Development of Ship Performance models based on Artificial Neural Networks and Operational Data

Thalassinos Anastasiou

Diploma Thesis

School of Naval Architecture and Marine Engineering
National Technical University of Athens

Supervisor: Assistant Prof. Nicolaos Themelis

Committee member: Prof. Georgios Zaraphonitis
Committee member: Prof. Konstantinos Spyrou

March 2022

Acknowledgements

I would like to express my gratitude for the guidance and support provided by my supervisor, Assistant Professor Nicolaos Themelis, whose dedication, knowledge, and advice were vital in the implementation of the current diploma thesis. I would like to express my profound thanks to him for our excellent collaboration, his patience, and his insights.

Furthermore, I would like to acknowledge Prisma Electronics' contribution to this thesis. This study was built on the data provided by the company's LAROS system.

I should also express my gratitude to the incredible open source community of Python users and developers for providing numerous essential libraries, and especially to the entire machine learning community for simplifying this work.

Last but not least, I want to express my gratitude to my family, my friends and fellow students for their incredible support not only during the completion of this diploma thesis but also throughout my studies.

Abstract

In this diploma thesis, two distinct types of data-driven models were developed: an artificial neural network and a multiple linear regression model. Both models attempted to forecast the fuel oil consumption of a crude oil tanker based operational data collected by an onboard monitoring system over an 18-month period. To ensure the reliability of the dataset, pre-processing is performed, which includes the removal of outlier data points via the imposition of thresholds and statistical filtering. After implementing data preprocessing, emphasis is placed on developing the artificial neural network with state-of-the-art training and optimization techniques to achieve the lowest possible error with a high degree of generalization capability. Then, the multiple linear regression model is developed, and both models are evaluated by computing critical metrics on an unknown dataset and by demonstrating their ability to construct fuel oil consumption - speed curves under a variety of loading and weather conditions. In both circumstances, the artificial neural network outperforms the multiple linear regression model, which is due to the presence of non-linearities in the physics of the problem. Python programming language has been used to carry out all of the processes in this thesis.

Περίληψη

Στην παρούσα διπλωματική εργασία κατασκευάζονται δύο στατιστικά μοντέλα: ένα τεχνητό νευρωνικό δίκτυο και ένα μοντέλο πολλαπλής γραμμικής παλινδρόμησης. Ο στόχος και των δύο μοντέλων είναι η ακριβής πρόβλεψη της κατανάλωσης καυσίμου της κύριας μηχανής ενός δεξαμενόπλοιου η οποία βασίζεται σε υπηρεσιακά δεδομένα που λαμβάνονται από ένα σύστημα παρακολούθησης επί του σκάφους σε διάρκεια 18 μηνών. Προκειμένου να αποκτηθεί ένα αξιόπιστο σύνολο δεδομένων, πραγματοποιείται μια διαδικασία προεπεξεργασίας η οποία περιλαμβάνει την αφαίρεση των ακραίων τιμών μέσω επιβολής ελαχίστων τιμών και στατιστικού φιλτραρίσματος. Στην συνέχεια, δίνεται έμφαση στη ανάπτυξη και βελτιστοποίηση ενός τεχνητού νευρωνικού δικτύου το οποίο θα μπορεί να προβλέπει με τη μέγιστη δυνατή ακρίβεια την κατανάλωση καυσίμου. Ύστερα, αναπτύσσεται το μοντέλο πολλαπλής γραμμικής παλινδρόμησης, και τα δύο μοντέλα αξιολογούνται και συγκρίνονται σε ένα "άγνωστο" σετ δεδομένων καθώς και στην ικανότητά τους να παράγουν τις καμπύλες της κατανάλωσης καυσίμου - ταχύτητας για διαφορετικές καταστάσεις φόρτωσης και καιρικές συνθήκες. Και στις δύο περιπτώσεις το ΤΝΝ πετύχαινει καλύτερα αποτελέσματα σε σχέση με το ΠΓΠ, γεγονός που οφείλεται στην ύπαρξη μη γραμμικότητας στη φυσική του εν λόγω προβλήματος.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Framework

1.1.1 Increasing demand for shipping companies

Increasing regulatory requirements and business competition are putting pressure on shipping companies to improve ship efficiency and sustainability. Technological advancements in electronics and sensors, global satellite connectivity, and technologies like big data analytics. Al techniques and digital twins are all enabling technologies. In this regard, monitoring ship performance is critical in the Mediterranean region.

Greek maritime companies are increasingly forming departments to evaluate vessel performance and energy efficiency. The choice to create such a department is based on the size and kind of the fleet, the company's commercial and ownership profile, and its feet Management strategy. Various performance monitoring and commercial management support tools can be built based on the company's objectives.[2]

If all analytic and tool work (e.g., software development) is done in-house, a typical performance department could include 3 to 5 persons, including the manager. On average, the team comprises of one or two performance engineers, a business analyst to coordinate software specifications, and a performance analyst to assist with administrative and research responsibilities. If a third-party company is responsible for the software development or consultancy, two individuals, including the manager, are sufficient.

1.1.2 Ship performance's critical areas

Monitoring ship performance improves situational awareness, optimizes voyage efficiency, identifies maintenance triggers, and assists with industry best practices (e.g., ISO 50001) and regulatory compliance (e.g., EU MRV). The hull, propeller, and main and auxiliary engines are targeted. Identifying areas for improvement and quantifying fuel savings and benefits are required to evaluate any future action plan.

Diagnostic data analytics is commonly used to optimize maintenance of equipment assets or to schedule underwater hull inspections and cleanings. Within this framework, KPIs are commonly expressed in terms of speed loss, power gain, or excess fuel consumption. These KPIs require an extensive library of reference data (such as speed trials or shop testing) and/or advanced ship propulsion models.

Another significant application of prescriptive data analytics is improving voyage efficiency. In addition, the development of onboard technologies for monitoring and improving energy management is typical. Another example is the detection of changes in sailing parameters such as shaft revolutions per minute, engine power, critical temperature data, and speed signals from various measuring devices using statistical approaches (such as generalized likelihood ratio detectors). In addition to better planning and timely guidance for all concerned parties, these tools spot abnormal mechanical or operational behavior while internally rating vessels of the fleet based on their operational profile. Furthermore, a precise prediction of vessel performance and fuel consumption is a useful tool in negotiating contracts and managing commercial obligations.

1.1.3 Data acquisition

The availability of precise data is critical in monitoring and evaluating ship performance. For some ship types and in some circumstances - e.g., LNG vessels - built-in integrated data acquisition systems exist. Typically, a data collection system is required to receive and store data from multiple sensors and devices. The data transfer is not standard, and there are numerous solutions such as third-party servers and cloud service, local-to-central server connection or a private cloud, automated emailing of locally processed data and private intranet platforms.

Data is provided in raw, tabular, and online report formats, as well as static or dynamic dashboards. In principle, third-party providers stream data straight from the ship to their cloud servers, whereas in-house developed platforms may be preferable for real-time data transfer to the company's database.

While onboard servers are used for temporary data storage and transmission, a hybrid infrastructure combining local servers and cloud services is becoming more common. The latter is constantly evolving due to the requirement for system compatibility and cost reduction. In addition to ensuring compliance with internal and external systems, a universal database structure (primarily SQL) must be developed that is available to any current or future system requiring data access. Of course, any data management plan should consider cybersecurity for communication between the vessel and the office.

Except for high frequency automated data collection, shipping companies still collect operational data from ships in the form of crew reports (e.g. daily or monthly, where an enterprise resource planning system can be utilized), focusing on critical information for the inspection of multiple systems. Laboratory data on lubricant and fuel oil analysis can also be used. Integrating data from many sources, such as weather providers, fuel analysis, and even manual inputs, is crucial to creating a viable data pool for each performance analysis.

1.1.4 Analytics techniques and tools

Massive amounts of data are now available, but only analysis can show their full worth. Depending on the department's competence and the application, physics-based or data-driven models can be used. There is a trend to develop in-house performance models and tools, but there are many third-party services that can provide ready-to-use solutions with analytics dashboards and ship propulsion mod-

els. Some can discover both well-established industry players who have expanded their product offerings to include analytics and modeling as well as start-ups that specialize in Al and analytics dashboard development.

Other examples of such services include the building of customisable dashboards that provide insights through multiple KPIs on a third-party service provider's cloud-based platform. To the contrary, adopting standards such as ISO 19030 for assessing changes in hull and propeller performance is a beneficial step toward standardization. Nonetheless, each organization may have its own internal metric and process. Modern approaches and advanced analysis have not and probably cannot be simply standardized. Long term, finding a balance or following both pathways is critical. When sailing data confirms the physical link, physics-based models are chosen. But this isn't always possible. For trim optimization, data driven techniques are employed to map and connect data clusters.

Latest tendency is to use advanced statistical regression methods alongside machine learning algorithms. However, issues including ship-specific applicability, algorithm retraining, and their use as "black boxes" must be considered. Benchmarking service data against reference data obtained from design data relies on the correctness of the relevant corrective procedures and filters employed. With no other option, CFD models or operational data are used to create starting points.

1.1.5 Future challenges

Performance monitoring has steadily gained prominence in shipping companies' decision-making strategies. Performance department involvement and data-driven business case review are critical in all energy and performance decisions. This could be used in anti-fouling system selection and evaluation, energy-saving retrofitting, and creative project evaluation. Moreover, dry-dock effects including hull and propeller fouling progress is reported to top management to help commercial and technical decision-making.

Numerous obstacles remain in revealing the added value of the massive amount of available data and transforming the organization into a highly sophisticated and efficient data-driven profit "machine." Handling massive amounts of data that are only useful for a limited time depending on the evaluation and report is an issue that should be addressed soon. Strategies and methods should be refined to meet evolving needs. Regardless, we believe a new era of data-driven performance monitoring has arrived, with exploitable potential for the maritime industry.

1.2 Literature Review

Petersen et al. [16] used a publicly available dataset collected from onboard sensors of a ferry over two months to train two types of neural network models: immediate and predictive. The immediate is a feed-forward neural network that estimates an output y_n (e.g. the main engine's fuel consumption) based on the current input vector x_n and ignores the propulsion parameters' temporal nature. To estimate the difference in the target variable at the following time step X_{n+1}, the predictive model uses TDNN (Tap-Delayed Neural Networks), a type of recurrent neural network. It is also used to fit a probability distribution that will be introduced as noise to the model's final predictions. The immediate model attained a mean relative error of

1.50 % on the estimation of FOC, whereas their TDNN model had no similar metric estimated in this study.

Ahlgren and Thern [1] used machine learning to develop models that forecast ship fuel consumption without human intervention in the model parameter selection. Despite the need of pre-processing, they merely tried to validate fuel consumption data. Their best model had similar accuracy to other researchers (4.8E-4 mean squared error in normalized model output), but employed fewer features as inputs (at max. 4 inputs).

A. Zalachoris [18] used operational data to create data-driven models for predicting fuel oil consumption (FOC) in his diploma thesis. During the three-year monitoring of two sister vessels, the first had a Mewis duct. Excessive filtering of the measured physical quantities resulted in outlier data points being rejected. So a solid dataset was created for multiple linear regression models to estimate FOC. An analysis of best subsets was used to pick models for each vessel. Notably, all four utilized square of water speed, mean draft, and wind speed as predictors, leaving out trim and rudder angle.

In his study, P. Karagiannidis [10] developed data-driven models for ship propulsion and carefully explored the effect of data pre-processing. In his study, he used a large, automatically gathered data set with a high sampling frequency to train models that predict the needed shaft power or main engine fuel consumption for a container ship sailing under arbitrary conditions. The statistical evaluation and preprocessing of the data were highlighted, and two techniques were introduced for this purpose. Additionally, state-of-the-art training and optimization strategies for Feed-Forward Neural Networks (FNNs) were used. His findings indicate that a careful filtering and preparation stage can considerably improve the model's accuracy.

However, the efforts in data-driven modeling of ship propulsion are not restricted to neural networks. Aldous [3] discussed ship performance analysis in greater detail. It is a study that takes a more systematic approach to the problem of data uncertainty and pre-processing, two critical components of the data-driven modeling procedure that have not been adequately addressed in earlier works. The data from noon reports (NR) are compared to continuous monitoring (CM) data, and the primary disadvantages of the first are as follows: (a) lack of standardization, (b) missing observations and inaccuracies, and (c) the inherent characteristic of expressing ship performance in terms of FOC only, rather than the more appropriate term shaft power. One of the concluding reasons is that CM datasets reduce the uncertainty of the collected data and hence of the resulting models used to evaluate the performance of vessels. This reasoning is also supported in other research, such as Themelis et al [14], which compares the predictive capability of the NR and CM datasets.

1.3 Thesis Objective & Structure

Ship's main engine fuel oil consumption is a parameter that is affected by loading, operational and weather conditions. Its accurate forecasting has been the topic of numerous studies in the recent years because it results in increased bunker savings. Thus, the objective of this study is to develop a regression model (Artificial neural network) capable of accurately predicting fuel oil consumption under a variety of loading and weather conditions and its comparison with linear models to deter-

mine whether the ANN's non-linear nature results in increased predictive capability. Since all of these statistical models are data-driven, we require a trustworthy dataset maintained by an on-board monitoring system. Thus, as a first step in this thesis, we will discuss the measured physical quantities and their associated equipment, Chapter 2. In the following chapter, Chapter 3, we will discuss the data pre-processing procedure that is used to filter the data in such a way that it is suitable for an accurate regression analysis. In the next chapter, Chapter 4, we will introduce some new features derived from existing ones that will aid us in modeling the problem more accurately. Chapter 5 will cover the development and selection of the ANN model that is most valid for our study's objective. In Chapter 6, we will cover the implementation of a multiple linear regression model (MLR) and in Chapter 7 we will compare its performance to that of the ANN model. Both models will be tested by computing crucial metrics on a unseen dataset and by demonstrating their ability to construct fuel oil consumption - speed curves under a variety of loading and weather situations. Finally, Chapter 8 contains the work's findings and further thoughts on the issue.

Chapter 2

Data Collection

A significant amount of data from the operation of a Crude Oil Tanker was made accessible for the current study. Table 2.1 lists the ship's main particulars.

Ship Type	165K COT
Length Between Perpendiculars	264 (m)
Breadth (moulded)	50 (m)
Depth (moulded)	23.1 (m)
T_d/T_s (moulded)	16 (m)/ 17.15 (m)
Engine's NCR	16794 (kW)
Engine's rpm (NCR)	87.9 (RPM)

Table 2.1: Ship's main particulars

The aforementioned ship was provided with sensors that kept track of the relevant propulsion parameters, as well as the ship's loading condition and relative wind speed and direction. Table 2.2 lists all of the parameters obtainable from the ship's operation. The collection period was from October 1, 2018 to March 30, 2020, with a recording interval of one minute. The total number of data points at baseline was 778925.

Parameter	**Abbreviation**	**Units**
Speed Over Ground	SOG	(kn)
Speed Through Water	STW	(kn)
Heading	H	(degrees)
Wind Speed	WS	(m/s)
Wind Direction	WD	(degrees)
Mean Draft	TM	(m)
Trim	TRIM	(m)
Propeller Shaft Power	PSP	(kW)
Propeller's Shaft Revolutions per Minute	RPM	(rpm)
Main Engine's Fuel Oil Consumption	FOC	(t/day)

Table 2.2: Measured Parameters

Speed through water describes the vessel's displacement per time unit if no current exists, whereas Speed over ground is essentially the absolute value of the total velocity vector in the ship's path of travel. Propeller shaft power is the amount of

power delivered from the engine to the shaft. Trim is the difference between the aft and forward drafts, whereas Mean draft is the vessel's draft at midship. Wind speed and direction refer to the magnitude and direction of the wind vector relative to the ship, at the anemometer's height. Heading indicates the vessel's absolute course in degrees, i.e., its position in relation to the North. Finally, Propeller 's Shaft revolutions per minute and fuel oil consumption are relatively self-explanatory. The units of the latter (t/day) indicate the rate of consumption at the time of measurement, not the mass of fuel consumed in a day.

The variables of the study are thus clearly defined. However, it would be helpful at this point to take a look at how they are measured on board the ship to gain a better understanding of the overall procedure as well as the obstacles that will inevitably arise.

A ship's operation data is collected using two methods: Noon reports and a continuous monitoring system (CMS). For this study's vessel the latter was employed. It is essential to examine briefly the different types of sensors that are required for capturing the signal of the parameters of interest.

The different sensors that are utilized to capture the parameters listed in Table 2.2 are first mapped and categorized in Table 2.3, and then their purpose is analyzed.

Parameter	**Device**
SOG, H	Global Positioning System (GPS)
STW	Speed log
PSP, RPM	Shaft torque meter
TM, TRIM	Pressure sensor
WS, WD	Anemometer
FOC	Mass Flow meter

Table 2.3: Measuring Equipment

- **Global Positioning System (GPS)**
  The GPS retrieves information about the ship's location in global coordinates, i.e., longitude and latitude. The calculation of the ship's speed over ground (SOG) is done by arithmetically deriving its position. The operation of the system requires constant communication with a satellite system to determine the position of the ship. The accuracy is high, but can be affected by currents.

- **Speed Logs**
  For measuring the vessel's speed through water, two most common sensors are used:

  - Doppler log: An acoustic speed log based on the Doppler effect in which the wave lengths of moving objects appear to shift in relation to the observer. This shift can be converted to speed, thereby giving a very accurate result. The Dual Axis Doppler Speed Log utilizes the Doppler shifted returns from high frequency acoustic energy transmitted into water to provide precise speed data, distance travelled, and water depth below the transducer. The transmitted signal is scattered back from the sea bottom and/or scatters in the water mass. The system amplifies the received signals and processes them to determine the Doppler shift.

  - Electromagnetic log: The electromagnetic log works by generating a small alternating current in a transducer producing an electromagnetic field in the adjacent water. As the vessel moves through the water, the voltage proportional to the speed is generated at 90 deg to the direction of travel. This signal voltage is detected by the probes and transmitted to the master electronic unit where it is amplified and processed digitally before being passed to the speed and distance displays.

- **Shaft torque meter**
  The shaft torque meter is a piece of equipment the measures the torque and the rotational speed of the shaft, and multiplies them to estimate the transmitted power's value. The instrument consists of strain gauges, arranged on a ring and mounted directly on the shaft for the continuous monitoring and logging the aforementioned values. The basic principle of operation is that any deformations of the strain gauges are transferred into voltages deviation which determine the strain of the shaft.

- **Pressure sensor**
  The draft of the ship can be estimated by the hydrostatic pressure on the hulls bottom surface. Sensors that measure the pressure are placed on the outer surface of hull's bottom and can deduce the instantaneous draft of the hull at the position that they are installed. From the measurement of the draft on two different longitudinal positions of the hull, the ship's trim can be calculated.

- **Anemometer**
  The wind anemometer is a device that provides both, the relative speed and direction of the wind with respect to the ship's orientation. It consists of a helicoid propeller and a vane the measure the wind's speed and direction, respectively. The angular displacement of the vane helps estimate the wind's relative direction, while the rotational speed of the helicoid propeller helps estimate the wind speed.

- **Mass flow meter**
  When it comes to measuring the fuel consumption in a ship, the most reliable way is to do it via mass flow meters because they eliminate the need for converting the volumetric flow into a mass flow, according to the fuel's density estimations. They are also known as Coriolis mass flow meters. The reason is that the Coriolis acceleration induces oscillations to the tubes of the device, that depends on the mass flow in them. As a result, the magnitude and the frequency of these oscillations help determine the fuel mass flow through the tubes.

Chapter 3

Data pre-processing

The purpose of this chapter is to delve into the nature of ship's data through its visualization and analysis. Although the collection of a range of data from ship operations is a useful endeavor from a scientific, technological and business point of view, the real value of these data will not arise without effective pre-processing.

When conducting a scientific research, data are rarely used in the form in which they were collected. This is because in most cases raw data may contain erroneous measurements due to malfunction or failure of one or more sensors. Sometimes also, the sensor fails to capture a measurement and the value at the dataset appears as a NaN. Furthermore, the researcher may be only interested in the part of the parameter's range that characterizes the under-investigation phenomenon, rather than the whole range of possible values. For example, a value near zero at speed implies that the ship was at the port, so there is no need to take this measurement into account if the study's goal is to determine the relationship between speed and fuel consumption.Last but not least, it is essential to remove points that are laying quite far from the curve whose equation describes the relationship between the two quantities. Numerous parametric statistics, such as mean and correlation, as well as any statistic based on them, are highly sensitive to outliers. Outliers can cause havoc with one's analysis because the assumptions underlying standard statistical procedures or models, such as linear regression and Artificial Neural Networks, are also based on the parametric statistic. All these mentioned before are the process known as data pre-processing.

At this point, it is essential to note that the initial dataset contains 778925 data points, each linked with a unique timestamp. So, if a missing, illogical value or an outlier appears in a parameter, the whole data point is discarded from the dataset. Although, because their number is modest in comparison to the total number of recordings, this does not pose any threat to the study's precision.

Initially, after discarding the illogical measurements, such as negative rpm or impossibly high values for speed, mean draft, etc. parametric plots are displayed. Following that, there will be a discussion about the relationships between natural quantities that have been established through scientific theory. The rest of the chapter is dedicated to data filtering.

The first part of the filtering process aims to provide a general framework for the analysis by setting thresholds for some parameters. All measurements above and below the thresholds are discarded. These simple preliminary filters exclude values with no physical meaning, such as negative ship speeds or negative fuel consumption

from the new dataset. Furthermore, this procedure effectively omits data, such as low water speeds or low engine power measured during port procedures and cargo handling. Data points associated with port operations cannot accurately represent the relationships between physical quantities and therefore result in outlier regions in the graphs. By excluding the lower speed and engine-related values, the "in port" data are not included in the new dataset, limiting the bias.

