediction model would provide ample time in search and rescue operations and potentially save a large amount of money and lives.

Additionally, for a military related scenario, having access to satellite products (images, data) which is able to be used in models for the identification of targets and the prediction of the trajectory of said targets would provide a military operation ample time to prepare and execute strategy in relation to the accuracy and timing of the prediction.

Potentially going outside of the scope of this project, some psychological or human factors could also play a role in the trajectory or route a vehicle takes [9]. For land based vehicles with a single controller (i.e. a driver) the human factor plays a larger role in decisions compared to a maritime vessel which is more likely to follow a structured route and can require multiple people to control and manage the vessel. In relation to this, there already exists research which covers both short and long term trajectory predictions for humans using both model and model-free based approaches [10].

In summary, in order to predict the kinematic state of any vehicle, we require knowledge of certain data parameters associated with the vehicle, such as its current location, speed, bearing or even environmental factors before designing, developing, implementing and evaluating a prediction model. This will be covered comprehensively in the later sections of this thesis.

1.3. Project Aims

As mentioned in the overview, the aim of the thesis is to attempt to answer the original question: "Can we predict a vehicle´s movement using satellite products?". In order to come to a conclusion,

partial conclusion or even a failure to conclude that question, we must set, research and attempt a set of objectives or goals that this thesis must accomplish.

Firstly, the aim of this thesis is to make a model using statistical or machine learning methods for predicting the kinematic state of a vehicle using either collected or pre-existing data to a reasonable accuracy. The prime objectives are as follows:

- Objective 1. Complete a comprehensive literature review of vehicle state prediction, their usage, the current state of state prediction research and its usage in accordance with satellite products. Make a conclusion about the current state of research, how current and past research can aid in this project as well as how this thesis could potentially aid in current research.
- Objective 2: Collect appropriate data, either original or pre-existing, to support the development of the model.
- Objective 3: Design and implement a model based on collected data parameters which can predict a vehicle´s kinematic state using either statistical methods or machine learning. The model must have reasonable performance metrics when compared to other similar models.
- Objective 4: Test and evaluate the aforementioned model's performance metrics such as accuracy, precision, etc… This could be done by comparing the metrics of other models to your own.
- Objective 5: Conclude the thesis by presenting findings, detail final model performance metrics and make recommendations for future work.

1.4. Project Plan

The project timeline has been broken down into a milestone combined with a deadline, this plan can be seen as follows with an accompanying timeline, this can be used as a reference for any similar future projects:

Milestone	Deadline
Finish Literature Review	**30th June 2023**
Finish Introduction Section	**7th July 2023**
Find Satellite Data Products	**17th July 2023**
Create Initial Models and Data with Preprocessing	**30th July 2023**
Interim Review of Document	**30th July 2023**
Finish Model Evaluation and Complete First Draft	**14th August 2023**
PROJECT DEADLINE	**28th AUGUST 2023**

Figure 1.0 - Project Milestone Table

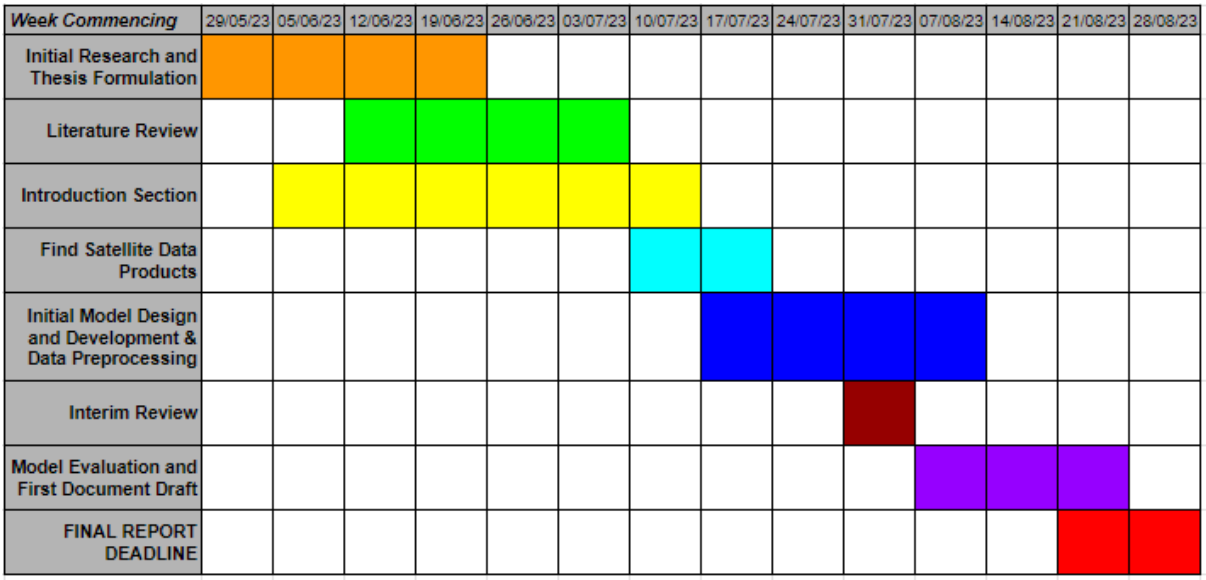


Figure 1.1 - Project Plan Diagram, GANTT Format

2. Literature Review

2.1. Brief Introduction

This literature review takes the style of looking at similar literature to the theme of this thesis. The methodology each piece of literature used will be reviewed followed by an evaluation of the findings produced by the paper, as well as a review into the methodology.

2.2. Similar Papers & Studies - A Literature Review

The first individual paper I would like to comprehensively examine is what I believe is the most relevant to the topic of this thesis. A master's thesis conducted by a student of the Naval Postgraduate School of the U.S. Armed Forces on the prediction of maritime vessels using a mixture of statistical and machine learning methods for predicting routes [5].

The results of this study found that it is possible to develop tools to predict vessel trajectories to the expected levels of a user using a random forest model to a 94% accuracy for future position prediction. A neural network was also used but achieved an accuracy of only 81%. The study also used a heatmap for the prediction region which is noted could be used in search and rescue operations, see Figure 2.0.

Accuracies also changed depending on how far a prediction was made - with predictions further into the future leading to less accurate readings.

It is also worth noting though that AIS was not originally designed to track ships. However, the information it provides certainly can provide enough information in order to conduct this task.

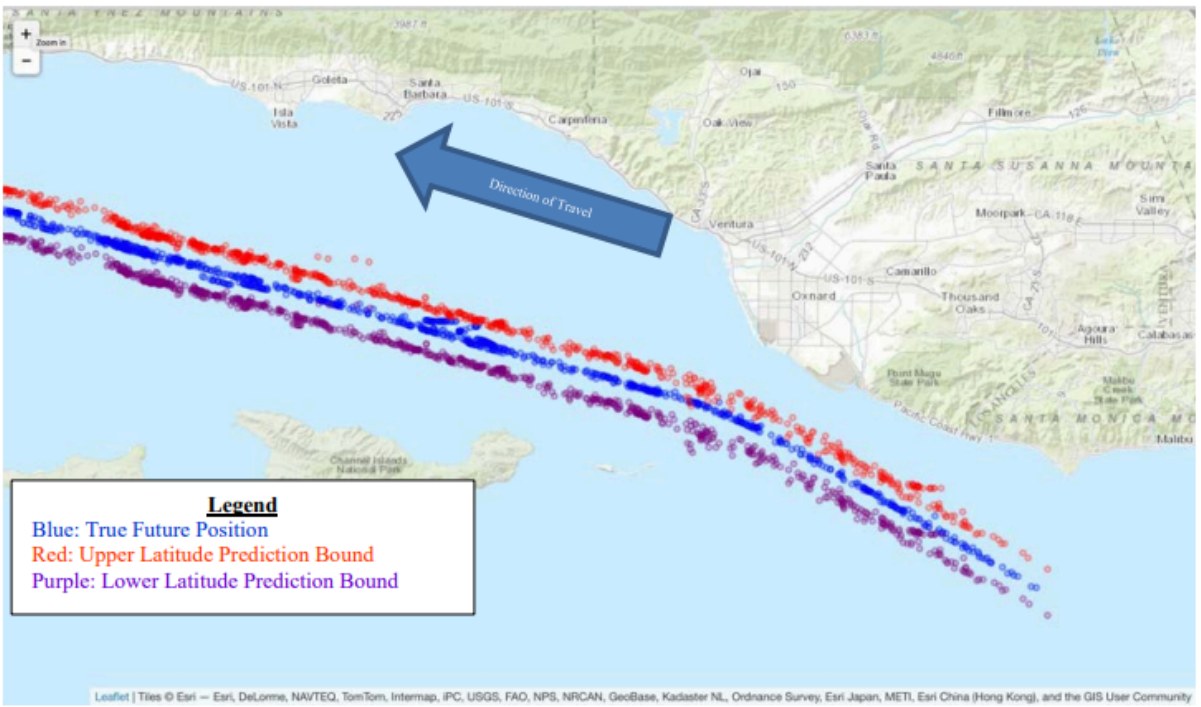


Figure 2.0 - Extract From Young's Paper Showing a Vessel's Predicted Route to Two Hours using a Random Forest Model

It's worth noting that using satellite products for maritime vehicle state prediction could be considered the most forefront application of VSPs for vehicles, as it's harder for cars/bikes/military vehicles and even harder still (if not improbable) for airborne vehicles to be predicted using satellite image products. It could also be used in the monitoring of “dark ships”, that is, ships that turn off their transponders and perform abnormal behaviours, you could use a combination of satellite products to detect, track and predict ship trajectories. [11]

Looking at trajectory prediction from another perspective, in another paper, Zhang, et al, 2017 [16], looked in their paper at using various machine learning methods for trajectory prediction in relation to computer vision. Zhang, et al, used an initial feature extractor to extract information from image and video data using a simple CNN (convolutional neural network) before moving onto a trajectory predictor model utilising LSTM (long short term memory).

The model was then used on both a real world vehicle dataset and a visual tracking benchmark used in the evaluation of prediction ability and on “tracking methods for tracking videos without pretrained trajectories” with the purpose of experimentation. Their tracking method is then compared against other tracking methods in the experiment: Struck, ASLA, CXT, CSK, SCM and DLT. Note that some of these methods are “classical” while others, such as DLT, are machine learning oriented.

When compared against other models in the experiment, success rate was noted as being 0.7436, which was comparable to SCM. However, the precision rate for their approach was 88.94%, outperforming the closest model (ASLA) by 3.9%. After the initial experiment, the average success rate on both datasets was analysed. The vehicle dataset yielded a result of 94%, which was the best result, more than ASLA and CSK by a margin of 3%. The benchmark dataset yielded a success rate of 68%, again, this was the best result, beating Struck by 2%.

However, Zhang, et al´s model was implemented using unoptimised code in MATLAB, further improvements to the model could have been made had the code been optimised, providing better results for the model.

The next paper I would like to examine is Hsu & Chen´s [3] paper on vehicle state estimation and prediction using state observers, in this paper, a mathematical approach is used in vehicle state estimation & prediction, with more emphasis on the vehicle itself.

Hsu & Chen mention in their paper a neglectance of current vehicle state estimation systems, for example, those used in vehicle control systems, through the increased usage of simplified models. As such, they decided to use a mathematical approach to tackle this problem.

Hsu & Chen´s model is based upon a vehicle model utilising two sets of Euler angles and four coordinate systems, this is used to describe multiple variables associated with the vehicle such as “vehicle yaw angle”, “vehicle pitch angle”, and “vehicle roll angle”. The paper creates novel mathematical models for a vehicle´s spring mass system, unsprung mass system, and road angles. The sprung mass system describes “6 DOFs” in relation to the body of the vehicle, the unsprung mass system describes subsystems of the vehicle: suspension, tires and steering.

The actual estimation and prediction models for the vehicles are derived using conventional EKF (Extended Kalman Filter).

The final accuracies for the state estimation and prediction model utilising the Extended Kalman Filter were 2.66% and 2.86%, respectively for state estimation and then for state prediction.

To continue on, I would now like to review two papers which correspond more with the data gathering of this project as well as AIS data itself: “Heterogeneous integrated dataset for Maritime Intelligence, surveillance, and reconnaissance” and “Maritime route and vessel tracklet dataset for vessel-to-route association” by Cyril Ray, Et. Al. and Clément Iphar, Et. Al. respectively, are two similar papers that utilise AIS data, particularly for route association tasks and data visualisation.

The paper by Cyril Ray, Et Al. specifically focuses on the collection of vessel data, even making direct mention to the kinematic state of a vessel, as well as showing mentions to other complementary data, such as that to do with the environment or specific route. [14]

The data itself is taken from October 1st, 2015 to March 31st, 2016 and covers the Celtic sea, the North Atlantic Ocean, the English Channel and the French Bay of Biscay. Overall, there are 34 different datasets included within the data file, with varying degrees of relevance to this project. There are also varying degrees of data size, with some datasets going into the range of GigaBytes. The data itself was collected from various maritime agencies including: The European Commission - Fisheries & Maritime Affairs, The French Naval Academy, SeaDataNet and open source data from data.gouv.fr, to count a few sources, all of which are official.

Useful and relevant data from within the datasets themselves includes:

- Maritime Mobile Service Identity (MMSI) data.
- Coordinates (Longitude & Latitude).
- Speed Over Ground (SOG) in knots.
- The true heading in degrees.
- Course Over Ground (COG).
- Degrees (direction of motion).
- Navigational status.

Secondly, the paper by Clément Iphar, et al. uses the above data to make route association for vessels and vice versa by extracting meaningful vessel data and using it to implement an algorithm which is able to predict which route the vessel is taking and vice versa. [15]

2.3. Evaluation of Methodology & The State of Research

Young, 2017, concluded using random forest classifiers worked for his model over the more advanced neural network classifier. Young´s random forest model yielded various accuracy results for a two hour prediction of a route, with average accuracy being at 94.24%, while the neural network under the same conditions produced an average accuracy of 81.19%,increasing the initial amount of epochs from the neural network from 1000 to 10000 did not significantly increase Young´s accuracy findings.

The findings of this paper could indicate that using traditional methods, such as random forest, could be preferable to machine learning methods, such as a traditional neural network.

Furthermore, Young also notes longer processing times for the neural network over the random forest model - however, it should be noted that Young ran his model from a regular macbook laptop

and not a workstation or supercomputer. Furthermore, the findings from Young´s study could aid in efficient search and rescue operations at sea and in logistics optimisation.

Zhang, et al, 2017, while their paper focused on computer vision for trajectory prediction, indicated from their findings that their method of using machine learning methods which utilised CNN and LSTM for performing predictions was also preferable over other methods.

The success rate of Zhang, et al´s model indicates that methods utilising a CNN/LSTM based approach could also be viable, if not preferred. However, this would only be relevant if the later model started to use image data, otherwise, it would not really be suited to purely numerical data such as AIS.

The usefulness of Ray, et al's data cannot be understated, with direct correspondence to what our model will need to collect and to a vessel's kinematic state, the data is highly useful and, therefore, must be considered for usage. The modified dataset for route-to-vessel prediction from Iphar, et al's paper should also be considered, as it could potentially have more preprocessing and useful variables. The datasets will be defined more comprehensively in section 3 of this paper to access the needs of the paper.

3. Data Collection and Preprocessing

3.1. Data Collection

In terms of data collection, the argument could be made either against producing original datasets or using existing datasets. Collecting the data ourselves will take more time/effort but will allow us to collect the exact data that we want to collect. Using existing datasets implies less effort taken to collect data with other organisations or individuals potentially having the means to gather and collect more data, there is a risk that data could be irrelevant though.

In terms of data we would like to collect, ideally, we could take data from various sensors on the vehicle. This could be from: GPS, IMU, AIS or camera data; we could also take vehicle data from non-sensors such as satellite images, weather data, traffic data and GNSS data. It is also worth noting that if we are predicting much more than just where the vehicle will be, we may have to predict vehicle position, velocity, acceleration and orientation to name a few data parameters we might want to take if that data is not already included in current sensor data, or in the case of making predictions far into the future, outside of potential timescale outliers.

Regardless of how we collect the data or what kind of data we would like to collect, it is necessary to collect the following data parameters at the minimum in order to make a prediction into a vehicle's kinematic state:

Table 3.0 - Useful Data Variables

Data Name	Data Description
Longitude	Longitudinal navigational information of the receiver/ego-vehicle.
Latitude	Latitudinal navigational information of the receiver/ego-vehicle.
Speed	The speed in which the receiver/ego-vehicle is moving in relation to when the data reading was taken. Measured in kilometres per hour.
Time	The calculated UTC time of the data reading with offset.
Orientation	The direction of the receiver/ego-vehicle.

