Explaining Time Series Data with Interpretable Modular Neural Networks

A Thesis Presented for the degree of Doctor of Philosophy

Qiqi Su

Department of Computer Science

City St George's, University of London

qiqi.su@ieee.org

Supervisors: Dr. Christos Kloukinas and Prof. Artur Garcez

October 2025

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Christos and Artur, for their invaluable guidance, support, and encouragement throughout this research journey. Their expertise and insightful feedback have been instrumental in shaping this work.

I am also deeply grateful to my family and friends for their unwavering support, patience, and understanding. Their belief in me has been a constant source of motivation, and this accomplishment would not have been possible without their love and encouragement.

Thank you all.

Declaration

I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only copies made for study purposes, subject to normal conditions of acknowledgement.

Abstract

Multivariate time series have had many applications in areas from healthcare and finance to meteorology and life sciences. Although deep neural networks (DNNs) have shown excellent predictive performance for time series, they have been criticised for being non-interpretable. Neural Additive Models, however, are known to be fully-interpretable by construction, but may achieve far lower predictive performance than DNNs when applied to time series. This work introduces FocusLearn, a fully-interpretable modular neural network capable of matching or surpassing the predictive performance of DNNs trained on multivariate time series. The creation of FocusLearn takes inspiration from modular neural networks and additive models, as well as several experiments aimed at improving the performance of a stand-alone LSTM. In FocusLearn, a recurrent neural network learns the temporal dependencies in the data, while a modified multi-headed attention layer learns to weight selected features while also suppressing redundant features. Modular neural networks are then trained in parallel and independently, one for each selected feature. This modular approach allows the user to inspect how features influence outcomes in the exact same way as with additive models. Experimental results show that this new approach outperforms additive models in both regression and classification of time series tasks, achieving predictive performance that is comparable to state-of-the-art, non-interpretable deep networks applied to time series, and sometimes outperforms them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The analysis of time series data holds significant importance across various practical domains, ranging from finance to healthcare. However, time series analysis presents a multifaceted challenge due to the inherent complexity of temporal data, characterised by dependencies, trends, and seasonality. The dynamic and sequential nature of time series data requires advanced analytical techniques to capture meaningful patterns. Artificial Intelligence (AI) models, particularly Recurrent Neural Networks (RNNs) and Transformers, have demonstrated remarkable capabilities in addressing these challenges. Their inherent ability to capture sequential dependencies and discern intricate temporal patterns positions them as formidable tools in time series analysis. Despite their capabilities however, the application of these models in sensitive domains needs a transparent understanding of the decision-making processes they employ. In contexts where the consequences of model predictions are critical, end users require insight into why specific decisions are made. This is the fundamental motivation behind the

emergence of eXplainable Artificial Intelligence (XAI). XAI aims to provide interpretability and transparency in the decision-making mechanisms of complex models, so that users can understand and trust why these models made certain decisions.

The imperative for transparent decision-making processes in AI models, particularly in sensitive domains, sets the foundation for the important connection between AI and explainability, highlighting the need to provide clear explanations to improve the reliability and acceptance of AI-based decision support systems.

However, the complexity in explaining RNNs or similar models utilised for time series analysis arises primarily due to their sequential nature, where the order of input elements is crucial for processing. This temporal aspect adds a layer of intricacy to understanding the flow of information through the network over time. Moreover, effectively training RNNs requires addressing challenges such as the vanishing or exploding gradient problem, which can impede the capability of the model to capture long-term dependencies within the data. Furthermore, RNN architectures, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), incorporate intricate gating mechanisms to regulate information flow. Grasping the intricacies of these mechanisms and their interactions further compounds the challenge of explaining RNNs.

Most importantly, a notable gap exists in the current landscape of methods tailored for explaining time series data. Some explanation methods are confined to handling either regression or classification tasks, while others are restricted to post-hoc interpretability (trying to explain the decision produced by a model) which are often criticised for low fidelity [3], instability [50], and even inaccuracy [88, 120], since they only approximate their underlying model.

1.2 Aims and Objectives

Motivated by the above, the main aim of this paper is to develop an interpretable Deep Neural Network (DNN) time series model that is capable of both prediction and explanation. To achieve this, the objectives of such model are:

- capable of effectively predicting time series data for both regression and classification tasks, and
- capable of providing explanations.

Given the complexity of such model, we will first experiment with existing state-of-the-art time series prediction models, such as LSTM, and focus on improving upon its prediction results. This initial phase aims to identify the critical features and capabilities necessary for accurate time series prediction. Following this, we will explore ante-hoc methods (interpretable models), such as Neural Additive Model (NAM), to understand their inherent strengths and limitations. By systematically evaluating these approaches, we aim to clarify the trade-offs between interpretability, accuracy, and computational efficiency. The insights gained from these investigations will be synthesised to inform the design and development of this interpretable time series model. To this end, several sub-objectives are defined as follows:

- experiment with time series DNN models that can predict better than a stand-alone LSTM,
- design and develop an interpretable model for time series analysis that can–
  - predict better than state-of-the-art interpretable models (NAM), and
  - match the predictive performance of DNN time series models, and if possible, outperform them.
- explanations provided by the developed interpretable time series model should–
  - not require post-hoc methods,
  - be more than just feature importance, and
  - be similar to the NAM-style explanation graphs, so these plots–
    - correspond exactly to the developed interpretable time series model,
    - are easy to interpret by non-expert users,
    - can indicated a trend, and
    - can be intervened upon to ask what-if questions.

1.3 Hypotheses

On Time Series DNN Models: It is hypothesised that combining LSTM with attention mechanisms will result in a time series model that outperforms a stand-alone LSTM in terms of prediction performance measured by metrics such as Accuracy for classification problems and Mean Absolute Percentage Error (MAPE) for regression problems.

On Time Series DNN Models: It is hypothesised employing an LSTM to first capture the data representations and temporal dependencies, followed by an attention mechanism to learn from the hidden states may be more effective than using an attention mechanism initially to process the inputs, followed by an LSTM, in terms of prediction performance measured by metrics such as Accuracy for classification problems and MAPE for regression problems.

On Interpretable Model Design: It is hypothesised that the proposed interpretable time series model will achieve better prediction performance than state-of-the-art interpretable models, such as NAM, as measured by metrics such as Accuracy for classification problems and MAPE for regression problems. Furthermore, it is hypothesised that the predictive performance of the proposed interpretable model will match or exceed that of traditional DNN-based time series models, demonstrating the feasibility of combining interpretability with high predictive accuracy.

It is also hypothesised that the proposed interpretable time series model can outperform NAM, and possibly DNN time series model, because:

- the combination of attention mechanisms and LSTMs for feature learning,
- having an Modular Neural Network (MNN) to select the features and maintain interpretability,
- learned attention weights are leveraged further as learnable weights for each neural network, and
- learning and inference paths are separated.

1.4 Main Contribution

Our research endeavours to bridge the the gaps in the current landscape of time series explanation methods by introducing an interpretable DNN designed explicitly for time series analysis, that does not require post-hoc explanation methods and is flexible enough for handling both regression and classification tasks. This interpretable DNN also aims to have a better predictive performance than current state-of-the-art interpretable DNN, such as NAM [4], as well as match the predictive performance of DNN time series models, such as LSTM, and if possible, outperform them.

To achieve this, we first experimented with techniques to improve the predictive performance of a stand-alone LSTM. Drawing inspiration from transformer architectures [11], we evaluated the benefits of incorporating an attention mechanism to allow the LSTM to selectively focus on the most relevant parts of the input sequence. Combining an attention mechanism with LSTMs would not only enhance the predictive performance of LSTMs by prioritising important information but can also be used as a way to explain the LSTMs [32, 49, 122, 110]. Several approaches for combining the attention mechanism with LSTM were explored to determine the optimal configuration that would yield the best predictive performance. These experiments aimed to assess how different ways of integrating attention into the LSTM architecture could improve overall accuracy.

Furthermore, in pursuit of an interpretable time series DNN model, we explored the strategy of integrating post-hoc explanation methods as domain knowledge to improve the predictive performance of NAM, which, while capable of generating strong explanations, struggles to achieve high predictive performance that is comparable with DNN models. Building on insights gained from these experiments, we ultimately developed an interpretable model for time series that balances predictive power with transparency.

This interpretable time series DNN model, as shown in Figure 1.1, also drew inspiration from **modular networks** [19, 143] and **additive models** [4, 42]. Building upon the evidence suggesting the interpretability of linear directions

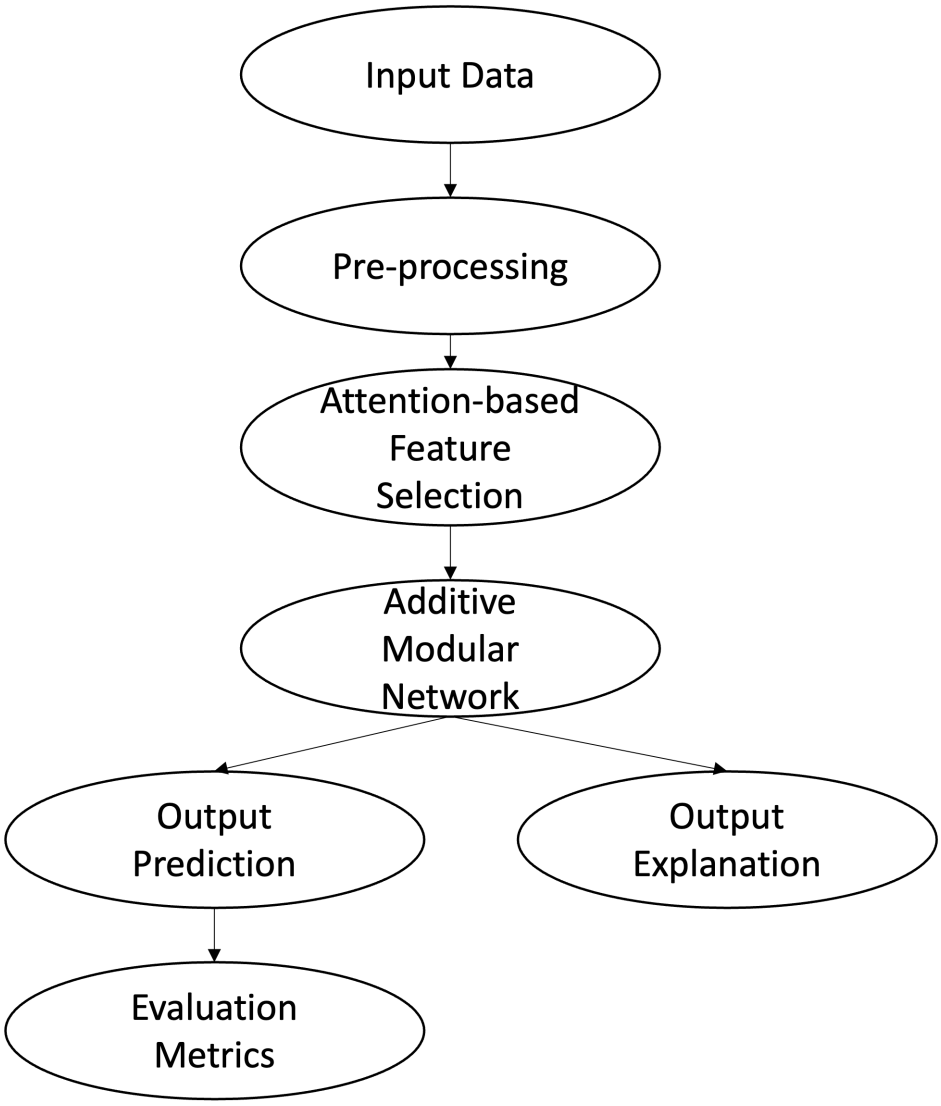


Figure 1.1: **High level representation of the proposed interpretable time series model.**