The second stage of the filtering procedure seeks to minimize the dataset even more by finding and removing statistically identified outlier data points. The relationships between physical quantities are used for this purpose. Firstly, the correlations between the variables are calculated, leading to a better understanding of the degree of interdependence between them. Then, highly correlated variables are paired and filtered by a procedure that omits the values of one parameter called "primary", based on the outliers of the other, called "secondary". The process, which is described in detail in the relevant chapter, efficiently pairs highly correlated parameters and greatly reduces outlier data points, resulting in a final filtered dataset that establishes the benchmarks for accurate predictive modeling.

3.1 Data Mining

As a first step in this analysis, the point is to demonstrate the raw dataset as it was obtained from the CMS. However, the presence of missing values and illogical measurements makes it difficult to visualize the distribution of each parameter and also to observe their relationship via scatter plots. Table 3.1 provides a clear picture of how many missing values the dataset includes. Thus, the obvious first step should be to clear the dataset from these values.

Parameter	**Missing values**	**Missing values (%)**
STW	0	0.000
SOG	186152	23.898
H	31	0.003
WD	77	0.009
WS	662	0.084
TM	5106	0.655
TRIM	3362	0.431
PSP	328	0.042
RPM	13338	1.712
FOC	211022	27.091

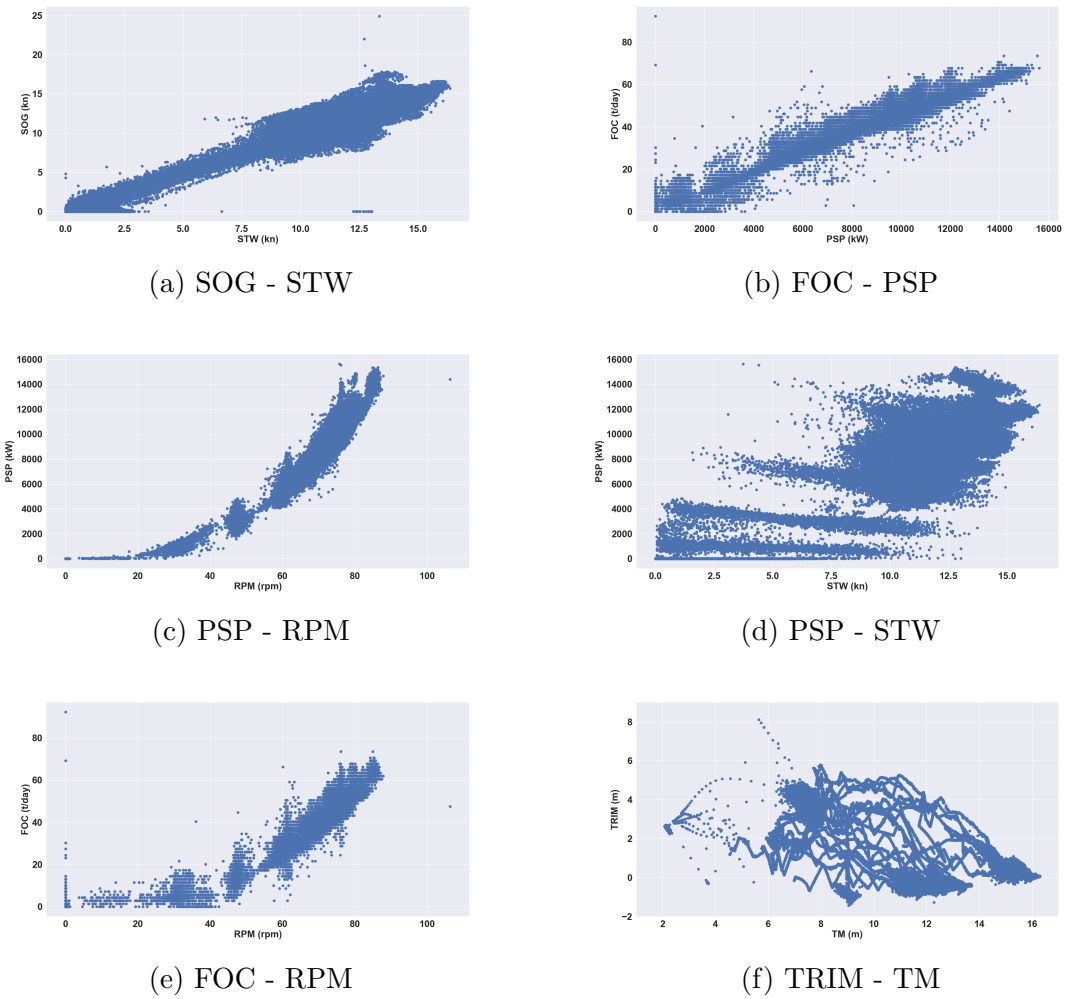
Table 3.1: Missing Values

The dataset currently contains 567903 data points out of 778925 after all missing values were excluded. Since the dataset is free of missing values the majority of the illogical values have been also excluded from the dataset. It had been observed that when a sensor fails to capture a signal, the remaining sensors will almost certainly capture data that do not correspond to reality, which is why illogical measurements were eliminated simultaneously with missing values.

Given that we now have a clearer picture of the dataset, we can better observe the relationship between the dataset's parameters and determine the course of processing for each, presented in Figure 3.1. For some of them (e.g., FOC - PSP), the

relationship, or at least the general trend, is already known thanks to our knowledge of the physics of the problem and will be helpful in removing outliers.

For convenience, we will henceforth refer to the dataset purged of missing values as **Dataset 1.1**.



(a) SOG - STW

(b) FOC - PSP

(c) PSP - RPM

(d) PSP - STW

(e) FOC - RPM

(f) TRIM - TM

Figure 3.1: **Dataset 1.1** Scatter Plots

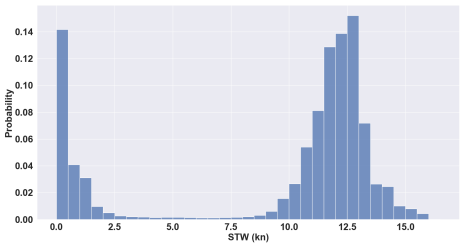
Figure 3.2 also provides histograms allowing for a more precise assessment of each parameter's distribution.

The SOG - STW graph, Figure 3.1a, illustrates the linear relationship between these two speed variables (over ground and through water). However, certain points deviate from the graph's main diagonal, resulting in the following outlier regions, listed in Table 3.2, the rest are due to sea current effect.

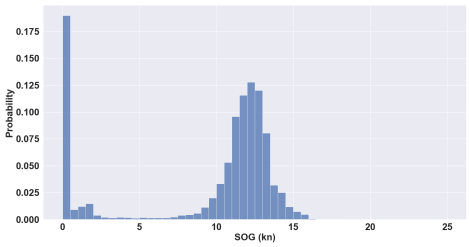
SOG (kn)	**STW (kn)**
[0,1]	[2,13]
[17,25]	[12.5,14]

Table 3.2: SOG - STW Outlier regions

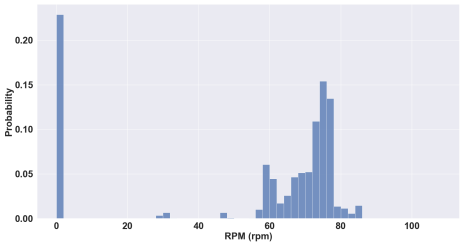
The first outlier region is probably the result of sensor malfunction, since it is very unlikely that SOG is so greatly reduced compared to STW.



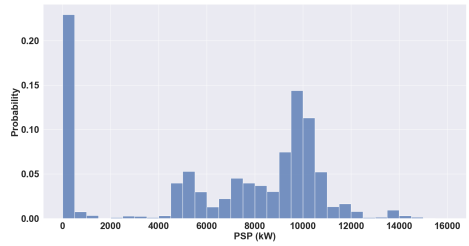
(a) STW



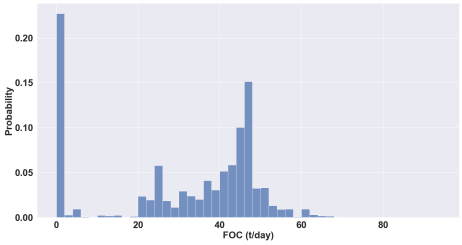
(b) SOG



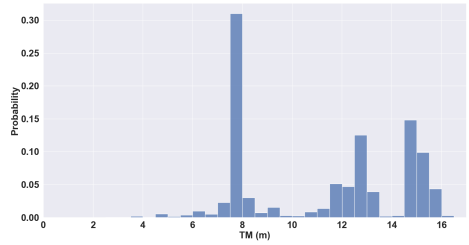
(c) RPM



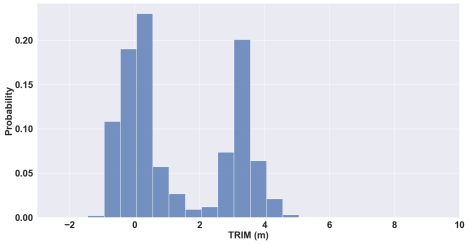
(d) PSP



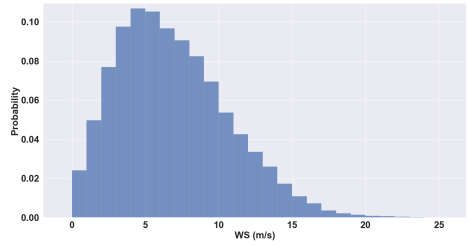
(e) FOC



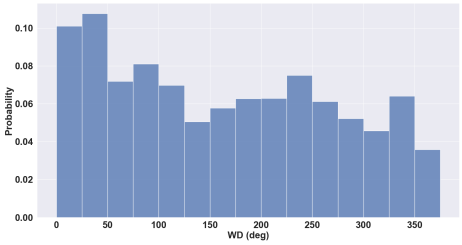
(f) TM



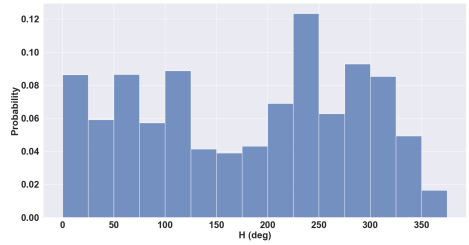
(g) TRIM



(h) WS



(i) WD



(j) H

Figure 3.2: **Dataset 1.1** Parameter's Histograms

The straight line in the FOC - PSP graph, Figure 3.1b, was anticipated, since thermodynamics states that the power transmitted to the shaft is a product of fuel consumption, reduced calorific value, and efficiency. In this case, outliers are the points that deviate from the graph's main diagonal and those where $FOC = 0$.

The PSP – STW curve, Figure 3.1d, denotes the relationship between power and speed. More precisely, is a compilation of power-speed curves for various hull and weather conditions, as well as mean draft and trim. However, the four horizontal regions of data points, as reported in Table 3.3, should be regarded as outlier regions since they clearly do not fit the expected curve.

PSP (kW)	**STW (kn)**
[0,2500]	[0,13]
[2500,5000]	[0.3,14]
[6000,9000]	[1.5,7]
[9000,16000]	[3,7]

Table 3.3: PSP - STW Outlier regions

Both PSP - RPM, Figure 3.1c, and FOC - RPM, Figure 3.1e, graphs are extremely similar, which is due to the fact that PSP and FOC have a strong linear connection, as shown in the FOC-PSP graph, Figure 3.1b. PSP and FOC are linked to RPM via a relationship that expresses the propeller law:

$$P = c \times n^a, a \in [3, 4] \tag{3.1}$$

However, FOC - RPM, Figure 3.1e, include a few outlier data points, which are mentioned in Table 3.4.

FOC (t/day)	**RPM (rpm)**
[0,100]	[0,5]
[8,20]	[55,65]
[35,55]	[55,65]

Table 3.4: FOC - RPM Outlier regions

Finally, there is no particular curve in the TRIM-TM graph, Figure 3.1f, but rather discrete areas of concentrated points representing various operational conditions. The regions where TRIM (m) $\in (1, 5.5)$, TM (m) $\in (6, 8.5)$ and TRIM (m) $\in (-1.5, 0.5)$, TM (m) $\in (8.5, 9.5)$ are representing the ballast conditions, and the regions where TRIM (m) $\in (-1, 1)$, TM (m) $\in (10.5, 14)$ and TRIM (m) $\in (-0.5, 2)$, TM (m) $\in (14, 16.5)$ are representing the laden conditions. The remaining points indicate transitional conditions, in-port operation, or inaccurate measurements.

3.2 Data Filtering

The purpose of this chapter is to process the dataset that is free of missing and illogical values, **Dataset 1.1**, by applying certain filters based on the student's knowledge of the dataset and the connections that exist between the investigated parameters. More specifically, the goal is to generate a dataset that corresponds to open sea conditions by eliminating the undesirable points from the existing collection of data.

As the objective of this thesis is to forecast fuel consumption, conditions faced by the vessel in or near port are meaningless. Thus, data that do not correspond to open sea conditions i.e., when the vessel has reached a speed below 7 (knots) are eliminated by applying threshold values to the parameters.

3.2.1 Threshold Values

3.2.1.1 Speed Thresholds

The ship's speed is one of the most significant - if not the most significant - operating parameter. It is a critical factor in controlling fuel consumption and a core issue for the shipowner. As previously mentioned in the relevant chapter, speed is monitored through the SOG and STW variables, which should both get non-negative values since the sensors compute the absolute values of the speed rather than its vectors. Additionally, low water speed values are connected with the vessel's entry into the port, operating inside it, or depart from it. These instances fall outside the scope of this study, as fuel oil consumption is either low or zero, as well as the horizontal regions of data points in the PSP-STW graph, Figure 3.1d. As a result, the following speed limits were applied:

1. SOG (kn) $\in (7, 18)$
2. STW > 7 (kn)

3.2.1.2 Power, RPM and Fuel Oil Consumption Thresholds

Propeller shaft power (PSP), revolutions per minute (RPM), and fuel oil consumption (FOC) of the provided set of parameters are directly related with the main engine. As a result, they are highly correlated, as seen by their respective graphs. However, the complete data set does not adequately reflect the overall under investigation sail condition. The following threshold values were applied in order to exclude points associated with port operation:

1. PSP > 4000 (kW)
2. RPM (rpm) $\in (50, 90)$
3. FOC > 8 (t/day)

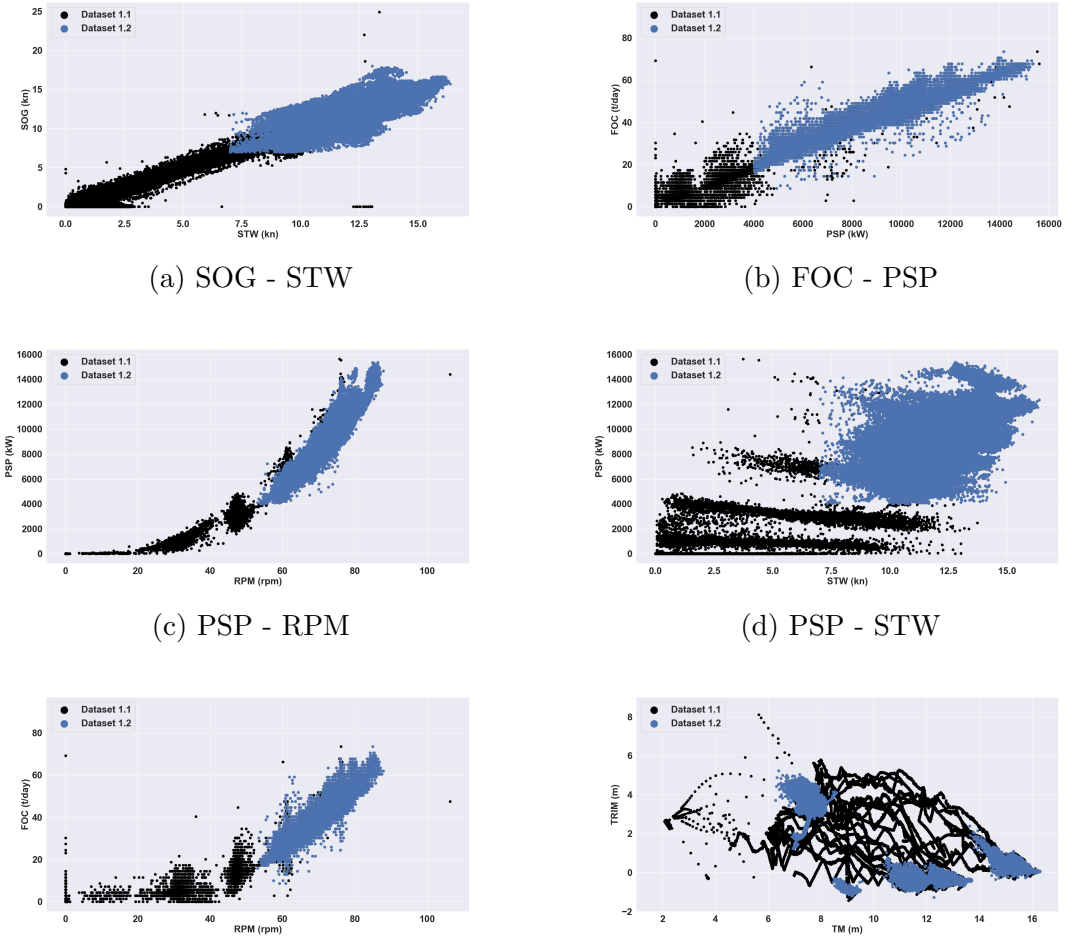
3.2.1.3 Mean Draft Thresholds

While mean draft (TM) measurements range from zero to scantling draft, it is clear that low values were recorded throughout cargo loading and unloading. While the ship sails on the open seas, TM cannot be less than the designed ballast condition draft. According to the TRIM-TM graph, Figure 3.1f, the stated value appears to be around 6 meters, which is why the following filter was applied:

1. TM > 6 (m)

3.2.1.4 Threshold filters application and graph comparison

By applying the previously discussed filters in **Dataset 1.1**, the dataset is partially cleaned, as outliers (e.g., port-related) or incorrect data are excluded. The following scatter plots, Figure 3.3, illustrate the consequent improvement. The black points represent measurements that do not satisfy the threshold values, whereas the blue points represent those that do. For convenience, we will henceforth refer to the cleaned dataset as **Dataset 1.2**.



(a) SOG - STW

(b) FOC - PSP

(c) PSP - RPM

(d) PSP - STW

(e) FOC - RPM

(f) TRIM - TM

Figure 3.3: **Dataset 1.1** vs **Dataset 1.2** Scatter Plots

In SOG - STW graph, Figure 3.3a, a sufficient number of outliers was identified. This is reasonable, as the minimum values for both SOG and STW were 7 knots. We needed to apply these filters in order to cope with the three horizontal lines shown in Figure 3.1d, PSP - STW graph. As a result, the linearity between STW and SOG is reduced, however in the following chapter, we will strengthen their linear relationship by employing statistical filters.

In FOC - PSP graph, Figure 3.3b, the applied filters slightly strengthened the linear relationship between PSP and FOC, as the majority of outliers were located below the applied filters values. However, points, above the applied filters, that deviating from the main diagonal slightly removed.

In FOC - RPM graph, Figure 3.3e, the use of the filters was successful as the

vertical region were RPM (rpm) $\in (45, 55)$, FOC (t/day) $\in (0, 35)$ in the FOC - RPM graph, Figure 3.1e, was removed.

Regarding the PSP - STW graph, Figure 3.3d, the vast majority of deviating measurements have been eliminated, and the graph thus resembles a cloud of data points including multiple power - speed curves under various hull and weather conditions, as well as mean draft and trim values.

The TRIM - TM graph, Figure 3.3f, is fundamentally different from the previous one. Apart from points having a draft of less than 6 (m), the majority of dispersed points in the graph's center have been deleted, leaving four main concentration regions and scattered points. The first two TRIM $>$ 1 (m), TM (m) $\in (6, 8.5)$ and TRIM $<$ 0 (m), TM (m) $\in (8.5, 9.5)$ are probably associated with the ballast condition while the other two TRIM $<$ 1 (m), TM (m) $\in (10.5, 13.8)$ and TRIM $<$ 2 (m), TM (m) $\in (13.8, 16.4)$ correspond to different laden conditions. Thus, the filters served their duty, as the loading conditions of the vessel can be recognized clearly.

3.3 Statistical Outlier Detection

At this stage, **Dataset 1.2** includes entirely measurements related to open sea conditions. The purpose of this chapter is to further process **Dataset 1.2** by exploiting the relationships that characterize the physical quantities. As an initial step, the correlations between the parameters are calculated and utilized in order to evaluate their interdependence. After calculating the correlation between all parameters, the goal is to exclude statistical outliers from the dataset using a Statistical filtering procedure explained in the relevant chapter. Finally, after evaluating the efficiency of each filter on the dataset through scatter plots, the ideal combination of filters is chosen and applied to the data points, resulting in a final dataset suitable for predictive modeling.

The reason we need to remove statistical outliers is that they cause the regression line to rotate and shift away from the general trend of the data. The further away the outlier is, the greater the effect on the line, until the line no longer represents the data's general trend. Outliers can have a significant effect on statistical results. As a result, such outliers are typically omitted in order to conduct a more robust analysis.[17]

3.3.1 Pearson Correlation Coefficients

The Pearson correlation coefficient is a measure of the linear dependence between two random variables (real-valued vectors), in our case the parameters of **Dataset 1.2** are the vectors. It was the first formal measure of correlation and remains one of the most widely used.

The Pearson correlation coefficient between two variables X and Y is defined formally as the covariance of the two variables divided by the product of their standard deviations (which acts as a normalization factor), and it can also be defined equivalently as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.2}$$

where $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ denotes the mean of x and $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ denotes the mean of y. The coefficient r_{xy} ranges from -1 to 1 and it is invariant to linear transformations of either variables.