Some additional data parameters could be taken such as journey start/end times or some additional parameters such as vehicle weight, max speed, environmental and traffic conditions. However, these parameters may not always be available and/or may serve to over-complicate certain models. Therefore, for the purposes of this project, it is necessary for the above data parameters to be taken as the minimum amount of required data parameters.

For the initial preliminary data analysis I gathered data using a U-Blox EVK-M101 GNSS receiver Using data from four different navigational satellite constellations: GPS, Galileo, BEIDOU and GLONASS which we will abbreviate as GGBG. The U-Blox EVK-M101 receiver is a GNSS

receiver using an active GNSS L1 band antenna. The kit is around 105 x 64 x 26 mm in terms of its dimensions. Data was configured to be collected through the U-Blox kit's accompanying software - UCenter2. GGBG distributes NMEA-0183 formatted messages containing our required data parameters, for initial data analysis, we use NMEA-0183 GGA & GLL formatted messages since this contains the appropriate data parameters [12]. This data will be used to analyse constellation accuracy and precision in order to determine which satellite navigational constellation will be used in the final model.

For the final model, I have taken the following approach: a maritime vessel approach where we will predict the trajectory of maritime vessels. Data was originally going to be available for land-usage as well, however, data was unfortunately made unavailable during early implementation.

For maritime route data, AIS data collected automatically from vessels operating in The Celtic Sea, North Atlantic Ocean, The English Channel and France's Bay of Biscay were used. The data was taken from the available public datasets from Ray et al's and Iphar et al's papers for AIS-enabled vessels [14] [15]. It is worth noting that most AIS systems will use "standard VHF transceivers" which enables the usage of GPS for precise positioning [13].

3.2. Preliminary Analysis of Collected Data

Before conducting analysis on the final model, an investigation into the usefulness and validity of GPS was conducted, with the aim of testing the accuracy and precision of the constellation and comparing that accuracy and precision over other available constellations.

Preliminary analysis was performed on originally collected UBlox GNSS receiver data to ascertain which navigation constellation our final model should use as well as to determine the usefulness of GPS. GNSS related parameters will be accessed and compared - particularly paying attention to the aforementioned constellation accuracy and precision. Data will also be visualised using relevant Python libraries, GeoJson data files of the University of Bradford campus and our specified data parameters such as longitude and latitude.

The receiver was first set-up to retrieve messages automatically using accompanying software included with the UBlox GNSS receiver, UCenter2. Pre-existing functions within the software were used to perform configurations, take readings and export the data as well as to specify data parameters (i.e. selected GNSS constellation, data columns, NAV enabling). The data was configured to be exported as a .csv file.

Data was gathered in 15 minute periods for each GNSS constellation using both internal and external GNSS receivers, making total collection time just over 2 hours. Hours reserved for data analysis were between 0800-1200 and 1300-1800. Data was taken during both these longer time periods to account for errors, setup time and redundancy. Weather conditions were harsher than normal during readings, which could have some effect on final metrics.

The received NMEA-0183 messages were first analysed in their raw format to determine the type of data used within the messages - the below figure shows an extract of the raw NMEA-0183 message data.

id	UBX-NAV-	UBX-NAV-	UBX-NAV-	UBX-NAV-	UBX-NAV-	UBX-NAV-	UBX-NAV-	UBX-NAV-	UBX-NAV-	UBX-NAV-PVT:sec
0	-1.8E+07	5.38E+08	209508	186	2022	11	14	10	40	40
1	-1.8E+07	5.38E+08	290569	4303	2022	11	14	14	19	31
2	-1.8E+07	5.38E+08	290628	4297	2022	11	14	14	19	35
3	-1.8E+07	5.38E+08	294319	3648	2022	11	14	14	33	2
4	-1.8E+07	5.38E+08	294297	3647	2022	11	14	14	33	6

Figure 3.0 - Ucenter2 Raw NMEA Data, as CSV.

Received data is highly numerical in nature, which aids in data pre-processing.

As mentioned above, we use Python to analyse this data, with Python we also utilise the following libraries:

Table 3.1 - Preliminary Data Analysis Libraries

Library Name	Library Description
Pandas	Used for data analysis and manipulation.
NumPy	Used for mathematical functions on arrays.
Seaborn	Used for data visualisation.
GeoPandas	Used for geospatial data functions.
Plotly Express	Used for further data visualisation.
MatPlotLib	Used for mathematical plotting.

After performing data exporting, we begin by importing the above libraries as well as the exported GGBG GNSS data we collected. Note that for the purposes of illustration, I will include code extracts below.

In [4]:
```
galileo_data = pd.read_csv("galileo_nmea_gagga.csv")
galileo_data_nc = galileo_data.rename(columns={"UBX-NAV-PVT:lon": "Longitude",
                                               "UBX-NAV-PVT:lat": "Latitude",
                                               "UBX-NAV-PVT:height": "Height",
                                               "UBX-NAV-PVT:pDOP": "Precision-DOP",
                                               "UBX-NAV-PVT:year": "Year",
                                               "UBX-NAV-PVT:month": "Month",
                                               "UBX-NAV-PVT:day": "Day",
                                               "UBX-NAV-PVT:hour": "Hour",
                                               "UBX-NAV-PVT:min": "Min",
                                               "UBX-NAV-PVT:sec": "Second",})
galileo_data_nc.head()
```

Out[4]:

	id	Longitude	Latitude	Height	Precision-DOP	Year	Month	Day	Hour	Min	Second
0	0	-17656063	537903942	209508	186	2022	11	14	10	40	40
1	1	-17653799	537902779	290569	4303	2022	11	14	14	19	31
2	2	-17653535	537902663	290628	4297	2022	11	14	14	19	35
3	3	-17650333	537900759	294319	3648	2022	11	14	14	33	2
4	4	-17650100	537900904	294297	3647	2022	11	14	14	33	6

Figure 3.1 - Importing of GNSS Constellation Galileo Data, Readability Dataset Modifications and Dataset Extract.

After the initial imports for each GNSS constellation and receiver data, we can perform feature extraction for the datasets, in this case we focus primarily on the longitude and latitude data, with the

addition of height and precision/DoP features. Longitude and latitude data is also stored as an integer datatype, this must be changed to a float for usage with the GeoPandas dataset as well as for later data analysis purposes.

```
In [6]: galileo_data_nc['Longitude'] = galileo_data_nc['Longitude'].astype(float)
        galileo_data_nc['Latitude'] = galileo_data_nc['Latitude'].astype(float)
        galileo_data_nc_lalo = galileo_data_nc.iloc[:, 0:5]
        galileo_data_nc_lalo.head()
```

Out[6]:

	id	Longitude	Latitude	Height	Precision-DOP
0	0	-17656063.0	537903942.0	209508	186
1	1	-17653799.0	537902779.0	290569	4303
2	2	-17653535.0	537902663.0	290628	4297
3	3	-17650333.0	537900759.0	294319	3648
4	4	-17650100.0	537900904.0	294297	3647

Figure 3.2 - Feature Extracted Galileo Dataset.

After performing necessary feature extraction, in order to use the longitudinal and latitudinal data for later analysis, further preprocessing is necessary, we are required to shape the data for later usage and analysis.

```
In [7]: galileo_data_nc_lalo['Longitude'] = galileo_data_nc_lalo['Longitude']/10000000
        galileo_data_nc_lalo['Latitude'] = galileo_data_nc_lalo['Latitude']/10000000

In [8]: galileo_data_nc_lalo.info
```

```
Out[8]: <bound method DataFrame.info of         id  Longitude   Latitude  Height  Preci
        sion-DOP
        0        0  -1.765606  53.790394  209508            186
        1        1  -1.765380  53.790278  290569           4303
        2        2  -1.765354  53.790266  290628           4297
        3        3  -1.765033  53.790076  294319           3648
        4        4  -1.765010  53.790090  294297           3647
        ...    ...        ...        ...     ...            ...
        2190  2190  -1.766808  53.790300  194877            309
        2191  2191  -1.766787  53.790304  194888            309
        2192  2192  -1.766762  53.790308  194837            309
        2193  2193  -1.766733  53.790313  194821            309
        2194  2194  -1.766708  53.790318  194489            309

        [2195 rows x 5 columns]>
```

Figure 3.3 - Data Shaping for Longitude and Latitude Galileo Data.

Note that the processing of import, feature extraction, data shaping, etc… was performed on each dataset for each GGBG GNSS constellation dataset - Figure 3.4 below shows a code extract of this pre-processing methodology for the GPS data, the same steps were followed for BeiDou and GLONASS as well.

```
In [10]: gps_data = pd.read_csv("gps_nmea_gagga.csv")
         gps_data_nc = gps_data.rename(columns={"UBX-NAV-PVT:lon": "Longitude",
                                                "UBX-NAV-PVT:lat": "Latitude",
                                                "UBX-NAV-PVT:height": "Height",
                                                "UBX-NAV-PVT:pDOP": "Precision-DOP",
                                                "UBX-NAV-PVT:year": "Year",
                                                "UBX-NAV-PVT:month": "Month",
                                                "UBX-NAV-PVT:day": "Day",
                                                "UBX-NAV-PVT:hour": "Hour",
                                                "UBX-NAV-PVT:min": "Min",
                                                "UBX-NAV-PVT:sec": "Second",})
         gps_data_nc['Longitude'] = gps_data_nc['Longitude'].astype(float)
         gps_data_nc['Latitude'] = gps_data_nc['Latitude'].astype(float)
         gps_data_nc_lalo = gps_data_nc.iloc[:, 0:5]
         gps_data_nc_lalo['Longitude'] = gps_data_nc_lalo['Longitude']/10000000
         gps_data_nc_lalo['Latitude'] = gps_data_nc_lalo['Latitude']/10000000
         gps_data_nc_lalo.info

Out[10]: <bound method DataFrame.info of         id  Longitude   Latitude  Height  Year
         0     2350  -1.765750  53.790303  213551  2022
         1     2351  -1.765749  53.790304  213560  2022
         2     2352  -1.765748  53.790304  213489  2022
         3     2353  -1.765748  53.790305  213504  2022
         4     2354  -1.765744  53.790307  213275  2022
         ...    ...        ...        ...     ...   ...
         1213  3563  -1.765731  53.790331  226129  2022
         1214  3564  -1.765728  53.790332  225941  2022
         1215  3565  -1.765726  53.790333  225806  2022
         1216  3566  -1.765725  53.790333  225785  2022
         1217  3567  -1.765723  53.790334  225621  2022

         [1218 rows x 5 columns]>
```

Figure 3.4 - GPS Data Pre-Processing Extract

After performing these initial steps, we can begin to analyse the data. Initially, we may want to understand how spread out the data is. Therefore, we analyse the standard deviation as well as complete a scatter graph of the latitude and longitude data (Figure 3.4, Table 3.5).

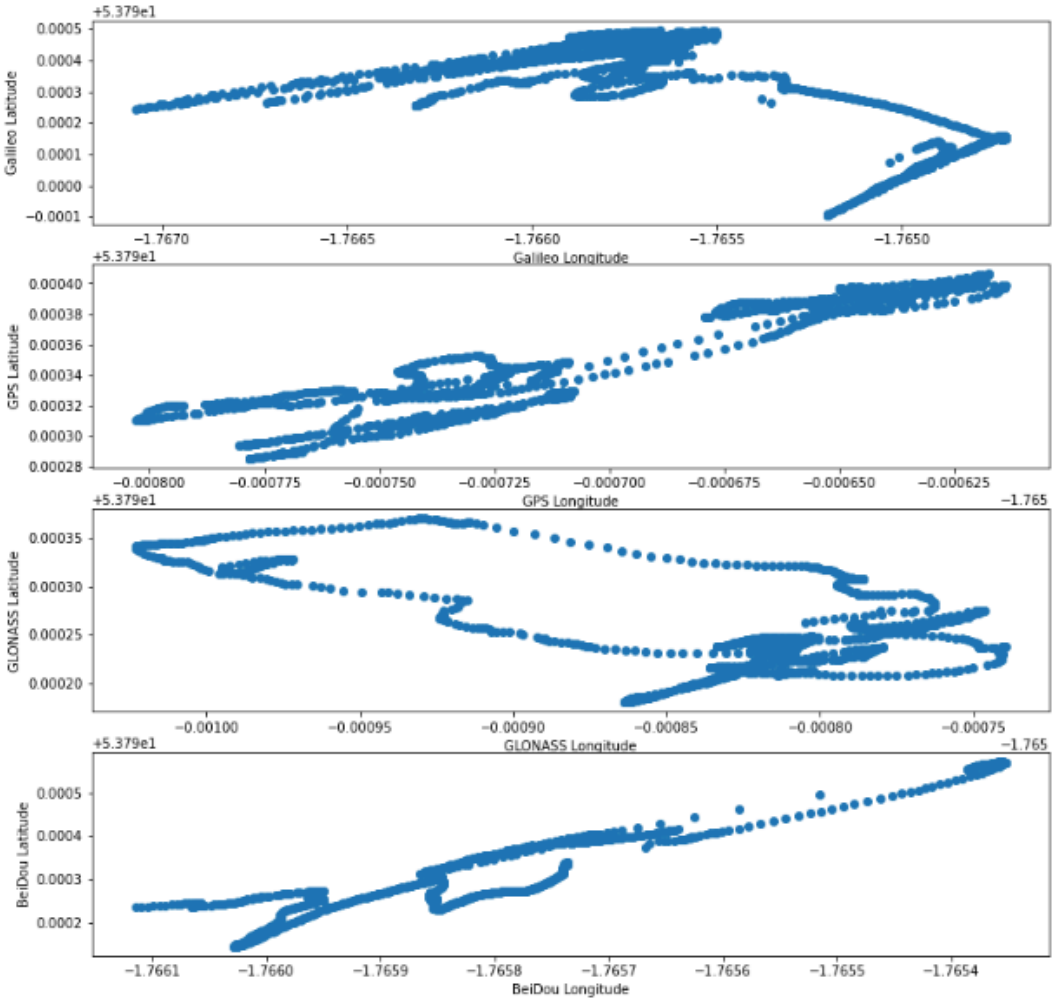


Figure 3.5 - Longitude/Latitude Scatter Graph

Table 3.2 - Longitude/Latitude Standard Deviation by GNSS Constellation

	GPS	Galileo	BEIDOU	GLONASS
Longitude σ	0.000054	0.000454	0.000152	0.000075
Latitude σ	0.000036	0.000143	0.000087	0.000047

The standard deviation was calculated using a mixture of the Pandas and Numpy library using the standard deviation formula (implemented with NumPy):

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

Eq. 1. Standard Deviation Formula

After this, we will perform further visualisation of the data, a few steps must be adhered to in order to achieve this:

- The data must be pre-processed to be in the correct format.
- The data must have a geometric medium to be projected onto.
- A medium in which we can project our data points onto our geometric medium.

To achieve the first of these goals, we use the GeoPandas library to change the current dataframe into one that can be projected onto a geometric medium, i.e. our map, using the GeoDataFrame data type.

The geometric medium must also be appropriate; a GeoJSON file format of the University of Bradford´s campus was used, taken from a GeoJSON file of the general West Yorkshire area. This can be used later on to make a geofence as well as giving us a medium in order to map our geometric data points.

To conclude the last of these goals, we use the Plotly Express library in order to create a visual representation of the University of Bradford campus using the original GeoJSON file as well as mapping our original geometric data points (i.e. our longitude and latitude values) onto the University of Bradford GeoJSON file.

The result is a traced map of individual points on our map, figure 3.5 below shows an extract of this using the data gathered from Galileo traced onto a map using Plotly Express.

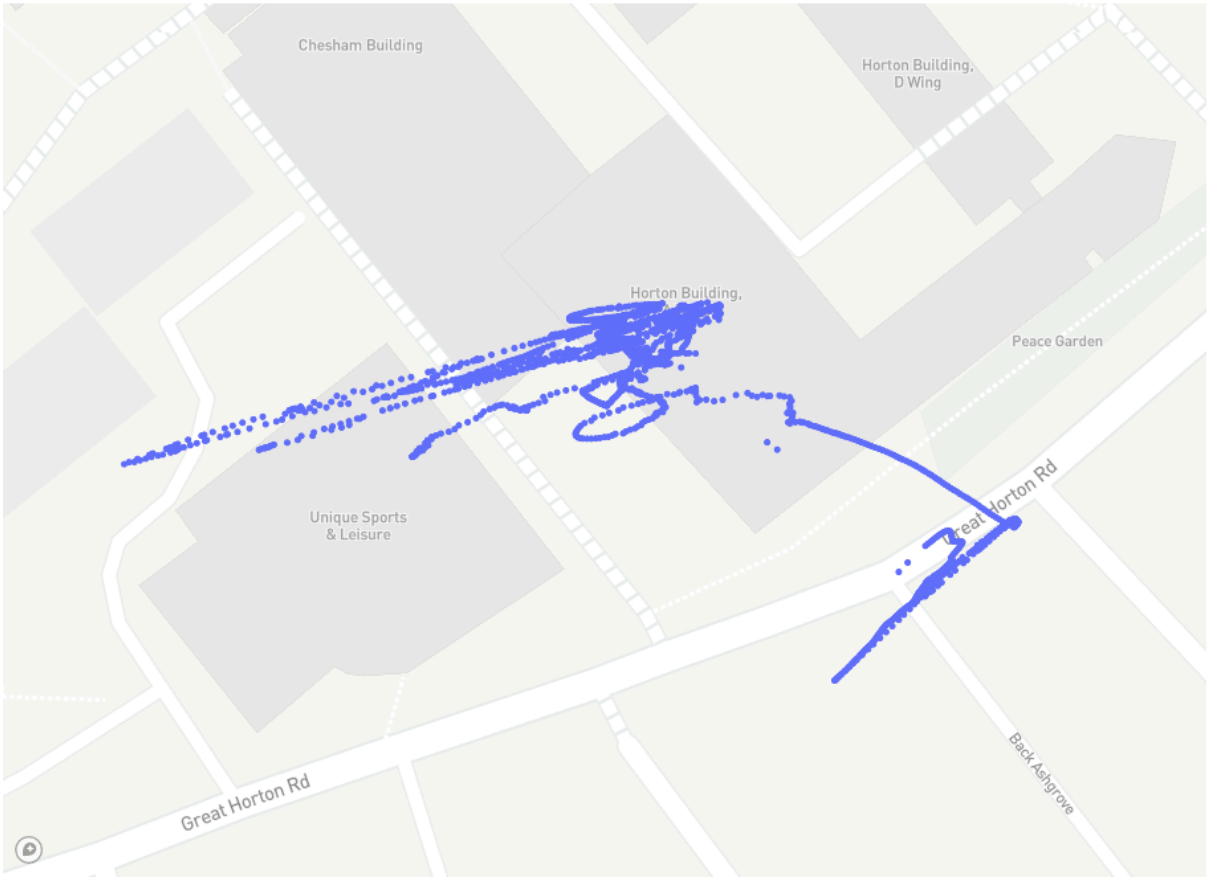


Figure 3.6 - Galileo Data Points Mapped Geographically using only Plotly Express

Finally, using a combination of the above we can create a geofence to analyse if any points completely fall outside of the University of Bradford campus, this can be used to see if the device

was moved outside the area we “fenced off” or if a data point taken was extremely inaccurate. The geofence can also then be applied later on.

In [21]:
```
polygon = gpd.read_file("bradford-uni.geojson")

fig, ax = plt.subplots(figsize=(10,10))

#keep in this order, or plotting data is not shown!!!
polygon.plot(ax=ax)
gps_geodata.plot(ax=ax, color="black")
plt.tight_layout()
plt.axis("on")

plt.show()
```

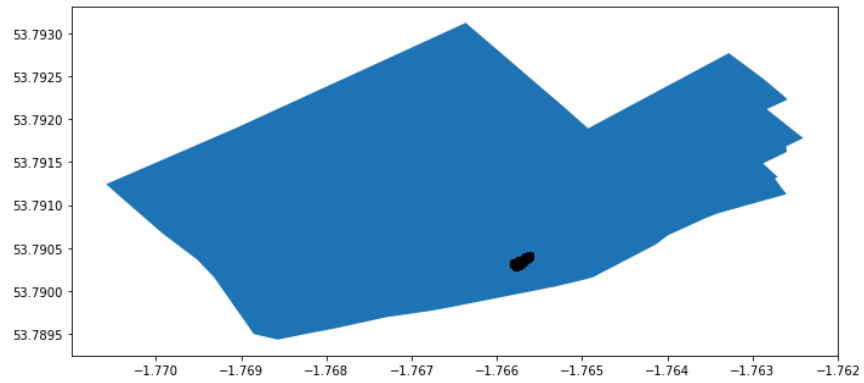


Figure 3.7 - The Mapped Geofence of the University of Bradford Campus

We can also analyse whether or not any points have fallen outside of our chosen geofence by analysing the dataframe itself.

In [22]:
```
mask = (polygon.loc[0, "geometry"])
pip_mask_geofence = gps_geodata.within(mask)

gps_geodata.loc[:,"geofence"] = pip_mask_geofence
gps_geodata.sample(5)
```

Out[22]:

	id	Longitude	Latitude	Height	Year	geometry	geofence
976	3326	-1.765739	53.790325	224904	2022	POINT (-1.76574 53.79032)	True
196	2546	-1.765741	53.790310	215995	2022	POINT (-1.76574 53.79031)	True
779	3129	-1.765632	53.790394	206031	2022	POINT (-1.76563 53.79039)	True
615	2965	-1.765673	53.790387	208445	2022	POINT (-1.76567 53.79039)	True
689	3039	-1.765655	53.790387	207595	2022	POINT (-1.76565 53.79039)	True

Note that all the values within the geofence are marked as "True" this is due to the geofencing polygon being relatively big in comparison to the gps location data points.

Therefore, all the data points fall within the generated geofence polygon (as it covers the entire university main campus).

Finally, we can map whether or not points fall inside or outside our geofence, in the case for the orignal map, all points will fall within since no points fall outside of the university campus.

Figure 3.8 - Analysis of the Dataframe Using Jupyter Notebook - All Values Fall within our Geofence.

Finally, we can employ a visualisation method where we map our geofenced points based on colour. We can map geofenced points in blue and non-geofenced points in red.

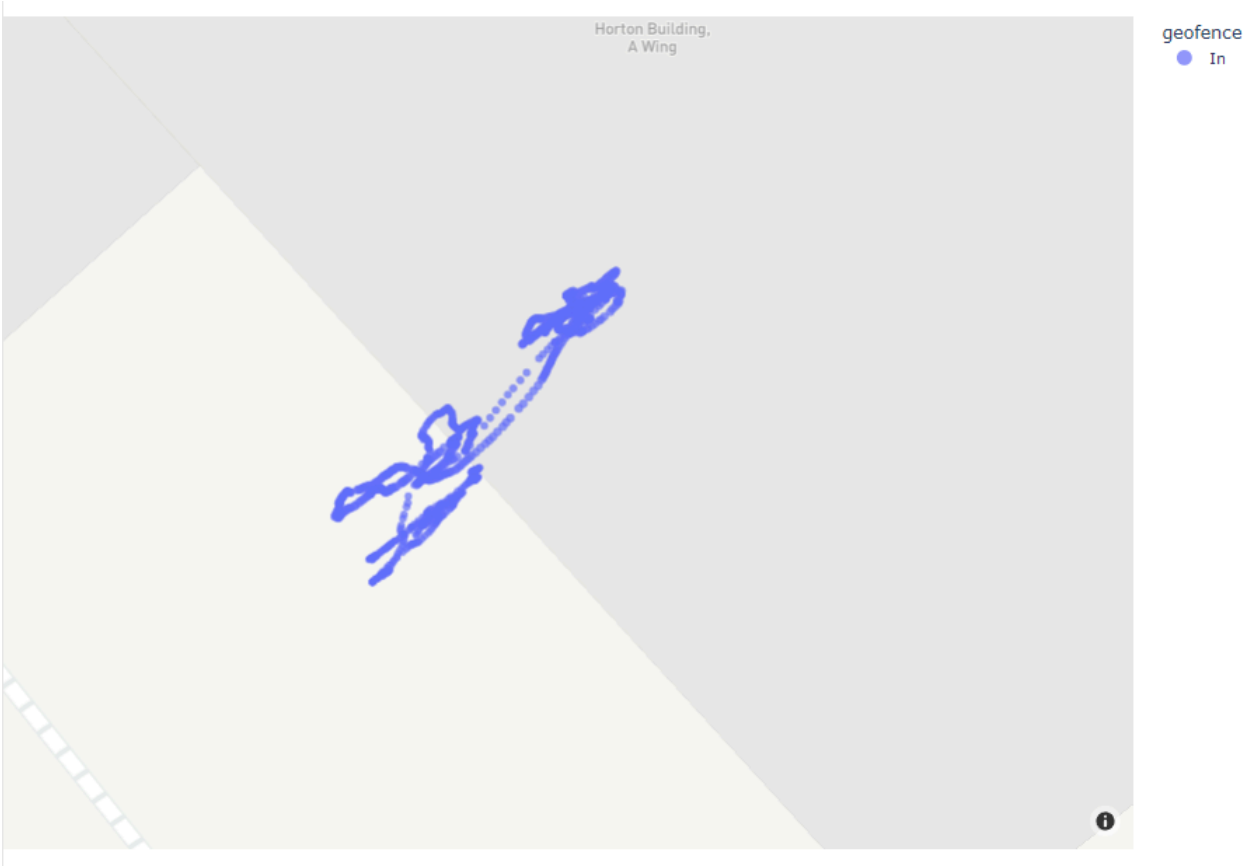


Figure 3.9 - Colour Coded Geofenced Points using GPS as a Reference

To finalise, we can make our final observations and conclude the initial analysis of the preliminary data, with some initial conclusions. Initially, we may want to analyse precision and accuracy using the following method of analysing the standard deviation we took before, as well as using the following visual method:

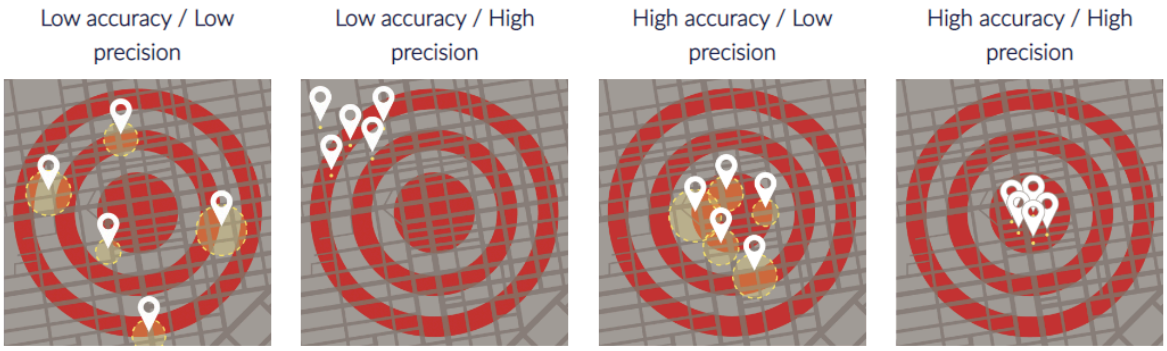


Figure 3.10 - Accuracy and Precision, A Visual Perspective

Using a visual and numerical method of evaluating the accuracy and precision of each constellation we can make some conclusions of both of these metrics for each constellation. Below we use the methods mentioned earlier to map and geofence each point for each constellation.

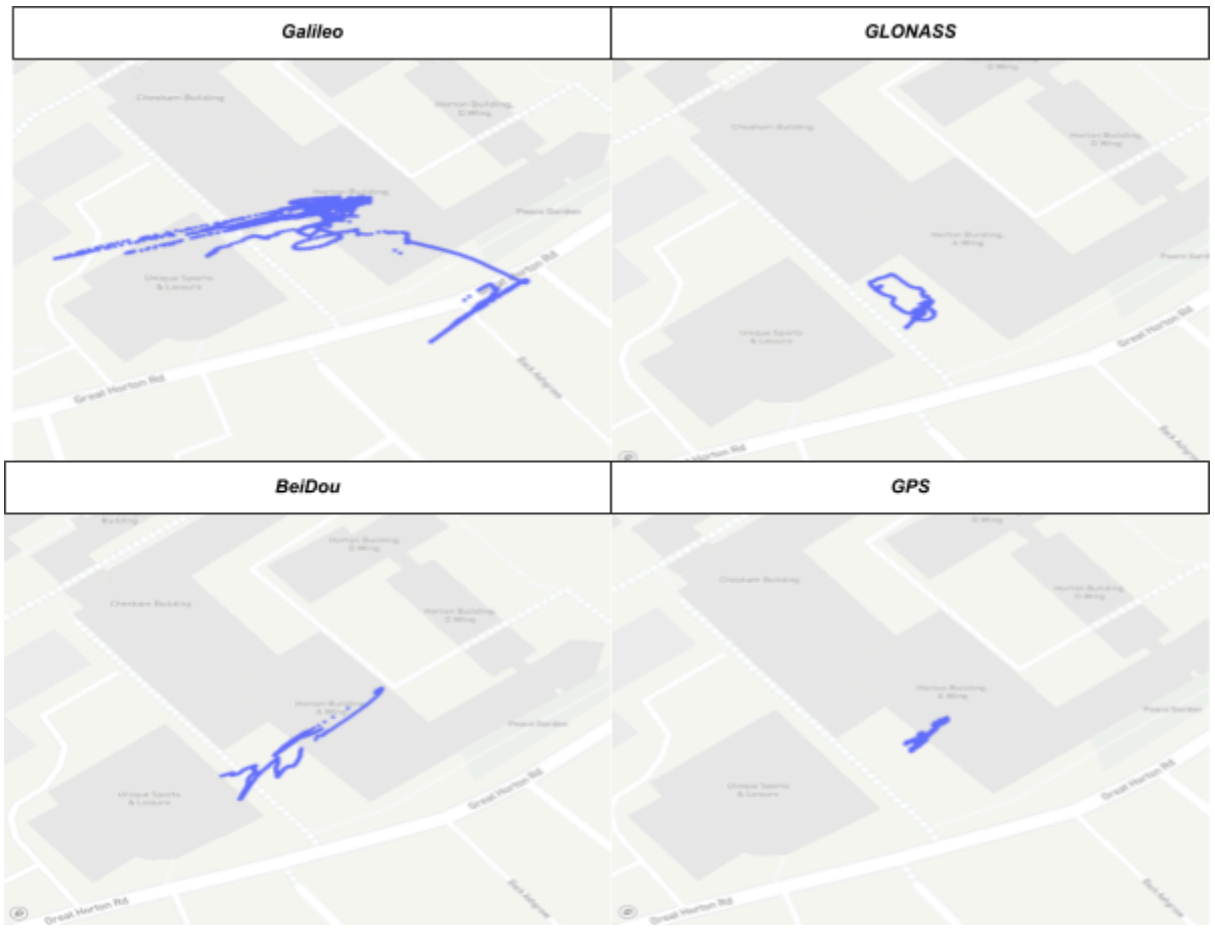


Figure 3.11 - Each Constellation´s Data Points Geometrically Mapped and Geofenced

Using both visual and numerical perspectives taken from Table 3.0 and Figure 3.10 respectively, we can make the following comments:

- GPS from a visual and numerical perspective has the highest precision and accuracy.
- GLONASS performed well in terms of precision (Table 3.0) but did not perform well in terms of accuracy compared to GPS (Figure 3.10).
- BeiDou seemed to perform well numerically, however, from a visual perspective, the accuracy AND precision were both low.
- Galileo performed the worst numerically and visually - this could be for a number of factors discussed below.

In terms of why we would actually need to take and analyse these readings, consider using a GPS device such as your phone, if the readings are inaccurate and imprecise, e.g. the device puts you 3-4 streets away from where you actually are, this can lead to situations where a person could get lost, incorrectly position themselves or just generally provide poor quality positioning information.

Thus, we can confirm through numerical-visual methods that GPS is the most accurate and precise for data readings, in this case. However a number of points could be made which could have affected the data readings:

- The first data entry points were taken from Galileo, which could have impacted accuracy and precision due to human errors in configuration.
- Weather factors could have impacted overall readings, due to adverse weather during data takings.

Another mention of AIS, an AIS transceiver will use GPS technology (e.g. a GPS receiver), above we took readings which indicated that GPS had the highest overall accuracy and precision when compared to other similar GNSS constellations. Therefore, we can conclude that it would be reasonable to use AIS which utilises GPS in our final model over other GNSS constellations.

Finally, in terms of future improvements, in terms of what we have covered in this section, a number of improvements could be considered:

- Usage of a smaller geofence limited to only the research labs could be considered more appropriate, due to our geofence of the entire campus already being quite large.
- Larger survey size for our data taken over longer periods of time, applying it to the overall theme of our thesis (vehicles) and taking more data overall. This could however negatively affect processing times for future models or analysis while positively impacting accuracy and precision readings.
- Prediction of future points - this would directly relate to the theme of this thesis, in that we would be predicting where a point would be to a certain degree of accuracy.

3.3. Vessel/Maritime Data Analysis

3.2.1. Data Background

As we mentioned the usefulness of the data from Ray, et. al. and Iphar, et al's papers cannot be understated, as such, I have decided to use this data for my own model, particularly, the data relating to tracklets in Iphar's data, which contains data relating to vessel coordinates, time, bearing and speed.

The data contains AIS information from vessels operating in seas around Brest, France and covers the Celtic Sea, North Atlantic Ocean, English Channel and the French Bay of Biscay. The data file itself is called "Maritime Routes and Tracklets" and contains five comma separated values (.csv) files, containing different pieces of data:

Table 3.3 - Data File Description

File Name	File Description
all_points.csv	This data represents maritime vessel routes as point clusters.
nomen.csv	Data representing route nomenclature.
poi_csv	Data representing ports of interest.
prototypes.csv	A prototype form of maritime route information.
tracklets.csv	Data representing maritime tracklets for vessels.

The two files we will focus on now are "all_points.csv" and "tracklets.csv" due to the fact that these data files contain the data variables we need in order to make a prediction.

The first file, “all_points.csv”, as mentioned table 3.1 contains vessel routes as point clusters, this type of data is useful for analysing vessel behaviour, identifying a port of call and studying vehicle trajectory patterns at a higher, overall level. However, overall using point cluster data to make predictions might prove more challenging, due to a lack of continuous trajectory data.

However, the second file, “tracklets.csv” will contain data which will have continuous trajectory data which could provide a better representation of the kinematic state of the vessel, this is because the maritime tracklets data contains more data on the sequence of positions and timestamps related to a vessel, making it easier to perform time series forecasts. This could imply that it would be easier to make a prediction model for vessel kinematic paths using this data.

Therefore, since the goal is to predict the kinematic state of a vehicle (vessel, in this case), it would be preferable to use the “tracklets.csv” file over “all_points.csv”. However, it is worth noting though that for predictions of overall behaviour of vessels or an overall pattern of vessel behaviour, “all_points.csv” may be preferred.

3.2.2. Initial Data Analysis and Preprocessing

We begin the initial preparation of our model by selecting the tools used to develop our model as well as preparing our data.

We will use Python to develop an implementation of our model along with the libraries we used in the preliminary data analysis of GNSS constellations. The initial data in the file was set with a non-standard delimiter, making it not readable with a base Pandas read_csv() function; the delimiter must be set before being readable.

In [2]:
```
#import dataset
db = pd.read_csv("tracklets.csv")
db
```

Out[2]:

	idtracklet|id1|mmsi1|speed1|course1|heading1|lon1|lat1|ts1|id2|mmsi2|speed2|course2|heading2|lon2|lat2|ts2|id3|mmsi3|speed3|course3|heading3|lon3|lat3|ts3|id4|mm
0	
1	
2	
3	
4	
...	
795	
796	
797	
798	
799	

800 rows × 1 columns

```
In [3]: #re-import dataset and set delimiter
        db = pd.read_csv("tracklets.csv", sep="|")
        db
```

Out[3]:

ading1	lon1	lat1	ts1	id2	...	ts4	id5	mmsi5	speed5	course5	heading5	lon5	lat5	ts5	route
511	-4.551695	48.344520	1447148032	4129047	...	1447148121	4129188	228005700	3.9	325.8	511	-4.551893	48.343353	1447148153	R_07
69	-4.733975	48.301247	1456438812	14817884	...	1456438842	14817927	249104000	8.5	68.5	69	-4.731818	48.301840	1456438851	R_14
86	-4.885512	48.404390	1451215056	8618323	...	1451215075	8618335	228017700	14.7	84.3	88	-4.882973	48.404560	1451215081	R_01
126	-4.924165	48.405033	1457099514	16024163	...	1457099527	16024186	227008170	12.4	130.0	119	-4.922998	48.404335	1457099531	R_11
73	-4.550415	48.351140	1445447976	2144611	...	1445448112	2144738	228017700	16.0	72.7	71	-4.535050	48.354515	1445448122	R_06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
511	-4.639665	48.315987	1446681907	3554191	...	1446681925	3554214	227730220	19.5	55.2	511	-4.636951	48.317173	1446681932	R_0
511	-4.547798	48.351620	1444922333	1585481	...	1444922338	1585485	227005550	23.2	69.3	511	-4.547202	48.351772	1444922338	R_0
332	-5.186498	48.084835	1445959658	2761038	...	1445959718	2761117	258316000	12.2	337.0	333	-5.188832	48.088500	1445959727	R_0
103	-4.465165	48.318333	1448009887	5167735	...	1448009892	5167744	228186700	10.9	105.0	101	-4.464665	48.318165	1448009893	R_0
308	-4.773423	48.041780	1445255813	1922136	...	1445255852	1922194	227364000	8.6	305.3	312	-4.775925	48.042880	1445255863	R_0

Figure 3.12 - Initial Data Before and After Setting a Delimiter

We are now able to read the data, here we can notice a few things wrong with the data that require preprocessing:

- The data is storing our time ("ts(n)" - short for timestamp, where n is the number associated with the timestamp) in UNIX timestamps, which are not inherently human-readable.
- The time series tracklet data should be chronological.
- Any missing or null values should be found and removed, this can also be done to check for gaps in timestamps.
- Other feature engineering, such as finding the time intervals between vessel positions or the distance travelled between positions.

Here is also some information about the data, for future reference:

- **"idtracklet":** The unique identifier for each tracklet in the file.
- **"id*n*":** Unique identifier for *n* vessels within the tracklet, in this case, *n*=5.
- **"mmsi*n*":** Maritime Mobile Service Identity, this is an AIS unique identifier for vessels.
- **"speed*n*":** Vessel speed in knots.
- **"course*n*":** Vessel direction of travel in degrees, typically known as course over ground.
- **"heading*n*":** The headings of the vessel, typically, this is the direction of the vessel's bow in degrees.
- **"lon*n*":** Vessel longitude, one of our primary variables.
- **"lat*n*":** Vessel latitude, another one of our primary variables.
- **"ts*n*":** Timestamp in terms of the associated data point.
- **"route":** Information about the route, has an accompanying file.

Given our discussion regarding required variables from earlier, our likely vehicle state predictor model is likely to utilise:

- **speed*n***
- **course*n***
- **heading*n***
- **lon*n***
- **lat*n***

After importing all relevant datasets, we begin by analysing the data for null values. We build a heatmap to visually analyse the data:

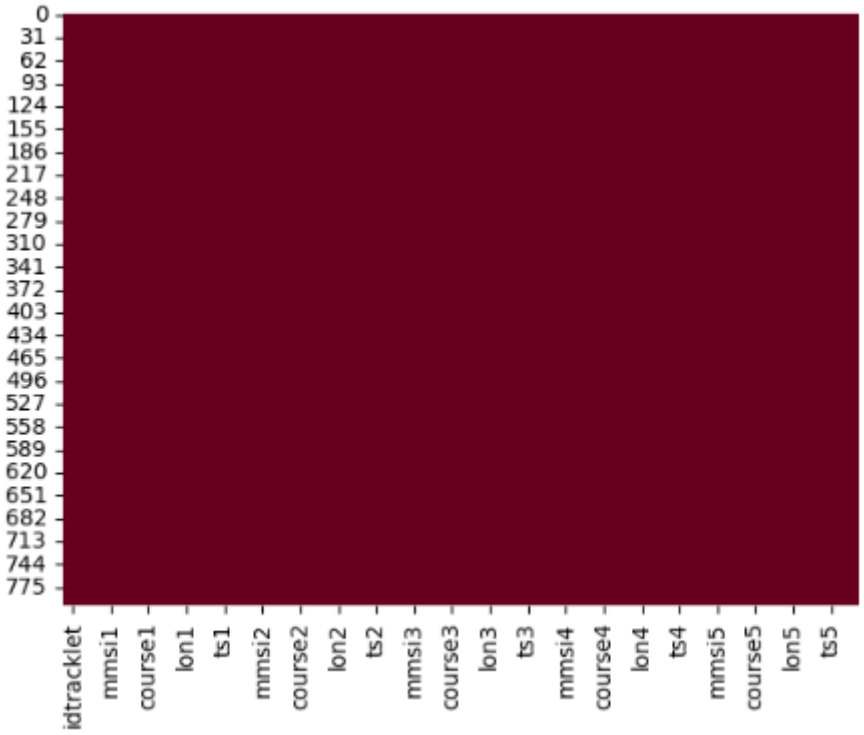


Figure 3.13 - Tracklet Dataset Heatmap

Here we can visually see that there are no null values contained within the dataset. However, since we are dealing with a large amount of data, we still implement the below function to drop any potential null values that have been missed.

```
In [7]: #drop potential na values
        db.dropna(inplace=True)
        db
```

Out[7]:

	idtracklet	id1	mmsi1	speed1	course1	heading1	lon1	lat1	ts1
0	1	4128997	228005700	6.7	175.0	511	-4.551695	48.344520	1447148032
1	2	14817869	249104000	8.7	69.0	69	-4.733975	48.301247	1456438812
2	3	8618319	228017700	14.6	83.7	86	-4.885512	48.404390	1451215056
3	4	16024159	227008170	12.3	125.0	126	-4.924165	48.405033	1457099514
4	5	2144526	228017700	16.5	73.4	73	-4.550415	48.351140	1445447976
...	...	...	...	...	...	...	...	...	...
795	796	3554186	227730220	19.1	56.9	511	-4.639665	48.315987	1446681907
796	797	1585479	227005550	23.1	68.9	511	-4.547798	48.351620	1444922333
797	798	2761022	258316000	12.1	338.0	332	-5.186498	48.084835	1445959658
798	799	5167729	228186700	10.9	104.0	103	-4.465165	48.318333	1448009887
799	800	1922119	227364000	8.8	303.4	308	-4.773423	48.041780	1445255813

800 rows × 42 columns

Figure 3.14 - Code Extract for Dropping Null Values

Now we may want to actually read the data contained within the UNIX timestamps, so below we implement code to read the UNIX timestamps. Here we can check the validity of the data (i.e. see if

the code correlates to its data description for when it was collected). Please note that this code is commented out in the actual implementation due to computational errors.

In [95]:
```
#only needed if you need to read the UNIX timestamps

#process timestamps to human readable format
db['ts1'] = pd.to_datetime(db['ts1'], unit='s')
db['ts2'] = pd.to_datetime(db['ts2'], unit='s')
db['ts3'] = pd.to_datetime(db['ts3'], unit='s')
db['ts4'] = pd.to_datetime(db['ts4'], unit='s')
db['ts5'] = pd.to_datetime(db['ts5'], unit='s')
db
```

Out[95]:

lat1	ts1	id2	...	ts4	id5	mmsi5	speed5	course5	heading5	l
48.344520	2015-11-10 09:33:52	4129047	...	2015-11-10 09:35:21	4129188	228005700	3.9	325.8	511	-4.551
48.301247	2016-02-25 22:20:12	14817884	...	2016-02-25 22:20:42	14817927	249104000	8.5	68.5	69	-4.731
48.404390	2015-12-27 11:17:36	8618323	...	2015-12-27 11:17:55	8618335	228017700	14.7	84.3	88	-4.882
48.405033	2016-03-04 13:51:54	16024163	...	2016-03-04 13:52:07	16024186	227008170	12.4	130.0	119	-4.922
48.351140	2015-10-21 17:19:36	2144611	...	2015-10-21 17:21:52	2144738	228017700	16.0	72.7	71	-4.535

Figure 3.15 - UNIX Timestamp Readability Code Extract

We also implemented code after the above to capture a correlation matrix for later usage. This is not pictured in the current code extracts here, for note, all code can be found in an accompanying file.

We next perform feature and target selection for our model. We take speed, course, heading and timestamp as our features and use longitude and latitude as our targets. We will predict vessel predictions to the next timestamp in this implementation for our model. We put the features and target variables into separate “x and y” datasets for later implementation.

Finally, with our preprocessing completed and our features and targets selected, we can proceed to split the data into training and testing datasets for implementation. This process is covered in the subsequent implementation section (4.2).

4. Vehicle State Predictors

4.1. The Usage of Vehicle State Predictors for Prediction Purposes

We are required to understand what exactly it is we are trying to predict, for example, for our AIS data we may want to predict the port that a vessel may go to, we may want to predict the route that a vessel might take, or we may want to predict a longitude and latitude point for where a vessel may go from a particular point on it´s route.

To begin, to keep with the theme of this thesis, we will begin by predicting where a vessel may be from a certain point in its route, essentially, we will undertake a time-series forecast for the kinematic state of the vessel.

In terms of the type of model we would like to use, we should try the deep learning approach over traditional statistical methods as they can offer more depth and performance metrics [8]. Before making this claim though, we will also test statistical methods below and compare the results against deep learning methods. The next section will cover the specific methodologies that will be used in the model.

It is also worth noting that traditional methods may also outperform deep learning and machine learning based methods, as seen in the examples provided in our earlier literature review. Therefore, traditional methods will also be included in this prediction model, along with machine learning approaches.

4.2. Methodology & Selection of Prediction Models

Using our initial processed input data above, we can assume a timeline for our predefined model (i.e. model flow design) which could be visualised as something like this:

Figure 4.0 - Simplified Model Flow Diagram

For this model, we will initially use two traditional models: a linear regression and a random forest model. We will also use a machine learning based model: a traditional neural network. The overall aim of these models is to take our initial input variables to then produce a longitude/latitude point in space as well as a point in time that the vehicle might be in (essentially an ETA).

The linear regression model should take our features, such as longitude and latitude, and our targets, which we will define within the implementation. Normally, an implementation of linear regression will work something like this [17]:

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 ... + b_n \cdot x_n$$

Eq. 2. Linear Regression

Where:

- “y” would be the target variable, sometimes referred to as the dependent variable.

- “b0” is the intercept.
- “b1 … b*n*” are feature coefficients.
- “x1 … x*n*” are independent features.

In the actual implementation, we use the Python library “sklearn” in order to implement this model in the form of code. Sklearn will adjust the feature coefficients by utilising OLS (Ordinary Least Squares) which will find coefficients that, as the name suggests, minimises a squared residual. The model will then be trained on our data, which will give rise to a prediction from the trained residuals and the above linear equation.

```
import sklearn
import pandas

#import the data
data = pandas.read_csv("our_data.csv")

#perform some preprocessing
data = data.formatUnix("column_name")
data.addColumn(calculateTimeIntervals)
data.encodeRouteInfo("columns…")

#feature selection
features = ["longitude", "latitude", "speed", course" …]

#data prep for features and targets
X = data[features]
y = data[['future_longitude', 'future_latitude']]

#split data
X_train, X_test, y_train, y_test = train_test_split(X, y, configurationParameters…)

#model creation
l_model = sklearn.LinearRegressionModel()
l_model.fit(X_train, y_train)

#predictions
Z = l_model.predict(X_test)

#below would go evaluations, testings, visualisations, etc…
```

Figure 4.1 - Pseudocode For Linear Regression Model Creation and Prediction

Random forest again will be a sklearn implementation. Random forests typically involve multiple decision trees, combining them to make a final prediction. Typically, a random forest will utilise bootstrapping, where we have a dataset *N*, where we select *N* samples to create subsets to train an entire decision tree. To reduce overfitting, randomness is introduced to the subsets, introducing diversity as well [18].

It is also worth noting that random forest models can be implemented as a classifier or as a regression model, it is more appropriate for our purposes to utilise a regression model though.

Typically a random forest is composed of *M* “trees”: *T1, T2…TM*. Where each tree “*TM“* takes a subset from the bootstrap “*Si”* of size *N* from some training data. A random subset “*F*” for features is considered for the split to reduce overfitting and increasing diversity. The tree is then constructed using “*Ti”*, *“Si”* and *“Fi”* to create recursive nodes.