in activation space, particularly through linear combinations of neuron activations [17], our final approach involves having each neural network learn a linear combination of selected features. This process is guided by the idea that linear directions are interpretable, and we aim to decompose the complexity of DNNs into fundamental units termed features. But different from [17], we emphasise the relevance of **modularity** during the learning phase, highlighting its significance in steering the model away from convoluted structures that may hinder interpretability or attempting to make sense of such a complex model post hoc. Leveraging the **additive model** paradigm, we aggregate the contributions of individual neural networks by summing them, following the principles of NAMs [4]. The incorporation of modularity in our methodology at learning time enhances interpretability, as it allows us to discern, analyse, and systematically learn the contributions of each feature separately, thereby facilitating a more transparent understanding of the DNN's decision-making processes.

This innovative model not only excels in predicting both regression and classification outcomes but also places a strong emphasis on being interpretable by design. By filling this void, our proposed approach not only contributes to the advancement of XAI in the intricate realm of time series analysis, but can be extended to other domains.

1.5 Publications

In reverse order.

- Qiqi Su, Christos Kloukinas, and Artur Garcez. "FocusLearn: Fully-Interpretable, High-Performance Modular Neural Networks for Time Series". In *2024 International Joint Conference on Neural Networks (IJCNN)*, Yokohama, Japan, 2024, doi: : 10.1109/IJCNN60899.2024.10651481.

- Qiqi Su and Eleftheria Iliadou. "Predicting and explaining hearing aid usage using encoder-decoder with attention mechanism and SHAP". In *2022 16th International Conference on Signal-Image Technology & Internet-*

Based Systems (SITIS), Dijon, France, 2022, pp. 308-315, doi: 10.1109/SITIS57111.2022.00053.

- Eleftheria Iliadou, Qiqi Su, Dimitrios Kikidis, Thanos Bibas, and Christos Kloukinas. "Profiling hearing aid users through big data explainable artificial intelligence techniques", *Frontiers in Neurology 13 (2022): 933940*, doi: 10.3389/fneur.2022.933940.

- Qiqi Su et al. "The SMART BEAR Project: An Overview of Its Infrastructure". In: Maciaszek, L.A., Mulvenna, M.D., Ziefle, M. (eds) *Information and Communication Technologies for Ageing Well and e-Health. ICT4AWE 2022*. Communications in Computer and Information Science, vol 1856. Springer, Cham. doi: 10.1007/978-3-031-37496-8_21.

- Vadim Peretokin, Ioannis Basdekis, Ioannis Kouris, Jonatan Maggesi, Mario Sicuranza, Qiqi Su, Alberto Acebes et al. "Overview of the SMART-BEAR technical infrastructure". In *Proceedings of the 8th International Conference on Information and Communication Technologies for Ageing Well and e-Health-ICT4AWE*, pp. 117-125. SciTePress, 2022, doi: 10.5220/0011082700003188.

In addition to the publications above, we have made the code of FocusLearn freely available at https://github.com/qisuqi/FocusLearn/tree/main. The earlier two publications are review papers produced as part of the broader research project – SMART BEAR, that motivated the initial direction of the proposed model, as well as the application of it to the Hearing Loss scenario explored later in Chapter 5. While not the primary focus of this work, they form part of the background context.

1.6 Thesis Outline

This thesis is structured into five chapters, each addressing distinct aspects of the research. **Chapter 1** provides an introduction to the study, outlining the background, research aims, and the structure of the thesis. **Chapter 2** re-

views the existing literature on time series prediction models and eXplainable AI (XAI) techniques, identifying key gaps and motivating the need for the proposed research. **Chapter 3** details the methods employed, including the design and implementation of various predictive models, the integration of post-hoc and ante-hoc explanation methods, and an interpretable time series model. **Chapter 4** presents the results, discussing the architectures of the developed models, their performance in prediction, and the outcomes of the explanation techniques, along with a comparative analysis. Finally, **Chapter 5** concludes the thesis by summarising the key findings, contributions to the field, limitations, and suggestions for future research.

Chapter 2

Related Work

> "If we knew what it was we were doing, it would not be called research, would it?"
>
> - Albert Einstein

In this section, we present a comprehensive review of the related literature in two main areas of this research: time series forecasting and classification models and time series eXplainable Artificial Intelligence (XAI) methods. The discussion for time series forecasting and classification models spans from traditional statistical techniques to machine learning (ML) algorithms and deep neural network (DNN) methods. We delve into the foundational principles, strengths, and limitations of each approach. Additionally, we categorised XAI methods that are applicable to time series techniques into post-hoc and ante-hoc, discussing the advantages and disadvantages inherent in these XAI techniques and shedding light on their utility in enhancing model interpretability and transparency.

2.1 Time Series Forecasting and Classification

Time series forecasting can be generalised as a process for predicting future values by extracting useful information from historical values and patterns. Thus, it can be treated as a standard regression problem but with time-varying features, such as the values of the features change over time, either continuously or at discrete intervals. The goal of solving a time series forecasting problem is to find a function $f(x)$, where $x = y_1, y_2, ..., y_n$, so that $\hat{y_{n+1}} = f(x)$ is as close to the ground truth y_{n+1} as possible. In terms of time series classification, the task shifts from predicting future values to assigning labels to sequences or subsequences by extracting useful information from its observed values and patterns.

Many different approaches have been proposed over the years to solve time series forecasting problems, with some of the earliest techniques dating back to the 1940s. These approaches can be grouped into three general categories: classical statistical, ML, and DNN methods. These time series forecasting models for each of the method are discussed as follows.

2.1.1 Statistical Methods for Time Series

One of the most commonly used statistical methods for analysing multivariate time series is Vector Auto Regression (VAR). VAR is an extension of the AR process, which focused only on univariate time series. In the AR process, the future values are linear combinations of the past values. To extend AR to the multivariate case, each variable is a linear combinations of the past values of itself and the past values of all other variables in the data. This way, the interactions and dependencies between several variables can be captured by the VAR, hence improving the understanding of the dynamic behaviour of the time series [35]. Besides being a well-known tool in the economics domain, VAR has been applied to regression tasks such as forecast COVID-19 cases and deaths [114] and electronic medical diary data [152], and classification tasks such as classifying multichannel signal [55] and cardiac arrhythmia classification [48].

Hidden Markov Model (HMM) [12] is another commonly used statistical model for time series forecasting and classification that derives from the theory of Markov chain. An HMM is formally defined to be a doubly stochastic process with a hidden underlying stochastic process that can only be observed through another stochastic process that produce the sequence of observed symbols [111]. HMM is one of the earliest techniques for analysing time-dependent data sequences. It was first applied in the area of speech processing [111] and has found success in many other practical domains such as stock trading [104], alignment of bio-sequences [56], and road collision prediction [154].

2.1.2 ML Methods for Time Series

Support Vector Machine (SVM) [36] and Random Forest (RF) [16] are two widely used ML models in time series forecasting. Originally developed to solve classification problems, SVM has been extended to solve regression and time series forecasting problems, such as crude oil price forecasting [158], stock forecasting [80], traffic flow [59], heartbeat classification [72]. The fundamental idea behind a SVM is that the model maps the input vectors with some non-linear mapping functions chosen *a priori* into some high dimensional feature space, and then constructs a linear decision boundary with special properties. Specifically, it maximises the margin, which is the distance between the decision boundary and the closest data points (known as support vectors) to ensure the high generalisation ability of the model [36].

RF is a combination of decision trees, with all trees having the same distribution in the forest and each tree depends on the value of an independently sampled random vector. Similar to SVM, RF can also be used for both classification and regression problems. In regression problems, the final output of a RF is the mean of the outputs of each tree. In classification problems, the outcome of a RF is the class that receives the most votes or has the highest probability. Diamond price prediction [129] and exchange rate [115] are some domains in which RF is used for time series forecasting, and breast lesions classification [141] and crop classification [138] for time series classification.

eXtreme Gradient Boosting (XGBoost) [27] is another decision tree ensemble algorithm, similar to RF, and it is based on the stochastic gradient boosting algorithm. The inherent stochastic boosting algorithm in XGBoost allows it to handle and capture the complex non-linear relationships between features and the target variable. Due to the fact that XGBoost employs decision trees as base learners, it is also capable of handling missing data without any pre-processing of the data [27]. During the tree construction process, XGBoost can assign missing values to the most appropriate splitting based on the information gain criterion, effectively utilising missing data without requiring imputation or removal techniques. The decision tree structure of XGBoost also allows it to be flexible in solving several different ML problems, including time series forecasting and classification. Tree-based models can handle both regression and classification problems, allowing XGBoost to adapt to different ML problems by simply changing the loss functions. Furthermore, the decision tree structure of XGBoost can natively handle missing data during training by automatically learns the best split to take when encountering missing data. Some time series problems using XGBoost include stock market forecasting [149], Covid-19 transmission [45], Parkinson's disease classification [2], and driving risk assessment [131].