The PCC gives an indication on the strength of the linear relationship between the two random variables x and y. The sign of the correlation coefficient is positive if the variables are directly related and negative if they are inversely related. If $r_{xy} = 0$ then x and y are said to be uncorrelated. The closer the value of $|r_{xy}|$ is to 1, the stronger the measures closeness to a linear relationship. This is because the association measure reflects the tendency of changes for each pair of corresponding expression levels in the two profiles.

The Pearson correlation coefficient measures the similarity of the changes in the expression levels of two profiles. Specifically it measures the strength of the linear relationship between two profiles.

If a particle's observation path is close to the true state's observation path, its path will be close to the true state's path when the observation noise is small. Thus, the PCC can be used to select particles in order to ensure that they are close to their true state.[19]

The **Dataset 1.2** computed coefficients are summarized and shown in Figure 3.4.

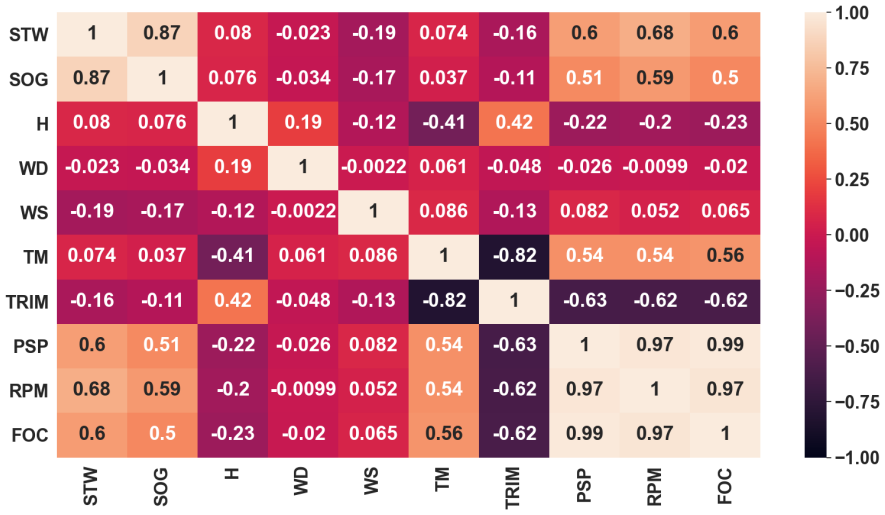


Figure 3.4: **Dataset 1.2** Correlation Heatmap based on Pearson Correlation Coefficients

As it can be seen from Figure 3.4, the majority of the pairs that share a large linear correlation have also been discussed in previous sub-chapters. The high correlated pairs are going to be examined in the following filtering procedure in order to determine if they efficiently eliminate the statistical outlier regions of the graphs without omitting useful set of data.

After thorough consideration, the following pairs are chosen for examination:

1. SOG - STW
2. PSP - RPM

3. FOC - PSP
4. FOC - RPM

3.3.2 Statistical Filtering Procedure

The following filtering procedure that we developed for detecting and rejecting outlying data points is described in detail here:

1. Select a primary parameter X whose values are to be filtered.
2. Split the primary parameter X in groups of values with range v.
3. Select a secondary parameter Y which is highly correlated with the primary parameter X.
4. For each group G_i of X, normalize Y with z-score normalization.
5. Select an outlier threshold k, where $k \in [2, 3.5]$.
6. For every respective value of Y in the G_i group, Y_{ij}, test if the following inequality is fulfilled:
$$|Y_{ij}| \leq k \tag{3.3}$$
7. If the inequality is not fulfilled, reject the data point.

The selection of the values of k and v has been discussed in [10] and [18]. To demonstrate the filters' effectiveness, scatter plots between the primary and secondary parameters were created, with the rejected points highlighted in red. Additionally, in order to determine which combinations of k and v are optimal for each filter, we experimented with multiple combinations and re-evaluated their effectiveness using scatter plots. The scatter plots that were used for evaluation were SOG – STW, PSP – RPM, FOC – PSP and FOC – RPM which led us to four pairs of variables and four filters. The relative analysis and graphs are provided in the following paragraphs.

3.3.2.1 Filter 1: SOG – STW

The SOG and STW parameters are filtered according to the previously described filtering procedure. Table 3.5 contains the outlier threshold k and the range v values.

Filter's Abbreviation	Filter 1
Primary Parameter	STW
Secondary Parameter	SOG
k	2.5
v **(kn)**	0.5

Table 3.5: SOG - STW Filter details

As stated earlier, the effectiveness of the filter is going to be assessed through SOG – STW scatter plot, Figure 3.5, where blue points represent **Dataset 1.2** and red points represent the statistical outliers detected by **Filter 1**.

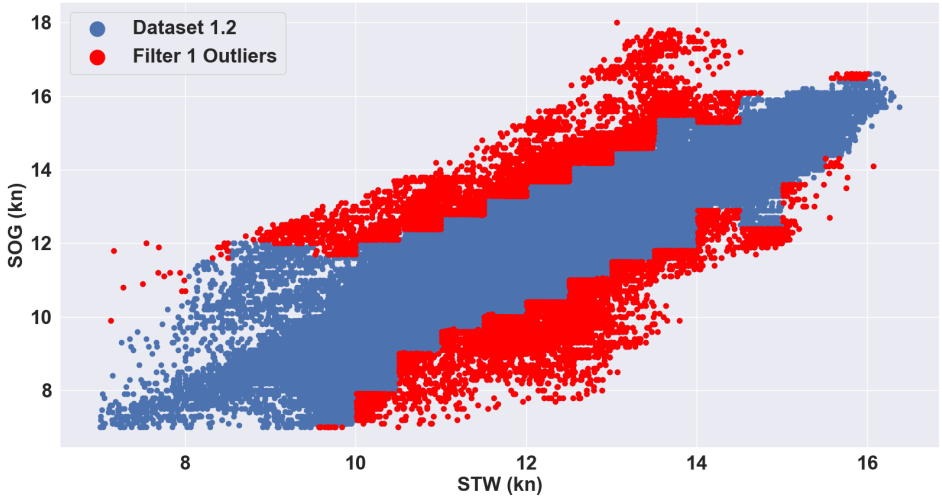


Figure 3.5: **Filter 1** Application to SOG - STW scatter plot

As shown in Figure 3.5, **Filter 1** correctly detects the majority of the outlying points. More precisely, measurements that deviate from the main diagonal where SOG (kn) $\in (11, 18)$, STW (kn) $\in (8.5, 15)$ and SOG (kn) $\in (7, 14)$, STW (kn) $\in (8.5, 16)$ are effectively detected, and their removal will enhance SOG - STW linear correlation. However, **Filter 1** fails to detect the outlier region where SOG (kn) $\in (10, 12)$, STW (kn) $\in (8, 9.8)$. Taking all of these into consideration, **Filter 1** has an overall beneficial impact.

3.3.2.2 Filter 2: PSP – RPM

The PSP and RPM parameters are filtered according to the previously described filtering procedure. Table 3.6 contains the outlier threshold k and the range v values.

Filter's Abbreviation	Filter 2
Primary Parameter	RPM
Secondary Parameter	PSP
k	2.5
v **(rpm)**	2.5

Table 3.6: PSP - RPM Filter details

As stated earlier, the effectiveness of the filter is going to be assessed through PSP – RPM scatter plot, Figure 3.6, where blue points represent **Dataset 1.2** and red points represent the statistical outliers detected by **Filter 2**.

The purpose of **Filter 2** was to identify outlier data points that "live" outside the graph's trendline, as the PSP – RPM scatter plot adequately expressed the propeller law after threshold values were applied. **Filter 2** detected these points correctly, but rejected some valid measurements where a ladder region is formed due to a lack of data in the sample. Taking all of these factors into account, **Filter 2** filter has an overall beneficial impact.

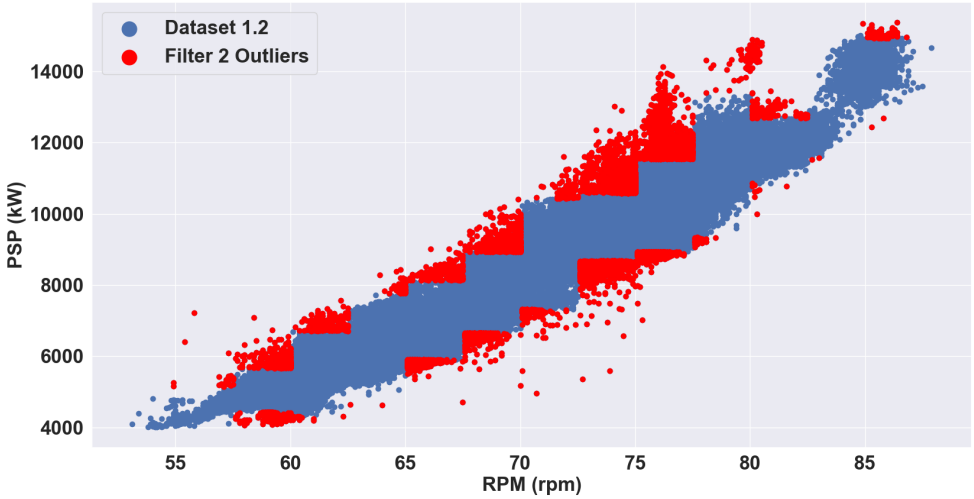


Figure 3.6: **Filter 2** Application to PSP - RPM scatter plot

3.3.2.3 Filter 3: FOC – PSP

The FOC and PSP parameters are filtered according to the previously described filtering procedure. Table 3.7 contains the outlier threshold k and the range v values.

Filter's Abbreviation	Filter 3
Primary Parameter	PSP
Secondary Parameter	FOC
k	2.5
v **(kW)**	100

Table 3.7: FOC - PSP Filter details

As stated earlier, the effectiveness of the filter is going to be assessed through FOC – PSP scatter plot, Figure 3.7, where blue points represent **Dataset 1.2** and red points represent the statistical outliers detected by **Filter 3**.

Before the application of **Filter 3**, there were many points that deviated from the graph's main diagonal, as discussed previously. After the application of the threshold values the relationship between PSP and FOC slightly improved. As illustrated in Figure 3.7, **Filter 3** has a significant impact, as it detects all data points that deviate from the graph's main diagonal, providing us with graph that perfectly represents the physical relationship between these parameters.

3.3.2.4 Filter 4: FOC – RPM

The FOC and RPM parameters are filtered according to the previously described filtering procedure. Table 3.8 contains the outlier threshold k and the range v values.

As stated earlier, the effectiveness of the filter is going to be assessed through FOC – RPM scatter plot, Figure 3.8, where blue points represent **Dataset 1.2** and red points represent the statistical outliers detected by **Filter 4**.

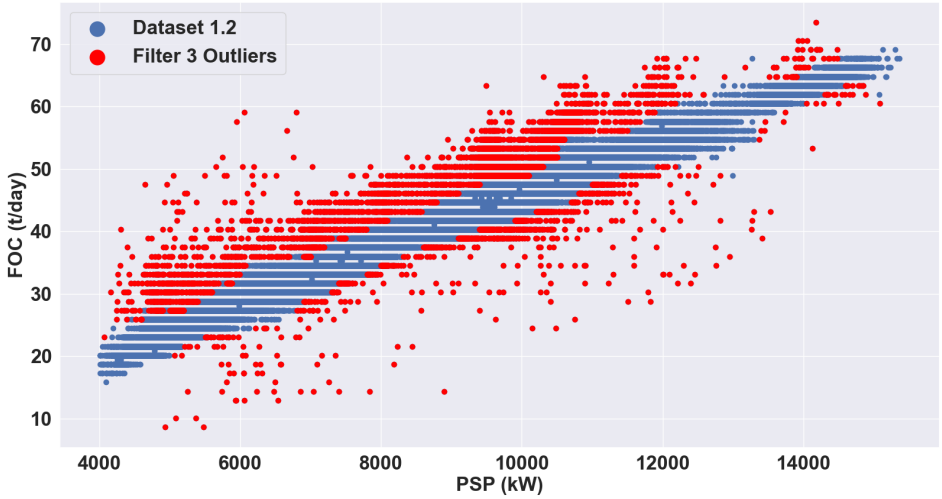


Figure 3.7: **Filter 3** Application to FOC - PSP scatter plot

Filter's Abbreviation	Filter 4
Primary Parameter	RPM
Secondary Parameter	FOC
k	2.5
v **(rpm)**	2.5

Table 3.8: FOC - RPM Filter details

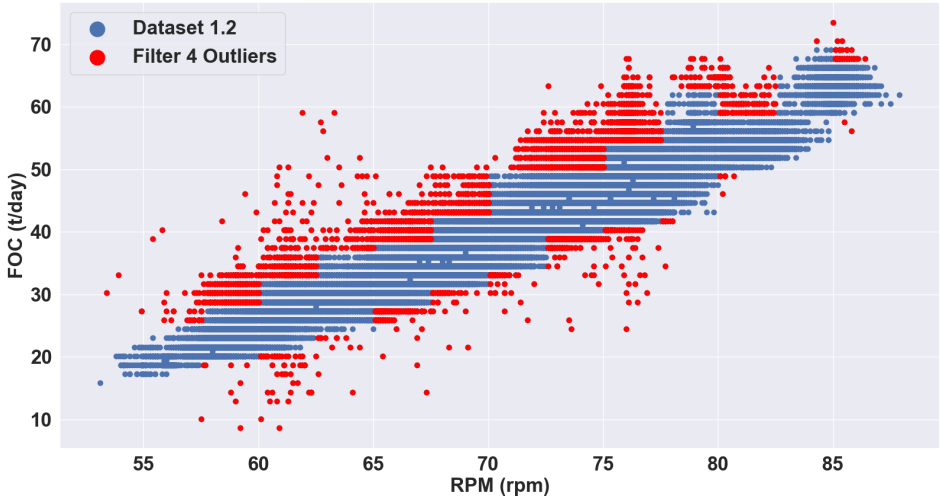
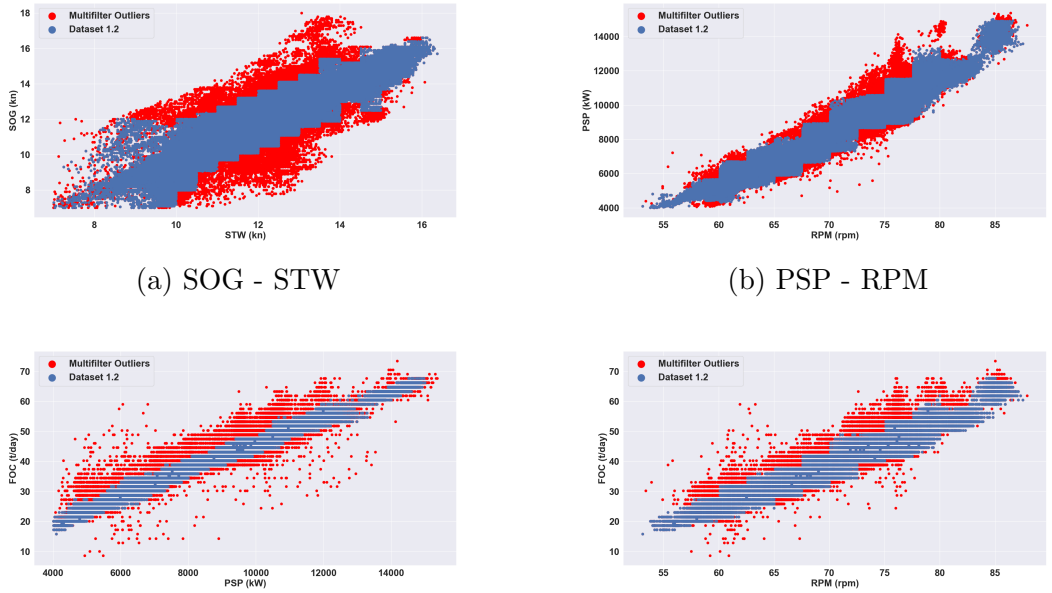


Figure 3.8: **Filter 4** Application to FOC - RPM scatter plot

The purpose of **Filter 4** was to identify outlier data points that "live" outside the graph's trendline and appear as vertical regions. **Filter 4** detected these points correctly, but rejected some valid measurements where a ladder region is formed due to a lack of data in the sample. Taking all of these factors into account, **Filter 4** filter has an overall beneficial impact.

3.3.2.5 Multifilter

Given that all of the individual filters were reviewed and their contribution to the dataset's correction appeared to be beneficial, the dataset's solid correction will be performed by combining these four filters, abbreviated as **Multifilter**. **Multifilter's** application to **Dataset 1.2** will result in a Final Dataset, abbreviated as **Final Dataset**, that accurately depicts the relationships between physical quantities. In the graph below, Figure 3.9, scatter plots are displayed where red points represent the outliers found by the **Multifilter** and blue points represent **Dataset 1.2**.



(a) SOG - STW

(b) PSP - RPM

(c) FOC - PSP

(d) FOC - RPM

Figure 3.9: **Multifilter's** application to **Dataset 1.2** scatter plots

3.4 Dataset's Review

The purpose of this chapter is to illustrate the effect of preprocessing on the dataset. Firstly, let us envision the process's structure using the flowchart below, Figure 3.10.

Prior to delving into the datasets statistical values, it's important noting how each preprocessing procedure affected the datasets size. Firstly, **Raw data** had 778925 data points; after missing and illogical values were extracted, **Dataset 1.1** was formed with 567903 data points, equivalent approximately 72.9 % of **Raw data**. Then, after the application of the Threshold values at the **Dataset 1.1**, **Dataset 1.2** was formed with 425504 data points, equivalent approximately 54.62 % of **Raw data**. Finally, the application of statistical filtering to **Dataset 1.2** resulted in the **Final Dataset**, which comprises 394753 data points, or about 50.67 % of the **Raw data**. Regarding the fact that the final dataset is about half the size of the initial dataset, this fact poses no threat, as the rejected data points offers no value to the following regression analysis.

As a further step, we will examine the influence of preprocessing on the statistical values of the dataset's parameters. Table 3.9 has extensive information regarding

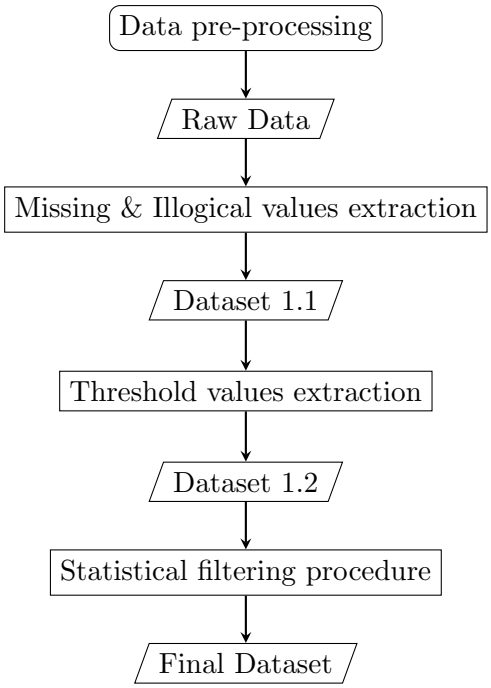


Figure 3.10: Data pre-processing structure

the parameters statistical values.

	mean	**std**	**min**	**25%**	**50%**	**75%**	**max**
STW (kn)	12.19	1.12	7.01	11.51	12.20	12.85	16.38
SOG (kn)	12.06	1.24	7.00	11.30	12.10	12.80	16.60
H (deg)	174.61	106.88	0.00	71.70	187.50	276.10	359.90
WD (deg)	162.10	109.25	0.00	60.88	156.28	251.43	360.00
WS (m/s)	7.43	3.64	0.00	4.64	7.02	9.73	26.36
TM (m)	11.58	3.09	6.28	7.92	12.44	14.82	16.20
TRIM (m)	1.27	1.70	-1.20	-0.07	0.39	3.32	5.03
PSP (kW)	8731.37	2090.71	4008	7213	9479	10139	15003
RPM (rpm)	71.17	6.48	53.10	67.20	73.20	75.90	87.50
FOC (t/day)	40.50	9.39	15.84	34.56	43.20	46.08	67.68

Table 3.9: Descriptive statistics for **Final Dataset**

In Table 3.9 the first column refers to the mean value, the second to the standard deviation, the third and the seventh to the minimum and maximum contained value, respectively. The three in-between columns are the first, second (median value) and third quartiles, respectively.

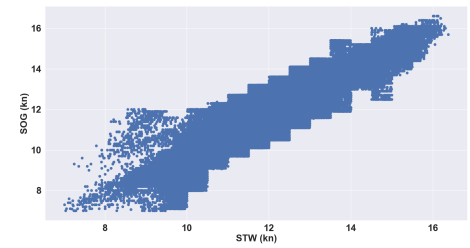
A quartile is a statistical term that describes a division of observations into four defined intervals based on the values of the data and how they compare to the entire set of observations. A quartile divides data into three points—the lower quartile, the median, and the upper quartile—creating four distinct groups. The lower quartile, or first quartile, is denoted as Q1 and is the value that falls between the dataset's smallest and median values. The second quartile, Q2, is also the median. The upper or third quartile, denoted as Q3, is the central point that lies between the median and the highest number of the distribution.[13]

Each quartile contains 25% of the total number of observations. Generally, the

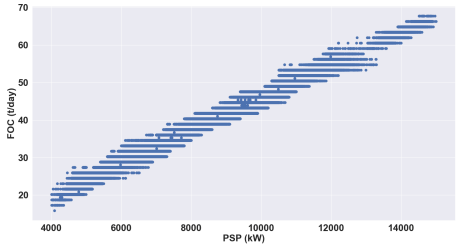
data is arranged in ascending order:

1. First quartile: the lowest 25% of numbers
2. Second quartile: between 25.1% and 50% (up to the median)
3. Third quartile: 50.1% to 75% (above the median)
4. Fourth quartile: the highest 25% of numbers

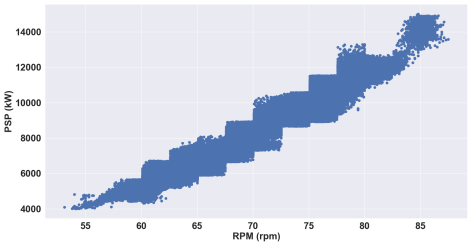
Finally, we will depict the relationships between the several parameters included in the **Final Dataset** using scatter plots, as illustrated in Figure 3.11.