For calculating a prediction using a random forest model, a combination of predictions taken from individual trees are used, we can utilise the following formula typically used in random trees to calculate this: [18]

$$RandomForest_{pred} = \frac{1}{M}\sum_{i=1}^{M} Tree_{pred}$$

Eq. 3. Random Forest Decision Tree Equation

Note that this is typically used in regression models, for classification models a modified but similar formula is used.

Moving onto the machine learning side, a traditional neural network, sometimes referred to as a MLP (multi-layer perceptron), consisting of an input layer: "*x1, x2 ... xn"*, hidden layer(s): "*h1, h2 ... hn"* ("*an"* in the below figure, in that case), and an output layer "*y1, y2, yn"*. Layers will contain "neurons" which essentially conduct computations and pass information or weightings onto the next layer [19].

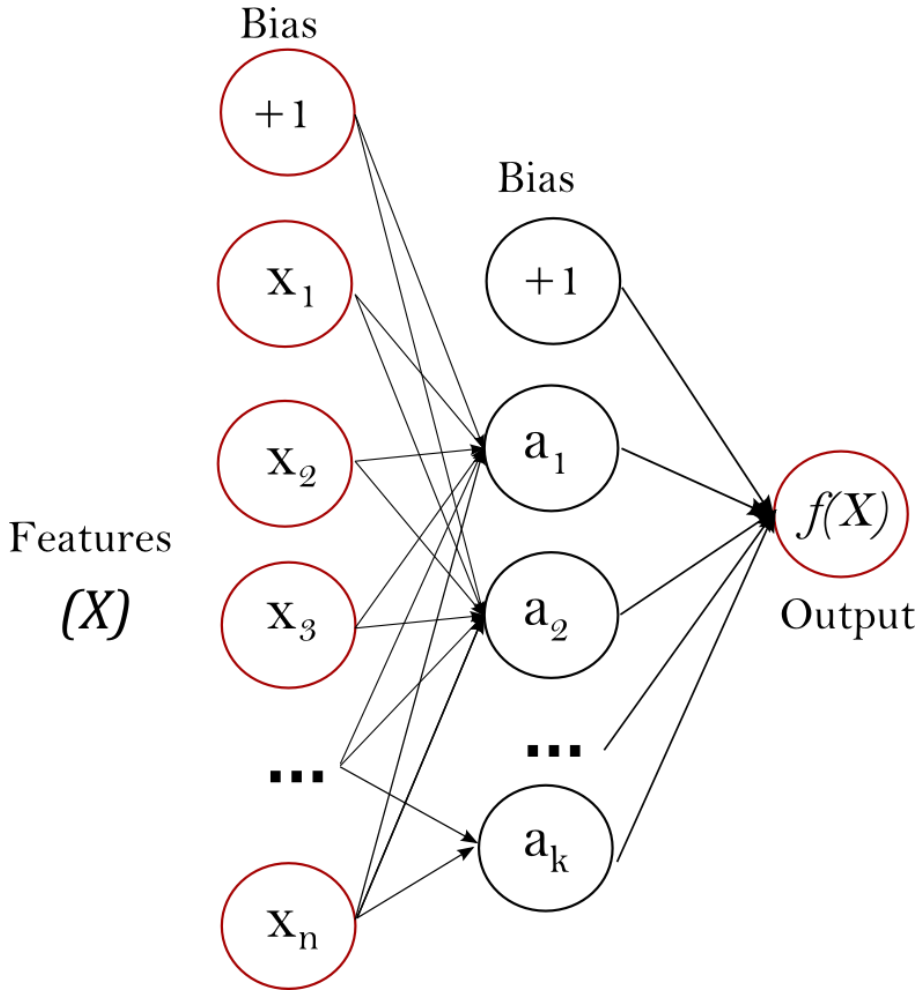


Figure 4.2 - A Traditional Neural Network Utilising One Hidden Layer [19]

It is worth noting that an activation function "*f*(*z*)" is used in the hidden and output layers, the computation of the neural network begins by feeding the input layer, computations within the hidden and output layer can be typically be calculated as the following:

For the hidden layer:

$$h_i = f(\sum_{j=1}^{n} w_{ij} x_j + b_i)$$

Eq. 4. Hidden Layer Equation

And again for the output layer:

$$y_i = f(\sum_{j=1}^{m} w_{ij} h_j + b_i)$$

Eq. 5. Output Layer Equation

Where:

- "*w_ij x_j"* would be the weights (information) which are connected through neurons "*i"* and "*j"* from the previous layer: *"x"* or *"h"*.
- "*b_i"* would be the bias.
- "*h"* or "*y*" in this case would be the output of the activation function.

Also note that the above are both considered activation functions and are considered as forward propagation processes, in which, input data is put through the neural network in order to calculate the predictions we want.

Commonly with neural networks a loss function would be used to measure the difference between predicted outputs from the neural network and actual "true" values. In the implementation, we would use MSE (mean square error) as our loss function.

Additionally, backpropagation and gradient descent can also be calculated. Backpropagation would be used to optimise our loss function by updating weights and biases, calculating the gradient of the loss function (backwards propagation), this can be done through the following formula:

$$\frac{\vartheta Loss}{\vartheta w_{ij}} = \frac{\vartheta Loss}{\vartheta y_i} \cdot \frac{\vartheta y_i}{\vartheta z_i} \cdot \frac{\vartheta z_i}{\vartheta w_{ij}}$$

Eq. 6. Backwards Propagation Formula

Where:

- "*z_i"* would be the weighted input of neuron *"i"*.
- "*y_i"* would be the weighted output of neuron "*i"*.
- "*w_ij"* would be the weight from connecting neuron *"i"* to connecting neuron *"j"*.

Note that *"vartheta loss"* is in relation to a partial derivative of the loss function, for example "Vartheta Loss / Vartheta *w_ij*" would represent the partial derivative of the loss function in respect to the weight "*w*" to the connecting neurons "*i*" and "*j*".

The above can be considered how much a loss function changes in relation to variables such as neuron output and input.

Similarly, the gradient descent can be calculated using the following formula:

$$w_{ij} \leftarrow w_{ij} - \alpha \frac{\vartheta Loss}{\vartheta w_{ij}}$$

Eq. 7. Gradient Descent Formula

Where:

- Alpha is the learning rate.

Note that this formula indicates the direction the calculated gradients (from the updated weights and biases) move for a reduction of loss.

Following an iterative process of forward and backward propagation as well as gradient descent can be considered model optimisation, which will increase a neural network´s ability to make more accurate and higher quality predictions.

Finally, please note that Sklearn may change aspects of the above formulas due to updates to the library, documentation, etc…

4.3. Implementing Prediction Models

This section will cover the implementation of each model, presenting the model's results as well as going over data visualisation for all vessels, as well as an individual vessel. Evaluations for each model can be found in sections 4.3 and 4.4. All models were implemented on a HP Laptop with an x64 Windows 10-based operating system. The laptop utilises 8GB of installed RAM and a 2.3GHz AMD CPU. Instead of a dedicated GPU, the models leveraged AMD integrated graphics with 2GB of VRAM.

Ideally, when our models make predictions, we would like to see predicted values close to, or with a similar pattern to, the original values. Note that predicted values are marked in red and original values are marked in blue.

4.2.1. Linear Regression

To begin, we implement our linear regression model following the previously mentioned methodology. We incorporate the above methodology using SkLearn. We split the dataset into testing and training data, allocating a 20% split of the data for testing. A fixed random state of 42 was used for reproducibility purposes and for consistency.

We fit and predict the model using the above methodology, implementing it with SkLearn. We then put the predicted longitude and latitude values (the targets) into a new dataframe. Predictions are made to the next timestamp in the sequence. A code extract of this process can be found below.

In [18]:
```
#fit the model
lon_model.fit(X_train, y_train['lon1'])  #Longitude prediction
lat_model.fit(X_train, y_train['lat1'])  #Latitude prediction
```

Out[18]:
```
LinearRegression
LinearRegression()
```

In [19]:
```
#predict the model
lon_predictions = lon_model.predict(X_test)
lat_predictions = lat_model.predict(X_test)
```

In [20]:
```
#predicted longitude and latitude values

#print(lon_predictions)
#print(lat_predictions)

lon_lat_p = pd.DataFrame({
    "Longitude": lon_predictions,
    "Latitude": lat_predictions
})

lon_lat_p
```

Out[20]:

	Longitude	Latitude
0	-4.644427	48.254778
1	-4.777951	48.272358
2	-4.707740	48.307352
3	-4.706132	48.270410
4	-4.671591	48.304778
...	...	...

Figure 4.3 - Code Extract Showing Model Fitting and Prediction.

After completing the above process, we proceed into mapping the data as well as visualising the data onto a geographical map of the aforementioned regions we are analysing.

Beginning with this process, we start by taking geographical parameters for the above data and encompassing it into a “Point” object, which describes it´s spatial characteristics. This geo-spatial data is then put into a GeoDataFrame, a specific dataframe for geographical coordinates used by the GeoPandas library. An additional library is used to incorporate the “Point” object - Shapely. The Shapely library is also used in generating usable geo-spatial data.

The generated geo-spatial data can then be projected onto a map, we use the Folium library in order to generate a map. The map is generated using the mean of the geo-spatial data points (i.e. the coordinates) where the map is centred around the mean of the generated points. Initially, all vessels with both the original and predicted routes are displayed, with predicted coordinates in red and original coordinates in blue. Predictions were made to the next available timestamp. Below is the final full predicted states.

Note that this is for all vessels and all predictions and is intended to show the general pattern of predictions vs the original vessel predictions.

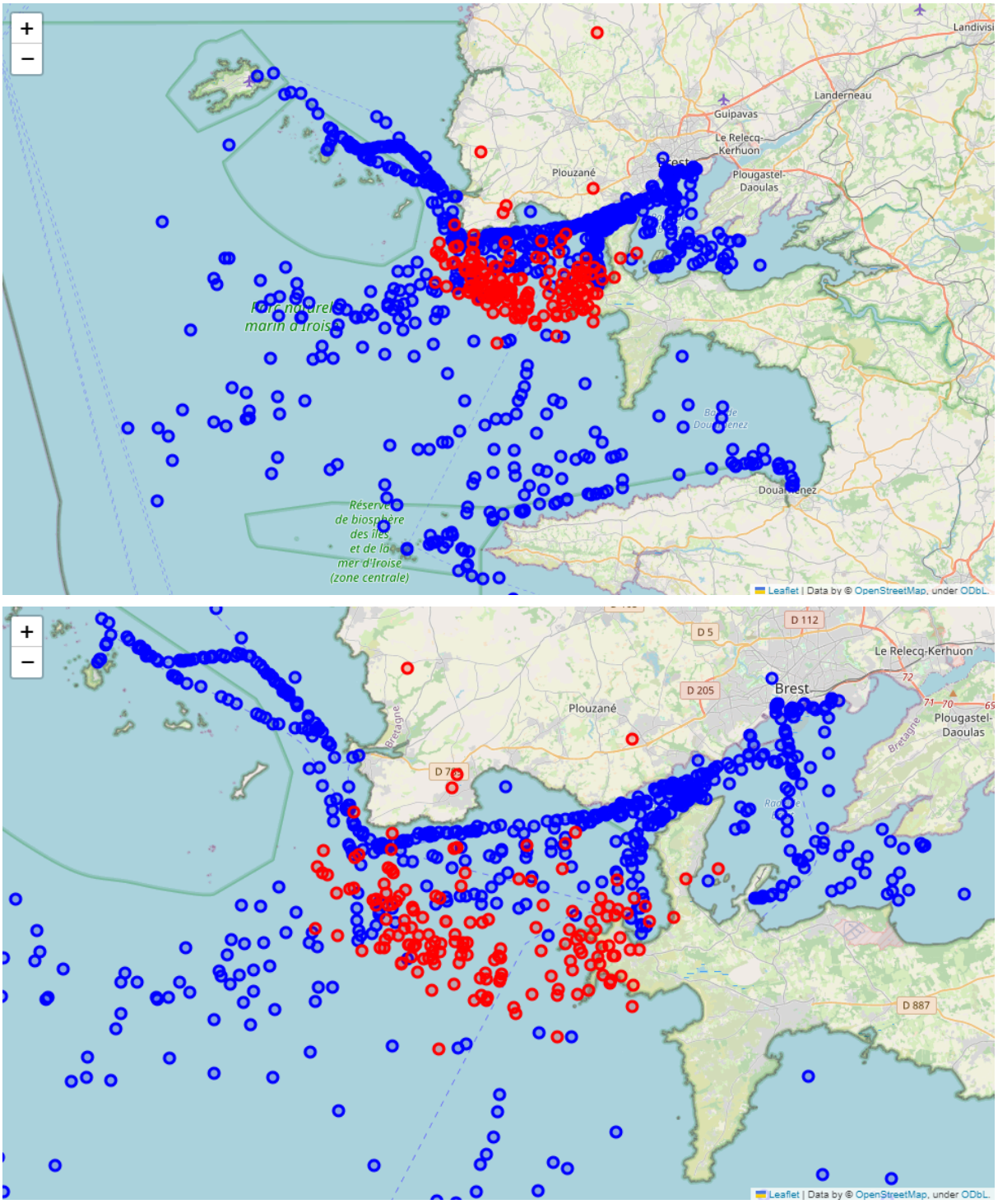


Figure 4.4 - Data Visualisation of Vessel State Prediction Using Linear Regression

The above could indicate that the vessels, in general, are coming into and out of port port as the data markers seem to congregate around port areas. However, some outliers show some predictions as being on land. These vessels can be referred to as having "land prediction abnormalities". Generally, vessel behaviour for the predicted values are showing an inaccurate congregation around the bay where the original markers are contained before venturing further out into sea. This issue could be caused by a lack of data, problems with the model (e.g. the model may only be making initial predictions, not accounting for what happens “later on”) or lack of relevant variables.

Looking into an individual vessel displaying this abnormal prediction behaviour, we can analyse the actual vs predicted route the vessel will take throughout it´s journey, as well as employ sequence markers to follow the predicted and actual routes.

Below shows a diagram for an individual vessel following its complete set of recorded routes alongside the predicted values from the model.

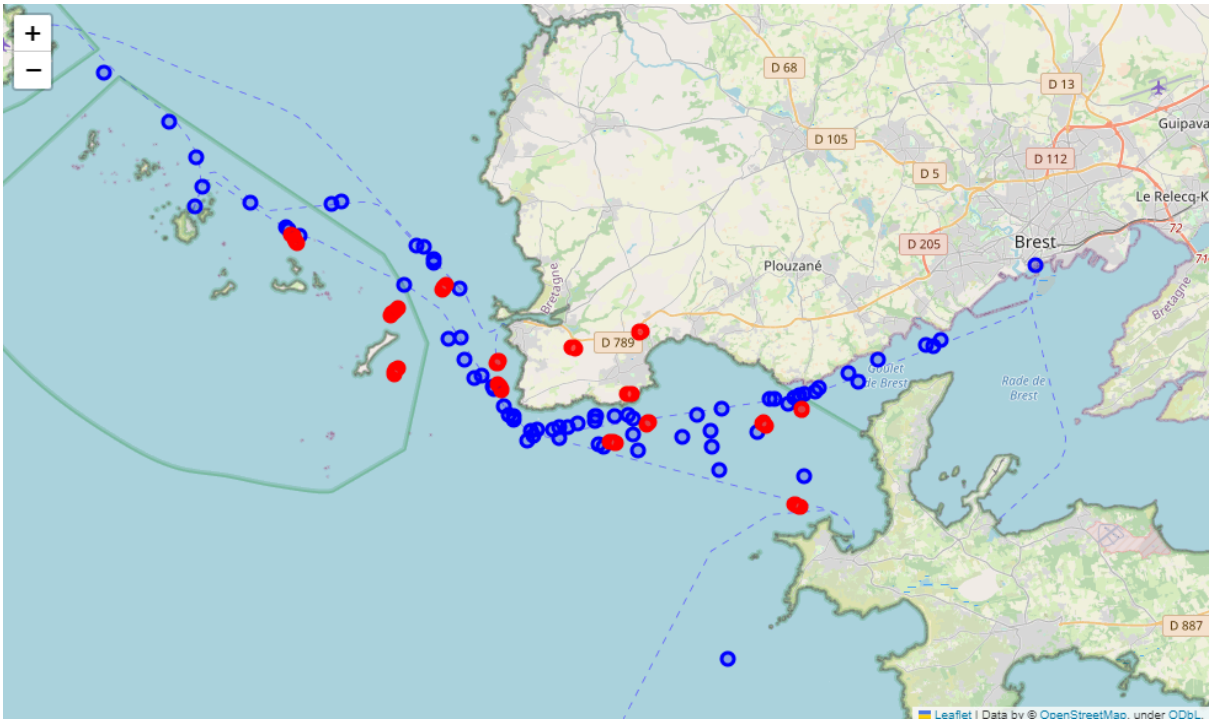


Figure 4.5 - Individual Vessel Route Prediction and Actual Value Markers, Linear Regression

Here we can see the results for the prediction of a single individual vessel - using MMSI (Maritime Mobile Service Identity - a type of temporary UID) 227008170. The above data seems to be congregated around certain points along the route a vessel takes, seemingly due to the fact that we predict up until a specific point, i.e. a timestamp. Some markers however are obviously inaccurate in the positioning, showing the ship as being on land, showing some inconsistency in the results for this particular ship showing abnormal prediction behaviour.