The main advantage of SVM over RF is that it tends to have a better generalisation and avoids overfitting, due to the fact that it implements the structural risk minimisation principle (SRM) [80]. SRM focuses on minimising an upper bound on the generalisation error rather than solely minimising the training error. While SRM is a concept that can be applied in various ML models, SVM explicitly integrates SRM into its framework by maximising the margin between classes and optimising its complexity to balance bias and variance. However, as XGBoost employs learning rate shrinkage and tree pruning regularisation techniques, it has been shown that XGBoost can improve generalisation performance and avoid overfitting even further.

2.1.3 DNN Methods for Time Series

DNN time series models overcome the challenge of linear statistical models [166] and outperform statistical models with predicting non-linear data [136], since no assumptions are made by the DNN models about the statistical distribution of the underlying time series [74]. There is also no restriction on the uses for regression or classification problems.

2.1.3.1 RNNs

Recurrent Neural Network (RNN) [43] are very popular for analysing multivariate time series problems due to their ability to remember past inputs and use them to inform future decisions. Many variants have been proposed over the years, like Long Short-Term Memory (LSTM) [58], Bidirectional LSTM (BLSTM) [123], and Gated Recurrent Unit (GRU) [31]. LSTM and GRU were introduced to overcome the gradient vanishing problem faced by an RNN [13], and in turn have shown to be useful for learning long-term dependencies [34]. Since LSTM has a more complex architecture than GRU, training a GRU requires fewer computational resources. On the other hand, LSTMs have shown to be more capable than GRU in handling large amount of data and have more parameters that can be adjusted to optimise their performance, making them a popular choice for analysing multivariate time series problems across a variety of real-life applications [61, 63, 121, 136].

2.1.3.1.1 LSTM

In the hidden layer of an LSTM network, there are some special units called memory cells that are recurrently connected, as well as their corresponding gate units, namely input gate, forget gate, and output gate [58], as shown in Figure 2.1. The input gate is responsible for preventing the memory stored in a memory cell from perturbations by irrelevant inputs. Similarly, the output gate is there so other units are protected from perturbations by currently irrelevant stored memory. To optimise the performance of the LSTM, information that is no

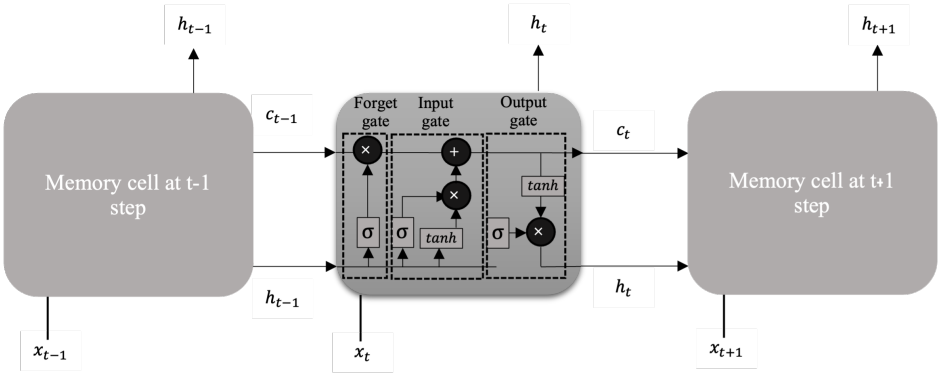


Figure 2.1: **An illustration of the LSTM network**.

longer required by the LSTM is removed in the mechanism of the forget gate.

At each timestep t, the cell takes an input vector, x_t, and produces an output vector, h_t, which also refers to the hidden state of the LSTM. Firstly, the cell needs to determine whether the information from the previous timestep, $t-1$, should be kept or not with the forget gate, f_t. The forget gate takes the input vector at current timestep, a_t, and the hidden state from the previous timestep, h_{t-1}, and produces an output between 0 and 1 where 0 represents "completely forget this information" and 1 represents "completely keep this information". Then the importance of the new information is quantified with the input gate, i_t. Then, a new vector named s_t is created which decides if the new information should be stored in the cell state or not. This is done by applying a hyperbolic tangent function, tanh, to the input vector at current timestep, a_t, and the hidden state from the previous timestep, h_{t-1}. The value of new information is then transformed to a value between -1 and 1, where -1 means the new information is subtracted from the cell state and 1 means the new information is added to the cell state. The current cell state, c_t, is finally updated by taking the previous cell state, c_{t-1}, the forget gate, f_t, the input gate, i_t, and s_t into consideration. Then, the output gate, o_t, determines what information from the cell state is going to be the output. Finally, the hidden state, h_t, at timestep t is updated with the current cell state, c_t, and the output gate, o_t.

2.1.3.1.2 Attention-based LSTM

In order to further enhance the performance of LSTMs, many researchers have proposed a variety of methods. In particular, the combination of LSTMs and attention mechanisms has become increasingly popular, as it is often able to provide better results, especially for sequence problems [89]. With an Encoder-Decoder architecture, Bahdanau *et al.* [11] first introduced the attention mechanism in a Neural Machine Translation task. The use of attention mechanisms in conjunction with encoder-decoders or LSTMs can prevent the model from putting too much weight on certain input features, as well as enable the model to obtain correlations between input and target features [155]. The attention-based LSTM architecture, initially tailored for NLP tasks, has emerged as a powerful tool and found extensive application in the realm of time series analysis [29, 86, 168]. This extension underscores the adaptability and versatility of attention-based LSTMs, demonstrating their effectiveness beyond their original linguistic context.

The architecture of attention-based LSTM models is largely consistent across various studies [29, 86, 168], with the structure involving the use of an attention mechanism to identify which features or time steps should receive more weights, followed by an LSTM to learn the corresponding attention weights and make predictions. This general framework allows for effective sequence modelling by dynamically weighing input elements based on their relevance. Where these models differ is in the specific type of attention mechanism employed. For instance, a standard attention mechanism was employed in the work of Cheng *et al.* [29], Liu *et al.* [86] proposed to adopt an evolutionary attention mechanism to discover the pattern for importance-based attention sampling during temporal relationship mining, while Zhang *et al.* [168] introduced an attention mechanism based on a deterministic attention framework, offering an alternative to the conventional stochastic (attention mechanism where the process of selecting which parts of the input data to focus on is probabilistic) or soft attention mechanisms.

2.1.3.1.3 Encoder-decoder

The Encoder-decoder system has been a popular tool for a range of task including those with a temporal element such as machine translation [30], text summarisation [83], and image captioning [145]. In the context of time series forecasting, the encoder processes historical time series and compresses them into a fixed-length context vector, capturing the important patterns and dependencies in the data. The decoder then uses the context vectors to generate and forecast future values, step by step. Most of the encoder-decoder systems used for time series forecasting are RNN-based [41, 82, 164], so that RNN variants such as LSTM, BLSTM, and GRU can still be used to learn the temporal information in the data.

2.1.3.1.4 Transformer

Transformers architecture [144] is based on the encoder-decoder architecture, where both encoder and decoder consist of multiple layers of self-attention and position-wise feed-forward networks. The self-attention mechanisms allow the model to weigh the importance of different parts of the input sequence (for the encoder) and the output sequence (for the decoder). The self-attention mechanism also allows the transformer to be highly parallelisable as multiple parts of the input sequence can be processed at the same time, as well as capable of handling long-term dependencies in the data. However, transformers are computationally expensive due to their size and complexity. Also, despite having a self-attention mechanism that can handle long-term dependencies, transformers do not inherently capture the temporal information in the data [162].

2.1.3.1.5 Attention-based Encoder-decoder

To overcome these limitations of transformers, integrating an attention mechanism into an RNN-based encoder-decoder system has gained traction [41, 146, 160, 171]. Using attention mechanisms with encoder-decoder systems in time series analysis allows the model to focus on parts of the input sequence that are most relevant in predicting future values. Also, the attention mechanism allows

long-term dependencies and subtle patterns in the data to be captured when combining with an encoder-decoder system, making this combination useful in capturing irregular patterns, trends, and seasonal variations in real-world time series data. Different attention mechanisms have been proposed to integrate into an encoder-decoder system, the two most popular ones are global attention [160] and self-attention [146, 171].

Similar to attention-based LSTMs, the architecture of attention-based encoder-decoder models remains largely consistent across different studies [146, 160, 171], typically integrating an attention mechanism between the encoder and decoder. This architecture enables the decoder to selectively focus on relevant parts of the encoded input sequence when generating output predictions. For instance, Du *et al.* [41] proposed a temporal attention mechanism to be incorporated between a BLSTM encoder and an LSTM decoder, allowing for a time-sensitive focus on the encoded information. In contrast, Yuan *et al.* [160] utilised a global attention mechanism between an LSTM-based encoder and decoder, which provides a broader perspective across all input time steps. Meanwhile, Violos *et al.* [146] and Zhou et al. [171] implemented a more complex architecture featuring two self-attention mechanisms within both the LSTM-based encoder and decoder, enhancing the model's ability to capture intricate dependencies. These variations illustrate the flexibility of the encoder-attention-decoder framework in adapting to different forecasting needs.

2.1.3.2 Attention Mechanisms

2.1.3.2.1 Self-attention

Self-attention [144] aims to relate different positions of an input sequence in order to compute the representations of the same sequence and weigh the importance of different parts of the input sequence when making a prediction. To do this, the query vector, $\boldsymbol{Q}$, which is a vector of specific element(s) in the input sequence, is compared with all other elements (the key vector, $\boldsymbol{K}$) in the input sequence. For each $\boldsymbol{K}$, its corresponding value vector, $\boldsymbol{V}$, contains information used to weigh the importance of difference parts in the input sequence. Self-attention, by default,

does not include positional information and treats all sub-sequences as identical individuals and loses the temporal order information [130]. Therefore, position encodings are often dependent on models such as transformers to provide some sense of the feature order. Since self-attention attends to all parts of the input sequence, it is able to capture dependencies between representations regardless of their distance in the sequence.

Given an input, $X = (x_1, x_2, ..., x_n)$, the self-attention transforms X into three vectors, Query ($\boldsymbol{Q}$), Key ($\boldsymbol{K}$), and Value ($\boldsymbol{V}$), using linear projections such that:

$$\boldsymbol{Q} = xw_Q, \quad \boldsymbol{K} = xw_K, \quad \boldsymbol{V} = xw_V \tag{2.1}$$

where w_Q, w_K, and w_V are learnable weights during the training process. The compatibility score is then calculated by taking the dot product of $\boldsymbol{Q}$ and $\boldsymbol{K}$ and scaled by the $\boldsymbol{V}$ vectors. Finally, softmax is applied to the attention outputs to normalise the compatibility scores across all $\boldsymbol{K}$ vectors as in Equation (2.2).

