Ocean Engineering 243 (2022) 110321

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng

Predicting VLCC fuel consumption with machine learning using operationally available sensor data

Christos Papandreou [a], Apostolos Ziakopoulos [b,*]

[a] Fleet Performance, Athenian Sea Carriers, 10-12 Kifissias St., GR-15125, Amarousion, Athens, Greece
[b] National Technical University of Athens, Department of Transportation Planning and Engineering, 5 Iroon Polytechniou St., GR-15773, Athens, Greece

ARTICLE INFO

Keywords:
Fuel oil consumption prediction
VLCC sensor Data
Machine learning
Polynomial regression
Artificial neural network
XGBoost regression

ABSTRACT

In the maritime industry, more accurate predictions of fuel oil consumption (FOC) could yield multidimensional results including more precise bunker calculations, emission reductions, more informed planning and limiting operational costs. However, models often require sophisticated data that may be partially unavailable to operators beforehand. The present research aims to develop accurate main engine FOC forecasting models that utilize exclusively data from sensors and simple weather data readily available in operational practice. Commonly available sensor data from a Very Large Crude Oil Carrier (VLCC) were used, comprising speed through water, relative wind direction, relative wind speed, mean draft, trim, days since last drydock and laden or ballast vessel state. Multivariate Polynomial Regression (MPR), Artificial Neural Networks (ANNs) and eXtreme Gradient Boosting (XGBoost) regression models were developed and evaluated based on their predictive accuracy for VLCC FOC. Results indicated that XGBoost had the best performance, yielding predictions within 5% of the true value more than 86% of the total cases, followed by MPR and ANN. In addition, accurate aggregate FOC forecasting was conducted with XGBoost for a laden voyage and a ballast voyage of a VLCC.

1. Introduction

Recently, the International Maritime Organization (IMO) introduced long-term strategies in an attempt to reduce the environmental footprint of maritime operations. To that end, the maritime industry has to commit to reducing its carbon intensity (CO_2 emissions per transport work) by at least 40% by 2030 compared to 2008 values, and pursuing efforts towards 70% reductions by 2050. Furthermore, the total annual greenhouse gas (GHG) emissions from international shipping have to be reduced by at least 50% by 2050, compared to 2008 values (International Maritime Organization IMO, 2018; Joung et al., 2020). According to the third IMO GHG study in 2014, maritime transport emissions amount for 2.5% of the total greenhouse gas emissions or, equivalently, 940 million tons of CO^2 annually. Inaction through a business-as-usual scenario could lead to an increase of shipping emissions of up to 250% by 2050 (Smith et al., 2014). It has been established by the relevant literature that immediate and rapid exploitation of available mitigation measures is critical to align with the goals set by the Paris agreement (Traut et al., 2018).

Apart from environmental benefits from global reductions, for every maritime operator, the identification of improvement margins in operating cost savings in the maritime industry is a constant pursuit. As Ronen (2011) states in a widely cited study, large vessel daily fuel consumption cost may exceed 100,000 USD daily, amounting to 75% of the operating costs. Macroscopically, fuel costs have displayed an increase of approximately 400% from 2000 to 2014 (International Chamber of Shipping, 2014). Therefore, timely and precise (re)fueling of vessels without the need of increased redundancies is the path to suppressing operating costs. To that end, accurate predictions in ship fuel consumption estimations can potentially save considerable amounts of money for ships of the aforementioned magnitude. Moreover, precise fueling can allow operators to take advantage of short fluctuations in fuel oil prices caused in the global market by any number of unexpected events.

These obvious benefits of more accurate forecasting for the estimation of fuel oil consumption (FOC) are beset by practical limitations. As the literature review below shows, researchers have created powerful models that exploit intricate sets of data to deliver very precise results. These models often include engine power (Jeon et al., 2018), which is the quantity FOC is physically mostly correlated with, revolutions per minute (RPM) (e.g. Ahlgren et al., 2019), and detailed weather quantities, such as wave/wind encounter angle or wave significant period

* Corresponding author.
E-mail addresses: chrispapobs@gmail.com (C. Papandreou), apziak@central.ntua.gr (A. Ziakopoulos).

https://doi.org/10.1016/j.oceaneng.2021.110321
Received 22 February 2021; Received in revised form 30 June 2021; Accepted 4 December 2021

(Farag and Ölçer, 2020). Very few large future-proof shipping companies with highly advanced internal performance departments currently invest in data featuring sophisticated weather parameters. At present, smaller or average-size shipping companies are typically not interested in investing to acquire such data. Furthermore, a number of performance software platforms do not provide detailed data either. Moreover, for smaller or average-size operators, weather models with sophisticated weather parameters may not be highly accurate days or weeks before the forecast time. Therefore, FOC forecast accuracy is limited by these limitations.

As a solution to that predicament, the aim of the present research is to provide accurate models that utilize exclusively data from sensors and simple weather data readily available in operational practice. Commonly available sensor data from a Very Large Crude Oil Carrier (VLCC) are used, and Multivariate Polynomial Regression (MPR), Artificial Neural Networks (ANNs) and eXtreme Gradient Boosting (XGBoost) regression models are developed and evaluated based on their predictive accuracy for VLCC FOC. As a note, the total FOC in a vessel mainly comprises of Main Engine (ME) FOC, Auxiliary Engine (AE) FOC and Boiler FOC. ME accounts for large percentages of the total consumption for VLCCs, while the other FOCs are more steady with their regular loads (Fan et al., 2021), and thus easily predicted. Therefore, the present research focuses exclusively on ME FOC prediction. Therefore, ME FOC is referred to simply as FOC in this paper.

The paper is organized as follows: in the next section, relevant literature is examined in terms of methodologies and utilized data. Afterwards, the methodological framework of the study is presented, comprising of an overview of the mathematical background of the implemented methods. The data collection and processing steps are then presented. Subsequently, the results of fitting the three methods in the data and the respective evaluation metrics for predictions on unseen (completely new) data are presented. Results are compared and discussed. The paper concludes with a summary of the present research results and their comparison with the results extracted from previous studies.

Models based on data from a second VLCC were trained and showcased in the Appendix as further validation of the present approach.

2. Literature review

The literature related to data-driven models for the ship FOC prediction typically focuses on creating predictive frameworks that utilize single or multiple models or algorithms. Traditional approaches involved using cubic functions such as the one proposed by Ronen (1982) to link vessel speed with FOC as well as emissions, at least above certain limits.

More recently, regression approaches have become increasingly intricate. One such example is LASSO (Least Absolute Shrinkage and Selection Operator) regression for FOC prediction as implemented by Wang et al. (2018). LASSO regression circumvents multicollinearity problems that can arise from correlated data in traditional multiple linear regression by augmenting the variable selection process. The authors conducted trials in which LASSO regression surpassed popular machine learning methods such as ANNs, Gaussian Process (GP) and Support Vector Machine (SVM) Regression. It is worth noting that the data utilized in this study were of low frequency and originated from a fleet of container ships.

The underlying relationships between speed & FOC and speed & engine power have also been examined within a Bayesian framework. Levantis et al. (2020) used a GP regression model correlating FOC with a product of speed through water (STW) and mean draft exponentials. The model was trained on data describing a year of operation of a specific VLCC and validated on a subsequent year on data from the same vessel. Regarding the FOC model, among other observations, the authors note that periods with high wind speed (i.e. $>$ 16kn) increase the respective prediction error.

Lately, machine learning (ML) models have risen to prominence in many quantitative disciplines, largely due to continuous technological advancements in hardware performance, data collection and transfer, software and algorithmic developments etc. ML models are data-driven models that learn without explicit programming but rather from experience accrued by the algorithms as they parse through datasets. The ML field has been developing rapidly owing to its cross-sectional nature, combining statistics, computer science, data science and artificial intelligence (Jordan and Mitchell, 2015). These trends have also been mirrored in the maritime industry.

ML algorithms can be automated, as shown by Ahlgren et al. (2019). In that study, the authors designed algorithms which were trained on data from short-time intervals of engine features including engine fuel rack position (FRP), engine RPM, turbo RPM and turbo exhaust temperature and aggregated fuel sum features from long-time intervals for FOC predictions. The authors utilized an automated supervised ML framework developed in a Python environment, resulting in a SVM Regression optimum. The study concludes that available operation data can augment noon report data for the prediction of FOC instantaneously.

A similar approach was adopted by Jeon et al. (2018), who compared the performance of Polynomial Regression (PR), SVM and ANN methods for the FOC prediction of a specific vessel. The input variable groups comprised variables from three domains: engine operations (including main engine power and shaft speed – RPM), navigation speed, and weather conditions. The ANN outperformed the other models in terms of accuracy, and investigation of various configurations indicated that the sigmoid and tangent sigmoid functions required less hidden layers and neurons, in contrast with the Rectified Linear Unit (ReLU) function.

These studies showcase excellent methodological approaches, however, the sophistication and unpredictable nature of engine performance data would make long-term predictions with such models difficult or less accurate than their real-time predictions. This is showcased in relevant research examining high-quality sensor data over two months for a ferry vessel. Both a Gaussian Process based on a functional probabilistic formulation and ANNs were employed to model ship fuel efficiency in both an instantaneous (real-time) and predictive settings. While the GP model showcased good capabilities in quantifying uncertainty, it was very sensitive to starting conditions and scaled poorly, and the ANNs displayed better performance overall (Petersen et al., 2012a, 2012b).

On a similar note, Uyanik et al. (2020) implemented an array of 15 different modelling techniques, including functional models and ML models, for FOC predictions for the main engine of a container type vessel. Various specialized engine performance parameters were utilized as input variables, including main engine power, fuel flow, exhaust outlet temperature, fuel oil inlet pressure, temperature and viscosity and others. Results indicated that ridge regression models (namely Bayesian Ridge, Kernel Ridge, Multiple Linear and Ridge Regressions) outperformed other tested methods. The authors note that their models are useful for engine performance observation by monitoring FOC, thus comparing performance to model predictions, so that abnormal situations can be detected earlier.

Apart from the degree of mathematic sophistication of each approach, it is well established that the quality of the modelling outcome greatly depends on the quality of the utilized data (Walther et al., 2016). Data sources of different origins – and thus, different qualities – can be considered to lead to different performance, a hypothesis examined by Gkerekos et al. (2019). The authors used noon report data and automated data logging & monitoring system data to train various ML models for FOC prediction including SVMs, Random Forest Regressors (RFRs), Extra Trees Regressors (ETRs) and ANNs. The study concluded that the automated data collection system increased accuracy by 7% compared to noon report data, while the ETR, ANN and RFR algorithms had the best performance overall. Nevertheless, this study uses performance data that are, due to their nature, a priori unknowable, such as engine RPM and propeller slip. Operators and researchers may be urged to transition to higher frequency data, as recent research indicates that

considerable improvements can be gained in terms of predictive power of FOC prediction algorithms such as ANN. Obtaining automated data has been found to require little extra instrumentation onboard with low cost (Petersen et al., 2020).

Comparable ML approaches have generalized from single-vessel applications to exploiting data from larger fleets in the predictive models, such as the aforementioned research of Wang et al. (2018). In a recent study, Le et al. (2020) exploited operational data from hundreds of container ships from a major shipping company in Korea and predicted FOC for five container ships of different size in a multiple-input-multiple-output ANN framework; comparisons with PR models were also conducted. Additionally, the scope of that study mainly concerned the investigation of the effect of cargo on FOC, and the final training features were selected based on the scope but also on the available statistical distributions. It is worth noting that the authors were mindful when balancing calibration time with hyperparameter tuning, which is an issue often overlooked by researchers. Results did not yield very high precision for container ship operations, which is reasonable given the large scale of that study.

The research that is arguably closest to the present work is that of Farag and Ölçer (2020). The authors applied PR and ANN methods in conjunction. Specifically, an ANN was employed to estimate an engine's break power, in a concept mirroring the latent variables of Structural Equation Models (SEM). Subsequently, the calculated break power was used in a PR model measuring FOC. FOC predictions showed a high accuracy when conducting voyage forecasts. Once again, the authors used quite sophisticated weather data such as wave & wind encounter angle, true swell/current angles and wave significant period that augmented model accuracy.