4.2.2. Random Forest Regressor

Secondly for our random forest model we follow the same steps for model setup, using the same configuration parameters (e.g. random state, test-training split). The only thing we add are the number of estimators (decision trees) to the random forest model, which we specify at 100.

Below are the results for all vessel predictions for the random forest regressor as well as the results for the same individual vessel showing a land abnormality:

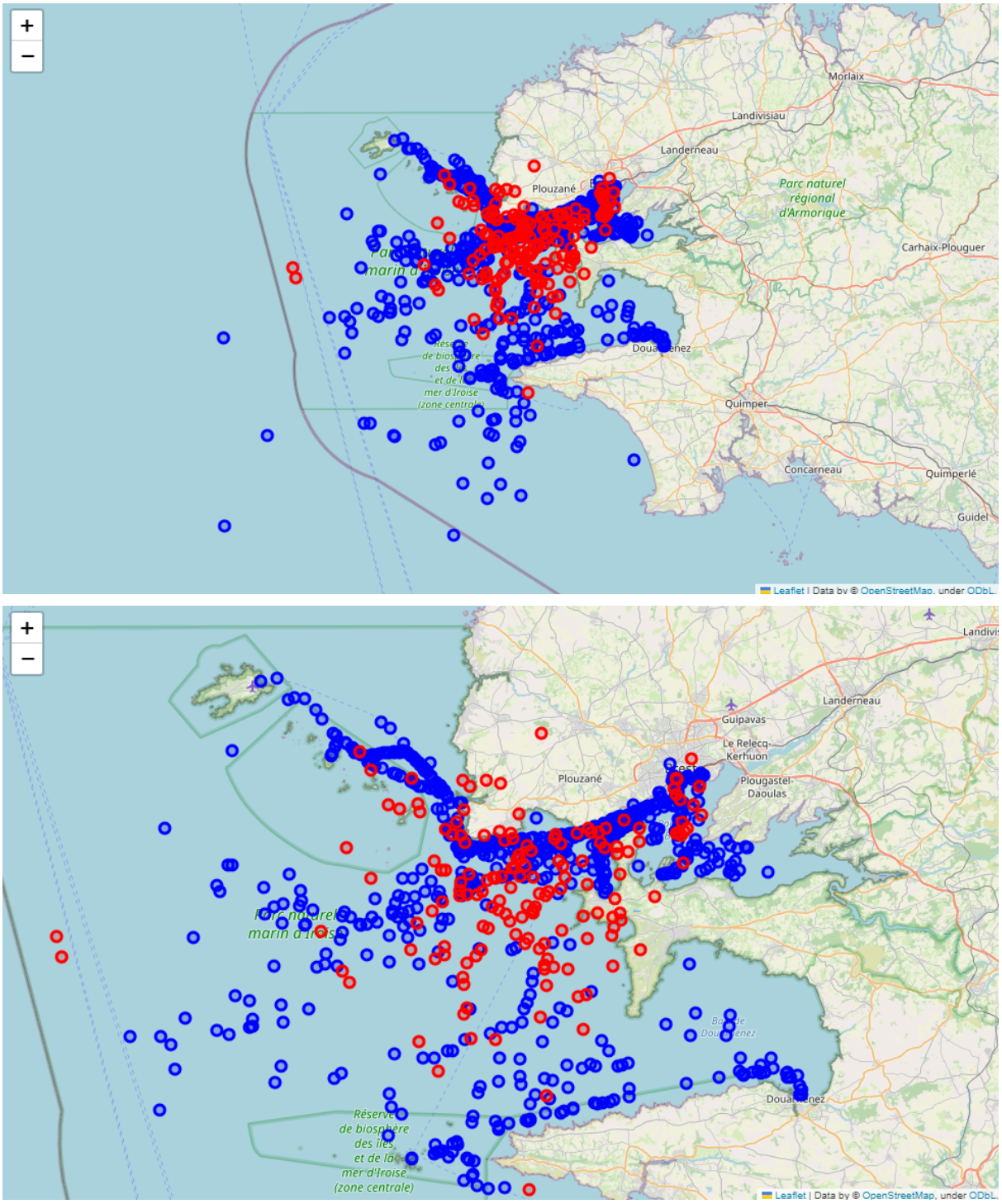


Figure 4.6 - Data Visualisation of Vessel State Prediction Using a Random State Regressor

Visually, it appears that the random forest regressor model seems to follow vessel routes more accurately, compared to linear regression. Here we also notice less extreme data congregation around a particular point and more visual evidence of "route following", there are some land abnormalities. Again, we will analyse an individual vessel showing a land abnormality as we did with the last model.

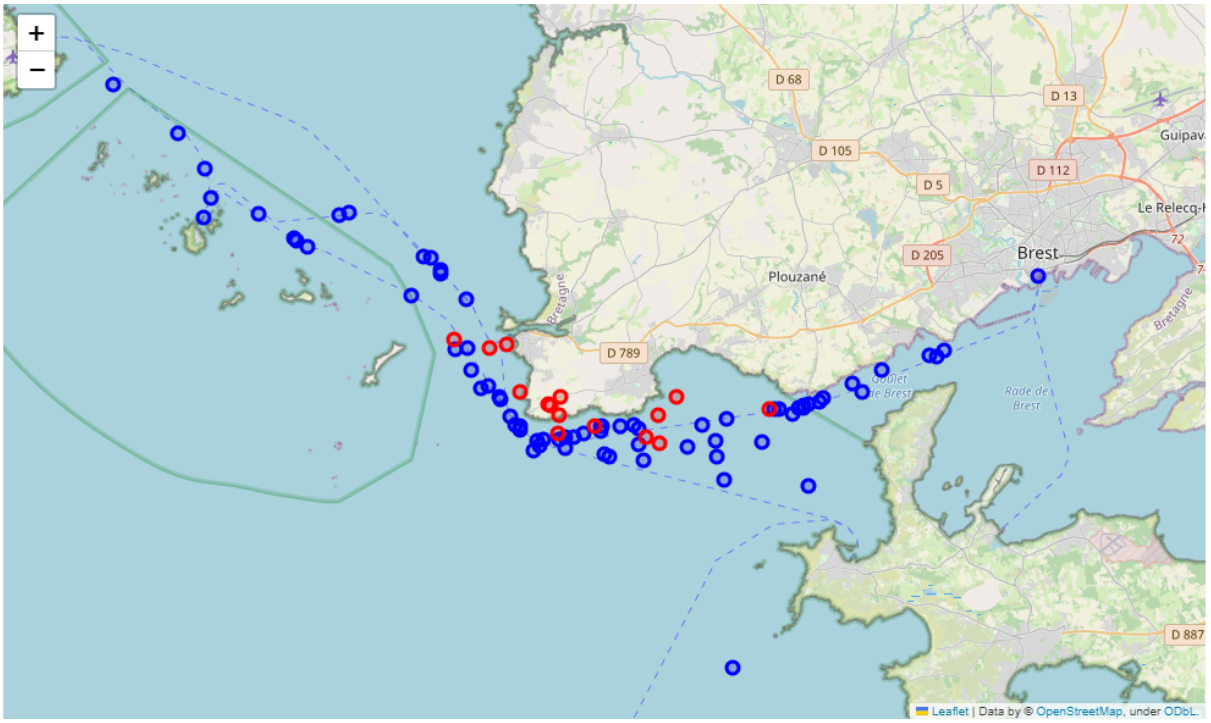


Figure 4.7 - Individual Vessel Prediction Data Visualisation, Random Forest Regressor

Here we include all the original markers for the same individual vessel we used for the linear regression model (MMSI - 227008170). For this, we can see that there are less predicted data points, however, we can see that despite there being less data points, visually, the predictions are following the original complete vessel route much more highly.

4.2.3. Neural Network

Finally, our neural network model, this model was utilised on similar configurations, however, we have to set up our neural network with different configurations for the model itself, as well as defining some prerequisites.

We begin by introducing a scalar to standardise our features. For this, we use what is commonly known as a "RobustScaler" which we implement with SkLearn using the following equation [20]:

$$SF = (X - Q1)/IQR$$

Eq. 8. Robust Scaler Equation

Where:

- X is the original value of a feature - e.g. speed.
- Q1 is the first quartile of a feature's value (25th percentile).
- IQR is the interquartile range.

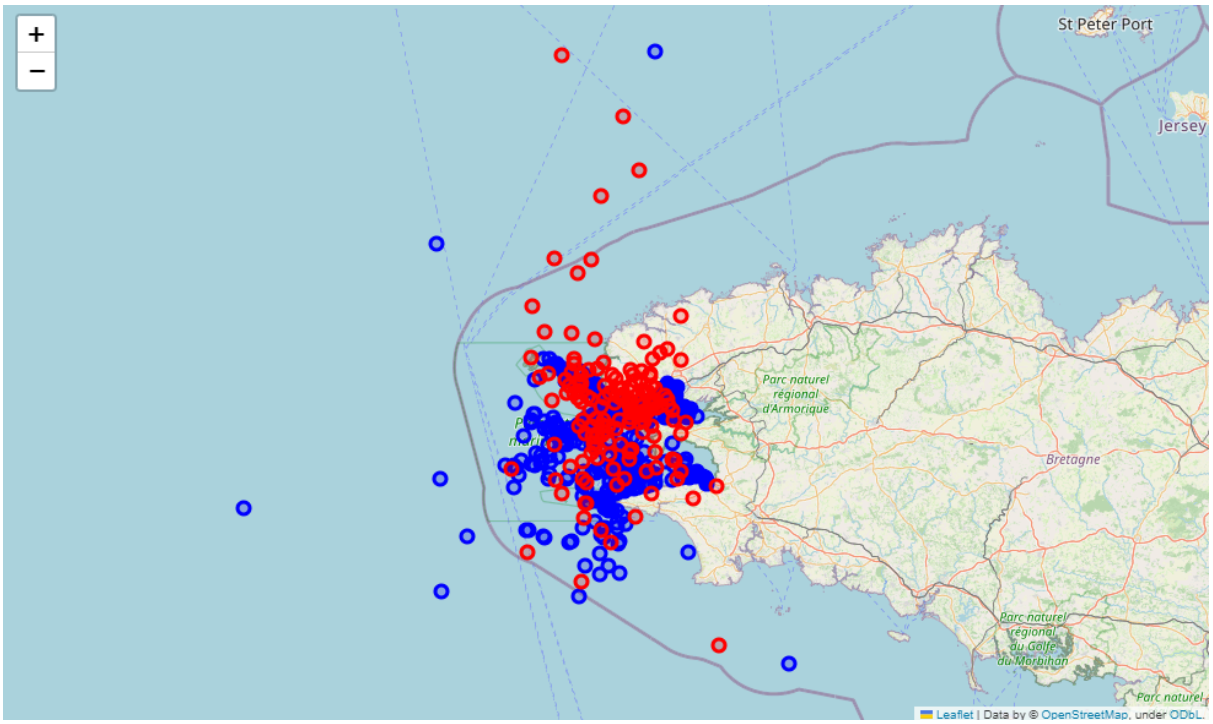
I considered using a Z-score normalisation scaler for this operation, however, a robust scaler worked better considering the nature of the data having lots of outliers and extreme values, it also worked better in maintaining the relative order of the data points compared to using the Z-score normalisation scaler. In practice, the standard scaler produced highly inaccurate geographical data, displayed as a line stretching all the way up to the Isle of Man, outside of our data range.

After implementing the robust scaler, the setup and the configuration procedures remain the same as the other models, except for the setup of the model itself.

Using the Tensorflow and Keras libraries to implement our model design, we define an input layer using the “Sequential” API, a linear stack of layers. Our input shape is the number of our features. We define 5 layers of: 64, 64, 32, 16 and 8 neurons, respectively. We also have an additional output layer with 2 neurons. The ReLU (rectified linear unit) activation function is used, introducing a level of non-linearity to the model. The model was fitted for 10000 epochs, using a batch size of 8 and a validation split of 20%.

Note that Tensorflow & Keras was used over SkLearn due to the fact that TensorFlow & Keras offer more model customization which increased the accuracy of the results over using the SkLearn implemented model, the same neural network model was used. The SkLearn model using the same configuration had issues during implementation, giving heavily and visibly inaccurate results.

After this, we compile, fit and predict the model and implement it into a dataframe that can be displayed on a map. Here is the completed implementation for all vessels, showing all predictions using the aforementioned parameters:



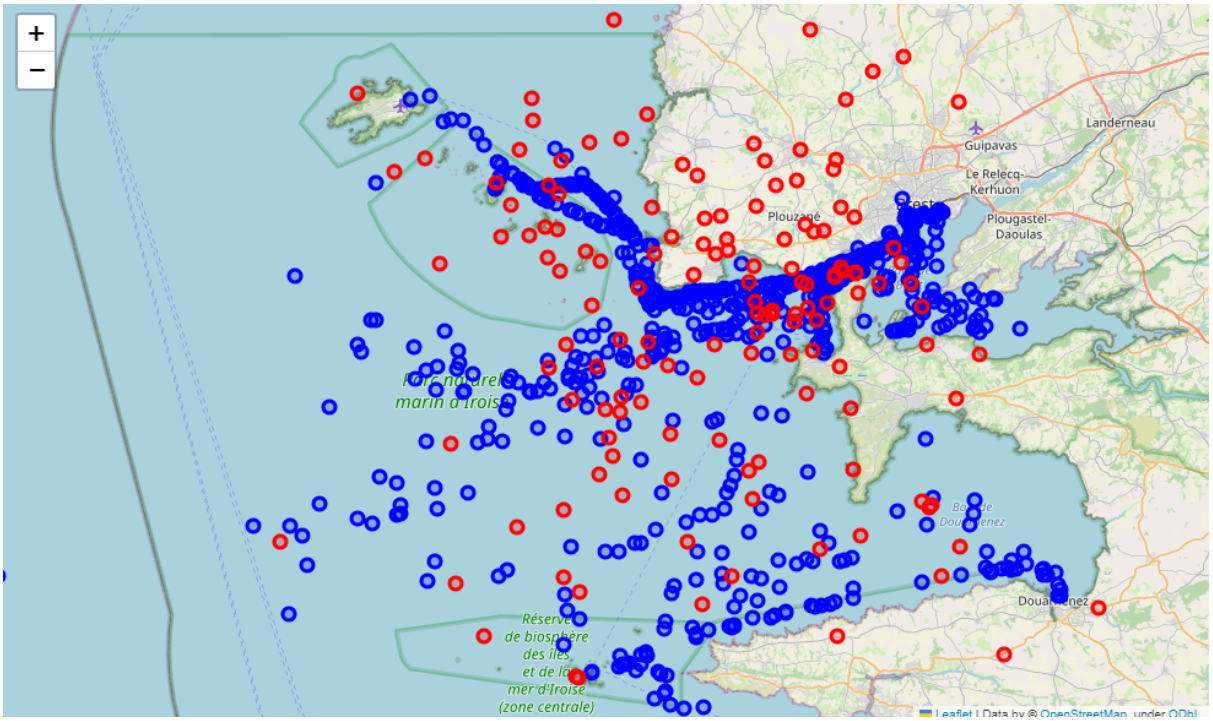


Figure 4.8 - Data Visualisation of Vessel State Prediction Using a Neural Network

Visually, we can see less “grouping” compared to the other models, however, there are more “land abnormalities”. Vessels from further out are now being predicted, showing that this model is moving outside of the bay, compared to other models.

Here are the results from the same individual vessel (MMSI “227008170”), the result was run 5 times: 3 times on 10000 epochs, 1 on 20000 epochs and 1 on 5000 epochs. with similar results:

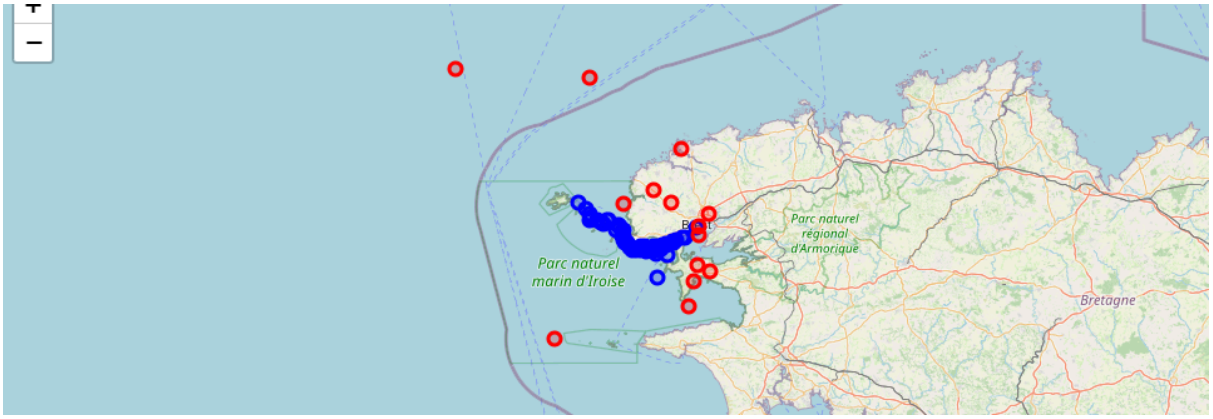


Figure 4.9 - Individual Vessel Prediction Data Visualisation, Neural Network

As you can see, from a visual perspective, the results are inaccurate and imprecise for the neural network. This could be due to overfitting or underfitting of the model, it could also be model complexity (i.e. a large number of epochs was used, potentially making the model too complex for the small amount of data that is contained) or a lack of data.

It is worth noting that computation time for the neural network was significantly longer compared to both models for linear regression and the random forest regressor.

4.4. Evaluation and Findings

Comprehensive table of evaluation metrics for each model:

Table 4.0 - Final Prediction Model Evaluation Metrics

Metric Name	Linear Regression	Random Forest	Traditional Neural Network
Standard Deviation (Lon)	0.18505	0.15781	0.20116
Standard Deviation (Lat)	0.15735	0.14052	0.18889
MSE (Lon)	0.03425	0.02493	0.04187
MSE (Lat)	0.02506	0.02004	0.03568
RMSE (Lon)	0.22645	0.10517	0.20463
RMSE (Lat)	0.06521	0.03791	0.18890
MAE (Lon)	0.13978	0.10811	0.13665
MAE (Lat)	0.10005	0.08214	0.12563
R2 Score (Lon)	0.11400	0.35508	-0.08307
R2 Score (Lat)	0.02693	0.22179	-0.38533

Each metric can have a different interpretation and effect on the quality of the model, as such, the evaluation of each metric separately seems the most appropriate course of action.

4.3.1. Standard Deviation

First, for our linear regression model, we will look at how spread out the data is by examining the original vs predicted vessel predictions on a scatter plot. We can use this to see how well our predictions are “mapped” onto the original values. A “healthy” graph would be indicated by having the predicted values having a close spread to the original values.

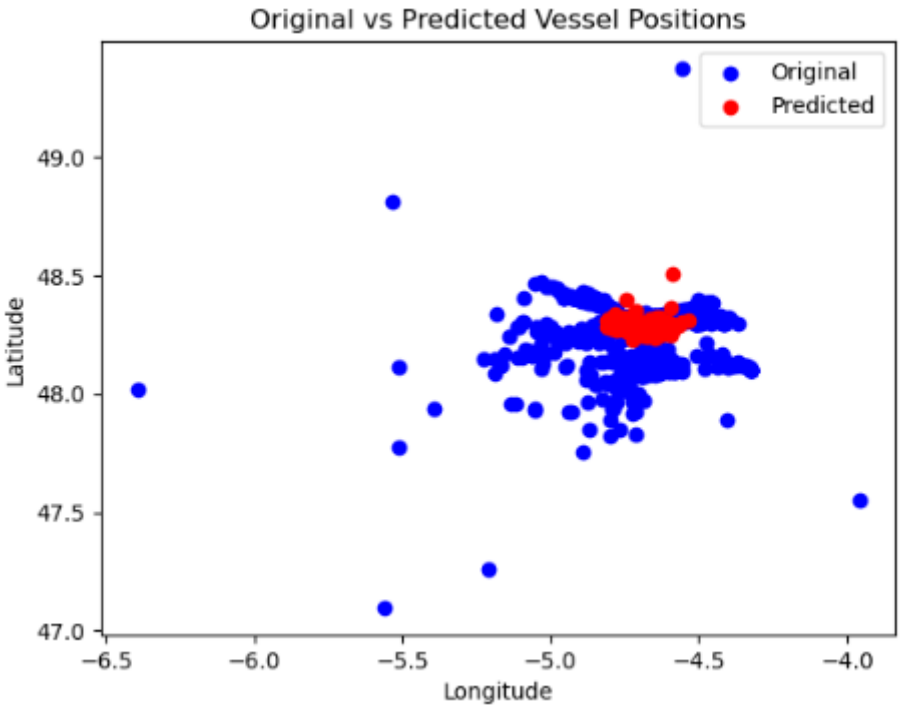


Figure 4.10 - Original vs Predicted Vessel Positions Scatter Diagram, Linear Regression

Here we can see that the values are not exactly spread evenly and are congregating in a small area, indicating that results are not exactly that accurate, from a visual perspective.

For the linear regression model, we analysed the spread of the data using a scatter plot. By also examining the standard deviation of our data using the following formula implemented using numpy we can get even more information which will enable us to further interpret our scatter graph more intricately:

Utilising our standard deviation formula, we are given a result of:

- Longitude Standard Deviation: 0.18505388384189272
- Latitude Standard Deviation: 0.15735459492718393

Also note that the original standard deviation is:

- Original Longitude Standard Deviation: 0.0
- Original Latitude Standard Deviation: 0.0

This result indicates that: the longitude and latitude for the original shows the "true value".
On average, it is suggested from the data that the model has a degree of accuracy of 18.5% from the true longitude and 15.7% degrees from the true latitude. This low interpretation can imply that the data is not spread out enough and is “congregating” in one spot.

Random forest regressor coordinate spread:

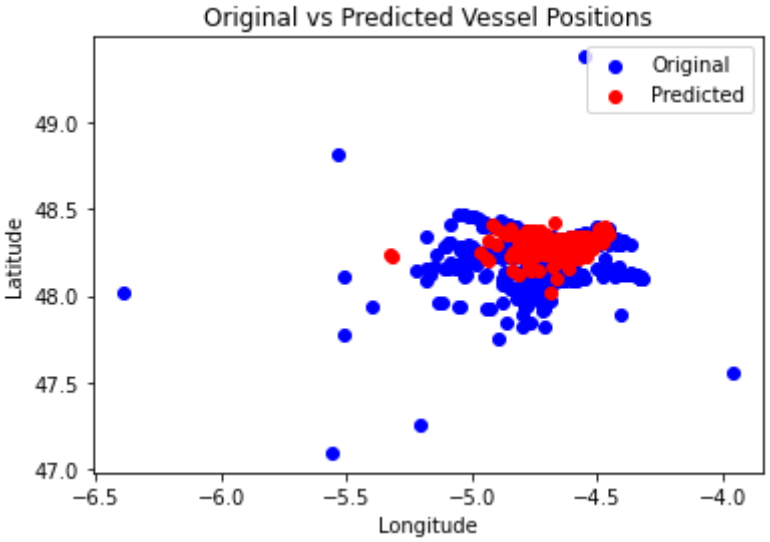


Figure 4.11 - Original vs Predicted Vessel Positions Scatter Diagram, Random Forest Regressor

Here we notice more spread on the data, yet the data is not spreading to outliers. While the data is still congregating around a similar spot to linear regression, the congregation of data is not as visually extreme, there is also more evidence of data spread around the original points, indicating that better predictions were being made compared to linear regression.

The standard deviation for this graph is:

- Longitude Standard Deviation: 0.15781871936065864
- Latitude Standard Deviation: 0.1405286748538807

Here we notice similar values to linear regression, however, they appear to be lower from a numerical point of view, indicating lower spread.

Finally, the neural network coordinate spread:

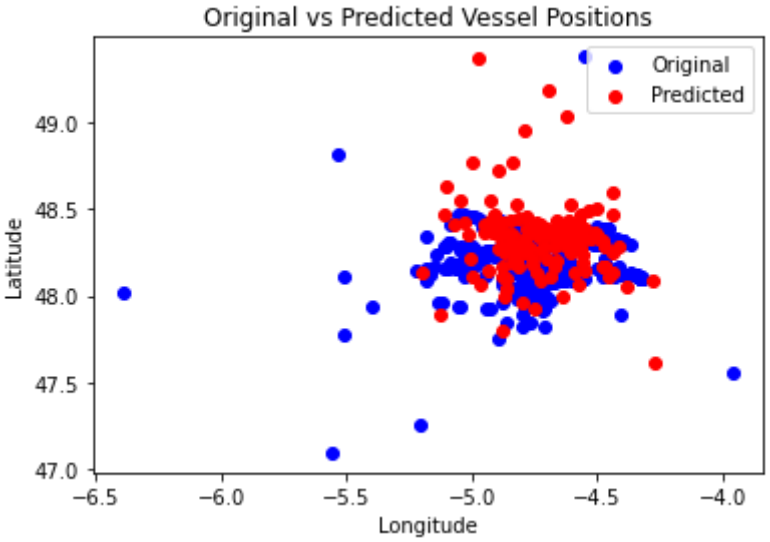


Figure 4.12 - Original vs Predicted Vessel Positions Scatter Diagram, Neural Network

Here we can see significantly less grouping of data, but more extreme outliers. There is also more spread of data around original values, indicating higher prediction accuracy when looking from a

visual perspective. The inclusion of outliers in the prediction markers are a major difference compared to the linear regression and random forest regressor models.

Finally, the standard deviation for this graph was:

- Longitude Standard Deviation: 0.20115755321493933
- Latitude Standard Deviation: 0.1888927834521122

These are higher values compared to the other two models, indicating a higher spread of the data.

4.3.2. MSE

Regarding the mean squared error, these are the values associated with each model:

Linear Regression:

- Longitude Model MSE: 0.03425
- Latitude Model MSE: 0.02506

Random Forest Regressor:

- Longitude Model MSE: 0.02493
- Latitude Model MSE: 0.02004

Neural Network:

- Longitude Model MSE: 0.04187
- Latitude Model MSE: 0.03568

It should be interpreted that the lower the MSE value, the higher the quality of the model. When comparing each model, the random forest regressor achieved the lowest MSE for both longitude and latitude models, indicating that this model made the best predictions from a MSE perspective.

By contrast, the neural network had the highest MSE values, this could be interpreted as being the lowest quality model from an MSE perspective. In another view, it could also be interpreted as the neural network model had more data variance compared to the other models.

4.3.3. RMSE

Regarding root mean square error scores, these are the values associated with each model:

Linear Regression:

- Longitude Model RMSE: 0.22645
- Latitude Model RMSE: 0.06521

Random Forest Regressor:

- Longitude Model RMSE: 0.10517
- Latitude Model RMSE: 0.03791

Neural Network:

- Longitude Model RMSE: 0.20463
- Latitude Model RMSE: 0.18890

As with the mean square error scores, a lower root mean square error score indicates a higher quality model. With this in mind, the random forest regressor model also achieved the lowest RMSE values, which could be interpreted as the model providing more accurate predictions.

By contrast, the linear regression model had the highest RMSE values, showing that there were more prediction errors compared to the other models, indicating lower quality.

4.3.4. MAE

Regarding mean absolute error scores, these are the values associated with each model:

Linear Regression:
- Longitude Model MAE: 0.13978
- Latitude Model MAE: 0.10005

Random Forest Regressor:
- Longitude Model MAE: 0.10811
- Latitude Model MAE: 0.08214

Neural Network:
- Longitude Model MAE: 0.13665
- Latitude Model MAE: 0.12563

Again, with MAE, a lower score indicates a higher quality model. The random forest regressor achieved the lowest MAE scores again, indicating that this model made the more accurate predictions. Linear regression had the worst longitude MAE metrics and the neural network had the worst latitude MAE metrics, indicating lower quality predictions for each respective part.

4.3.5. R2 Score

Regarding R-squared (R2) scores, these are the values associated with each model:

Linear Regression:
- Longitude Model R2 score: 0.11400
- Latitude Model R2 score: 0.02693

Random Forest Regressor:
- Longitude Model R2 score: 0.35508
- Latitude Model R2 score: 0.22179

Neural Network:
- Longitude Model R2 score: -0.08307
- Latitude Model R2 score: -0.38533

Interpreting R2 scores is fairly straightforward. A score close to 1 indicates the model is fitting well with the data, a score closer to 0 indicates that the model isn't good at explaining the data variance, a negative score indicates that the model is a poor fit for the data.

For our models, the random regressor model had the highest R2 scores overall, this could be interpreted as the model fits the data much better and is better at performing predictions on this data.

Linear regression had lower R2 scores, which means that while the data and model were fitting somewhat, there wasn't much explanation for the variance in data and indicates a lower prediction quality.

Finally, our neural network had negative R2 scores, this is a clear indicator that the model did not perform well at all and was not a good fit for the data, indicating lower prediction quality.

4.3.6. Residual Graph Interpretation

For regression models, interpretation of a linear plot can be just an important measure of indication that a model is providing a good fit for its associated data. It could be interpreted that a healthy residual plot would exhibit no clear pattern, whereas a bad one would have some kind of pattern such as a line, curve or wave. [21] Different patterns for a residual plot could also have different implications for how "healthy" the plot is and what kind of problems the plot is having such as heteroscedasticity or nonlinearity, which, when plotted, would take the appearance of an "explosion" from left to right and a curve, respectively. [22]

Linear Regression Longitude:

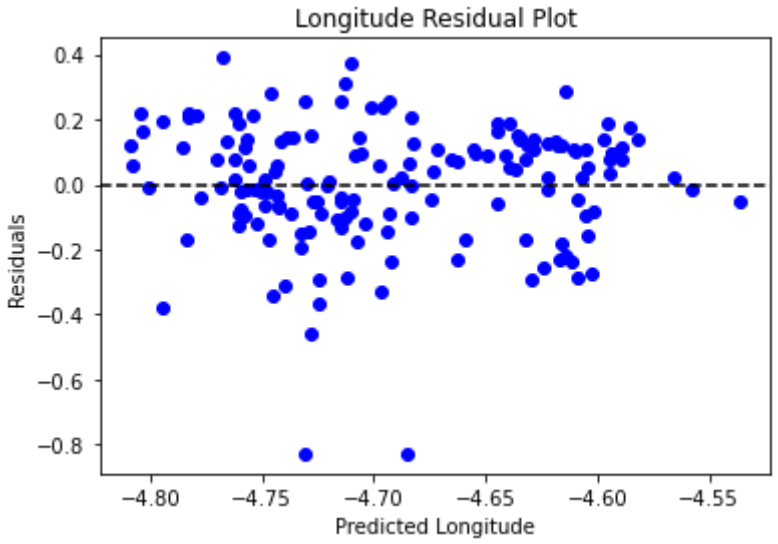


Figure 4.13 - Residual Graph, Linear Regression Longitude

Here we can see evidence of a somewhat healthy plot, with a healthy amount of symmetry which could indicate that the model is fitted well when it comes to predicting longitude.