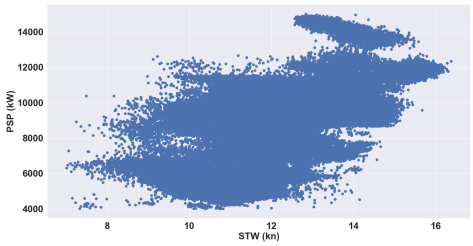
(a) SOG - STW



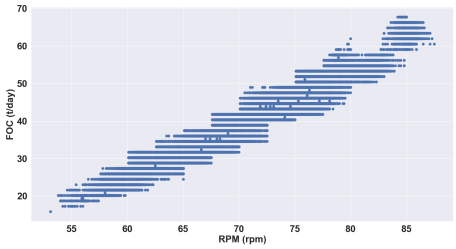
(b) FOC - PSP



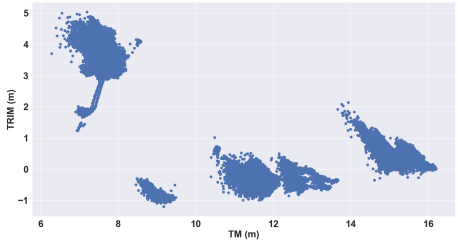
(c) PSP - RPM



(d) PSP - STW



(e) FOC - RPM



(f) TRIM - TM

Figure 3.11: **Final Dataset** Scatter Plots

Chapter 4

Feature Engineering

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. Although feature engineering is an informal topic, it is widely recognized and accepted as critical to success in practical machine learning. Given that the objective of this thesis is the development of predictive models, feature engineering is critical.

Feature engineering involves the following steps:

1. Feature Generation: This procedure aims to generate more relevant features from the dataset's parameters in order to improve the predictive model's accuracy or to minimize the model's error.

2. Feature Selection: This procedure aims to select the final subset of features that are most useful to the problem.

This chapter demonstrates how to apply this two-step technique. The primary objective is to select the features that will result in the most efficient model for the ship's fuel oil consumption prediction.

4.1 Feature Generation

Feature generation is the process of deriving new features from one or multiple existing features, potentially for use in statistical analysis. This process adds new information to be accessible during the model construction and therefore hopefully result in a more accurate model.

As a first step, let us provide the **Final Dataset's** accessible features (or parameters) in the following table, Table 4.1.We decided to generate two new weather-related parameters from the existing ones. The first feature that was generated was the "sea current", abbreviated as SC, which is simply the result of the subtraction of the STW from the SOG:

$$SC = SOG - STW \tag{4.1}$$

The second one is slightly more complicated, so let's go over the stages involved in its creation:

1. Convert Wind Direction (WD) units from degrees to rad.

2. Apply the trigonometric function cosine to Wind Direction.
3. Multiply the cosine of Wind Direction to Wind Speed (WS) in order to get the new feature named "Wind Effect", abbreviated as WE:

$$WE = WS \cos WD \tag{4.2}$$

Parameter	Units
SOG	(kn)
STW	(kn)
H	(deg)
WS	(m/s)
WD	(deg)
TM	(m)
TRIM	(m)
PSP	(kW)
RPM	(rpm)
FOC	(t/day)

Table 4.1: **Final Dataset's** parameters

Wind Effect denotes the effect of the wind in the ship's resistance in the longitudinal direction and when is negative, it is tailwind; when it is positive, it is headwind.

Finally, a total of 12 features were created from the initial 10, as shown in Table 4.2. However, each of these 12 features should be evaluated for their suitability for inclusion in a machine learning model, as some may be co-linear with others or redundant, as discussed in the next section.

Parameter	Units
SOG	(kn)
STW	(kn)
H	(deg)
WS	(m/s)
WD	(deg)
TM	(m)
TRIM	(m)
PSP	(kW)
RPM	(rpm)
FOC	(t/day)
SC	(kn)
WE	(m/s)

Table 4.2: Total set of available parameters

4.2 Feature Selection

The dataset contained 12 features at the summary of the Feature Generation. However, features whose attributes are irrelevant to the problem should not be included

in the model. Some features will be more critical than others to the model's accuracy. There will also be features that will be redundant in the context of other features. Feature selection tackles these issues by automatically selecting the subset of features that are most useful to the problem.

Table 4.2 features will be evaluated in terms of their potential contribution to the performance of the models. As this study's objective is to develop models that forecast Fuel Oil Consumption, at this section it will be determined which of the 11 features will be used as inputs in the model in order to accurately predict Fuel Oil Consumption values.

In order to choose which features to use as inputs or predictors in our models, we must consider both the statistical evidence for the examined features and the model's application. Thus, the linear correlation coefficient (Pearson's correlation coefficient) for the selected features of the model is estimated, only with respect to the target variable (Fuel Oil Consumption) in order to gives us a first insight. The results are shown in Figure 4.1.

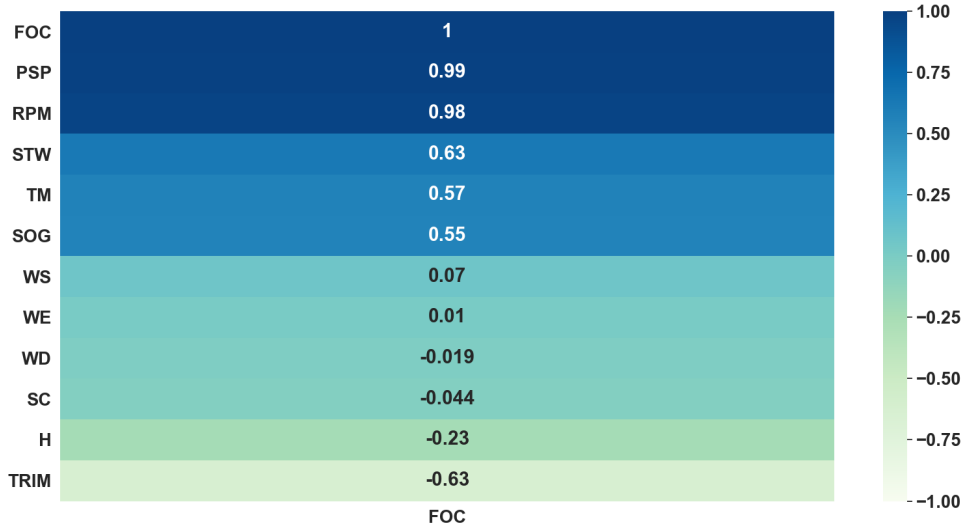


Figure 4.1: Pearson Correlation Coefficients values between input features and target variables.

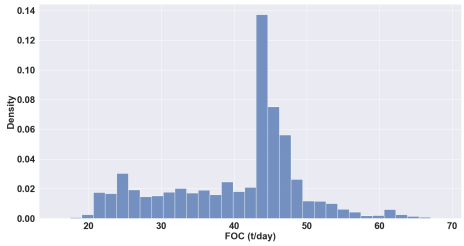
If we use highly correlated features (with the target variable) as inputs, the predictive power of the models will most likely increased. However, it is not worrying if the coefficient value is almost zero since it only declares that there is no linear correlation. Other than that, the application of the model determines the combination of features that will be used as inputs. Given that the model's application is to predict FOC for various weather and loading conditions, we are only interested in those features that contribute to this application's success. Thus, the features that we will be used as inputs in our models are listed in the following table, Table 4.3.

TM and TRIM represent the loading parameters, SC and WE represent the weather parameters. Between STW and SOG we decided to include STW as the speed parameter.The rest of the features were omitted because they did not contribute to the application's goal.

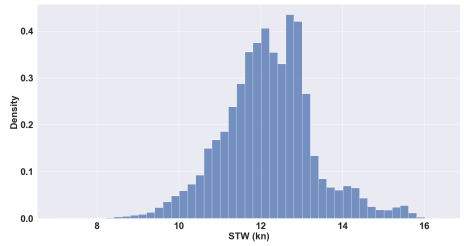
Parameter	Units
STW	(kn)
TM	(m)
TRIM	(m)
SC	(kn)
WE	(m/s)

Table 4.3: Dataset's parameters that will be used as inputs in the models

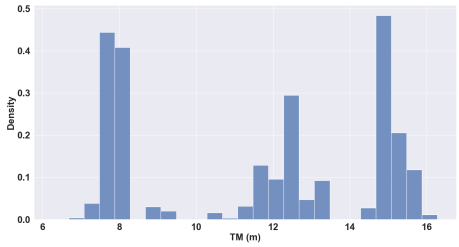
Finally, Figure 4.2 provides histograms allowing for a more precise assessment of each feature's distribution.



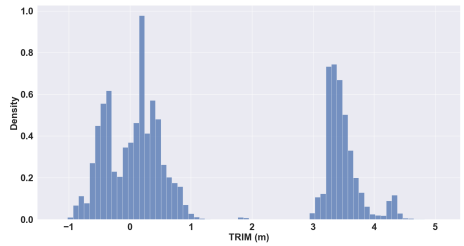
(a) FOC



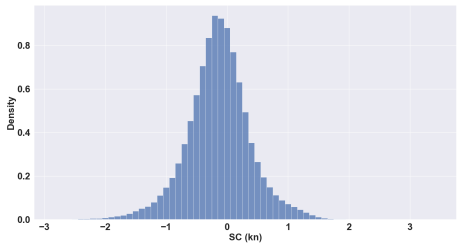
(b) STW



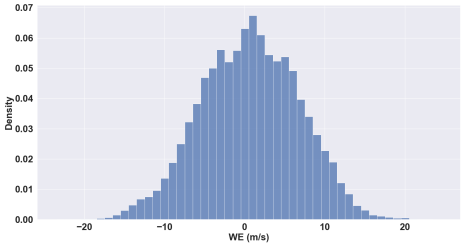
(c) TM



(d) TRIM



(e) SC



(f) WE

Figure 4.2: Model's inputs - output histograms

Chapter 5

Artificial Neural Networks: Model Design & Selection

This chapter describes the process of developing a neural network, including the selection of its fundamental structure and hyperparameters. We propose that readers unfamiliar with the fundamentals in the field of Artificial Neural Networks first read Appendix A. It aims to familiarize readers with the essential principles of Artificial Neural Networks (ANNs) and to demonstrate their mathematical formulation in order to facilitate discussion of the architecture and parameters of ANNs. All of these details have been customized for each model and are based on the predicted data's quality and quantity. As a result, all significant factors will be defined and discussed in detail in this chapter.

5.1 Model Design

When we refer to model design, we are referring to a series of precise stages that are carried out before the training procedure. It is necessary to determine which features or parameters will be used as inputs and which as outputs as a first step, but this topic was covered in the previous chapter.

5.1.1 Data Scaling

Deep learning neural networks learn how to map inputs to outputs from examples in a training dataset. The weights of the model are initialized to small random values and updated via an optimization algorithm in response to estimates of error on the training dataset. Given the use of small weights in the model and the use of error between predictions and expected values, the scale of inputs used to train the model are an important factor.

We decided to transform the features by scaling each them to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. Thus, all the data are normalized within the range $[0, 1]$, according to the following equation:

$$\mathbf{X}_{scaled} = \frac{\mathbf{X} - min\left(\mathbf{X}\right)}{max\left(\mathbf{X}\right) - min\left(\mathbf{X}\right)} \tag{5.1}$$

5.1.2 Data Shuffling

Shuffling the data is critical to avoiding model learning that is biased. If we consider all of the ship dataset's measured variables to be random variables, we may be confident that they are not static. As a result, data collected in the future will contain additional information based on actual changes to our physical system, the ship. If the data are not shuffled, the model will be trained on hull conditions that correspond to a specific time frame of the ship's operation, and examples pertaining to the system's subsequent behavior will be ignored.

5.1.3 Data Split

The data must be partitioned into two sets: training data for training the model, and testing data for providing an unbiased evaluation of the final model fit based on the training dataset. The testing dataset is used to evaluate the performance of the neural network once it has been thoroughly trained. Additionally, a subset of the training data, referred to as the Validation dataset, is used to evaluate the fit of a model to the training data set while tuning model hyperparameters. The validation dataset is different from the test dataset that is also held back from the training of the model, but is instead used to give an unbiased estimate of the skill of the final tuned model when comparing or selecting between final models.

In the following designed neural networks, the Training dataset is chosen as the 80% of the dataset, the Test dataset is chosen as the 20% of the dataset and the Validation dataset is chosen as the 20% of the Training dataset, Figure 5.1.

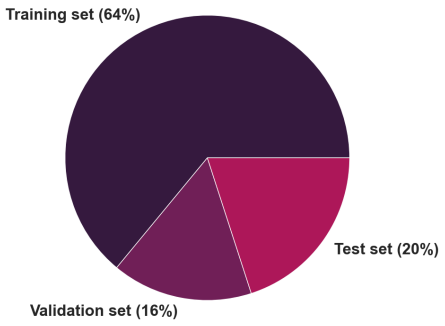


Figure 5.1: Training, Validation & Testing data sets

5.1.4 Fine-Tuning of Model Hyperparameters

This section will present and explain the basic hyperparameters of the designed neural networks. It's worth noting that the model design technique is governed by a plethora of guidelines and empirical instructions. However, there is no "golden rule" for selecting a hyperparameter; it is greatly dependent on the designer's perspective and the outcomes of "Trial-and-Error". The latter term refers to repeated efforts at fitting the model, each time selecting a different set of hyperparameters in order to gain an overview of the most efficient ones.

As can be appreciated in Figure 5.2, the current practice in the selection of design hyperparameters for ANN is based on the trial and error procedure, where a large number of ANN models are developed and compared to one another. If the level of a design hyperparameter is changed and does not have effect in the performance of the net, then a different design hyperparameter is varied, and the experiment is repeated in a series of approaches. The observed answers are examined in each phase, to determine the best level in each design hyperparameter.[15]

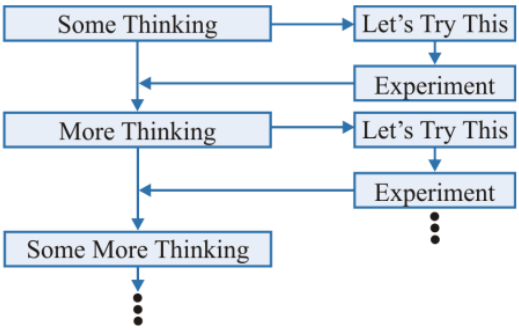


Figure 5.2: Trial-and-error procedure in the selection of ANN Hyperparameters

5.1.4.1 Number of Hidden Layers

For a large number of problems, a model with a single hidden layer can produce acceptable results. A multilayer perceptron (MLP) - a type of feed-forward ANN - with a single hidden layer can theoretically model even the most complex functions if it has enough units.

However, for complex problems, deep networks outperform shallow networks in terms of parameter efficiency. In other words, they can model complex functions with exponentially fewer units than shallow nets, which enables them to achieve significantly higher performance with the same amount of training data.

Real-world data is frequently structured hierarchically, and deep neural networks take advantage of this fact automatically: lower hidden layers model low-level structures (e.g. line segments of various shapes and orientations), intermediate hidden layers model intermediate-level structures (e.g. squares, circles), and the highest hidden layers and output layer model high-level structures (e.g. faces). Not only does this hierarchical architecture aid Deep Neural Networks (DNNs) in convergent to a good solution more quickly, but it also enhances their ability to generalize to new, unknown datasets that are entirely different to the input datasets.[7]

The methodology we used was to add layers until the error stabilized, then add another, and so on until the error improved. If there was no improvement, we kept the fewest possible layers that achieved this performance.

5.1.4.2 Number of Units per Hidden Layer

As for the hidden layers, it used to be common to size them to form a pyramid, with fewer and fewer units at each layer. However, this practice has been largely abandoned because it seems that using the same number of units in all hidden layers performs just as well in most cases, or even better; plus, there is only one

hyperparameter to tune, instead of one per layer. That said, depending on the dataset, it can sometimes help to make the first hidden layer bigger than the others. It should be taken into consideration that, if a layer has too few units, it will not have enough representational power to preserve all the useful information from the inputs and some of them may be lost.[7]

The methodology we used was to add units until the error stabilized, then add even more, and so on until the error improved. If there was no improvement, we kept the fewest possible units per layer that achieved this performance.

5.1.4.3 Activation Function

In a neural network, an activation function specifies how the weighted sum of the input is transformed into an output from a unit or units within a layer. Many activation functions are nonlinear and may be referred to as the "nonlinearity" in the layer or the network design. The choice of activation function has a large impact on the capability and performance of the neural network.

Technically, the activation function is used within or after the internal processing of each unit in the network, although networks are designed to use the same activation function for all units in a layer. A network may have three types of layers: input layers that take raw input from the domain, hidden layers that take input from another layer and pass output to another layer, and output layers that make a prediction. All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model. In this thesis all models used the linear activation as the activation function of the output layer.

Typically, a differentiable nonlinear activation function is used in the hidden layers of a neural network. This allows the model to learn more complex functions than a network trained using a linear activation function.

There are possibly three activation functions worth considering for use in hidden layers; they are as follows:

- Rectified Linear Activation (**ReLU**)
- Logistic (**Sigmoid**)
- Hyperbolic Tangent (**Tanh**)

The rectified linear activation function, or **ReLU** activation function, Figure 5.3, is perhaps the most common function used for hidden layers. It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as **Sigmoid** and **Tanh**. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained.[8]

Linearity means that the slope does not plateau, or saturate, when the input gets large. ReLU issue where all the negative values becoming zero decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately. Having said that, if ReLU is going to be used as activation function

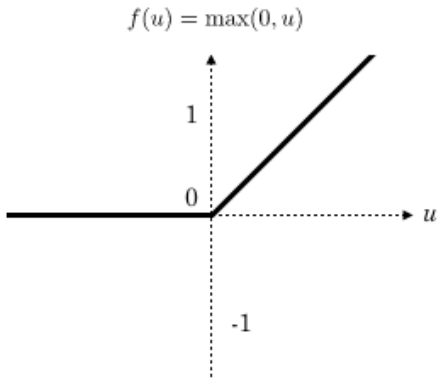


Figure 5.3: **ReLU** Function

this may involve standardizing variables to have a zero mean and unit variance or normalizing each value to the scale 0-to-1.

Furthermore, prior to training a neural network, it's necessary to initialize the network's weights to small random values. When using ReLU in a network and initializing weights to small random values centered on zero, then by default half of the units in the network will output a zero value. Thus, when using the ReLU function for hidden layers, it is a good practice to use a "He Normal" or "He Uniform" weight initialization prior to training.

5.1.4.4 Optimizer

Optimizers are used in machine learning to tune the parameters of a neural network so that the cost function is minimized. Thus, the optimizer selection is critical, as it can make the difference between a good and a poor training.

Optimizers are divided into two families: gradient descent optimizers (e.g. SGD) and adaptive optimizers (e.g. Adam, RMSprop). This division is entirely operational in nature, requiring manual tuning of the learning rate in the case of Gradient Descent algorithms, whereas it is automatically adapted in the case of adaptive algorithms — hence the name. However, gradient descent optimizers are subjected to the choice of a good learning rate, where unfortunately, this choice is not straighforward. A solution to this problem is to slowly decrease the learning rate value in order to make the updates smaller and smaller, so avoiding high oscillations. Furthermore, there is a possibility of getting stuck into a suboptimal local minima. Adaptive optimizers have been introduced to solve the issues of the gradient descent's algorithms. Their most important feature is that they don't require a tuning of the learning rate value. However, Gradient descent optimizers like SGD with the usage of a learning rate decay schedule might outperform Adaptive optimizers.

5.1.4.5 Numbers of Epochs & Batch size

The number of epochs is a hyperparameter that specifies how many times the learning algorithm will pass the training dataset. One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches.

The batch size is a number of samples processed before the model is updated. The size of a batch must be more than or equal to one and less than or equal to

the number of samples in the training dataset while a large batch size can assist the model to fit better on noisy data. The number of epochs can be set to an integer value between one and infinity. However, it should be noted that a large number of epochs is usually responsible for overfitting the model. Both are integer numbers and both are hyperparameters for the learning algorithm, i.e. parameters for the learning process, rather than internal model parameters discovered during the learning process.

There are no hard-and-fast rules for configuring these parameters. We must experiment with various values to see which one works best for our situation.

5.1.4.6 Error function

The Error Function is used to determine the performance of a Machine Learning model. A Error function, in its simplest form, compares predicted and actual values. Appropriate Error function selection contributes to the model's credibility and reliability. The Error functions that are available for regression are as follows:

- Mean Squared Error (**MSE**): Is the average of the squared differences between the actual and the predicted values. For a data point Y_i and its predicted value $\hat{Y}_i$, where n is the total number of data points in the dataset, the mean squared error is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{5.2}$$

  MSE penalizes high errors caused by outliers by squaring the errors.

- Root Mean Squared Error (**RMSE**): Is the root squared mean of the difference between actual Y_i and predicted values $\hat{Y}_i$, where n is the total number of data points in the dataset, the root mean squared error is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{5.3}$$

  RMSE can be used in situations where we want to penalize high errors but not as much as MSE does.

- Mean Absolute Error (**MAE**): It takes the average sum of the absolute differences between the actual and the predicted values. For a data point Y_i and its predicted value $\hat{Y}_i$, where n is the total number of data points in the dataset, the mean absolute error is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{5.4}$$

  MAE is more robust to outliers. The insensitivity to outliers is because it does not penalize high errors caused by outliers.

- Mean Absolute Percentage Error (**MAPE**): Is similar to that of MAE, with one key difference, that it calculates error in terms of percentage, instead of raw values. Due to this, MAPE is independent of the scale of our variables.

For a data point Y_i and its predicted value $\hat{Y}_i$, where n is the total number of data points in the dataset, the mean absolute percentage error is defined as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| 100 \tag{5.5}$$

5.1.4.7 Regularization

Regularization is a method which is used to tackle the overfitting problem of the machine learning models. Overfitting is a phenomena which occurs when a model learns the information and noise in the training data to a degree that it negatively effects the performance of the model on new data. Generally, a good model does not give more weight to a certain feature. The weights are uniformly divided. This may be done by using regularization. There are two types of regularization as follows:

- Lasso Regularization (**L1**): L1 Regularization or Lasso Regularization adds a penalty to the error function. The penalty is the sum of the absolute values of weights multiplied with the tuning parameter, which decides how much we want to penalize the model.
- Ridge Regularization (**L2**): L2 Regularization or Ridge Regularization also adds a penalty to the error function. But the penalty here is the sum of the squared values of weights multiplied with the tuning parameter, which decides how much we want to penalize the model.

5.2 Model Selection

Model selection is the process of choosing one among many candidate models for a predictive modeling problem. This section will present the various models that were developed and compare them in terms of performance in order to determine the most appropriate model for the thesis' application. The developed models hyperparameters are presented in Table 5.1. We should note that Model #3 uses Exponential Decay as a learning rate schedule.

	Model #1	**Model #2**	**Model #3**	**Model #4**
Hidden Layers	2	8	8	8
Units per Hidden Layer	64	64	256	256
Activation Function	ReLU	ReLU	ReLU	ReLU
Weights Initializer	Random	Random	Normal	Normal
Optimizer	Adam	RMSprop	SGD	Adam
Initial Learning Rate	10^{-5}	10^{-3}	10^{-3}	10^{-3}
Epochs	50	100	160	80
Batch Size	64	64	64	64
Error Function	MAPE	MAPE	MAPE	MAPE
Regularization	-	-	L2(0.01)	-

Table 5.1: ANN Models' Hyperparameters

In order to determine which of these four models is the most suitable in terms of performance we must examine the following:

- **Learning Curves**: A learning curve, in general, is a graph that depicts time or experience on the x-axis and learning or improvement on the y-axis. Learning curves are widely used in machine learning for algorithms that learn (optimize their internal parameters) incrementally over time, such as deep learning neural networks. The metric used to assess learning may be maximizing, in which case higher scores (larger numbers) indicate greater learning. However, it is more common to use a minimizing score, such as loss or error, where better scores (smaller numbers) indicate greater learning and a value of 0 indicates that the training dataset was learned perfectly with no errors.

  During the training of a machine learning model, the current state of the model at each step of the training algorithm can be evaluated. It can be evaluated on the training dataset to give an idea of how well the model is "learning". It can also be evaluated on a hold-out validation dataset that is not part of the training dataset. Evaluation on the validation dataset gives an idea of how well the model is "generalizing". Thus, we have the training learning curve and the validation learning curve.

  It is common to create dual learning curves for a machine learning model during training on both the training and validation datasets. The learning curves we used are calculated on the metric by which the parameters of the model are being optimized, e.g. loss.[6]

  The shape and dynamics of a learning curve can be used to diagnose the behavior of a machine learning model and in turn perhaps suggest at the type of configuration changes that may be made to improve learning and/or performance. Three common dynamics are likely to be observed in learning curves:

  - Underfit: Underfitting refers to a model that cannot learn the training dataset.
  - Overfit: Overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset.
  - Good Fit: A good fit is the goal of the learning algorithm and exists between an overfit and underfit model. A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values.

- **Metrics**:

  - **Coefficient of determination or R Square (R^2)**: R^2 measures how much variability in dependent variable can be explained by the model. It is the square of the Correlation Coefficient (R) and that is why it is called R Square. R Square is a good measure to determine how well the model fits the dependent variables. Is expressed as a value between 0 and 1, with 1 indicating perfect fit and thus a highly reliable model for future forecasts, and 0.0 indicating that the model does not accurately model

the data at all. Is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)^2} \tag{5.6}$$

where Y_i is the real value, $\hat{Y}_i$ is the predicted value, n is the total number of data points in the dataset and $\bar{Y}$ is the mean value of Y in the dataset.

- Mean Squared Error (**MSE**)
- Root Mean Squared Error (**RMSE**)
- Mean Absolute Error (**MAE**)
- Mean Absolute Percentage Error (**MAPE**)

5.2.1 Training & Validation

The following Table, Table 5.2, provides the results of an extended training and validation procedure.

	MSE $(\mathbf{t/day})^2$	**RMSE** **(t/day)**	**MAE** **(t/day)**	**MAPE** **(%)**	$\mathbf{R^2}$
Model #1					
Training	20.258	4.500	3.194	8.104	0.763
Validation	20.153	4.489	3.178	8.083	0.764
Model #2					
Training	12.480	3.532	2.335	5.934	0.854
Validation	12.966	3.600	2.375	5.984	0.848
Model #3					
Training	5.181	2.276	1.278	3.331	0.941
Validation	5.332	2.309	1.283	3.334	0.934
Model #4					
Training	3.128	1.768	0.925	2.384	0.954
Validation	3.590	1.894	0.984	2.552	0.949

Table 5.2: ANN Models' metrics during Training-Validation procedure

The following Figures, Figures 5.4, 5.5, 5.6 and 5.7, illustrates the models' learning curves, with the blue curve representing the training error (MAPE) during the training procedure and the orange curve representing the validation error (MAPE) during the training procedure, which is calculated following the end of each epoch.

As illustrated in Figures 5.4, 5.5, 5.6 and 5.7, both training and validation error (MAPE) are sharply decreasing until a specified number of epochs, which varies for each model, and then the training and validation errors adopt a nearly linear behavior that gradually decreases and eventually terminates; the fact that the training error decreases to a stable value indicates a good fit. Additionally, the fact that the training error is slightly lower than the validation error and that the two errors eventually converge demonstrates a case of good fit.

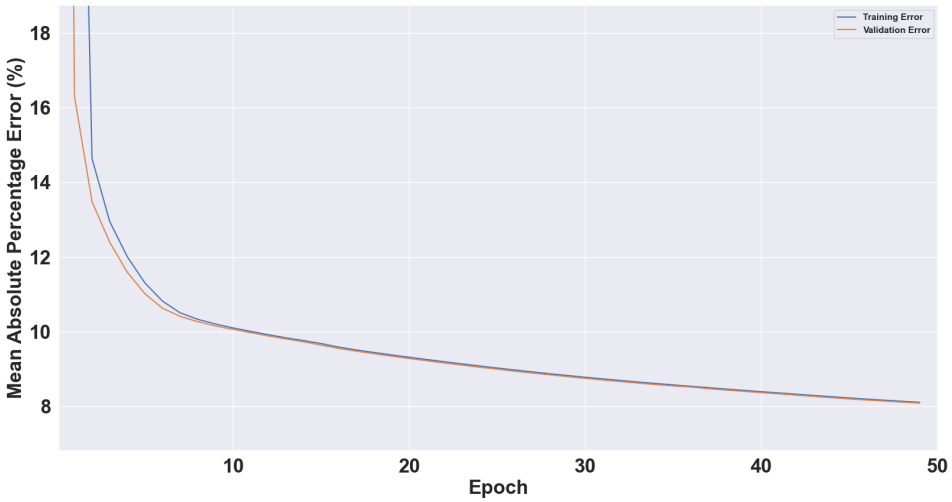


Figure 5.4: **Model #1** Learning curves

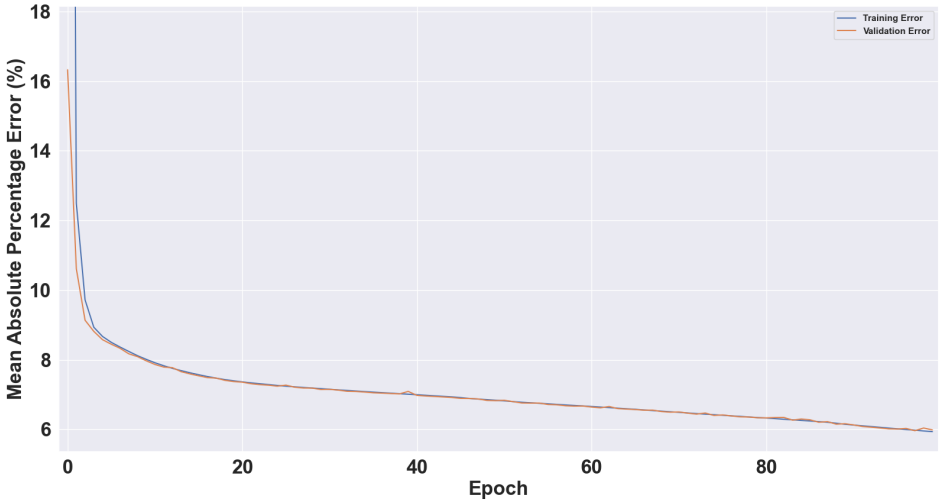


Figure 5.5: **Model #2** Learning curves

As shown in table 5.2, all models have reached an error level that is more than acceptable, but Model #4 outperforms the other three models in terms of performance. However, Model's #4 learning curves, Figure 5.7, are indicating that the model has started to overfitting because validation loss is significantly larger than training loss, thus Model #3 will be used as the final ANN model.

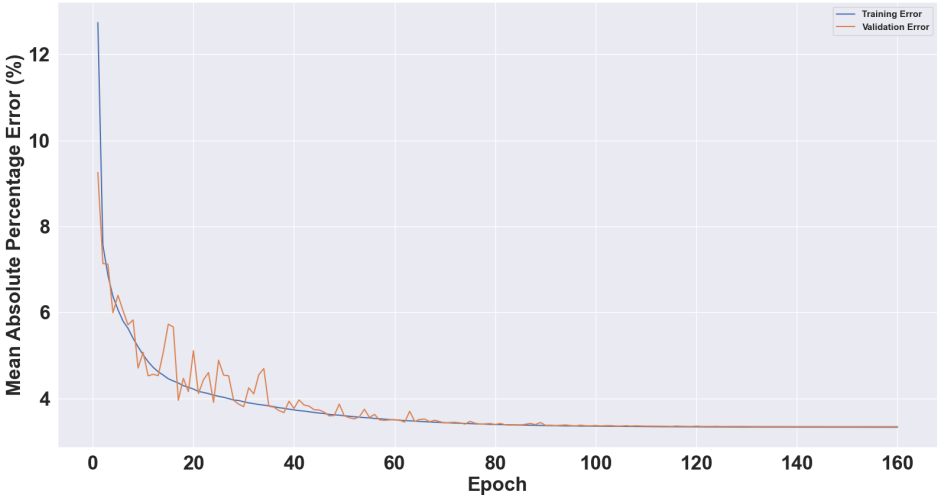


Figure 5.6: **Model #3** Learning curves

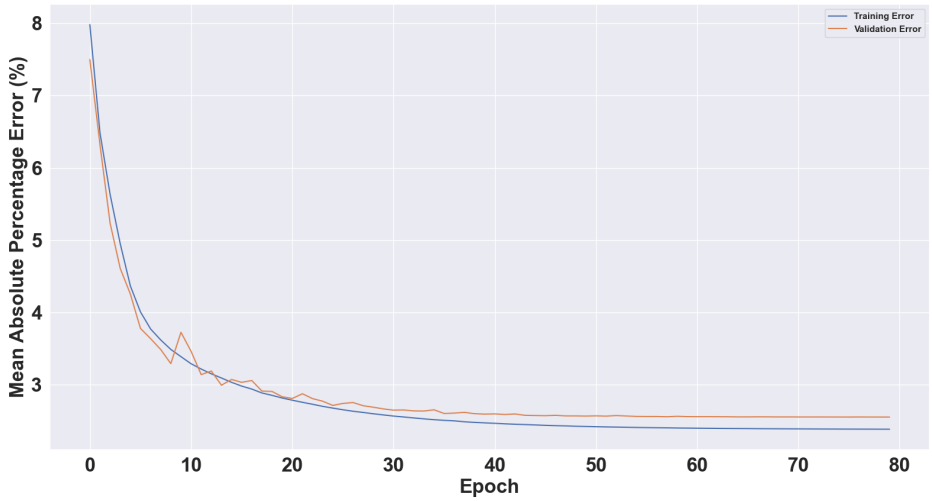


Figure 5.7: **Model #4** Learning curves

5.2.2 Testing

After evaluating all four models during the training-validation procedure and selecting Model #3 as the final ANN model, the model is tested on completely unknown data in order to examine its ability to handle unseen data. The fitting of the ANN model is depicted by FOC predicted vs FOC measured graph, Figure 5.8. To provide a better visualization of the model's fitting, the ideal fitting line $y = x$ is shown in blue.

Furthermore, we will examine the trajectories of measured and predicted FOC values to determine whether the ANN Model adequately captures the dynamics of the problem, as illustrated in Figure 5.9. The blue values in Figure 5.9 represent FOC values from the test data, while the orange values represent the ANN model's predictions. If we look at Figure 5.9, we can see that the model adequately captures

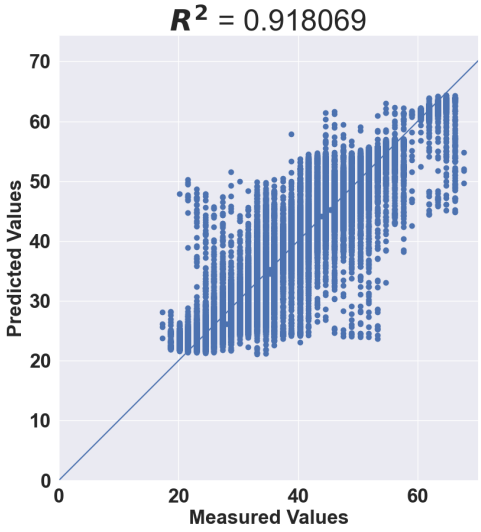


Figure 5.8: **ANN Model** testing in test data: Predicted vs Original Data

the dynamics of the problem to a sufficient degree.

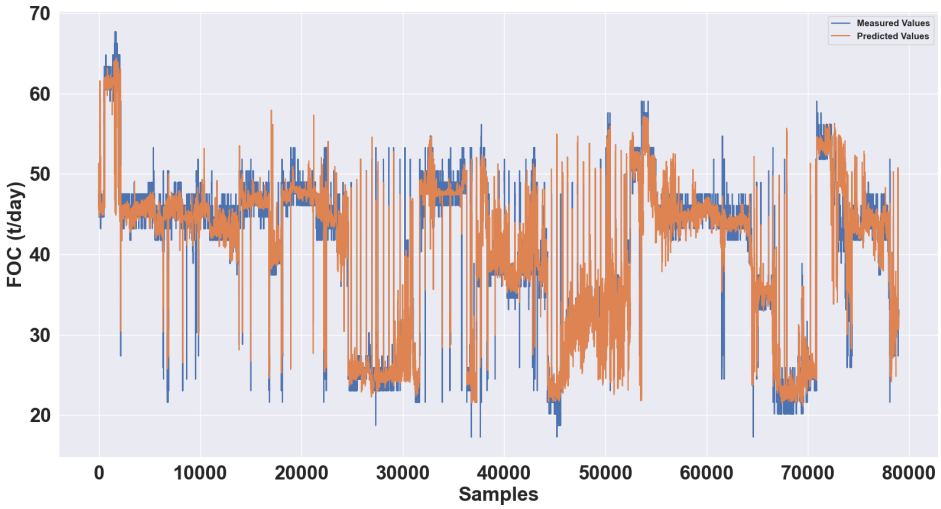


Figure 5.9: **ANN Model**: Trajectories of measured and predicted FOC values, for the test data.

The following table, Table 5.3, provides the Final ANN Model metrics at the test dataset.

	MSE $(\mathbf{t/day})^2$	**RMSE** **(t/day)**	**MAE** **(t/day)**	**MAPE** **(%)**	$\mathbf{R}^2$
Test	7.167	2.677	1.554	4.159	0.918

Table 5.3: ANN Model metrics at the Test set

Moreover, to provide insight into the MAPE distribution at each data point in

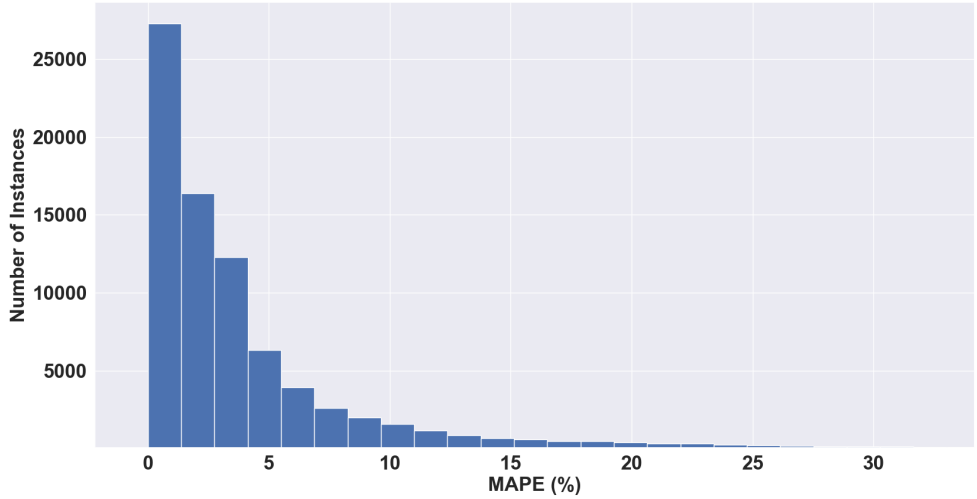


Figure 5.10: **ANN Model**: MAPE histogram at the test set.

the test set, we will provide a MAPE histogram, Figure 5.10. As implied by Figures 5.8, 5.9, 5.10 and the individual Table's 5.3 metrics, both the fitting results and the errors are more than satisfactory.

Chapter 6

Multiple Linear Regression Model

Given that the model's application to ship performance is to generate Fuel Oil Consumption - Speed curves under a variety of loading and weather conditions, we decided to develop a Multiple Linear Regression (**MLR**) model and compare its performance to that of the (**ANN**) model. As a result, this chapter will discuss the development of the (**MLR**) model.

6.1 Mathematical formulation

A Multiple Linear Regression (MLR) model describes how a dependent variable depends linearly on a series of independent variables. Of course, the model is based on the assumption that there is a linear relationship between the dependent and the independent - also within their group - variables. The population regression equation for a response variable $\mathbf{y}$ and k explanatory variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ can be written as:

$$y_i = \beta_0 + \varepsilon_i + \sum_{j=1}^{k} \beta_j x_{ij} \tag{6.1}$$

where:

- y_i: i-th observation of response variable
- x_{ij}: i-th observation of j-th explanatory variable
- β_0: regression constant term
- β_j: slope coefficient of j-th explanatory variable
- ε_i: random error of i-th observation

Coefficient β_j shows the magnitude of change in the dependent variable brought about by the increase of the independent variable x_j by one unit when the rest of the independent variables are held constant. Errors ε_i are assumed to be normally distributed (with mean $\mu = 0$ and variance σ^2) and homoscedastic (have constant variance) across the range of predicted values.[4]

The sample regression equation is similar to that of the population and consists of the predicted value and the residual, as expressed below:

$$y_i = \hat{y}_i + e_i = b_0 + e_i + \sum_{j=1}^{k} b_j x_{ij} \tag{6.2}$$

where:

- $\hat{y}_i$: predicted value of the i-th observation of response variable
- b_j: estimator of the j-th regression coefficient β_j
- e_i: error (residual) of i-th observation and estimate of regression error ε_i

Coefficients b_j are calculated using the method of least squares, which demands the minimization of the squares of the residuals $e_i = y_i - \hat{y}_i$. The initial model can be written as $\mathbf{y} = \mathbf{X}\beta + \varepsilon$ with the terms being in the following form:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Least squares method:

$$\sum_{i-1}^{n} \mathbf{e}_i^2 = \mathbf{e}^T\mathbf{e} = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) = min \tag{6.3}$$

The values of β_j for which the equation is satisfied are the b_j and the providing formula is:

$$\mathbf{b} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{y} \tag{6.4}$$

As a result, the predicted and the observed values of the response variable are related through the following equations:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{b} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{y} = \mathbf{H}\mathbf{y} \tag{6.5}$$

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T \tag{6.6}$$

where $\mathbf{H}$ is called the Hat matrix.

6.2 Model Development & Testing

In a similar fashion with Artificial Neural Networks, the MLR model will use as input variables **Final Dataset** parameters after the process of Feature Engineering. To properly compare the MLR model to the ANN model, we will use the same training and test datasets and also the same metrics to evaluate the performance of the MLR model.

There is also a model selection process in MLR models, in which various combinations of input variables are examined to determine the best subset in terms of performance, but this thesis is emphasized in the development of the ANN models. Thus, based on the available literature, we consulted [18] and [4] research and we decided to use the following subset, presented in Table 6.1, as input in our MLR model.

Parameter	Units
STW^3	$(kn)^3$
TM	(m)
TRIM	(m)
SC	(kn)
WE	(m/s)

Table 6.1: MLR Model input variables

Thus, the resulted MLR model which derived from the training set is described by the follwing equation:

$$FOC = \begin{bmatrix} STW^3 & TM & TRIM & SC & WE & 1 \end{bmatrix} \begin{bmatrix} 0.01029916 \\ 0.82579659 \\ -1.85931877 \\ 0.03459591 \\ -0.01553687 \\ 14.16311875 \end{bmatrix} \tag{6.7}$$

After the model's coefficients have been determined at the training dataset its time to test the MLR model on completely unknown data in order to examine its ability to handle unseen data. To begin, we will examine the trajectories of measured and predicted FOC values to determine whether the MLR Model adequately captures the dynamics of the problem, as illustrated in Figure 6.1. Where the blue values represent FOC values from the test data and the orange values represent the MLR model's predictions.

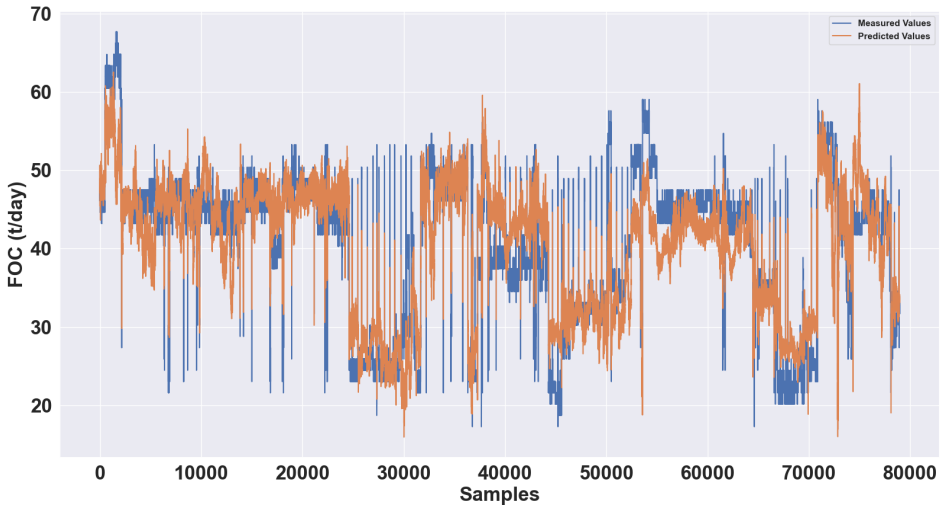


Figure 6.1: **MLR Model**: Trajectories of measured and predicted FOC values, for the test data.

As seen in Figure 6.1, the MLR model tracks the problem's dynamics but falls short of capturing noisy data.

The fitting of the MLR model is also depicted by FOC predicted vs FOC measured graph, Figure 6.2. To provide a better visualization of the model's fitting, the ideal fitting line $y = x$ is shown in blue.

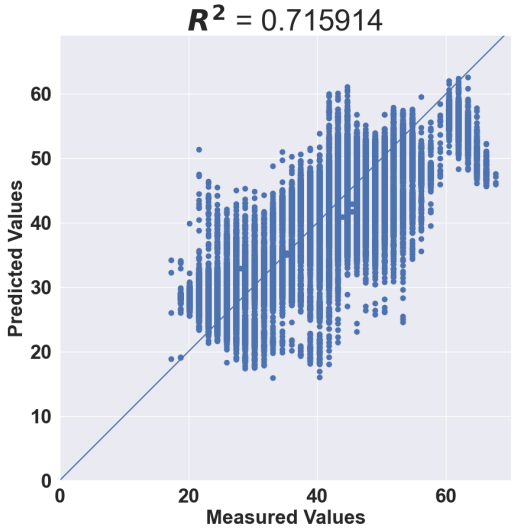


Figure 6.2: **MLR Model**: testing in test data: Predicted vs Original Data

The following table, Table 6.2, provides the MLR Model metrics at the test dataset.

	MSE $(t/day)^2$	**RMSE** (t/day)	**MAE** (t/day)	**MAPE** (%)	$\mathbf{R}^2$
Test	24.851	4.985	3.897	10.424	0.716

Table 6.2: MLR Model metrics at the Test set

Finally, to provide insight into the MAPE distribution at each data point in the test set, we will provide a MAPE histogram, Figure 6.3.

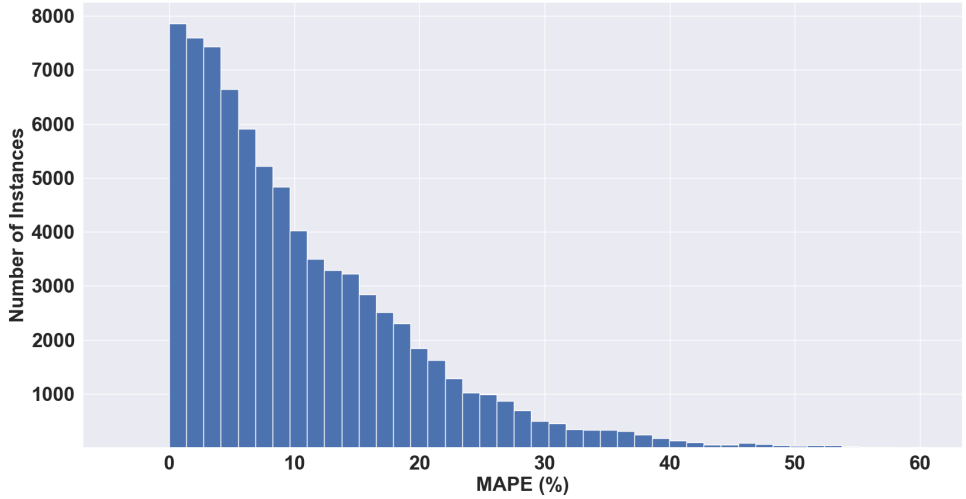


Figure 6.3: **MLR Model**: MAPE histogram at the test set.

Chapter 7

Models' Application & Comparison

This Chapter will present a comparison analysis of the developed models' predictive capability under various loading and weather conditions where the predicted fuel oil consumption is plotted against speed through water.

We decided to examine six different loading and weather conditions, presented in the following table, Table 7.1. Mean draft (TM) and trim (TRIM) represent the loading parameters, while sea current (SC) and wind effect (WE) represent the weather parameters. To assist the reader, we will convert the wind effect from m/s to Bft; when the wind effect is negative, it is called tailwind; when it is positive, it is called headwind. When the sea current is negative, the vessel is facing the current; when it is positive, the vessel is moving in the same direction as the current.

	TM (m)	**TRIM (m)**	**SC (kn)**	**WE (m/s)**	**WE (Bft)**
Case #1	7.9	3.65	-0.1	8	5
Case #2	8.9	3.3	-0.1	-2	2
Case #3	15.3	0.2	-0.1	4	3
Case #4	7.9	3.65	0.115	1.4	1
Case #5	8.9	3.3	-0.1	1	1
Case #6	12.5	-0.65	-0.13	2.5	2

Table 7.1: Examined Cases

In order to generate FOC vs Speed curves the following algorithm was carried out:

1. The user selects the values of the loading and weather parameters, e.g. TM=10 (m)
2. The user selects a range v between 0 and 1, e.g. $v = 0.2$
3. The algorithm scans the dataset to find all the available data points that contain the previously selected values multiplied within the range v, e.g. TM_{min}=8 (m) and TM_{max}=12 (m)
4. The algorithm corrects FOC values based on Admiralty Coefficient: $FOC = FOC_{ref} \left(\frac{TM}{TM_{ref}}\right)^{\frac{2}{3}}$

5. The algorithm prints in the console the number of data points it has found as well as the minimum and the maximum value of the subset's Speed through water

6. The ANN and MLR models are "fed" with the subset, for the ANN model the data are scaled and for the MLR model STW raises to the power of 3 and then MAPE and R^2 are calculated

7. The user defines the minimum and maximum value of STW as well as the step and the algorithm generates a dataset, where STW starts from the minimum value and ends at the maximum value with the user defined step, while the other parameters are held constant at every data point

8. The ANN and MLR models are "fed" with the user defined dataset, for the ANN model the data are scaled and for the MLR model STW raises to the power of 3

9. The models predict the fuel oil consumption and the FOC - STW curves are generated for both of them

The FOC - STW curves for the examined cases are shown in Figures 7.1, 7.2, 7.3, 7.4, 7.5 and 7.6 where the green color represents the curves generated by the ANN model and the yellow color represents the curves generated by the MLR model. Table 7.2, list the values of MAPE and R^2 for the examined cases.

	ANN		**MLR**	
	MAPE (%)	R^2	MAPE (%)	R^2
Case #1	4.83	0.91	12.95	0.68
Case #2	10.46	0.75	12.08	0.49
Case #3	3.16	0.78	10.11	0.13
Case #4	4.96	0.90	17.27	0.56
Case #5	9.38	0.83	12.62	0.69
Case #6	3.11	0.86	11.13	0.30

Table 7.2: Examined Cases Metrics

As illustrated in in Figures 7.1, 7.2, 7.3, 7.4, 7.5 and 7.6 and Table 7.2, the ANN model outperforms the MLR model in all six investigated cases. The reason the ANN model outperforms the MLR is that nonlinearities exist in the data, and the MLR always considers the dependency between FOC and STW to be linear. As a result, while the MLR model captures some patterns quite well when the relationship between FOC and STW is linear, it fails to capture patterns when the relationship between FOC and STW is nonlinear.

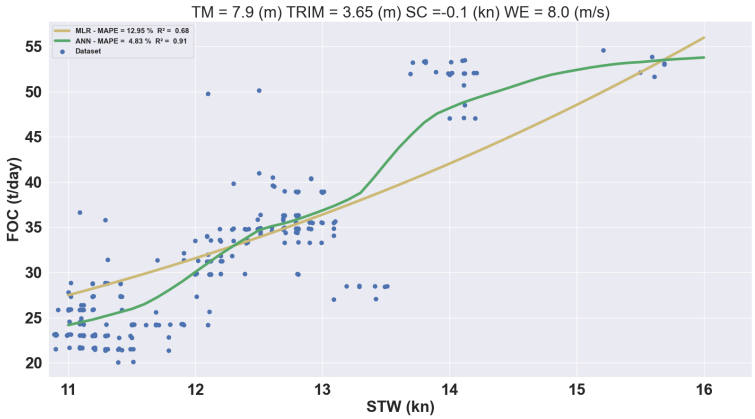


Figure 7.1: FOC - STW curves: Case #1

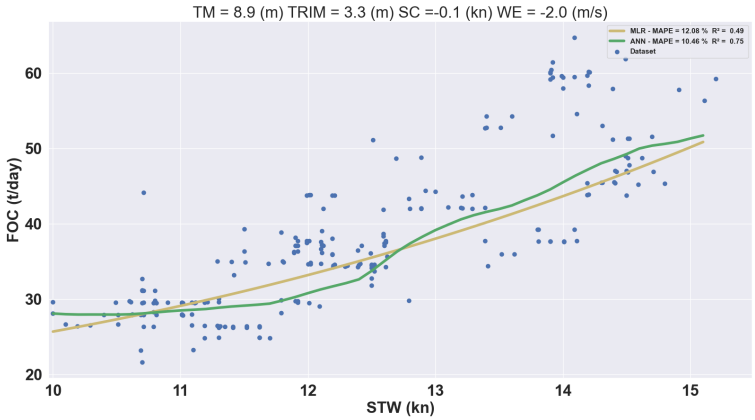


Figure 7.2: FOC - STW curves: Case #2

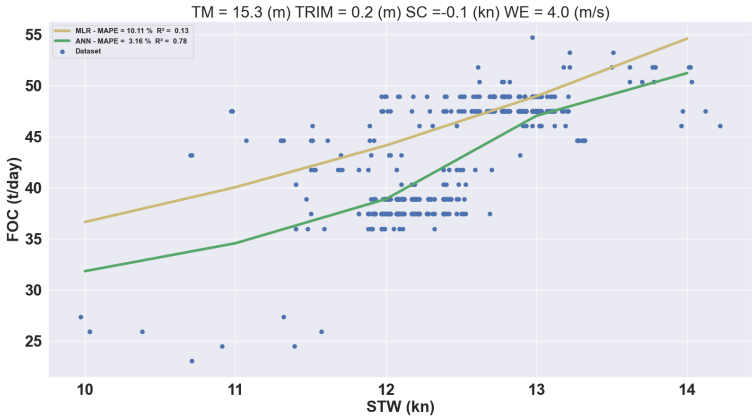


Figure 7.3: FOC - STW curves: Case #3

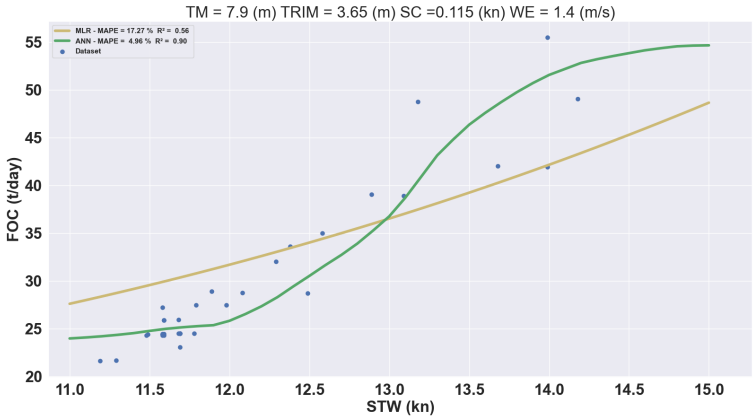


Figure 7.4: FOC - STW curves: Case #4

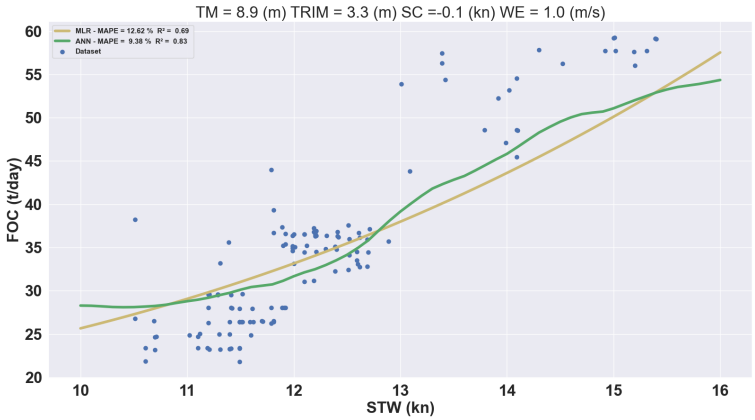


Figure 7.5: FOC - STW curves: Case #5

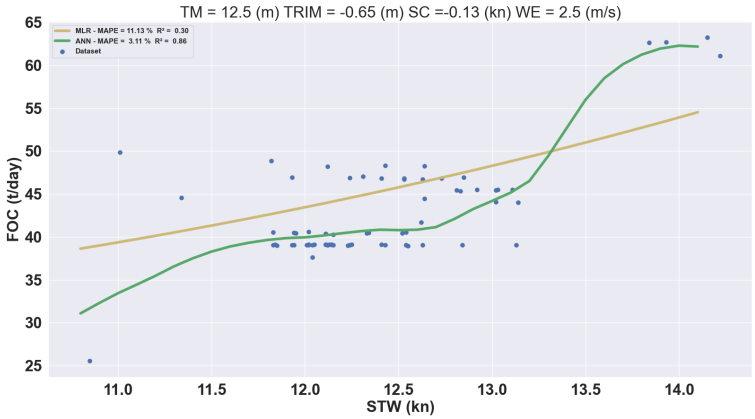


Figure 7.6: FOC - STW curves: Case #6

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Two distinct machine learning models were developed in this thesis: an Artificial Neural Network and a Multiple Linear Regression model. Although data were pre-processed prior to model design in order to achieve the lowest possible error, the effect of pre-processing on the prediction of ship fuel oil consumption has been examined in [11].

The primary objective of this thesis was to develop an "abstract" Artificial Neural Network capable of forecasting fuel oil consumption using non-engine related parameters. Due to the absence of engine-related parameters in the designed model, it was difficult to achieve a low error. Thus, determining how to tune the model properly in order to achieve the lowest possible error without overfitting the training data was extremely difficult. We experimented with various architectures, activation functions, optimizers, and a variety of other tuning parameters in order to obtain a 4.159 (%) MAPE and 0.918 R^2 on test data without overfitting the training data.

We have discovered that unless a regularization technique is used, a deep neural network with a large number of units will most likely overfit. Additionally, when properly tuned, the Stochastic Gradient Descent optimizer can achieve a lower error than Adam, and a deep neural network with eight units per hidden layer can achieve slightly better prediction than a Multiple Linear Regression model.

Finally, we demonstrated that a deep neural network with proper hyperparameter tuning can outperform a multiple linear regression model and can be used to solve ship propulsion problems that require predictive modeling..

8.2 Suggestions for Future Work

In this work, both ANN and MLR models tried to model the problem as a pattern recognition problem. It would be interesting if different types of machine learning models (e.g. Recurrent Neural Networks) were developed in order to face the problem as a time series forecasting. Additionally, it would be very interesting to have access to additional weather parameters (e.g. Significant Wave Height, Peak Period, Water Temperature) in order to gain a better understanding of how weather affects fuel oil consumption.

Bibliography

[1] Thern M. Ahlgren F. "Auto machine learning for predicting ship fuel consumption". In: *ECOS 2018-the 31st International Conference on Efficiency* (2018).

[2] N.Themelis et al. "Performance Monitoring". In: *Marine Technology* (July 2021).

[3] Lucy Gemma Aldous. "Ship operational efficiency: performance models and uncertainty analysis". PhD thesis. UCL (University College London), 2016.

[4] D. Apostolou. "Development of a statistical model for forecasting fuel consumption with application to ship route assessment". In: *National Technical University of Athens* (2021).

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. SPRINGER-VERLAG NEW YORK, 2016.

[6] Jason Brownlee. *How to use Learning Curves to Diagnose Machine Learning Model Performance*. Aug. 2019. URL: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.

[7] A. Geron. "Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow." In: O'Reilly, 2019. ISBN: 9781492032649.

[8] Brownlee Jason. *How to Choose an Activation Function for Deep Learning*. Jan. 2021. URL: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/.

[9] Brownlee Jason. *Overfitting and underfitting with machine learning algorithms*. Aug. 2019. URL: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

[10] P. Karagiannidis. "Data-driven Ship Propulsion modeling with applications in the Performance Anaysis and Fuel Consumption prediction". In: *National Technical University of Athens* (2019).

[11] Pavlos Karagiannidis and Nikos Themelis. "Data-driven modelling of ship propulsion and the effect of data pre-processing on the prediction of ship fuel consumption and speed loss". In: *Ocean Engineering* 222 (2021), p. 108616. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.108616. URL: https://www.sciencedirect.com/science/article/pii/S0029801821000512.

[12] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989). DOI: 10.1162/neco.1989.1.4.541.

[13] D. Liberto. *Quartile Definition*. Nov. 2021. URL: https://www.investopedia.com/terms/q/quartile.asp.

[14] Themelis N. et al. "A comparative study on ship performance assessment based on noon report and continuous monitoring datasets". In: *Proceedings, 12th Conference of Hellenic Institute of Marine Technology* (2018).

[15] Jose Manuel Ortiz-Rodriguez et al. "Robust Design of Artificial Neural Networks Methodology in Neutron Spectrometry". In: Jan. 2013. ISBN: 978-953-51-0935-8. DOI: 10.5772/3409.

[16] Joan P. Petersen, Ole Winther, and Daniel J. Jacobsen. "A Machine-Learning Approach to Predict Main Energy Consumption under Realistic Operational Conditions". In: *Ship Technology Research* 59.1 (2012), pp. 64–72. DOI: 10.1179/str.2012.59.1.007. eprint: https://doi.org/10.1179/str.2012.59.1.007. URL: https://doi.org/10.1179/str.2012.59.1.007.

[17] M. S. Rahman and Khalid Al-Amri. "Effect of Outlier on Coefficient of Determination". In: *International Journal of Education Research* 6 (Mar. 2011).

[18] A. Zalachoris. "Evaluation of the efficiency of a Mewis duct by performance monitoring". In: *National Technical University of Athens* (2019).

[19] Haomiao Zhou et al. "A new sampling method in particle filter based on Pearson correlation coefficient". In: *Neurocomputing* 216 (2016), pp. 208–215. ISSN: 0925-2312.

Appendix A

Artificial Neural Networks

Neural networks are mathematical models that store information by utilizing learning algorithms inspired by the brain. Due to their use in machines, neural networks are collectively referred to as an "artificial neural network". Neural networks, also known as artificial neural networks (ANNs) are a subset of machine learning and are at the heart of deep learning algorithms. Nowadays, the term "machine learning" is frequently used in this field; it refers to the scientific discipline concerned with the design and development of algorithms that enable computers to learn from data, such as sensor data or database entries. Machine-learning research is primarily concerned with automatically learning to recognize complex patterns and making intelligent decisions based on data. As a result, machine learning is inextricably linked to fields like statistics, data mining, pattern recognition, and artificial intelligence. Although neural networks are a popular framework for performing machine learning, there are numerous other methods available, including logistic regression and support vector machines.

Similar to the brain, neural networks are composed of numerous neurons connected by numerous connections. In a variety of applications, neural networks have been used to model unknown relationships between various parameters using a large number of examples. Classification of handwritten digits, speech recognition, and stock price prediction are all successful applications of neural networks. Additionally, neural networks are increasingly being used in medical applications. There are numerous types of neural networks. The Hopfield network, the multilayer perceptron, the Boltzmann machine, and the Kohonen network are all examples of various types of neural networks.

The majority of Machine Learning (ML) problems can be classified as follows:

- **Supervised learning**: A type of machine learning in which the model is fed with labeled training data. The learning algorithm attempts to model the relationships and dependencies between the target prediction output and the input labeled features. Thus, based on the relationships it has learned, the model can predict the output values for new and completely unknown data. Supervised learning is classified into two major subcategories:

    1. Regression Models: A regression model is used to investigate the relationship between two or more variables and estimate one variable based on the others. In this kind of modeling, continuous values are predicted. Because the target in this thesis is a continuous variable, the constructed neural networks are classified as a subclass of regression modeling.

2. Classification models: A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data. In this kind of modeling, discrete values are predicted.

- **Unsupervised Learning**: Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention

- **Semi-supervised Learning**: In the preceding two types, either no labels are present for all observations in the dataset or all observations have labels. Among these two extremes is semi-supervised learning. In many practical instances, the cost of labeling is fairly significant, as it requires the expertise of trained human experts. Thus, when the bulk of observations lack labels but contain some, semi-supervised techniques are the best options for model formation. These methods take advantage of the fact that while the group memberships of unlabeled data are unknown, this data contains critical information about the group parameters.

- **Reinforcement Learning**: Reinforcement Learning is a type of Machine Learning that aims at using observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk. Reinforcement learning algorithm (called the agent) continuously learns from the environment in an iterative fashion. In the process, the agent learns from its experiences of the environment until it explores the full range of possible states.

A.1 Linear Basis Function Models

Before presenting the fundamentals of the implemented neural network, a description of the Linear Basis Function Models should be included. Despite the fact that in this study there is not always a linear connection between the various input and output datasets, it would be quite instructive and clarifying to write a few things about the simplest regression model, the **Linear Regression Model**. Although linear models have significant limitations as practical techniques for pattern recognition, particularly for problems involving input spaces of high dimensionality, they have nice analytical properties and form the foundation for more sophisticated models.

Given a training data set comprising N observations $\{x_n\}$, where $n = 1, ..., N$, together with corresponding target values $\{t_n\}$, the goal is to predict the value of t for a new value of $\mathbf{x}$. In the simplest approach, this can be done by directly constructing an appropriate function $y(\mathbf{x})$ whose values for new inputs $\mathbf{x}$ constitutes the predictions for the corresponding values of t. More generally, from a probabilistic perspective, the aim is to model the predictive distribution $p(t|\mathbf{x})$ because this expresses the uncertainty about the value of t for each value of $\mathbf{x}$. From this conditional distribution predictions of t can be made, for any new value of $\mathbf{x}$, in such a way as to minimize the expected value of a suitably chosen loss function. One of the most common choices of loss function for real-valued variables is the Squared Loss,

$L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2$, for which the optimal solution is given by the conditional expectation of t.

The simplest linear model for regression is one that involves a linear combination of the input variables

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{D} w_j x_j \tag{A.1}$$

where $x = (x_1, ..., x_D)^T$. This is often simply known as linear regression. The key property of this model is that it is a linear function of the parameters $w_1, ..., w_D$. It is also, however, a linear function of the input variables x_i, and this imposes significant limitations on the model. We therefore extend the class of models by considering linear combinations of fixed nonlinear functions of the input variables, of the form

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \tag{A.2}$$

where $\phi_j(\mathbf{x})$ are known as basis functions. By denoting the maximum value of the index j by $M - 1$, the total number of parameters in this model will be M. The parameter w_0 allows for any fixed offset in the data and is sometimes called a bias parameter. It is often convenient to define an additional dummy "basis function" $\phi_0(\mathbf{x}) = 1$ so that

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \tag{A.3}$$

where $\mathbf{w} = (w_0, ..., w_{M-1})^T$ and $\phi = (\phi_0, ..., \phi_{M-1})^T$. By using nonlinear basis functions, we allow the function $y(\mathbf{x}, \mathbf{w})$ to be a nonlinear function of the input vector $\mathbf{x}$. Functions of the form (A.2) are called linear models, however, because this function is linear in $\mathbf{w}$. It is this linearity in the parameters that will greatly simplify the analysis of this class of models.

It has also been seen that the assumption of linearity in the parameters led to a range of useful properties including closed-form solutions to the least-squares problem. Furthermore, for a suitable choice of basis functions, arbitrary non-linearities can be modeled in the mapping from input variables to targets. It might appear, therefore, that such linear models constitute a general purpose framework for solving problems in pattern recognition. Unfortunately, there are some significant shortcomings with linear models, which will cause study to turn to more complex models such as support vector machines and neural networks. The difficulty stems from the assumption that the basis functions $\phi_j(\mathbf{x})$ are fixed before the training data set is observed and is a manifestation of the curse of dimensionality. As a consequence, the number of basis functions needs to grow rapidly, often exponentially, with the dimensionality D of the input space.

Fortunately, there are two properties of real data sets that can be exploited to help alleviate this problem. First of all, the data vectors $\{\mathbf{x}_n\}$ typically lie close to a non-linear manifold whose intrinsic dimensionality is smaller than that of the input space as a result of strong correlations between the input variables. Neural network models, which use adaptive basis functions having sigmoidal nonlinearities, can adapt the parameters so that the regions of input space over which the basis functions vary corresponds to the data manifold. The second property is that target variables may have significant dependence on only a small number of possible

directions within the data manifold. Neural networks can exploit this property by choosing the directions in input space to which the basis functions respond.

A.2 Feed-Forward Neural Networks

The linear models for regression, as aforementioned, are based on linear combinations of fixed nonlinear basis functions $\phi_j(\mathbf{x})$ and take the form

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right) \tag{A.4}$$

where $f(\cdot)$ is a nonlinear activation function in the case of classification and is the identity in the case of regression. Our goal is to extend this model by making the basis functions $\phi_j(\mathbf{x})$ depend on parameters and then to allow these parameters to be adjusted, along with the coefficients $\{w_j\}$, during training. There are, of course, many ways to construct parametric nonlinear basis functions. Neural networks use basis functions that follow the same form as (A.4), so that each basis function is itself a nonlinear function of a linear combination of the inputs, where the coefficients in the linear combination are adaptive parameters.

This leads to the basic neural network model, which can be described a series of functional transformations. First, we construct M linear combinations of the input variables $x_1, ..., x_D$ in the form

$$\alpha_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{A.5}$$

where $j = 1, ..., M$, and the superscript (1) indicates that the corresponding parameters are in the first 'layer' of the network. We shall refer to the parameters $w_{ji}^{(1)}$ as weights and the parameters $w_{j0}^{(1)}$ as biases. The quantities α_j are known as activations. Each of them is then transformed using a differentiable, nonlinear activation function $h(\cdot)$ to give

$$z_j = h(\alpha_j). \tag{A.6}$$

These quantities correspond to the outputs of the basis functions in (A.4) that, in the context of neural networks, are called hidden units. The nonlinear functions $h(\cdot)$ are generally chosen to be sigmoidal functions such as the such as the logistic sigmoid or the "tanh" function. Following (A.4), these values are again linearly combined to give output unit activations

$$\alpha_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{A.7}$$

where $k = 1, ..., M$,and K is the total number of outputs. This transformation corresponds to the second layer of the network, and again the $w_{k0}^{(2)}$ are bias parameters. Finally, the output unit activations are transformed using an appropriate activation function to give a set of network outputs y_k. The choice of activation function is determined by the nature of the data and the assumed distribution of target variables and follows the same considerations as for linear models. Thus, for standard regression problems, the activation function is the identity so that $y_k = \alpha_k$.

All these various stages van be combined in order to give the overall network function that, for sigmoidal output unit activation functions, takes the form

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{A.8}$$

where the set of all weight and bias parameters have been grouped together into a vector $\mathbf{w}$. Thus. the neural network model is simply a nonlinear function from a set of input variables $\{x_i\}$ to a set of output variables $\{y_k\}$ controlled by a vector $\mathbf{w}$ of adjustable parameters.

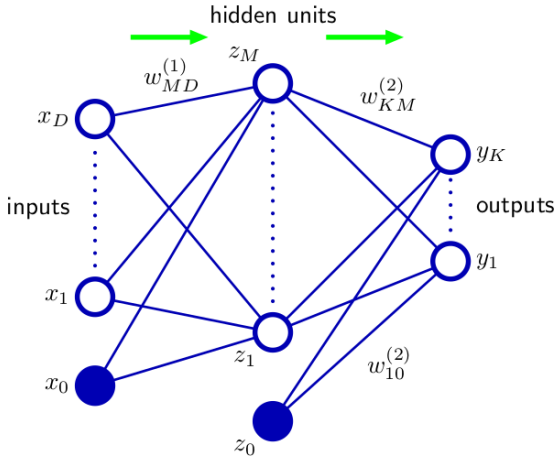


Figure A.1: Network diagram for the two-layer neural network corresponding to (A.8). The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links inputs coming from additional input and hidden variables x_0 and z_0. Arrows denote the direction of information flow through the network during forward propagation.

This function can be represented in the form of a network diagram as shown in Figure A.1. The process of evaluating (A.8) can then be interpreted as a forward propagation of information through the network. It should be emphasized that these diagrams do not represent probabilistic graphical models because the internal nodes represent deterministic variables rather than stochastic ones.

As previously discussed, the bias parameters in (A.5) can be absorbed into the set of weight parameters by defining an additional input variable x_0 whose value is clamped at $x_0 = 1$, so that (A.5) takes the form

$$\alpha_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i. \tag{A.9}$$

The second-layer biases can be similarly absorbed into the second-layer weights, so that the overall network function becomes

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right). \tag{A.10}$$

As can be seen from Figure A.1 the neural network model comprises two stages of processing, and the neural network is also known as the multilayer percepton, or MLP. A key difference compared to the perceptron, however, is that the neural network uses continuous sigmoidal nonlinearities in the hidden units, whereas the perceptron uses step function nonlinearities. This means that the neural network function is differentiable with respect to the network parameters, and this property will play a central role in network training.

If the activation functions of all the hidden units in a network are taken to be linear, then for any such network an equivalent network without hidden units can always be found. This follows from the fact that the composition of successive linear transformations is itself a linear transformation. However, if the number of hidden units is smaller than either the number of input or output units, then the transformations that the network can generate are not the most general possible linear transformations from inputs to outputs because information is lost in the dimensionality reduction at the hidden units. In general, however, there is little interest in multilayer networks of linear units.

The network architecture shown in Figure A.1 is the most commonly used one in practice. However, it is easily generalized, for instance by considering additional layers of processing each consisting of a weighted linear combination of the form (A.7) followed by an element-wise transformation using a nonlinear activation function. Note that there is some confusion in the literature regarding the terminology for counting the number of layers in such networks. Thus the network in Figure A.1 may be described as a 3-layer network (which counts the number of layers of units, and treats the inputs as units) or sometimes as a "single-hidden-layer" network (which counts the number of layers of hidden units).

A.2.1 Network Training

So far, the neural networks have been viewed as a general class of parametric nonlinear functions from a vector $\mathbf{x}$ of input variables to a vector y of output variables. A simple approach to the problem of determining the network parameters is to minimize a sum-of-squares error function. Given a training set comprising a set of input vectors $\{\mathbf{x}_n\}$, where $n = 1, ..., N$, together with a corresponding set of target vectors $\{\mathbf{t}_n\}$, the error function is minimized

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2 . \tag{A.11}$$

However, a much more general view of network training could be provided by first giving a probabilistic interpretation to the network outputs. Here, the use of probabilistic predictions will provide with a clearer motivation both for the choice of output unit nonlinearity and the choice of error function.

Starting with regression problems, and for the moment considering a single target variable t that can take any real value, it is assumed that t has a Gaussian distribution with a $\mathbf{x}$-dependent mean, which is given by the output of the neural network, so that

$$p(t|\mathbf{x}, \mathbf{w}) = \mathcal{N}\left(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}\right) \tag{A.12}$$

where β is the precision (inverse variance) of the Gaussian noise. For the conditional distribution given by (A.12), it is sufficient to take the output unit activation

function to be the identity, because such a network can approximate any continuous function from $\mathbf{x}$ to y. Given a data set of N independent, identically distributed observations $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, along with corresponding target values $\mathbf{t} = \{t_1, ..., t_N\}$, the corresponding likelihood function can be constructed

$$p\left(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta\right) = \prod_{n=1}^{N} p\left(t_n|\mathbf{x}_n, \mathbf{w}, \beta\right). \tag{A.13}$$

Taking the negative algorithm, the error function is obtained

$$\frac{\beta}{2}\sum_{n=1}^{N}\left\{y\left(\mathbf{x}_n, \mathbf{w}\right) - t_n\right\}^2 - \frac{N}{2}\ln\beta + \frac{N}{2}\ln\left(2\pi\right) \tag{A.14}$$

which can be used to learn the parameters $\mathbf{x}$ and β. Note that in the neural networks literature, it is usual to consider the minimization of an error function rather than the maximization of the (log) likelihood, and so here this convention should be followed. Consider first the determination of $\mathbf{w}$. Maximizing the likelihood function is equivalent to minimizing the sum-of-squares error function given by

$$E\left(\mathbf{w}\right) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{y}\left(\mathbf{x}_n, \mathbf{w}\right) - \mathbf{t}_n\right\}^2 \tag{A.15}$$

where additive and multiplicative constants have been discarded. The value of $\mathbf{w}$ found by minimizing $E\left(\mathbf{w}\right)$ will be denoted $\mathbf{w}_{ML}$ because it corresponds to the maximum likelihood solution.

Having found $\mathbf{w}_{ML}$ the value of β an be found by minimizing the negative log likelihood to give

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left\{y\left(\mathbf{x}_n, \mathbf{w}_{ML}\right) - t_n\right\}^2. \tag{A.16}$$

Note that this can be evaluated once the iterative optimization required to find $\mathbf{w}_{ML}$ is completed. There is a natural pairing of the error function (given by the negative log likelihood) and the output unit activation function. In the regression case, the network can be seen as having an output activation function that is the identity, so that $y_k = \alpha_k$. The corresponding sum-of-squares error function has the property

$$\frac{\partial E}{\partial \alpha_k} = y_k - t_k \tag{A.17}$$

which should be used when discussing error backpropagation.

A.2.1.1 Parameter Optimization

The next task is that of finding a weight vector $\mathbf{w}$ which minimizes the chosen function $E\left(\mathbf{w}\right)$. At this point, it is useful to have a geometrical picture of the error function, which can be seen as a surface sitting over weight space as shown in Figure A.2.

First note that if a small step in weight space is made from $\mathbf{w}$ to $\mathbf{w}+\delta\mathbf{w}$ then the change in the error function is $\delta E \simeq \delta\mathbf{w}^T\nabla E\left(\mathbf{w}\right)$, where the vector $\nabla E\left(\mathbf{w}\right)$ points in the direction of greatest rate of increase of the error function. Because the error

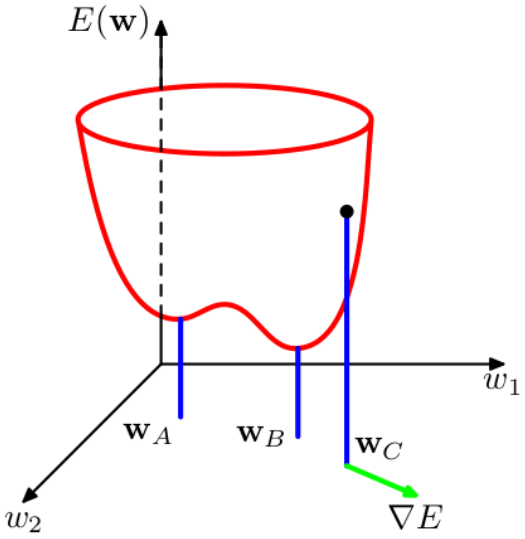


Figure A.2: Geometrical view of the error function $E(\mathbf{w})$ as a surface sitting over weight space. Point $\mathbf{w}_A$ is a local minimum and $\mathbf{w}_B$ is the global minimum. At any point $\mathbf{w}_C$, the local gradient of the error surface is given by the vector ∇E.

$E(\mathbf{w})$ is a smooth continuous function of $\mathbf{w}$, its smallest value will occur at a point in weight space such that the gradient of the error function vanishes, so that

$$\nabla E(\mathbf{w}) = 0 \tag{A.18}$$

as otherwise a small step could be made in the direction of $-\nabla E(\mathbf{w})$ and thereby further reduce the error. Points at which the gradient vanishes are called stationary points, and may be further classified into minima, maxima, and saddle points.

The main goal is to find a vector $\mathbf{w}$ such that $E(\mathbf{w})$ takes its smallest value. However, the error function typically has a highly nonlinear dependence on the weights and bias parameters, and so there will be many points in weight space at which the gradient vanishes (or is numerically very small). Indeed, for any point $\mathbf{w}$ that is a local minimum, there will be other points in weight space that are equivalent minima. For instance, in a two-layer network of the kind shown in Figure A.1 with M hidden units, each point in weight space is a member of a family of $M!2^M$ equivalent points.

Furthermore, there will typically be multiple inequivalent stationary points and in particular multiple inequivalent minima. A minimum that corresponds to the smallest value of the error function for any weight vector is said to be a global minimum. Any other minima corresponding to higher values of the error function are said to be local minima.

For a successful application of neural networks, it may not be necessary to find the global minimum (and in general it will not be known whether the global minimum has been found) but it may be necessary to compare several local minima in order to find a sufficiently good solution.

Because there is clearly no hope of finding an analytical solution to the equation (A.18) a good alternative could be the iterative numerical procedures. The optimization of continuous nonlinear functions is a widely studied problem and there

exists an extensive literature on how to solve it efficiently. Most techniques involve choosing some initial value $\mathbf{w}^{(0)}$ for the weight vector and then moving through weight space in a succession of steps of the form

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \delta\mathbf{w}^{(\tau)} \tag{A.19}$$

where τ labels the iteration step. Different algorithms involve different choices for the weight vector update $\delta\mathbf{w}^{(\tau)}$ Many algorithms make use of gradient information and therefore require that, after each update, the value of $\nabla E(\mathbf{w})$ is evaluated at the new weight vector $\mathbf{w}^{(\tau+1)}$.

A.2.2 Gradient descent optimization

The simplest approach to using gradient information is to choose the weight update in (A.19) to comprise a small step in the direction of the negative gradient, so that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E\left(\mathbf{w}^{(\tau)}\right) \tag{A.20}$$

where the parameter $\eta > 0$ is known as the learning rate. After each such update, the gradient is re-evaluated for the new weight vector and the process repeated. Note that the error function is defined with respect to a training set, and so each step requires that the entire training set be processed in order to evaluate ∇E. Techniques that use the whole data set at once are called batch methods. At each step the weight vector is moved in the direction of the greatest rate of decrease of the error function, and so this approach is known as gradient descent or steepest descent.

In order to find a sufficiently good minimum, it may be necessary to run a gradient-based algorithm multiple times, each time using a different randomly chosen starting point, and comparing the resulting performance on an independent validation set.

There is, however, an on-line version of gradient descent that has proved useful in practice for training neural networks on large data sets.[12] Error functions based on maximum likelihood for a set of independent observations comprise a sum of terms, one for each data point

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}). \tag{A.21}$$

On-line gradient descent, also known as sequential gradient descent or stochastic gradient descent (SGD), makes an update to the weight vector based on one data point at a time, so that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_n\left(\mathbf{w}^{(\tau)}\right) \tag{A.22}$$

This update is repeated by cycling through the data either in sequence or by selecting points at random with replacement. There are of course intermediate scenarios in which the updates are based on batches of data points.

A.2.3 Error Backpropagation

The main goal in this section is to find an efficient technique for evaluating the gradient of an error function $E(\mathbf{w})$ for a feed-forward neural network. It is apparent that this can be achieved using a local message passing scheme in which information is sent alternately forwards and backwards through the network and is known as error backpropagation, or sometimes simply as backprop.

It should be noted that the term "backpropagation" is used in the neural computing literature to mean a variety of different things. For instance, the multilayer perceptron architecture is sometimes called a "backpropagation network". The term "backpropagation" is also used to describe the training of a multilayer perceptron using gradient descent applied to a sum-of-squares error function. In order to clarify the terminology, it is useful to consider the nature of the training process more carefully. Most training algorithms involve an iterative procedure for minimization of an error function, with adjustments to the weights being made in a sequence of steps. At each such step, two distinct stages can be distinguished. In the first stage, the derivatives of the error function with respect to the weights must be evaluated. As it is evident, the important contribution of the backpropagation technique is in providing a computationally efficient method for evaluating such derivatives. Because it is at this stage that errors are propagated backwards through the network, the term backpropagation should be used specifically to describe the evaluation of derivatives. In the second stage, the derivatives are then used to compute the adjustments to be made to the weights. It is important to recognize that the two stages are distinct. Thus, the first stage, namely the propagation of errors backwards through the network in order to evaluate derivatives, can be applied to many other kinds of network and not just the multilayer perceptron. Similarly, the second stage of weight adjustment using the calculated derivatives can be tackled using a variety of optimization schemes, many of which are substantially more powerful than simple gradient descent.

A.2.3.1 Evaluation of error-function derivatives

We now derive the backpropagation algorithm for a general network having arbitrary feed-forward topology, arbitrary differentiable nonlinear activation functions, and a broad class of error function. The resulting formulas will then be illustrated using a simple layered network structure having a single layer of sigmoidal hidden units together with a sum-of-squares error.

Many error functions of practical interest, for instance those defined by maximum likelihood for a set of independent and identically distributed data, comprise a sum of terms, one for each data point in the training set, so that

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}). \tag{A.23}$$

Here, the problem of evaluating $\nabla E_n(\mathbf{w})$ should be considered for one such term in the error function. This may be used directly for sequential optimization, or the results can be accumulated over the training set in the case of batch methods.

Consider first a simple linear model in which the outputs y_k are linear combina-

tion of the input variables x_i so that

$$y_k = \sum_i w_{ki} x_i \tag{A.24}$$

together with an error function that, for a particular input pattern n, takes the form

$$E_n = \frac{1}{2} \sum k \left(y_{nk} - t_{nk} \right)^2 \tag{A.25}$$

where $y_{nk} = y_k \left(\mathbf{x}_n, \mathbf{w} \right)$. The gradient of this error function with respect to a weight w_{ji} is given by

$$\frac{\partial E_n}{\partial w_{ji}} = \left(y_{nj} - t_{nj} \right) x_{ni} \tag{A.26}$$

which can be interpreted as a "local" computation involving the product of an "error signal" $y_{nj} - t_{nj}$ associated with the output end of the link w_{ji} and the variable x_{ni} associated with the input end of the link.[5]

We shall now see how this simple result extends to the more complex setting of multilayer feed-forward networks. In a general feed-forward network, each unit computes a weighted sum of its inputs of the form

$$a_j = \sum_i w_{ji} z_i \tag{A.27}$$

where z_i is the activation of a unit, or input, that sends a connection to unit j, and w_{ji} is the weight associated with that connection. The sum in (A.27) is transformed by a nonlinear activation function $h\left(\cdot\right)$ to give the activation z_i of unit j in the form

$$z_j = h\left(a_j\right). \tag{A.28}$$

Note that one or more of the variables z_i in the sum in (A.27) could be an input, and similarly, the unit j in (A.28) could be an output.