In the ML field, XGBoost has risen to prominence for being a highly performing, fast and adaptable methodology for a number of classification and regression tasks. XGBoost has not yet been implemented for FOC predictions in the maritime domain, to the knowledge of the authors, despite the fact that it has been successfully used for fuel or energy consumption estimates in other domains.

Indicatively, Chen et al. (2018) considered an ensemble model comprising of several (10) XGBoost models and a feedforward ANN, all combined with ridge regression, in order to predict household electricity consumption. Feature importance estimates were provided via information gain, and the proposed model ensemble outperformed classical regression models with 30% error reductions; in addition, XGBoost had the best individual model performance. Wang et al. (2021) endeavored to predict the fuel consumption of mining trucks with several ML methodologies, including XGBoost. Initially, MPR models were employed to showcase the non-linear effects of multi-dimensional features on fuel consumption. After training several ML models, it was shown that XGBoost had reduced errors compared to Random Forests (RF), ANN and k-nearest neighbors (KNN), while it reached similar performance with an SVM. XGBoost provided increased interpretability compared to SVM, which led the authors to select it as their model of choice for mining truck fuel consumption predictions.

XGBoost models appear to reach good performance with a limited amount of features. Zhang et al. (2018) utilized XGBoost for fault detection in wind turbines. Specifically, the authors used Random Forest classifiers to rank sensor signal features and constructed features by importance and then train XGBoost classifiers for fault detection. Despite utilizing only the top three features, the outcome appears robust and resilient to overfitting when compared with a SVM. Ma and Yan (2019) utilized XGBoost for energy consumption estimations of electric vehicles, based on input from the respective battery management system. Model performance stabilizes with five or more input features, while the authors report that their model meets working demands. Therefore, urged by these promising recent trends, it is highly interesting to test the performance of XGBoost for FOC predictions in the maritime domain.

As per the aforementioned, model performance is greatly affected by data quality. However, for industrial operators working in practical applications, data attainability is as critical as their quality. Both of these facts provided an impetus for combining well-established methods (such as PR) and state-of-the art ML techniques (such as ANN and XGBoost) with reliable, readily available sensor data used in operational practice for FOC predictions.

3. Methodological approach

In this section, a brief outline of the methodologies utilized in this paper is presented.

3.1. Multivariate Polynomial Regression

Polynomial regression (PR) belongs to the family of traditional multivariate linear regression analyses. In this type of regression, the dependent variable, y, is regressed on n-th degree polynomials of a single independent variable, x. This functional relationship can be expressed by the following equation:

$$y = f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n \quad \text{Eq. (1)}$$

Univariate PR can be extended to Multivariate Polynomial Regression (MPR) to include several independent variables, x_i. This functional relationship can be expressed by the following equation:

$$y = f(x_i) = \beta_{00} + \beta_{11} x_1 + \beta_{12} x_1^2 + \ldots + \beta_{1n} x_1^n + \beta_{21} x_2 + \beta_{22} x_2^2 + \ldots + \beta_{2n} x_2^n \quad \text{Eq. (2)}$$

Such regression methods have been widely used in scientific or statistical contexts and the underlying mathematical structures are available in several sources (e.g. Kleinbaum et al., 2013). A major underlying problem with MPR is multicollinearity, in other words, the case where independent variables are linearly interdependent (Sinha, 2013). Multicollinearity issues lead to misleading model fits and have to be anticipated and accounted for by researchers accordingly. In order to counter such issues, most modern statistical software packages create polynomials with an orthogonal polynomial sequence from the independent variables (Shacham and Brauner, 1997).

3.2. Artificial Neural Network

Artificial Neural Networks (ANNs) are a widely used family of supervised ML methods seeking to emulate information processing in the way the human brain does. These structures have evolved from modelling relatively simplistic processes to massive, sophisticated structures with numerous elements and interrelations, used for applications in pattern recognition, classification and regression problems, data mining and other aspects of artificial intelligence. ANNs have been extensively reported as a promising ML method due to their ability to handle non-linear stochastic processes (e.g. Agatonovic-Kustrin and Beresford, 2000; Shu and Burn, 2004).

In their standard forms, ANNs consist of three main components: (i) Neurons, which receive one or more inputs and yield a single output. Neuron groups are arranged as subsequent units called layers. (ii) Connections, connecting neuron layers by providing the output of the antecedent layer as input for the following layer. Connections are assigned weights to express different relative importance. (iii) The propagation (or activation) function, which is the mathematical function computing outputs from the inputs and the connection weights as a weighted sum (Dawson and Wilby, 1998).

Following Samarasinghe (2006), mathematically, the fundamental process of an ANN can be expressed by the following equation:

$$u_i = \sum_{j=1}^{n} \left(w_{ij} x_j + b_i \right) \quad \text{Eq. (3)}$$

where w_{ij} is the weight value for layer i and neuron j, x_i is the input value of layer i, b_i is the bias weight of layer i and n is the total number of connected neurons. The neuron output, y_i, is then calculated as a function of the previous process $f(u_i)$:

$$y_i = \varphi(u_i) = \varphi\left(\sum_{j=1}^{n}\left(w_{ij}x_j + b_i\right)\right) \quad \text{Eq. (4)}$$

All layers between the input (foremost layer) and the output (last layer) are considered hidden layers. While there is no hard limit, typically ANNs with a large number of hidden layers are considered Deep Neural Networks (DNNs), which can be applied to supervised and unsupervised tasks and can also circumvent the necessity of feature engineering (Schmidhuber, 2015). There is evidence that increased layer number can increase model accuracy (Zhao et al., 2015), however, the opposite has also been reported in maritime research for FOC; Farag and Ölçer (2020) stated that more nodes and layers decreased model performance in many cases. It is evident that ANN architecture is malleable in principle and must be tailored to the examined problem on a case-by-case basis.

The theoretical structure of a simple ANN with a single hidden layer is shown on Fig. 1.

There are numerous mathematical functions available for implementation as activation functions in ANNs. It is apparent that the selection of a particular activation function influences model outputs. Within this research, the widespread Rectified Linear Unit (ReLU) function was adopted after several modelling attempts on the VLCC data as described in Section 5.2. ReLU is known to lead to rapidly converging ANNs; is a pseudo-linear function that does not become trapped in local minima and allows for unhindered backpropagation (Hara et al., 2015).

As with other ML methods, ANN implementation consists of two main phases: training and testing (a validation phase may also be interjected between the two). For the training phase, ANN structure is explored and finalized. Using the portion of the dataset used for training (training dataset), the weights are calculated, typically based on an equal division of error along the connections. This is a repetitive process across several epochs (iterations) that can often lead to considerable training durations. Due to the fact that the errors of the previous epoch are the basis of training, this process is called backward propagation (or backpropagation). For the training phase, the ANN elements are finalized and data travel from the input to the output layers in order to conduct predictions, in a process termed forward propagation.

To obtain the optimal ANN structure for each problem, parameters governing the model form, known as hyperparameters, are determined. Hyperparameters may include the number of neurons/layers, learning rate (defines how quickly parameters are updated), dropout rate (defines where random neurons are deactivated to avoid overfitting), number of epochs, and others. As they are unknown a priori, typical practice involves training several combinations of hyperparameters either manually or automatically, using grid (serial) or random search strategies.

3.3. XGBoost

EXtreme Gradient Boosting (XGBoost) is a supervised ML technique encompassing multiple Classification And Regression Trees (CART). Being an ML technique, it is inherently data-driven, and thus free of any prior assumptions concerning underlying relationships in the data. Any obtained relationships are products of what lies strictly within the provided dataset. XGBoost originated from the seminal work of Chen and Guestrin (2016), who expanded known tree-boosting techniques to handle sparse data, approximate problems for better memory handling

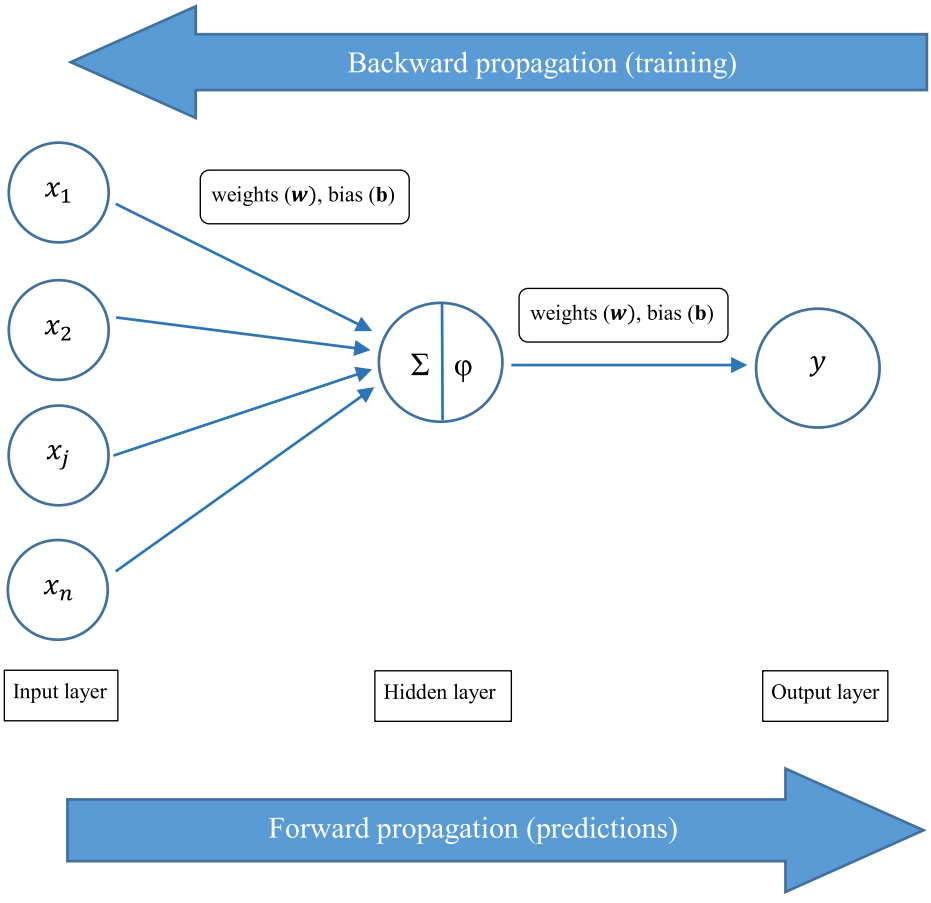

Fig. 1. Example of a simple ANN with a single hidden layer.

and ultimately be more scalable. Due to its adaptability and effectiveness, XGBoost has been consistently outperforming competitor methodologies in numerous ML competitions (Nielsen, 2016). The aforementioned properties of XGBoost urged the selection of this technique for an exploratory application in FOC prediction within the present study.

Apart from speed and scalability, a considerable comparative advantage lies in the iterative mechanisms of XGBoost. With every iteration, new trees are created and trained to fit the residual errors of previous iterations. This has allowed the algorithm to integrate corrections of previous errors in future iterations of the algorithm (Dong et al., 2020), allowing it to outperform other methodologies. XGBoost can be applied in both classification and regression problems.

An overview of the algorithm with more detailed explanations is described in Chen and Guestrin (2016). In short, if a mapping function is considered between variables:

$$\hat{y} = f(x_i) \quad \text{Eq. (5)}$$

where y is the dependent (or response) variable, $\hat{y}$ is the predicted value of the dependent variable and x_i are the independent (or explanatory) n variables across i observations, then a regression tree ensemble model uses a number of functions K additively to predict y, so that:

$$\hat{y} = \varphi(x_i) = \sum_{k=1}^{K} f(x_i) \quad \text{Eq. (6)}$$

Then the difference between a prediction $\hat{y}_i$ and a true value y_i can be measured via a loss function which is differentiable and convex, defined as $l(\hat{y}_i, y_i)$. In other words, the loss function expresses the distance between predictive values and the training data. A common choice of l is the mean squared error for a set of parameters φ_i (XGBoost developer team, 2019):

$$l(\varphi_i) = \sum_{i=1}^{I} (\hat{y}_i - y_i)^2 \quad \text{Eq. (7)}$$

Furthermore, a penalizing term, $\Omega(f)$, is introduced for model complexity control such that:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda c^2 \quad \text{Eq. (8)}$$

where γ, λ are penalizing coefficients, T is the number of leaves in the regression tree. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf, creating a flowchart, and c is the weight assigned to each leaf. Fig. 2 depicts an example of a single decision tree from the ensemble utilized in XGBoost, and the respective components are labeled. Here, the input is a single typical ship variable, speed through water (*stw*), and the output is the observed FOC over the course of one measuring interval, FOC_{int} (numbers are hypothetical). In this example, the defining value of speed through water is 10 Kn and the different results are represented in the tree leaves.

In this theoretical example, if *stw* exceeds the threshold of 10 Kn, the tree increases the prediction of FOC, and reduces it in the opposite case. Having obtained the loss function, $l(\hat{y}_i, y_i)$, and the penalizing term, $\Omega(f)$, the objective function can be formulated as:

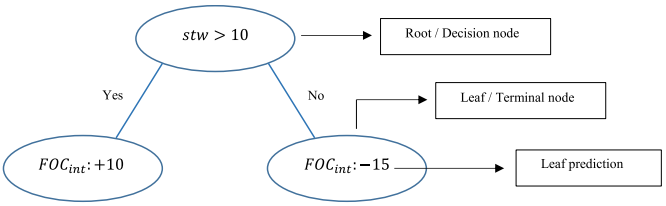

Fig. 2. Example of a single decision tree from an XGBoost ensemble.

$$L(\varphi_i) = \sum_{i=1}^{I} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f) \quad \text{Eq. (9)}$$

Chen and Guestrin (2016) note that since nested functions exist inside the objective function, traditional optimization methods cannot be used. They circumvented this obstacle by using second-order approximation from Taylor's Theorem to transform L to a function in the Euclidean domain, so that optimization methods can be now used. The function ensemble then reverts to simple quadratic functions that can be minimized normally. The exact mathematical formulae from this point onward depend on the structure of the loss function.

Similarly with ANN, XGBoost involves a phase for hyperparameter tuning. Some of the typical hyperparameters that can be tuned are:

- Learning rate (also known as eta), governing the magnitude of iterations for minimizing the cost function
- Gamma, governing the minimum loss reduction that can justify making a partition on a tree
- Maximum tree depth, greatly governing ensemble complexity and overfitting
- Evaluation metric, which is a target for minimization such as RMSE, RMSLE, MAE and others
- Number of k-folds for each cross-validation task
- Number of rounds that are tested before convergence of the cost function is finalized

Therefore, following good ML practices, the hyperparameters of XGBoost algorithms should initially be tuned before their final executions, and their predictions should subsequently be evaluated.

3.4. Model evaluation metrics

As per standard practice, common evaluation metrics are used uniformly to measure and assess model performance of models obtained by the three aforementioned methods. Let n be the number of data points, y_i be the true value of the dependent variable and $\hat{y}_i$ be the predicted value of the dependent variable.

Then the Mean Absolute Error (MAE), also known as Mean Absolute Deviation (MAD), is defined as:

$$MAE = MAD = \frac{1}{N}\sum_{i=1}^{n} |y_i - \hat{y}_i| \quad \text{Eq. (10)}$$

The mean squared error (MSE) is defined as

$$MSE = \frac{1}{N}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad \text{Eq. (11)}$$

Respectively, the square root of MSE is the RMSE:

$$RMSE = \sqrt[2]{\frac{1}{N}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad \text{Eq. (12)}$$

The reason for researchers often preferring RMSE over MSE is that RMSE is a metric on a same scale as the dependent variable, instead of squared. This can be viewed as similar to preferring standard deviation over the variance of a variable.

In lieu of other known metrics, such as MAPE, another simple metric is hereby devised which bridges the gap between regression and classification metrics: custom accuracy (CA). The reasoning behind custom accuracy is to denote as acceptable each prediction that falls within an acceptable margin from the truth. As the term implies, custom accuracy grants an intuitive percentage of correct continuous predictions, classified as correct if they are within the custom threshold that the researcher selects (hence the term 'custom accuracy').

In the context of a continuous variable, such as FOC in the present research, predictions within 5% of the true value are accepted as

'accurate'. In mathematical terms, CA is expressed as follows; firstly the total number of predictions is classified as accurate or not accurate:

$$N_{\widehat{y_i},\,5\%} = \begin{cases} accurate, & if\ \dfrac{|y_i - \widehat{y_i}|}{y_i} \le 0,05 \\ not\ accurate, & otherwise \end{cases} \quad \text{Eq. (13)}$$

and then CA is obtained as the percentage of accurate predictions from the total:

$$CA_{5\%} = \frac{N_{\widehat{y_i},\,5\%}\,[accurate]}{N_{total}} \quad \text{Eq. (14)}$$

As CA is a percentage of accuracy, it receives values between 0 and 100%; higher values indicate models with better predictive ability. Researchers seeking higher precision could easily adjust the accuracy percentage, for instance to 1%, 0.1% and so on.

The traditional R^2 metric is reported for all models as well. Finally, when assessing model predictions it is recommended to examine the plot of predicted values against the true values of the predicted variable.

4. Data description and preprocessing

In this section, an outline of the utilized vessel data is provided, along with the preprocessing steps that were undertaken before model training.

4.1. Data collection from sensors

The data used for the training of the models described in this paper are high frequency automated data that originate from sensors onboard the vessel. The time interval between sensor measurements are a few seconds depending on the measured parameter. However, due to the amount of data created the sensor software averages these readings to 15 min time intervals. This creates a considerable volume of data which are adequate for model training, without entering the big data area where additional computational issues might arise.

Theoretically, the most critical inputs for creating a fuel consumption model are, among others, draft, engine power, RPM, speed through water, relative wind direction, relative wind speed, relative wave direction, relative wave height, depth, currents, trim, time since last drydock and time since last hull cleaning.

However, as discussed previously, the aim of the present research is to create FOC models that are based exclusively on operationally available data, easily obtained by shipping companies in daily practice. Therefore, inputs that cannot be easily predicted were considered only for data cleaning but not as inputs for the models. The expected tradeoff from this approach is some loss of accuracy in the models, as it is known that engine power and RPM are highly correlated with FOC. However, the intent of the authors is to have readily useable models for applied predictions.

Based on the previous, the final input variables utilized for the models are the following:

1. Speed through water
2. Relative wind direction
3. Relative wind speed
4. Mean draft
5. Trim
6. Days since last drydock
7. Laden or Ballast (binary format)

Normally, more conditions than days since last drydock can be used to describe the condition of a vessel. This parameter could be split in further categories to create different parameters for time since drydock, time since last hull cleaning and time since propeller polishing. However, due to the lack of hull cleaning and propeller polishing in the 5-year period of the examined vessel these parameters have been omitted and only days since last drydock are considered. As a note, the binary Laden/Ballast variable was retained mainly for computational reasons, as its inclusion was found to enable quicker model convergence during the numerous trials and iterations of the modelling process.

4.2. Data filtering

Automated sensors have shown great promise in recording data used for sophisticated ML models. However, various issues may arise in terms of accuracy, variations, signal noise, calibration necessity etc. In those cases, sensor data are compromised and need to be discarded. Proper data cleaning and filtering steps must be taken to ensure that only good observations remain in the dataset.

For the considered dataset, erroneous data (NaN, zeros etc.) were initially removed. Additional logical filters were applied to remove sensor measurement errors (i.e. inconsistent speed – FOC pairings, which yet remained within the acceptable range after initial filtering). FOC key performance indicators (KPIs) such as specific FOC (SFOC) are also monitored to remove erroneous values. SFOC is calculated as:

$$SFOC = \frac{10^6 * FOC}{24 * Power} \quad \text{Eq. (15)}$$

When FOC represents the ship daily fuel consumption in tonnes, and Power represents the power of the main engine of the vessel in KW.

The acceptable values of the data (barring logical filtering) are shown on Table 1:

4.3. Description of the examined vessel profile

The examined vessel is a VLCC (Very Large Crude Oil Carrier) which trades globally. After applying the previous variable selection and data cleaning steps, several descriptive statistics of the remaining data could be extracted and examined. Descriptive statistics of the utilized vessel data are presented on Table 2:

After data cleaning, the total number of available observations was 107,389 data points (rows). Additionally to the previous continuous variable data, the binary variable laden/ballast was considered. In 85,753 observations, the VLCC was laden while on the remaining 21,636 it was in ballast.

4.4. Dataset partition and standardization

Following common ML convention and good practices (e.g. Bishop, 1995; James et al., 2013), the resulting cleaned dataset is partitioned in three subsets:

i. The training dataset: used for the fitting process, also known as model learning

Table 1
Acceptable values for the utilized VLCC data.

Parameter	Min acceptable value	Max acceptable value
Speed over ground [Kn]	1	20
Speed through water [Kn]	1	20
Speed through Water – Speed over Ground [Kn]	−2	2
Relative Wind Speed [Kn]	0	60
Mean Draft [m]	4	25
Trim [m]	−10	10
RPM [rev/min]	1	90
Power [KW]	1000	30,000
FOC (Main Engine) [tn]	10	120
SFOC [g/KWh]	150	250

Table 2
Descriptive statistics for the utilized VLCC data.

Vessel parameter	Descriptive statistics						
	Average	Min	Q1	Median	Q3	Max	St. Dev.
Speed through water [Kn]	12.8	1.0	12.2	13.0	13.4	17.2	1.1
Relative Wind Direction [Degrees]	169.1	0.0	31.4	125.3	323.9	360.0	136.8
Relative Wind Speed [Kn]	17.1	0.0	8.9	16.2	24.6	59.5	9.7
Mean Draft [m]	18.5	8.6	19.2	20.8	21.3	22.2	4.4
Trim [m]	−1.0	−5.2	−1.3	−0.8	−0.3	0.9	1.1
Days since last dry dock [number of days]	875.1	5.0	401.5	852.1	1353.3	1791.6	536.4
FOC (Main Engine) [tn]	61.3	10.1	46.1	63.4	72.3	119.0	15.1

ii. The validation dataset: used to provide an unbiased evaluation of model fit on the training dataset while tuning model hyperparameters
iii. The test dataset, can be considered as the 'unseen' dataset: used to conduct unbiased predictions of final models and check their accuracy with data never encountered by the models before

This partition is completely random, and seed functions were used to ensure the same random results between trials of different model configurations. Identical datasets were used for training, validation and testing for the MPR, ANN and XGBoost models. Both the Python library (train_test_split library function of sklearn.model_selection package) and the R-studio package (caTools) create the subsets by randomizing the partition of the initial set while ensuring that the dependent (target) variable maintains similar distributions in each subset. Regardless, each input variable was inspected so as to ensure that the test and validation partition subset lied within the limits of the training partition.

For this study, the training dataset was a randomized 80% of the original dataset, amounting to 85,911 data points (rows). The validation dataset was a randomized 10% of the original dataset, amounting to 10,739 data points. Naturally, the test dataset was the last remaining randomized 10% of the original dataset, amounting to 10,739 data points as well.

Subsequently to the partition, the training dataset undergoes standardization. Dataset standardization, also known as feature scaling or normalizing, is a process involving mathematical adjustments to produce datasets where all variables have normalized ranges. This process has been integrated in ML practices due to gains in computational speeds and ease of convergence (Ioffe and Szegedy, 2015) thanks to the comparable orders of magnitude of the normalized variables. In this research, the popular standardization (z-score normalization) was adopted. For each variable, x_i, the mean, μ_i, and standard deviation, σ_i, were calculated. Each standardized variable, x_i', is obtained by subtracting the mean from the original values and dividing the difference by the standard deviation.

$$x_i' = \frac{x_i - \mu_i}{\sigma_i} \quad \text{Eq. (16)}$$

These quantities, known as scalers, are retained from the training dataset and used for the scaling of validation and testing datasets afterwards. This transfer occurs so that new data do not require new information to be scaled, and that the scalers do not receive 'unseen' test data.

5. Results

In this section, the developed models MPR, ANN and XGBoost are presented and their results and performance are discussed. All ML models were developed in both Python (with Spyder and the sklearn package) and R-studio (mlr package) integrated development environments (IDEs) independently. During performance evaluation, only small decimal differences were observed between models, which is expected and can be attributed to different core coding and different random partition of the datasets; in this section, the results from Python are reported. Results from the identical process run with data of another vessel are provided in the Appendix.

Model predictions are conducted using the test dataset, with data not encountered previously by the models; values are converted to non-normalized values for easier comprehension of the respective plots.

5.1. Multivariate Polynomial Regression

For the MPR model, an eighth (8th) degree polynomial regression model, in other words, including degrees 1 to 8 for each of the seven variables outlined in section 4.1, plus an intercept, was fitted on the training dataset (using the Polynomial Features package of Python). Since MPR is a functional method, no phase was required for hyperparameter tuning. Nonetheless, the validation set was not used to ensure uniform training for all models.

The resulting MPR model was found to yield overall good fit and predictions from the scatterplot examination. However, there were scarce instances (<0.5% of the total test dataset) for which the high exponentials led to unreasonable predictions ($FOC_{pred} > 150$ t/d or negative). The presence of these values led to exceedingly high error metrics, due to the explosive behavior of said exponentials. Indicatively, the initial MPR model evaluation returned metric values indicative of subpar performance, specifically: RMSE = 250.80, MAE = 8.67, $CA_{5\%}$ = 73.78% and $R^2 = 0.002$.

Obviously, these few outlier predictions were not representative of the overall behavior of the MPR model. In operational practice, the MPR could still be used for predictions with one simple correction: Any extreme predictions (i.e. those outside the [0, 150] range for FOC) could be removed by the operator. To showcase the MPR model capabilities, the final model evaluation was conducted after that correction, which yielded the corrected MPR model, MPRc.

The final MPRc model evaluation returned metric values as follows: RMSE = 4.65, MAE = 2.44, $CA_{5\%}$ = 74.05% and $R^2 = 0.903$. Based on the test data, the plot of MPRc predicted values contrasted with the true values for main engine FOC is depicted on Fig. 3.

Typically, caution is advised against using too high-order polynomials in MPR, in order to avoid cases of model overfitting. However, the good performance on the test data (and also consistently on the second vessel data shown in the Appendix) indicates that the model has good overall fit. The R^2 diagram of Fig. 4 provides another illustration of the MPRc results obtained for the test dataset.

These results, although not as accurate as the more sophisticated XGBoost model, can actually be utilized to some extend to performance monitoring evaluation of the vessel and decent predictions of main engine FOC. The practical value of a decent MPR model, such as the present, increases further if computational time and resource considerations are taken into account: MPR was by far the quickest model to train from start to finish and be ready for predictions compared to the two ML models.

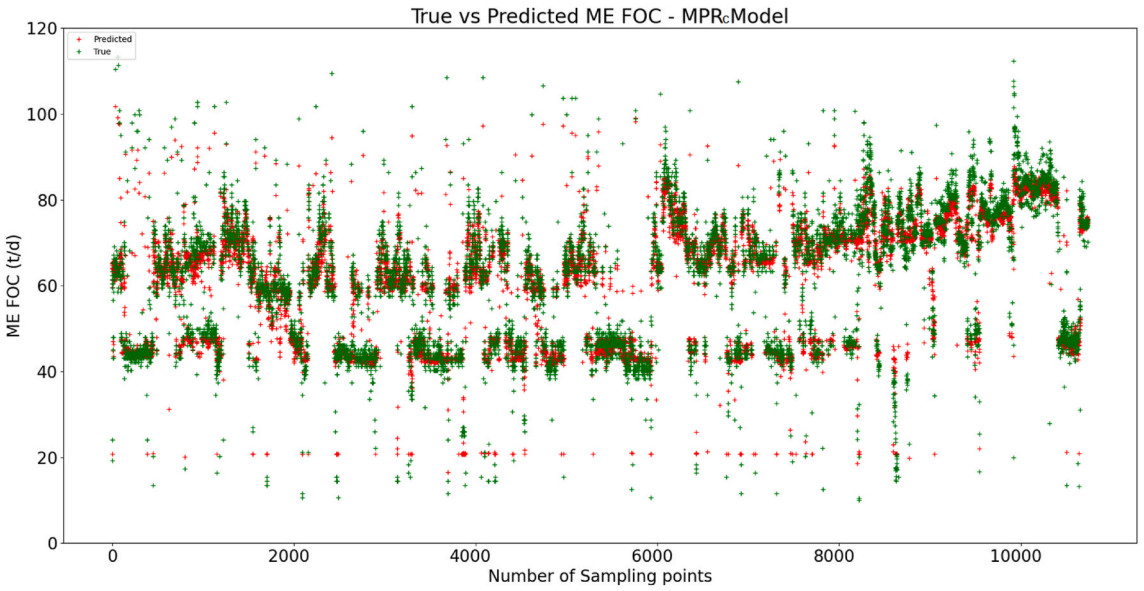

Fig. 3. MPRc scatterplot of predicted FOC values vs true FOC values for the test (unseen) data.

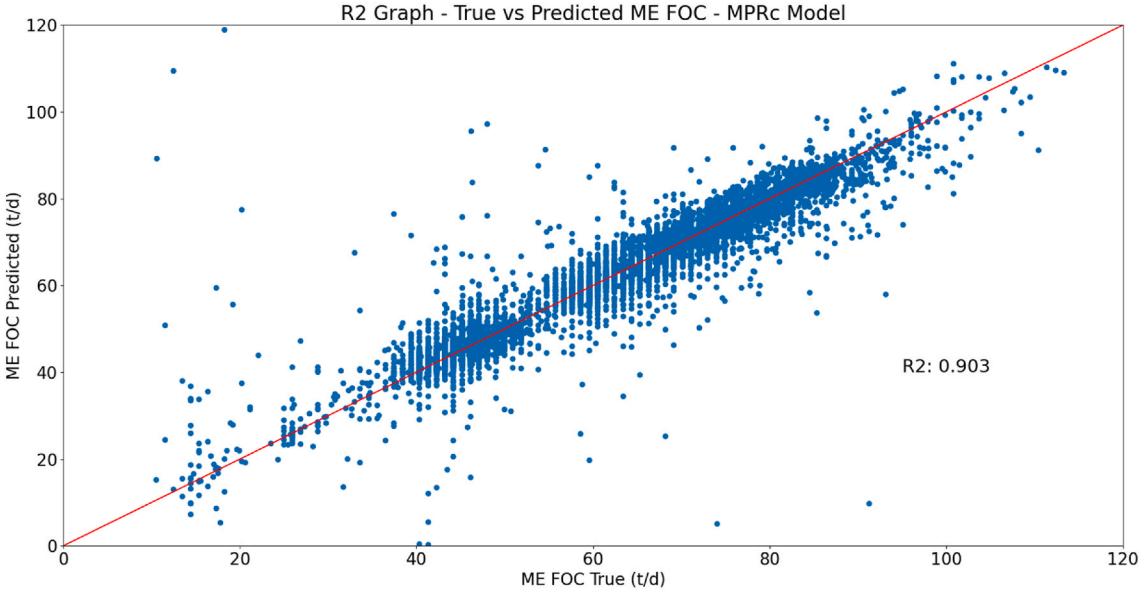

Fig. 4. MPRc true values by predictions.

5.2. *Artificial Neural Network*

In addition to the previous steps, the ANN model required a number of specific additional steps to properly configure, outlined briefly as follows:

1. The (filtered and scaled) training dataset is loaded.
2. A sequential ANN model with dense and dropout layers is created. Regarding the technical details, on Python this is created through the Keras deep learning API and the KerasRegressor wrapper for compatibility with the hyperparameter tuning algorithm.
3. A Design of Experiment (DoE) was created aiming to determine the range of each ANN hyperparameter.
4. A technique that creates random hyperparameter combinations was employed (Python library sklearn) to create 1,000 combinations for ANN parameters.
5. These combinations underwent 5-fold cross-validation and then their performance on validation data was compared.
6. The model with the optimal performance was evaluated on the test data

The utilized loss function of the ANNs was MSE based. Four additional hyperparameters were initially considered: (i) number of hidden layers, (ii) epochs, (iii) optimizer & (iv) activation function. These hyperparameters were not randomly determined during hyperparameter tuning; instead, they were fixed beforehand. After several early modelling attempts it was determined that an ANN with 10 hidden layers, 200 epochs, the Adam optimizer and the ReLU activation function yielded consistently better results than any reduction (of number of hidden layers or epochs) or variation (of optimizer or activation). Therefore, the decision to keep ReLU and these hyperparameters fixed was made, a decision which would aid in decreasing the overall

Table 3
ANN hyperparameter range and optimal values.

Hyperparameter	Lower Limit	Upper Limit	Optimized Value
Batch size (in a power of 2)	32	256	64
Learning rate	0.0008	0.0048	0.0025
Adam Opt Beta 1	0.8700	0.9300	0.8991
Adam Opt Beta 2	0.9970	0.9999	0.9983
No of neurons at layer 1	12	20	20
No of neurons at layer 2	14	28	22
No of neurons at layer 3	22	38	36
No of neurons at layer 4	20	50	34
No of neurons at layer 5	26	60	60
No of neurons at layer 6	30	68	44
No of neurons at layer 7	30	72	52
No of neurons at layer 8	30	72	34
No of neurons at layer 9	26	54	36
No of neurons at layer 10	20	50	46
Dropout percentage at layer 1	1%	5%	1.410%
Dropout percentage at layer 2	1%	5%	1.700%
Dropout percentage at layer 3	1%	5%	1.450%
Dropout percentage at layer 4	1%	5%	3.860%
Dropout percentage at layer 5	1%	5%	1.020%
Dropout percentage at layer 6	1%	5%	2.380%
Dropout percentage at layer 7	1%	5%	1.860%
Dropout percentage at layer 8	1%	5%	1.350%
Dropout percentage at layer 9	1%	5%	1.360%
Dropout percentage at layer 10	1%	5%	3.470%

parameters of the problem that required determination.

The remaining randomly tested ANN combination hyperparameter range and obtained optimized values appear on Table 3:

When the best-performing ANN hyperparameter combination was determined, the model was re-ran from scratch with an even larger number of epochs (1000 epochs) to detect any further gains in performance. The optimal performance was obtained in 270 epochs. It should be noted that, due to extremely high computational times, it would be infeasible to run such a high number of epochs during the tuning phase.

The final ANN model evaluation returned metric values as follows: RMSE = 3.87, MAE = 2.47, $CA_{5\%}$ = 73.40% and R^2 = 0.9412. Based on the test data, the plot of ANN predicted values contrasted with the true values for main engine FOC is depicted on Fig. 5.

It should be noted that the ANN model was the most intense computationally, taking approximately forty (40) times more than that of XGBoost for an equal number of models. The R^2 diagram of Fig. 6 provides another illustration of the ANN results obtained for the test dataset.

The ANN model has outperformed MPRc in regards to R^2, however it has scored a slightly lower CA value. This ostensible discrepancy is due to the mathematical structure of the two performance metrics. R^2 is prone to fluctuations caused by isolated outliers that have disproportionately high leverage compared to the overall patterns of data. CA is essentially an aggregate measurement of roughly accurate predictions. Outliers would affect R^2 more than CA after a specific range. The outlier clouds of the models have different structure as can be observed from Figs. 4 and 6, which explains the difference in metrics accordingly.

5.3. XGBoost regression

Similarly with the process for ANN, XGBoost training required a number of specific additional steps to properly configure, outlined briefly as follows:

1. The (filtered and scaled) training dataset is loaded.
2. An XGBoost model is created. Regarding the technical details, on Python this is created through the XGBoost package which is compatible with the hyperparameter tuning algorithm.
3. A Design of Experiment (DoE) was created aiming to determine the range of each XGBoost hyperparameter.
4. A technique that creates random hyperparameter combinations was employed (Python library sklearn) to create 40,000 combinations for XGBoost parameters.
5. These combinations underwent 5-fold cross-validation and then their performance on validation data was compared.
6. The model with the optimal performance was evaluated on the test data

For XGBoost, the only fixed hyperparameter is the objective function. In the present case, regression with a squared error loss function was implemented, which is considered the most appropriate for non-linear regression problems. The remaining randomly tested XGBoost combination hyperparameter range and obtained optimized values appear on Table 4:

The final XGBoost model evaluation returned metric values as follows: RMSE = 2.82, MAE = 1.49, $CA_{5\%}$ = 86.14% and R^2 = 0.965. Based on the test data, the plot of XGBoost predicted values contrasted with the true values for main engine FOC is depicted on Fig. 7.

It is evident that the XGBoost model displays performance that is superior compared to the MPRc and ANN models, as it displays considerably lower error metrics. More importantly, the high CA score denotes that more than 86% of the predictions of the model fall within 5% of the true value of FOC data. The R^2 diagram of Fig. 8 provides another illustration of the XGBoost results obtained for the test dataset.

To summarize modelling results, an overview of the evaluation metrics scored by the three models appear on Table 5. By evaluating all four metrics, XGBoost was found to be the best-performing model compared to MPRc and ANN.

5.4. XGBoost elasticity analysis

Due to its superior performance, the XGBoost algorithm was selected for further examination through a simple form of elasticity analysis, similar to the one encountered in economics or transport research (e.g. Washington et al., 2020). Here, elasticity analysis provides utility, because even if an ML model does perform well to specific KPIs with a given dataset, the manner in which each variable is treated has to be verified as reasonable, abiding to the laws of physics and interpretable in human terms.

An intuitive example is the following: Speed through water (STW) is known to be of critical importance for predicting power. Increasing STW by 1 Knot requires a certain amount of power. As a rule of thumb, further increasing STW by an additional 1 Knot requires a larger increment of power compared to the previous step, due to increased wave-making resistance, in other words, greater energy loss, among other factors. Therefore, if an ML model produces a linear relationship between STW and power, which by the rules of naval architecture is incorrect, then they can be considered to fail the 'sanity check' that elasticity analysis provides. This would apply even more in the case where a model would model the STW and FOC relationship as negative correlation and somehow managed to yield decent predictions.

Based on the previous, a simple elasticity analysis was conducted on the XGBoost model with the optimal performance, to ensure that the model 'understands' the problem to an extent and that sane, reasonable results are going to be produced for any plausible input. Typically, an elasticity analysis is conducted by fixing all variables to a constant value and changing a single input variable at a time. Results are depicted on Fig. 9; FOC is plotted on the vertical axis on all diagrams.

From this group of diagrams, it is suggested that the XGBoost model captures the relationships between the input variables and FOC correctly and quantifies them in an appropriate manner. Finally, it should be noted that from operational experience in the VLCCs considered in this paper, a common exponent in a power regression between Speed & FOC is about 3 for Ballast state and less than 2 for Laden state, which is in line

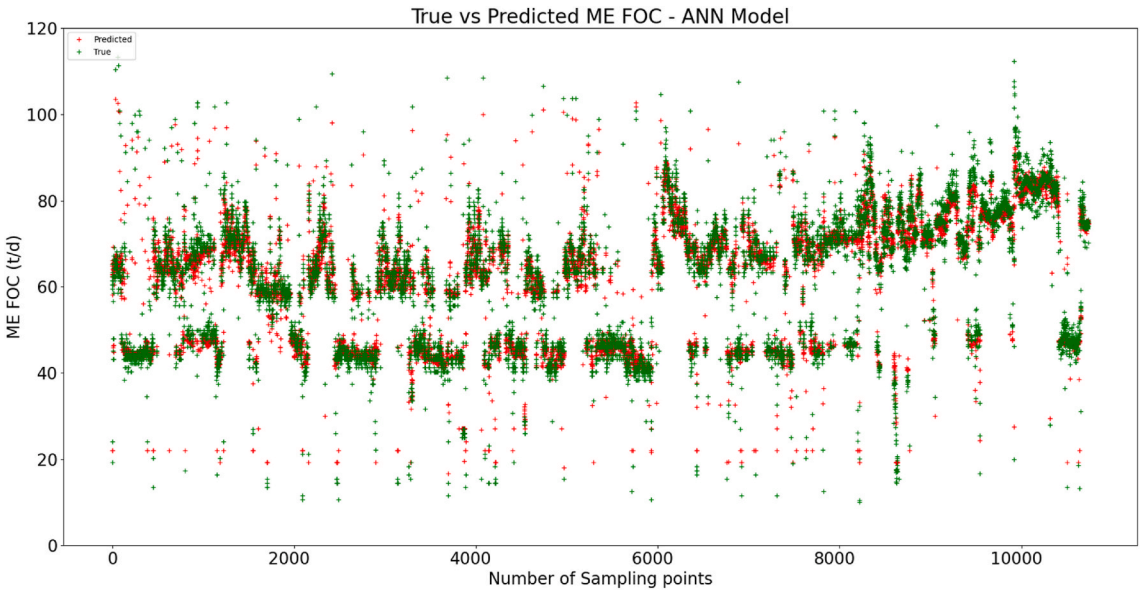

Fig. 5. ANN scatterplot of predicted FOC values vs true FOC values for the test (unseen) data.

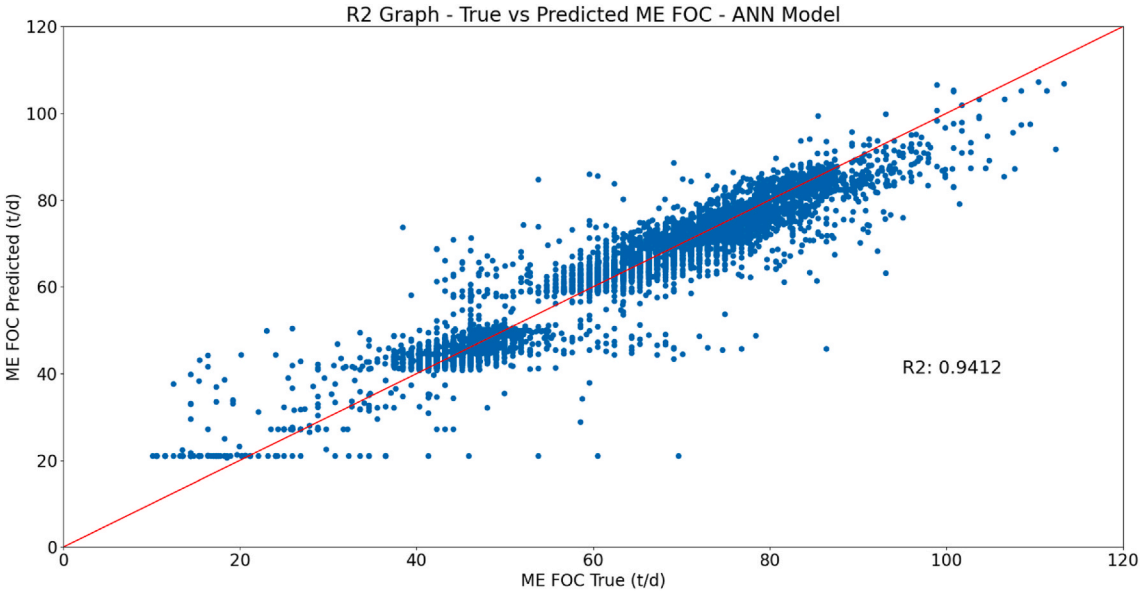

Fig. 6. ANN true values by predictions.

Table 4
XGBoost hyperparameter range and optimal values.

Hyperparameter	Lower Limit	Upper Limit	Optimized Value
Col. sample by tree	0.1000	1.0000	0.9000
Subsample	0.5000	1.0000	0.7978
Learning rate	0.0100	0.5100	0.0448
Lambda	1.0000	4.5000	1.0000
Gamma	0.5000	5.0000	5.0000
Max delta step	0.0000	10.0000	8.3160
Alpha	0.0000	1.0000	0.5000
Max depth	2	15	12
Min child weight	1	10	1
N estimators	100	1000	685

with data of Clarksons (2012) as reported by Wahl and Kristoffersen (2012). The obtained results of 3.128 and 1.798 respectively, shown in the first two diagrams, are another confirmation for the validity of the created XGBoost model.

5.5. XGBoost voyage forecasting

To showcase the feasibility of XGBoost for FOC forecasting, and to provide further validation of this method, the performance of the created model was tested for individual voyages. This evaluation was conducted for two different voyage predictions: (i) one with a duration of 24 days, when the ship was laden and (ii) one with a duration of 8 days, when the ship was in ballast. As the authors had only one dataset for each vessel to their disposal, the training/testing datasets had to be adapted for voyage forecasting. Specifically, all available data until the beginning of the particular voyage were used as training data, while the test dataset contained the voyage to be forecasted. Hyperparameters were kept as tuned from the previous process.

This evaluation also considers the voyage aggregate percentage error metric, as it provides a good estimate of the discrepancy between prediction and reality for operators to utilize in practice. This metric was calculated by dividing the absolute difference between the sum of true

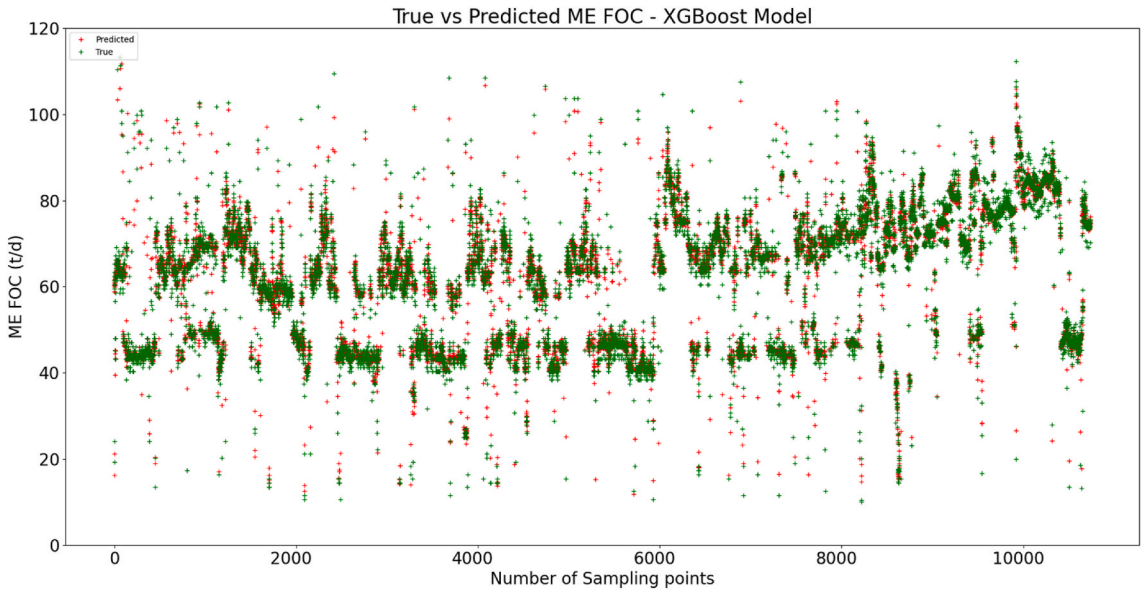

Fig. 7. XGBoost scatterplot of predicted FOC values vs true FOC values for the test (unseen) data.

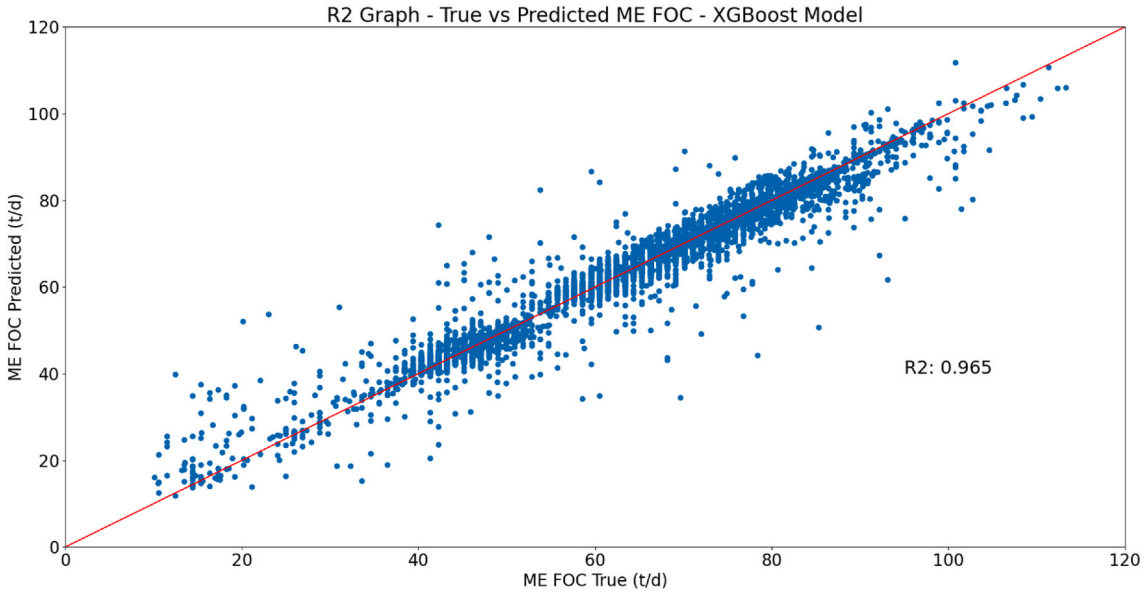

Fig. 8. XGBoost true values by predictions.

Table 5
Evaluation metric overview for the three models.

Evaluation Metric	Model		
	MPRc	ANN	XGBoost
RMSE	4.65	3.87	2.82
MAE	2.44	2.47	1.49
$CA_{5\%}$	74.05	73.40	86.14
R^2	0.903	0.941	0.965

values and the sum of predictions with the sum of true values. In equation form, following previous notation, the percentage error is:

$$Percentage\ Error = \frac{\left|\sum_{i=1}^{n}(y_i) - \sum_{i=1}^{n}(\hat{y}_i)\right|}{\sum_{i=1}^{n}(y_i)} \quad \text{Eq. (17)}$$

For the laden voyage, the XGBoost model evaluation returned metric values as follows: RMSE = 3.10, MAE = 2.20, $CA_{5\%}$ = 86.95% and R^2 = 0.832. The voyage aggregate percentage error between the aggregated true and predicted main engine FOC was 0.65%. The plot of XGBoost FOC predicted values contrasted with the true FOC values for the laden voyage is depicted on Fig. 10. Overall, XGBoost yielded a good performance for the laden voyage.

For the ballast voyage, the XGBoost model evaluation returned metric values as follows: RMSE = 3.42, MAE = 1.66, $CA_{5\%}$ = 77.04% and R^2 = 0.791. The voyage aggregate percentage error between the aggregated true and predicted main engine FOC was 0.15%. The plot of XGBoost FOC predicted values contrasted with the true FOC values for the ballast voyage is depicted on Fig. 11.

The ballast voyage performance was worse in terms of evaluation metrics compared to the laden voyage. This deterioration was caused most likely from the fact that the utilized dataset contained much fewer data points when the ship was in ballast. However, when examining the voyage aggregate percentage error, the point-by-point discrepancies

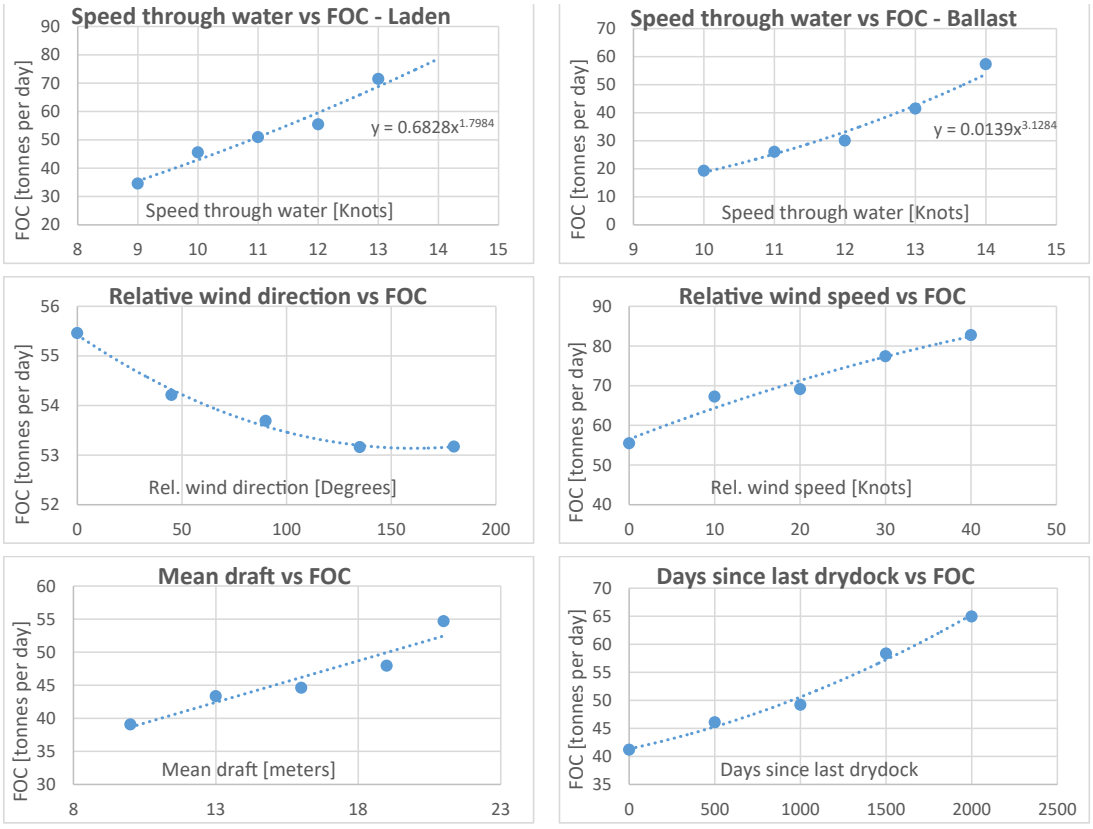

Fig. 9. XGBoost model elasticity results.

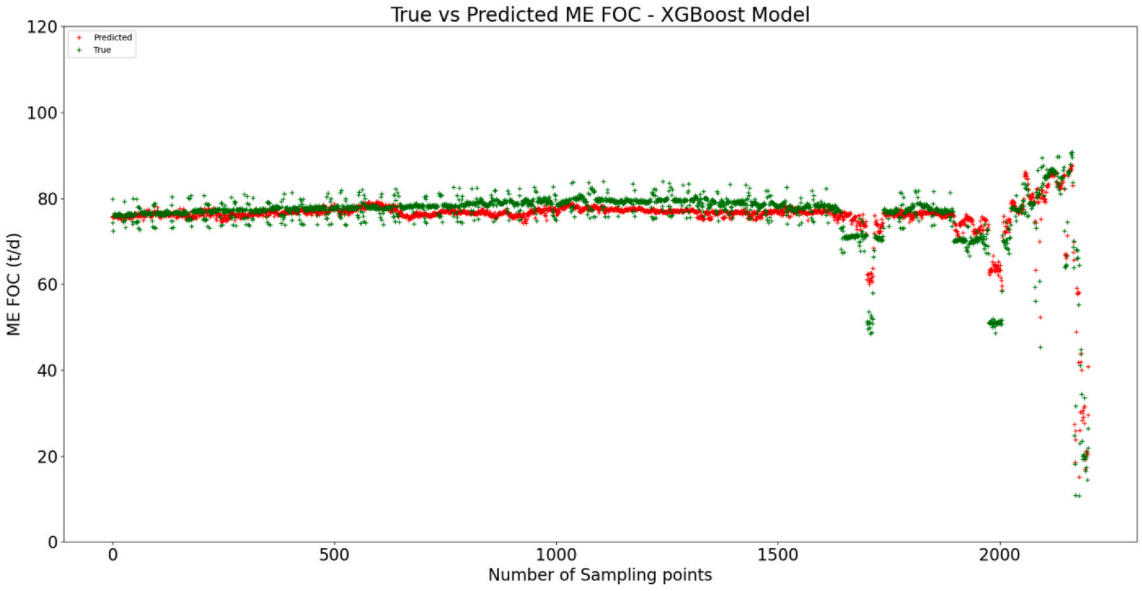

Fig. 10. XGBoost scatterplot of predicted FOC values vs true FOC values for the laden voyage.

seem to be mitigated by aggregation, which would serve the needs of the operators. Overall, it is apparent that on the aggregate level, the XGBoost model served its forecasting purpose successfully for these two voyages. Naturally, more data would be needed for wide-scale validation, but this is a promising start, especially considering that the present study uses only data that are easily obtainable in daily operations.

6. Discussion

The previous results showcase models used to forecast FOC from operationally available data from sensors onboard a VLCC vessel. As expected, the three different models performed differently, with varying degrees of success. By evaluating the RMSE and MAE error metrics, R^2 values, as well as the CA5% score, XGBoost was found to be the best-performing model compared to MPRc and ANN, yielding predictions within 5% of the true value for more than 86% of cases.

For XGBoost, an examination of Fig. 7 suggests that the times where model predictions fall off may be on instances where the true values are also non-clustered, solitary observations or misleading data due to inertia (explained in the following). Moreover, Fig. 9 indicates several

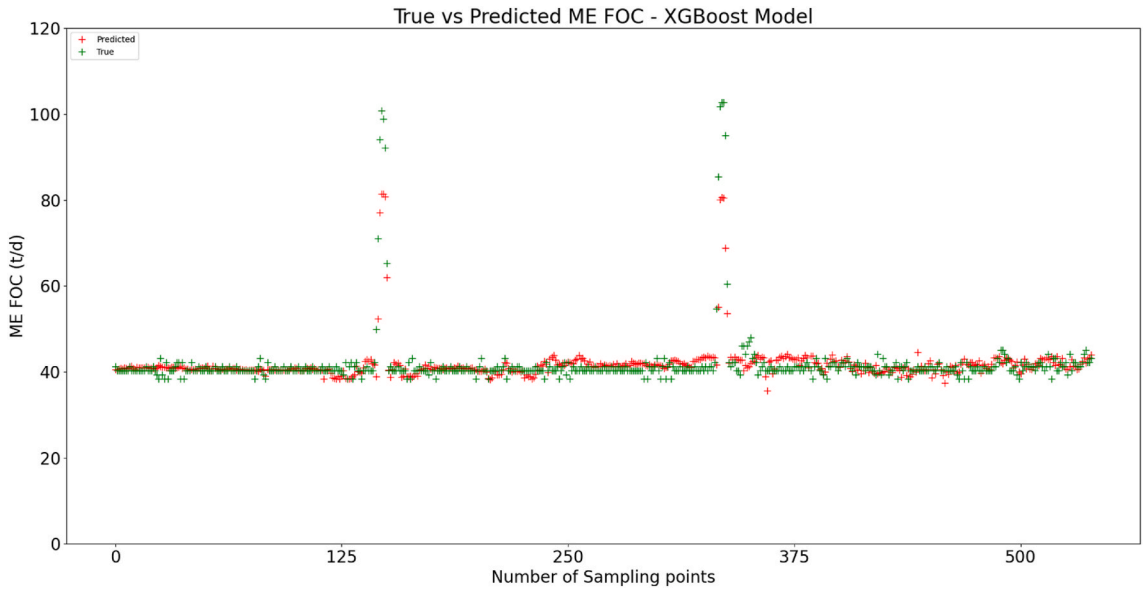

Fig. 11. XGBoost scatterplot of predicted FOC values vs true FOC values for the ballast voyage.

non-linear trends captured by the XGBoost tree ensemble, e.g. for speed through water and days since drydock. This might grant credence to the belief of industrial operators that the relationship between FOC and some predictors is apparently non-linear (Du et al., 2019). It should be also noted that due to much faster computational times for XGBoost, it was possible to examine a far larger number of random hyperparameter combinations compared to ANN models.

The relatively better performance of MPRc compared to ANNs merits elaboration. Findings where regression outperformed ML have been reported in previous literature (Wang et al., 2018), and low data frequency has been suggested as a contributing cause for that effect (Farag and Ölçer, 2020). This was not the case in the present study, however, as the utilized data had a high resolution. A possible explanation is that further efforts must be dedicated to obtain the optimal ANN structure for the present formulation of the FOC forecasting problem as, in general, the structure of an ANN directly influences its performance (e.g. Shanmuganathan, 2016). It should also be mentioned that MPRc required some correction of few instances of unreasonable FOC predictions by discarding very few outlier predictions, an action which was not needed for either of the ML methods. This necessity should be taken into account for operators wishing to use a model 'as is', without any interventions, though such filters are easy to automate. Conversely, MPRc would not be yielding as promising results if these problematic predictions were retained.

Another consideration is the data itself. It is possible that the limitations set by the scope of the present study, leading to the exclusion of main engine performance data and highly detailed weather data, caused underperformance of ANNs. The case remains that, for the present research, both MPRc and XGBoost were comparatively more easy to specify and train. These trends describing the relative and absolute performance of the three analytical methods were also mirrored in the results described in the Appendix.

During the exploratory modelling phase of this research, several additional model forms and alternatives were considered, such as: (1) logarithmic (log-linear) analysis of FOC, (2) count-data analysis of FOC with Generalized Linear Models (GLMs), (3) discretization of FOC and converting the present regression problem into a classification one, (4) higher polygon orders in MPR, (5) feature engineering of additional features based on polynomial functions from the fundamental 7 features mentioned in section 4.1 for ANN and XGBoost and others. All these alternative analyses yielded worse or illogical results and were excluded from further examination.

Based on the aforementioned, as well as on the Appendix results, it can be concluded that the XGBoost algorithm can be widely utilized with data commonly available to any operator or shipping company. The only requirement for the model to function is high-frequency automated data. With current technology, these data are easily procurable at very affordable prices. XGBoost proved a very adaptive, computationally efficient and highly accurate ML algorithm. It also provided promising aggregate forecasts for a laden voyage and a ballast voyage which had durations of several days, further showcasing its utility for such estimates. At present, there are no visible barriers to its application for FOC prediction to any pilot vessel of a fleet with the appropriate data handling and methodological steps. In addition, little effort would be required for generalization to the entire fleet depending on vessel type and size.

Further research could be undertaken based on the aforementioned results. First of all, these results showcase the usefulness of the XGBoost algorithm, which, in combination with high computational speed, show promising new research directions for XGBoost implementation in FOC prediction. There are still margins of improvement to be gained from a closer examination of the data. In particular, there are instances when a VLCC lowers engine power to decelerate, and FOC drops, however, due to inertia, vessel speed remains temporarily high. An inverse trend could be also identified during the acceleration phase. The related data points might confuse algorithms; their identification and particular treatment is a complex matter which could form further research questions. An investigation of which operationally available variables or data types favor specific algorithms is another promising research direction. Naturally, examining the validity of sensor data and the degree to which sensor faults or non-calibrated values affect models would be fruitful as well, and ways to eliminate any measurement biases could lead to even more accurate predictions.

One of the findings of the study is the suboptimal ANN performance. It is possible that ANN as a method may not be fitting with the scope of the present study and its data limitations. However, past literature on the topic has demonstrated the utility of ANN for FOC predictions. Consequently, further investigation of ANN suitability may be required. A related pending question is the investigation and identification of the optimal ANN model structure; the number of layers (dense & dropout) and the number of neurons at each layer is of paramount importance when determining the optimal structure for a regression problem such as the present one. The exploration of additional ANN types such as Long Short-Term Memory (LSTM) Recurrent Neural Networks also merits

investigation.

Simple wave parameters, to the extend where they could be easily obtainable and useable for prediction during daily operations, could constitute a worthwhile addition to the exploited data. The presented process ought to be extended to more vessel types, to determine the transferability of the utilized methods or to find more appropriate ones for each examined vessel type. The integration of the present models into weather routing systems could be a meaningful way in order to minimize FOC during a voyage.

7. Conclusions

The present research aimed to provide accurate models that utilize exclusively data from sensors and simple weather data readily available in operational practice. Commonly available sensor data from a Very Large Crude Oil Carrier (VLCC) were used, and Multivariate Polynomial Regression (MPR), Artificial Neural Networks (ANNs) and eXtreme Gradient Boosting (XGBoost) regression models were developed and evaluated based on their predictive accuracy for VLCC FOC. Results indicated that XGBoost had the best overall performance, yielding predictions within 5% of the true value for more than 86% of total cases. It was followed by corrected MPR (MPRc) as second (accurate predictions for 74% of the total cases) and ANN as a close third (accurate predictions for 73% of the total cases).

The contribution of the present study is both methodological, by testing a popular ML algorithm (XGBoost) that has not been implemented previously for FOC prediction in VLCCs to the knowledge of the authors, and practical, through the utilization of easily obtainable data. Throughout the present research, emphasis was placed on exploiting data exclusively obtained from sensors readily available in operational practice. This standard ensured that the developed models can be used by operators for daily FOC forecasting or for the FOC prediction of entire voyages. Utilizing approaches such as the one outlined in the present research can increase energy efficiency in an attainable manner and ensure cost-competitive operations.

Funding details

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

Christos Papandreou: Conceptualization, Methodology, Software, Data curation, Formal analysis, Visualization, Investigation, Writing – original draft, Writing – review & editing. **Apostolos Ziakopoulos:** Conceptualization, Methodology, Software, Data curation, Formal analysis, Visualization, Investigation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix: FOC forecasting results for a second VLCC

The MPR, ANN and XGBoost methods were trained, validated and tested on a second VLCC as well, to investigate the transferability capabilities of the adopted methodological approach. The developed models were fitted without altering any of the hyperparameters. The procedure followed is identical with the one described in Section 5, without hyperparameter optimization for the ML models (the MPRc process is identical). Results are presented in the following. Overall, the same findings outlined in the main paper are observed: XGBoost has the best overall performance, followed by MPRc as second and ANN as a close third.

Multivariate Polynomial Regression – Second VLCC.

For the second VLCC, the MPRc model returned evaluation metric values as follows: RMSE = 4.09, MAE = 2.44, $CA_{5\%} = 74.68\%$ and $R^2 = 0.908$. Based on the test data, the plot of MPR predicted values contrasted with the true values for main engine FOC is depicted on Figure A2.

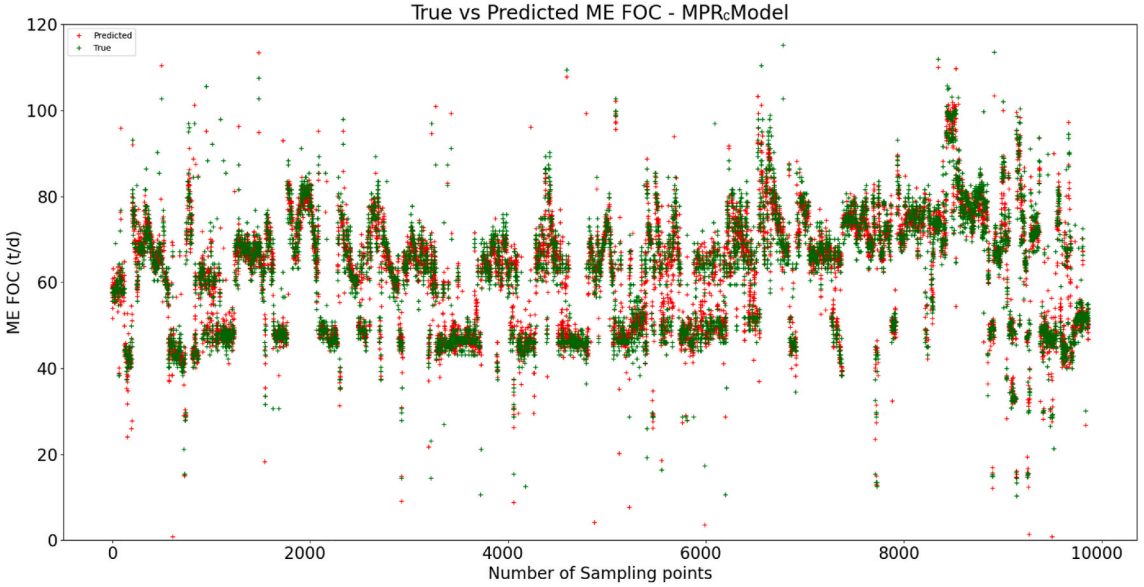

Fig. A1. MPR scatterplot of predicted FOC values vs true FOC values for the second VLCC test data

Artificial Neural Network – Second VLCC.

For the second VLCC, the ANN model returned evaluation metric values as follows: RMSE = 4.28, MAE = 2.89, $CA_{5\%}$ = 68.14% and R^2 = 0.900. Based on the test data, the plot of ANN predicted values contrasted with the true values for main engine FOC is depicted on Figure A2.

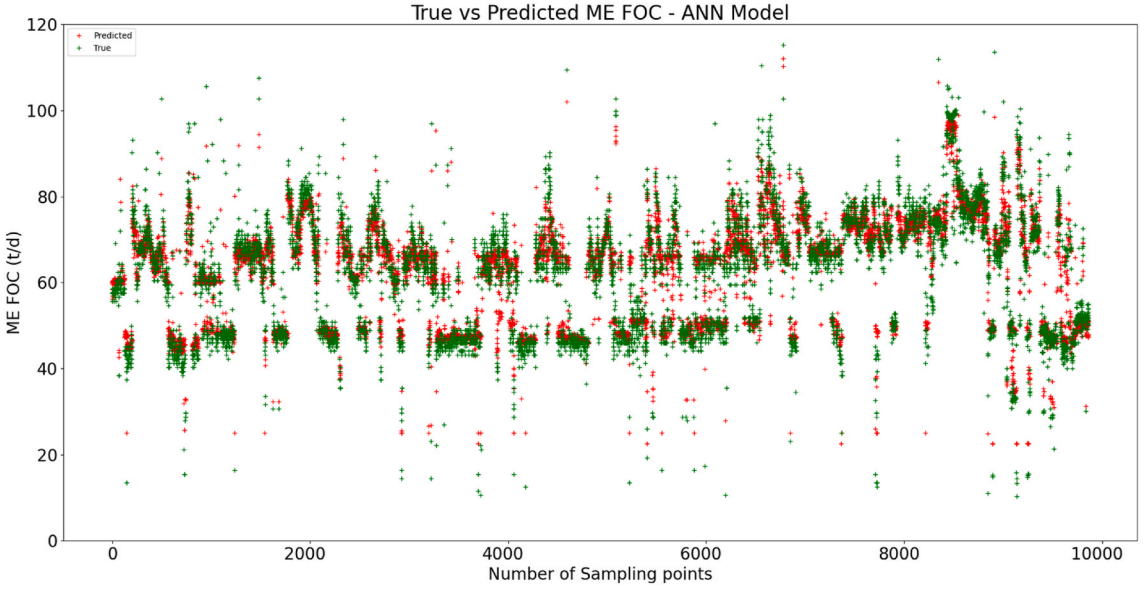

Fig. A2. ANN scatterplot of predicted FOC values vs true FOC values for the second VLCC test data

XGBoost Regression – Second VLCC.

For the second VLCC, the XGBoost model returned evaluation metric values as follows: RMSE = 2.49, MAE = 1.39, $CA_{5\%}$ = 87.21% and R^2 = 0.966. Based on the test data, the plot of XGBoost predicted values contrasted with the true values for main engine FOC is depicted on Figure A2.

In the opinion of the authors, these results are a good indication that the discussed methods can be generalized to different vessels. This is because the absolute and relative performance of the three methods is retained when transitioning to 'unseen' training data from a different vessel. The models appear to be accurate enough without optimization, and with correct data manipulation they can avoid overfitting. The very good overall performance of XGBoost perseveres, showing a very promising method for FOC forecasting problems.

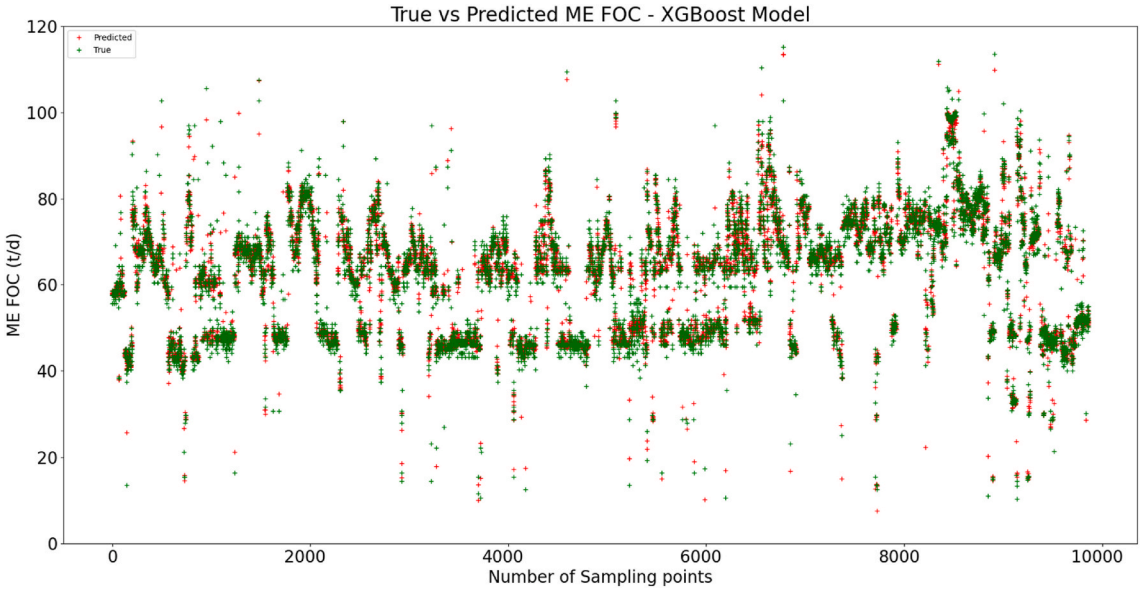

Fig. A3. XGBoost scatterplot of predicted FOC values vs true FOC values for the second VLCC test data

An overview of the evaluation metrics scored by the three models appear on Table A1.

Table A1
Evaluation metric overview for the three models for the second VLCC

Evaluation Metric	Model		
	MPRc	ANN	XGBoost
RMSE	4.09	4.28	2.49
MAE	2.44	2.89	1.39
$CA_{5\%}$	74.68	68.14	87.21
R^2	0.908	0.900	0.966

Once again, XGBoost appears to outperform the other two considered methods across all examined metrics. As per the aforementioned, the examination of the second VLCC is an investigation of transferability and validation of the models. Hence, the tuned hyperparameters of the ML algorithms were retained from the first VLCC. Training and testing was conducted normally afterwards on data from the second VLCC. The slight improvement of XGBoost performance compared to the first VLCC is therefore due to randomness in the data.

References

Agatonovic-Kustrin, S., Beresford, R., 2000. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. J. Pharmaceut. Biomed. Anal. 22 (5), 717–727.

Ahlgren, F., Mondejar, M.E., Thern, M., 2019. Predicting dynamic fuel oil consumption on ships with automated machine learning. Energy Procedia 158, 6126–6131.

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford university press.

Chen, T., Guestrin, C., 2016, August. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining. ACM, pp. 785–794.

Chen, K., Jiang, J., Zheng, F., Chen, K., 2018. A novel data-driven approach for residential electricity consumption prediction based on ensemble learning. Energy 150, 49–60.

Clarksons, 2012. Clarksons database. Available: https://www.clarksons.net/portal.

Dawson, C.W., Wilby, R., 1998. An artificial neural network approach to rainfall-runoff modelling. Hydrol. Sci. J. 43 (1), 47–66.

Dong, W., Huang, Y., Lehane, B., Ma, G., 2020. XGBoost algorithm-based prediction of concrete electrical resistivity for structural health monitoring. Autom. ConStruct. 114, 103155.

Du, Y., Meng, Q., Wang, S., Kuang, H., 2019. Two-phase optimal solutions for ship speed and trim optimization over a voyage using voyage report data. Transp. Res. Part B Methodol. 122, 88–114.

Fan, H., Tu, H., Enshaei, H., Xu, X., Wei, Y., 2021. Comparison of the economic performances of three sulphur oxides emissions abatement solutions for a very large Crude carrier (VLCC). J. Mar. Sci. Eng. 9 (2), 221.

Farag, Y.B., Ölçer, A.I., 2020. The development of a ship performance model in varying operating conditions based on ANN and regression techniques. Ocean Eng. 198, 106972.

Gkerekos, C., Lazakis, I., Theotokatos, G., 2019. Machine learning models for predicting ship main engine Fuel Oil Consumption: a comparative study. Ocean Eng. 188, 106282.

Hara, K., Saito, D., Shouno, H., 2015. July). Analysis of function of rectified linear unit used in deep learning. In: 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

International Chamber of Shipping (ICS), 2014. Shipping, world trade and the reductions of CO2 emissions. Available online: https://safety4sea.com/ics-shipping-world-trade-reduction-co2-emissions/. (Accessed 18 January 2021).

International Maritime Organization (IMO), 2018. IMO action to reduce greenhouse gas emissions from international shipping: implementing the initial IMO strategy on reduction of GHG emissions from ships. Available online: https://www.imo.org/en/MediaCentre/HotTopics/Pages/Reducing-greenhouse-gas-emissions-from-ships.aspx. (Accessed 27 January 2021).

Ioffe, S., Szegedy, C., 2015. June). Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. PMLR, pp. 448–456.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning, 112. springer, New York, p. 18.

Jeon, M., Noh, Y., Shin, Y., Lim, O.K., Lee, I., Cho, D., 2018. Prediction of ship fuel consumption by using an artificial neural network. J. Mech. Sci. Technol. 32 (12), 5785–5796.

Jordan, M.I., Mitchell, T.M., 2015. Machine learning: trends, perspectives, and prospects. Science 349 (6245), 255–260.

Joung, T.H., Kang, S.G., Lee, J.K., Ahn, J., 2020. The IMO initial strategy for reducing Greenhouse Gas (GHG) emissions, and its follow-up actions towards 2050. J. Int. Maritime Safety, Env. Affairs, and Shipping 4 (1), 1–7.

Kleinbaum, D.G., Kupper, L.L., Nizam, A., Rosenberg, E.S., 2013. Applied Regression Analysis and Other Multivariable Methods. Nelson Education.

Le, L.T., Lee, G., Park, K.S., Kim, H., 2020. Neural Network-Based Fuel Consumption Estimation for Container Ships in Korea. Maritime Policy & Management, pp. 1–18.

Levantis, M., Paereli, S., Betz, W., 2020. Speed-Power Models – A Bayesian Approach. In: Proceedings of the 5th Hull Performance & Insight Conference - HullPIC'20, Hamburg, pp. 26–28. October 2020.

Ma, F., Yan, X., 2019. Research on the energy consumption estimation method of pure electric vehicle based on XGBoost. In: 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE). IEEE, pp. 1021–1026.

Nielsen, D., 2016. Tree Boosting with Xgboost-Why Does Xgboost Win "every" Machine Learning Competition? (Master's thesis, NTNU).

Petersen, J.P., Jacobsen, D.J., Winther, O., 2012a. Statistical modelling for ship propulsion efficiency. J. Mar. Sci. Technol. 17 (1), 30–39.

Petersen, J.P., Winther, O., Jacobsen, D.J., 2012b. A machine-learning approach to predict main energy consumption under realistic operational conditions. Ship Technol. Res. 59 (1), 64–72.

Petersen, J.P., Fredriksen, J., Jacobsen, D.J., 2020. Bridging the gap between noon data and automated data. In: Proceedings of the 5th Hull Performance & Insight Conference - HullPIC'20, Hamburg, pp. 26–28. October 2020.

Ronen, D., 1982. The effect of oil price on the optimal speed of ships. J. Oper. Res. Soc. 33 (11), 1035–1040.

Ronen, D., 2011. The effect of oil price on containership speed and fleet size. J. Oper. Res. Soc. 62 (1), 211–216.

Samarasinghe, S., 2006. Neural Networks for Applied Sciences and Engineering: from Fundamentals to Complex Pattern Recognition. Crc Press.

Schmidhuber, J., 2015. Deep learning in neural networks: an overview. Neural Network. 61, 85–117.

Shacham, M., Brauner, N., 1997. Minimizing the effects of collinearity in polynomial regression. Ind. Eng. Chem. Res. 36 (10), 4405–4412.

Shanmuganathan, S., 2016. Artificial neural network modelling: an introduction. In: Artificial Neural Network Modelling. Springer, Cham, pp. 1–14.

Shu, C., Burn, D.H., 2004. Artificial neural network ensembles and their application in pooled flood frequency analysis. Water Resour. Res. 40 (9).

Sinha, P., 2013. Multivariate polynomial regression in data mining: methodology, problems and solutions. Int. J. Sci. Eng. Res. 4 (12), 962–965.

Smith, T.W.P., Jalkanen, J.P., Anderson, B.A., Corbett, J.J., Faber, J., Hanayama, S., et al., 2014. Third IMO GHG Study 2014. International Maritime Organization (IMO), London.

Traut, M., Larkin, A., Anderson, K., McGlade, C., Sharmina, M., Smith, T., 2018. CO2 abatement goals for international shipping. Clim. Pol. 18 (8), 1066–1075.

Uyanik, T., Karatuğ, Ç., Arslanoğlu, Y., 2020. Machine learning approach to ship fuel consumption: a case of container vessel. Transport. Res. Transport Environ. 84, 102389.

Wahl, M.E.B., Kristoffersen, E., 2012. Speed Optimization for Very Large Crude Carriers (VLCCs): Potential Savings and Effects of Slow Steaming (Master's Thesis).

Walther, L., Rizvanolli, A., Wendebourg, M., Jahn, C., 2016. Modeling and optimization algorithms in ship weather routing. Int. J. e-Navigation and Maritime Eco. 4, 31–45.

Wang, Q., Zhang, R., Lv, S., Wang, Y., 2021. Open-pit mine truck fuel consumption pattern and application based on multi-dimensional features and XGBoost. Sustain. Energy Technol. Assessments 43, 100977.

Wang, S., Ji, B., Zhao, J., Liu, W., Xu, T., 2018. Predicting ship fuel consumption based on LASSO regression. Transport. Res. Transport Environ. 65, 817–824, 1.
Washington, S., Karlaftis, M.G., Mannering, F., Anastasopoulos, P., 2020. Statistical and Econometric Methods for Transportation Data Analysis. CRC Press.
XGBoost developer team, 2019. XGBoost Documentation. Official website. Available online: https://xgboost.readthedocs.io/en/latest/index.html. (Accessed 3 February 2021).
Zhang, D., Qian, L., Mao, B., Huang, C., Huang, B., Si, Y., 2018. A data-driven design for fault detection of wind turbines using random forests and XGboost. IEEE Access 6, 21020–21031.
Zhao, Z., Xu, S., Kang, B.H., Kabir, M.M.J., Liu, Y., Wasinger, R., 2015. Investigation and improvement of multi-layer perceptron neural networks for credit scoring. Expert Syst. Appl. 42 (7), 3508–3516.