Linear Regression Latitude:

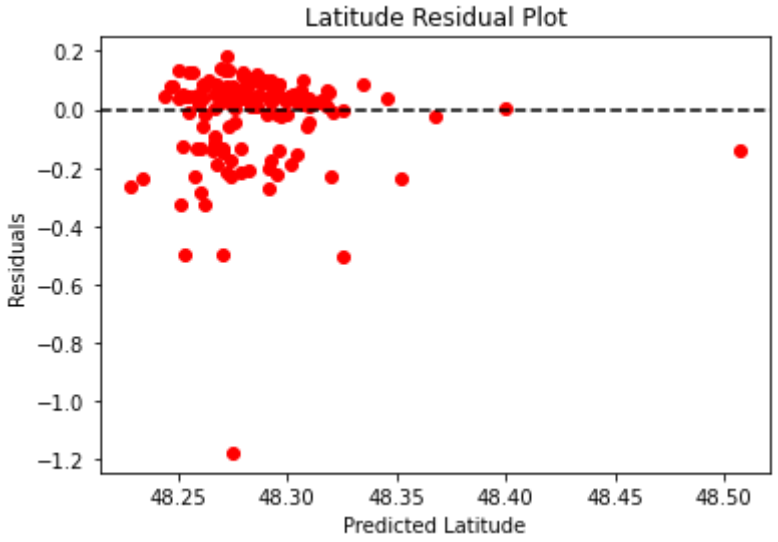


Figure 4.14 - Residual Graph, Linear Regression Latitude

Here there is evidence of heteroscedasticity, meaning that the residuals are getting larger. This is an indication that our model requires some kind of improvement or that the model is not a good fit for the data. Improvements could be things such as using a larger sample size or using other useful variables. There are also some extreme outliers which could be interpreted as issues with data entry (e.g. some of the data already contained outliers which could impact this graph).

Random Forest Regressor Longitude:

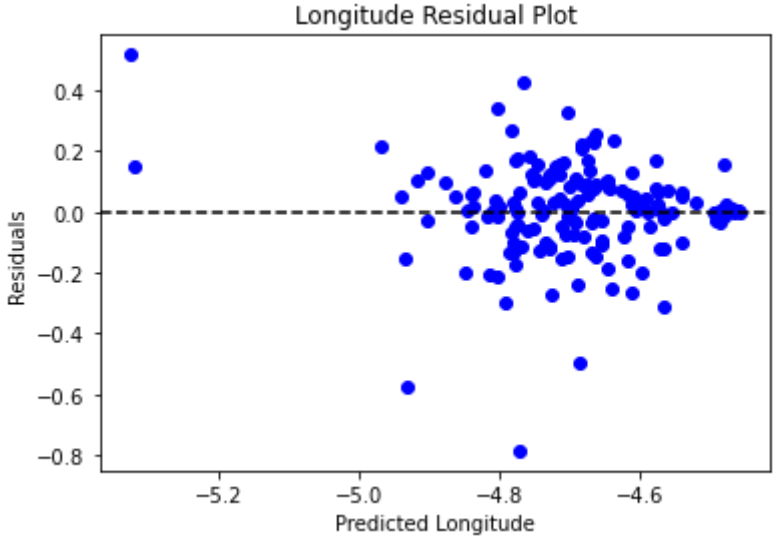


Figure 4.15 - Residual Graph, Random Forest Regressor Longitude

Here the model looks simply ok, there is a small amount of heteroscedasticity, but not as extreme as the last plot. This could imply that the model is fitting well, but with some issues that may require a larger sample size or the addition of another useful variable. There is also a good degree of symmetry to indicate that the model is doing well. Again, for this plot there are some outliers.

Random Forest Regressor Latitude:

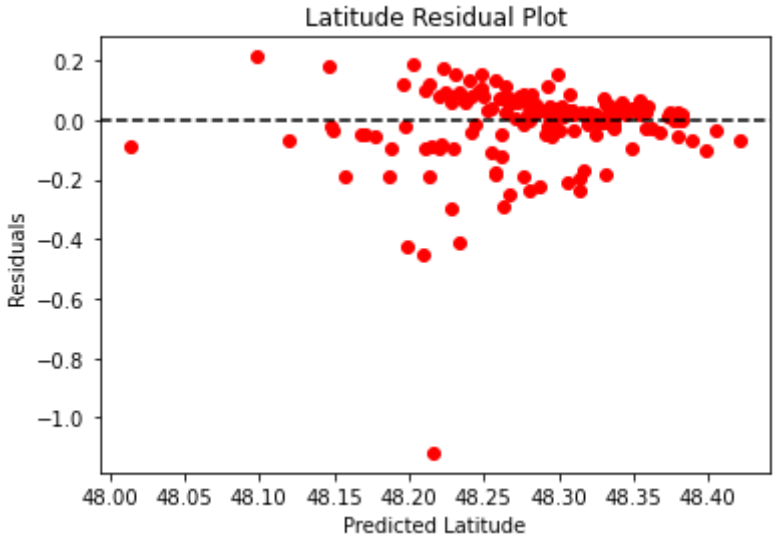


Figure 4.16 - Residual Graph, Random Forest Regressor Latitude

Some level of heteroscedasticity but not so much to indicate a completely unhealthy model. Only two extreme outliers. Otherwise, the graph is looking somewhat healthy with small heteroscedasticity but also with some ok symmetry.

Neural Network Longitude:

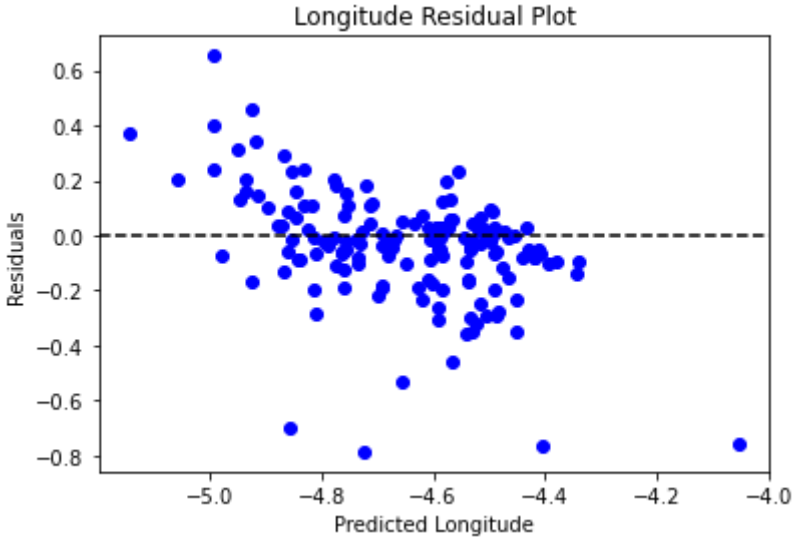


Figure 4.17 - Residual Graph, Neural Network Longitude

More heteroscedasticity and hints of non-linearity, indicating there could be some problems with the model, there are also some outliers. However, there is still some symmetry, implying the model is not extremely unhealthy and may just require some adjustments - larger sample sizes, more useful variables, changes to the model itself could also help such as changing the amount of layers, neurons or epochs.

Neural Network Latitude:

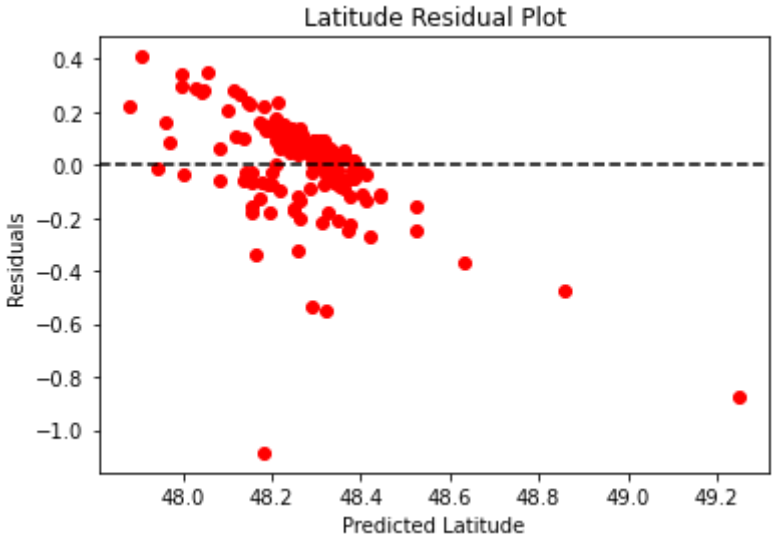


Figure 4.18 - Residual Graph, Neural Network Latitude

More extreme heteroscedasticity with some hints of nonlinearity as well as some extreme outliers, it could be implied that this model has not fit well with the data. No evidence for strong symmetry.

Overall, most of the graphs showed evidence of heteroscedasticity, indicating that all of the models could be having issues with fitting the data properly. All plots had outliers of varying levels of extremity.

5. Conclusion

5.1. Summarisation of Findings and Concluding Comments

Initially, I would like to discuss how well the models are fitting to the data. The implications of the residual plots indicates that most of the models struggled with fitting the data, due to the varying amounts of heteroscedasticity present in most of the model plots.

From a visual perspective after interpreting each scatter graph, including the addition of standard deviation, I would conclude that the neural network performs the best from the perspective of standard deviation and data spread. However, an argument could be made that the model did not perform well from this perspective due to issues in the implementation with the congregation of markers in the Bay of Biscay.

In terms of model evaluation metrics, including the MSE, RMSE, MAE and R2 score. The random forest regressor performed the best overall and therefore produced the highest quality predictions from this perspective. By contrast, the neural network produced the worst evaluation metrics overall, most notably with the production of a negative R2 score, indicating a poor model fit to the data.

Looking at the implementation itself, from a purely visual perspective, linear regression performed the worst. With the random forest regressor and neural network producing more “realistic” results.

In short, I would conclude that the random forest regressor produced the best quality predictions, supported by its model evaluation metric scores, reasonably healthy residual plots and from the visual implementation results.

5.2. Future Recommendations

The first recommendation I would like to give is the more obvious one, the inclusion of more models. Other regression models, deep learning, more complex machine learning models. This would give more insight into which model is “the best” or the better fit for this task, due to the amount of available models out there, there are bound to be ones that are a better fit than the ones implemented above.

In order to expand the model further, we could extend the scope of this project to cover human trajectories. Human trajectory prediction already exists, with multiple models, studies and open source code freely available for usage [9].

You could also implement more “visual” methods of predicted vehicle states from optical, radar or SAR based products. This could include:

- A classification based system which detects a vehicle from an image. This classification model could also detect the type of vehicle (e.g. car, boat). This would serve as a precursor to a vehicle state prediction model.
- A prediction model predicting variables such as longitude, latitude, speed, potential destinations, etc…

The implementation of a CNN to both detect vessels, cars, people, etc… then predicting their movement could be the next stage of development for this model and could be the most significant and realistic recommendation for future model development.

Going even further into an image classifier, you could incorporate image enhancement algorithms, in this case, with satellite products, to magnify and optimise images for higher readability and model accuracy (i.e. make the image “look better” so that the new model can identify more vehicles to a higher quality).

My final recommendation would be the addition of a comprehensive geofence added to the current model to eliminate longitude and latitude markers that appear on land, rather placing them in a location closest to the sea (relative to the predicted land position). This could be done with the addition of a GeoJSON file to simulate the geofence alongside an implementation to move the marker to a reasonable position.

6. Bibliography

[1] Margot, K.C. and Kettler, T. (2019). Teachers' perception of STEM integration and education: a systematic literature review. International Journal of STEM Education, [online] https://doi.org 6(1). Available at: https://doi.org/10.1186/s40594-018-0151-2. Last Accessed: 17/06/2023

[2] Li, X., Guvenc, L. and Aksun-Guvenc, B. (2023). Vehicle State Estimation and Prediction. [online] https://arxiv.org/. Available at: https://arxiv.org/abs/2304.11694. Last Accessed: 22/06/2023

[3] Hsu, L.-Y. and Chen, T.-L. (2008). Vehicle Full-State Estimation and Prediction System Using State Observers. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/4663951. Last Accessed: 22/06/2023

[4] Stuparu, D.-G., Ciobanu, R.-I. and Dobre, C. (2020). Vehicle Detection in Overhead Satellite Images Using a One-Stage Object Detection Model. [online] https://www.ncbi.nlm.nih.gov/. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7696426/ [Accessed 29 Jun. 2023].

[5] Young, Brain L., (2017). PREDICTING VESSEL TRAJECTORIES FROM AIS DATA USING R. [online] https://apps.dtic.mil/. Available at: https://apps.dtic.mil/sti/pdfs/AD1046595.pdf [Accessed 29 Jun. 2023].

[6] Zhang, R., Guo, Y., Long, Y., Zhou, Y. and Jiang, C. (2022). Vehicle Motion State Prediction Method Integrating Point Cloud Time Series Multiview Features and Multitarget Interactive Information. [online] https://www.hindawi.com/. Available at: https://www.hindawi.com/journals/jat/2022/4736623/ [Accessed 29 Jun. 2023].

[7] Yang, T., Wang, X., Yao, B., Li, J., Zhang, Y., He, Z. and Duan, W. (2016). Small Moving Vehicle Detection in a Satellite Video of an Urban Area. [online] https://www.ncbi.nlm.nih.gov/. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038801/ [Accessed 29 Jun. 2023].

[8] Do, H. (2019). Are Deep Learning models always the best? [online] https://towardsdatascience.com/. Available at: https://towardsdatascience.com/the-almighty-deep-learning-or-is-it-270d09ea2f27 [Accessed 08 Jul. 2023].

[9] Yue, J; Dinesh Manocha; Wang, H; Wang, H. (2023), Human Trajectory Prediction via Neural Social Physics. [online] https://arxiv.org/ Available at: https://arxiv.org/pdf/2207.10435v2.pdf [Accessed 11 Jul. 2023].

[10] Unknown (2023), Trajectory Prediction on ETH/UCY. [online] https://paperswithcode.com/. Available at: https://paperswithcode.com/sota/trajectory-prediction-on-ethucy [Accessed 11 Jul. 2023].

[11] Unknown (2021), "Locating Dark Ships in the Era of Space-based RF Geolocation," https://www.he360.com/, https://www.he360.com/resource/locating-dark-ships-in-the-era-of-space-based-rf-geolocation/ (accessed Jul. 16, 2023).

[12] U. Unknown, (Unknown) “NMEA-0183 messages: Overview,” Recieverhelp, Unknown. https://receiverhelp.trimble.com/alloy-gnss/en-us/NMEA-0183messages_MessageOverview.html (Last accessed Jul. 21 2023).

[13] Wikipedia Contributors, “Automatic identification system,” Wikipedia, Nov. 26, 2019. https://en.wikipedia.org/wiki/Automatic_Identification_System (accessed Jul. 23, 2023).

[14] C. Ray, R. Dréo, E. Camossi, A.-L. Jousselme, and C. Iphar, “Heterogeneous integrated dataset for Maritime Intelligence, surveillance, and reconnaissance,” Data in Brief, vol. 25, p. 104141, Aug. 2019, doi: https://doi.org/10.1016/j.dib.2019.104141. (accessed Jul. 23, 2023).

[15] C. Iphar, A.-L. Jousselme, and G. Pallotta, “Maritime route and vessel tracklet dataset for vessel-to-route association,” Data in Brief, vol. 44, p. 108513, Oct. 2022, doi: https://doi.org/10.1016/j.dib.2022.108513. (accessed Jul. 23, 2023).

[16] Wang L, Zhang L, Yi Z. “Trajectory Predictor by Using Recurrent Neural Networks in Visual Tracking" (2017). IEEE Explore. Retrieved from: https://ieeexplore.ieee.org/document/7935541 (accessed Aug. 05, 2023).

[17] Unknown. “sklearn.linear_model.LinearRegression”. (2007) scikit-learn.org. Retreived from: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (accessed Aug 07. 2023).

[18] Unknown. “sklearn.ensemble.RandomForestRegressor”. (2007) scikit-learn.org. Retreived from: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#:~:text=A%20random%20forest%20regressor.,accuracy%20and%20control%20over%2Dfitting (accessed Aug 07. 2023).

[19] Unknown. “1.17. Neural network models (supervised)”. (2014) scikit-learn.org. Retreived from: https://scikit-learn.org/stable/modules/neural_networks_supervised.html (accessed Aug 08. 2023).

[20] Unknown. “sklearn.preprocessing.RobustScaler”. (2007) skikit-learn.org. Retrieved from: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html (accessed Aug 18. 2023).

[21] “Zach”. “What is Considered a Good vs. Bad Residual Plot?”. (2023) statology.org Retrieved from: https://www.statology.org/good-vs-bad-residual-plot/ (accessed Aug 20. 2023).

[22] Unknown. “Interpreting Residual Plots to Improve Your Regression”. (2023) qualtrics.com Retreived From: https://www.qualtrics.com/support/stats-iq/analyses/regression-guides/interpreting-residual-plots-improve-regression/ (accessed Aug 20. 2023).