For each pattern in the training set, we shall suppose that we have supplied the corresponding input vector to the network and calculated the activations of all of the hidden and output units in the network by successive application of (A.27) and (A.28). This process is often called forward propagation because it can be regarded as a forward flow of information through the network.

Now consider the evaluation of the derivative of E_n with respect to a weight w_{ji}. The outputs of the various units will depend on the particular input pattern n. However, in order to keep the notation uncluttered, we shall omit the subscript n from the network variables. First we note that E_n depends on the weight w_{ji}, only via the summed input a_j to unit j. We can therefore apply the chain rule for partial derivatives to give

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \tag{A.29}$$

We now introduce a useful notation

$$\delta_j = \frac{\partial E_n}{\partial a_j} \tag{A.30}$$

where the δ's are often referred to as errors for reasons we shall see shortly. Using (A.27), we can write

$$\frac{\partial a_j}{\partial w_{ji}} = z_i. \tag{A.31}$$

Substituting (A.30) and (A.31) into (A.29), we then obtain

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i. \tag{A.32}$$

Equation (A.32) ells us that the required derivative is obtained simply by multiplying the value of δ for the unit at the output end of the weight by the value of z for the unit at the input end of the weight (where $z = 1$ in the case of a bias). Thus, in order to evaluate the derivatives, we need only to calculate the value of δ_j for each hidden and output unit in the network, and then apply (A.32).

As we have seen already, for the output units, we have

$$\delta_k = y_k - t_k \tag{A.33}$$

provided we are using the canonical link as the output-unit activation function. To evaluate the δ's for hidden units, we again make use of the chain rule for partial derivatives,

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{A.34}$$

where the sum runs over all units k to which unit j ends connections. The arrangement of units and weights is illustrated in Figure A.3.

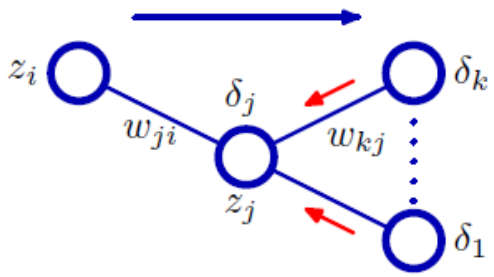


Figure A.3: Illustration of the calculation of δ_j for hidden unit j by backpropagation of the δ's from those units k to which unit j sends connections. The blue arrow denotes the direction of information flow during forward propagation, and the red arrows indicate the backward propagation of error information.

Note that the units labelled k could include other hidden units and/or output units. In writing down (A.34), we are making use of the fact that variations in a_j give rise to variations in the error function only through variations in the variables a_k. If we now substitute the definition of δ given by (A.30) into (A.34), and make the use of (A.27) and (A.28), we obtain the following backpropagation formula

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{A.35}$$

which tells us that the value of δ for a particular hidden unit can be obtained by propagating the δ's backwards from units higher up in the network, as illustrated in Figure A.3. Note that the summation in (A.35) is taken over the first index on w_{kj} (corresponding to backward propagation of information through the network), whereas in the forward propagation equation $z_k = h\left(\sum_j w_{kj} z_j\right)$ it is taken over the second index. Because we already know the values of the δ's for the output units, it follows that by recursively applying (A.35) we can evaluate the δ's for all of the hidden units in a feed-forward network, regardless of its topology.

Appendix B

Overfitting in Artificial Neural Networks

B.1 Definition

The best way to conceptualize supervised machine learning is as approximating a target function f that maps input variables $\mathbf{X}$ to an output variable y.

$$y = f(\mathbf{X}) \tag{B.1}$$

A critical factor to consider while learning the target function from the training data is the model's generalizability to new data. Generalization is critical since the data we collect is only a sample, it is incomplete and noisy.

Inductive learning is the term used in machine learning to refer to the process of learning the target function from training data. Induction is the process of determining general concepts from specific examples, which is precisely what supervised machine learning tasks want to accomplish. This is in contrast to deduction, which works in the opposite direction and strives to derive specific concepts from general rules. Generalization refers to the extent to which the concepts learnt by a machine learning model apply to specific situations not encountered during the learning process. A successful machine learning model should be capable of generalizing well from training data to any data in the problem domain. This enables us to make future predictions based on data that the model has never seen. When discussing how successfully a machine learning model learns and generalizes to new data, there is a term used in machine learning called overfitting and underfitting. Overfitting and underfitting are the two primary causes of machine learning algorithms performing poorly. A model that is underfitting has a high bias and a low variance. It cannot learn the problem regardless of the specific samples in the training data. A model that is overfitting has a low bias but a high variance. The model over-learns the training data, and performance varies significantly when new unseen examples or even statistical noise is added to the training dataset's examples.

In statistics, the term "fit" refers to the degree to which a target function is approximated. This is appropriate terminology for machine learning, because algorithms for supervised machine learning attempt to approximate the unknown underlying mapping function for the output variables given the input variables. The term "goodness of fit" is frequently used in statistics to refer to measures used to determine how well an approximation of a function matches the target function.

While some of these techniques are applicable to machine learning (e.g., calculating residual errors), others assume that we know the form of the target function we are approximating, which is not the case in machine learning. If we knew the form of the target function, we could make predictions directly from it, rather than attempting to learn an approximation from noisy training data samples.

Overfitting is a term that refers to a model that is overly accurate in its approximation of the training data. Overfitting occurs when a model learns so much detail and noise in the training data that it has a detrimental effect on the model's performance on new data. This means that the model captures noise or random fluctuations in the training data and learns them as principles. The issue is that these principles do not apply to new data, which has a detrimental effect on the models' ability to generalize. Overfitting is more frequent with nonparametric and nonlinear models (e.g. Neural Networks), which have a greater degree of freedom while learning a target function. As a result, many nonparametric machine learning algorithms incorporate parameters or approaches for limiting and constraining the amount of detail learned by the model.[9]

B.2 Overfitting detection

The main causes of overfitting are: model's complexity (i.e. in our case a model with 8 hidden layers and 256 units per layer), noisy data and limited size of training data. We encountered the first of the three causes mentioned above, model complexity. Complex models have the advantage of significantly reducing error compared to simpler models (e.g., a neural network with 8 hidden layers and 8 units per layer), but they are risky because they may have over-fitted the training data, which is why they have such low error compared to simpler models.

We must emphasize that there is no universally accepted method for detecting overfitting or underfitting. However, the most common pattern for overfitting can be seen on learning curve plots, where model performance on the training dataset continues to improve (e.g. loss or error continues to fall) and performance on the validation set improves to a point and then begins to get worse. In the following figure, Figure B.1, we can take a look at a the learning curves of an overfitted model, in our case Model #4 from Chapter 5.

As illustrated in Figure B.1, after epoch 40 and before the stop of the training procedure at epoch 80 , training and validation loss continue parallel albeit with a gap between them. The fact that validation loss has not increased indicates that the model might not begun to overfit, but the gap between training and validation loss raises doubts about whether the model has begun to overfit or not.

As previously stated, there is no golden rule for determining whether or not a model has overfit the training data.. Model's 4 training loss at epoch 80 is 2.384 % and validation loss is 2.552 % , the deviation between them is 6.58 %. The fact is that we were unsure whether this variance signals an overfit, and hence needed to conduct additional research. Thus, we evaluated Model #4 in the creation of the FOC - STW curves, where y denotes FOC and x denotes STW, and the remaining four inputs are held constant to represent the loading and weather conditions. As a result, a scatter plot between FOC and STW is produced, with the line established by the model indicates the model's fit, like Figure B.2.

Thus, the following figures, Figures B.3, B.4, B.5 and B.6, illustrate the curves

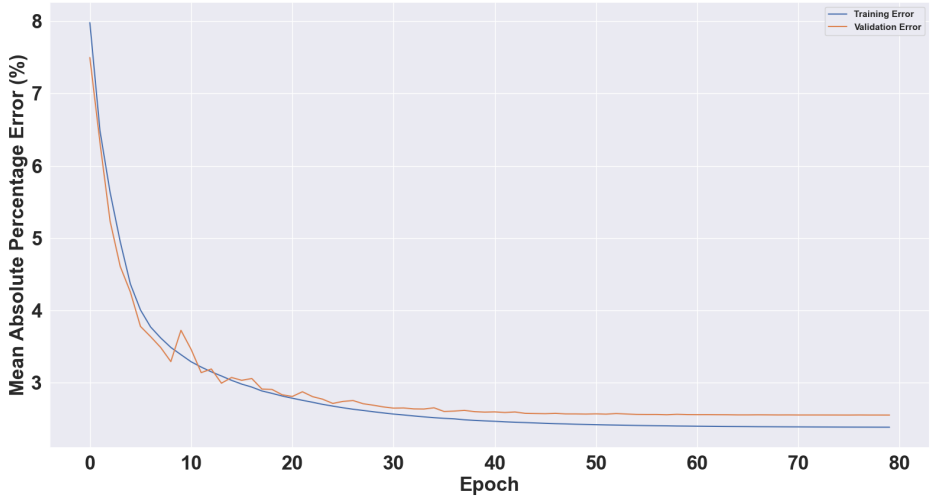


Figure B.1: Overfitted Model's Learning curves

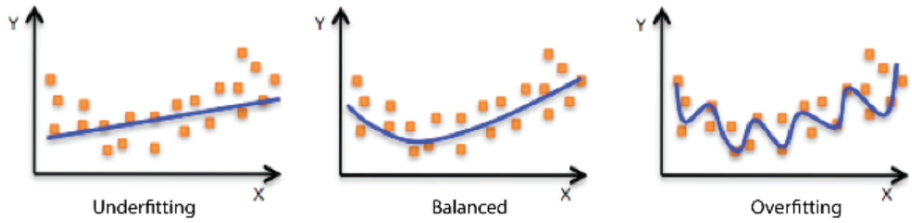


Figure B.2: Model Fitting: Overfitting, Underfitting, and Balanced

that were generated for different examined cases in the same manner as we did in Chapter 7 to determine whether Model #4 has overfitted the training data.

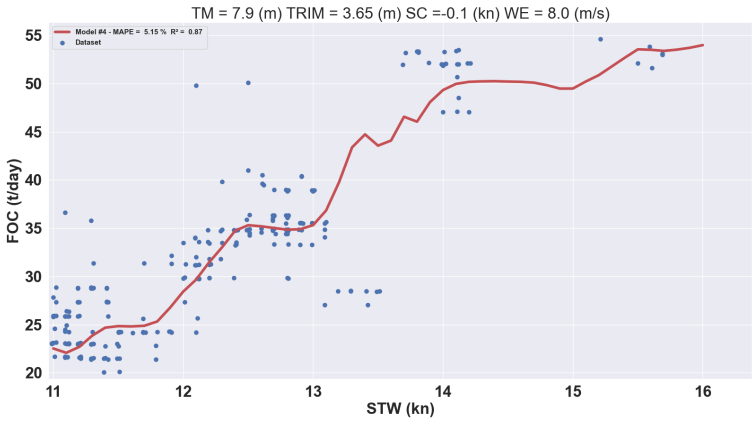


Figure B.3: Overfitted Model's FOC - STW curve: Case #1

As shown in Figures B.3, B.4, B.5 and B.6, Model #4 has overfitted the training data, resulting in curves that resemble the third graph in Figure B.2, which is a clear sign of overfit, or it produces curves that violate the physics of the problem.

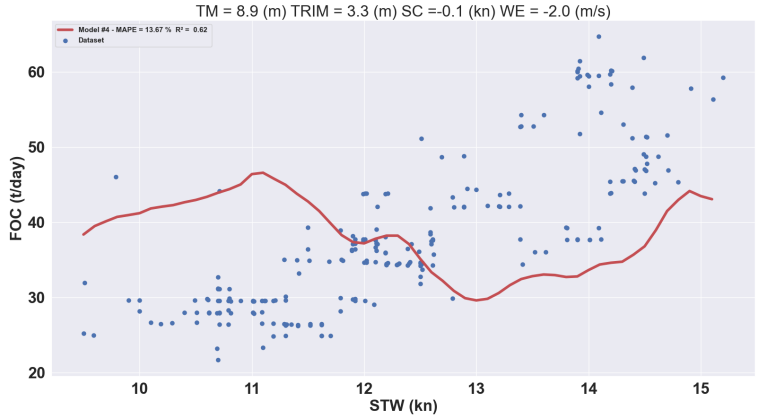


Figure B.4: Overfitted Model's FOC - STW curve: Case #2

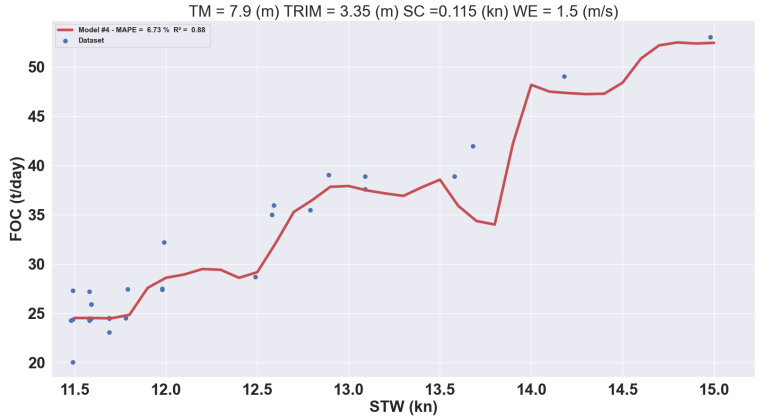


Figure B.5: Overfitted Model's FOC - STW curve: Case #3

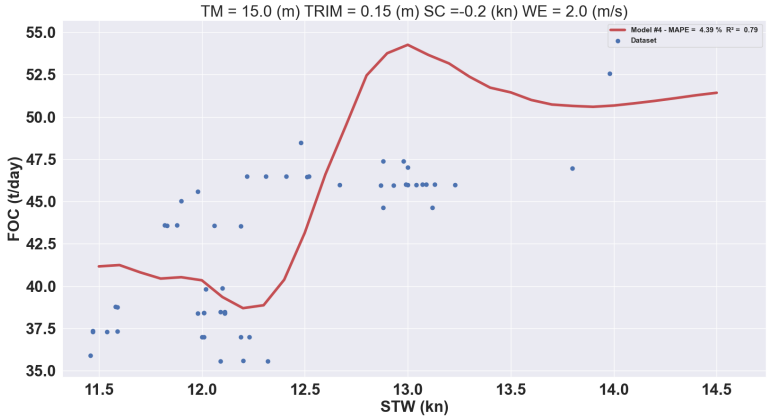


Figure B.6: Overfitted Model's FOC - STW curve: Case #4

B.3 Overfitting deflection

The problem of training a deep neural network that is capable of generalizing well to new data is a difficult topic, because a model with insufficient capacity will be

unable to learn the problem, whereas a model with excessive capacity will learn it extremely well and will overfit the training dataset. Both cases result in a model that does not generalize well.

A modern approach to minimizing generalization error is to use a larger model, which may require the use of regularization during training to keep the model's weights small. These techniques not only reduce overfitting, but they can also result in a faster model optimization and improved overall performance.

Regularization is a technique that modifies the learning algorithm slightly in order to improve the model's generalizability. This also improves the model's performance on unseen data. In machine learning regularization penalizes the coefficients, whereas in deep learning it penalizes the weight matrices of the units. There are a few different techniques in order to apply regularization in deep learning:

- **L1 & L2 Regularization**: The most frequently used types of regularization are L1 and L2. Both L1 and L2 Regularization impose a penalty on the error function. The penalty in L1 is calculated by multiplying the absolute values of the weights by the tuning parameter λ, which specifies how much we want to penalize the model, and the penalty in L2 is calculated by multiplying the sum of the squared values of weights by the tuning parameter λ. L1 regularization will produce simpler models with fewer parameters, since it provides consistent rewards to reduce weights values (weights set to zero), whereas L2 regularization will produce more complex models with weights near but likely not at zero, since it provides diminishing rewards to reduce weights values. In L2 the model it is able to learn more complex data patterns. L2 regularization is usually recommended in neural networks because it acts as a guardrail but doesn't interfere too much with the complex workings of neurons.

- **Dropout**: Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network. More specific, is a technique where randomly selected units are ignored during training. This means that their contribution to the activation of downstream units is temporally removed on the forward pass and any weight updates are not applied to the unit on the backward pass. Dropout is implemented simply by randomly selecting units to be dropped out with a specified probability p (e.g. 20%) during each weight update cycle. Dropout is used exclusively during the training phase of a model and is not used to assess the model's skill.

- **Early stopping**: Early stopping is a technique for bringing the training procedure to a halt when the loss on the validation dataset begins to increase and the loss on the training dataset begins to decrease (in the case of minimizing the loss). This technique, however, is not always applicable because overfitting does not always manifest itself in learning curves, as proved with Model #4.

We tried each of the aforementioned regularization techniques, but L2 regularization with the tuning parameter $\lambda = 0.01$ appeared to be the most effective technique and it was apllied in the Model #3 from Chapter 5, which was selected as the final

ANN model. Thus, in the following figures, Figures B.7, B.8, B.9 and B.10, we will compare Model #4 (in red) with the Final ANN Model (in green) in the generation of FOC - STW curves.

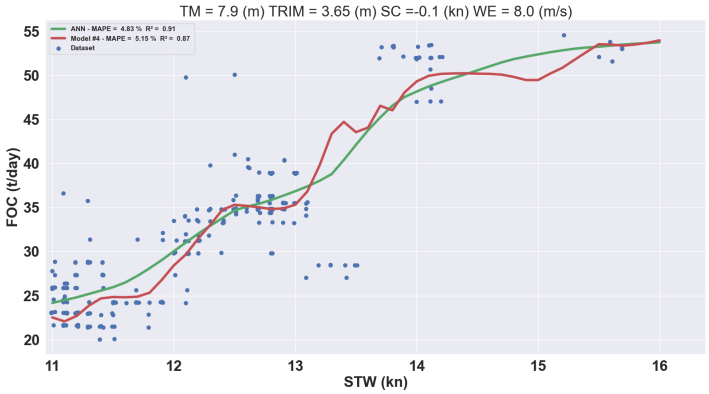


Figure B.7: Overfitted vs Balanced Model FOC - STW curve: Case #1

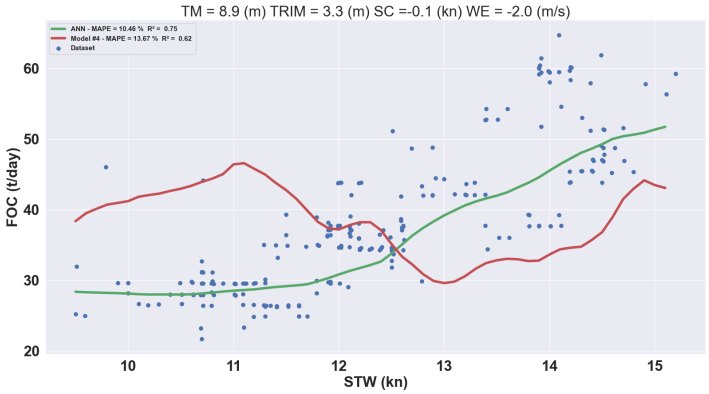


Figure B.8: Overfitted vs Balanced Model FOC - STW curve: Case #2

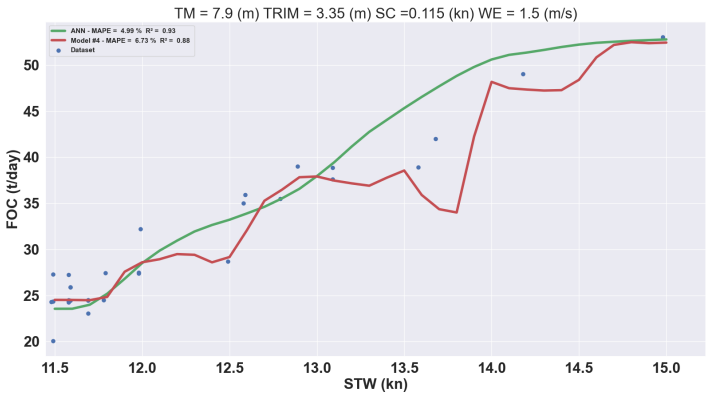


Figure B.9: Overfitted vs Balanced Model FOC - STW curve: Case #3

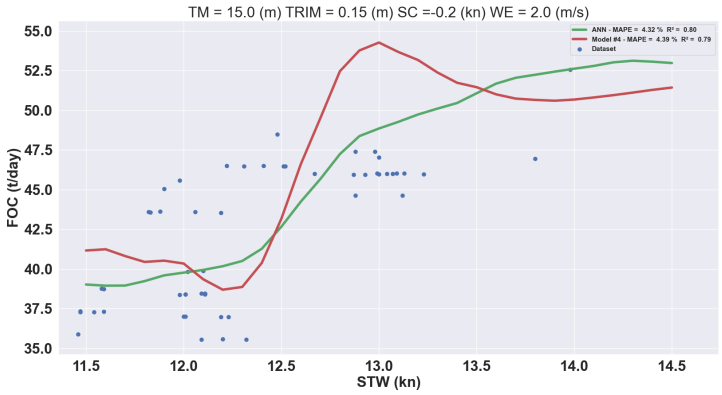


Figure B.10: Overfitted vs Balanced Model FOC - STW curve: Case #4

As illustrated in the figures above, regularization enabled the final ANN model to capture the problem dynamics quite effectively while maintaining a relatively low error.