The 8th International Symposium on Ship Operations, Management & Economics (SOME)
7-8 March 2023, Athens GR

www.sname.org

Time Series Analysis for Digital Twins in Green Shipping

Lazaros Avgeridis[1] (V)**, Konstantinos Lentzos**[1] (V)**, Dimitrios Skoutas**[1] (V)**, Ioannis Z. Emiris**[1] (V)
1. ATHENA Research Center

A promising way for the waterborne industry towards decarbonization and emissions reduction is through digitalization and in particular via Digital Twins (DTs) technology. In this context, the DT provides insights for optimal decision-making, predicting potential future events, or even detecting irregularities in the behavior of the ship to reduce its carbon emissions and energy consumption. To achieve this, we propose an architecture for automated data capture, processing, and analysis. The analysis component of this infrastructure leverages machine learning (ML) algorithms for time series data, such as anomaly detection and forecasting. Importantly, to understand how these algorithms make a certain prediction we also provide a detailed look at current approaches used to interpret these models. Finally, we demonstrate a practical use case, where time series analysis can prove especially useful when applied to real-world vessel data.

KEY WORDS: digital twins; time series; anomaly detection; forecasting; explainability; vessel; machine learning

INTRODUCTION

Internet-of-Things (IoT) devices generate increasingly massive amounts of time series data. IoT technology is becoming ubiquitous in many industries including shipping, retail, healthcare, manufacturing, transportation, agriculture, utilities, automobile, etc. All of these domains involve some kind of temporal measurements, usually via sensors, censuses, transaction records, etc., so the capture of a sequence of observations indexed by time stamps is crucial for providing insights into the past evolution of some measurable quantity.
More specifically, a time series is a sequence of data points (observations) ordered in time. Typically, time is treated as a discrete variable in this context. The pervasiveness of time series has also generated an increasing demand for performing various analysis tasks on time series data, including visualization, discovery of recurrent patterns, clustering, anomaly detection, segmentation, forecasting and data simulation among others.

Anomaly detection (also called outlier detection) for example represents one of the most prominent fields of time series analysis. Anomaly detection is central to comprehending what is the normal behavior of data captured by an IoT system and detecting instances that deviate from the norm. The data are considered normal if they obey the usual operating characteristics for that specific system. It is reasonable to assume that usual operation of a system can change over time and this can happen for a variety of reasons.

Forecasting is another important area of time series analysis, as it is of paramount importance in many domains: in meteorology, to guide informed decision-making for air and maritime navigation; in digital transactions, to detect abnormal or fraudulent situations; in stock investing, to predict the future trajectories of stocks of interest; in medicine, to predict the spread of a disease, estimate mortality rates or assess time-dependent risk. In essence, forecasting is based on the following principle: knowing the past behaviors of a given system, it is possible to make predictions on its nearby or long-term behaviors.

A time series represents the temporal evolution of a dynamic system that one seeks to describe, explain and predict. One such system could be a moving object, e.g., a private vehicle, an airplane or a vessel. In fact, there have been many recent attempts to aid the maritime industry to enter a new phase by gradually adapting to modern digitization principles. In particular, this can be achieved via Digital Twins (DTs) technology (Leandro 2021; Danielsen-Haces 2018; Grange 2018; Bole 2017). The general conception of a digital twin has three core elements: the physical entity (vessel), the digital or virtual entity and the bidirectional communication that is established between these two entities. The whole purpose of this technology is to maintain a virtual system that basically mirrors the physical one. In practice, this is done by continuously updating the state of the virtual entity using measurements coming from the physical entity. The collected measurements are then processed and analyzed to support decision making, which leads to actions applied to the waterborne system to optimize its operation.

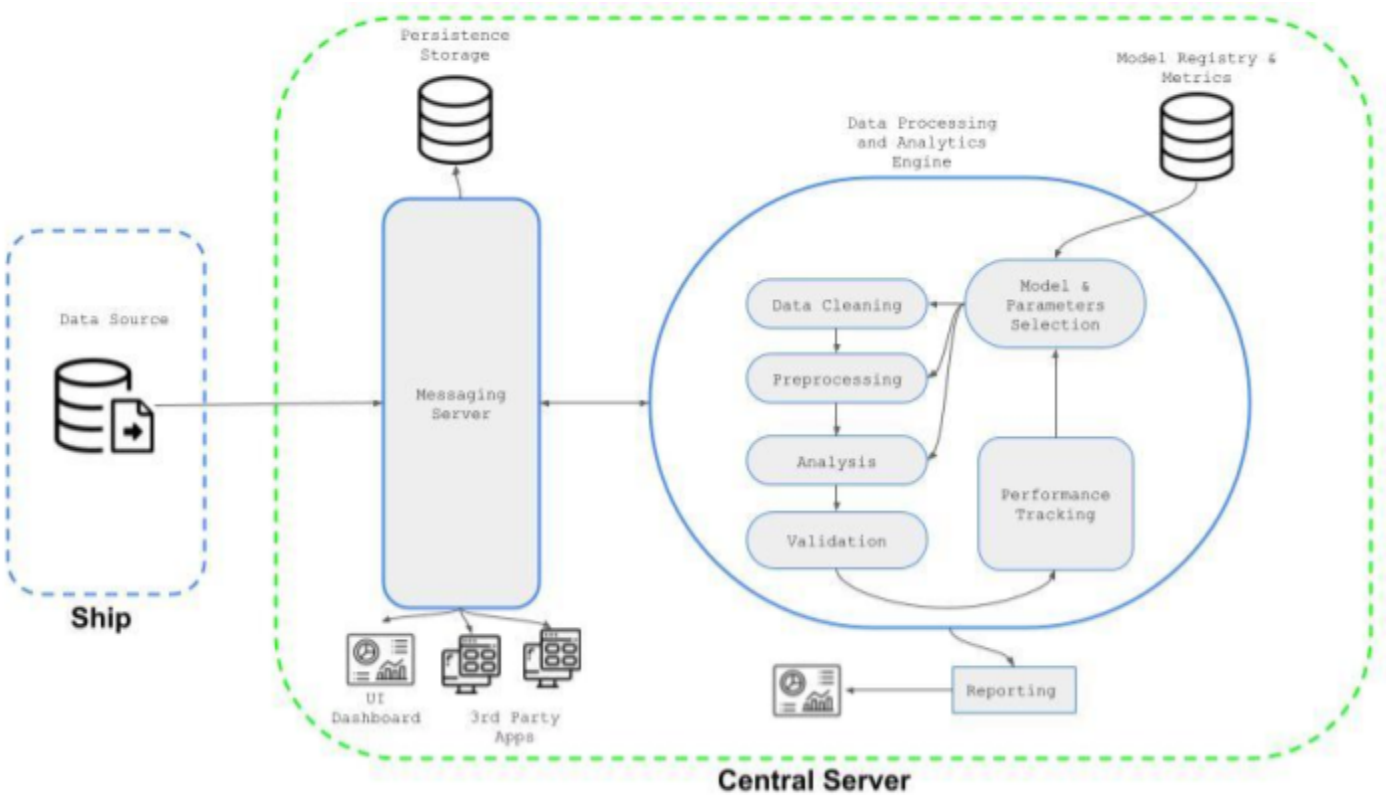

Fig.1 : The proposed architecture for capturing, processing and analyzing time series data.

To be more precise, vessels that adhere to this framework are equipped with onboard sensor technologies that can measure critical quantities regarding the status of the vessel in question, such as speed, coordinates, fuel consumption, engine temperature, etc. Importantly, these sensor updates are transmitted in real-time.

The main goal of many of these efforts is to allow ship designers, builders and operators to collaborate on reducing costs, improving efficiency, boosting safety and most significantly, reducing ship emissions to the natural world, thus aiming to achieve decarbonization in the long term. This can be achieved by leveraging an appropriate technological infrastructure, which will collect the captured sensor observations and analyze them to provide valuable feedback to the human observers. This will enable stakeholders to make environmental-friendly decisions based on the provided information. For example, potential alerts and insights from such a system could be directed towards the captain to adjust the speed and direction of the ship in a way that would optimize fuel consumption.

This paper is organized as follows: Section 2 presents our proposed architecture for collecting, storing, processing and analyzing time series data. Section 3 is devoted to two extensively studied areas of time series analysis, anomaly detection and forecasting, along with insights into the methodology used to explain the results of the most widely used machine learning models. Section 4 demonstrates the results produced by 2 time series analysis algorithms applied on real world vessel data. Finally, Section 5 concludes our work by identifying future research directions in the field of time series analysis.

ARCHITECTURE OVERVIEW AND MAIN COMPONENTS

In this Section we describe a high-level architecture for storing, processing, and analyzing time series data in a scalable and resilient manner. Such an infrastructure can generate real-time actionable insights extracted from the captured data, in order to guide stakeholders toward avoiding potential risks and making better decisions. In principle, this system architecture can be connected to any data source generating real-time events and immediately start producing meaningful results.

The architecture of the proposed pipeline is illustrated in Fig. 1. It consists of 3 main components: the data source, the messaging system, and the data processing and analytics engine. It should be noted that the architecture presented here is still under

development and that it drew inspiration from a preliminary version considered in the context of the European project “Digital Twins for Green Shipping” (DT4GS). As such, a lot of technical details regarding for instance the transmission, collection and storage of the data are yet to be configured.

Data Source

The pipeline will be powered by vessel data that arrive in the form of sensor observations. These observations are captured by onboard sensor technologies available on the ship and may measure quantities such as velocity, fuel consumption, acceleration, engine temperature etc. Since these observations are ordered based on time and arrive at fixed time intervals *t*, a straightforward way to model them is through time series data. Since multiple sensor updates are received at each timestamp, the produced time series are characterized as multivariate.

This time series data can either arrive in real-time, as they are produced, and be sent directly to the Messaging Server (middle), or be initially stored in an external database and then sent to the Messaging Server. In both cases, data collection and transmission will be handled by the appropriate infrastructure installed on the ship. It is also expected that there will be incorrect, corrupted, or incomplete data points within the captured observations; hence, a data cleaning process is typically necessary prior to any analysis.

Messaging System

Event streaming platforms play a central role in ingesting, storing, and processing real-time data in a scalable and resilient manner. In our case, an event (also record or message) is a notification that "something happened" in the vessel, i.e, an update in the state of the ship. An event stream is a continuous unbounded series of such events. The start of the stream may have occurred before we started processing the stream. The end of the stream is at some unknown point in the future, i.e when the vessel reaches a port. Each event in a stream carries a timestamp that denotes the point when the event occurred. Events are ordered based on this timestamp. In general, the definition of time series data aligns with that of event streams.

Typically, these systems function according to the popular producer-consumer paradigm. Simply put, in the core of this paradigm there are 2 (computer) processes, one that sends events (or messages), also called the producer, and one that receives events (or messages), also called the consumer.

There are several reasons to utilize such a system in our architecture. Firstly, these systems completely decouple producers and consumers, meaning that producers are not aware of who is going to consume the data that they are producing. Conversely, consumers are not aware of who produced the data they are consuming. That enables adding or removing components from/to our architecture without significantly affecting existing ones. Secondly, event streaming platforms provide scalable and durable storage for real-time data, as they are distributed systems, purposely built to ingest millions of data records per second. The data are then stored in fault-tolerant storage to keep them safe from data losses.

In our pipeline, the Messaging System, which will be located on a central server, communicates with both the Data Source (left) and the Data Processing and Analytics Engine (right). More specifically, incoming events originating from the data source are initially written to the Messaging System’s storage. Then, these events are read from the same storage location and used as input to the data analytics component. The results of each data analysis run are written back to the Messaging System, which is also responsible for forwarding them to third-party applications or UI dashboards. In addition, the Messaging System is also connected to a persistent storage, which serves the purpose of backing up our time series data for future use.

One of the most widespread event streaming platforms today is Apache Kafka (kafka.apache.org). Kafka is an open-source software program that lets users store, read, and analyze streaming data. As a distributed system, Kafka can run on several servers to leverage different servers’ processing power and storage capacity. Certain key features of Kafka include scalability, fault tolerance, durability, reliability, zero downtime, performance, replication and extensibility. Another well-known alternative is RabbitMQ (rabbitmq.com), an open-source message broker that is lightweight and supports asynchronous messaging services. RabbitMQ runs on different operating systems and cloud infrastructure and can also be deployed in a distributed environment for high availability. Moreover, it has a flexible plug-in approach and a variety of tools to support continuous integration and operational metrics.

Data Analysis

The most crucial component in the pipeline is the Data Processing and Analytics Engine and is located on the same central server as the Messaging System. This part involves a multi-step procedure, which includes applying data pre-processing, analytics and evaluation operations in succession. In essence, the desired outcome here is to employ techniques that will enable us to detect irregularities in the sensor observations or predict the future behavior of the vessel. The underlying data analysis is executed by a library of available models for time series analysis. It is also essential to track the performance of these models so that we can gradually enhance the model selection process by examining previous analysis runs.

Once the messaging system sends the data in raw form, the data processing and analysis engine is triggered. Initially, the data pass through two separate pre-processing stages. The first stage

is Data Cleaning. Essentially, its purpose is to perform detection and removal of erroneous values, as well as missing values imputation. Subsequently, the cleaned data are in the appropriate form to be sent to a persistence storage through the Messaging System for later use, may it be for future comparisons or simply for creating a large database of historical observations. The second stage involves Preprocessing, and is more model-specific, meaning that it applies extra data transformations required by the selected model to run, such as normalization, dimensionality reduction etc.

A question that emerges here is how exactly to choose the most fitting model(s) for each analysis run. The answer is given by consulting the ongoing "Performance Tracking" component, which stores statistics regarding the performance of previous models' execution. According to this information, the best performing models are chosen and configured in the "Model and Parameters Selection" subtask and loaded by sending a request to the externally maintained "Model Registry & Metrics" logging system. If no past information is available related to model performance, then for the initial analysis all models may be used in order for the performance tracker to start collecting statistics. Moreover, different models often have different constraints on the data format and characteristics they expect to receive as input, which may affect their applicability or impact their performance. Consequently, the choice of model also influences the operations taking place in the "Data Cleaning" and "Preprocessing" subtasks.

Up to this point, both pre-processing steps have been successfully applied on the data and the most appropriate model(s) have been selected for proceeding with the actual analysis. In this work, we are focusing specifically on two very widely studied and used time series analysis tasks, namely anomaly detection and forecasting., which we further discuss in more detail in Section 3. Depending on the specific type of problem we wish to tackle, we can select the appropriate subset of models and conduct the "Analysis" step. As input to the selected models, we use the cleaned data we prepared in the previous steps.

Once the analysis is completed, we collect the output of our models and perform a final "Validation" step. At this stage, we are interested in measuring the quality of the produced outputs, i.e. the predictions in the case of Machine Learning methods, by introducing a set of metrics (e.g., accuracy) or simply reporting experiment-specific information such as whether a model managed to process the entire dataset or not, execution time, memory requirements etc. These statistics are also sent to the "Performance Tracking" service, which will guide the model selection process in future analysis experiments, as well as a remote "Reporting" server, which compiles all validation results and displays them in an easy to comprehend graphical environment, intended for those who manage the data analytics engine. Finally, the predictions of our models are sent back to the Messaging System, as mentioned previously.

Evidently, the "Analysis" subtask relies heavily on a library of models that perform the data analysis. To properly track the performance of the models and their version history, we consider third-party software that could assist us in managing their whole lifecycle. After all, current machine learning methods often require thorough experimentation. This translates to training a particular model instance, usually for prolonged periods of time, by feeding it with a large set of data observations, then evaluating the model and measuring its performance by using standardized metrics, and finally trying to interpret the results and draw valid conclusions.

There are several tools available for tracking and managing machine learning experiments, which can be generally divided into two categories : (a) experiment tracking only and (b) management of the entire lifecycle. For example, Weights & Biases (wandb.ai) is a popular machine learning platform belonging to the former category and designed mainly for experiment tracking, even though also capable of (limited) dataset versioning and model management. Its key features include a user-friendly and interactive dashboard, which allows users to visualize and compare results of the model training process. On the other hand, there exist solutions such as the open-source platform MLflow (mlflow.org) that belong to the latter category and hence offer both experiment tracking and reproducibility along with model deployment and central storage. MLflow in particular supports logging of various experiment metadata, packaging ML code in a format to reproduce runs on any platform, deploying ML models and storing them in a central repository.

TIME SERIES ANALYSIS

In this Section we are going to examine 2 broad areas of time series analysis : (i) anomaly or outlier detection and (ii) forecasting. In addition, given that a lot of methods employed in these two areas make use of machine learning models, we deem necessary to understand both how and why a model makes a certain prediction. Thus, a summary of approaches for explaining machine learning models is presented last.

Anomaly Detection

An anomaly can be thought of as an unexpected change in the state of a system, which is outside of its local or global norm. There are three general types of time series anomalies distinguished in the literature, in particular point, contextual, and collective anomalies (Pang 2021). We briefly outline them below:

Point anomalies refer to data points that deviate remarkably from the rest of the data. Point anomalies usually make up for a small percentage of the total data observations. The affected

system after experiencing these short anomaly intervals returns to its previous normal state. Point anomalies can also represent statistical noise or noise produced by faulty sensing equipment.

Contextual anomalies refer to data points within the expected range of the distribution, but which deviate from the expected data distribution, given a specific context. Contextual anomalies if taken in isolation may be within the range of expected values, but when taken in the context of the surrounding observations constitute anomalies.

Collective anomalies refer to sequences of points that do not repeat a typical previously observed pattern. Individual observations within a collective anomaly may or may not be anomalous, it is only when they appear as a group that they arouse suspicion.

The first two categories, namely, point and contextual anomalies, are referred to as point-based anomalies. whereas, collective anomalies are referred to as subsequence anomalies.

Anomaly detection problems are supposed, by definition, to handle and process unpredictable rare events characterized by many unknown factors such as their structures, distributions and irregularities. Anomalies are typically rare data instances, contrasting to normal instances that often account for an overwhelming proportion of the data. In (Pang 2021), authors discuss the main challenges in anomaly detection problems. In particular, they discuss about low anomaly detection recall rates (high false positives), the need for handling high dimensional data, noise-resilient models and detection of complex anomalies such as subsequences anomalies and detecting anomalies in a more contextual setting by considering two or more data sources.

In (Schmidl 2022), the authors categorize the 71 algorithms studied using various criteria. One viable categorization is by considering the method family into which each algorithm can be categorized. These method families characterize the algorithms by their general approach of determining the abnormality of specific points or subsequences within the time series. The resulting six method families are the following: (1) forecasting methods, (2) reconstruction methods, (3) encoding methods, (4) distance methods, (5) distribution methods, and (6) isolation tree methods.

Anomaly detection algorithms can also be classified by their learning type, i.e unsupervised, supervised and semi-supervised. However, both (Pang 2021) and (Schmidl 2022) point out that supervised learning approaches are quite unpopular in practice and thus the majority of techniques developed in anomaly detection research is based mainly on semi-supervised and purely unsupervised settings and methods. This is due to the fact that labeled training data with both normal and anomaly classes are quite restricted and rare in real-world scenarios. It is difficult and costly to collect labeled anomalies or annotate them, and even if the instances are labeled the method will suffer from serious class imbalance issues. Consequently, for the rest of this section we focus on recent unsupervised techniques for detecting anomalies in time series data. Such methods can be used to populate the models registry mentioned in Section 2.3 with a pool of available anomaly detection algorithms for time series data. Moreover, we outline several recent benchmarks, which can be used to evaluate and compare the performance of these algorithms, thus bootstrapping the metrics registry and providing guidance for model selection and parameter tuning.

Unsupervised Methods do not require a set of labeled samples to fit the parameters of a particular machine learning model. An unsupervised subsequence anomaly detection approach for data streams is proposed in (Boniol 2021) called SAND (Streaming Subsequence Anomaly Detection). SAND supports real-time analytics, as the algorithm is online and does not require access to the entire dataset. Anomalies are detected based on their distance to a data structure that represents normal behavior. Moreover, SAND does not require domain knowledge, and thus can detect domain-agnostic anomalies. Finally, the algorithm is able to adapt to distribution drifts, i.e., to changes in the data generation process.

Another state-of-the-art unsupervised anomaly detection method is proposed in (Campos 2022), which uses autoencoder ensembles. The ensemble employs multiple basic anomaly detection models built on convolutional sequence-to-sequence autoencoders. The idea behind using autoencoders in anomaly detection is summarized below. An autoencoder consists of an encoding phase that compresses a time series T into a compact representation and a decoding phase that reconstructs an output time series T' from this representation. The representation is only able to capture patterns that reflect normal behavior in the original time series and not anomalies. The difference between observations in T and in T' is called the reconstruction error. Intuitively, the higher reconstruction error which means less likely to be from the input distribution, the higher the anomaly score. A threshold can be set to discriminate anomaly from normality. The autoencoder ensemble the authors use employs multiple autoencoders to avoid overfitting.

Anomaly detection based on transformers has been introduced as for example in (Tuli 2022). Transformer models proved to be more efficient than *Long short-term memory* (*LSTM*) networks, mostly due to parallel computing in their architecture. In (Tuli 2022), authors proposed to combine transformers with neural generative models for better reconstruction models in anomaly detection. In particular, they propose an adversarial training procedure to amplify reconstruction errors. A form of adversarial training, a paradigm utilized when training Generative Adversarial Networks (GANs), is processed by two transformer encoders and two transformer decoders to gain

stability. More methods about using transformers in anomaly detection settings can be found in (Wen 2022).

Benchmarking Toolkits were created to satisfy the demand for commonly accepted ways to compare and analyze the performance of anomaly detection approaches**,** since research in this area is still very active.

The process of picking the most suitable detection algorithm for a given problem is rarely straightforward. This is due to the fact that many algorithms are remarkably heterogeneous in their detection approaches and the current number of proposed algorithms is large. What makes the necessary evaluations even harder is the limited collection of publicly available datasets, a careful consideration of runtime and quality aspects, and the need to evaluate a multitude of specific properties including anomalies that are uni-/multi-variate, point/sequence, unique/repeating, shape/magnitude etc. anomalous.

One solution to the above problem is TimeEval (Wenig 2022). TimeEval is an extensible, scalable and automatic benchmarking toolkit for time series anomaly detection algorithms. It includes an extensive data generator and supports both interactive and batch evaluation scenarios. TimeEval is written in Python and consists of four major components: a dataset generator called GutenTAG, the core evaluation engine called Eval, a Python API, and a respective GUI built on top of the API. TimeEval offers configuration options and user-defined code/plugins to incorporate evaluation settings. It can execute custom algorithms and work with external datasets.

Another alternative proposed in (Khelifati 2021) is VADETIS (Validator for Anomaly Detection in Time Series), a graphical evaluator for anomaly detection techniques. VADETIS allows to visualize time series data, perform anomaly detection in a supervised setting, generate anomalies by specifying their type, scale and frequency. Finally after choosing a performance metric, VADETIS evaluates a set of detection techniques, compares their performance, and recommends the optimal technique. Vadetis is implemented as a web application with the Python Django framework.

In (Paparrizos 2022) an extensive benchmark, TSB-UAD, is proposed to evaluate univariate time-series anomaly detection methods. The authors identified, collected and processed 13766 time series with labeled anomalies spanning across multiple domains, applications, anomaly types, and anomalies of increasing difficulty. Algorithms ranked high in TSB-UAD are expected to demonstrate good performance when applied in a new context. More specifically, TSB-UAD consists of three dataset categories: public, artificial and synthetic.

Theseus (Boniol 2022) is a web application developed using Python and the Dash framework. It is based on TSB-UAD and helps users to navigate and compare several anomaly detection methods as well as visualizing and evaluating their methods' outcomes.

Finally, based on the taxonomy of anomalies, Lai (2021) presents a general synthetic benchmark with 35 corresponding synthetic datasets and 4 multivariate real world datasets from different domains to evaluate anomaly detection algorithms.

Forecasting

Traditional time series forecasting methods based on probability and statistics have achieved great results in real-world applications They can be generally divided into linear and nonlinear models, with linear models mainly including: autoregressive models (AR), moving average (MA) models, autoregressive moving average models (ARMA) and autoregressive integrated moving average (ARIMA) models. Classic nonlinear models mainly consist of Threshold Autoregressive (TAR) models, Constant Conditional Correlation (CCC) models and conditional heteroscedasticity models. In addition, there are some important classical prediction methods based on exponential smoothing, such as: Simple Exponential Smoothing (SES), Holt's linear trend method, Holt-Winters' multiplicative method, Holt-Winters' additive method and Holt-Winters' damped method. For a more detailed presentation of conventional forecasting approaches, we refer the interested reader to the following surveys: (De Gooijer 2005), (Liu 2021), (Dama 2021).

Nevertheless, in the following we focus on more recent methods that employ Machine Learning, and in particular deep learning, techniques for time series forecasting. As in the case of anomaly detection discussed above, such methods could be used to populate the model registry, offering a suite of time series forecasting algorithms.

Machine Learning Architectures for Time Series Forecasting are particularly suitable for finding the appropriate complex nonlinear mathematical function to turn an input into an output. Hence, machine learning models provide a means to learn temporal dynamics in a purely data-driven manner.

Artificial Neural Networks (ANNs) can be employed for nonlinear processes that have an unknown functional relationship and as a result are difficult to fit. The main concept with ANNs is that inputs are passed through one or more hidden layers each of which consists of hidden units, or nodes, before they reach the output variable.

In time series forecasting, as in other domains, the emergence of competing neural network architectures has relegated simple ANNs to the background.

Convolutional Neural Networks (CNNs) are a specific kind of deep neural networks, proposed for and mostly dedicated to image analysis and aimed at preserving spatial relationships in the data, with very few connections between the layers. Having

connections from all nodes of one layer to all nodes in the subsequent layer, as in regular ANNs, proved extremely inefficient and thus CNNs arose from the observation that a careful pruning of the connections, based on domain knowledge, boosts performance.

Although initially designed to handle two-dimensional image data, CNNs can be used to model both univariate and multivariate time series. A one-dimensional CNN will just operate over a sequence instead of the usual matrix input. More importantly, CNNs naturally lend themselves to process multivariate time series. A multivariate time series will be fed to the CNN as various vectors corresponding to the initial channels.

Recurrent Neural Networks (RNNs) were designed to handle sequential information (Lipton 2015). Typically, RNNs are capable of making predictions over many time steps, in time series. An RNN achieves the same task at each step (with varying inputs): the sequence (x_1, x_2 , · · · , x_t, x_{t+1}, · · ·) is input to the RNN, element by element (one step at a time).

For instance, the usage of an RNN-based architecture for multivariate time series forecasting is demonstrated in (Munkhdalai 2019). In particular, the model called AIS-RNN, consists of two main components: an encoding network and a forecasting network.

Long short-term memory networks (LSTMs) are the most widely-used subclass of RNNs, as they perform better than RNNS in capturing long dependencies. LSTMs are intrinsically RNNs in which changes were introduced in the computation of hidden states and outputs, using the inputs. They were developed to address some limitations that occurred during the training process of regular RNNs.

Transformers, like RNNs, were designed to handle sequential data and tackle problems in natural language processing (for instance, translation). Transformers were repurposed to address forecasting in time series. The encoder and decoder form the two parts of the transformer's architecture.

The encoder mainly consists of an input layer, an input encoding, a positional encoding mechanism, and a stack of identical encoder layers. In the seminal work of (Vaswani 2017), the decoder is composed of an input layer, an input encoding, a positional encoding mechanism, a stack of identical decoder layers and an output layer.

Optimal Forecasting Model Selection Autoforecast (Abdallah 2022) is a *meta-learning* approach that allows for the quick inference of the best time-series forecasting model for an unseen dataset. Given a new time series dataset, AutoForecast selects the best performing forecasting algorithm and its associated hyperparameters without having to first train or evaluate all the models on the new time series data to select the best one. The meta-learner is trained on the models' performances on historical datasets and the time-series meta-features of these datasets.

Another *meta-learning* approach is proposed in (Saadallah 2021) in a deep reinforcement learning context and in particular an actor-critic algorithm. The policy network (actor) is trained to gain experience on how to select the best set of weights given a previous combination of models in the ensemble. The goal is to learn a policy in the continuous action space, in which the actions are determined to be the set of weights of the ensemble.

Explainability in Time Series Analysis

In many safety-critical domains there may be some major risks if (machine learning) models are directly used as black-box tools that can magically solve problems. An effective approach to mitigate this is to have methods that provide explanations about their outcomes. These interpretability methods can be classified according to various criteria. For instance, a general categorization could be based on the universality of the method. Model-agnostic methods can be used to explain any model (Ribeiro 2016), whereas model-specific methods are specifically built to add interpretability for a certain type of model. Another categorization is based on whether these methods are incorporated in the model structure or applied independently of it (Rojat 2021), (Theissler 2022). The two categories of approaches that arise in this case are post-hoc and ante-hoc. Post-hoc are model-agnostic methods that provide explanations of a model by extracting relationships between feature values and their predictions (Moradi 2021). On the other hand, ante-hoc methods offer explainability as part of the model's settings and components providing automatically any generated explanations at the time of prediction (Sarkar 2022). Ante-hoc methods are considered as model-specific, given that they can be used only to explain themselves. We present more details for each categorization in the following paragraphs.

Post-hoc Methods vs. Ante-hoc Methods Post-hoc explainability approaches are separated from the model they explain and can provide insight into what a model has learned after training without changing or taking advantage of its underlying structure. Post-hoc methods are commonly selected to explain convolutional neural networks.

Ante-hoc explainability methods are studied whenever there is the possibility to generate explanations from the model's inner components. Such models can be considered directly interpretable due to their simple structure or transparency by design. Examples include decision trees, recurrent neural networks, transformers, etc., and in particular their attention mechanism.

Local vs. Global Explanations Another categorization between explanations is whether they are local or global. Local explanations methods are those which make predictions sample

by sample, as for example convolutional neural networks. Although recurrent neural networks make their predictions sample by sample, they have a memory state that retains the knowledge built. Their latent representations are designed to handle one or several samples depending on whether the internal states are reset after each prediction. The explanations are therefore local if the internal states represent one instance, or global if the internal states represent several instances. Global explanation approaches provide an explanation that describes the overall logic of the entire model for any input instance. Models able to produce global explanations do not usually process the data sample by sample. Some research also extends the methods generating local explanations to produce global explanations (Oviedo 2019).

Other methods Other explainability methods return representative examples from the training dataset as explanations of a typical behavior. Shapelets (Guidotti 2021) are able to produce explanations for time series (Medico 2021), (Vandewiele 2021). Shapelets are subsequences that are maximally representative of a class. A shapelet is called discriminant if it is present in most series of one class and absent from series of the other classes. In a classification setting, the goal is to find the most discriminant shapelets given some labeled time series data (Ye 2009).

In (Cho 2021) a clustering approach is studied based on subsequences. Explanations based on subsequences identify sub-parts of a time series responsible for the classification outcomes. Each cluster is composed of a list of temporal sequences that activate the same nodes. Highly activated temporal regions are extracted that contribute to activating internal nodes and are identified in order to obtain their general shapes.

Other model-agnostic, post-hoc approaches use feature selection methods (Pang 2017), as for example for anomaly detection we may use a subset of features that makes an anomaly even more abnormal (Siddiqui 2019). In that way those features constitute a valid explanation for that anomaly.

Explanation Systems TSExplain (Chen 2021) helps users to understand the underlying evolving explanations for aggregated time series. It works similarly to ExplainData feature of Tableau or Explain Increase of PowerBI. Users need to create a query with a group-by clause and use an aggregation function as for example the sum. The system provides explanations on the value of the aggregation function between two time steps of data's granularity to highlight which features had the greatest impact on the result. Internally, TSExplain models the explanation problem as a segmentation problem over the time dimension. Intuitively, explanations tend to be continuous over time and they would expect to span some time before disappearing or drifting to some other explanations. Thus, the problem can be framed as a segmentation problem over time *t*, as we would like explanations within each segment to be coherent, while explanations across neighboring segments to be distinct.

OUR USE CASE

We studied marine data from [AIS], which are free and publicly available. The data are real-world data collected from sensors of variable datetime granularity. The dataset contains the following attributes:

- MMSI - Maritime Mobile Service Identity value (integer as text)
- BaseDateTime - Full UTC date and time (YYYY-MM-DDTHH:MM:SS)
- LAT - Latitude (decimal degrees as double)
- LON - Longitude (decimal degrees as double)
- SOG - Speed Over Ground (knots as float)
- COG - Course Over Ground (degrees as float)
- Heading - True heading angle (degrees as float)
- VesselName - Name as shown on the station radio license (text)
- IMO - International Maritime Organization Vessel number (text)
- CallSign - Call sign as assigned by FCC (text)
- VesselType - Vessel type as defined in NAIS specifications (int)
- Status - Navigation status as defined by the COLREGS (text)
- Length - Length of vessel (see NAIS specifications) (meters as float)
- Width - Width of vesses (see NAIS specifications) (meters as float)
- Draft - Draft depth of vessel (see NAIS specification and codes) (meters as float)
- Cargo - Cargo type (SEE NAIS specification and codes) (text)
- TransceiverClass - Class of AIS transceiver (text) (unavailable in 2017 dataset)

It is clear that MMSI, VesselName, IMO, CallSign, Length, Width, Cargo and TransceiverClass can be considered as vessel's characteristics. In particular, the MMSI uniquely identifies the vessel.

In this work, we focus on SOG, which is the speed of the vessel relative to the surface of the earth. We fix the granularity of the observed values to every 5 minutes using the mean and filling missing values if any with forward fill. We present the resulting values of one vessel from May 28, 2021, to June 4, 2021, in Fig. 2.

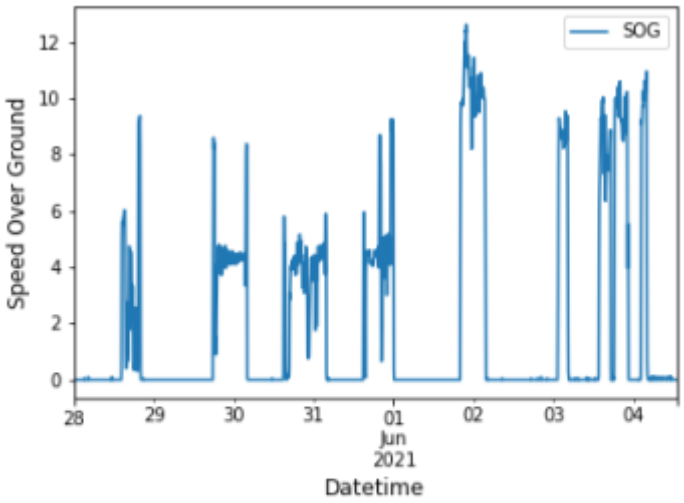

Fig.2 : Visualization of the Speed Over Ground (SOG) data attribute.

From the figure, we can easily extract the vessel's trips, which can be further verified with the corresponding spatial information (LAT, LON).

Due to the fact that labeled anomalies do not exist in this dataset for the SOG attribute, we had to select an appropriate unsupervised learning approach to detect outliers. In general, many types of anomalies can emerge in datasets, but by looking at Fig. 2 in particular, we can clearly discern some extremum data points. These observations may or may not be true outliers in practice. Still, it would be valuable to be able to detect them and attempt to explain why and how they appear, i.e we would like to know if unusually high-speed values are caused by extreme weather conditions or engine failure.

Consequently, we utilize the power and robustness of decision trees to identify anomalies in time series data. More specifically, we use the Isolation Forest algorithm to predict whether a certain point is an outlier or not in an unsupervised manner. Isolation Forest (IF) (Liu 2008) is a tree ensemble method that belongs to the algorithm family "Isolation tree methods" mentioned in Section 3.1 (Anomaly Detection). It explicitly identifies anomalies instead of profiling normal data points. In other words, detects anomalies purely based on the fact that anomalies are data points that are few and different. The detection of them is accomplished without employing any distance or density measure. As a hyperparameter, we set the contamination to 0.01. This number expresses what proportion of outliers are present in the data. Fig. 3 shows the detected anomalies (highlighted in red).

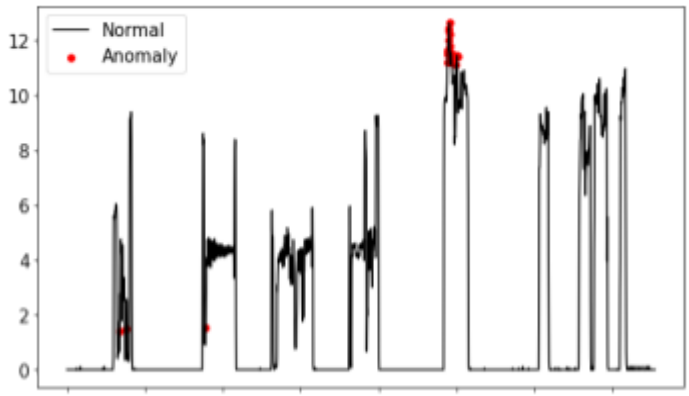

Fig.3 : Anomaly detection using the Isolation Forest (IF) algorithm.

Examining the results produced by IF we can see that most anomalies detected correspond to exceptionally high values of SOG, approximately over 11 knots. Additionally, the algorithm identifies some outliers with values below 2 knots that do not seem to deviate significantly from the rest of the data depicted here. We hypothesize that this behavior is related to our choice of the contamination hyperparameter and might indicate that we should either reduce or increase its value and execute the experiment again.

One of the most extensively used and straightforward approaches for labeling outliers is Inter Quartile Range (IQR). This is a very robust technique based on statistics and it is essentially a measure of how stretched or squeezed is the distribution of the data. To be more precise, the IQR is defined as the difference between the 75th and 25th percentiles of the data. To calculate the IQR, we divide our data points into quartiles or four rank-ordered even parts via linear interpolation. These quartiles are denoted by Q_1 (also called the lower quartile), Q_2 (the median), and Q_3 (also called the upper quartile). The lower quartile corresponds to the 25th percentile and the upper quartile corresponds to the 75th percentile, so IQR $= Q_3 - Q_1$. Any data point that falls outside of either 1.5 times the IQR below the first quartile or 1.5 times the IQR above the third quartile is considered an outlier.

Fig. 4 demonstrates anomalies detected by the IQR method. We observe that IQR classifies as anomalies, data points with SOG values approximately over 11 knots, similar to the IF method. However, it expands to data points with values closer to 10 knots. Furthermore, some outliers towards the end of the x-axis are identified, which are equal to or slightly over SOG value of 10 knots.

Uncovering anomalies in the data can also be done by calculating the standard deviation of the dataset of interest and then using it as the cutoff point for locating irregular behavior.

In this sense, the Standard Deviation method is similar to IQR. In other words, we define an upper and lower limit as $U = \mu + 3*stdev$ and $L = \mu - 3*stdev$ respectively, where μ is the dataset mean and *stdev* is the dataset standard deviation. After calculating these bounds, we classify all points lying outside of this region as outliers.

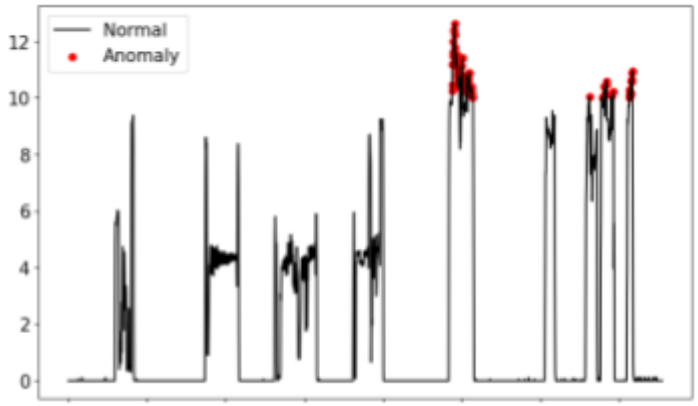

Fig.4 : Anomaly detection using Inter Quartile Range (IQR).

The results of the Standard Deviation method are shown in Fig. 5. This approach finds the smallest number of anomalies compared to IF and IQR. Nonetheless, the detected outliers have SOG values between 11 and 12 knots, which was also the case for the other two techniques.

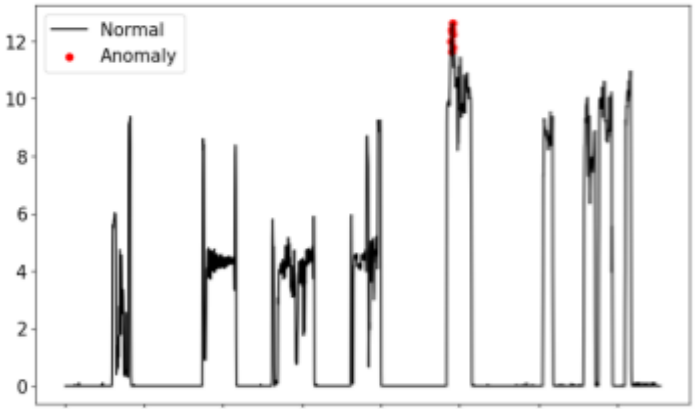

Fig.5 : Anomaly detection using Standard Deviation.

Moreover, in an attempt to eliminate random short-term variations and extrema and to highlight other components, such as trend, season, or cycle, present in our time-series, we computed the simple moving average of the data, also known as the rolling mean. A sliding window of one hour was used and the smoothed counterpart of Fig.2 is presented in Fig. 6.

We then calculated the difference between the original SOG time-series Y_{raw} and the smoothed SOG time-series Y_{smo}. On the resulting time-series Y_{dif} we applied Standard Deviation as our outlier detection method (Fig. 7). Lastly, we projected the anomalies detected on Y_{dif} to Y_{raw} (Fig.8). Outliers in Y_{dif} are located roughly below $y = -3$ and above $y = +3$, while the projected outliers in Y_{raw} are scattered along the entire y-axis. For instance, we can see that every point with SOG value equal to 0 following a region of steep decrease is classified as an anomaly.

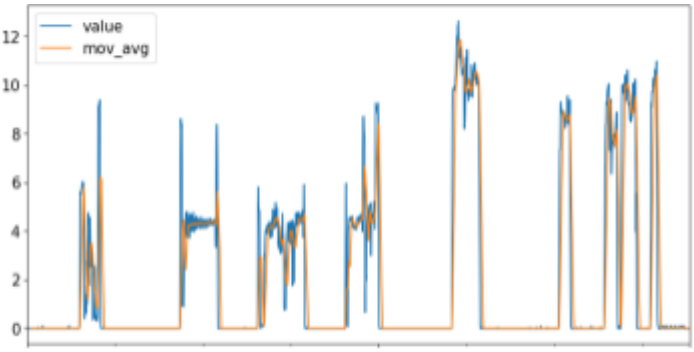

Fig.6 : Visualization of both the original and smoothed SOG data attribute.

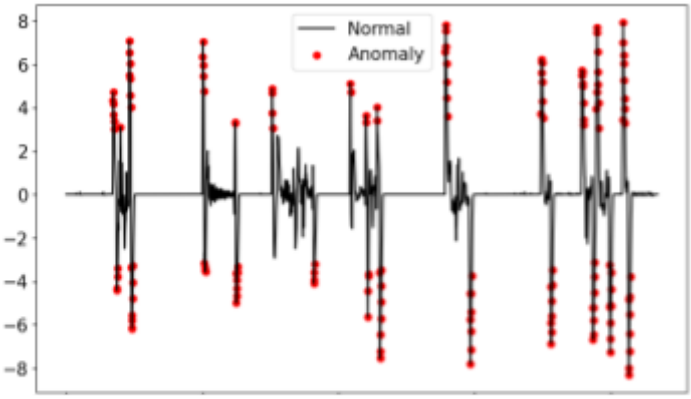

Fig.7 : Anomalies detected by applying Standard Deviation on the time-series of differences.

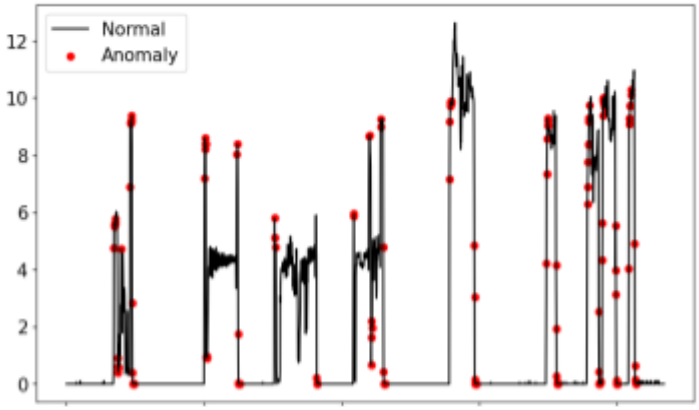

Fig.8 : Anomalies uncovered for the raw SOG time-series by "transferring" anomalies produced by Standard Deviation for the time-series of differences.

We note that applying the IQR and standard deviation methods to the smoothed time series generate the same point anomalies (Fig.9).

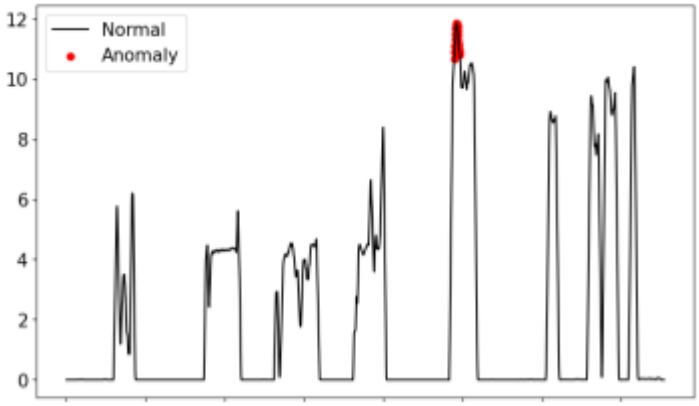

Fig.9 : Application of IQR, Standard Deviation and Isolation Forest on the smoothed SOG data attribute all produce the same anomalies.

As zero values may affect the anomaly detection algorithms, we apply below the standard deviation method using three ranges *2*stdev*, *1.75*stdev* and *1.5*stdev* see figures 10, 11 and 12 respectively. For the range defined by 3*stdev no anomalies are detected.

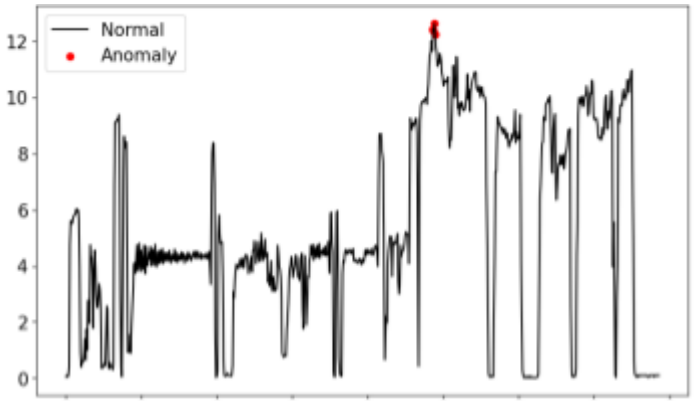

Fig.10 : Standard Deviation with parameter 2, on the SOG time-series, where values equal to zero have been filtered out.

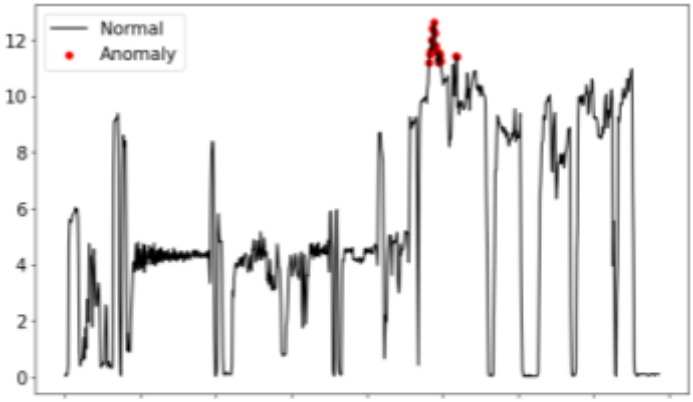

Fig.11 : Standard Deviation with parameter 1.75, on the SOG time-series, where values equal to zero have been filtered out.

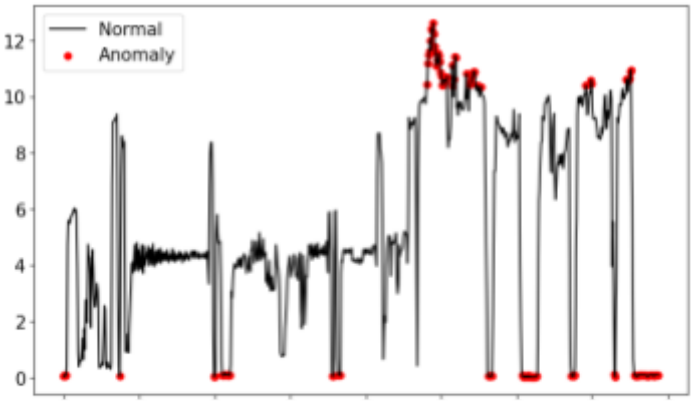

Fig.12 : Standard Deviation with parameter 1.5, on the SOG time-series, where values equal to zero have been filtered out.

To elaborate more on this idea, we isolate the subseries that contain no zero values. Those can be considered as the individual trips of the ship. We apply the standard deviation method using the *2*stdev* range to detect possible anomalies. We illustrate our results in the figures 13, 14 and 15.

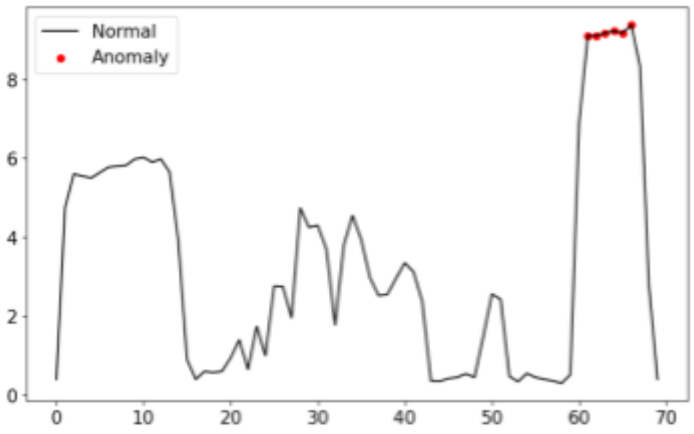

Fig.13 : Anomalies detected during the trip A that was made between 2021-05-28 14:15:00 and 2021-05-28 20:00:00.

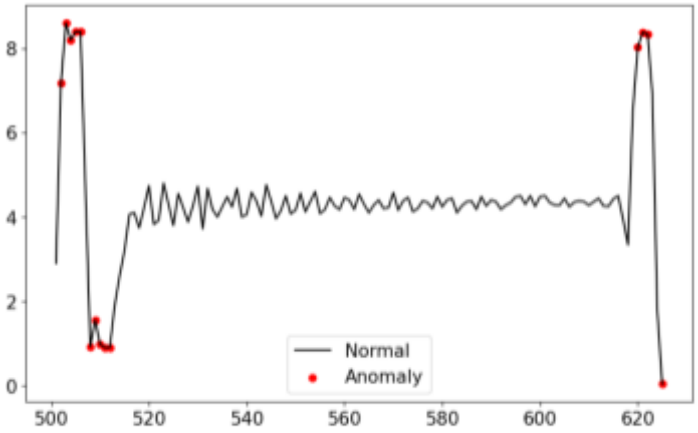

Fig.14 : Anomalies detected during the trip B that was made between 2021-05-29 17:45:00 and 2021-05-30 04:05:00.

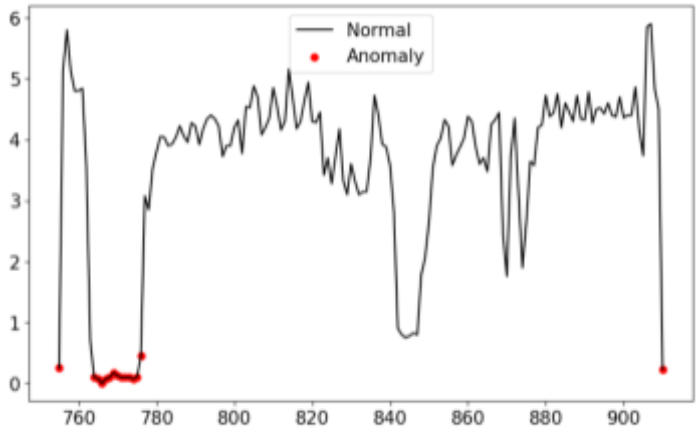

Fig.15 : Anomalies detected during the trip C that was made between 2021-05-30 14:55:00 and 2021-05-31 03:50:00.

We conclude that trip A contains some anomalies mostly at the end of its time as it travels with relatively high speed at those time points. Trip B shows high acceleration followed by sharp deceleration at the beginning and at the end of the trip. Trip C can be considered without any anomalies.

Lastly, anomaly detection using forecasting is examined last. The main idea is that past points can generate a forecast of the next point with the addition of some random variable, which is usually white noise. Forecasted points in the future will generate new points and so on. We predict the new point from past datums and find their difference in magnitude. After choosing a threshold, we are able to identify anomalies. To test this technique, we used a popular module in time-series called *Prophet* (Prophet). We note that Prophet takes into account some additional features for example the predicted time series variable, the upper and lower limit of the target time series variable, and the trend, to facilitate anomalies' detection. Utilizing the same data, we identify the anomalies depicted in Fig. 10. Since this technique is based on forecasting, it usually leads to results of low quality. This is not the case though.

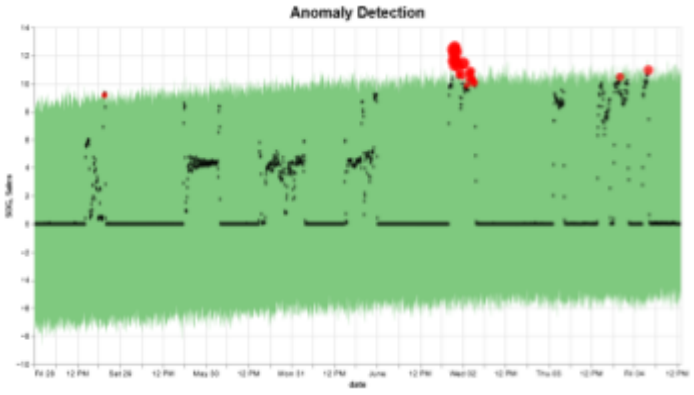

Fig.16 : Anomaly detection using a forecasting approach.

CONCLUSIONS

With the growth in time series data availability in recent times, mainly due to the increase in usage of IoT devices, one can get immense value out of performing various analysis tasks on them. The shipping industry in particular can greatly benefit by adopting Digital Twins technology. Through this novel approach, industry stakeholders will be able to discover insights for enhanced decision-making in shipping operations and importantly for reducing the carbon footprint of the main ship classes. Our proposed infrastructure offers an effective solution for automated data capture, processing, and analysis, which can be leveraged in the context of Digital Twins applied in maritime use cases.

After reviewing the current state of time series analysis literature, especially anomaly detection and forecasting, our future endeavors will be targeted towards developing a model library of our own. More specifically, our intention is to efficiently implement some of the most widely used anomaly detection and forecasting algorithms, perform experiments on vessel data/publicly available time series datasets, evaluate their performance, and combine all of the above in an open-source time series library.

ACKNOWLEDGEMENTS

This work was supported by the EU Horizon Europe project DT4GS (Grant No. 101056799).

REFERENCES

[AIS] https://marinecadastre.gov/ais/

Abdallah, Mustafa, Ryan Rossi, Kanak Mahadik, Sungchul Kim, Handong Zhao, and Saurabh Bagchi. 2022. "Autoforecast: Automatic time-series forecasting model selection." *In Proceedings of the 31st ACM International Conference on Information and Knowledge Management, CIKM* '22, page 5–14.

Bole, Marcus, Gabriel Powell, and Eric Rousseau. 2017. "Taking Control of the Digital Twin."

Boniol, Paul, John Paparrizos, Yuhao Kang, Themis Palpanas, Ruey S. Tsay, Aaron J. Elmore, and Michael J. Franklin. 2022. "Theseus: Navigating the labyrinth of time-series anomaly detection." *Proc. VLDB Endow.*, 15(12):3702–3705.

Boniol, Paul, John Paparrizos, Themis Palpanas, and Michael J. Franklin. 2021. "SAND: streaming subsequence anomaly detection." *Proc. VLDB Endow.*, 14(10):1717–1729.

Campos, David, Tung Kieu, Chenjuan Guo, Feiteng Huang, Kai Zheng, Bin Yang, and Christian S. Jensen. 2022. "Unsupervised time series outlier detection with diversity-driven convolutional ensembles." *Proc. VLDB Endow.*, 15(3):611–623.

Chen, Yiru, and Silu Huang. 2021. "TSexplain: Surfacing evolving explanations for time series." *In Proceedings of the 2021 International Conference on Management of Data,* SIGMOD '21, page 2686–2690.

Cho, Sohee, Wonjoon Chang, Ginkyeng Lee, and Jaesik Choi. 2021. "Interpreting internal activation patterns in deep temporal neural networks by finding prototypes." *In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 158–166.

Dama, Fatoumata, and Christine Sinoquet. 2021. “Time Series Analysis and Modeling to Forecast: a Survey.”

Danielsen-Haces. 2018. “Digital twin development: condition monitoring and simulation comparison for the revolt autonomous model ship.”, Master’s thesis, Norwegian University of Science and Technology.

De Gooijer, Jan G., and Rob Hyndman. 2005. “25 Years of IIF Time Series Forecasting: A Selective Review.” *SSRN Electronic Journal.* 10.2139/ssrn.748904.

Guidotti, Riccardo, and Anna Monreale. 2021. “Designing shapelets for interpretable data agnostic classification.” *In Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES ’21, page 532–542.

Khelifati, Abdelouahab, Mourad Khayati, Philippe Cudre-Mauroux, Adrian Hanni, Qian Liu, and Manfred Hauswirth. 2021. “Vadetis: An explainable evaluator for anomaly detection techniques.” *IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2661–2664.

La Grange, Elgonda. 2018. "A Roadmap for Adopting a Digital Lifecycle Approach to Offshore Oil and Gas Production.", *Offshore Technology Conference*, vol. , pp. 15.

Lai, Kwei-Herng, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and XiaHu. 2021. “Revisiting time series outlier detection: Definitions and benchmarks.” *In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Leandro, Kennedy, Luís Bitencourt Jr, Guilherme Franzini, Alfredo Neto, Giovanni Amaral, Guilherme Rangel, Guilherme Martins, Edgard Malta, Raul Dotta and Paulo Videiro. 2021. “A methodology for development of a digital twin ship.”

Lipton, Zachary. 2015. “A Critical Review of Recurrent Neural Networks for Sequence Learning.”

Liu, Zhenyu, Zhengtong Zhu, Jing Gao and Cheng Xu. 2021. "Forecast Methods for Time Series Data: A Survey," in *IEEE Access*, vol. 9, pp. 91896-91912, doi: 10.1109/ACCESS.2021.3091162.

Liu, F. T., K. M. Ting and Z. -H. Zhou. 2008. "Isolation Forest," *Eighth IEEE International Conference on Data Mining*, pp. 413-422..

Medico, Roberto, Joeri Ruyssinck, Dirk Deschrijver, and Tom Dhaene. 2021. “Learning multivariate shapelets with multi-layer neural networks for interpretable timeseries classification.” *Adv. Data Anal. Classif.*, 15(4):911–936.

Moradi, Milad, and Matthias Samwald. 2021. “Post-hoc explanation of black-box classifiers using confident itemsets.” *Expert Syst. Appl.*, 165:113941.

Munkhdalai, Lkhagvadorj, Tsendsuren Munkhdalai, Kwang Ho Park, Tsatsral Amarbayasgalan, Erdenebileg Batbaatar, Hyun Woo Park, and Keun Ho Ryu. 2019. “An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series.” *IEEE Access*, 7:99099–99114.

Oviedo, Felipe, Zekun Ren, Shijing Sun, Charles M. Settens, Zhe Liu, Noor Titan Putri Hartono, Savitha Ramasamy, Brian L. DeCost, Siyu Isaac Parker Tian, Giuseppe Romano, Aaron Gilad Kusne, and Tonio Buonassisi. 2019. "Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks." *npj Computational Materials*, 5:1–9.

Pang, Guansong, Longbing Cao, Ling Chen, and Huan Liu. 2017. "Learning homophily couplings from non-iid data for joint feature selection and noise-resilient outlier detection." *In Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 2585–2591.

Pang, Guansong, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. 2021. "Deep learning for anomaly detection: A review." *ACM Comput. Surv.*, 54(2).

Paparrizos, John, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. 2022. "TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection." *Proc. VLDB Endow.*, 15(8):1697–1711.

[Prophet] https://facebook.github.io/prophet/

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?: Explaining the predictions of any classifier." *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144.

Rojat, Thomas, Raphael Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Diaz-Rodriguez. 2021. "Explainable artificial intelligence (xai) on timeseries data: A survey." *ArXiv*, abs/2104.00950.

Saadallah, Amal, Maryam Tavakol, and Katharina Morik. 2021. "An actor-critic ensemble aggregation model for time-series forecasting." *In IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2255–2260.

Sarkar, Anirban, Deepak Vijaykeerthy, Anindya Sarkar, and Vineeth N. Balasubramanian. 2022. "A framework for learning antehoc explainable models via concepts." 2022 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10276–10285.

Schmidl, Sebastian, Phillip Wenig, and Thorsten Papenbrock. 2022. "Anomaly detection in time series: A comprehensive evaluation." *Proc. VLDB Endow.*, 15(9):1779–1797.

Siddiqui, Md Amran, Alan Fern, Thomas G. Dietterich, and Weng-Keen Wong. 2019. "Sequential feature explanations for anomaly detection." *ACM Trans. Knowl. Discov. Data*, 13(1).

Theissler, Andreas, Francesco Spinnato, Udo Schlegel, and Riccardo Guidotti. 2022. "Explainable ai for time series classification: A review, taxonomy and research directions." *IEEE Access*, 10:100700–100724.

Tuli, Shreshth, Giuliano Casale, and Nicholas R. Jennings. Tranad. 2022. "Deep transformer networks for anomaly

detection in multivariate time series data.” *Proc. VLDB Endow.*, 15(6):1201–1214.

Vandewiele, Gilles, Femke Ongenae, and Filip De Turck. 2021. “Gendis: Genetic discovery of shapelets.” *Sensors*, 21(4).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. “Attention is all you need.” *In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 6000–6010.

Wen, Qingsong, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. “Transformers in time series: A survey.”

Wenig, Phillip, Sebastian Schmidl, and Thorsten Papenbrock. 2022. “Timeeval: A benchmarking toolkit for time series anomaly detection algorithms.” *Proc. VLDB Endow.*, 15(12):3678–3681.

Ye, Lexiang and Eamonn Keogh. 2009. “Time series shapelets: A new primitive for data mining.” *In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’09, page 947–956.