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{d_k}})\boldsymbol{V} \tag{2.2}$$

Huang *et al.* [62] designed a time series forecasting network, named DSANet, which showed that self-attention can be effective at capturing dependencies between multiple time series. Informer [171] is a network for long-sequence time series forecasting and it has shown that self-attention can efficiently handle extreme long input sequence.

2.1.3.2.2 Global attention

The difference between global attention and self-attention is that global attention allows elements in a sequence to be compared with other elements in a different, external sequence, rather than within the same sequence as in self-attention. The idea of global attention was introduced by Luong *et al.* [95] in a machine translation task with an encoder-decoder system. Global attention is similar to

the first ever attention mechanism (soft attention) proposed by Bahdanau *et al.* [11], such that all hidden states in the encoder are considered and the context vectors are computed as a weighted average over all hidden states in the input sequences given the compatibility score as the weight. The key difference between global and soft attention is that global attention also takes all hidden states in the decoder into consideration, whereas soft attention only considers the previous hidden states in a uni-directional decoder. Since global attention considers all hidden states in both the encoder and the decoder, it allows models to access information from beyond the immediate sequence, enhancing their capacity to deal with tasks such as summarisation [153] and machine translation [26].

In time series domain, Yuan *et al.* [160] adopted a global attention with an encoder-decoder architecture for a real-time disturbance detection problem. They concluded that the high prediction accuracy of their proposed model can be attributed to the introduction of the global attention, as it allows the proposed model to extract the most relevant historical information at each prediction timestep [160].

2.1.3.2.3 Multi-head attention

Vaswani *et al.* [144] also introduced the idea of a Multi-head Attention mechanism that is capable of jointly focusing on parts of the input sentences with attention from different representation sub-spaces. In multi-head attention, each attention head is a self-attention that attends to an input sub-sequence as a way to reduce the computational complexity and $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ can be linearly projected with different projections to d_q and d_v dimensions, respectively. By having several self-attention as attention head in multi-head attention, each attention can focus on different parts of the input sequence simultaneously and gives the model richer representations by capturing different attention patterns.

Muformer [163], a long sequence time series forecasting model uses a multi-head attention to prune abundant information and enhances the model expression due to the fact that a multi-head attention can capture more characteristic information than a single head attention mechanism. In the work of [159], it was also

found that a multi-head attention can enhance the prediction ability because it can learn the long-term relationship of data at different levels of detail.

2.2 Explainable AI

ML models and DNNs are increasingly used for a variety of tasks, including natural language processing and computer vision. However, there is now widespread recognition that users demand more from DNNs than just effective predictive performance in real life domains. They also need to be interpretable/explainable, especially in high-impact domains, such as healthcare. As a result, the development of explainable AI (XAI) is gaining traction, and many sectors rely on XAI systems for explainable and interpretable ML models. XAI is an umbrella term for any research that is trying to explain black-box models and seeks to make DNNs more transparent and understandable to humans, which can provide greater trust in AI decisions. XAI methods provide an explanation of why a certain decision was reached by a model, which can help decision-makers make more informed decisions and identify potential biases in the models.

XAI techniques can be categorised into ante-hoc and post-hoc ones. Ante-hoc interpretability can be achieved by creating self-explanatory interpretable models [40] such that interpretability is already built-in since the creation of the model [18]. A post-hoc method on the other hand aims for DNN model explanation, thus retaining the possibility of high prediction accuracy. In a post-hoc method, interpretability is created after the model creation [18] and can be further divided into two categories: model-agnostic and model-specific. Model-agnostic methods can be applied to a variety of models, whereas model-specific methods can only be applied to one type of model.

In the context of Explainable AI (XAI), the terms *understanding, interpreting*, and *explaining* are often used interchangeably [39]. However, there is a growing consensus in the literature that interpretability and explainability, while closely related, have distinct meanings. Some works do not explicitly differentiate between them [96], whereas others emphasise a fundamental distinction. According

Table 2.1: **An overview of XAI methods on time series**.

Model	Approach	Methodology	Scope	Time-series Specific?	Learning Task	Ante-hoc / Post-hoc	Data Type	Explanation Type	Explanation Outcome
N-Beats [108]	Interpretable Models	Decomposition	Global	Yes	Regression	Ante-hoc	Univariate	Decomposition into trend and seasonality	Forecast graph
ERNN [32]	Interpretable Models	Attention Mechanism	Local	Yes	Classification	Ante-hoc	Multivariate	Feature importance analysis	Feature importance
RNN-LSTM [49]	Interpretable Models	Attention Mechanism	Global	Yes	Classification Regression	Ante-hoc	Multivariate	Feature importance weights	Feature importance
SAX-VSM [124]	Interpretable Models	SAX	Global		Classification	Ante-hoc	Multivariate	Time-series pattern importance	Class decision
Neuro-fuzzy [109]	Interpretable Models	Fuzzy Logic	Global	Yes	Regression	Ante-hoc	Multivariate	Rule-based representation	Linguistic rules
FCM [148]	Interpretable Models	Fuzzy Logic	Global	Yes	Regression	Ante-hoc	Multivariate	Graph-based causal relationships	Feature interactions
NAM [4]	Interpretable Models	Feature Attribution	Global	No	Classification Regression	Ante-hoc	Univariate Multivariate	Direct interpretation via shape function	NAM-styled graph
SPAM [42]	Interpretable Models	Additive	Global	No	Regression Classification	Ante-hoc	Univariate Multivariate	Decomposition of polynomials	Higher-order feature interaction
NATM [71]	Interpretable Models	Additive	Global	Yes	Regression Classification	Ante-hoc	Univariate Multivariate	Feature contribution map	Feature importance
HNAM [47]	Interpretable Models	Additive	Global	Yes	Regression Classification	Ante-hoc	Univariate Multivariate	Hybrid explanations	Partial feature interaction
PAA [46]	Interpretable Models	Shapelets	Global	Yes	Classification	Ante-hoc	Multivariate	Time-series shaplets	Class decision
FCN-CAM [150]	Explanation Generation	Backpropagation-based	Local	Yes	Classification	Post-hoc	Multivariate	Feature importance contribution	Feature importance
DeepLIFT [132]	Explanation Generation	Backpropagation-based	Local Global	No	Classification	Post-hoc	Multivariate	Feature importance contribution	Feature importance
ConvTimeNet [73]	Explanation Generation	Perturbation-based	Local	Yes	Classification	Post-hoc	Multivariate	Convolution-based relevance maps	Feature importance
CoMTE [9]	Explanation Generation	Perturbation-based	Local	Yes	Classification	Post-hoc	Multivariate	Counterfactual	Feature-relevance map
LIME [116]	Surrogate Model Fitting	Feature Attribution	Local	No	Classification Regression	Post-hoc	Univariate Multivariate	Local surrogate-based	Feature importance
Kernel SHAP [94]	Surrogate Model Fitting	Feature Attribution	Local	No	Classification Regression	Post-hoc	Univariate Multivariate	Additive feature attribution	Feature Importance
AFS [53]	Explanation Generation	Attention Mechanism	Local Global	No	Classification Regression	Post-hoc	Univariate Multivariate	Feature selection	-
MFS [21]	Explanation Generation	Attention Mechanism	Local Global	Yes	Classification Regression	Post-hoc	Univariate Multivariate	Feature selection	-

to Rudin, "interpretable ML focuses on designing models that are inherently transparent, whereas explainable ML provides post-hoc explanations for black-box models, which are either too complex for human comprehension or proprietary in nature" [120]. Despite such definitions, there is little agreement on a universal distinction, as the interpretation of these terms is often application-specific.

In this work, we adopt a practical perspective aligned with our specific domain. We categorise interpretable models as ante-hoc and explainable models as post-hoc, following Rudin's framework. Interpretability, in our case, means that the model itself provides inherently understandable outputs – specifically, through NAM-styled explanation graphs. These graphs visualise the learned relationships between input features and predictions, allowing direct human interpretation without the need for external explainers. In contrast, explainability is achieved through surrogacy, where additional post-hoc techniques, such as feature attribution (*e.g.,* SHAP, LIME) or counterfactual reasoning, are used to analyse model behaviour after training.

While these distinctions apply broadly to machine learning, explaining time-series models presents unique challenges. Time-series data are inherently sequential, often contain temporal dependencies, and may exhibit non-stationarity, which complicates interpretability. Unlike tabular or image data, where individual features or pixels are typically independent, time-series explanations must account for temporal relationships and dynamic patterns. Several design choices influence the explainability of time-series models. One critical decision is whether to use global explanations, which provide insights into the overall model behaviour, or local explanations, which focus on individual predictions. Another consideration is the choice of explanation method: feature attribution techniques (*e.g.,* SHAP, LIME) highlight important input values, while sequence-based methods, such as attention mechanisms in deep learning models, inherently provide interpretability by assigning weights to different time steps.

A summary of key XAI methods applicable to time-series models is provided in Table 2.1.

2.2.1 Post-hoc XAI Methods

Due to the possibility of post-hoc methods of retaining the high accuracy of a DNN, such methods such as Local Interpretable Model-agnostic Explanation (LIME) [116] and SHapley Additive exPlanation (SHAP) [94] may be useful to overcome the accuracy-interpretability trade-off challenges of ante-hoc approaches. However, these methods suffer from low fidelity [3], instability [50], and even inaccuracy [88, 120], since they only approximate their underlying model.

2.2.1.1 Feature Attribution Methods

LIME [116] was originally designed for classification problems in the domains of computer vision and Natural Language Processing that explains individual model predictions by approximating the model locally around the prediction. SHapley Additive exPlanations (SHAP) [94] is another example of the Feature Attribution method that utilises Shapley values [127] from game theory and assigns each

feature an importance value for a given prediction to explain the outcome of a model. Lundberg et al., [94] also proposed different versions of SHAP, where Kernel SHAP (Linear LIME + Shapley Values) is the model-agnostic approximation and Linear SHAP, Max SHAP, Low-order SHAP, and Deep SHAP (Deep LIFT + Shapley Values) are the model-specific approximations. Since LIME and Kernel SHAP are model-agnostic XAI methods, they have both been applied to time series, where LIME was employed in time series health-risk classification [68] and Kernel SHAP has been used to explain financial time series classification [100].

2.2.1.1.1 SHAP

SHAP [94] assigns importance to each feature according to the Shapley values from Game Theory and aims to explain the predictions by computing the contribution of each feature to the model predictions. Mathematically, the SHAP explanations can be written as follows:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{2.3}$$

where g is the explanation model of the prediction model, $z' \in 0, 1^M$ where z' is the binarised feature and M is the number of binarised input features, ϕ_0 is the model output without binarised inputs, and $\phi_i \in \mathfrak{R}$ are the Shapley values [94]. SHAP is a local XAI method meaning that the method is designed to explain only the model prediction on a single data instance. However, it is also possible to obtain a global explanation with SHAP through aggregation, by calculating the mean absolute SHAP values for each feature across the data set. Therefore, the relative impact of all features over the entire data set and the global importance of each feature can be determined.

SHAP also satisfies three important properties: Local Accuracy, Missingness, and Consistency [94]. Local accuracy requires consistency between the outputs of the explanation model and the prediction model. Missingness requires features missing in the original input to have no impact on the output. Lastly, consistency ensures that the impact of a feature does not decrease as it increases or remains

the same. Local accuracy is particularly important for providing explanations, as it ensures that the explanation model is less susceptible to adversarial attacks [69].

However, SHAP is often criticised for long computation time, especially for complex underlying model or high-dimensional feature space. This is because as number of features grows, the possible combinations of features grows exponentially. It was found that the computational complexity would become intractable for more than ten features [1]. Due to this, Lundberg *et al.* [94] proposed Kernel SHAP that uses LIME to approximate the underlying model to speed up computation time. However, one major limitation of Kernel SHAP is that it assumes linearity and feature independence [1]. But, feature independence rarely happens in real-life data. By assuming feature independence, SHAP suffers from inclusion of predictions based on unrealistic data when features are correlated [98], as well as producing misleading explanations with highly correlated features [101]. It was later then found that SHAP does not just produce misleading explanations with highly correlated features, it can do so with all features, and there is a surprising flexibility in choosing the influence of each features [97]. The experiments conducted by Marques-Silva and Huang [97] show that it is easy to attain misleading explanations from SHAP, and found that the predictions for data points in the feature space that are unrelated with the given instance can have a critical impact on the relative order of feature importance.

2.2.1.1.2 LIME

Given an instance, the decision of the black box model is approximated by LIME linearly in a local neighbourhood of the instance [116] . The explanation produced by LIME is obtained by,

$$\xi(x) = \underset{g \epsilon G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g), \tag{2.4}$$

where $g \in G$ is defined to be an explanation as a model and G is a group of potentially interpretable models, $\Omega(g)$ is the complexity measure of the explanation $g \in G$, $x \in \mathbb{R}^d$ is the original representation of an instance to be explained, $f(x)$ is the probability that x belongs to a certain class in a classification problem, $\pi_x(z)$ is a proximity measure between an instance z to x, and finally $\mathcal{L}(f, g, \pi_x)$ is the fidelity measure of how unfaithful g is in the locality defined by π_x when approximating f [116].

As a model-agnostic method, LIME minimises $\mathcal{L}(f, g, \pi_x)$ without making assumptions about $f(x)$. This means that LIME can be used to explain any underlying model. Therefore, $\mathcal{L}(f, g, \pi_x)$ is approximated, weighted by π_x, by sampling instances around the binary vector of x, $x' \in \{0, 1\}^d$, instead of x in order to learn the local behaviour of f as inputs vary. The families of different potentially G include decision trees, linear models, and rule-based models. Thus, the complexity measure $\Omega(g)$ could be the depth of the tree for decision trees or the number of non-zero weights for linear models [116].

Simply put, LIME functions by perturbing or tweaking a few instances of the original input data and observing the corresponding changes in the model predictions. Subsequently, a simple and interpretable surrogate model is trained on these perturbed instances to approximate the behaviour of the original model in the local vicinity of the instance of interest. The predictions of this surrogate model then serve as explanations for the specific decision made by the complex model.

While LIME serves as a valuable tool for unveiling the decision rationale behind individual predictions, it inherently lacks the capacity to provide global explanations comparable to methods like SHAP. This limitation arises from the focus of LIME on providing insights at a local level, rendering it less effective in capturing the overall behaviour of a complex model across the entire data set. One of the challenges associated with LIME is the necessity to define a meaningful neighbourhood around the instance of interest for sampling perturbed data. The difficulty in precisely delineating this neighbourhood often results in subjective choices, introducing potential bias into the explanations generated [133], hence compromising the trust and transparency issues. Additionally, the explanations

provided by LIME may exhibit instability, as demonstrated by [6], who illustrated significant variations in explanations for closely located instances. Such variability undermines the reliability of the interpretability of LIME.

2.2.1.2 Backpropagation-based Saliency Methods

Saliency methods are used to highlight the importance or relevance of input features to predictions made by a model [67]. One of the Backpropagation-based Saliency methods uses Class Activation Mapping to highlight the sub-sequences in the input time series data that have the greatest influence on the output classification prediction of the model. Wang *et al.* [150] applied this method in their experiments with Convolutional Neural Network (CNN) on over 80 data sets from the UCR time series classification data set repository [1]. Another Backpropagation-based Saliency method is Deep Learning Important FeaTures (DeepLIFT) [132], which assigns contribution scores to each neuron according to the difference between its activation and a "reference activation". Lundberg *et al.* [94] adapted DeepLIFT into SHAP, named Deep SHAP, to take advantage of the compositional nature of DNN models to improve computational performance. Ismail *et al.* [67] later compared the performance of Deep SHAP with several other saliency-based methods using RNN, Temporal Convolutional Network, and Transformers in a multivariate time series classification problem. Their results have shown that saliency-based methods with DNN architecture generally fail to identify feature importance over time reliably and accurately in time series data; if a feature is assigned to high saliency in a given time step, then almost all other features tend to have high saliency in that time step regardless of their actual values [67].

2.2.1.3 Perturbation-based Methods

The main concept behind a Perturbation-based method is to remove, mask, or alter input features and compute their contribution to the prediction by running a forward pass on the newly transformed inputs and measuring the difference with the original input [117]. ConvTimeNet [73] uses several CNN blocks to learn the

[1] `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`

time series, then occludes a sub-sequence of the time series by using the occlusion sensitivity method to observe the changes in the outcome for the predicted classes.

Another Perturbation-based method is counterfactual: "why event P happened instead of event Q". Counterfactual methods delve into causal inquiries by exploring scenarios where altering certain features of the input data would lead to different model predictions. In the context of interpretable machine learning, counterfactual explanations identify the smallest changes to feature values necessary to transform a prediction to a predefined output. For instance, in a credit card application scenario, if Mr. A, a 30-year-old individual with a £30k annual income, receives a rejection, the generated counterfactual explanations could indicate that an annual income of £35k might have led to an acceptance.

White and Garcez [151] proposed Counterfactual Local Explanations viA Regression (CLEAR), a method to explain the predictions of any classifiers. CLEAR is built on top of two explanatory methods: boundary counterfactual (*b*-counterfactual) and LIME, where *b*-counterfactual states "the minimum changes needed for an observation to flip its classification". Counterfactual Multivariate Time Series Explanation (CoMTE) proposed by Ates *et al.* [9] aims to provide a counterfactual-based explanation specifically for multivariate time series classification problems by finding a minimal number of time series substitutions that will flip the prediction. In theory, Perturbation-based methods can be applied to both classification and regression tasks, as long as it is possible to compute the distance values between the different outputs of the model [117].

2.2.1.4 Attention Mechanism Methods

Attention Mechanism methods can be used to explain a RNN by assigning corresponding values to the importance of the different sub-sequence of the time series according to the model. Choi *et al.* [32] proposed an Explainable Recurrent Neural Network (ERNN) for classifying genotypes. ERNN is an ensemble model that combines a CNN and an LSTM, where CNN acts as the feature extractor and LSTM learns the temporal dependencies. Next, the LSTM is used to produce a hidden state at each time step and the attention weights, or the importance, are

assigned to each hidden state by running a Feed-forward Neural Network (FNN) attention function [32]. Finally, another FNN is stacked on top of the ensemble model to make the classification by using the weighted sum of the hidden states, output of weights of the attention function, and a context vector as inputs.

Ge *et al.* [49] proposed an RNN-LSTM-based architecture for ICU mortality classification, interpreting feature importance directly from learned weights. Schockaert *et al.* [122] extended this approach to multivariate time series forecasting in the iron-making domain, introducing a dynamic temporal attention mechanism applied to LSTM hidden states. The attention weights were derived from dot products of LSTM outputs, enabling local temporal interpretability through context vectors. Pantiskas *et al.* [110] focused on regression tasks by combining a temporal convolutional network (TCN) with attention, using attention maps to visualise the model's focus on input series. However, these models are typically designed and evaluated for either classification [32, 49] or regression tasks [110], but not both.

While it is true that neural networks can technically support both classification and regression tasks by altering the output layer and loss function, in practice, many existing interpretable time series models are tightly coupled to a single task type. Their design and evaluation pipelines are specialised – for example, classification-focused models often rely on discrete label interpretation mechanisms, whereas regression models use continuous forecasting structures.

2.2.2 Ante-hoc XAI Methods

2.2.2.1 Shaplets Methods

Shaplets is a XAI method that explains the models through representative examples and is specific to the time series problems. Shaplets, which are time series sub-sequences that are maximally representative of a class, were first introduced by Ye *et al.* [156] to address the limitations that were present in the time series classification problems, such as the ability to gain insight into the time series data beyond classification accuracy and overcoming the time and space complexity in

nearest neighbour algorithms. One limitation here is that finding the optimal shaplets in the raw time series space using brute force can result in a long computation time. To address this, Fang *et al.* [46] proposed an interpretable time series classification framework that searches the shaplet candidates from the Piecewise Aggregate Approximation (PAA) word space in order to reduce the computation costs.

2.2.2.2 SAX – Symbolic Aggregate approXimation Methods

PAA is an example of the structure-based time series classification approach, where the classification model is built on top of the approximate representation of the raw time series. This approximate representation is created by dividing the time series into segments and summarising each segment with a single value, reducing the complexity of the raw data while maintaining important patterns for classification. Another example is Symbolic Aggregate approXimation (SAX) that transforms raw time series input into strings. In fact, SAX converts the time series into a PAA representation first, and then translates the representation into strings [117]. Senin and Malinchik [124] proposed to use SAX and Vector Space Model (VSM) for a faster, more precise, and interpretable time series classification framework. In their framework, SAX-VSM, real-valued time series are transformed into combined collections of SAX words. Then a VSM based on term frequency-inverse document frequency, $tf * idf$, weighting scheme is used to translate the collections into class-characteristic weight vectors based on the observed frequency of a term in a class. Finally, these weight vectors are used in classification by using Cosine similarity.

2.2.2.3 Fuzzy Logic Methods

Fuzzy Logic is an approach that is based on "degrees of truths" rather than just "True" or "False" as in the Boolean logic, using values between 0 and 1. Pavia and Dourado [109] developed a neuro-fuzzy system that combines the advantages of neural networks in terms of their learning capabilities with the transparency of fuzzy systems [161] in univariate time series forecasting. In particular, the neuro-

fuzzy system is based on the methodology of linguistically interpretable fuzzy models [109], so that each of the membership functions can be assigned with a linguistic label after training. Therefore, a set of linguistic rules can be used to describe the model obtained under the analysis. Wang *et al.* [150] proposed a novel extension of the Fuzzy Cognitive Map (FCM) named Deep FCM (DFCM) to perform multivariate time series forecasting. The DFCM system leverages the advantage of DNNs in prediction and FCM in interpretation.

2.2.2.4 Interpretable Additive Models

A recent interest in adding interpretability to Generalised Additive Models (GAM) [57] has demonstrated that they are capable of learning both time series prediction and classification problems. GAM variants, such as Neural Additive Model (NAM) [4], GA^{2}M [92], Neural Interaction Transparency [140], and Neural Basis Model [113], are computationally efficient and can handle large data sets.

2.2.2.4.1 NAM and its Extensions

NAMs [4] extend GAMs by leveraging neural networks to learn flexible, feature-wise shape functions while maintaining interpretability. Unlike traditional deep learning models, where features interact in complex ways, NAMs enforce feature-wise independence by assigning a separate neural network to each input feature. These individual networks are trained jointly using backpropagation, enabling the model to learn arbitrarily complex, non-linear relationships in a structured and interpretable manner.

Formally, given an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ with n features, NAM models the prediction function as in Equation (2.5), where each $f_i(x_i)$ is a neural network responsible for learning the contribution of a single feature x_i to the overall prediction. Since the outputs of these feature-wise subnetworks are combined via a simple summation (without learnable weights), the impact of each feature on the final prediction is independent of other features or any scaling factors.

$$f(x) = \sum_{i}^{n} f_i(x_i) \tag{2.5}$$

The impact of a feature on prediction can then be interpreted and visualised by plotting its corresponding shape function $f_i(x_i)$, providing an exact depiction of how NAM arrived at a prediction. NAM is also capable of learning non-linear relationships in the data, making it an attractive choice for many real-world applications. Unfortunately, the predictive performance of these models does not match that of more popular, non-interpretable time series models; for this reason, they cannot simply be regarded as a substitute for non-interpretable models [42].

The Scalable Polynomial Additive Model (SPAM) approach [42] builds on NAM by providing higher-order interactions between features and is intended to replace non-interpretable models for large scale data due to its high performance. The main difference between SPAM and NAM is that the intelligibility of SPAM arises from using tensor rank decomposition of polynomials, so higher-order feature interactions can be learned to provide an inherently interpretable model. However, the explanations provided by SPAM cannot be readily visualised in the same manner as NAM and, as such, they are harder to interpret than NAM.

Jo *et al.* [71] introduced Neural Additive Time-series Models (NATM), which is a modification of the NAM tailored specifically for time series data. NATM extends the NAM framework by incorporating multi-task learning, enabling the simultaneous prediction of multiple outputs in a multivariate time series context [71]. The importance of each feature in NATM is captured through a contribution map, providing insights into the model's decision-making process. Additionally, two parameter-sharing methods were introduced to reduce computational complexity, thereby enhancing the efficiency of NATM. Hierarchical Neural Additive Model (HNAM) [47] is another adaptation of NAM for time series, with a primary focus on capturing covariate interactions to better understand feature interdependence. HNAM employs a more intricate architecture, featuring several components designed to learn and model these complex interactions, including covariates embedding and projection, a multilayer perceptron (MLP) for non-linear processing, a temporal convolution layer to extract local features across

adjacent timesteps, and an attention block that contains level and coefficient networks. Each feature is processed in an individual level and coefficient network, as inspired by NAM.

Instead of modifying the architecture of NAM, many researchers have been exploring the idea of combining NAM with other methods. Utkin *et al.* [142] proposed SurvNam in the domain of survival analysis by combining survival model explanations [77, 78, 142] and NAM, as well as making modifications to the architecture of NAM. In SurvNAM, the loss function used to train NAM is changed to an expected loss function, which takes into account a function of the peculiarity of the output of the black-box survival model (*i.e.,* the peculiarity of the Cox extension model), as well as the fact that it takes the form of a convex loss function [142]. Furthermore, to avoid overfitting the sub-networks when computing the shape functions, either the Lasso method or the L_1-norm is used for regularisation. With SurvNAM, both local and global explanations can be obtained. Another modification was proposed by Kim *et al.* [75] named Higher Order Neural Additive Model (HONAM) aims to model the arbitrary orders of feature interactions as NAM cannot capture the co-relationship between features. The results have shown that by using their novel Feature Crossing for High Interpretability, HONAM has a higher or comparable prediction performance compared to MLP and XGBoost but still provides high quality interpretations.

2.2.2.5 Feature Selection with Attention Mechanism

Temporal Fusion Transformer (TFT) [87] was introduced recently for a multi-horizon forecasting task with added interpretability by visualising the persistent temporal pattern with attention mechanism. Another example is the Interpretable Temporal Attention Network (ITANet) [170] for COVID-19 forecasting and inferring the importance of government interventions. However, TFT faces a limitation that it needs a large amount of data to achieve a good predictive performance [165]; whereas ITANet is only able to provide feature importance as an explanation.

Feature selection has been demonstrated to be an effective approach for prepar-

ing high-dimensional data when performing a variety of machine learning tasks [53]. Several different architectures that incorporate an attention mechanism into DNNs for feature selection have been proposed recently, such as Attention-based Feature Selection (AFS) [53], Multiattention-based Feature Selection (MFS) [21], and TabNet [8]. Attention mechanism can learn the temporal relationship in the data, especially when used in conjunction with RNNs, better than traditional feature selection methods — wrapper feature selection methods tend to suffer from high computation complexity, while filter methods evaluate feature weights using only generic characteristics of the data [21, 53]. In AFS, several attention mechanisms are utilised in an "attention module" to select features directly from the input, then a "learning module" where different DNNs can be used to learn the selected inputs. MFS share a similar architecture with AFS, the main difference between the two is that two different attention mechanisms are employed in the "attention module" where one is learning the variable attention and the other one is learning the temporal attention in the inputs. However, in both AFS and MFS, interpretability only exists in the "attention module" and not the "learning module" where the actual learning takes place. TabNet is an interpretable sequential multi-step processing network and introduced a sequential attention mechanism for feature selection. Although TabNet achieved state-of-the-art results compared to non-interpretable DNN models, the explanations provided only considers feature attributions, and not the exact depiction of feature contribution as in NAM.

2.3 Modular Neural Networks

The idea of modularity has been researched as early as the 1980s and adopted in neural information processing in the late 1990s. It was recognised then [19] that traditional ANNs are "black-box" in nature, while Modular Neural Networks (MNNs) are more interpretable and explainable. An MNN follows the divide-and-conquer principle [143] and is inspired by the important biological fact that neurons in human brains are sparsely connected in a clustered and hierarchical fashion, rather than completely connected [19]. Instead of a monolithic network,

an MNN consists of multiple smaller NN modules, each responsible for learning a distinct part of the problem. The outputs of these modules are then combined to generate the final prediction for the entire network. In this way, using an MNN allows complex learning problems to be simplified [143], as well as to mimic "human thinking" to resolve these large-scaled complex problems [19].

Formally, given an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the predictive function of an MNN can be defined as in Equation (2.6), where $f_i(x_i)$ represents the output of the ith module processing a subset of the input features and g is a composition function that aggregates the module outputs into a final prediction.

$$f(x) = g(f_1(x_1), f_2(x_2), \ldots, f_n(x_n)) \tag{2.6}$$

MNNs are different to an ensemble approach, where a set of neural networks are combined, each of which is designed as a general function approximator [128]. In MNNs on the other hand, the problem is decomposed so that it is dealt with by several modular components with each modular component performing a different task [128]. Sharkey [128] also listed several reasons to why an MNN should be used. First, a modular system can achieve results which would not be possible with a single neural network since the capability of each module can be exploited. For example, Catfolis and Meert [22] combined a RNN and a feedforward network in order to incorporate temporal information with a more static approach. A modular system also offers flexibility to its designs, where a wide range of different techniques can be adopted in each of the module. For example, Kirsch *et al.* [76] proposed a training algorithm that can flexibly choose neural modules based on the data to be processed. Finally, with a modular approach, computational complexity can be reduced, hence make an MNN scalable and extensible to the large-scale MNN implementation [25, 128], and most importantly, making the overall system easier to understand [60, 128].

The relationship between modules in an MNN can be categorised into three ways: successive, supervisory, and cooperative [128]. In a successive relationship, the global task is decomposed into successive tasks where each task is carried out by a module [128], so that an MNN undergoes multiple stages of learning in a se-

quential way [25]. In the work of Bourlard and Morgan [15], a speech recognition model is designed where an ANN is used to compute the emission probabilities required in Hidden Markov Models. Modules in an MNN can also be in a supervisory relationship to each other [128]. For example in McCormack [99], one module is trained to select the parameters of a second module by observing the effect of various parameter values on the performance of the second module. A cooperative relationship is the opposite of the successive relationship, with the modules not implicated in sequential processing [128]. In a cooperative relationship, input could be processed with a number of modules, and then either the output of one module selected or the output of several modules are combined in some manner. All modules are trained jointly so all parameters of different NNs are updated simultaneously by minimising a loss function defined at the global level [25].

Due to the fact that an MNN is more interpretable than traditional ANNs and the overall system is easier to understand [19, 60, 128], many interpretable MNNs have been developed in recent years. MultiModN [137] is a multi-modal, multi-task, interpretable MNN for a number of predictive tasks. MultiModN uses task-agnostic encoders to produce an evolving latent representation to be queried by any number of combination of multi-task, model-agnostic decoder modules after each input [137]. MultiModN can also offer both local and global explanations by presenting individual modality contributions and cumulative contributions with a heatmap. Another interpretable MNN example is the Modular Clinical Decision Support Networks (MoDN) [139]. Similar to MultiModN, MoDN also adopts an architecture encoder and decoder, where the encoder updates the vector representation of the input data with the value of newly collected feature, while a diagnosis decoder extracts predictions from the vector representation at any stage of the consultation [139]. The diagnostic trajectories can be visualised to see how the predictions evolves.

2.4 Trade-off Between Predictive Accuracy and Interpretability

A fundamental challenge in XAI is the trade-off between interpretability and predictive accuracy. In many cases, models designed for high interpretability, such as additive models, inherently sacrifice predictive performance. Conversely, DNNs achieve state-of-the-art accuracy but require post-hoc explainability methods to justify their predictions. This trade-off presents a crucial gap in the development of XAI for time-series models.

In this work, we define interpretability as the model's ability to provide NAM-styled explanation graphs, which directly reveal the relationships between input features and outputs. NAMs and other additive models, such as GAMs, fall into this category. These models offer full transparency but struggle to capture complex interactions and dependencies in time-series data, leading to lower predictive accuracy.

On the other hand, deep learning models, including DNNs and RNNs, excel at capturing intricate patterns in sequential data. However, their inherent opacity requires the use of surrogate post-hoc explainability techniques such as SHAP, LIME, and attention mechanisms. While these methods help in understanding model predictions, they introduce an additional layer of uncertainty since explanations are derived rather than being an inherent property of the model.

Several strategies have been proposed to bridge this gap between interpretability and accuracy. **Self-explaining Neural Networks:** Models such as Self-Explaining Neural Networks (SENN) [5] and Explainable Boosting Machines (EBMs) [93] aim to balance interpretability and accuracy by enforcing structured, human-readable decision processes. **Regularisation Techniques:** Certain methods introduce constraints, such as sparsity penalties [119] or monotonicity constraints [105], to make black-box models more transparent. **Casual and Counterfactual Explanations:** Instead of feature attribution, causal inference and counterfactual reasoning generate explanations that align more closely with human intuition [37, 102, 147, 151].

However, none of these existing approaches provide NAM-styled explanation graphs, which are central to our definition of interpretability. As a result, rather than relying on post-hoc or regularisation techniques, we focus on modifying additive models, especially NAM, to improve their predictive power while maintaining their inherent interpretability. By enhancing the predictive accuracy of NAM, we aim to close the gap between interpretability and accuracy without sacrificing the transparency offered by NAM-styled representations.

2.5 Gaps in the Current Literature

As shown in Table 2.1, while various model-agnostic XAI methods exist, such as LIME, SHAP, and ConvTimeNet, they are often not designed for time series forecasting. Although the work by Wang *et al.* [150] has been evaluated on three different DL models – Multilayer Perceptrons (MLP), Fully Convolutional Networks (FNN), and Residual Networks (FCN) – it has not yet been explored on RNN or LSTM. Similarly, many XAI methods proposed to explain the predictions made by AI models commonly employed for time series forecasting, such as LSTM [38, 84, 103], tend to omit the temporal dimension. Moreover, there is currently a limited amount of XAI methods specifically targeting regression problems, and even fewer methods that have been adapted for time series forecasting analysis. Even though some of the XAI methods for time series, such as perturbation-based saliency methods, can be used for both classification and forecasting tasks, they have not yet been tailored explicitly to the challenges of forecasting, particularly in capturing time-dependent explanations.

Despite the ability of NAM to produce explanation graphs that precisely describe the reasoning behind its decisions, it has not been extensively applied to the time series domain. Furthermore, to the best of our knowledge, NAM remains the only XAI method capable of producing NAM-styled explanation graphs, making it uniquely suited for interpretable forecasting. Although Jo *et al.* [71] introduced NATM as a modification of NAM for time series problems, their approach was limited to generating feature importance through contribution maps rather than utilising the full explanatory potential of NAM. Similarly, while Feddersen *et al.*

[47] proposed HNAM, which draws inspiration from NAM, their method does not process each feature through an individual neural network, thereby failing to produce the structured explanation graphs that NAM is designed for.

These limitations reveal a significant gap in the current literature: while NAM presents a promising framework for explainability, there has been little effort to adapt or extend it for time series forecasting. Existing adaptations, such as NATM and HNAM, either simplify the explainability aspect or do not fully exploit NAM's potential to generate detailed, interpretable representations of feature contribution. This gap underscores the need for further research to refine NAM-based methods for time series forecasting, ensuring both enhanced predictive performance and temporally-aware explanatory insights. By addressing this gap, our work aims to extend the explanatory power of NAM to the time series domain while preserving its interpretability and improving its effectiveness in forecasting applications.

2.6 Summary

This chapter provided an overview of the related literature in two main areas that inspired this research: time series forecasting and classification models and time series eXplainable Artificial Intelligence (XAI) methods. The discussion for time series forecasting and classification models spans traditional statistical techniques, Machine Learning (ML) algorithms, and Deep Neural Network (DNN) methods. Despite the high predictive performance of DNNs models in time series forecasting, they are unfortunately non-interpretable. We have also explored the landscape of XAI methods that can be used to explain time series data, examined the strengths and limitations of each approach, categorising them into post-hoc and ante-hoc techniques, as well discussing the advantages and disadvantages inherent in these XAI techniques and shedding light on their utility in enhancing model interpretability and transparency.

Chapter 3

Methods

"If you can't convince them, confuse them."

- Harry S. Truman

Our research journey commenced with a deliberate focus on univariate time series analysis, aiming to establish a foundational understanding through simplified analyses. Subsequently, we transitioned to exploring multivariate time series analysis, recognising its broader applicability in real-world scenarios.

Through our exploration of univariate analysis, it became evident that Deep Neural Network (DNN) models outperformed traditional statistical methods, such as Hidden Markov Model (HMM) [112] (see Appendix A.1 for details), thus motivating our continued investigation into DNN-based approaches. Specifically, the Long Short-Term Memory (LSTM) [58] model emerged as a promising candidate, showcasing superior performance in univariate settings, prompting us to extend its application to multivariate time series. Building on this, we aim to experiment with enhanced LSTM-based architectures – such as LSTM combined with attention mechanisms – with the goal of improving predictive performance over the

standard stand-alone LSTM model. This work investigates whether such LSTM-based configurations can offer measurable advantages in handling multivariate time series prediction tasks.

Previous research has demonstrated that integrating attention mechanisms into LSTMs and encoder-decoder architectures can significantly enhance their predictive performance [29, 41, 86, 146, 160, 168, 171]. To further investigate whether attention mechanisms can improve the predictive accuracy of a stand-alone LSTM when applied to multivariate time series data, we will experiment with two attention-based approaches: attention-based LSTM (AttnLSTM) and attention-based encoder-decoder (AttnED). The key difference between these models with the existing work in time series domain is that AttnLSTM employs a self-attention to selectively focus on parts of the inputs to allow LSTM to prioritise relevant information, while AttnED incorporates self-attention between LSTM-based encoder and decoder, allowing the model to better capture long-term dependencies and subtle patterns across sequences. By comparing these distinct configurations, we aim to identify which architecture yields the best predictive performance.

We also delved into state-of-the-art post-hoc eXplainable AI (XAI) methods during the initial phases of this research, such as SHapley Additive exPlanation (SHAP) [94] and Local Interpretable Model-agnostic Explanations (LIME) [116], recognising their potential to explain complex, non-interpretable models. Although the potentially high predictive performance of a DNN can be retained using post-hoc methods, explanations from post-hoc methods are often criticised for being not faithful to the underlying model because they can only *approximate* the underlying prediction model [3, 50, 88, 120]. On the other hand, ante-hoc methods are interpretable-by-design, meaning that their explanations would be more faithful than post-hoc methods. However, their predictive performance are often less than desirable, such as N-Beats [108] (see Appendix A.2 for details) and Neural Additive Models (NAM) [4] that were investigated during the initial phases.

Insights gained from experiments on AttnLSTM and AttnED have then led to the creation of FocusLearn. FocusLearn is an interpretable DNN model inspired

by NAM that does not require a post-hoc method but can still outperform interpretable DNNs, such as NAM itself, in terms of predictive performance, whilst matching the predictive performance of non-interpretable models, such as LSTM.

The predictive performance of AttnLSTM, AttnED, and FocusLearn will be compared with benchmark models (see section 3.1.4) on four time series data sets (see section 3.1.2). The selection of these data sets is based on their ability to capture key challenges in time-series analysis, including varying number of features and number of instances. This ensures a diverse and representative benchmark for evaluating model effectiveness. Using a small but diverse set allows for deeper analysis, including extensive hyper-parameter tuning and ablation studies, which would be less feasible with a larger data set pool given available computational resources.

While additional data sets could provide further insights, the experimental scope was constrained by available computational resources. The hardware used for this study[1] supports moderate workloads but presents limitations when scaling to larger data sets or computationally intensive models, particularly deep learning architectures with large parameter spaces. Additionally, cloud-based computing was not utilised due to budget constraints. Despite these considerations, the selected data sets offer a representative evaluation of the models within the available computational framework.

3.1 Experimental Setup

In this section, we discuss the essential components required for setting up the experiments. Firstly, we discuss the prediction evaluation metrics employed to assess the effectiveness and performance of the proposed models, ensuring a comprehensive and objective analysis of experimental outcomes. Next, we address the selection and characteristics of different data sets. Subsequently, we discuss the pre-processing techniques employed to enhance the quality and usability of the data. Moreover, we examine benchmark models utilised as reference points

[1] AMD Ryzen 5 3600X 6-Core Processor, 16GB RAM, and GeForce RTX 3060 GPU (12GB VRAM).

for comparison, demonstrating their significance in gauging the performance of all models. Lastly, the training procedure is discussed, describing aspects such as optimisation algorithms and hyperparameter tuning to facilitate reproducibility and reliability.

3.1.1 Evaluation Metrics

To test the generalisability of all models, three standard error estimators are used to evaluate the prediction on *unseen* test sets for regression tasks: Symmetric Mean Absolute Percentage Error (sMAPE), Mean Absolute Scaled Error (MASE), and Weighted Absolute Percentage Error (WAPE).

$$sMAPE = \frac{100}{N} \sum_{i=1}^{N} \frac{|y_i - \tilde{y_i}|}{\frac{|y_i|+|\tilde{y_i}|}{2}}, \tag{3.1}$$

$$WAPE = \frac{\sum_{i=1}^{N} |y_i - \tilde{y_i}|}{\sum_{i=1}^{N} |y_i|} \times 100, \tag{3.2}$$

$$MASE = \frac{\frac{1}{N} \sum_{i=1}^{N} |y_i - \tilde{y_i}|}{\frac{1}{N-1} \sum_{i=2}^{N} |y_i - y_{i-1}|}, \tag{3.3}$$

In Equations (3.1) - (3.3), y_i is the true value, $\tilde{y_i}$ is the predicted value, and N is the number of data points.

sMAPE is a scale-independent metric that measures the relative error between predicted and actual values. It is robust to scale variations and provides an interpretable percentage-based measure of forecasting accuracy, making it useful for comparing data sets with different value ranges. WAPE is useful when dealing with data sets that have highly imbalanced target distributions. By weighting absolute errors by their corresponding true values, WAPE ensures that predictions for high-value targets do not disproportionately dominate the error calculation. MASE accounts for autocorrelation in time-series data by normalising the absolute error against a naïve benchmark (such as a simple moving average). This makes it particularly useful for evaluating performance relative to a baseline and ensures that errors are comparable across data sets.

Each of these metrics addresses different aspects of predictive performance. While sMAPE provides an intuitive, percentage-based error measure, MASE ensures that performance is assessed relative to a baseline, and WAPE accounts for value imbalances in the data set. Together, they offer a more complete picture of regression model performance.

Accuracy, F1 Score, and Area Under Curve (AUC) are used for classification tasks. F1 score is the harmonic mean of precision and recall. Each of the metrics is calculated as follows, where TP/TN and FP/FN are the positive/negative false positive/negative, respectively:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.4}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.6}$$

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.7}$$

AUC measures the area under the receiver operating characteristic (ROC) curve, which is a graphical representation of how well the model performed and shows the relationship between True Positive Rate and False Positive Rate.

Accuracy is a straightforward metric that measures the proportion of correctly classified instances. However, it can be misleading in imbalanced data sets. The F1 Score is the harmonic mean of precision and recall, making it useful when class distributions are imbalanced. It ensures that both false positives and false negatives are accounted for, preventing misleading results due to class imbalance. AUC evaluates the trade-off between true positive and false positive rates across different classification thresholds. It is particularly useful for assessing model discrimination ability in cases where class probabilities are important.

By using these three metrics, classification performance is evaluated from different perspectives: Accuracy provides a broad correctness measure, F1 Score accounts for class imbalance, and AUC quantifies the model's ability to distinguish between classes across varying thresholds.

As predicted values are short-term time series in the regression tasks, a visual estimation metric is also used by plotting true future time series against predicted time series, to determine how well models predict the ups-and-down pattern in future values in addition to error estimators. Finally, the computation time of the models is also taken into account as an evaluation metric for both regression and classification tasks.

3.1.2 Data

Two regression and two classification data sets are considered. These data sets are chosen for a diverse set of real-world data sets, in terms of number of features, number of instances, and applied domains, to increase our confidence in the generalisability of our findings.

3.1.2.1 Air Data Set

In the Beijing Multi-Site Air-Quality (*Air*) data set [167], 7 years of meteorological data along with 4 years of $PM_{2.5}$ data of Beijing, China at 36 monitoring sites were collected. $PM_{2.5}$ refers to the fine particulate matter (PM) concentration in the air with aerodynamic diameter of less than $2.5\mu m$. Particulate matter includes all airborne particles that are not gases, consisting of various chemical compounds, some of which are toxic. Due to their small size, these particles can enter the bloodstream and reach vital organs like the heart and brain, posing significant health risks, particularly to vulnerable groups such as children, the elderly, and those with respiratory issues[2]. The task with this data set is to predict future $PM_{2.5}$ values.

3.1.2.2 Energy Data Set

Appliances energy consumption (*Energy*) [20] is a regression data set that contains temperature and humidity data in various rooms of a low energy building,

[2] https://www.gov.uk/government/statistics/air-quality-statistics/concentrations-of-particulate-matter-pm10-and-pm25

along with weather data from a nearby airport weather station (Chievres Airport, Belgium), to predict energy consumption of the appliances in Watts hour (Wh). A ZigBee wireless sensor network was used to monitor the temperature and humidity conditions of the house at a frequency of around 3.3 minutes, and these data were then averaged for 10 minutes periods. The weather data from Chievres Airport (Temperature outside, Pressure, Humidity outside, Wind speed, Visibility, and Dew point) were downloaded from a public data set from Reliable Prognosis [3]. The actual energy data from the appliances were also logged every 10 minutes with m-bus energy meters.

3.1.2.3 EEG Data Set

In the EEG Eye State (*EEG*) data set [107], continuous electroencephalogram (EEG) measurement are collected from the Emotive EEG Neuroheadset. EEG is a test that measures electrical activity in the brain. In the EEG test, small, metal disc named electrodes are attached to the scalp, and they can detect the electrical impulses of brain cells communications. The duration of the EEG measurement is 119 seconds, and eye state (*e.g.,* whether the eyes are closed or open) are detected by a camera during EEG measurement and added to the data set manually later, where 1 indicates eye-closed and 0 indicates eye-open. The task is a binary one to classify eye state based on the EEG measurement.

3.1.2.4 Weather Data Set

Another binary classification data set is the Rain in Australia (*Weather*) data set[4], which contains 10 years of daily weather observations drawn from a number of weather stations in Australia. Features include location, minimum/maximum temperature, amount of rainfall, and wind gust. The task is to predict whether or not it will rain tomorrow.

[3]Energy: `rp5.ru`

[4]Weather: `http://www.bom.gov.au/climate/data/` and `http://www.bom.gov.au/climate/dwo/`

3.1.2.5 Summary

Table 3.1 summarises the data sets selected for this study. The selection was guided by several key factors to ensure that the data sets were suitable for evaluating the proposed model's feature selection capability and predictive performance. An important requirement was that the data sets contain a sufficient number of features to allow for meaningful feature selection, a core aspect of the model design. However, to maintain computational feasibility given available hardware constraints, data sets with extremely high dimensionality (100+ features) were not included, as such large-scale time-series data sets are also not widely available. Additionally, the data sets selected span a range of instance counts to enable a reliable assessment of model scalability and generalisation.

Air and *Energy* are regression data sets but differ in structure and complexity. *Air* involves environmental sensing data collected across multiple geographic sites, where features include meteorological variables and air pollution indicators with moderate levels of noise and a relatively smooth seasonal trend (see Appendix D.2.1). In contrast, *Energy* consists of high-frequency indoor sensor data that captures dynamic fluctuations in energy usage, reflecting a more complex temporal pattern due to its higher variance and clearer seasonality. This contrast enables testing model robustness across both smoother and more volatile regression targets (see Appendix D.1.1).

The classification data sets, *EEG* and *Weather*, differ in terms of data scale and class imbalance. *EEG* is a small and noisy data set where the goal is to detect eye state from electrical brain activity. It is relatively balanced (class ratio of 1:0.82), allowing for straightforward binary classification evaluation. Whereas *Weather* is a large-scale data set with over 137,000 samples and is significantly imbalanced (class ratio of 1:0.26), providing a contrasting challenge where precision, recall, and class imbalance handling become crucial. These differences help test how well models generalise across tasks of varying difficulty and structure.

Missing data is also present in two data sets (0.98% in *Air*, 9.83% in *Weather*), making them suitable for evaluating the model's resilience to incomplete information. This is particularly important in time series data, where missing values

Table 3.1: **Descriptions of the four selected time series data sets**.

Name	Air	Energy	EEG	Weather
Source	[167]	[20]	[107]	http://www.bom.gov.au/climate/data/
Instances	420,768	19,735	14,892	137,910
Features	16	29	14	21
Class Distribution	-	-	1: 0.82	1: 0.26
Feature Type	Mixed	Continuous	Continuous	Mixed
% Missing Data	0.98%	0%	0%	9.83%
Single or Multiple Sequences	Multiple sequence	Single sequence	Single sequence	Single sequence

can disrupt the temporal continuity and introduce bias or signal distortion if not handled appropriately. Effective time series models must be able to infer patterns even when observations are sporadically absent, either due to sensor failure, transmission issues, or recording constraints. By testing on data sets with real-world missingness, the evaluation provides insight into how robust the model is under conditions of partial observability – an essential quality for practical deployment.

The concept of complexity in this context refers not only to feature dimensionality and instance count but also to signal quality, temporal variability, and class imbalance. *EEG*, for example, is complex due to its noisy, while *Weather* is complex due to its large scale and imbalanced distribution. *Energy* is complex because of its strong seasonality and multiple correlated features. Together, these data sets ensure that the evaluation is not biased towards a single data profile.

Beyond statistical characteristics, selected data sets come from diverse real-world application domains, ensuring that the findings of this study have practical applicability beyond a single domain. This diversity enhances the robustness of model evaluation and supports the generalisability of the performance metrics. Furthermore, the data sets provide a meaningful test bed for assessing the proposed model's ability to select relevant features, given that time-series data often involve complex relationships between variables. While additional data sets could further reinforce these findings, the selected data sets strike a balance between computational feasibility and empirical validation, allowing for a thorough investigation of the model's strengths and limitations. Future work will explore higher-dimensional data sets to extend the findings of this study.

3.1.3 Pre-processing the Data

3.1.3.1 Handling Missing Data

Handling missing data is an important pre-processing step to be taken for all data sets, in particular since the percentage of missing data in domains such as medical data can be as high as 98% [23]. Therefore, it would not be feasible to simply delete the rows that contain missing data. Aside from missing variable values in two of the data sets, there are also missing timestamp values resulting in gaps in the time series. As all data sets used in this study are longitudinal, *i.e.,* *Air*, *Energy*, *EEG*, and *Weather*, such that same variable is repeatedly measured at different times, the Trajectory Mean imputation method is suitable here [169], where the mean of the observed values along the time trajectory, or the trend, of its values over time is imputed. Trajectory Mean imputation method is suitable for longitudinal data because it leverages the temporal nature of the data by capturing the underlying trend in this repeated-measures data, thus allowing for more accurate estimation of missing values because data points are not treated as independent observations [169].

Mathematically, the trajectory mean imputation for the missing value, $x_i(t)$, can be expressed as equation 3.8.

$$\hat{x}_i(t) = \frac{1}{n_i} \sum_{k=1}^{n_i} x_i(t_k) \tag{3.8}$$

In Equation (3.8) $\hat{x}_i(t)$ is the imputed value for variable i at time t, n_i is the number of available neighbouring values used for imputation, and $x_i(t_k)$ represents the available neighbouring values of variable i at different time points t_k.

3.1.3.2 Encoding Categorical Variables

Categorical variables where the categories have a meaningful order are transformed with ordinal encoding and nominal variables are transformed with one-hot

encoding, to appropriately represent the nature of the data. