FORECASTING METEOROLOGICAL VARIABLES AND ANTICIPATING CLIMATIC ABERRATIONS OF AN OCEANIC BUOY USING A NEIGHBOUR BUOY

by

AMRUTH SAGAR KUPPILI

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2021

To my parents Vijaya Chandra, Bharathi and sister Ashmitha

Thank you for unbounded support and care!

Table of Contents

List of Tables

List of Figures

Abstract

Weather buoys or oceanic buoys are floating sensors that measure meteorological data. They are crucial decision-making infrastructures but are often a single point of failure. For instance, the Smart Atlantic Herring Cove Buoy (SMA-H) is owned by the Centre for Ocean Ventures and Entrepreneurship (COVE) and used by the Port of Halifax and Atlantic Pilotage Authority to support operational efficiency, safety, and situational awareness for marine transportation. The variables from the SMA-H buoy are judged by the marine pilots to safely drive the vessels into the port, considering the wind and tidal effects. The efficient movement of vessels is critical to the sustainability and reliability of the port, as delayed vessel movement results in high wasted costs and possible negative reputational impacts. Such an important buoy will be taken down for maintenance for a period of six weeks or may temporarily fail. Therefore, a redundancy model is required for the buoy during downtime to avoid loss and negative reputation. The previous work developed proof-of-concept machine learning models (SVM, Random Forest, and Neural Network) to check the feasibility of predicting one buoy using another buoy. The authors gathered information from different data sources and concluded ECCC buoy as the reliable source of input for the machine learning models to predict significant wave height and wind speed. The results from the prototype models were encouraging and motivated the present work to develop optimally functioning, production-ready models for additional variables, namely maximum wave height and wave period.

The main aim of the current study is to develop enhanced, optimal, and production-ready models for predicting meteorological variables of the SMA-H buoy, in addition to anticipating anomalies like strong winds and snowstorms. Five machine learning models were developed, four for predicting four variables (normal conditions) of the buoy and one for anticipating anomalies. The normal conditions model produced good results as compared to the state-of-art results in identifying regular weather conditions. Furthermore, the anomaly detection model produced superior results, accurately identifying 95 out of 100 anomaly instances.

Acknowledgements

I thank my supervisor Dr. Luis Torgo who has been my support system, without whom this thesis would not have been possible. I express my gratitude to him for being patient in handling my writing. His constant guidance has helped me polish my skills in predictive analysis, especially in forecasting time series. I also thank, DeepSense for funding my thesis and being at my reach all the time for technical support. I would like to thank my parents Vijaya Chandra Kuppili and Bharathi Kuppili, for always supporting me, and my sister Ashmitha Kuppili for encouraging me in every moment of life. Finally, a special thanks to Kayalvizhi for motivating me to accomplish my study.

Chapter 1

Introduction

With the massive rise in data production throughout the globe, concern about data availability has become trivial. The essential contributors of the data include Internet of Things, sensors, and recently, smartphones. As per the statistics, in 2020, people produced at least 2.5 exabytes (10^{18} zeroes) of data every day, and it is estimated that 463 exabytes will be generated by 2025 [20]. As of 2021 January, there are 2.6 billion internet users with 319 million new users to the internet. Currently available data is so huge that it takes a person approximately 181 million years to download all the data from the internet at 46 Mbps download speed. Furthermore, on a monetary perspective, the worth of data is estimated at $77 billion [26] by 2023. Numbers appear to grow more consistently, yet, the greater part of the organisations break down 12% of the information leaving the leftover 88% unanalysed [26].

There are different types of data, including nominal, ordinal, discrete and continuous. Time series data is a form of interval-based data that falls into either of the referenced categories depending on the usage. This thesis primarily uses time series data.

Time series data is a collection of observations accumulated over time. The separation between two observations in a time series is referred to as frequency and typically is hourly, weekly, monthly, quarterly, and so on. Examples include collecting weather data every hour, sales of a product for every month, and yearly revenues. Time series data is called univariate if a single variable is considered at each time step, and multivariate when more than one variable is included for analysis. Analysing such a temporal pile gives several important insights into the data, thereby delivering numerous advantages to the analyst.

Visualising includes capturing patterns and trends of a variable over time. Many organisations from different sectors rely on visualising historical data that involves user activity to achieve respective demands by forecasting prospective trends. While

the process of visualising the patterns of time series data is called time series analysis, the anticipation process is known as time series forecasting. Mentioned processes will be discussed in detail in Chapter 2.

1.1 Motivation

Time series data are regularly generated from numerous sensors, with weather buoys being one among the resources in the research. A weather buoy is a floating sensor that is moored to the ocean floor and detects meteorological factors such as significant wave height, wind speed, and wave period. They provide major assistance in decision-making for various coastal activities but are often a single point of failure. For instance, the Smart Atlantic Herring Cove Buoy (SMA-H buoy) [37] is owned by the Center for Ocean Ventures and Entrepreneurship [2] and used by Port of Halifax (PoH) [78] and Atlantic Pilotage Authority (APA) [1] to support operational efficiency, safety, and situational awareness of marine transport.

The responsibilities of the organisations include operating, maintaining, and administering the safe passage of vehicles into the port. They track variables from the SMA-H buoy like significant wave height, wave period, and wind speed to analyse the effects on the vessels and make decisions regarding the safe entry into the ports. The efficient movement of vessels is fundamental for the reliability of the port since delayed movement results in safety risks, higher costs and a poor reputation of the port. Hence, it is crucial to develop a temporary redundancy for the SMA-H buoy in the event of a failure. The basic idea of our study is to use machine learning to anticipate SMA-H buoy values using the ECCC (Environment and Climate Change Canada, [3]) buoy that is 13 km away in the open ocean.

Jesuseyi et al. [35] developed a proof-of-concept machine learning model to determine if the values of one buoy can be used to estimate the other buoy. In particular, the authors developed three simple machine learning models (Random Forest Regressor [18], Support Vector Regressor [4], Neural Network [102]) to predict significant wave height and average wind speed. This work concluded that Random Forest was the best performing algorithm with a mean squared error of 0.16 meters and 2.92 meters/second for significant wave height and average wind speed, respectively, with

smaller testing data than the current ones. The results demonstrated that it is achievable to anticipate one buoy using the other. These encouraging results as well as the hypothesis to use Machine Learning to anticipate rare events such as hurricanes and snowstorms inspired us to proceed with the work described in this thesis.

1.2 Goals and Objectives

The rising reliance on weather buoys has become frequent in many organisations for maritime transport. Owing to the disadvantages of sensor failure, the requirement for a temporary redundancy system has prodigiously increased. Hence, with the assumption of no substantial changes in weather conditions from the nearby buoy, an idea to implement a machine learning model to use proximal buoy (ECCC) for predicting the concerning buoy (SMA-H) is attempted in previous work [35]. Although with minimal configuration and testing, the prototype model rendered successful initial results.

The current study aims to develop a machine learning model that considers all the cyclical ups and downs like frequent patterns and delivers season-oriented outcomes throughout the year. Furthermore, a model capable of anticipating rare events such as hurricanes, snowstorms, and strong winds that deters the vessel movements is also regarded as one of the study's primary goals to take necessary actions to avoid damage to the port. To increase the interpretation speed of the results for faster decision-making, a real-time web page to visualise the results graphically was also developed, providing a way to schedule predictions every hour to make the machine learning model feed on live data. This kind of scheduling decreases manual intervention and automatically projects the results onto the page.

1.3 Research Methodology

A Long Short Term Memory (LSTM) [54] Neural Network capable of learning order dependence in sequence prediction problems serves as the foundation model for normal condition prediction and anomaly anticipation. Both procedures developed to address the two mentioned goals use ECCC data as features to predict SMA-H conditions. However, the model's structural organisation differed in achieving the demands of

the goals mentioned earlier. Specifically, two techniques are primarily employed for normal condition prediction to capture linear aspects such as trends and non-linear features such as data fluctuations. While LSTM is used to acquire insight into the non-linear side of the data like sudden intermediate fluctuations, Auto Regressive Integrated Moving Average (ARIMA) is used to capture the linear portion of the data [34]. We also propose a novel fusion model, Ens-LSTM, which focuses on learning a relationship between LSTM and ARIMA to combine linear and non-linear components in prediction. On the LSTM front, stacked LSTM is utilised to forecast SMA-H variables by effectively ordering many neural network layers, each made of LSTM nodes. ARIMA, which can learn patterns of a single variable (univariate), predicts a particular target variable. Therefore, four ARIMA models are utilised for four SMA-H variable anticipations. All the predictions are then mapped with the actual values and fed to Ens-LSTM for combined predictions.

For anomaly detection, an unsupervised technique using the Stacked LSTM autoencoder (Recon-LSTM-AE) is proposed. An autoencoder with the same output as input is utilised to extract important information from data. It feeds on normal data that is free of uncommon events and learns the normal data to the greatest extent possible. A threshold loss in training the model is formulated as a boundary for a normal event. Finally, an event is defined as a normal or anomaly based on the amount of loss incurred during reconstruction.

1.4 Scientific Contributions

The main contributions of the study include:

- A redundancy model for SMA-H buoy at the time of failure, which reduces maintenance costs besides producing efficient results, was developed. Eventually, after testing, this model can be deployed in different scenarios or environments where some physical buoys can be taken down within a specific vicinity once the model achieves reliable standards.
- Novel Fusion model is examined for the effective prediction of meteorological variables to detect linear and non-linear characteristics of the input ECCC data. This kind of implementation can be replicated with other systems apart from

detecting meteorological variables that have high dynamism, for instance determining behaviour of one monitoring system using other.

- A reconstruction error based method that utilises LSTM neural network to detect anomalies near the SMA-H buoy using ECCC buoy is proposed, and the efficiency is analysed. This kind of implementation is not only limited to determining meteorological anomalies but also in other alarming devices.

1.5 Outline

The thesis is organised as follows. Chapter 2 introduces the notion of time series data with methodologies to analyse and forecast future steps. Additionally, works involving time series data for anticipating future events in the maritime sector are presented. Chapter 3 specifies the details regarding the data used in the current study and describes the methodologies used to forecast normal circumstances and detect anomalies, besides mentioning the evaluation procedures. Furthermore, a description of methods for visualising predictions is presented. Later in Chapter 4, algorithms are applied to the data and analysed. The outcomes generated by the algorithm are provided and comparative results are also specified to visualise the efficiency of each model mentioned. Finally, Chapter 5 concludes the thesis by discussing the limitations and future work of the study.

Chapter 2

Background and Related Work

2.1 Background Concepts

A study on time series is similar to analysing other kinds of data with an additional dimension known as time. The significant characteristic of the time series dataset is their internal relation between the records; that is, the order of the data must not be modified throughout the analysis to preserve the temporal dependence. This chapter provides an in-depth analysis of time series and methods to analyse, forecast, and evaluate them.

The chapter initially discusses an introductory section, 2.1.1, that covers explanations of when time series predictions are helpful, accompanied by the various ways of statistical forecasting and different metrics used. Furthermore, Section 2.1.4 talks about estimation methods to test the significance of the metrics. A different application of detecting anomalies and ways of doing it is presented in 2.1.3, and 2.2 present works related to the current study in the maritime perspective. Finally, in Section 2.1.3 metrics related to anomaly detection are mentioned.

2.1.1 Time Series Analysis

A time series is a collection of repeated measurements considered over time. In other words, let Y be a time series record measured at time $t = 1$, $Y^1 = \{x_1^1, x_2^1, x_3^1....x_n^1\}$ where x_i^t is a measurement of a variable i at time t. Depending on the number of variables considered for analysis at a single point in time, the analysis is categorised into two types: Univariate or Multivariate time series. Both categories are of major interest in this thesis as our learning algorithms are based on these variants. A typical time series is observed over time at regular intervals. Time series data collection is massive because the sensor instrumentation is proliferating worldwide, generating a relentless stream of data to analyse. Analysing such large amounts of data is essential

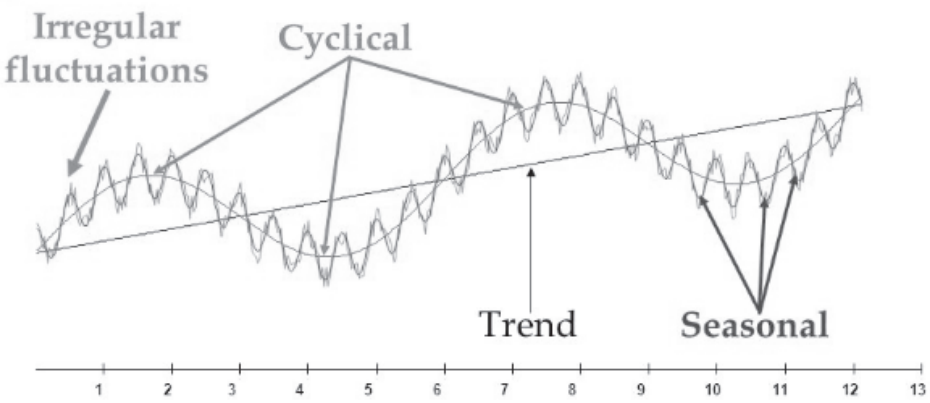

Figure 2.1: Components of Time Series (figure taken from [50])

in various fields of study; for instance, in health care, valuable insights are gained [95] by decomposing the time series and analysing the behaviour and properties. Also for efficient hospital maintenance during emergencies [57]; in transport efficient traffic regulation on roads [44] and seas [11] are achieved using seasonal ARIMA and neural networks; in marine, fish production is predicted [90]; in networking potential attacking patterns are identified [108] using ARIMA, and so on.

Time series data is a resource for crucial patterns, trends, and several underlying relations. Hence, such a temporal structure should be analysed carefully for good interpretations.

A straightforward way to analyse raw time series data is to plot a line chart and decompose it to visualise recognisable components like trends, seasonal, cyclic, and irregular patterns. A trend T^t is a progressive movement, either upward or downward, in the mean value of the variable—for example, variation in the price of a wine bottle over a considerable period of time. Seasonality S^t is a component of time series behavior that repeats on a frequent basis [72]—for instance, the demand for ice cream is higher in summer. While cyclical fluctuations C^t correspond to periodical but not seasonal variations, irregular variations I^t are non-random sources of series variation [5] and are a combination of seasonal and cyclical variations [50].

On an elemental front, a time series Y^t can be decomposed into three models: Additive Model, Multiplicative Model, and Mixed Model [5]. An additive model is chosen when the seasonal variations are almost constant, and a multiplicative model is chosen when the amplitude of seasonal variations are almost proportional to secular

trend [5]. The mixed model is a combination of additive and multiplicative model and they are written as follows:

$$\begin{aligned} \textbf{Additive Model} \qquad & Y^t = T^t + C^t + S^t + I^t \\ \textbf{Multiplicative Model} \qquad & Y^t = T^t \cdot C^t \cdot S^t \cdot I^t \\ \textbf{Mixed Model} \qquad & Y^t = T^t + (C^t \cdot S^t \cdot I^t) \end{aligned} \tag{2.1}$$

Stationarity is another significant characteristic of time series that is relevant to the earlier described properties. A stationary time series has constant mean and variance over time [83], implying it should possess no trend and seasonality. A conventional algorithm like ARIMA requires data to possess no trends or seasonality. Hence data should be made stationary before training. In contrast, if an algorithm assuming stationary data is fed with non-stationary time series, the model assumptions are violated, resulting in poor forecasts [**?**]. For example, More detailed analysis and preprocessing will be discussed in further chapters.

2.1.2 Forecasting Time Series

The process of estimating future values of any variable of interest using the existing historical records is often referred to as time series forecasting. The primary goal is to derive a relationship between the future and past events—this kind of forecasting is regularly used, for instance, in the commercial sector [55, 85] neural networks were used to develop marketing strategies by analysing the effects of promotion and in the non-commercial sector [42] to enhance living conditions by predicting future climatic disasters, in the government [24] to boost safety by making short-term forecasting of property crime, and in the private sector to increase revenue [84], and several other organisations to benefit their respective institutions. Although predicting future events remains consistent across various corporations, the prediction horizon may differ depending on the requirement. In other words, an ice cream vendor might be interested in next month's sales, while a construction firm may be interested in land prices for the next ten months. The former is called single-step prediction as it predicts the next recurrent value, and the latter is referred to as multi-step prediction, and multiple future events are predicted. This thesis is primarily concerned with single-step prediction.

There are various methods in the literature to model time series. This section explains the most commonly used statistical modeling techniques. An in-depth explanation of machine learning approaches used in this thesis will be given in later chapters.

Naïve Approach

The Naïve method estimates the next value of the series to be the same as the most recent observation. This kind of prediction is useful to capture the trends and patterns of the data. The forecast $\hat{y}_{t+1}$ is mathematically expressed [88] as:

$$\hat{y}_{i+1} = y_i \tag{2.2}$$

The main demerit of this approach is that the prediction range is constrained to one future step ahead because the approach just copies the previous value. Hence, the Naïve method can not predict any changes and is thus not useful for long range future predictions.

Moving Average Approach

The Moving Average model is another simple type of univariate time series forecasting. In this approach, the future prediction of a variable is determined as the mean of k past values. These k values are collectively called a window. The window size is determined by the number of observations that significantly impact the current observation. This method identifies trends based on the pattern captured by the moving average [15]. The smoothness of the trend is found to be directly proportional to the length of the window [83].

Let all the historic data be denoted by $y_1, y_2, ..., y_t$, then the future estimate $\hat{y}_{t+1}$ with past window of size k is given by:

$$\hat{y}_{t+1} = \frac{\sum_{p=t-k-1}^{t-1} y_p}{k} \tag{2.3}$$

To summarise, the wider the window, the smoother the peaks and valleys. However, it is often necessary to capture as many peaks as possible to identify the trends and patterns in the time series, so the window size cannot be too large or too small.

Exponential Smoothing Approach

This technique is similar to the moving average, but it varies in that decreasing weights are assigned to the previous observations, thereby giving more importance to the recent observation than the older one. This approach effectively tries to remove the noise from the data, resulting in more accurate forecasting [15]. Below formulation, for smoothing statistic, s_t is known as "Brown's simple exponential smoothing" and is algebraically written [17] as

$$\begin{aligned} s_0 &= y_0, \quad t = 0 \\ s_t &= \alpha y_t + (1-\alpha) s_{t-1}, \quad t > 0 \end{aligned} \tag{2.4}$$

where α is smoothing factor and $0 < \alpha < 1$. y_t is the current time series observation. The degree of smoothing is inversely proportional to the value of α [83]. The α value close to one has a less smoothing effect, resulting in higher importance to recent changes and vice versa. Exponential smoothing can be further extended to Double Smoothing, also called "Holt's Exponential Smoothing" and Triple Smoothing, also known as "Winters' Three Parameter Linear Seasonal Exponential Smoothing." While the former is used for trendy data, the latter is applied for both trend and seasonality. More intuitive explanations and mathematical relations are presented in work by Marco [83].

Auto Regressive Integrated Moving Average (ARIMA)

ARIMA is a predictive analysis model that analyses and predicts potential patterns based on time series information. It is a generalisation of the AutoRegressive Moving Average model (ARMA) that incorporates the concept of Integration to create ARIMA. The current study utilises ARIMA as one of the base models for anticipating normal conditions. This section provides a high-level overview of ARIMA's components. A more comprehensive description of how to select the orders of the constituents and underlying algebraic expressions is presented in Section 3.2.1.

- **Auto Regressive (AR)** is a model that forecasts an observation based on the dependency of the previous observations, also known as lagged observations.

- **Integrated (I)** is the order of differencing the current value with the lagged observation to render the series stationary. Unlike ARMA, which necessitates data to be in a stationary form, an Integrated mechanism is added to ARIMA to handle non-stationary data.
- **Moving Average (MA)** model uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Evaluation Metrics

Forecasting a potential time step isn't everything. The model should be assessed for efficiency and reliability. A model is claimed to be efficient and reliable only after meeting standard benchmarks. There are various ways to gauge the algorithm's performance, and metrics are one among those. This section gives an in-depth explanation of some of the metrics used in this thesis.

A metric is calculated based on the residual term. A *residual* is the difference between the predicted value and actual value and is sometimes referred to as error. The primary objective of majority of the time series models is to minimise the error value to the greatest extent possible so that the predictions are close to actual observations. The *residual* or *error* E for forecast $\hat{y}$ and actual observation y at time t is given by:

$$E_t = \hat{y}_t - y_t \tag{2.5}$$

Based on Equation 2.5, we will start with Mean Absolute Error (MAE). Mean Absolute Error is defined as the average absolute difference of the predicted and original value. It is given for n observations as:

$$MAE = \frac{\sum_{t=1}^{n} |E_t|}{n} = \frac{\sum_{t=1}^{n} |(\hat{y}_t - y_t)|}{n} \tag{2.6}$$

Mean Squared Error (MSE) is defined as the average squared difference of the residuals. Mean Squared Error has an advantage over Mean Absolute Error. MSE highly penalises for large errors and is more sensitive to outliers than MAE [10]. This is considered advantageous because a low MSE score would indicate a presence of potential outliers (anomalies), which is the goal of part of the thesis as discussed in Section 3.3.

MSE is also known as Mean Squared Prediction Error (MSPE). It is mathematically expressed for n observations as:

$$MSE = MSPE = \frac{\sum_{t=1}^{n}(E_t)^2}{n} = \frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n} \tag{2.7}$$

Root Mean Squared Error (RMSE) is defined as the root of the Mean Squared Error. It is also known as Root Mean Square Deviation (RMSD) and is given for n observations as:

$$RMSE = RMSD = \sqrt{\frac{\sum_{t=1}^{n}(E_t)^2}{n}} = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}} \tag{2.8}$$

All the metrics mentioned above have a significant shortcoming of being scale-dependent; for example, an MAE, MSE, or RMSE of five does not reveal any detail about the model's goodness. Metric interpretation requires prior knowledge of the average value of the variable. Percentage-based metrics, which measure accuracy in terms of percentage, address the above disadvantage and are mainly used [61] to compare the errors between differently scaled datasets, thereby giving a comprehensive insight into the predictions. An example of percentage-based errors is the Mean Absolute Percentage Error (MAPE).

Mean Absolute Percentage Error is defined as the average of all the residuals divided with their respective actual values. It is given as:

$$MAPE = 100 * \frac{\sum_{t=1}^{n}\left|\frac{E_t}{y_t}\right|}{n} = 100 * \frac{\sum_{t=1}^{n}\left|\frac{\hat{y}_t - y_t}{y_t}\right|}{n} \tag{2.9}$$

MAPE also has a major drawback: it produces undefined results if the actual values are around zero [61]. Hyndman et al. [56] proposed a different scale-independent error to cope with the above disadvantage and known as Mean Absolute Scaled Error (MASE). It is the ratio of mean absolute error of forecast values to the mean absolute error of in-sample one step naive forecast (error in predicting lagged observation). Furthermore, MASE considers seasonality into the equation to give seasonal-specific metrics. MASE is given by:

$$MASE = \frac{\sum_{t=1}^{n}\left|\frac{E_t}{\frac{1}{n-m}\sum_{t=m+1}^{n}|y_t - y_{t-m}|}\right|}{n}, \tag{2.10}$$

where m = seasonal period, and $m = 1$ for non seasonal data.

Note: All the mentioned metrics in Equations 2.6, 2.7, 2.9, 4.1 are interpreted as, the lower the value of the metric, the better the score is.

2.1.3 Anomaly Detection

In several machine learning processes, the values that deviate from the norm, commonly called outliers, are considered hindrances in improving the model's performance. For example, Li et al. [63] in the context of classification, and Patel et al. [80] in the context of clustering regarded outliers as they were degrading the performance of the model and thus eliminated them. Conversely, a prime motive of identifying outliers is called outlier detection or Anomaly Detection. The real-world examples of its use cases include monitoring applications like video surveillance [106, 76] to correct field of view of cameras and fraud detection [81] to detect money laundering. Depending on context and domain, these abnormal points are referred to as outliers, anomalies, and novelties. In this thesis, the term used is anomaly.

Anomalies are categorised into four different types on the basis of their existence, and they are given as:

- **Point or Global Anomalies** are anomalies where a single point deviates from the rest of the data. Example: A transaction from a credit card can be identified as an anomaly if the amount spent significantly deviates from the other transactions.
- **Collective Anomalies** can be defined as data points that seem abnormal when grouped but appear regular when considered individually. Example: A single person purchasing a car is considered normal, while the entire town purchasing a car on the same day could be considered an anomaly.
- **Contextual or conditional Anomalies** are referred to as anomalies with context as an additional attribute. A Point and Collective anomaly can be called Contextual if they are determined by the context. A time series anomalies can be considered contextual as they are restricted by the temporal background. Nonseasonal demand, such as high sales of snow jackets in the autumn, is an example of a contextual exception because it would not be an anomaly in winter. In other words, it is an anomaly in autumn because of the seasonal context.

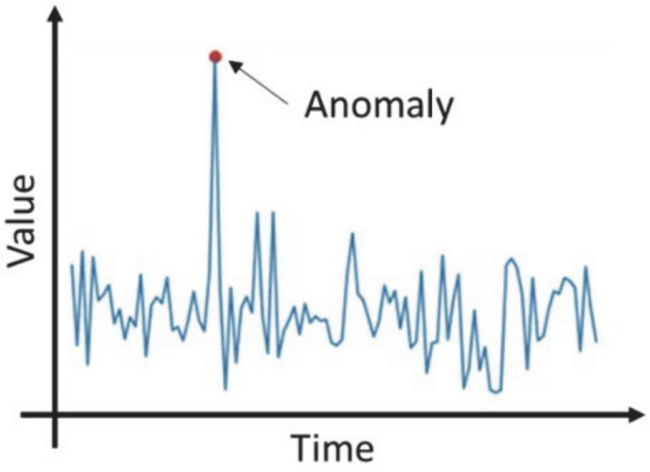

Figure 2.2: Time series with anomaly (figure taken from [12])

- **Change points** are defined as initial points where a change of pattern begins. This is useful to alarm triggering situations. For example, detecting the initiating point for a sensor malfunction may provide invaluable feedback to businesses to minimise losses and downtime.

Time Series Anomaly Detection

Anomalies in time series are data points that do not agree with the common seasonal, trend, or cyclic pattern and deviate from normal behavior. Importantly, the perspective of anomaly varies when identified in time series as they have a temporal dependency in the equation. For example, a rise in temperature in winter would be an anomaly, whereas it is normal in summer. On a high level, the anomalies are characterised by the previous values and trends. Hence, deviations are found in terms of statistical properties with the rest of the sequence, especially in time series. An anomaly can be visualised in Figure 2.2, deviating from the rest of the pattern. A wide range of methods is available to recognize irregularities, from statistical approaches to machine learning approaches, depending on the complexity and data requirements.

Statistical Anomaly Detection techniques

Statistical approaches analyse data based on the statistical properties of the distribution of the data and classify the novelties present in data. The robustness of these

approaches is dependent on the quality and amount of the data as the parameters are estimated from the distribution of training data [68]. They are simple and easy to compute. This section sums up certain straightforward and compelling mathematical methods as explained by Alam in a blog [12] for distinguishing abnormalities.

Z-score: A z-score in statistical distribution determines how far a data point is from the mean of the entire distribution in terms of standard deviations. A z-score of one indicates that the data instance is one standard deviation away from the data's mean. A data point is regarded as an anomaly if the z-score exceeds a threshold number of standard deviations. Hence, this is also known as anomaly score in some literature [60]. Statistical values such as mean and standard deviation are required to measure a z-score for a data point. Z-score of variable x for mean μ and standard deviation σ is mathematically expressed as:

$$z\text{-}score = \frac{x - \mu}{\sigma} \tag{2.11}$$

This approach is easy to calculate but comes with a limitation. The z-score produces poor results if the dataset is minimal or doesn't follow the Gaussian distribution. The Z-score relies on the mean and standard deviation of the data to measure dispersion which is not robust to the outliers.

Modified Z-score: To overcome the shortcomings of the z-score, certain modifications are imposed on a traditional z-score approach. Owing to the sensitivity of the dependent variable to outliers, the median is used in place of the mean. Furthermore, Standard Deviation is replaced with Median Absolute Deviation, and the constant value is multiplied to equate it to Gaussian Distribution. The Median Absolute Deviation of a gaussian is approximately equal to 0.6745 times the Standard Deviation. The advantage of using modifies Z-score is that it does not expect data to be in the gaussian form as the median is considered, and also, it is robust to outliers. Summing up altogether, Modified z-score for mean μ is given by

$$Modified\ Z\text{-}score = \frac{0.6745 * (x - \mu)}{Median\ Absolute\ Deviation} \tag{2.12}$$

Interquartile Range: Initially, the data is sorted and grouped into quarters called quartiles. The center quartile is referred to as the median quartile, while the left 25^{th} percentile and right 75^{th} percentile quartiles are referred to as the First (Q_1) and Third (Q_3) quartiles, respectively. The difference of (Q_1) and (Q_3) is known as Inter Quartile Range (IQR). For comparison, an example Figure 2.3 representing the portrayal of quartiles are shown. The concept behind using IQR to detect anomalies is that a point is called an anomaly if it is far from the (Q_1) and (Q_3) quartiles. How much further is given mathematically as 1.5 times the Inter Quartile Range. In other words, if a point is 1.5 times the IQR away from the First and Third Quartiles, it is classified as an exception. 1.5 is a discerning constant that controls the sensitivity range [21]. While less than 1.5 would perceive normal points as outliers, a greater number would ignore outliers. The upper and lower bounds for the thresholds are numerically given as:

$$\begin{aligned} \textit{Inter Quartile Range } (IQR) &= Q_3 - Q_1 \\ \textit{lower bound} &= Q_1 - 1.5 * (IQR) \\ \textit{upper bound} &= Q_3 + 1.5 * (IQR) \end{aligned} \tag{2.13}$$

The important limitation of IQR is that it assumes data to be normally distributed according to norms of gaussian distribution. So, the data have to be scaled before identifying outliers.

Categories of Machine Learning Anomaly Detection techniques

With the rapid production of huge data and increased dependence among various data variables, the complexity in determining dependencies has also increased. Classifying variations solely on mathematical properties could be complex for huge data. Thus, more comprehensive interpreters that can capture non-linear dependencies must capture the underlying relations among multiple variables. Machine learning algorithms are one of the approaches that use nonlinear entities to capture complex patterns. However, there are different variations of machine learning methodologies, and they are classified into three categories based on the nature of the approach. This section summarises such categories of approaches with algorithm examples.

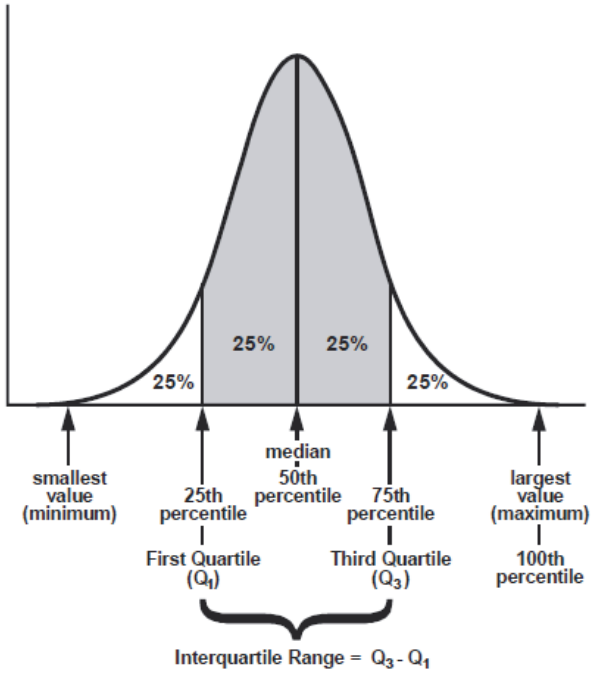

Figure 2.3: Depiction of Quartiles taken from [36]

Supervised Learning: Supervised Learning involves a simple technique called classification that models on labelled data. The main aim of this category of algorithms is to identify a given instance into one of the labelled categories. This approach can be used to model both normality and abnormality. Hence, any machine learning classifier like Decision Tree Classifier [40], etc., can be used.

The main drawback of this category is that it can be difficult and time consuming to obtain labelled data. Many data sources do not generate data with labels, making it harder to categorise every data point. Additionally, this approach requires sufficient examples for every label to capture the relationships with the independent variables. In other words, having an equal number of abnormal labels is not common, which results in an imbalanced dataset and thus requires augmentation. Augmentation is a process of increasing the number of minimally numbered instances either by appropriately tweaking the variables or just duplicating them to balance the number of instances for every label. An augmentation technique like SMOTE [22] is required to augment the data and obtain better results. However, the cost for misclassifying a rare instance is often higher and misguides the model, which is not recommended, especially in meteorology. For instance, the current study involves making predictions

that are useful for marine pilots to decide whether to bring vessels into the ports. Any wrongful determination in predictions results in negative impacts and additional costs.

Unsupervised Learning: Unsupervised Learning has no labels in the data, and therefore only limited pre-processing is needed as labeling is not required. Syarif et al. [98] demonstrated that when the anomalies were greater, clustering algorithms like k-Means [64], improved k-Means [75] yielded more accurate results than classification approaches (naïve bayes, decision tree). The fundamental theory is that clustering-driven algorithms rate data points based on the inherent properties of the dataset. Characteristics like distances and densities give the estimation to determine whether a point is normal or an anomaly. Initially, the data points are clustered with respective centroid values for training data. So, when a new point in testing data arrives, the distance of the point and nearest centroid value is calculated, and if it is high, it is designated as an anomaly. In some techniques, densities are used in place of distances to determine anomalies. Depending on the nature of separation, anomalies are again categorised as global and local in clustering techniques. *Global* are the points farther away from the dense regions, whereas *Local* outliers are marginally separated from the dense region.

However, there are limitations to clustering techniques as well. Markus et al. [45] presented a comparative study on various unsupervised anomaly detection techniques like k-Nearest Neighbor (k-NN), Local Outlier Factor (LOF) and concluded that clustering algorithms like LOF performed poorly to identify global outliers. Finally, it is critical to choose algorithms depending on how distant anomalies need to be detected. If the requirement is unknown, global-based clustering strategies can be used. Markus [45] also proposes that for global outliers, nearest-neighbor algorithms are used and LOF for local outliers.

Another unsupervised learning approach is explored, where the learning model is trained with normal data instances. Maximum training loss is considered a normalcy boundary. Finally, an instance from testing data having a loss greater than the boundary is regarded as an anomaly. Unsupervised algorithms like Autoencoders are used to capture the patterns of the normal data. Such a method is very useful [93, 103] for problems with a lot of regular data and a few cases of unusual events. This type

of approach is of major interest in this study, and an extensive walk-through about the exact methodology will be presented in Section 3.3.2 and a comprehensive flow is depicted in Figure 3.23 in the mentioned section.

Semi-supervised Detection: A semi-supervised process involves both supervised and unsupervised approaches to identify anomalies. Given the rarity of odd cases, this method attempts to capture the usual behavior of the data to the fullest degree possible and detects exceptions where they deviate from the standard, similar to unsupervised learning. On the contrary, this approach uses supervised learning to capture normal behavior. Firstly, supervised algorithms like LSTM and ARIMA are used to learn normal behavior patterns, and the errors are bounded using an unsupervised method. Thus it is named semi-supervised. Figure 2.4 depicts a high-level flow map of how the semi-supervised solution works. A more intuitive procedure is given in further chapters.

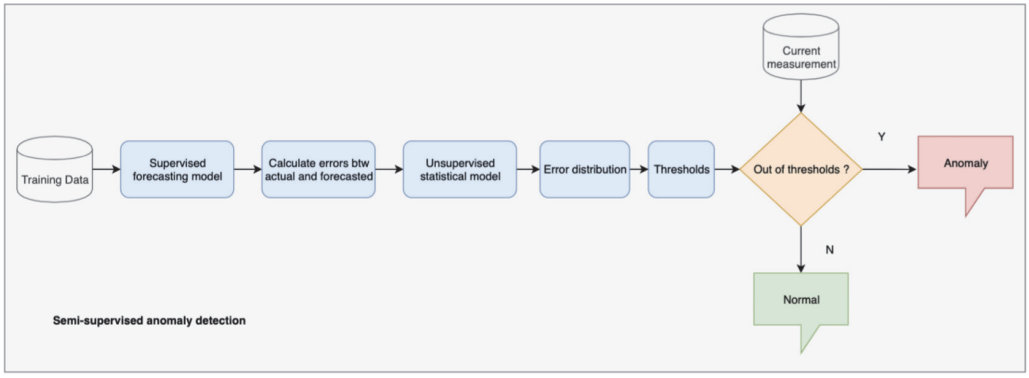

Figure 2.4: Fundamentals of Semi-supervised algorithm, figure taken from [100]

Anomaly Detection Metrics

Regardless of the path chosen, the end goal is to classify a data instance into normal or anomaly. As a result, this problem can be considered a form of Binary Classification paired with time when analyzing the model. Precision, Recall, and F1-score are considered potential indicators for evaluating this model and will be explained shortly. Accuracy is not recommended due to the imbalanced nature of the data for rare event instances. In other words, Accuracy evaluates the model's ability to detect

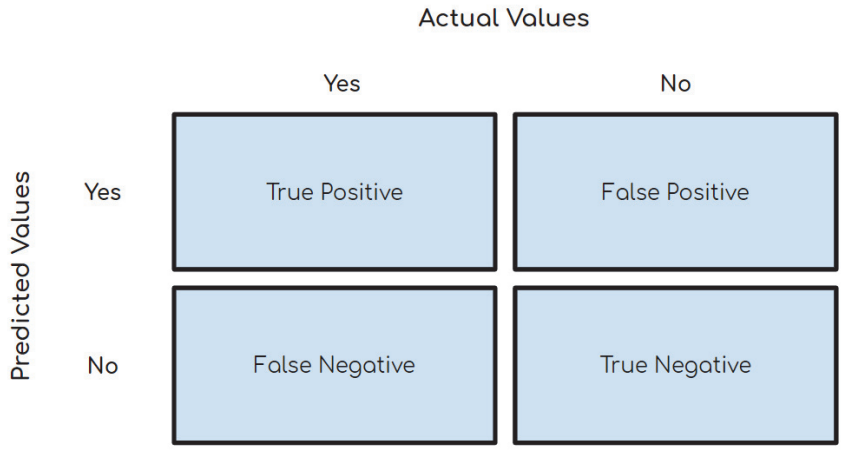

Figure 2.5: Confusion Matrix

both anomalies and normal instances on the whole, which could be biased towards the normal instances. For example, if there are 98 normal and two rare instances in a testing data, Accuracy projects a 98% score for accurately forecasting normal and completely missing anomalies, which is a warning. Therefore, Accuracy is not considered in the study. All the mentioned metrics are based on a confusion matrix that describes the performance of a classification model. A typical confusion matrix is shown in Figure 2.5.

Let True Positive TP, True Negative TN, False Positive FP, False Negative FN be the constituents of the confusion matrix. Precision gives the accuracy of the model in predicting positives out of the total predicted positives. It is given by:

$$Precision = \frac{TP}{TP + FP} \tag{2.14}$$

Recall is the measure to determine how correctly the model captures actual positives in the test data. It is termed as:

$$Recall = \frac{TP}{TP + FN} \tag{2.15}$$

F1-score is used in scenarios where both recall and precision are important. It is essentially the harmonic mean of Precision and Recall and can be calculated as follows:

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.16}$$

Note: All scores close to one indicate that the model is excellent at estimating, while zero indicates that the model is worse.

2.1.4 Time Series Performance Estimation

Performance estimation is a process of estimating the loss of a predictive model on unobserved data. By loss we mean the value of the metrics discussed in Section 2.1.2. Cross Validation is a standard approach for estimating performance if the data is independent and identically distributed (i.i.d) [43]. Also, Cross Validation involves data shuffling for accurate estimation, which is not valid in the case of time series since the order of observations is mandatory to preserve the temporal dependency while forecasting. Cerqueira et al. [104] examined out-of-sample methods and cross validation methodologies on 62 real-world and three artificial time series. An important inference from the work of Cerqueira [104] is that, although cross validation can be extended to stationary time series, the out-of-sample method produced more accurate estimates when dealing with non-stationary time series. This subsection presents some of the out-of-sample and prequential performance estimation techniques discussed by Cerqueira [104].

Out-of-sample Method

The general idea of the Out-of-sample (OOS) method is to leave the trailing data (ending data instances) for testing and train the model with the piloting data (starting data instances). Although this method loses information in the interest of testing, the temporal order is preserved, allowing the model to capture intermediate patterns between the observations.

The data is divided into two sections in a simple OOS method: training data and testing data with no shuffling. In the context of the split point, it is critical to choose a correct point such that the model has enough data to derive the most information from the training data. Furthermore, the splitting point is the primary attribute that distinguishes various estimation methods as listed below.

Figure 2.6: Simple Holdout Procedure (figure taken from [104])

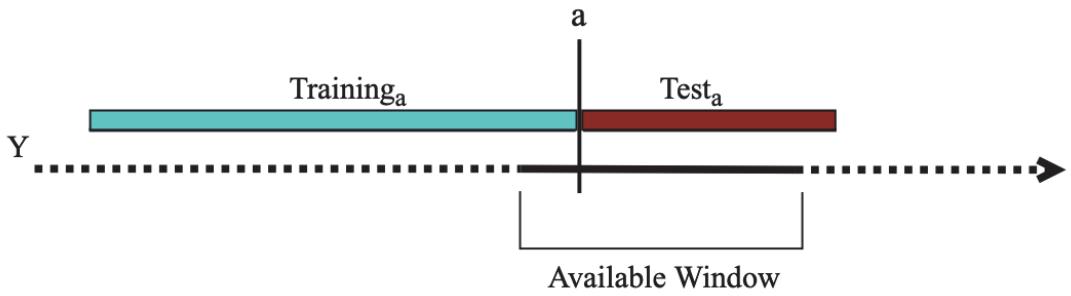

Figure 2.7: Repeated Holdout Procedure (figure taken from [104])

- **Holdout** a simple data splitting technique where the data is partitioned at a random position that maximises the output. Holdout is depicted in Figure 2.6.
- **Repeated Holdout** is a split point selection procedure, in which a splitting point is selected in a sliding window that sub-samples the data, possibly overlapping data. Figure 2.7 shows an example of a single iteration of repeated holdout procedure. A breaking point 'a' is selected from the window, and data is divided according to training and testing size constraints.

Prequential Method

The Prequential or interleaved-test-then-train approach is another performance estimation technique that is mainly used in data stream mining. In this technique, the unseen data is used for testing before training the model. This is visualised as a sequential block of instances [71], and then variants are developed based on the utilisation of the blocks, as shown in Figure 2.8. The below is an overview of the variants:

- **Prequential Blocks** (Preq-Bls) is the first Prequential variant. As shown on the left side of Figure 2.8, the first two blocks are utilised for testing and training

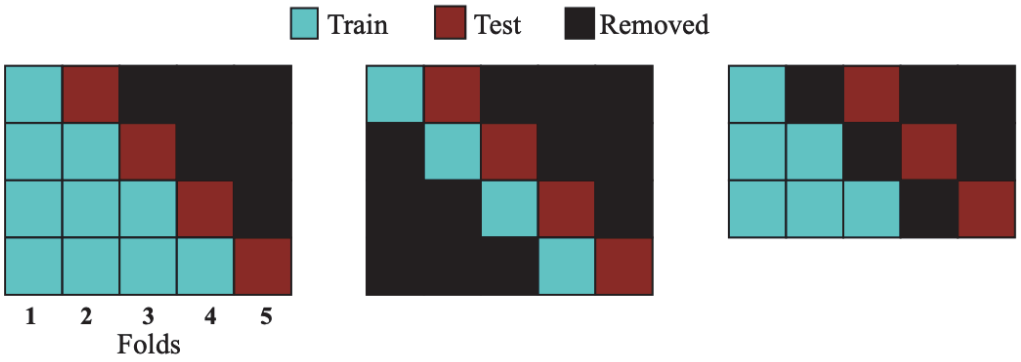

Figure 2.8: Variants of Prequential Approach (figure taken from [104])

in the initial iteration. The first block is merged with the second block in the following iteration, and new unseen data is used for testing. This procedure is repeated until the whole data is used.

- **Prequential Slide Blocks** (Preq-Sld-Bls) is another variant of Prequential, where, unlike Preq-Bls, the older blocks are ignored in a sliding window fashion rather than overlapping, as shown in the center depiction of Figure 2.8. The intuition is that the model does not need to be trained on older data deprecated in stationary time series.
- **Prequential Blocks Gap** (Preq-Bls-Gap) as seen on the right side of Figure 2.8 is the final variant of Prequential, where a gap block is introduced between training and testing data to reduce the dependency between them.

2.2 Related Work in Maritime Sector

Since this inception, time series forecasting has spread across a wide range of sectors. Meteorology is one of them and is a primary area of focus in this research. Weather forecasting is a long-standing problem; in 1936, McNish [69] attempted to compare meteorological variables of one source from another; in 1951, Craddock [27] used statistical techniques to predict the weather. Furthermore, multitudes of studies in weather forecasting have emerged for a variety of causes. Studies include predicting regular weather conditions to help sailors, anticipating abnormal conditions like storms, hurricanes, and winds to warn people living along the coast. While Section

2.1.2 discussed statistical methods to deal with time series, this section gives some related references to the current study in terms of machine learning in maritime.

Time Series Prediction in Maritime Domain

Integrating machine learning techniques with maritime has played an important role in assisting the movement of vessels and ships into the ports besides helping coastal activities. Mahjoobi et al. [66] in 2009 used Support Vector Machine (SVM) to predict significant wave height with a computational time constraint and made a comparative analysis with Artificial Neural Network. The study results show that SVM produced better estimations and that its errors are marginally lower than Neural Network. More recently, in 2018, Petros et al. [59] used a simple linear regression technique and deployed it on a micro-controller to predict short-term wind speeds and achieved promising results with 2.165 MSE. This portable machine learning equipment that uses minimal computation was used onboard a boat. Despite the limitations in computational capacity, previously mentioned studies produced significant results in maritime data. These studies inspire us to achieve better outcomes with improved computing capacity and volume of data, as in our case.

From a deeper perspective, all variables in the current study are inclined to Meteorology, allowing us to review from that viewpoint. Nikita et al. [96] used ARIMA on meteorological data to forecast rainfall and temperature for the next fifteen years and gained notable results with mean squared errors of less than one. However, the study is used for far-future predictions, which is converse to our requirement. For a short-range prediction, Bari et al. [14] used Seasonal ARIMA to estimate monthly precipitation with a 95% confidence interval to gain reliable results in anticipating floods. On the Neural Network side, Karevan et al. [58] and Akram et al. [110] demonstrated the advantages of having improved performance for LSTM's meteorological estimations. Zheng et al. [112] in their research exhibited the use of combining ARIMA with various machine learning algorithms like LSTM, Random Forest, SVM to forecast air quality and concluded that the ensembling architecture produced improved results than individual models. Ernesta et al. [47] applied ARIMA to predict wind speed and obtained noteworthy results with acceptable mean squared error for short duration forecasts and suggested coupling ARIMA with another model for better output.

This supported the idea of having a hybrid model that is ARIMA-LSTM.

In terms of predictive ability, ensembles are proven to be more efficient in weather forecasting. Murray [74] described the importance of ensemble forecasting and the substantial results it produced in analysing space weather. Also, Mahabub et al. [65] demonstrated the efficacy of using ensembles in weather forecasting using machine learning algorithms like Decision tree Regressor and Category Boosting. Recently in 2021, although not in the field of maritime, Emmanuel et al. [28] proposed a combination of ARIMA and LSTM to forecast Exports in Indonesia. This hybrid model outperformed individual models with lower mean squared, and root mean squared errors, thereby complementing a fusion model.

Maritime Anomaly Detection

Anomaly Detection can be regarded as a crucial backup because of its unparalleled advantages, particularly in coastal activities. Extensive research has been in operation to identify anomalies to prevent incurring loss. Sandeep et al. [97], for example, used two traditional Artificial Neural Networks, one for distinguishing and the other for determining the cause of anomalies in a ship's transponder operations, and he was 99 percent precise in both cases. This is a conventional way of classifying anomalies in the supervised anomaly detection method. On an unsupervised front, Houxiang et al. [33] used LSTM based Variational Autoencoder to maritime data in a reconstruction-based fault detection algorithm. This approach yielded positive results. A similar approach was implemented in other fields; for instance, Pankaj et al. [67], and Nguyen et al. [77] used autoencoders to detect sensor and retail management anomalies. Furthermore, Oleksandr et al. [87] also produced motivating results using the same technique on the sound events dataset. Thus, the current research focuses on implementing the same because of its usefulness and lack of consideration on applying the same to maritime data.

2.3 Concluding Remarks

Researchers are in constant pursuit to achieve higher accuracy in modeling any inconsistent data that benefits the prediction. Time series estimations are mainly used to solve crucial real-world problems to make life better and easier. The current work

focuses on solving one of those real-world problems caused by an oceanic buoy's maintenance. A buoy that plays a pivotal role in decision-making at the ports for marine transit is used frequently by port authorities and incurs a loss to ports during downtime. Hence, a redundancy machine learning algorithm is proposed to predict the buoy's values using a proximal secondary buoy. Another model is also developed to anticipate calamities like strong winds that hinder the vessel movement. The study is a time series problem, and machine learning methodologies are implemented for meteorological estimations and anomaly detection. The coming chapters discuss the different algorithms utilised in the thesis and the comparison of their performance to develop a redundancy model.

Chapter 3

Materials and Methods

In Chapter 2 we have discussed some classical approaches of forecasting time series and generic ways of evaluating a time series model. Additionally, statistical methods of identifying anomalies in time series and their metrics were also discussed. The current chapter drives the reader along a pipeline from dataset acquisition to final execution, covering different facets of the machine learning methodologies used in our study. In particular, Section 3.1 presents the process of obtaining the dataset and the pre-processing steps performed after acquisition. Sections 3.2 and 3.3 give a concise discussion of the techniques used in forecasting normal conditions and anomalies, respectively. Finally, Section 3.4 presents screenshots of the Web application that was designed for real-time visualization of the predictions.

3.1 Dataset

The main aim of the current study is to predict variables and identify anomalies at the Herring Cove Buoy (also referred to as SMA-H buoy) using a nearby buoy. As mentioned earlier, this study is a continuation of Jesuseyi et al. [35], who acquired datasets from multiple sources and filtered them to finalise a reliable source useful for predicting the target variables (Herring Cove). This section gives an overview of the authors' steps to collect, pre-process and create a training set.

Initially, raw data was collected from three buoys and five land stations. Data access methods differed between datasets. SMA-H buoy and some land stations provide automated public access through ERDDAP. ERDDAP is a data server that provides a simple, consistent way of downloading datasets in various formats. Other buoy data was received from experts directly through e-mails and website downloads. The collected datasets [35] are described in Table 3.1. Some datasets are excluded because of inconsistency, and some do not have enough useful information for prediction. Detailed reasons for inclusion and exclusion of a specific dataset are explained in the

Dataset Name	Type	Description	status
SMA-H	Buoy	96,934 rows, 55 columns	Target
ECCC (Environment and Climate Change Canada)	Buoy	122,367 rows, 23 columns	Feature
AZMP_HLX	Buoy	15,706 rows, 40 columns	Excluded
Shearwater	Land Station	59,880 rows, 28 columns	Excluded
Osborne Head	Land Station	49,704 rows, 28 columns	Excluded
Halifax Pier 31	Land Station	259,631 rows, 7 columns	Excluded
Halifax Fairview	Land Station	186,344 rows, 11 columns	Excluded
Halifax Pier 9c	Land Station	226,544 rows, 12 columns	Excluded

Table 3.1: Description of Datasets (taken from [35])

previous work [35]. The ranges mentioned in the table are till 2020 January.

Our study employs 10 data variables, six of which serve as features and four as target variables. All the variables are repeatedly measured according to their respective buoy time periods, once in an hour for ECCC and twice for SMA-H buoy. A short description including source and usage is presented in Table 3.2 and few essential terms are as follows:

- Significant wave height: The average wave height from crest to trough of highest one-third of the waves.
- Maximum wave height: The highest vertical difference of crest and trough of a wave.
- Wave peak period: The wave period of the most energetic waves in the wave spectrum.
- Gust wind speed: The speed of a wind that is increased above the average.
- Average wind speed: The mean speed of the wind considered in a spectrum i.e one hour.

- MEDS: Marine Environmental Data Section [31], a data repository government entity.

Variable Notation	Variable Name	Usage	source
VCAR	Characteristic significant wave height (m), calculated by MEDS	Feature	ECCC
VWH$	Characteristic significant wave height (m), reported by buoy	Feature	ECCC
VCMX	Maximum zero crossing wave height (m), reported by buoy	Feature	ECCC
VTP$	Wave spectrum peak period (s), reported by buoy	Feature	ECCC
WSPD	Horizontal wind speed (m/s)	Feature	ECCC
GSPD	Gust wind speed (m/s)	Feature	ECCC
wave_ht_sig unit(m)	significant wave height (m)	Target	SMA-H
wind_spd_avg unit(m s-1)	Average wind speed (m/s)	Target	SMA-H
wave_ht_max unit(m)	Maximum wave height (m)	Target	SMA-H
wave_period_max unit(s)	Maximum wave period (s)	Target	SMA-H

Table 3.2: Description of current dataset

3.1.1 Data Preprocessing

On a high level, the previous work was mainly concentrated on checking the feasibility of predicting buoy values using a nearby buoy and employed all the required data cleaning steps. Pre-processing included eliminating noise, mapping time stamps between different datasets, creating hourly and three-hour datasets, and visualising. Depending on the relevance, visualisations helped eliminate few datasets and concluded that the ECCC buoy dataset should be a workable dataset for predicting the target. In addition, the finalised dataset was trained on Random Forest Regressor, SVM, and Neural Network to check the feasibility and obtained promising results implying success in feasibility checks.

The current research focuses on developing an optimized and efficient method for forecasting an SMA-H buoy's meteorological variables and demonstrates an anomaly detection approach for predicting uncommon event occurrences such as hurricanes, snowstorms, and severe winds that impede marine transportation. Based on the dataset from previous work generated by bridging multiple data resources, novel methodologies are implemented to accomplish the mentioned goals.

The used dataset, ECCC, ranged from November 2013 to June 2020. After analysing the dataset, it was discovered that ECCC was unavailable for more than three years from August 2015 to October 2018 and did not have values in that period. One possible solution is to trim the data before 2018, resulting in the loss of 2 years of essential information. To avoid the loss, the three-year nulls were removed to use the available data efficiently, and the 2018 data was appended to 2015. Later, a total of around 240 intermediate missing records were linearly interpolated based on the surrounding values. The dataset comprises varying scales, and it requires scaling, a transformation technique that translates each column to a specified range. Transformation is essential because machine learning models only consider numbers ignoring the units. So, for example, 100 liters of milk is given lower preference than the 101 miles/second, which are not comparable. MinMax scaler with range (-1,1) was selected to correspond with the *tanh* activation function range in the LSTM Network. In Table 3.2, wave_ht_sig unit(m) (Significant Wave Height), wind_spd_avg unit(m s-1) (Average Wind Speed), wave_ht_max unit(m) (Maximum Wave Height), and, wave_period_max unit(s) (Maximum Wave Period) are the target variables from SMA-H buoy and the rest are predictor variables from ECCC. Figure 3.1 shows sample data after scaling. Furthermore, correlation among data columns was studied to gain better insights into data and was presented for reference in Figure 3.2. Correlations aid in choosing related variables when training an ARIMA model.

ARIMA is designed to predict univariate data based on stationary data and thus requires closer examination to check stationarity. While this paragraph discusses the ways of examining stationarity, a brief explanation about ARIMA is presented in Section 3.2.1. Recall that a time series is stationary if the statistical properties like mean and variance do not change over time. In other words, a stationary time series has no trend or seasonality. Thus, trends and seasonality in the data

index	VCAR	VWH$	VCMX	VTP$	WSPD	GSPD	wave_ht_sig unit(m)	wind_spd_avg unit(m s-1)	wave_ht_max unit(m)	wave_period_max unit(s)
2013-11-07 16:00:00	-0.599476	-0.600000	-0.814103	-0.430657	-0.531915	-0.479675	-0.789474	-0.126615	-0.807229	-0.65
2013-11-07 17:00:00	-0.604712	-0.600000	-0.846154	-0.445255	-0.627660	-0.552846	-0.768421	-0.421189	-0.771084	-0.58
2013-11-07 18:00:00	-0.672775	-0.680000	-0.878205	-0.459854	-0.648936	-0.650407	-0.747368	-0.467700	-0.746988	-0.57
2013-11-07 19:00:00	-0.643979	-0.653333	-0.865385	-0.328467	0.000000	-0.048780	-0.684211	-0.400517	-0.686747	-0.54
2013-11-07 20:00:00	-0.484293	-0.493333	-0.769231	-0.357664	0.180851	0.065041	-0.747368	-0.002584	-0.759036	-0.56

Figure 3.1: Sample Data Set

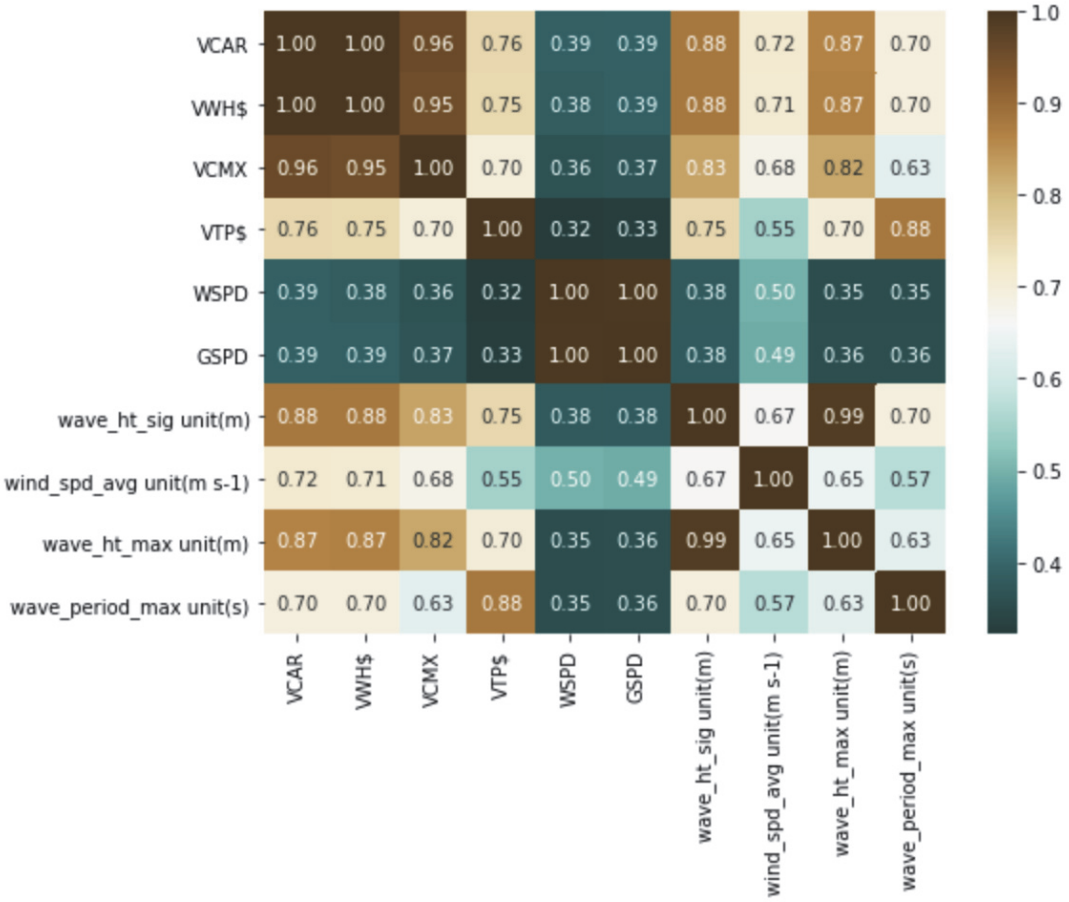

Figure 3.2: Correlation between variables in data

should be removed before feeding it to the algorithm. Stationarity in a time series is verified using statistical tests like Augmented Dickey-Fuller (ADF) test and Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test, which will be explained shortly. Later, the differencing method removes seasonality and trend, which is performed

when defining a model. More about differencing will be discussed in Section 3.2.2. The earlier mentioned statistical methods are based on the unit root, which is a characteristic of time series that exists if the value of α is one in the Equation 3.1 for the value of time series at time t and exogenous variables X_e [86] (variables that are determined independently of the model, which influence the determining variable).

$$Y_t = \alpha Y_{t-1} + \beta X_e + \epsilon_t \tag{3.1}$$

An overview of the tests is as follows:

- **ADF test** is used to determine the presence of unit root in the series [32], thus checks the stationarity of the series. If the p-value of the series is less than the confidence level of 0.05, then the null hypothesis can be rejected, indicating the series as stationary. The null and alternative hypothesis for the ADF test is:

  Null hypothesis : The series has a unit root.
  Alternative hypothesis : The series has no unit root.

- **KPSS test** is converse of ADF test where the null and alternative hypothesis as defined in documentation [32] are as follows:

  Null hypothesis : The process is trend stationary.
  Alternative hypothesis : The series has a unit root (series is not stationary).

The p-values for all the variables were examined after applying the tests and are presented in Table 3.3. The results determine stationary for the ADF test and non-stationary for the KPSS test. Hence the series is difference stationary which requires differencing to make it stationary according to the documentation [32].

3.2 Normal Conditions Prediction

As mentioned earlier, the current study has two main goals: predicting meteorological variables of SMA-H buoy and identifying anomalies. This section discusses the normal conditions prediction part. A model is developed to predict the variables of the SMA-H buoy using an ECCC buoy that is 13 kilometers away in the open ocean. Specifically, LSTM from the machine learning side and ARIMA from classical

Variable Notation	ADF test p-value	KPSS test p-value
VCAR	2.6e-30	0.01
VWH$	1.9e-30	0.01
VCMX	2.1e-27	0.01
VTP$	4.4e-25	0.01
WSPD	5.4e-13	0.01
GSPD	2.01e-12	0.01
wave_ht_sig unit(m)	2.6e-30	0.0
wave_ht_max unit(m)	0.0	0.01
wave_period_max unit(s)	0.0	0.01
wind_spd_avg unit(m s-1)	0.0	0.01

Table 3.3: Statistical test results

approaches are chosen as a base for the prediction as they are regarded to produce promising results for time series problems [109, 53]. Owing to the advantages of ensembling approaches [23, 92], a new ensemble method is also proposed and evaluated. The resulting models can be used as temporary replacements during times of buoy downtime. This section provides an in-depth explanation of the approaches' internal workings and their implementations used in the present study.

3.2.1 ARIMA

ARIMA, an acronym for Auto Regressive Integrated Moving Average, was developed by George Box and Gwilym Jenkins [16] in 1970 to describe the variations in time series using a mathematical approach. ARIMA is predicated on the notion that the historical values and their errors should forecast future observations. Past values are commonly referred to as lags. Any stationary time series data can be modeled with ARIMA. Seasonal data is modeled using SARIMA, short for Seasonal-ARIMA, an extension topic of ARIMA that is not of primary importance in this study.

ARIMA is characterised by three terms; order of Auto-Regressive, Differencing, and order of Moving Average. They are denoted by p, d, q respectively in literature and their values are identified by visualising Auto Correlation Function (ACF) and Partial Auto Correlation Function (PACF) plots. Firstly, the autocorrelation function gives the dependency of a current observation on its previous values. Therefore, the ACF plot describes how well a present value is related to its past values considering trends, seasonality, and cyclic. PACF, on the other hand, explains the correlation of

the residuals with the next lag value, thereby removing all the effects explained by earlier lags and gives a pure correlation between series and lags. This section explains the intuition of all three terms, and they are as follows.

- **Auto Regressive (p)** (AR) model is a linear predictive modeling technique that estimates future forecasts based on previous lags. Statistically, AR defines the future forecast as a linear algebraic function of previous terms. The Auto Regressive process of order m, AR(m) for future forecast Y_t which depends on previous m lags at time t is given by:

$$\begin{aligned} Y_t &= \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_m Y_{t-m} + \epsilon_t \\ Y_t &= \alpha + \sum_{j=1}^{m} \beta_j Y_{t-j} + \epsilon_t \end{aligned} \tag{3.2}$$

  where
  $Y_{t-m} = m^{th}$ lag,
  α = intercept term,
  β = Coefficient of lag term,
  ϵ = Error term

  The parameter p is determined using a graphical approach where a Partial Auto-Correlation (PACF) plot is depicted. PACF can be imagined as a function that finds the correlation of the residuals (leaving the contributions of intermediate values) with the next lag value, removing already considered effects from the previous lags and giving pure relation between the series and lags. The number of AR terms can be derived from PACF by examining the lag that crosses the confidence level.

- **Moving Average (q)** (MA) specifies that the forecasting error is a linear function of the lagged errors. MA in ARIMA is different from statistical Moving Average, where the mean is calculated for different subsets of the full dataset. MA of future forecast Y_t which depends on q lagged errors at time t is given by:

$$\begin{aligned} Y_t &= \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + ... + \phi_q \epsilon_{t-q} \\ Y_t &= \alpha + \epsilon_t + \sum_{j=1}^{q} \phi_j \epsilon_{t-j} \end{aligned} \tag{3.3}$$

where
ϕ_t = Coefficient of forecast error at t,
ϵ_t = Error term at t

Error terms are the errors of the auto regressive model of respective lags. For instance, Error terms ϵ_t and ϵ_{t-1} are given by:

$$\begin{aligned} Y_t &= \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_0 Y_0 + \epsilon_t \\ Y_{t-1} &= \beta_2 Y_{t-2} + \beta_3 Y_{t-3} + ... + \beta_0 Y_0 + \epsilon_{t-1} \end{aligned} \tag{3.4}$$

q is determined using the Auto Correlation Plot (ACF), which depicts the correlation between series and lagged values. The number of MA terms can be obtained by looking at the number of lags crossing the significant line in the plot.

- **Differencing (d)** is associated with the integrated part of the model affecting the level of differencing to the series. Differencing is defined as subtracting the current value from the previous seasonal value to make the series stationary. Auto Regressive in ARIMA is a regression model that uses lags as predictors, and it is well known that regression models perform better when the predictors do not correlate (Multicollinearity) [39]. Hence, the series should be made stationary, that is, without trends and seasonality. The order of differencing is determined by the series' ACF plots, which necessitates recurrent differencing for the order number of times until the needed condition is achieved. Section 3.2.2 explains the criterion's aims. The following equations give an understanding about the differencing for different order numbers for a time series Y_t at time t resulting in new time series y_t:

$$\begin{aligned} \text{if } d = 0 &: y_t = Y_t \\ \text{if } d = 1 &: y_t = Y_t - Y_{t-1} \\ \text{if } d = 2 &: y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) \end{aligned} \tag{3.5}$$

We must be careful not to over-difference the series since this would result in an incorrect effect of model parameters. ACF plots of the differenced sequence should

be inspected, and if a lag is rapidly approaching far negative, differencing should be stopped; otherwise, differencing should be continued.

The ACF and PACF plots produced in this study will be seen in Section 3.2.2. Following the determination of the original p, d, q values, the Akaike's Information Criterion (AIC) score should be used as an assessment measure to determine the best combination. The most optimal solution is a combination that minimizes the AIC to the maximum degree possible. AIC includes a correction factor for the number of parameters in a model, thereby penalising complexity. It is used to identify the best qualitative model that uses a goodness-of-fit score, explaining how well the model fits the given dataset without overfitting it.

Overall, after combining AR, MA, d, ARIMA model's equation for forecast Y_t for differenced series y_{dt} is expressed as below:

$$Y_t = \alpha + \sum_{j=1}^{m} \beta_j y_{dt-j} + \epsilon_t + \sum_{j=1}^{q} \phi_j \epsilon_{t-j} \tag{3.6}$$

3.2.2 ARIMA Forecast

ARIMA was developed to learn the patterns of a single variable, which is a significant limitation because the current study focuses on multivariate prediction. Therefore, four ARIMA models were built to predict *wave_ht_sig unit(m)*, *wind_spd_avg unit(m s-1)*, *wave_ht_max unit(m)*, and, *wave_period_max unit(s)*, each with different set of influencing variables (exogenous variables). Orders of AR, MA, and differencing degrees for four ARIMA models are determined by plotting ACF and PACF plots and evaluating AIC scores for all the four target variables. Furthermore, external variables that impact the target can also be included while training and are referred to as exogenous variables. Exogenous variables are selected depending on the correlation of predictors with the target. A detailed explanation is presented in the below sections.

Significant Wave height ARIMA Forecast

PACF and ACF plots for wave_ht_sig unit(m) are depicted in Figure 3.3 and 3.4, respectively. The left depiction shows a wide gap in the center, indicating the missing years as mentioned. Also, it is observed that three lags are crossing the significant

line (blue zone) in the PACF plot. Hence, AR (p) term is considered as three. The ACF plot shows that after the second differencing, the lags hit zero, indicating the differencing should be limited to one, hence $d = 1$. Furthermore, four lags crossed the significant zone in the first differencing ACF plot. Therefore, MA (q) terms should be restricted to four. As a result, MA (q) can be limited to four. Concerning exogenous variables, VCAR, VWH$, VCMX from predictor variables are chosen with the help of the correlation matrix depicted in Figure 3.2. In summary, ARIMA(3,1,4) is considered to be an optimal setting based on the AIC score.

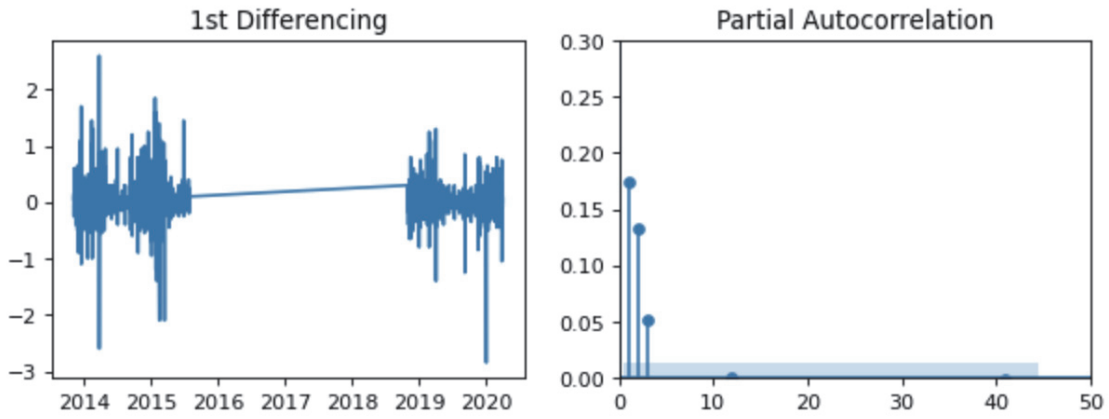

Figure 3.3: Significant Wave height PACF Plot

Maximum Wave Height ARIMA Forecast

PACF and ACF plots for wave_ht_max unit(m) are presented in Figures 3.6 and 3.5, respectively. The ACF plot (3.5) shows that lag values fall to zero after a single differencing to indicate the stationary series. Thus, d should be considered as zero. On the other hand, the PACF plot (3.6) shows two lag values as influential, and hence AR (p) is set to two. Regarding the order of MA (q), the ACF plot shows a correlation with a large number of lags, which makes the problem complicated if many lags are taken. Based on the AIC score, three is chosen as the optimal value for MA (q). In addition, VCAR, VWH$, VCMX are selected as exogenous variables using the correlation matrix (Figure 3.2). In summary, ARIMA(2,0,3) is used as the prediction model for Maximum Wave Height.

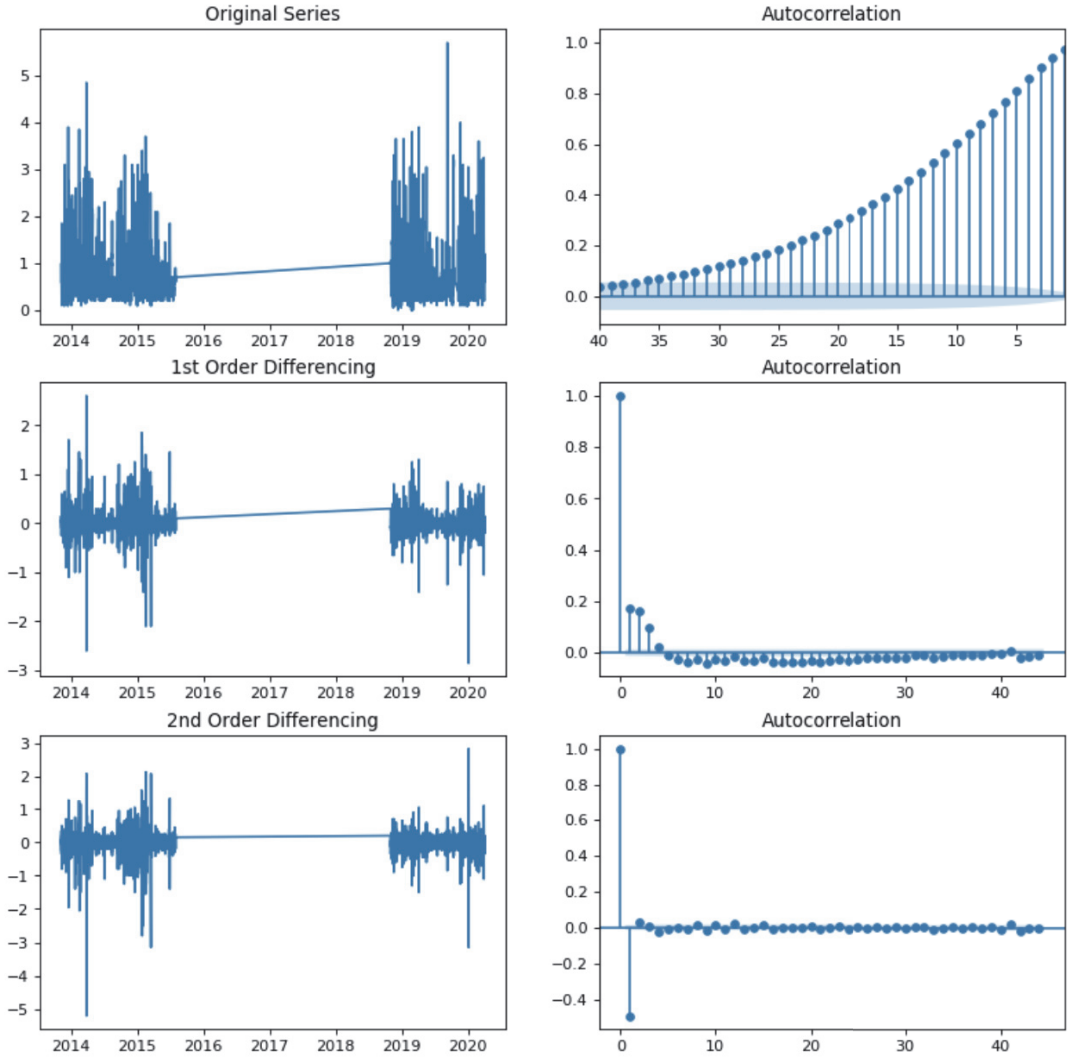

Figure 3.4: Significant Wave height ACF Plot

Maximum Wave Period ARIMA Forecast

PACF and ACF plots for wave_period_max unit(s) are presented in Figures 3.8 and 3.7, respectively. Similar to the case of Maximum Wave Height, Lags tend to fall to zero after first differencing. Hence d should be taken as zero. MA (q) also looks ambiguous from the ACF plot (3.7). So, the AIC score is calculated to set q to four. Regarding the PACF plot (3.8), four lags crossed the significant region allowing us to take AR (p) as four. VCAR, VWH$, VCMX, VTP$ are considered to be exogenous variables using the Figure 3.2. In summary, ARIMA(4,0,4) is the used predictive model for the Maximum Wave Period.

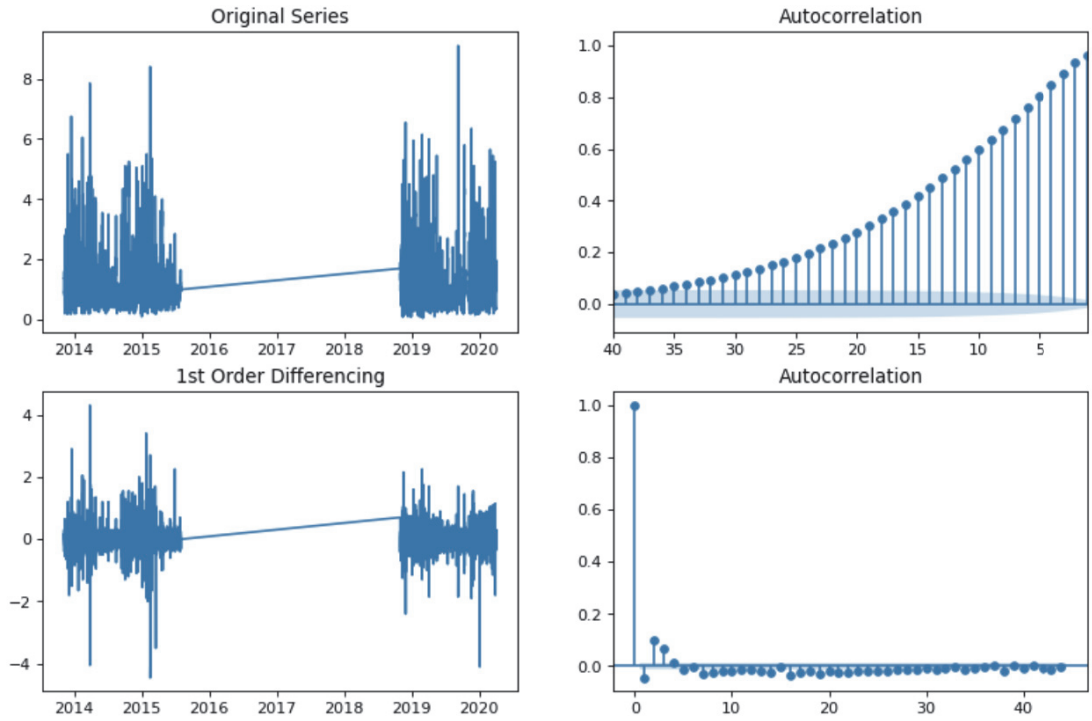

Figure 3.5: Maximum Wave height ACF Plot

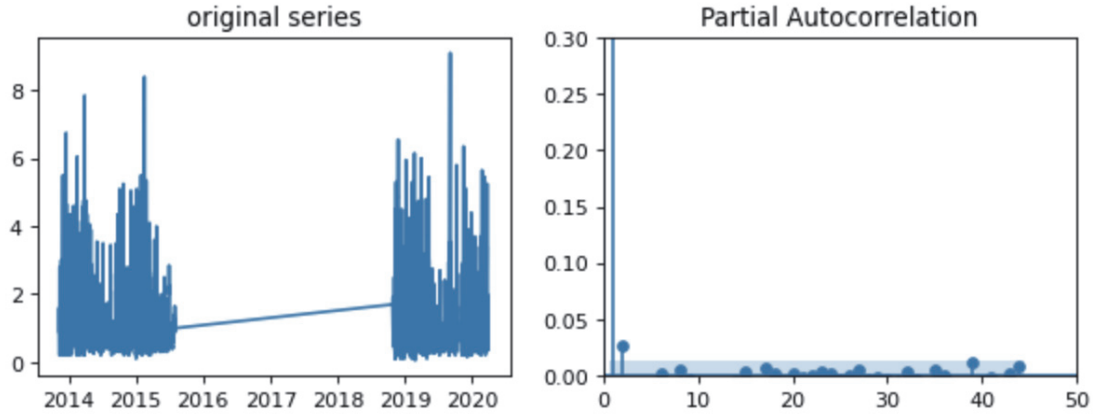

Figure 3.6: Maximum Wave height PACF Plot

Average Wind Speed ARIMA Forecast

ACF and PACF plots for wind_spd_avg unit(m s-1) are presented in Figures 3.9 and 3.10, respectively. Similar to the previous cases, Lags went below zero after the first differencing. Hence, d will be considered as zero. Also, less information is gained from the ACF plot to choose MA (q), which makes us rely on AIC. After AIC

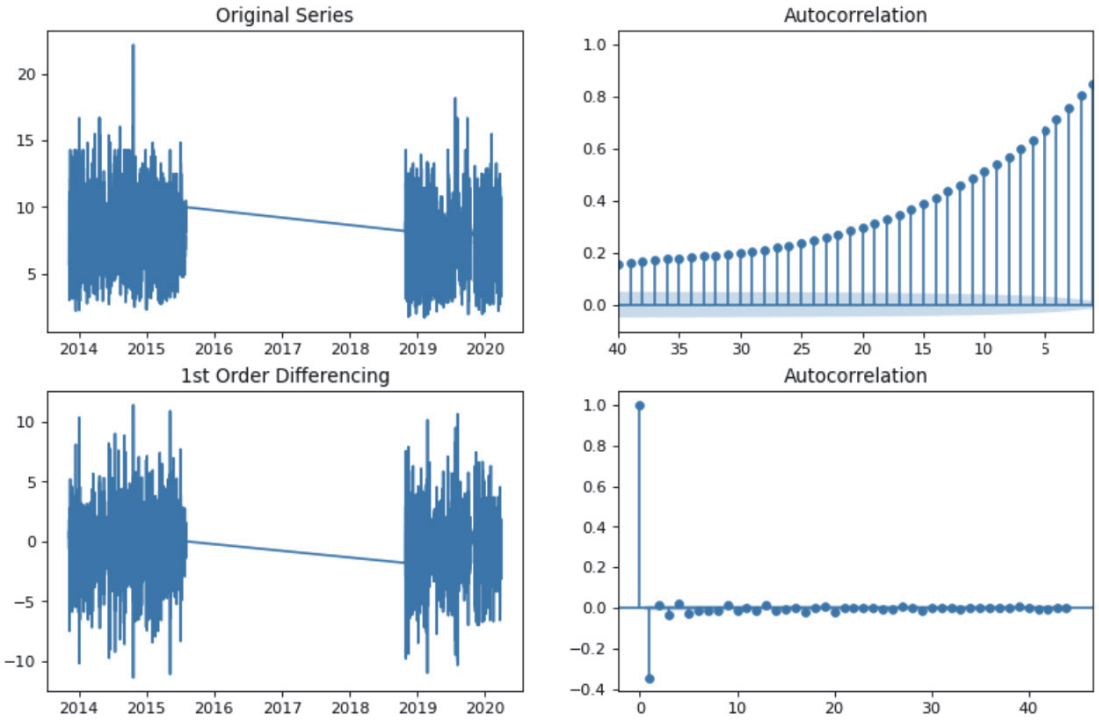

Figure 3.7: Maximum Wave Period ACF Plot

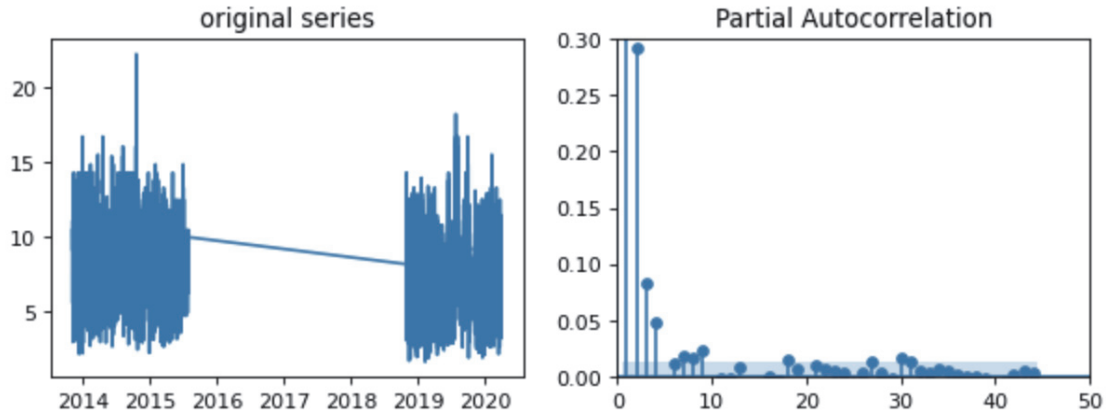

Figure 3.8: Maximum Wave Period PACF Plot

calculation, MA is chosen as one. PACF plot clearly shows a single lag that has full information. So, AR (p) term is set to one. WSPD and GSPD are also selected in addition to VCAR, VWH$, VCMX, VTP$ because the target variable, average wind speed, corresponds to the speed variables, which is the case of WSPD (Wind Speed), GSPD (Gust Speed). Conclusively, ARIMA(1,0,1) is used for predicting Average Wind Speed.

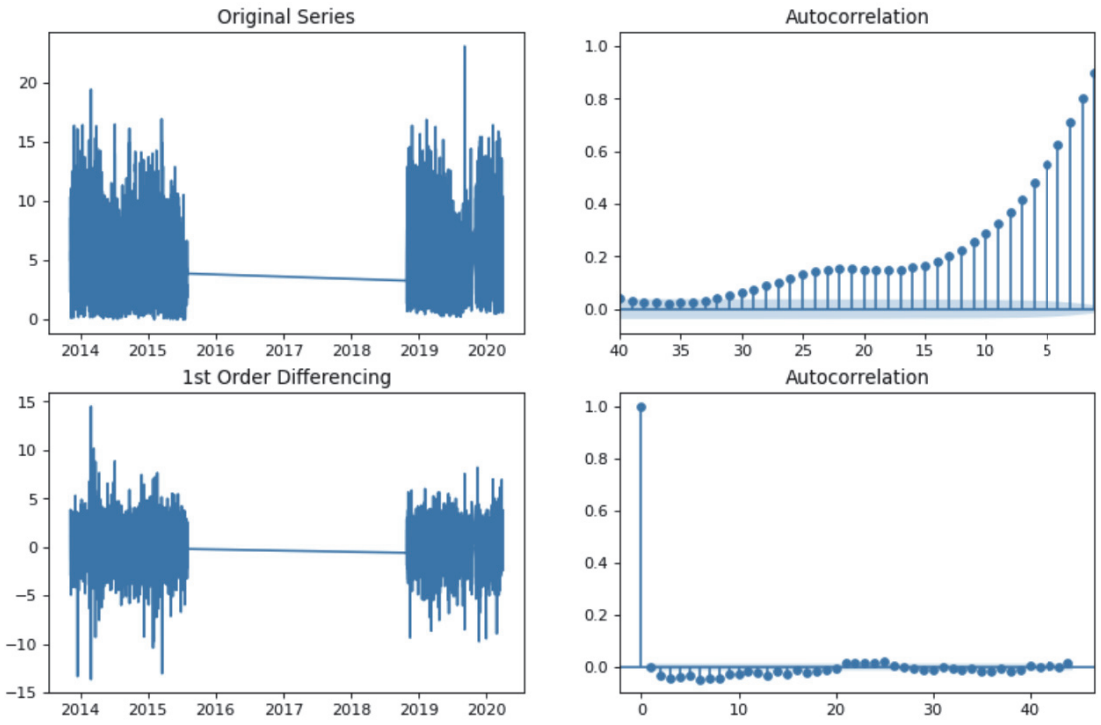

Figure 3.9: Average Wind Speed ACF Plot

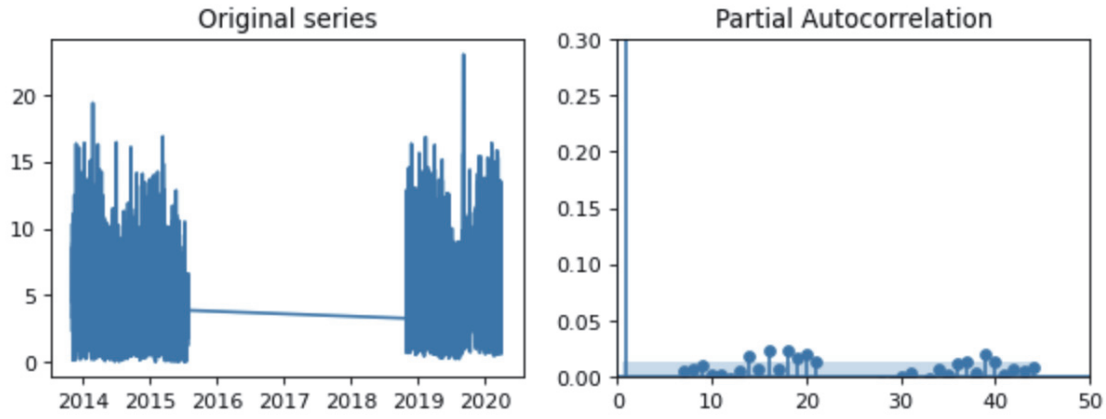

Figure 3.10: Average Wind Speed PACF Plot

3.2.3 Neural Network

Neural Networks, also known as Artificial Neural Networks (ANN), are Machine Learning algorithms modeled after the human brain. Clustering and classification [70] are few among the applications of ANN. As seen in Figure 3.12, a standard ANN has three layers: an input layer to accept input, a hidden layer to process input and an output layer to display the result. Input can be an image represented by a 28x28

(pixels) size matrix, sound information, text, or matrix. An artificial Neural Network is made up of computational units called neurons linked together in various layers. Neurons multiply the input with the appropriate weights, sum it with other values, apply a bias, and pass it to the activation function to remove the expanding range before forwarding it to the next layer. Examples for activation function include sigmoid or tanh. A simple neuron from a book by Guesmi [49] is depicted in Figure 3.11. The output Y_j of neuron for inputs X_j is mathematically expressed as:

$$Y_j = \varphi(\sum_{j=1}^{n} W_{kj} X_j + b_k) \tag{3.7}$$

where

b_k = bias for layer k,

W_{kj} = Weight of j^{th} neuron of k^{th} layer,

$\varphi(.)$ = Activation function

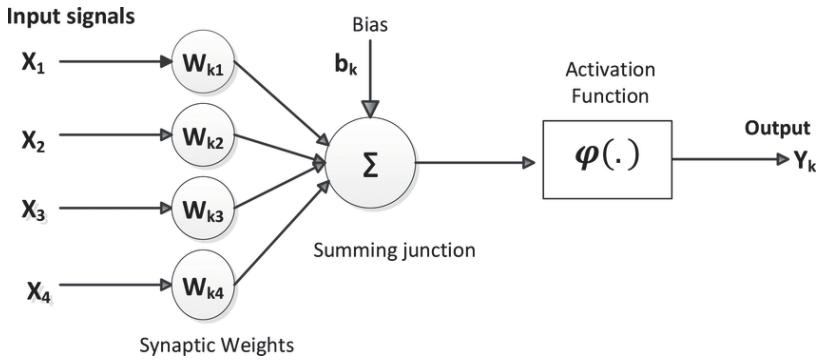

Figure 3.11: Artificial Neuron

The entire network operates iteratively, which means that it processes one record of data at a time and changes the weights accordingly. Since weights are modified from the last layer, the mechanism is known as backpropagation. A cycle is said to be finished when every record has been processed. This cycle is known as an epoch. The model is said to be trained when it has reached the necessary number of epochs. If the number of hidden layers is high, the architecture is referred to as Deep Neural Network. Deep Neural Networks aid in articulating dynamic patterns and the interpretation of sophisticated relationships in large datasets. Figure 3.13 illustrates the architecture of a Deep Neural Network. The following sections include a high-level description of a particular form of ANN architecture known as a Recurrent Neural

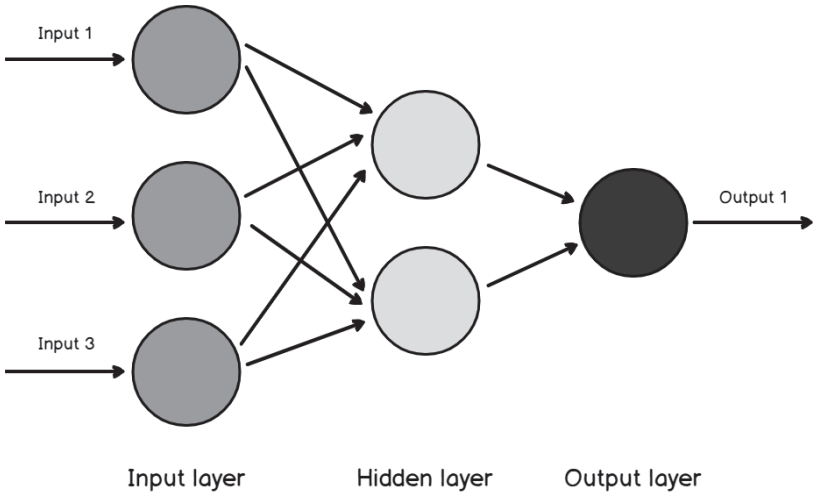

Figure 3.12: Artificial Neural Network

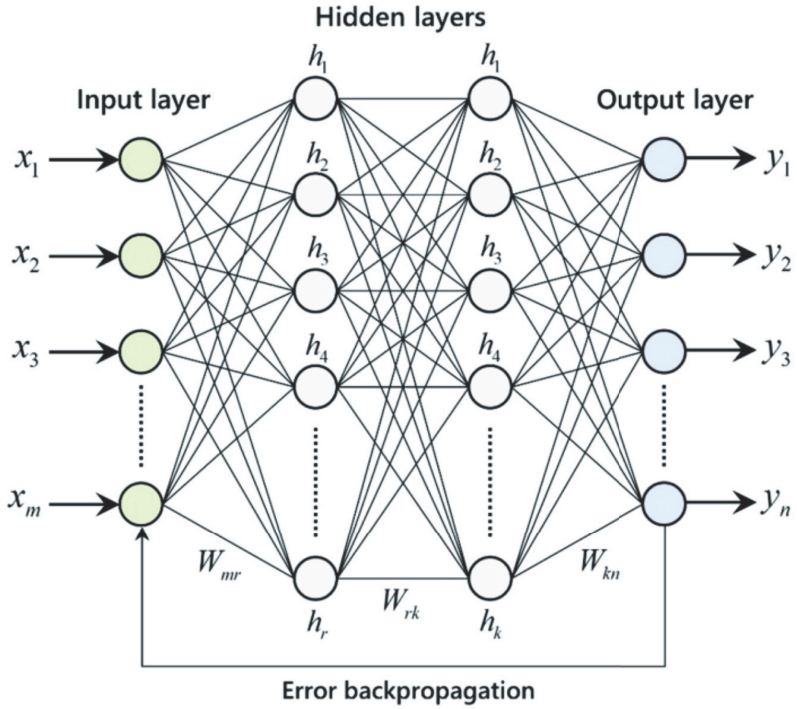

Figure 3.13: Deep Neural Network

Network, which is well-known for handling time series data.

3.2.4 Recurrent Neural Network

Recurrent Neural Network (RNN) is a class of ANN, which is commonly used in speech recognition [46] or natural language processing [111], to capture sequential

characteristics and use patterns to determine the next likely scenario. From a time series perspective, RNN has connections forming a directed graph along a temporal sequence to exhibit dynamic behavior. In simpler terms, conventional neural networks are incapable of storing critical knowledge that can be used to forecast the future outcome of a sequence. Recurrent Neural Networks, on the other hand, have loops that cause knowledge to persist. The left side of the Figure 3.14 depicts a standard rolled RNN, in which neural network A takes input x_t and processes it to generate output h_t for feature t. The loop passes information from one step to the next. An elaborated architecture is presented on the right side of Figure 3.14. A deeper insight into the layer of RNN (Figure 3.14) is shown in Figure 3.15 where a single activation function is used.

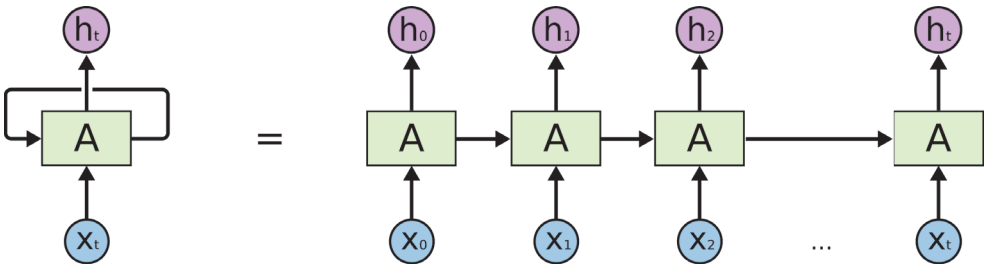

Figure 3.14: Unrolled Recurrent Neural Network, figure taken from [79]

If the outcome is dependent on the short term or does not include knowledge from the distant past, RNN yields successful results in forecasting the potential outcome of a series. However, efficiency seems to be skewed for longer-term dependencies [41], which is a significant necessity in time sequence. Long Short Term Memory (LSTM) was created solely to address this shortcoming in RNN.

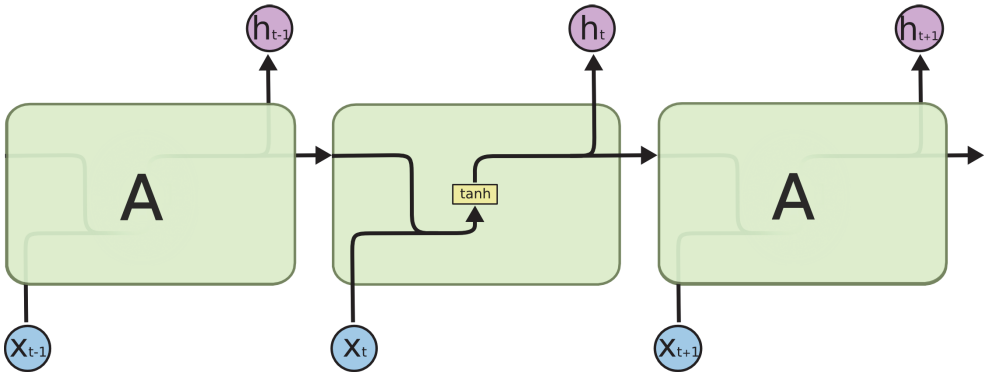

Figure 3.15: Internal working of RNN, Figure taken from [79]

3.2.5 Long Short Term Memory Neural Network

Long Short Term Memory (LSTM) Neural Networks are a class of RNN developed to overcome long-term dependencies. They were introduced by Hochreiter and Schmidhuber in 1997 [54]. LSTMs retain information for a long period to address long-term dependencies. The architecture of LSTM is depicted in Figure 3.16 which shows the increase in processing within the repeating module as compared to RNN (Figure 3.15).

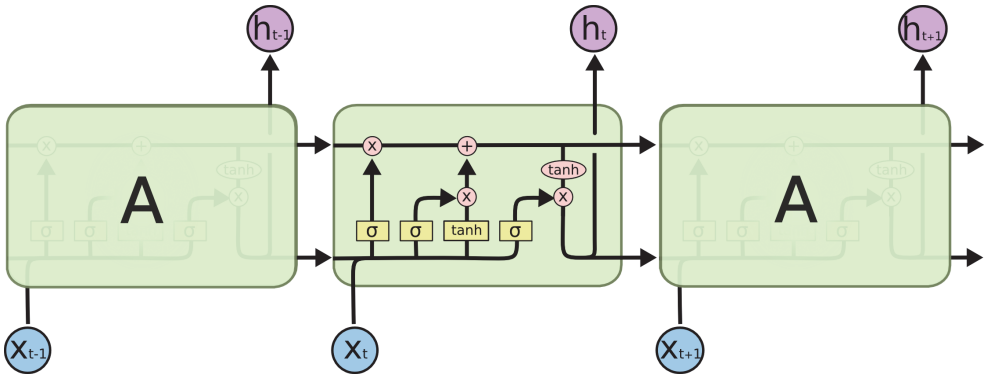

Figure 3.16: Internal working of LSTM, Figure taken from [79]

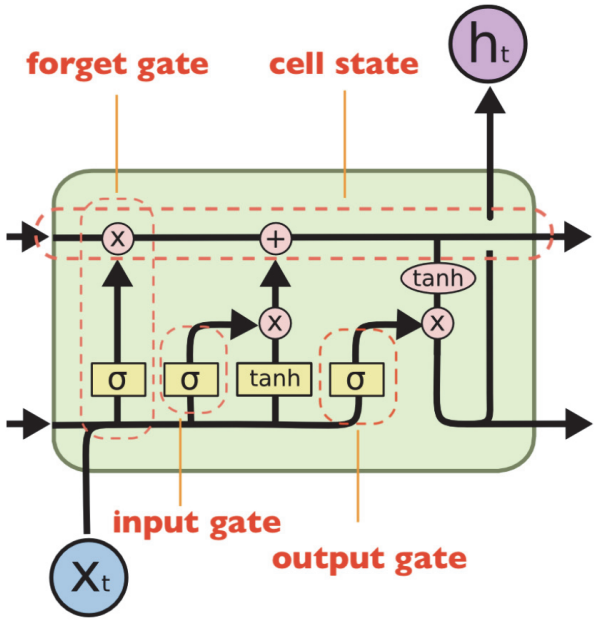

Figure 3.17: LSTM internal description, Figure taken from [79]

LSTM mainly has four components: input gate, output gate, forget gate, and cell state, which are described below:

- **Forget gate** is the initial gate that is encountered in a cell. It decides what data to go on to the cell state using a sigmoid activation function. It takes hidden state h_{t-1} and current input x_t and processes them. The intuition behind using sigmoid function is, the range of sigmoid is 0 to 1. So, when the output f_t from sigmoid point-wise multiplies, the values nearest to one persists on the cell state whereas the other are dropped (forget). The mathematical expression for forget gate looks like:

$$f_t = \Sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.8}$$

where
W_f = weight vector for forget gate,
b_f = bias

- **Input gate** is also used to update the cell state. Hidden state h_{t-1} and current input x_t are passed through sigmoid (to capture important data) function and tanh (to regulate the network) functions. Then sigmoid output s_t and tanh output p_t are multiplied to filter the tanh output, and the final result is updated to the cell state.

$$\begin{aligned} s_t &= \Sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ p_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned} \tag{3.9}$$

where
W_i, W_c = weight vector for Sigmoid and tanh component respectively,
b_i, b_c = bias for sigmoid and tanh

- **Cell state** acts as an information carrier in a layer. It passes across the repeating modules and accumulates necessary information or discarding irrelevant information within the cell. It could be updated using the forget gate and input gate. Mathematical expression for new cell state C_t for previous cell state C_{t-1} is given by:

$$C_t = f_t * C_{t-1} + s_t * p_t \tag{3.10}$$

- **Output gate** decides the next hidden state to be passed to the next cell. Hidden state information about previous inputs plays a crucial role in predictions.

While previous hidden state h_{t-1} and current input x_t are passed to the sigmoid function, new cell state C_t is passed through the tanh function. sigmoid output (o_t) and tanh outputs are multiplied to form a new hidden state h_t that is passed to the next cell. An algebraic expression for h_t is given as:

$$\begin{aligned} o_t &= \Sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ t_t &= o_t * tanh(C_t) \end{aligned} \tag{3.11}$$

where
W_o = weight vector for output gate,
b_o = bias

3.2.6 LSTM Forecast

LSTM mandates data to be in a specific format before forecasting. It accepts data in a 3-dimensional format and thus needs to be transformed. In particular, the input layer for the LSTM should be of the form (nof_sequences, nof_timesteps, nof_features) where nof_sequences is the length of the training data, nof_timesteps correspond to the number of previous records that are useful for predicting the future value and nof_features is the number of feature columns existing in a sequence, 6 in our case. The time step value of one is selected as the optimal value as the performance does not appear to fluctuate substantially with the increasing time steps. Instead, computation increased. Thus, after transformation, each record has a length equal to the number of time steps. Training data of length x, time steps of 1, and six columns are transformed into the shape (x, 1, 6).

The current prediction is called single step multivariate prediction as four variables (wave_ht_sig unit(m) (Significant Wave Height), wind_spd_avg unit(m s-1) (Average Wind Speed), wave_ht_max unit(m), and, wave_period_max unit(s)) are predicted for one future step. A deep LSTM network is used to predict four SMA-H buoy variables because the depth of the network plays a major role in producing efficient accuracy [52]. In particular, stacked LSTM is used to capture complex patterns within the data. Stacked LSTM was introduced by Graves et al. [46] and produced effective results in sequence prediction problems. Figure 3.18 represents the architecture of k-stacked LSTM layers. Settings about the layers will be explained in

Section 4.2.1 and the forecast's outcomes will be addressed in the Section 4.2.2.

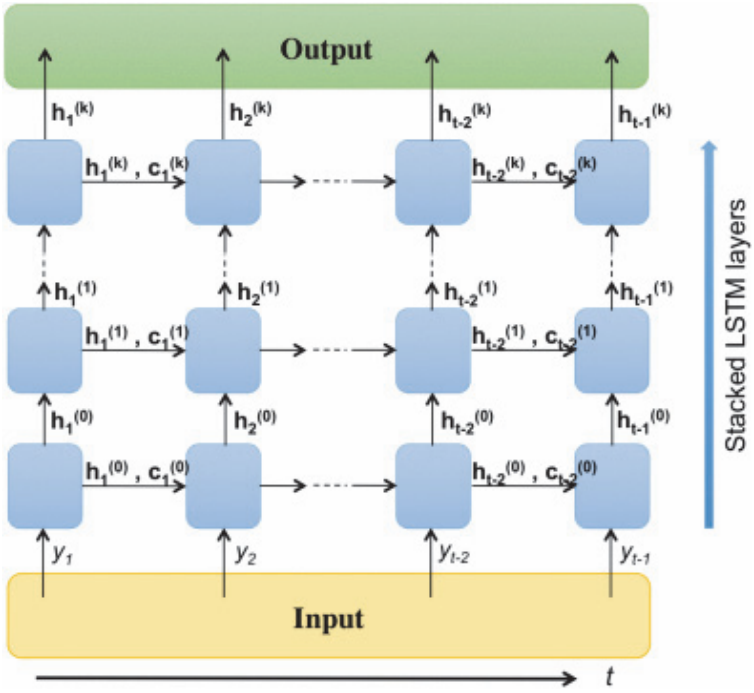

Figure 3.18: Stacked LSTM, Figure taken from [105]

3.2.7 LSTM-ARIMA-LSTM Fusion Model

The main goal of this work is to capture ECCC buoy's information to the greatest extent possible to minimise the loss in predicting SMA-H buoy variables. Meteorological data has a high degree of variability and inconsistency in it. Moreover, both linearity and non-linearity in the data have a tremendous impact on the estimation which cannot be overlooked.

Although non-linear algorithms like LSTM and linear models like ARIMA yielded promising results, we have postulated the hypothesis that their prediction ability can be further improved by fusing non-linear and linear algorithms. Thus, the idea of LSTM-ARIMA originated. Studies [34, 107, 25] rendered this combination successful and useful in respective areas of application. In our study, predictions from ARIMA and LSTM are saved and compared with the actual values, thus forming a dataset. A sample dataset is shown in Figure 3.19, which has stderr from ARIMA to build confidence intervals if required. A new model, LSTM (named Ens-LSTM in this

study), is depicted in the block diagram of whole architecture in Figure 3.20. Ens-LSTM is another LSTM that formulates an equation between the prior LSTM and ARIMA by training on data shown in 3.19. Ens-LSTM is trained on the constructed dataset. Finally, predictions from Ens-LSTM are considered the last level of predictions. Thus the name, LSTM-ARIMA-LSTM. Comparative results of the ensemble model (Ens-LSTM) with individual models will be presented in the Section 4.3 of Chapter 4.

index	lstm	arima	stderr	actual
2018-08-03 10:00:00	0.272159	0.300280	0.179689	0.3
2018-08-03 11:00:00	0.193263	0.298045	0.241764	0.3
2018-08-03 12:00:00	0.271659	0.300513	0.292163	0.3
2018-08-03 13:00:00	0.294796	0.299862	0.316357	0.3
2018-08-03 14:00:00	0.275384	0.299036	0.331854	0.3

Figure 3.19: Combined Data

3.3 Anomaly Detection Model

Anomaly detection is another main goal of our work, with normal condition prediction being the other. Both processes are mandated to happen at the same time during buoy maintenance. In other words, the actual variables of the buoy will be unavailable. The normal conditions model will predict them. Anticipating anomalies based on the predictions of the first model might have shortcomings as the predictions will never be 100% accurate. So, avoiding inaccuracies and additional overheads, a separate model with a prime goal of identifying anomalies using ECCC variables is proposed.

Anomaly Detection in the current work mainly focuses on identifying hurricanes, strong winds, and snowstorms in the surrounding regions of the SMA-H buoy based on the variables from the ECCC buoy. From the correlation matrix in Figure 3.2 (page 31), it can be observed that the SMA-H buoy variables (wave_ht_sig unit(m),

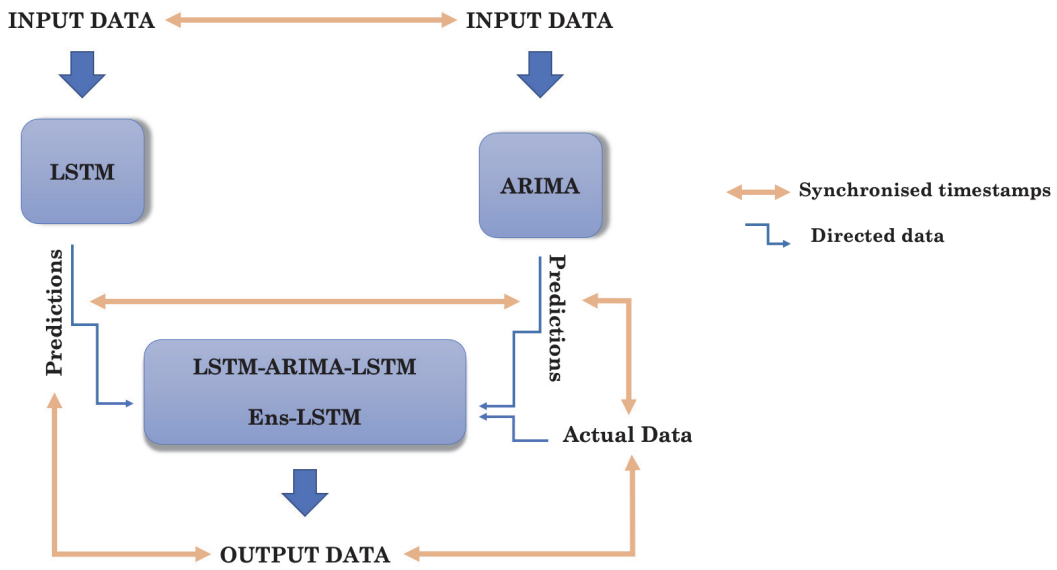

Figure 3.20: Ens-LSTM block diagram

wind_spd_avg unit(m s-1), wave_ht_max unit(m), and, wave_period_max unit(s)) correlate with other variables (ECCC variables) on an average by about 75%, which encourages the use of ECCC in anomaly detection.

Anomalies can be detected using a variety of techniques. Classification, for example, may be used to categorise a data instance as normal or abnormal. However, since the data in this research is unbalanced, classification suffers from distinguishing outliers. We performed a simple classification task on the available data. Our model identified normal instances with 97% accuracy but achieved a mere 27% accuracy in classifying anomalies, which is disappointing. Metrics tend to be marginally high at about 40% when the abnormal data is augmented using Synthetic Minority Oversampling Technique (SMOTE), which is also not convincing. This is due to the involvement of high dynamism in the meteorological data. Therefore, a semi-supervised approach using a reconstruction-based autoencoder model (named recon-enc-dec model in this thesis) was tried.

The proposed semi-supervised solution employs an autoencoder to feed on data with a high proportion of common instances and a low proportion of unusual instances, proportionate to our situation. The model is trained on real-world data and thereby forecasts real-world events. A reconstruction error threshold is assigned based

on the maximum loss in predicting the same training data. So, when the same model attempts to predict an abnormal instance, it fails with a great loss which is a desirable characteristic for our anomaly identification procedure. A detailed explanation about autoencoders is presented in Section 3.3.1.

3.3.1 AutoEncoder

Autoencoders are a type of neural network where the output is the same as the input. Autoencoders essentially has three components: encoder, latent-representation (Code), and decoder, as shown in the Figure 3.21. The encoder compresses the received input into lower-dimensional data accumulating all the necessary information and eliminating noise. Latent representation refers to the learned lower-dimensional encoding of the data. The latent representation contains all of the requisite data material, such as patterns and relationships obtained from the learning of the encoder. The decoder then tries to recreate the data from the latent representation to get as similar to the original data as possible.

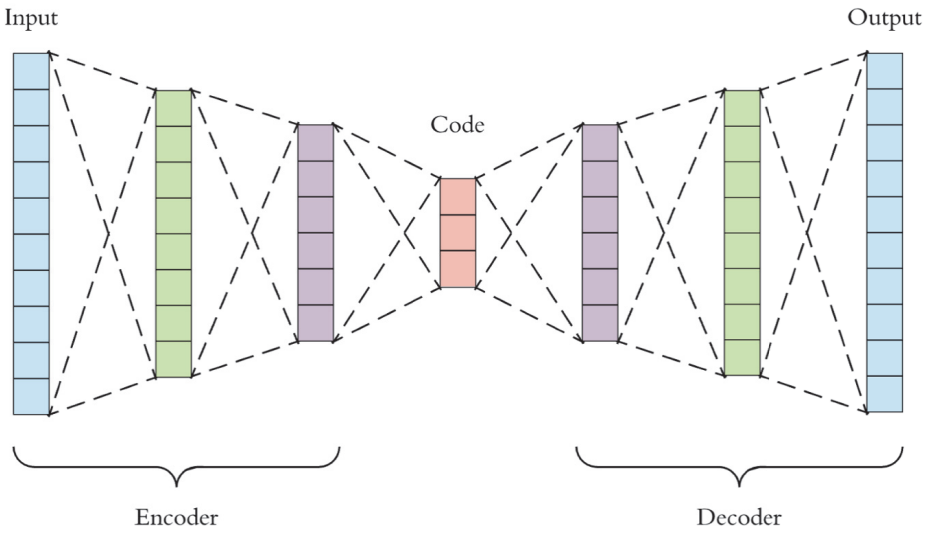

Figure 3.21: Representation of Autoencoder, Figure taken from [13]

Autoencoders are considered unsupervised because they do not need labels. However, since they create their labels (input variables) for training, they may be considered self-supervised. One critical feature of autoencoders is that they are data specific. For example, if an autoencoder is trained to interpret sound data and then

asked to verify animal pictures, it would fail miserably. Furthermore, the decoded data is never exactly the same as the original data. As a result, it is a lossy algorithm. The current research is less concerned with any stated limitations because we are concerned with reconstruction failure based on training results.

In terms of specifications, autoencoders require the following parameters to be defined by the user:

- **code size** or latent representation size is the level of compression required for the input. If the compression size is small, more compression is attempted.
- **network size** is proportional to the information extracted from the data. The deeper the network is, the more features are extracted. If the number of layers is too high, the noise will also be extracted from the data, which should be avoided. This also leads to overfitting. In Figure 3.21, two layers are used in encoder and decoder. The number of nodes within a layer should be selected carefully.
- **layer type** is also an influencing parameter, which controls the performance of the whole network. The layers are populated with machine learning units which should be selected according to the problem. LSTM cell is chosen in this thesis, which is a good handle for time series, is selected as the intermediate layers for Autoencoder.
- **Loss** function is one of the parameters for the model, which the model evaluates while encoding and decoding and tries to minimise the loss to produce the output as close as to the input. Mean Squared Error is commonly used as a loss function for numerical data.

LSTM Autoencoder

Autoencoders are primarily used to capture nonlinear correlations between features to improve detection accuracy. Furthermore, they take a sequence and result in a sequence; thus, this multivariate problem is referred to as a sequence to sequence (seq2seq) prediction. On the other hand, since LSTM was expressly developed to read and produce sequence patterns, it is used as a base algorithm for autoencoder. In the proposed method, variables of ECCC buoy are used as features and fed to the LSTM

autoencoder (named Recon-LSTM-AE in this study). The LSTM encoder interprets and stores all the underlying relations and patterns between the features in a latent representation code. The decoder component of Recon-LSTM-AE decodes the latent representation and tries to recreate the initial series. The maximum loss in reconstructing an instance is calculated. An architecture of a typical LSTM-Autoencoder taking real-valued input sequence $X^{(1)}, X^{(2)}, ..., X^{(n)}$ and generating output sequence $\hat{X}^{(1)}, \hat{X}^{(2)}, ..., \hat{X}^{(n)}$ is depicted in Figure 3.22.

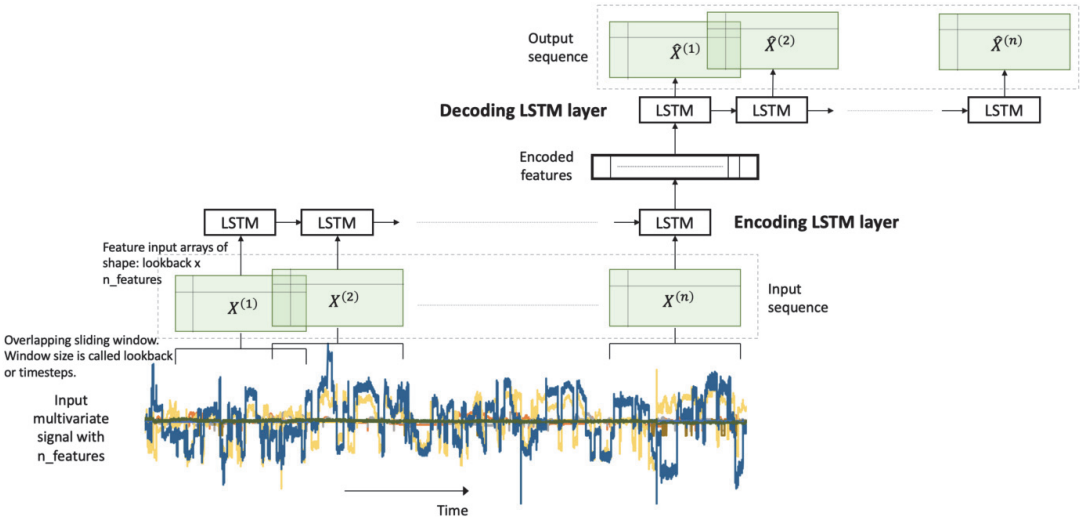

Figure 3.22: LSTM Autoencoder, Figure copied from [91]

3.3.2 Anomaly Detection Methodology

Owing to the advantages of LSTM in sequence predictions, this study proposes a semi-supervised approach to implement an LSTM based autoencoder to capture all the important relations from the ECCC buoy. ECCC buoy, which is 13 km away from the target buoy, showed an 80% correlation in anomaly occurrences which is a notable advantage for predicting anomalies near SMA-H buoy.

The current study utilises a semi-supervised reconstruction-based technique in forecasting anomalies. Initially, Recon-LSTM-AE is trained on filtered data to have a high proportion of normal instances and fewer rare instances. The model is then made to reconstruct the same pattern as the input data. In other words, given an input, Recon-LSTM-AE attempts to predict an output close to the input sequence. The loss in reconstructing the original training data is calculated. Overall maximum loss

while reconstructing training data is considered as a threshold error value. Threshold error values are calculated for every variable. Now that the model is trained, it has to be tested on new unseen data. Our model identifies a particular instance as an anomaly if the reconstruction loss exceeds the threshold error value calculated during training. Figure 3.23 outlines the flow of the anomaly detection algorithm for better understanding.

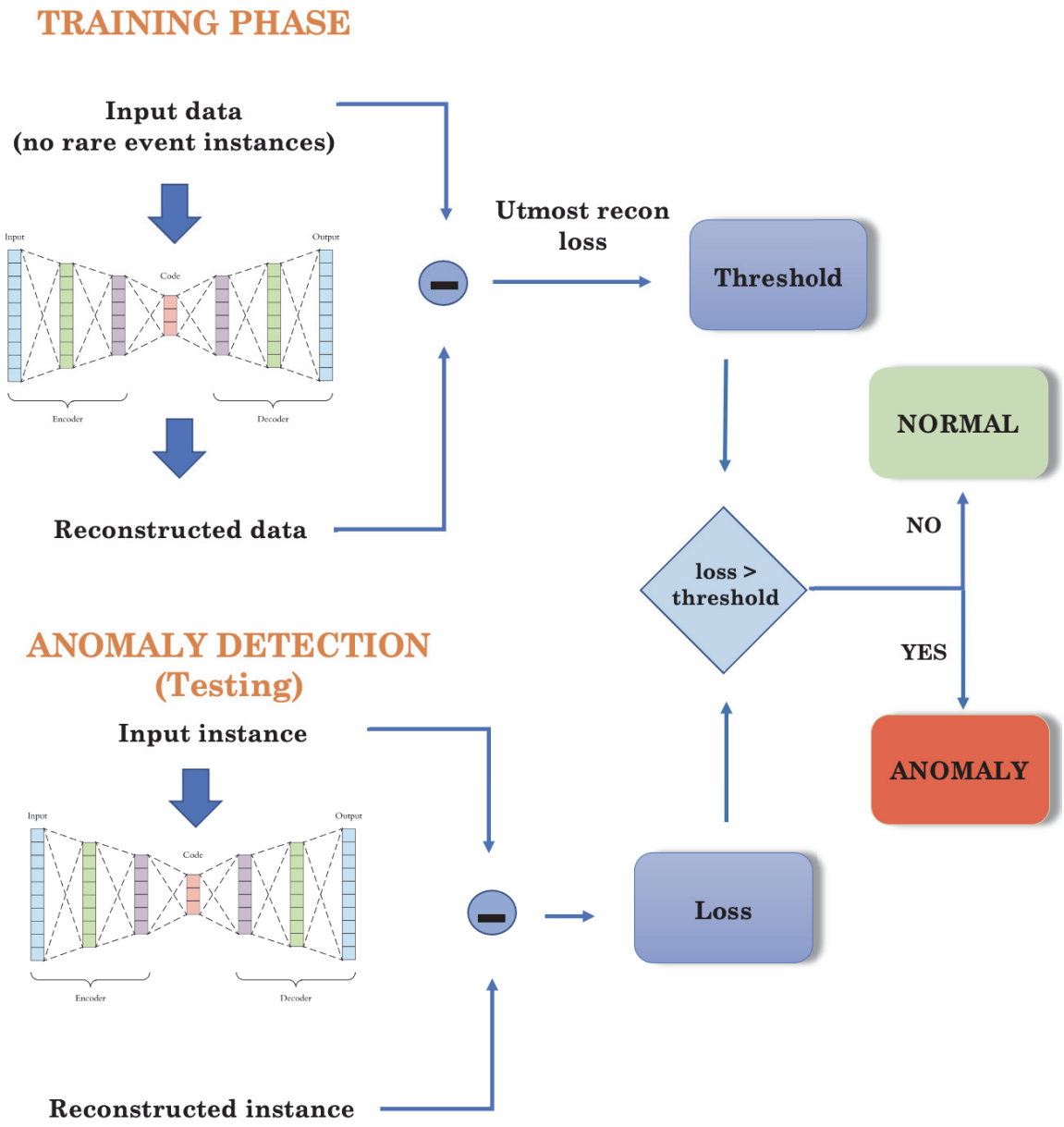

Figure 3.23: Anomaly Detection Flow Chart

3.4 Implementation

A web page with graphical notations of machine learning predictions was designed with the aim of quick interpretability. SMA-H buoy plays a pivotal role in making decisions at the seaport for marine transport. Thus, the buoy's downtime will have severe negative impacts on economic and reputational fronts. Machine learning models were designed as a redundancy model for the original SMA-H buoy to overcome the shortcomings. Hence, any forecasts made or anomalies identified must be interpreted quickly. In summary, a web interface will be used to track all the activities of the meteorological variable predictions, including an alert notification for an anticipated anomaly. Two pages were included in the web page, namely the home page and graphical page. Figures 3.24, 3.25, 3.26, 3.27 show screenshots of the home page and graphical page in the two different events that are normal and abnormal conditions.

Firstly, the web application's home page (Figures 3.24 and 3.26) incorporates a feature to train the model in the event of poor performance. Training action mandates login, which is possible only for registered authorities. Predictions from the most recent forecast are also presented in home page as seen in Figures 3.24 and 3.26. Additionally, a coloured bar is incorporated to determine the prediction quickly.

Secondly, the graphical page constitutes two primary operating components: real-time continuous scheduler management and dynamic graphical simulations.

3.4.1 Real-time Prediction and Graphical Visualisations

As the original SMA-H buoy determines variables periodically, it is important for a temporary machine learning substitute to generate continuous periodic forecasts. Since ECCC calculates variables every hour, the model must forecast variables at the same frequency. A scheduler, namely APScheduler [48] was integrated with the model to make hourly predictions. Interactive buttons for managing scheduler operations can be seen in Figures 3.25 and 3.27.

From the graphical perspective, meteorological predictions for all the four target variables are taken from Ens-LSTM and plotted against the time frame as seen in the figures. The values are plotted every hour automatically without manual intervention after the scheduler is activated. Additionally, a red alert bar is designed to pop in the

case of a probable anomaly event as shown in Figure 3.27. The anomaly alert is based on the anomaly detection model, Recon-LSTM-AE. The error bars in the graph are determined using the variance of the predictions in ARIMA forecasting and require further testing for reliability, which will be a part of future work.

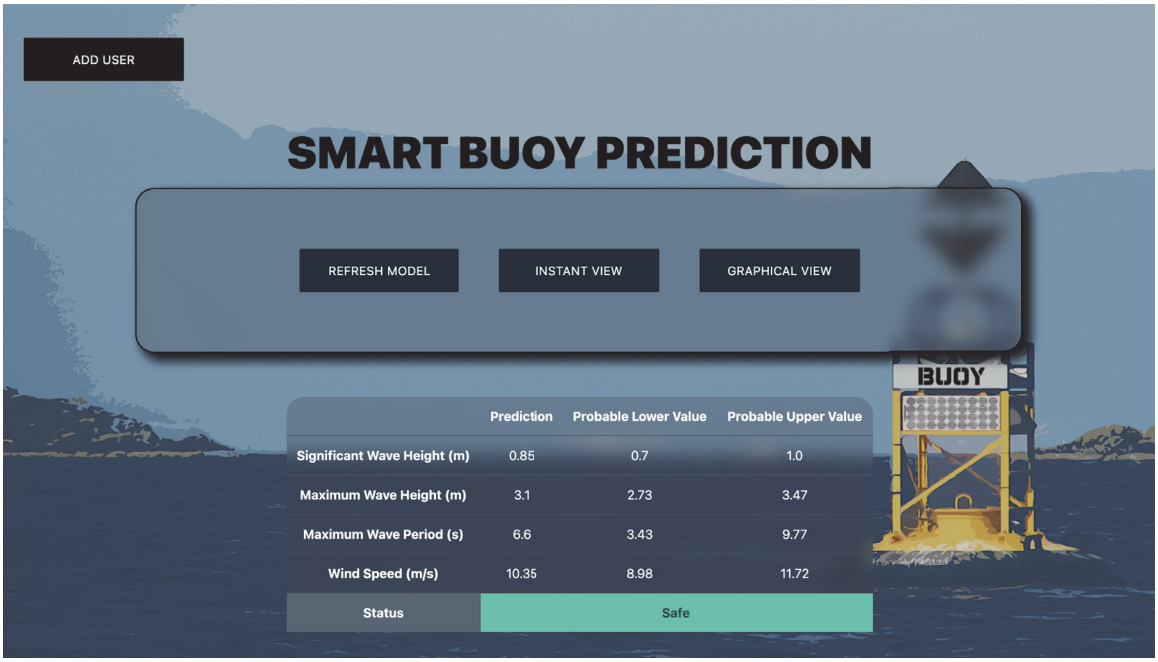

Figure 3.24: Web Page Home Screen in normal event

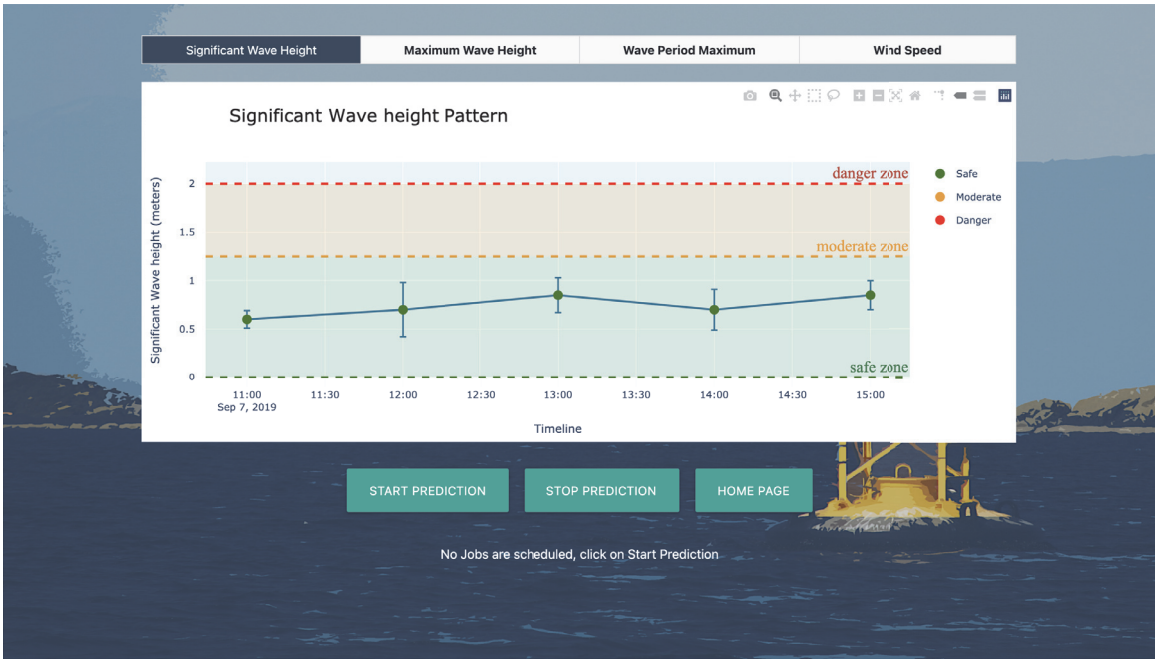

Figure 3.25: Web Page Live Simulation Screen in normal event

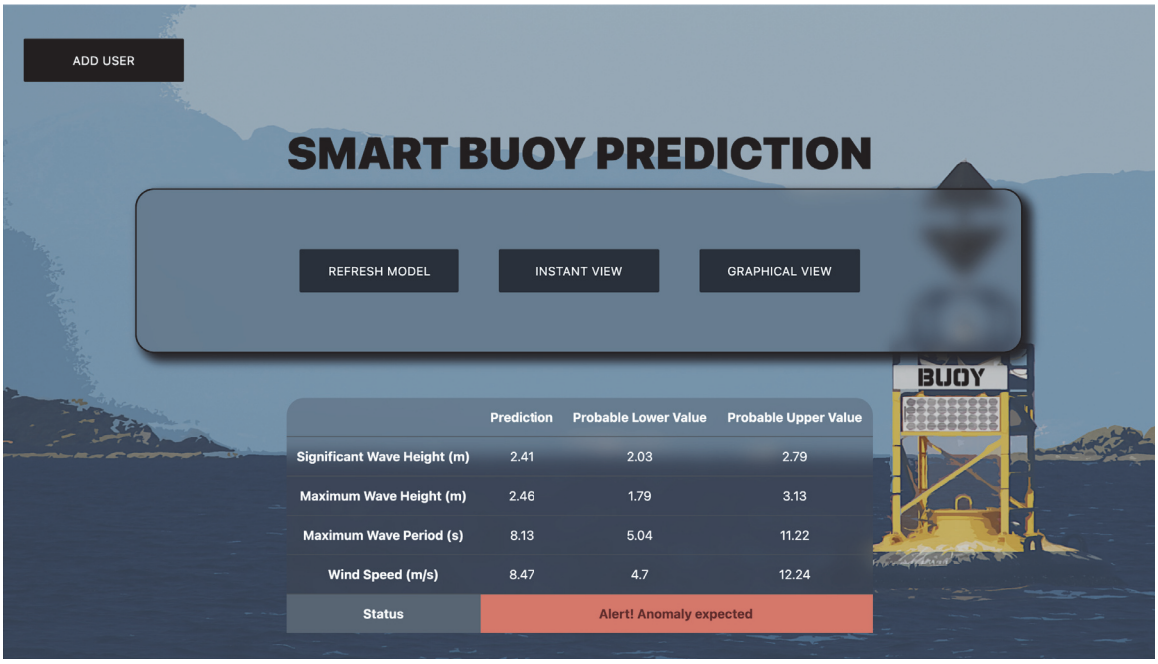

Figure 3.26: Web Page Home Screen in anomaly event

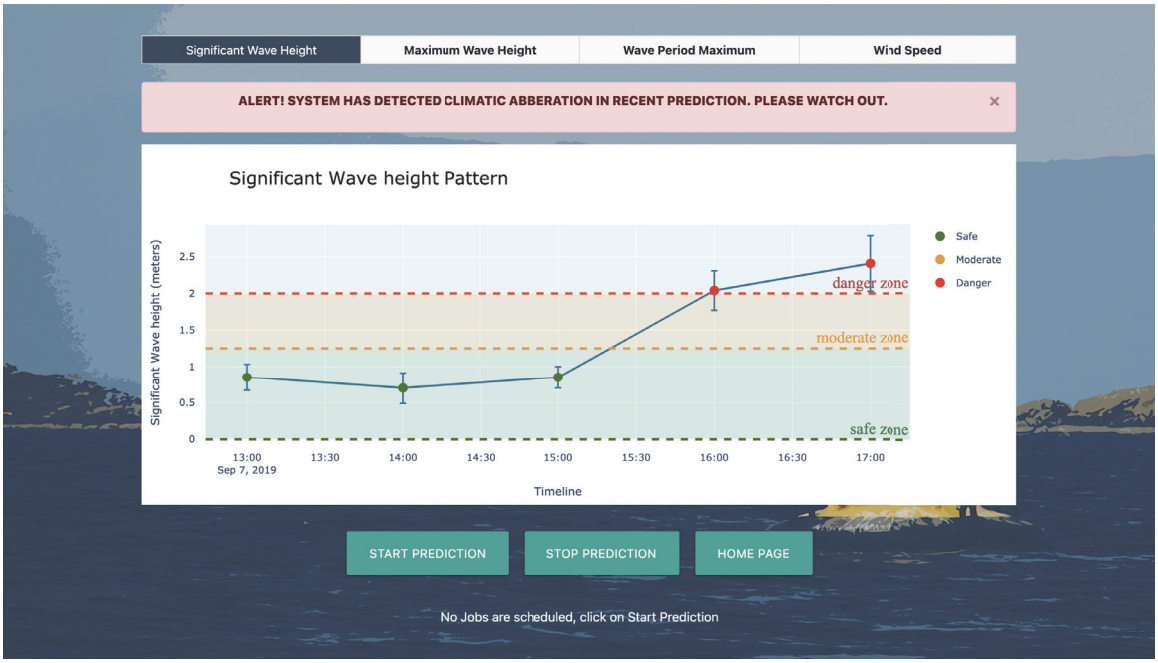

Figure 3.27: Web Page Live Simulation Screen in anomaly event

Chapter 4

Experiments and Results

This chapter demonstrates the architecture and performance of the models explained in the previous chapter. Firstly, Section 4.1 demonstrates the complete flow involved in constructing an anomaly detection model including evaluating precision, recall, and F1-score. Later, Section 4.2 provides a descriptive explanation about the architecture of models developed for anticipating normal conditions alongside a comparison of all model outcomes. Finally, an overview of all the inferences from the results obtained is presented.

The proposed methodology uses Keras 2.3.1 [8], which was built on top of TensorFlow version 2.1.3 to implement neural networks. In Keras, ANN is represented by keras *models*, which are a composition of keras *layers*. For a simpler implementation, a Sequential framework in Keras is used to build layers instead of functional API, which is used to build complex models. Finally, LSTM [9] layers are added to the model to develop a full structure.

DeepSense's [30] HPC cluster is used for project development. Especially, models are trained and tested on the IBM Power 8 GPU machines that each includes 20 CPU cores and 2 Tesla P100 GPUs with 16GB of memory in each GPU [29].

4.1 Anomaly Detection

To summarise, anomalies are identified using a reconstruction-based technique where the autoencoder (Recon-LSTM-AE) reconstructs an instance with a loss greater than the calculated threshold. An explanation of the architecture of the network is presented below.

4.1.1 AutoEncoder Setting

On the structural front, there are three key components involved in the development of Recon-LSTM-AE: Hidden Layers in Encoder and Decoder, Output Layer,

and Latent Representation. Hidden layers are tuned with parameters that include activation function, recurrent activation function, and return sequences, which are discussed further below. Figure 4.1 presents an intuitive architecture of the autoencoder configuration used in the current study.

- **Activation function** regulates a neuron's state by making it active or inactive. It adds non-linearity to a standard neural network. The activation feature is used in an LSTM layer for cell state and hidden state, as seen in Figure 3.17. Current study uses hyperbolic tangent function, tanh, as an activation function which is a default implementation for LSTM cell. The function ranges between -1 to 1, which necessitated rescaling of entire data to range between -1 and 1 using scikit-learn's [82] (0.23.2) MinMaxScaler [19].
- **Recurrent Activation function** is similar to the activation function but it is applied to input, forget and output gate in Figure 3.17. Hard Sigmoid function, which is a default implementation, is used in LSTM layers in the current study and is denoted by σ. Hard Sigmoid ranges from 0 to 1 and is mathematically defined in the documentation [101] as:

$$hard\_sigmoid = f(x) = \begin{cases} 0, & \text{if } x < -2.5 \\ 1, & \text{if } x > 2.5 \\ 0.2 * x + 0.5, & \text{if } -2.5 \leq x \leq 2.5 \end{cases} \tag{4.1}$$

- **Return Sequences** is a parameter in the LSTM layer which accepts a boolean value. When this parameter is set to true, LSTM cells emit hidden state output for each input time step. Figure 3.16 shows each LSTM cell emitting a cell state and a hidden state for every input time step. Since we used a single time step, each layer produces a single array of hidden states. In our analysis, the return_sequences parameter is set to true for all hidden layers except the layer near the latent representation to output an encoded feature vector.
- **Hidden layers** are the intermediate structures that connect the input layer to latent representation and then to the output layer. The layers between input and latent representation are collectively called an encoder, and layers between latent representation and output are known as a decoder. However, the layers

in the decoder are stacked in reverse order of the encoder. Initially, four hidden layers were used in the encoder and decoder, with 512, 256, 128, and 64 LSTM units (nodes) in each. The model seemed to predict irregularities well in training data but slightly failed to identify them in testing data, indicating overfitting. Furthermore, the described setting required an 18-minute computation cycle, which is reduced to 12 minutes with a different design and comparative effectiveness. The second design used two layers of 225 and 175 LSTM units in the encoder and decoder. The performance slightly improved as compared to the prior setting. Heaton, in his study [51] mentioned a rule of assigning hidden layers $2/3^{rd}$ the size of input plus output layer. The current study opted for the same equation to downsize the hidden layers in the encoder and decoder. Finally, the number of nodes in the encoder and decoder were reduced to 84 and 56 nodes yielding significant which are presented in Section 4.1.4 .

- **Repeat Vector** from Keras layers [6] serves as a link between the encoder and decoder. It iteratively replicates the encoded function vector over a set number of time steps. In our case, since the number of time steps is one, it no longer replicates and instead sends the encoded function vector to the decoder.
- **TimeDistributed layer** from Keras layers [7] is a wrapper that allows the application of a layer (Dense in our case) to every temporal slice. The TimeDistributed layer creates a vector of length equal to the features outputted from the previous layer using the Dense layer [73] with the equivalent number of features in the input. Now matrix multiplication with this final layer and the last hidden layer is performed to produce the output with the same shape as input data. For instance, the last hidden layer in our case produces an output of shape 1x56, and TimeDistributed creates a vector of shape 56x6 (number of features in ECCC). They both are multiplied to produce output 1x6 which is the shape of input.

4.1.2 AutoEncoder Evaluation

Evaluation is the major step in assessing the performance of a model. In Anomaly Detection, the developed model, Recon-LSTM-AE, is trained on unlabeled data that

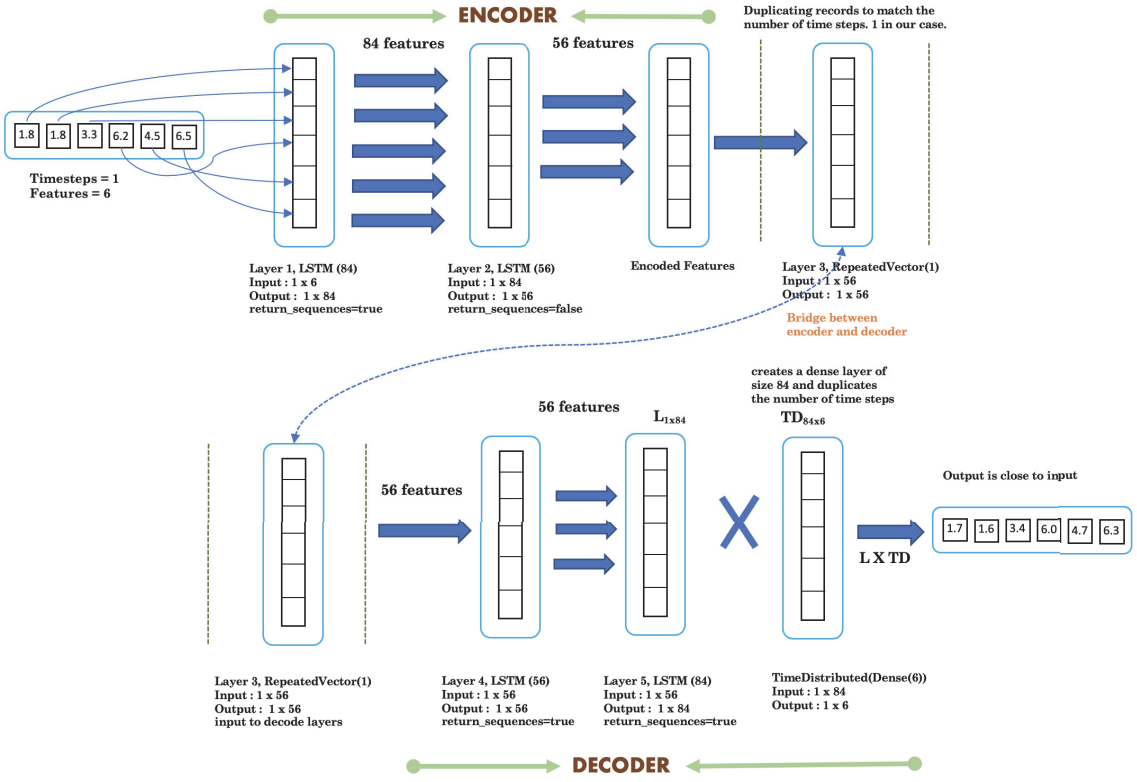

Figure 4.1: Structure of Recon-LSTM-AE used in this thesis

is free from rare event instances, an unsupervised learning type. After training, the maximum loss, reconstruction threshold, is calculated for respective variables of ECCC.

The performance of the trained model is assessed on testing data which is completely new that is not part of training data. Initially, the dataset is divided into training and testing parts according to the estimation procedure explained in Section 4.1.3. Testing data, a composition of ECCC and SMA buoy variables, is labeled anomaly or normal based on the threshold limits of the SMA-H buoy variables. Threshold values for each SMA-H variable are gathered from domain authorities, and an event is referred to as an exception if more than one variable exceeds the threshold at the same time [89]. Thus the problem is called semi-supervised due to the apparent marking of data for evaluation. Later, the training ECCC instances are reconstructed using Recon-LSTM-AE, and the maximum loss is noted, which is termed as reconstruction threshold. Testing data is reconstructed, and the instances whose loss exceeds the reconstruction threshold are termed anomalies. The problem now is to check whether

the predicted anomalies match the actual anomalies, making the evaluation similar to the classification assessment. Hence, Precision, Recall, F1-score can be opted to evaluate the model. Precision and Recall for the anomaly class examine the model from an anomaly standpoint, as opposed to Accuracy, which is biased for imbalanced data. For better estimation, the F1-score, a harmonic mean of Precision and Recall, is used for better evaluation. They are algebraically defined in page 20.

Based on the metric equations in Section 2.1.3, the terminology in terms of anomaly anticipation is given by:
TP : True Positive : Number of anomalies correctly identified.
TN : True Negative : Number of non-anomalies correctly identified as non-anomalies.
FP : False Positive : Number of normal instances identified as anomalies.
FN : False Negative : Number of anomalies predicted as normal

4.1.3 Anomaly Detector Estimation

A machine learning model should be properly evaluated to check the reliability of the produced results by comparing with different estimates. Precision (Equation 2.14), Recall (Equation 2.15), F1-score (Equation 2.16) are calculated and a significant estimate is computed using an estimation approach, repeated holdout, as discussed in Section 2.1.4. To summarise, repeated holdout is an estimation technique used to determine if metrics produced are attributable to random noise or are genuinely significant. Figure 4.2 depicts the repeated holdout procedural flow. A window of consideration is arbitrarily picked from the real data, and a point that follows the size constraints of train and test data is chosen. The data on the left is used for learning, while the data on the right is used for evaluation. In terms of data ordering, the data is never shuffled to emphasize the temporal dependency in the data and force the algorithm to forecast only future cases based on previous cases to get reliable predictive performance [104], as it does in real-time. This process is repeated a definite number of times as shown in Figure 4.2, and an average of all the produced results is concluded as a final metric.

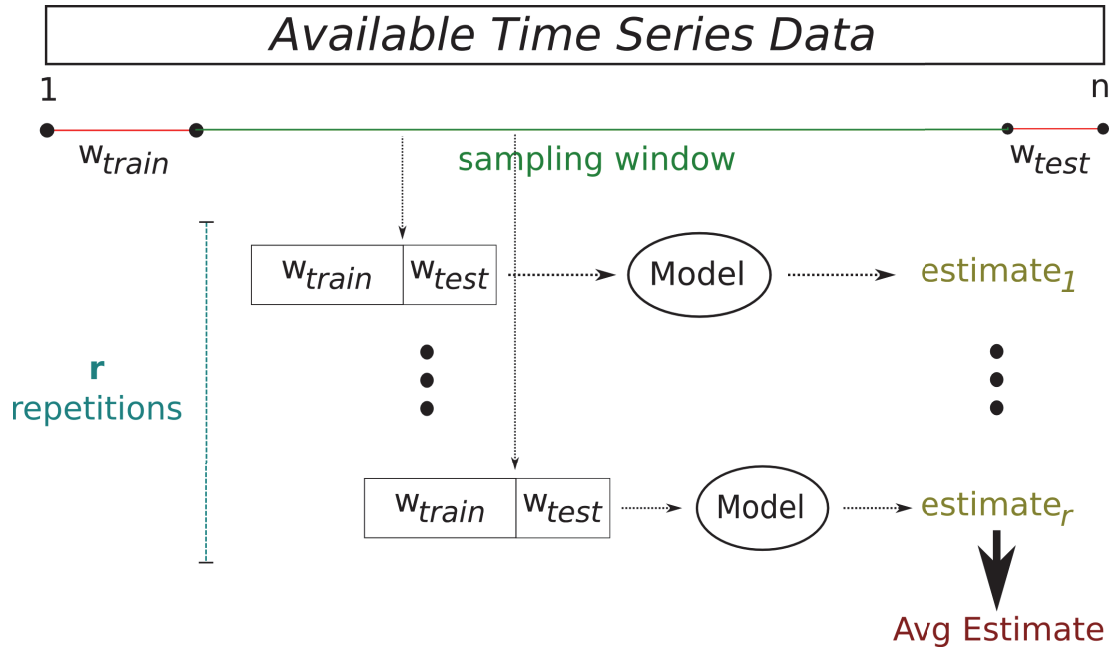

Figure 4.2: Estimation procedure in detail, figure taken from [99]

4.1.4 Anomaly Detector Results

Following the development of Recon-LSTM-AE, the model is trained on the input data that is divided to have one time step at a time. Higher number of time steps like 3, 5 were experimented and they produced similar results with greater computation. Hence one time step is used for training. The learned model's efficiency is evaluated using metrics such as precision, recall, and f1-score. To ensure that the results obtained are robust, the Repeated-Holdout estimation process, as explained in Section 4.1.3, is used, which repeatedly trains, tests, and estimates the metrics on different sub-samples of data, as illustrated in Figure 4.2.

The results obtained in various iterations are shown in the Table 4.1. From the table, it is evident that the performance of Recon-LSTM-AE rendered outstanding results in every iteration with an overall average precision, recall, and f1-score of 98.04%, 100%, 98.9%, respectively. Also, a typical confusion matrix of iteration 8 for anomaly detection is depicted in Figure 4.3 for better understanding. Furthermore, anomalies are portrayed on a time series graph for variables of SMA-H in Figures 4.4, 4.5, 4.7, 4.6. The high accuracy in identifying anomalies can be seen in four separate time series anomaly graphs, where the green markers denote real anomalies and the

iter's	train/test count	precision	recall	f1-score	normal detections	anomaly detections
1	38645/9660	95.4	100	97.6	9376/9389	271/271
2	33984/8687	99.5	100	99.7	8487/8488	199/199
3	23787/6081	99.2	100	99.5	5956/5957	124/124
4	13903/3474	99.1	100	99.5	3351/3352	122/122
5	6801/1736	94.3	100	97.0	1683/1686	50/50
6	30582/7818	98.9	100	99.4	7628/7630	188/188
7	31228/7806	99.4	100	99.7	7616/7617	189/189
8	23421/5854	98.5	100	99.2	5720/5722	132/132
Average Estimates		98.04	100	98.9		

Table 4.1: Anomaly Detection Results obtained for every iteration

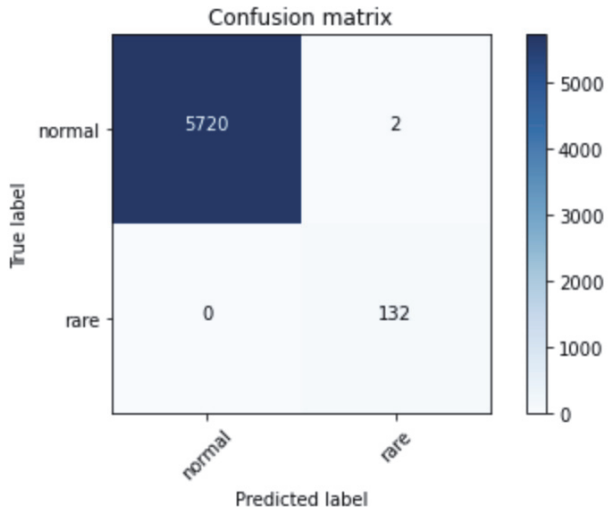

Figure 4.3: Typical confusion matrix for iteration 8 in Table 4.1

red markers reflect anticipated anomalies. Another notable aspect is that there exist points above the threshold line in the wave period anomaly graph (Figure 4.7), which are not considered as an actual anomaly during analysing because of the collective variable threshold assumption. In other words, during evaluation, an instance is considered to be a real exception only if more than one of the four target variables exceeds the respective threshold limits [89]. The same interpretation applies to the anomalous points found below the threshold line.

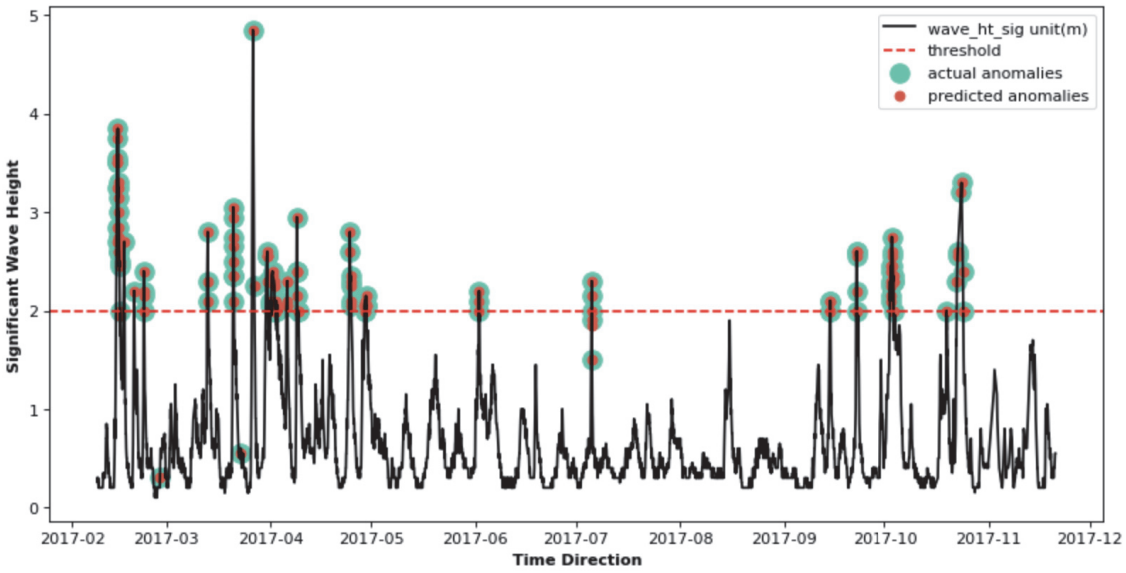

Figure 4.4: Anomaly detection in wave_ht_sig unit(m) variable

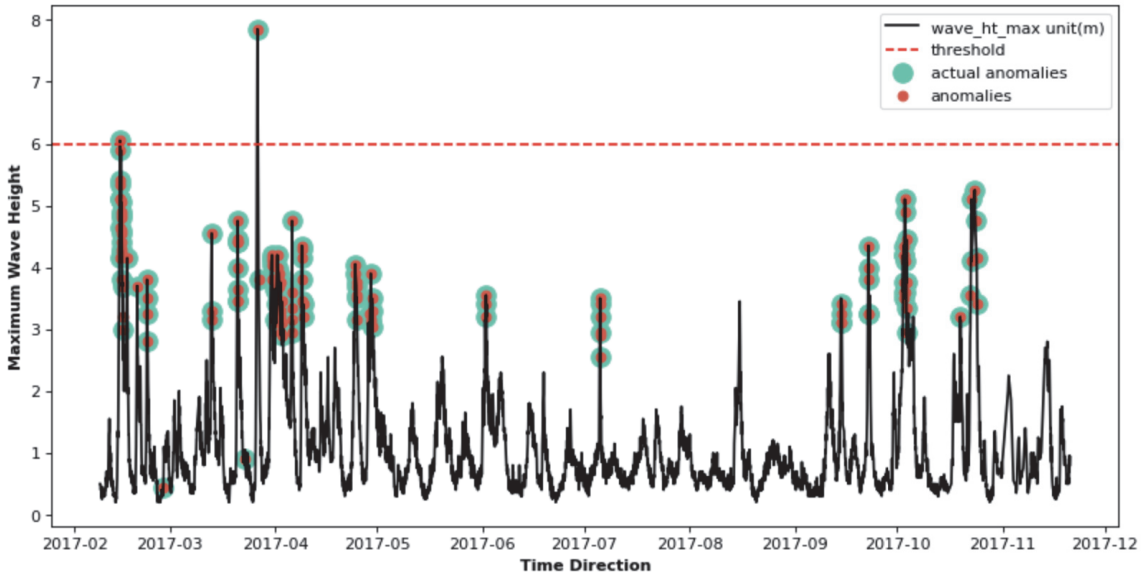

Figure 4.5: Anomaly detection in wave_ht_max unit(m) unit(m) variable

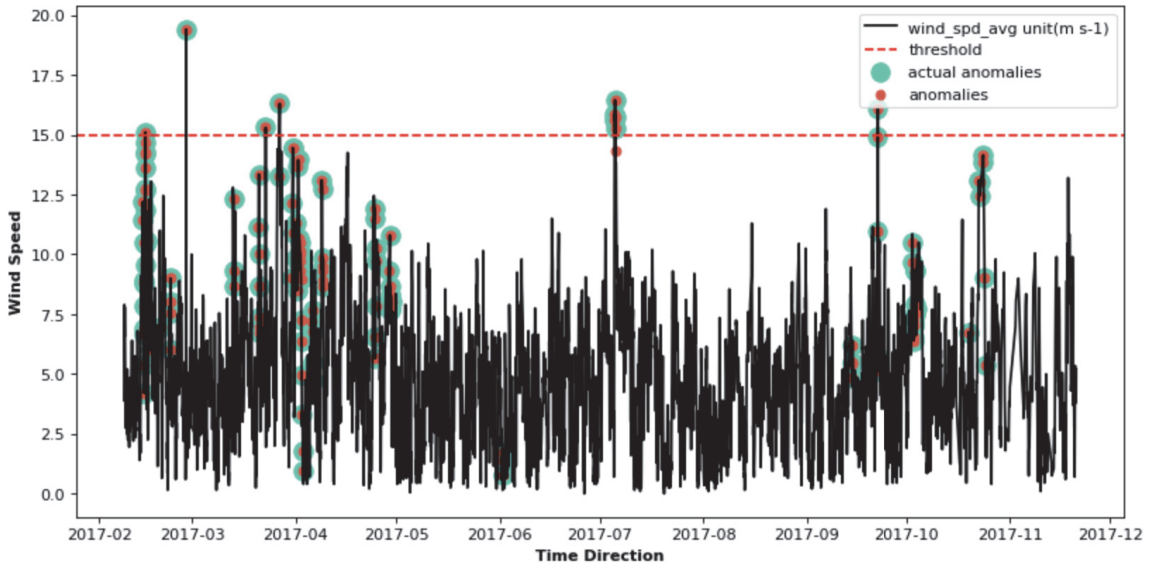

Figure 4.6: Anomaly detection in wind_spd_avg unit(m s-1) unit(m) variable

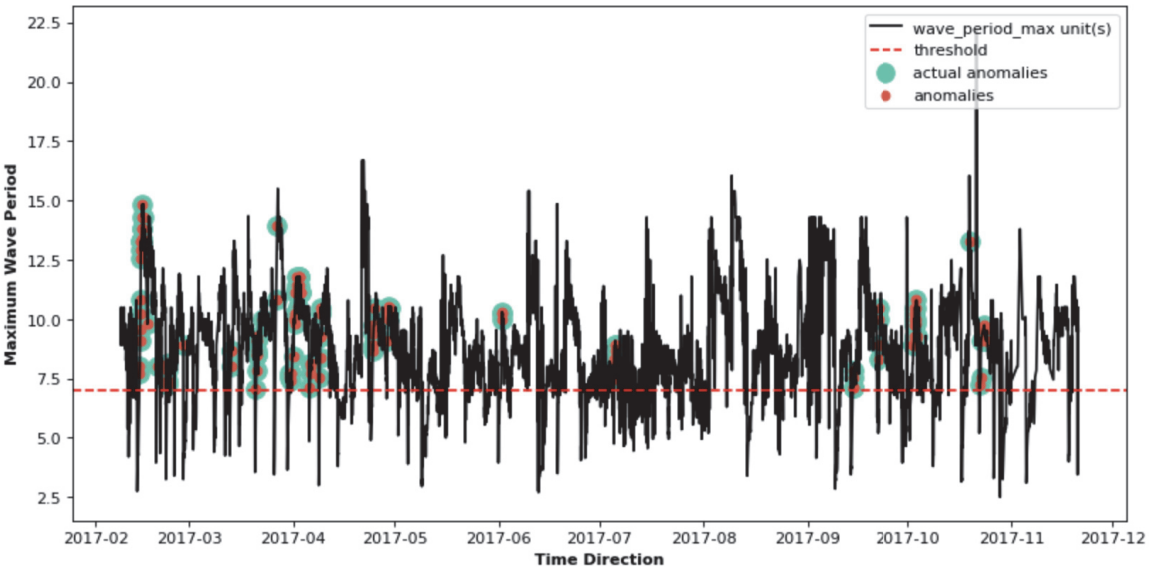

Figure 4.7: Anomaly detection in wave_period_max unit(s) unit(m) variable

4.2 Normal Conditions Prediction

In the current study, normal events are initially estimated using a non-linear model, LSTM, and linear model, ARIMA. Although they rendered reliable results, the performance of the estimations is expected to be improved further by combining both linear and non-linear models. Predictions from LSTM and ARIMA are integrated to establish a non-linear function of both the algorithms using another LSTM. While an averaging model that takes the average of both models' forecasts can be used, a new model is added to experiment with the relationships between the two models to achieve higher efficiency. Results of individual and combined models are presented in the below sections.

4.2.1 LSTM Setting

LSTM is constructed using the Keras Sequential framework, which stacks Keras layers in the order of submission. LSTM network involves three major constituents: input, output, and hidden layer. The depth of the network typically depends on the number of layers in the architecture. Hidden layers, in turn, accept multitudes of parameters that characterise the working of a network. A comprehensive description of the internal working of the architecture is presented below.

- **Input Layer** for LSTM architecture accepts 3-dimensional data: the number of records, the number of time steps (previous records), and the number of features in each dimension. So, the existing 2-dimensional data in this work is transformed to a required shape by considering a single time step at every learning iteration. The input layer then gives the entry point for the data into the network and passes it to the hidden layers.
- **Hidden layers** are an intermittent collection of nodes associated with weighted connections. In terms of nodes, LSTM cells are used to build an LSTM layer. Two layers each of 178 LSTM nodes are used as hidden layers (thus called as stacked LSTM). While additional layers or nodes produced the same results with excess computational time, lesser layers resulted in decreased efficacy due to underfitting. Different configurations of LSTM layer parameters like activation function, recurrent activation function were examined. The setting that is the

same as the case of Recon-LSTM-AE as explained in Section 4.1.1 is observed to have a limited loss and thus opted in this procedure. In addition, the return sequences parameter is set to true in all layers except for the one linked to output.

- **Output Layer** is the final layer that generates results. Since the previously hidden layer's return sequences parameter is false, the output layer receives a single array of the processed vector, the size of which is 178. The vector is then processed by the Dense layer [73] which has an output shape equivalent to the number of target variables, thus yielding the final output. Figure 4.8 depicts a detailed overview of all the layers for a clearer understanding of the overall architecture.

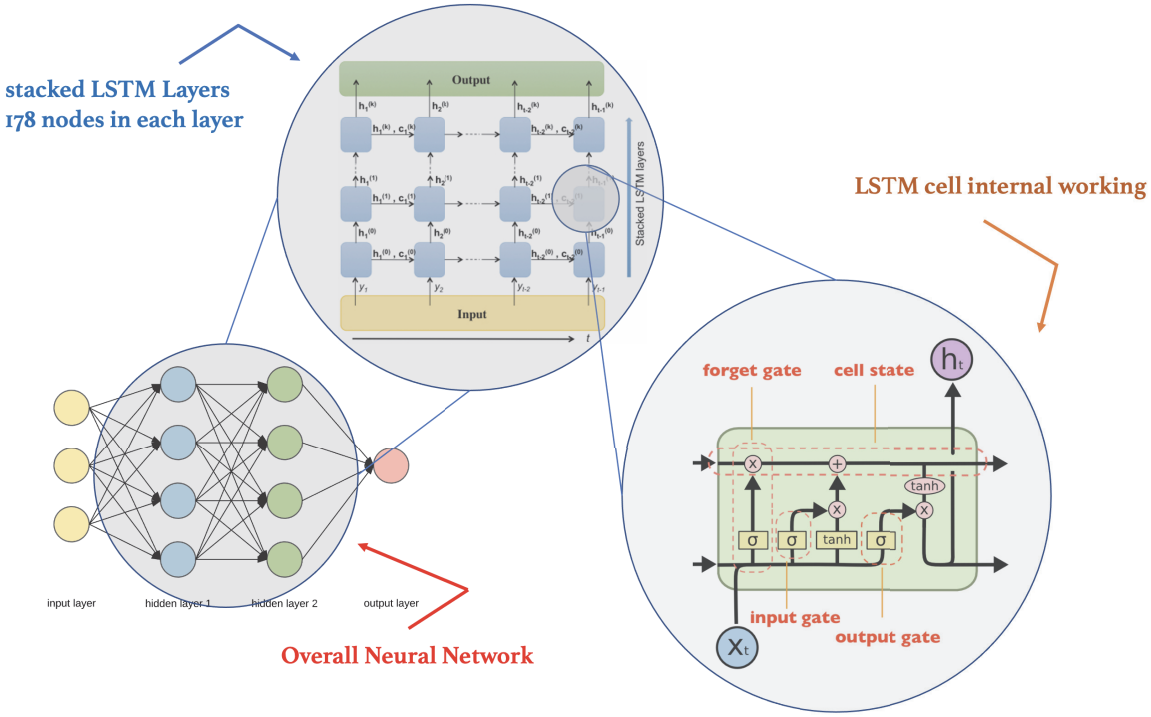

Figure 4.8: LSTM Model complete internal architecture

4.2.2 LSTM Evaluation and Results

Following the model's construction, it is trained on customized data according to the acceptance requirement of LSTM. Subsequently, the learned model is tested on new

data to determine the model's actual performance. During testing, it is important to consider that the model learns from past data to predict future observations [104] since a time series model is explicitly developed to predict the future based on previous observations.

In terms of evaluation, few metrics were dropped in the study due to the following intrinsic reasons. Firstly, Mean Absolute Percentage Error (MAPE, Equation 2.9)) produced infinite and undefined results when encountered a zero actual value which is common in meteorological variables. Secondly, Mean Absolute Error (MAE, Equation 2.6) generated biasing results in the case of negative predictions for meteorological variables, which must be a significant deterioration trigger. Finally, the Mean Absolute Scaled Error is dropped as it calculates the mean absolute difference of the training set. Especially, it can not be used as a comparative measure as the size of the training set for the fusion model is much smaller than the base model, thereby resulting in higher values for the fusion model. Although they are scale-dependent and vulnerable to outliers, Root Mean Squared Error (RMSE, Equation 2.8) and Mean Squared Error (MSE, Equation 2.7) are opted to evaluate the performance because the features are re-scaled to minimise scaling dependence. In addition, a separate model is developed to identify anomalies, rendering outliers less of a concern in the current predictions.

MSE and RMSE are calculated repeatedly with varying sizes of training and testing data. In particular Repeated-Holdout approach is used for performance estimation. Table 4.2 presents the results of the model for size variants, and the average of all the estimates is considered the final estimation. From the table, MSE and RMSE for all the four target variables have acceptable values. However, MSE for maximum wave period and average wind speed tends to be varying. This is due to the possession of scales and average values; that is, wave period values are measured in seconds, for which uncertainty of four seconds is common. The same holds for average wind speed. Since it is calculated in m/s, an uncertainty of around four is reasonable.

Figures 4.9, 4.10, 4.11, 4.12 shows a schematic comparison between actual values and LSTM's predictions of wave_ht_sig unit(m), wave_ht_max unit(m), wave_period_max unit(s), wind_spd_avg unit(m s-1) on vertical axis and time order on horizontal axis.

LSTM made reliable predictions anticipating values close to the original values, including detecting rare values that are beneficial. It should be inferred from the graph that LSTM showed some disappointing results in August of 2019 due to the presence of noisy values in feature variables. Feature variables, VCAR and VWH$, recorded zeroes that impaired the functionality of LSTM in that particular month. Conversely, LSTM rendered exceptional results when the feature variables are appropriate. Finally, the model is regarded as a reliable backup for normal condition forecasts. This thesis tests the feasibility of creating a novel model to see how effectiveness can be increased. Results of that model are presented in later Section 4.2.7.

Acronym notations:

sight: Significant Wave Height: wave_ht_sig unit(m)

mxwvht: Maximum Wave Height: wave_ht_max unit(m)

wvprd: Maximum Wave Period: wave_period_max unit(s)

wndspd: Average Wind Speed: wind_spd_avg unit(m s-1)

iter's	train/test count	sight (m)		mxwvht (m)		wvprd (s)		wndspd (m/s)	
		MSE	RMSE	MSE	RMSE	MSE	RMSE	MSE	RMSE
1	20886/2319	0.113	0.337	0.294	0.542	3.068	1.751	11.935	3.454
2	19726/3479	0.109	0.331	0.297	0.545	3.098	1.760	14.250	3.775
3	18565/4640	0.152	0.390	0.396	0.629	3.493	1.868	12.901	3.591
4	17405/5800	0.159	0.398	0.410	0.640	4.319	2.078	12.362	3.515
5	16245/6960	0.139	0.373	0.363	0.602	4.026	2.006	9.400	3.065
6	15084/8121	0.145	0.381	0.380	0.616	3.903	1.975	6.459	2.541
7	13924/9281	0.137	0.370	0.358	0.598	4.415	2.101	5.760	2.400
8	12764/10441	0.159	0.399	0.436	0.660	4.811	2.193	7.852	2.802
Average Estimates		0.139	0.372	0.366	0.604	3.891	1.966	9.892	3.142

Table 4.2: LSTM Results for normal condition predictions

4.2.3 ARIMA Setting

To summarise, ARIMA, acronym for Auto Regressive Integrated Moving Average is a univariate time series model designed to forecast values based on the previous observations. ARIMA is comprised of three paramaters: Auto Regressive (p), Integrated (d), Moving Average (q). They are selected based on the stationarity of the series

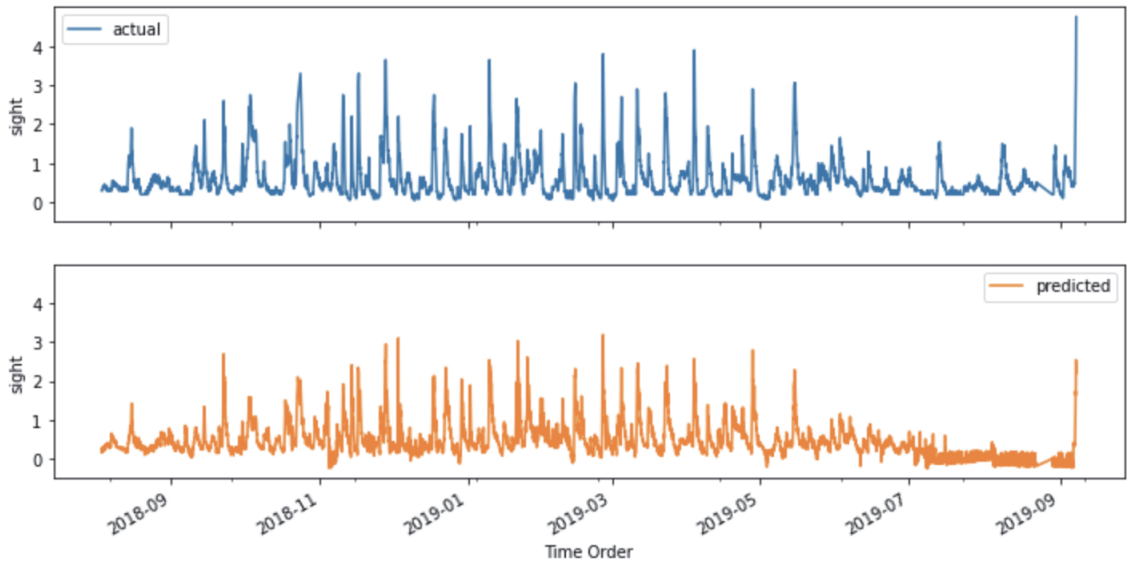

Figure 4.9: Actual and LSTM predictions of significant wave height comparison

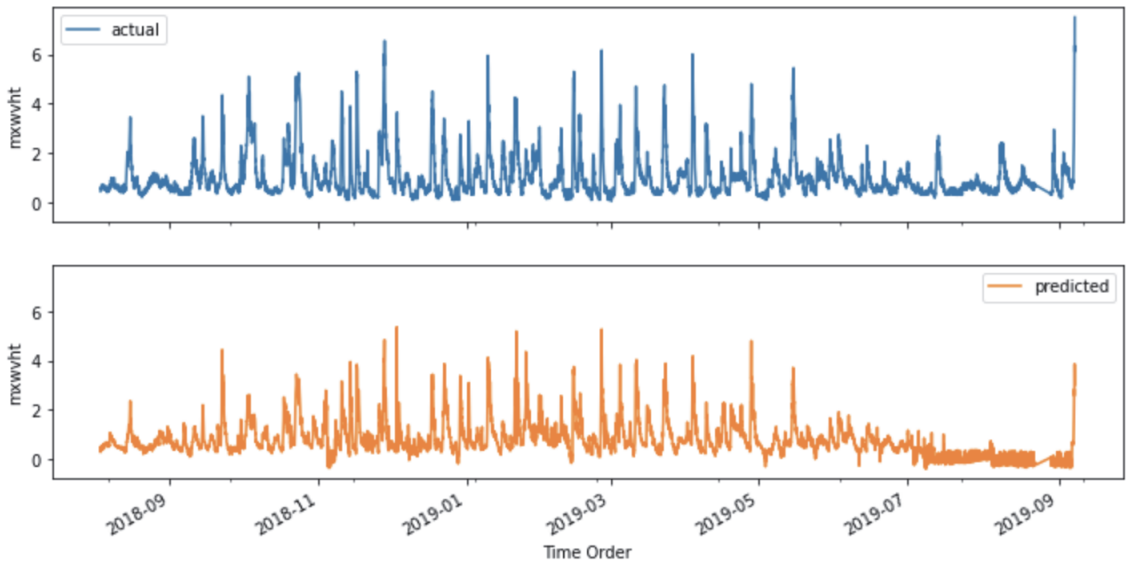

Figure 4.10: Actual and LSTM predictions of maximum wave height comparison

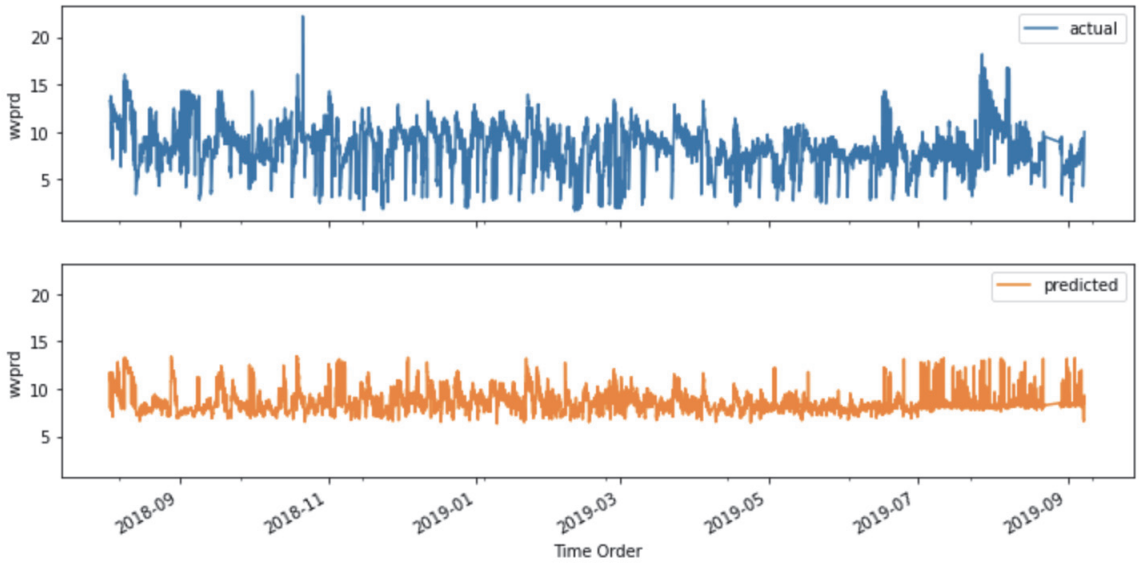

Figure 4.11: Actual and LSTM predictions of maximum wave period comparison

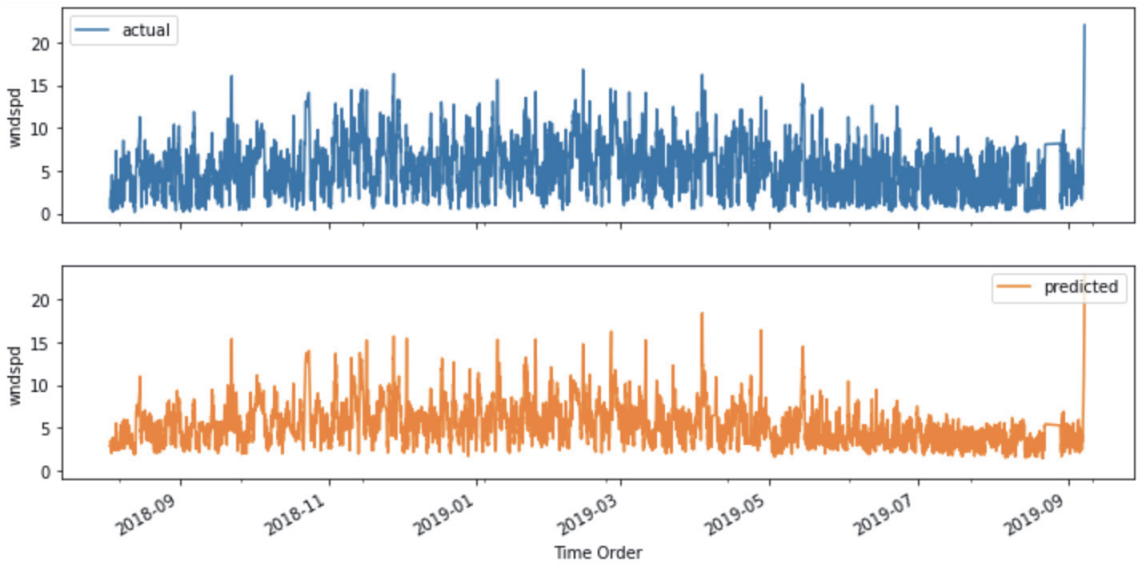

Figure 4.12: Actual and LSTM predictions of average wind speed comparison

and dependency of an observation on lagged values and errors. Initially, ACF and PACF plots for all the four target variables depicted in Section 3.2.1 were visualised to examine the dependency of a current observation on lagged observations and lagged errors. Time series module from statsmodel [94] package was used in the development of ARIMA. Depending on the plots(Figures 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.7), the parameters of four target variables for ARIMA(p,d,q) are tabulated in 4.3. Additionally, exogenous variables from ECCC that influence the prediction were included while defining the model and they are mentioned in the table.

variable name	configuration	exogenous variables
significant wave height	ARIMA(3,1,4)	VCAR,VWH$,VCMX
maximum wave height	ARIMA(2,0,3)	VCAR,VWH$,VCMX
maximum wave period	ARIMA(4,0,4)	VCAR,VWH$,VCMX,VTP$
average wind speed	ARIMA(1,0,1)	VCAR,VWH$,VCMX,WSPD, GSPD

Table 4.3: ARIMA configuration for target variables

4.2.4 ARIMA Evaluation and Results

ARIMA, being a univariate time series model, can be used to predict only one variable. Hence, four separate ARIMA models are developed to model four target variables. MSE and RMSE were used to evaluate the performance of ARIMA models for all four variables. They are calculated repeatedly using repeated holdout estimation procedures and averaged to get a final estimation. Metrics of different iterations are presented in Table 4.4.

According to Table 4.4, MSE and RMSE for all the four variables are acceptable. However, wvprd and wndspd have a wider range than the other variables, which is also acceptable due to the scale factor discussed in the LSTM predictions scenario in the previous section. Figures 4.13, 4.14, 4.15, 4.16 shows a graphical comparison for the actual series and ARIMA forecasts. It can be visualised that all the target predictions are acceptable except for wvprd. Predictions for wvprd appear to be generated around the mean of the original series, that is 10, because of the huge

variance in data and poor correlation between noise and residual error [62].

Furthermore, evidence demonstrating the superiority of mean forecasting models versus box-Jenkins models, such as ARIMA, are explained [38]. For instance, a statement summarises that the box-Jenkins model provided the least accurate predictions, with 17 percent larger inaccuracy than a naïve forecast on 29 monthly series data, confirming the empirical evidence of mean series generation superiority. Therefore, our model attempted to forecast data centered on the mean, which is considered a reliable model for the wvprd variable.

iter's	train/test count	sight (m)		mxwvht (m)		wvprd (s)		wndspd (m/s)	
		MSE	RMSE	MSE	RMSE	MSE	RMSE	MSE	RMSE
1	20886/2319	0.130	0.360	0.329	0.574	4.382	2.093	10.004	3.163
2	19726/3479	0.193	0.439	0.393	0.627	4.747	2.178	11.380	3.373
3	18565/4640	0.206	0.454	0.397	0.630	6.206	2.491	10.537	3.246
4	17405/5800	0.152	0.390	0.343	0.585	6.224	2.494	9.186	3.030
5	16245/6960	0.150	0.388	0.337	0.580	5.056	2.248	8.334	2.886
6	15084/8121	0.176	0.419	0.328	0.572	4.777	2.185	7.641	2.764
7	13924/9281	0.286	0.534	0.340	0.583	4.929	2.200	7.005	2.646
8	12764/10441	0.144	0.379	0.346	0.588	4.788	2.188	6.506	2.550
Average Estimates		0.179	0.420	0.351	0.592	5.138	2.259	8.824	2.957

Table 4.4: ARIMA Results for normal condition predictions

4.2.5 Fusion Model Setting

Ens-LSTM is a novel LSTM model that is intended to learn the relation between LSTM and ARIMA. Initially, a dataset is constructed based on the LSTM and ARIMA predictions, as illustrated in Figure 3.19, and is used to train and test the fusion model. Due to data size limits, the current study used a smaller LSTM network. In other words, the resulting dataset will be the same size as the testing data used for initial models (LSTM and ARIMA). As a result, the amount of the dataset is constrained, limiting the size of the fusion network. Four LSTM networks for four target variables are built to capture the pure relationship of LSTM and ARIMA for each variable. The architecture of all fusion models is constant throughout the study.

Like the LSTM model described in the preceding section for normal circumstances, the fusion model contains an input, hidden, and output layer. While the input format

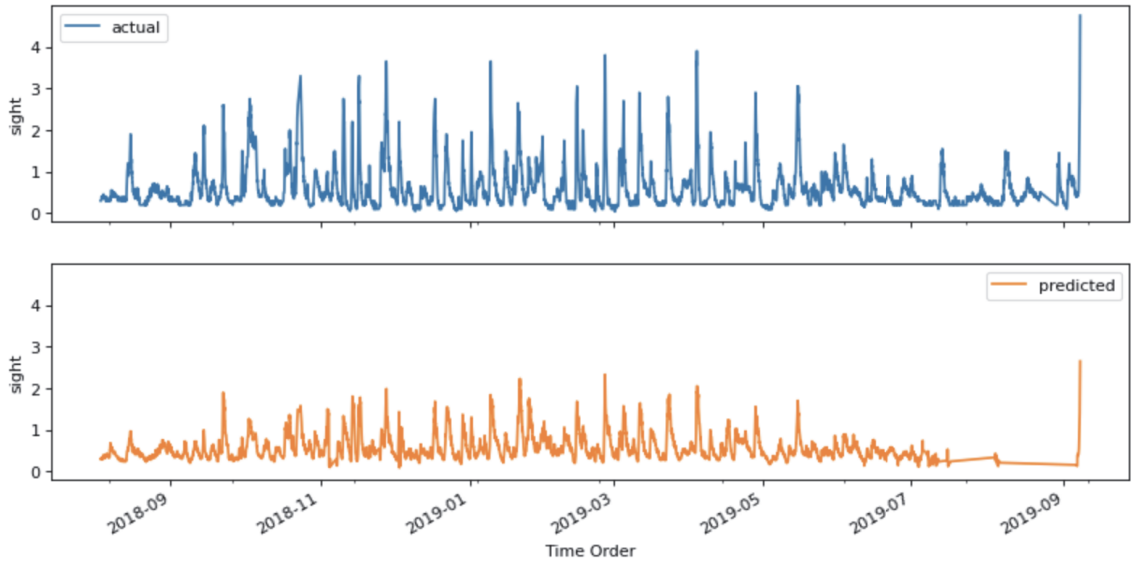

Figure 4.13: Actual and ARIMA predictions of significant wave height comparison

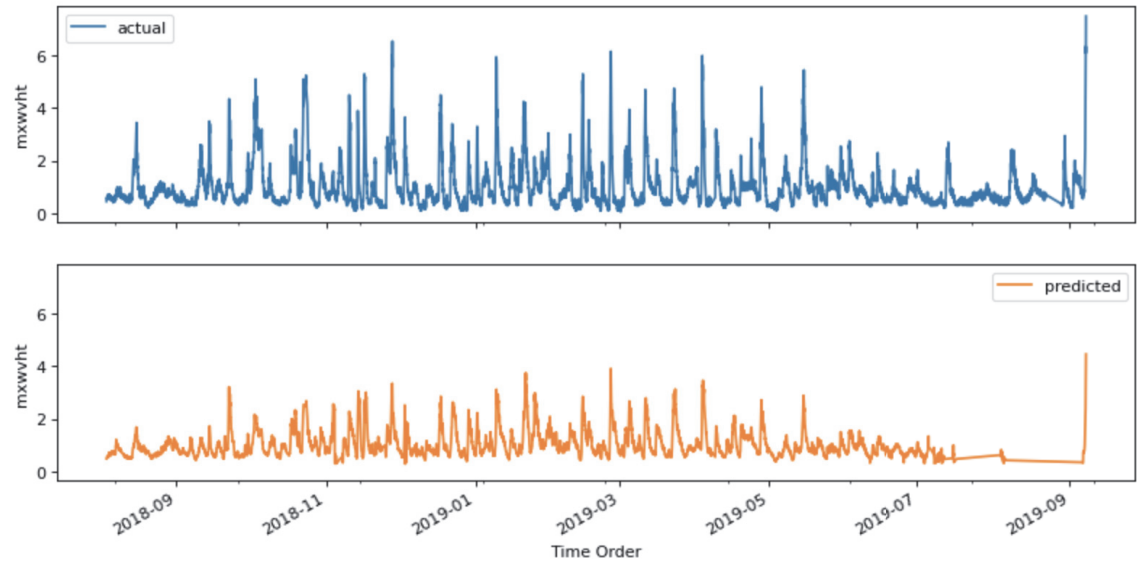

Figure 4.14: Actual and ARIMA predictions of maximum wave height comparison

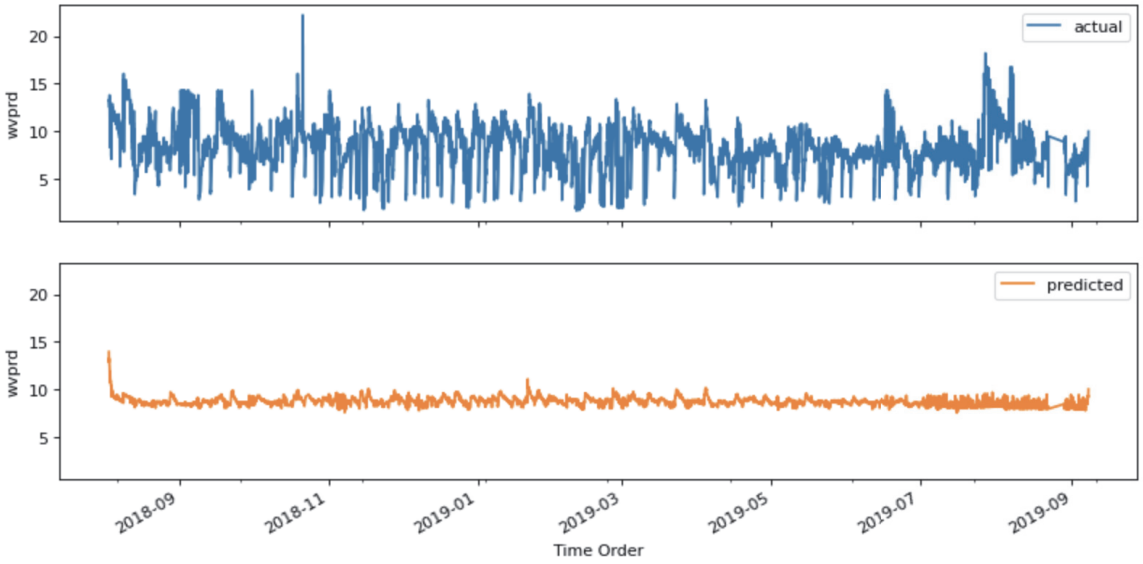

Figure 4.15: Actual and ARIMA predictions of maximum wave period comparison

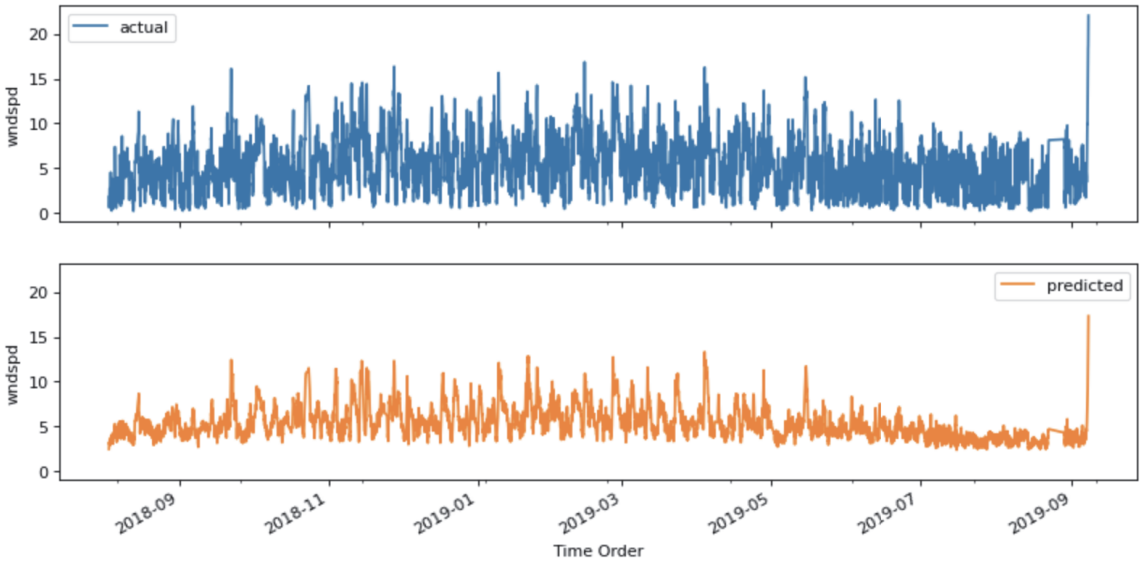

Figure 4.16: Actual and ARIMA predictions of average wind speed comparison

remains the same, the number of hidden layers is lowered to one, and the number of nodes is limited to 50. The output layer returns a single value showing the target value. An increase in the number of layers or nodes rendered the model to overfit the data. As a result, the architecture mentioned above has been finalised for usage as a fusion model.

4.2.6 Fusion Model Evaluation

The evaluation of the ensemble model is similar to the evaluation of individual models. Root Mean Squared Error (RMSE) (Equation 2.8), Mean Squared Error (MSE) (Equation 2.7), Mean Absolute Percentage Error (MAPE) (Equation 2.9) are calculated and a significant estimate is computed using an estimation approach, Repeated-holdout, as discussed in Section 2.1.4. The estimating technique is identical to that of anomaly detection estimation procedure (Page 63, Section 4.1.3) and the flow is depicted in Figure 4.2. All the metrics produced during each iteration are averaged to obtain a reliable metric value.

4.2.7 Fusion Model Results

MSE and RMSE are utilised as defining metrics for fusion model assessment. They are computed repeatedly using the repeated hold out method, and a final estimate is generated by averaging the metrics collected in each iteration. In 4.5, the results for each iteration are tabulated. The necessary inference from the table is the positive progression of metrics with increasing size, i.e., the loss in prediction tends to decrease with increasing dataset size. Iteration 2, which gave comparatively superior results, is a situation in which 80% of the total data is utilised to train initial LSTM, ARIMA and predictions are acquired from the remaining 20% of the data. 70 percent of the prediction data is then utilised to train the Ens-LSTM. Hence, it is important to have quality training for the initial model to achieve superior outcomes for the fusion model. Finally, even with limited training, the fusion model produced competent results compared to the initial models in terms of performance. As a result, if the fusion model is trained with adequate data, it will produce even better outcomes.

Figures 4.17, 4.18, 4.19, 4.20 depict the comparison of predictions for all the three developed models with the actual values. The comparison demonstrates that the

iter's	train/test count	sight (m)		mxwvht (m)		wvprd (s)		wndspd (m/s)	
		MSE	RMSE	MSE	RMSE	MSE	RMSE	MSE	RMSE
1	10024/2783	0.126	0.355	0.333	0.577	4.242	2.059	9.168	3.027
2	8353/2319	0.120	0.347	0.308	0.555	4.700	2.168	7.021	2.649
3	7518/5567	0.153	0.391	0.416	0.645	5.445	2.333	8.202	2.864
4	6682/1855	0.107	0.328	0.298	0.546	3.495	1.869	7.342	2.709
5	5847/2783	0.133	0.365	0.347	0.589	4.530	2.128	7.864	2.804
6	5011/1391	0.105	0.324	0.289	0.538	2.961	1.720	4.422	2.102
7	3340/927	0.091	0.301	0.424	0.651	2.031	1.425	5.684	2.384
8	2505/1855	0.267	0.516	0.763	0.873	4.903	2.214	6.675	2.583
Average Estimates		0.160	0.393	0.397	0.621	3.638	1.589	7.547	2.736

Table 4.5: Fusion model results for normal condition predictions

fusion model attempts to mimic the performance of LSTM, indicating that the initial LSTM was given higher weight. Figure 4.19 shows how ARIMA's wvprd forecasts attempted to influence the fusion model to create mean-centered forecasts. However, because of the effect of LSTM, the fusion model was able to exhibit some nonlinearity in the predictions. This is one of the benefits of using a fusion model. In other words, even if either of the linear and non-linear models fail severely, the fusion model still manages to produce normal results with a lower loss. To conclude, it can be stated that the performance of the initial models has a significant influence on the performance of the fusion model, and therefore the need for extra data helps the fusion model perform better.

4.3 Results Overview

Firstly, anomalies in the data were identified using a reconstruction-based methodology. The Recon-LSTM-AE in the current study achieved an average recall and f1-score of 100% and precision of 98.04% (Table 4.1). The notable takeaway from the thesis is that the study managed to produce a model that can anticipate anomalies with high precision compared to state-of-art results.

Secondly, for normal predictions, while LSTM is considered to model the non-linear part, ARIMA is opted to capture trends and seasonal components within the data. One LSTM for all four target variables and four ARIMA for each target variable was built and evaluated separately. The initial models managed to forecast normal

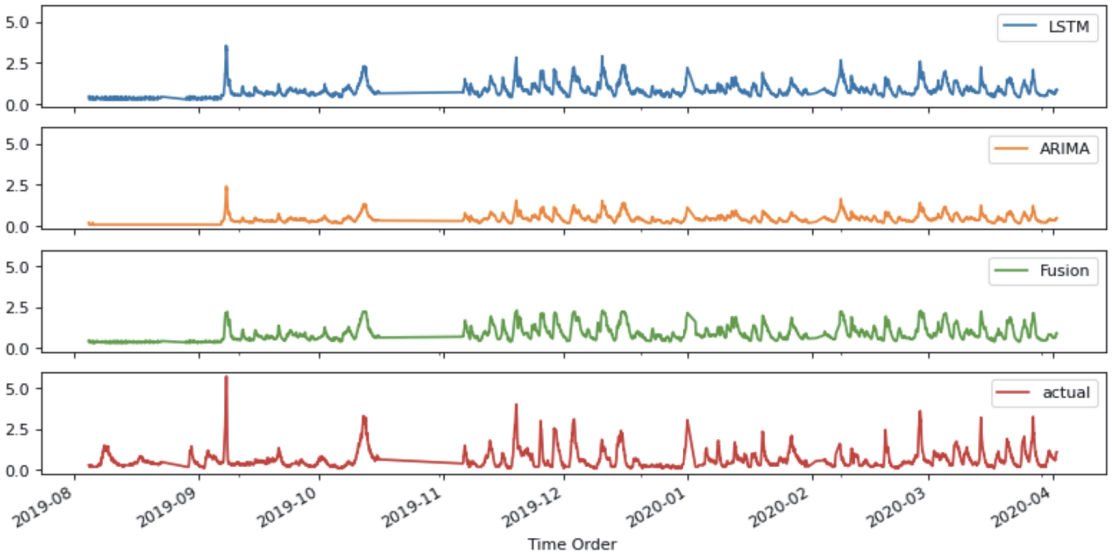

Figure 4.17: Fusion, LSTM and ARIMA significant wave height comparison

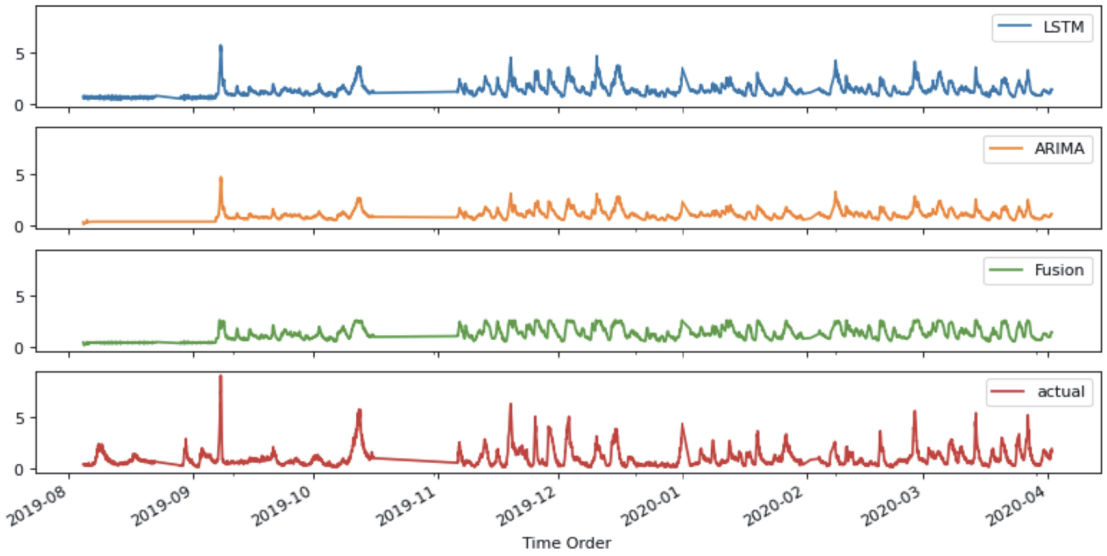

Figure 4.18: Fusion, LSTM and ARIMA maximum wave height comparison

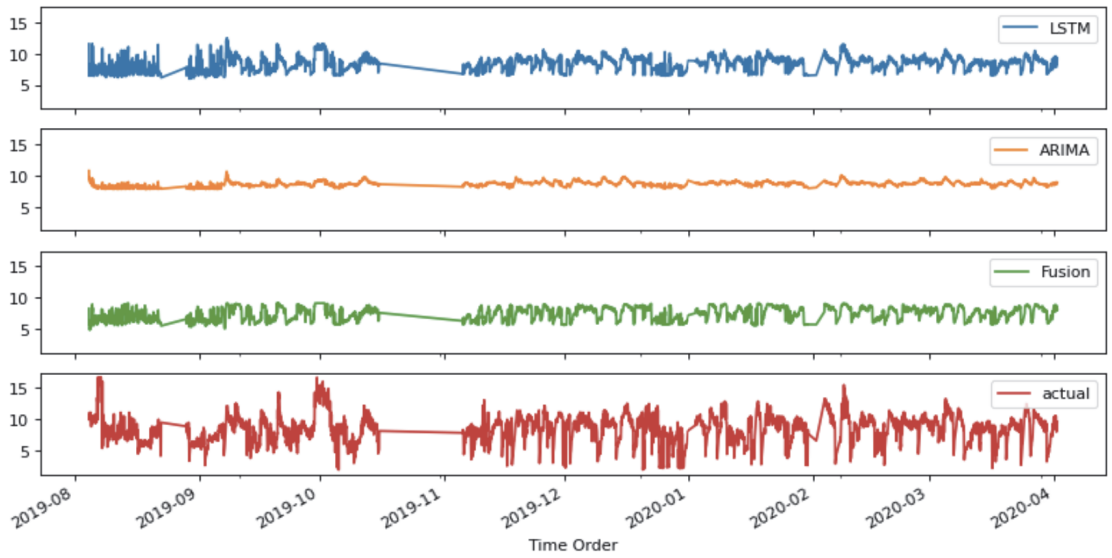

Figure 4.19: Fusion, LSTM and ARIMA maximum wave period comparison

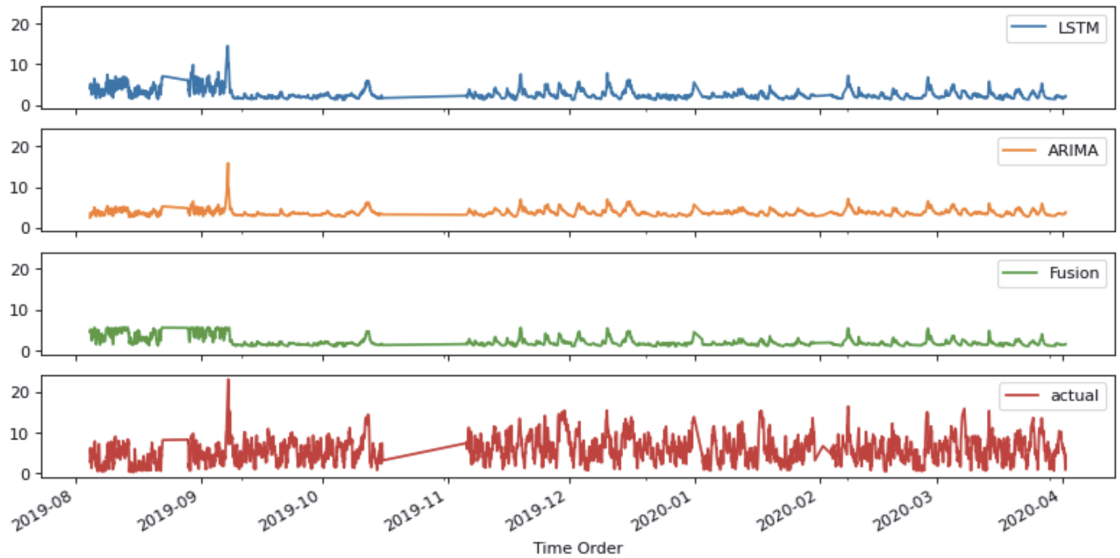

Figure 4.20: Fusion, LSTM and ARIMA average wind speed comparison

behavior with reliable MSE and RMSE for all the variables (Table 4.4, 4.2). However, models faced performance deterioration in predicting maximum wave period variable due to enormous variance and less correlation of lagged errors within the respective variable. To account for both linearity and non-linearity to increase efficiency, a third model was (Ens-LSTM) built as a fusion of the initial model's predictions to frame a relationship between the two models.

All metrics for all the models (LSTM, ARIMA, Ens-LSTM), including the Random Forest used in the previous work, are compared. To ensure the results are not skewed, a new evaluation procedure is adopted and is depicted in Figure 4.21. Initially, the main dataset is divided into training and testing parts denoted by $tr1$, $te1$. Baseline models LSTM, ARIMA, and previous work model-Random Forest are trained on $tr1$. $tr1$ is again divided into two parts $tr2$, $te2$. $te2$ can be called a validation set. Another LSTM and ARIMA are trained on $tr2$, and predictions from $te2$ are considered as fusion datasets. Ens-LSTM is trained on the fusion dataset and tested on the main test set, $te1$ taking input predictions from the initially developed base models. The metrics are calculated and presented in Table 4.6. The metrics and standard deviation of all the models are compared including the Random Forest that is used in the previous work, for six iterations of repeated holdout estimation procedure. Ens-LSTM produced superior metrics in predicting wind speed compared to the baseline models and produced similar results compared to random forest. Also, fusion model produced equipotential results for significant wave height and maximum wave height variables, despite having less training data than the baseline models. For a detailed interpretation, the predictive performance of all the models for the same data instances is presented in Table 4.6. Several comparative graphs (Figures 4.17, 4.18, 4.19 and 4.20) show the predictive performance of all the developed models. Based on the obtained results, it is believed that Ens-LSTM would render even better outcomes with more adequate training, which will be addressed in future work.

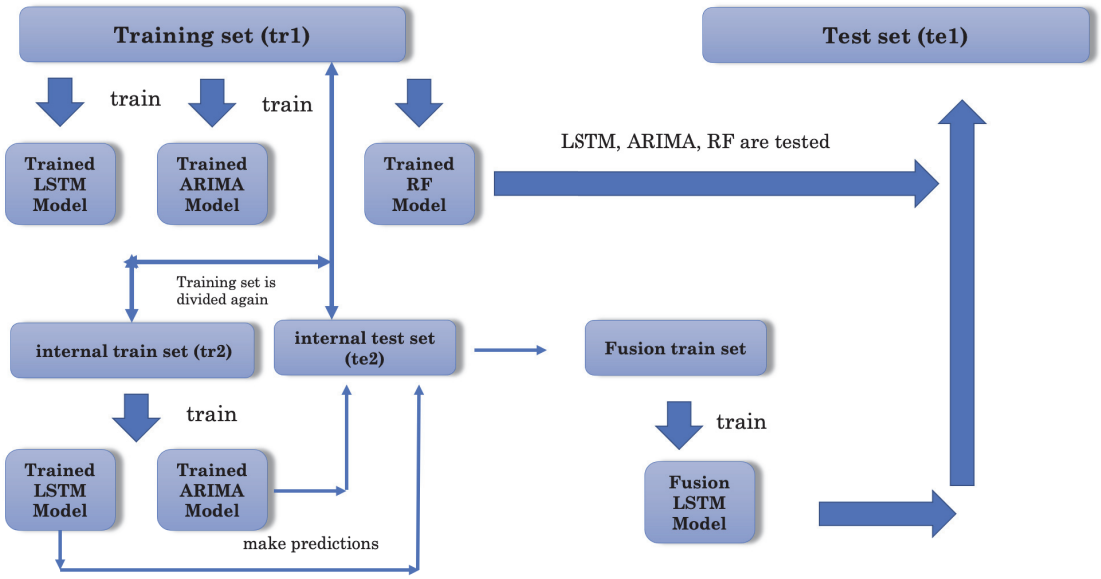

Figure 4.21: Evaluation procedure for the models

Iter	model	sight		mxwvht		wvprd		wndspd	
		MSE	RMSE	MSE	RMSE	MSE	RMSE	MSE	RMSE
1	RF	0.12	0.35	0.42	0.65	4.40	2.10	5.68	2.38
	LSTM	0.16	0.40	0.43	0.66	3.52	1.88	7.72	2.78
	ARIMA	0.20	0.45	0.39	0.62	4.55	2.13	9.60	3.10
	Fusion	0.15	0.39	0.51	0.71	4.20	2.05	6.90	2.63
2	RF	0.11	0.33	0.42	0.65	5.07	2.25	6.41	2.53
	LSTM	0.13	0.36	0.36	0.60	3.92	1.98	8.01	2.83
	ARIMA	0.21	0.46	0.34	0.58	5.08	2.25	7.04	2.65
	Fusion	0.15	0.39	0.60	0.77	4.37	2.09	7.30	2.70
3	RF	0.11	0.33	0.41	0.64	5.10	2.26	5.98	2.45
	LSTM	0.13	0.36	0.34	0.58	4.04	2.01	7.51	2.74
	ARIMA	0.15	0.39	0.33	0.57	4.95	2.22	7.38	2.72
	Fusion	0.17	0.41	0.46	0.68	5.03	2.24	6.20	2.49
4	RF	0.14	0.37	0.32	0.57	5.07	2.25	5.41	2.33
	LSTM	0.14	0.37	0.36	0.60	3.92	1.98	6.01	2.45
	ARIMA	0.20	0.45	0.34	0.58	5.08	2.25	7.04	2.65
	Fusion	0.13	0.36	0.49	0.70	4.88	2.20	5.90	2.43
5	RF	0.15	0.39	0.31	0.56	5.23	2.29	4.14	2.03
	LSTM	0.14	0.37	0.35	0.59	4.21	2.05	5.71	3.29
	ARIMA	0.13	0.36	0.34	0.58	4.90	2.21	6.20	2.49
	Fusion	0.14	0.37	0.53	0.73	4.55	2.13	5.20	2.28
6	RF	0.12	0.35	0.39	0.62	4.97	2.23	5.76	2.40
	LSTM	0.15	0.39	0.37	0.61	4.12	2.03	7.09	2.66
	ARIMA	0.17	0.41	0.35	0.59	5.01	2.24	6.52	2.55
	Fusion	0.16	0.40	0.49	0.70	4.79	2.19	6.17	2.48
Avg RF ± std		**0.14** ± 0.02	0.35 ± 0.02	0.38 ± 0.05	0.61 ± 0.04	4.97 ± 0.27	2.23 ± 0.06	6.56 ± 0.71	2.65 ± 0.16
Avg LSTM ± std		**0.14** ± 0.01	0.38 ± 0.01	0.37 ± 0.03	0.61 ± 0.02	**3.96** ± 0.22	1.99 ± 0.06	7.01 ± 0.86	2.46 ± 0.17
Avg ARIMA ± std		0.18 ± 0.03	0.42 ± 0.04	**0.35** ± 0.02	0.59 ± 0.02	4.93 ± 0.18	2.22 ± 0.04	7.30 ± 1.10	2.69 ± 0.20
Avg Fusion ± std		0.15 ± 0.01	0.39 ± 0.02	0.41 ± 0.04	0.72 ± 0.03	4.63 ± 0.29	2.15 ± 0.07	**6.28** ± 0.68	2.50 ± 0.14

Table 4.6: Results comparison between Random Forest, LSTM, ARIMA, and Ens-LSTM,

Chapter 5

Conclusion

This chapter gives an overall view of the thesis, including major results, limitations, and future work.

5.1 Summary

Time series data is a set of observations acquired at a regular interval of time. The frequency of collection is determined by the period between subsequent observations being accumulated. For example, sales are reported monthly, whereas weather is reported every hour. In today's world, the main focus is on using such recordings to benefit specific sectors or organisations. For successful interpretation, raw time series data requires modifications such as denoising and other necessary preprocessing such as imputing. Future occurrences are then anticipated based on the patterns in the prior data. The forecasting process is accomplished through the use of a machine learning model.

The SMA-H buoy measures numerous time series meteorological characteristics, including significant wave height (meters), maximum wave height (meters), maximum wave period (seconds), and average wind speed (meters/second), which are prioritised by domain experts and thus included in the research. Many organisations employ SMA-H buoys for decision-making in order to authorise ship movement and other coastal operations. The objective of the study is to create a redundancy model for the SMA-H buoy in the event of a failure utilising an ECCC buoy 13 kilometers distant in the open ocean. Furthermore, a model is explicitly developed to predict unusual phenomena such as hurricanes and snowstorms that hinder vessel movement.

Firstly, the idea behind anomaly detection is to build an autoencoder (called Recon-LSTM-AE in this work) that can learn patterns from normal data to the maximum extent achievable. The greatest loss in reconstructing training data is determined and utilised as the boundary for the normal occurrences. So, when the

same model attempts to reconstruct an anomalous event, it completely fails, resulting in a massive loss. When a loss exceeds the threshold error, it is termed an abnormality. To test model performance, measures like recall and F1-score were used.

Abnormal events were removed from all ECCC training data, and data that mostly have normal event instances was sent to stacked Long Short Term Memory (LSTM) for training. The greatest loss is calculated after training by recreating the training data. Later, the loss for each occurrence of testing is computed and categorised as abnormal or normal based on whether it is larger than or less than the threshold. Before predicting, the testing data is labeled normal or abnormal based on the threshold SMA-H variable values given by the domain experts. The threshold values for significant wave height, maximum wave height, maximum wave period, and average wind speed are 2 meters, 6 meters, 7 seconds, and 15.0 meters/second, respectively. Finally, the model generated anomalies and actual anomalies are compared. The model delivered superior results with 98.94% precision and 98.9% f1-score. The metrics are calculated using the repeated holdout estimation process (Figure 4.2) to determine if the results are significantly true. The average of metrics for multiple data size variants resulted in the same outcomes proving the model's dominance in anomaly detection. Details of metrics for every iteration are listed in Table 4.1.

Two types of machine learning models, LSTM and ARIMA, were implemented to learn non-linear and linear elements for normal condition predictions. A third model, namely Ens-LSTM, was studied and trained on the data based on the predictions from the previously mentioned models to capture both linear and non-linear entities. All the machine learning models were trained on historical data and evaluated on future occurrences. Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) were employed as metrics to assess the performance of the models. Repeated holdout (Figure 4.2) procedure is used as estimation methodology to estimate the metrics generated by the models.

Initially, an LSTM neural network was implemented that uses ECCC buoy's variables as input data to predict SMA-H buoy's variables. The model was trained and tested to yield average MSEs of 0.13 meters, 0.36 meters, 3.89 seconds, 9.89 meters/second, and RMSEs of 0.37 meters, 0.60 meters, 1.96 seconds, 3.14 meters/second for four target SMA-H variables. Detailed outcomes for every iteration are described

in Table 4.2. Secondly, ARIMA is used to discover linear trends in the data. ARIMA being a univariate model only learns patterns of a single variable. Hence four ARIMA models with distinct influencing variables are constructed. Influence variables are chosen based on the correlation matrix from figure 3.2. All the ARIMA models produced MSEs of 0.17 meters, 0.35 meters, 5.13 seconds, 8.80 meters/second, and RMSEs of 0.42 meters, 0.59 meters, 2.25 seconds, 2.90 meters/second for SMA-H variables. Outcomes are described in Table 4.4. The predictions from LSTM and ARIMA are mapped with original values to build a new dataset, which is then fed to Ens-LSTM. Ens-LSTM is trained on a smaller dataset since the dataset must be built from earlier model predictions, which is a significant restriction. Finally, the fusion model yielded MSE values of 0.16 meters, 0.39 meters, 3.63 seconds, and 7.54 meters/second. Detailed results of fusion model are presented in Table 4.5. Although with a relatively very small dataset, the fusion model produced competent results as LSTM and ARIMA.

A comparative analysis is made between all the models using an evaluation procedure depicted in Figure 4.21 and are shown in Table 4.6. The fusion model achieved better results for the average wind speed variable than all the other models, with a 6.28 mean squared error. LSTM proved superior in predicting significant wave height and maximum wave period with 0.14 and 3.96 mse. ARIMA stayed marginally higher than the other models in predicting the maximum wave height variable with 0.35 mse. It is believed that the fusion model could produce superior results if another machine learning model replaces ARIMA that degraded the performance. However, LSTM in the current study can be used as a reliable temporary solution as it managed to produce expected outcomes until the fusion model is thoroughly trained.

Overall, a baseline model like LSTM can be used as temporary redundancy as the results are satisfactory. However, with more testing, the fusion model can be readily deployed after integrating it with the prediction intervals. For anomaly anticipation, Recon-LSTM-AE can be readily deployed in real-time to anticipate extreme conditions, and thereby necessary steps can be taken.

On the GUI front, a web page developed for faster interpretability can be accessed once hosted on a public server. The webpage gives access to train the model in case of deterioration, and other tasks like scheduling the predictions can also be performed.

To conclude, the overall implementation helps stakeholders analyse the predictions from the machine learning model and interpret them quickly to make faster decisions.

5.2 Limitations

Limitations of the current study include:

- Fusion model in normal condition prediction is trained on a considerably smaller dataset than the initial LSTM and ARIMA models, which is considered a significant constraint for the model's performance. Although the fusion model produced equivalent results, it can be improved by further training.
- The confidence intervals in the Figure 3.27 and Figure 3.25 are constructed from the variance in the ARIMA predictions assuming that the residuals in ARIMA and fusion model are not significantly different.

5.3 Future Work

Future work of the current study includes:

- The baseline models for fusion setup have to be experimented with other machine learning models to check if superior performance can v=be achieved.
- Confidence intervals should be tested further and should be integrated with the fusion model.
- After accumulating sufficient data, for instance, six months of equivalent data, the fusion model must be trained to produce more effective results than the baseline models.
- Web page has to be tested with real-time users and hosted on a public server to make it live and accessible for other users.
- Currently, four fusion models are trained to deliver results for all four target variables simultaneously. A single model with relevant dependent variables for each target variable should be implemented. In other words, all the data should be merged to have a single dataset to train a single fusion model for all four target variables, thereby reducing the complexity.

Bibliography

[1] Atlantic pilotage authority home page. https://www.atlanticpilotage.com.

[2] Center for ocean ventures and entrepreneurship home page. https://coveocean.com.

[3] Environment and climate change canada buoy. https://www.meds-sdmm.dfo-mpo.gc.ca/isdm-gdsi/waves-vagues/data-donnees/data-donnees-eng.asp?medsid=C44258.

[4] Support vector regressor api. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

[5] *Time Series*, pages 536–539. Springer New York, New York, NY, 2008.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, and Matthieu Devin. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, and Matthieu Devin. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[8] Martín Abadi, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[9] Martín Abadi, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[10] akhilendra. 5 statistical methods for forecasting quantitative time series. https://akhilendra.com/evaluation-metrics-regression-mae-mse-rmse-rmsle/, May 2016.

[11] Haitham M Al-Deek. Use of vessel freight data to forecast heavy truck movements at seaports. *Transportation research record*, 1804(1):217–224, 2002.

[12] Khanin Artur. Time series and how to detect anomalies in them — part i. https://becominghuman.ai/time-series-and-how-to-detect-anomalies-in-them-part-i-7f9f6c2ad32e, 2020.

[13] Will Badr. Auto-encoder: What is it? and what is it used for? (part 1). `https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726`, 2019.

[14] SH Bari, MT Rahman, MM Hussain, and Sourav Ray. Forecasting monthly precipitation in sylhet city using arima model. *Civil and Environmental Research*, 7(1):69–77, 2015.

[15] bista solutions. 5 statistical methods for forecasting quantitative time series. `https://www.bistasolutions.com/resources/blogs/5-statistical-methods-for-forecasting-quantitative-time-series/`, May 2016.

[16] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[17] Robert Goodell Brown. *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.

[18] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[19] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[20] Jacquelyn Bulao. How much data is created every day in 2021? `https://techjury.net/blog/how-much-data-is-created-every-day/#gref`.

[21] Shivam Chaudhary. Why “1.5” in iqr method of outlier detection? https://towardsdatascience.com/why-1-5-in-iqr-method-of-outlier-detection-5d07fdc82097, 2019.

[22] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[23] Jie Chen, Guo-Qiang Zeng, Wuneng Zhou, Wei Du, and Kang-Di Lu. Wind speed forecasting using nonlinear-learning ensemble of deep learning time series prediction and extremal optimization. *Energy conversion and management*, 165:681–695, 2018.

[24] Peng Chen, Hongyong Yuan, and Xueming Shu. Forecasting crime using the arima model. In *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 5, pages 627–630. IEEE, 2008.

[25] Hyeong Kyu Choi. Stock price correlation coefficient prediction with arima-lstm hybrid model. *arXiv preprint arXiv:1808.01560*, 2018.

[26] Madeline Connall. Top 20 big data statistics for 2021. `https://www.sigmacomputing.com/blog/top-20-big-data-statistics/`.

[27] JM Craddock. The analysis of meteorological time series for use in forecasting. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 15(2):167–190, 1965.

[28] Emmanuel Dave, Albert Leonardo, Marethia Jeanice, and Novita Hanafiah. Forecasting indonesia exports using a hybrid model arima-lstm. *Procedia Computer Science*, 179:480–487, 2021.

[29] DeepSense. Deepsense computing resources. `https://docs.deepsense.ca/index.php?title=Resources`.

[30] DeepSense. Deepsense home page. `https://deepsense.ca`.

[31] DFO(2020). Marine environmental data section. `https://www.meds-sdmm.dfo-mpo.gc.ca`.

[32] Stats Documentation. Stationarity and detrending (adf/kpss). `https://www.statsmodels.org/stable/examples/notebooks/generated/stationarity_detrending_adf_kpss.html`.

[33] André Listou Ellefsen, Emil Bjørlykhaug, Vilmar Æsøy, and Houxiang Zhang. An unsupervised reconstruction-based fault detection algorithm for maritime components. *IEEE Access*, 7:16101–16109, 2019.

[34] Dongyan Fan, Hai Sun, Jun Yao, Kai Zhang, Xia Yan, and Zhixue Sun. Well production forecasting based on arima-lstm model considering manual operations. *Energy*, 220:119708, 2021.

[35] Jesuseyi Will Fasuyi, Jason Newport, and Chris Whidden. A machine learning redundancy model for the herring cove smart buoy. *Journal of Ocean Technology*, 15(3), 2020.

[36] Centers for Disease Control and Prevention. Principles of epidemiology in public health practice: Measures of spread. https://www.cdc.gov/csels/dsepd/ss1978/lesson2/section7.html#ALT27.

[37] Center for Ocean Ventures. Smart herring cove buoy (sma-h buoy). https://www.smartatlantic.ca/station_alt.html?id=halifax.

[38] forecaster (https://stats.stackexchange.com/users/29137/forecaster). Is it unusual for the mean to outperform arima? Cross Validated. URL:https://stats.stackexchange.com/q/125016 (version: 2014-11-21).

[39] Jim Frost. Multicollinearity in regression analysis: Problems, detection, and solutions. https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/.

[40] Johannes Fürnkranz. *Decision Tree*, pages 263–267. Springer US, Boston, MA, 2010.

[41] Claudio Gallicchio. Short-term memory of deep rnn. *arXiv preprint arXiv:1802.00748*, 2018.

[42] A. Geetha and G. M. Nasira. Time series modeling and forecasting: Tropical cyclone prediction using arima model. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 3080–3086, 2016.

[43] Seymour Geisser. The predictive sample reuse method with applications. *Journal of the American statistical Association*, 70(350):320–328, 1975.

[44] Bidisha Ghosh, Biswajit Basu, and Margaret O'Mahony. Time-series modelling for forecasting vehicular traffic flow in dublin. In *84th Annual Meeting of the Transportation Research Board, Washington, DC*, 2005.

[45] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[46] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[47] Ernesta Grigonytė and Eglė Butkevičiūtė. Short-term wind speed forecasting using arima model. *Energetika*, 62(1-2), 2016.

[48] Alex Grönholm. Apscheduler documentation. https://apscheduler.readthedocs.io/en/stable/versionhistory.html#id2.

[49] Latifa Guesmi, H. Fathallah, and Mourad Menif. *Modulation Format Recognition Using Artificial Neural Networks for the Next Generation Optical Networks*. 02 2018.

[50] Aishwarya Gulve. Everything about components of time series: Part-1. `https://aishwaryagulve97.medium.com/everything-about-components-of-time-series-part-1-7476fb521477`, 2020.

[51] Jeff Heaton. The number of hidden layers. `https://www.heatonresearch.com/2017/06/01/hidden-layers.html`, 2017.

[52] Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. *Advances in neural information processing systems*, 26:190–198, 2013.

[53] Siu Lau Ho and Min Xie. The use of arima models for reliability forecasting and analysis. *Computers & industrial engineering*, 35(1-2):213–216, 1998.

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[55] Richard G. Hoptroff. The principles and practice of time series forecasting and business modelling using neural nets. *Neural Computing & Applications*, 1(1):59–66, 1993.

[56] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[57] Farid Kadri, Fouzi Harrou, Sondès Chaabane, and Christian Tahon. Time series modelling and forecasting of emergency department overcrowding. *Journal of medical systems*, 38(9):1–20, 2014.

[58] Zahra Karevan and Johan AK Suykens. Spatio-temporal stacked lstm for temperature prediction in weather forecasting. *arXiv preprint arXiv:1811.06341*, 2018.

[59] Petros Karvelis, Theofanis-Aristofanis Michail, Daniele Mazzei, Stefanos Petsios, Andrea Bau, Gabriele Montelisciani, and Chrysostomos Stylios. Adopting and embedding machine learning algorithms in microcontroller for weather prediction. In *2018 International Conference on Intelligent Systems (IS)*, pages 474–478. IEEE, 2018.

[60] Kevin S Killourhy and Roy A Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134. IEEE, 2009.

[61] Sungil Kim and Heeyoung Kim. A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 32(3):669–679, 2016.

[62] Michael L. Poor fit of an arima model. Cross Validated. URL:https://stats.stackexchange.com/q/278638 (version: 2017-05-10).

[63] Weizhi Li, Weirong Mo, Xu Zhang, John J Squiers, Yang Lu, Eric W Sellke, Wensheng Fan, J Michael DiMaio, and Jeffrey E Thatcher. Outlier detection and removal improves accuracy of machine learning approach to multispectral burn diagnostic imaging. *Journal of biomedical optics*, 20(12):121305, 2015.

[64] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[65] Atik Mahabub, Al-Zadid Sultan Bin Habib, M RubaiyatB Hossain Mondal, Subrato Bharati, and Prajoy Podder. Effectiveness of ensemble machine learning algorithms in weather forecasting of bangladesh. In *Innovations in Bio-Inspired Computing and Applications: Proceedings of the 11th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2020) held during December 16–18, 2020*, page 267. Springer Nature.

[66] J Mahjoobi and Ehsan Adeli Mosabbeb. Prediction of significant wave height using regressive support vector machines. *Ocean Engineering*, 36(5):339–347, 2009.

[67] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[68] Markos Markou and Sameer Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[69] Alvin Greene McNish. Statistical aspects of long-range weather-forecasting. *Eos, Transactions American Geophysical Union*, 17(1):124–129, 1936.

[70] Hugo Meinedo and Joao Neto. A stream-based audio segmentation, classification and clustering pre-processing system for broadcast news using ann models. In *Ninth European Conference on Speech Communication and Technology*, 2005.

[71] Dharmendra S Modha and Elias Masry. Prequential and cross-validated regression estimation. *Machine Learning*, 33(1):5–39, 1998.

[72] Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[73] Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[74] Sophie A Murray. The importance of ensemble techniques for operational space weather forecasting. *Space Weather*, 16(7):777–783, 2018.

[75] Shi Na, Liu Xumin, and Guan Yong. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 63–67, 2010.

[76] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, and Xinghuo Yu. Spatiotemporal anomaly detection using deep learning for real-time video surveillance. *IEEE Transactions on Industrial Informatics*, 16(1):393–402, 2019.

[77] HD Nguyen, Kim Phuc Tran, S Thomassey, and M Hamad. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57:102282, 2021.

[78] Government of Canada. Port of halifax home page. `https://www.portofhalifax.ca`.

[79] Christopher Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[80] Vaishali R Patel and Rupa G Mehta. Impact of outlier removal and normalization approach in modified k-means clustering algorithm. *International Journal of Computer Science Issues (IJCSI)*, 8(5):331, 2011.

[81] Ebberth L Paula, Marcelo Ladeira, Rommel N Carvalho, and Thiago Marzagao. Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 954–960. IEEE, 2016.

[82] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[83] Marco Peixeiro. The complete guide to time series analysis and forecasting. `https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775`, 2019.

[84] Luis Nobre Pereira. An introduction to helpful forecasting methods for hotel revenue management. *International Journal of Hospitality Management*, 58:13–23, 2016.

[85] Hean-Lee Poh, Jingtao Yao, and Teo Jašic. Neural networks for the analysis and forecasting of advertising and promotion impact. *Intelligent Systems in Accounting, Finance & Management*, 7(4):253–268, 1998.

[86] Selva Prabhakaran. Augmented dickey fuller test (adf test) – must read guide. `https://www.machinelearningplus.com/augmented-dickey-fuller-test/`, 2019.

[87] Oleksandr I Provotar, Yaroslav M Linder, and Maksym M Veres. Unsupervised anomaly detection in time series using lstm-based autoencoders. In *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, pages 513–517. IEEE, 2019.

[88] G Athanasopoulos R J Hyndman. Forecasting: Principles and practice. `https://otexts.com/fpp2/simple-methods.html`, 2018.

[89] Capt Rae. Meeting notes from assumptions for oceanographic data, April 2021.

[90] Rohan Kumar Raman, TV Sathianandan, AP Sharma, and BP Mohanty. Modelling and forecasting marine fish production in odisha using seasonal arima model. *National Academy Science Letters*, 40(6):393–397, 2017.

[91] Chitta Ranjan. Lstm autoencoder for extreme rare event classification in keras. `https://towardsdatascience.com/lstm-autoencoder-for-extreme-rare-event-classification-in-keras-ce209a224cfb`, May 2019.

[92] Matheus Henrique Dal Molin Ribeiro and Leandro dos Santos Coelho. Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series. *Applied Soft Computing*, 86:105837, 2020.

[93] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*, 2019.

[94] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[95] Jaydip Sen and Tamal Chaudhuri. A time series analysis-based forecasting framework for the indian healthcare sector. *Journal of Insurance and Financial Management*, 3(1), 2017.

[96] Nikita Shivhare, Atul Kumar Rahul, Shyam Bihari Dwivedi, and PRABHAT KUMAR SINGH Dikshit. Arima based daily weather forecasting tool: A case study for varanasi. *MAUSAM*, 70(1):133–140, 2019.

[97] Sandeep Kumar Singh and Frank Heymann. Machine learning-assisted anomaly detection in maritime navigation using ais data. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 832–838. IEEE, 2020.

[98] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. Unsupervised clustering approach for network anomaly detection. In *International conference on networked digital technologies*, pages 135–145. Springer, 2012.

[99] Luis Torgo. Lecture notes in performance estimation, November 2020.

[100] Paul Truong. Approaches to anomaly detection. https://medium.com/safetycultureengineering/approaches-to-anomaly-detection-20de4983d23, 2020.

[101] Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[102] Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, and year=2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[103] Vincent Vercruyssen, Wannes Meert, Gust Verbruggen, Koen Maes, Ruben Baumer, and Jesse Davis. Semi-supervised anomaly detection with an application to water analytics. In *2018 ieee international conference on data mining (icdm)*, volume 2018, pages 527–536. IEEE, 2018.

[104] Cerqueira Vitor, Torgo Luis, and Mozetič Igor. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, 109(11):1997–2028, 2020.

[105] Lijing Wang, Jiangzhuo Chen, and Madhav Marathe. Tdefsi: theory-guided deep learning-based epidemic forecasting with synthetic information. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 6(3):1–39, 2020.

[106] Yuan-Kai Wang, Ching-Tang Fan, Ke-Yu Cheng, and Peter Shaohua Deng. Real-time camera anomaly detection for real-world video surveillance. In *2011 International Conference on Machine Learning and Cybernetics*, volume 4, pages 1520–1525. IEEE, 2011.

[107] Zheng Wang and Yuansheng Lou. Hydrological time series forecast model based on wavelet de-noising and arima-lstm. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1697–1701. IEEE, 2019.

[108] Xiaojun Xu, Shuliang Wang, and Ying Li. Identification and predication of network attack patterns in software-defined networking. *Peer-to-Peer Networking and Applications*, 12(2):337–347, 2019.

[109] Liu Yunpeng, Hou Di, Bao Junpeng, and Qi Yong. Multi-step ahead time series forecasting for different data patterns based on lstm recurrent neural network. In *2017 14th web information systems and applications conference (WISA)*, pages 305–310. IEEE, 2017.

[110] Mohamed Akram Zaytar and Chaker El Amrani. Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. *International Journal of Computer Applications*, 143(11):7–11, 2016.

[111] Othman Zennaki, Nasredine Semmar, and Laurent Besacier. Inducing multilingual text analysis tools using bidirectional recurrent neural networks. *arXiv preprint arXiv:1609.09382*, 2016.

[112] Hong Zheng, Yunhui Cheng, and Haibin Li. Investigation of model ensemble for fine-grained air quality prediction. *China Communications*, 17(7):207–223, 2020.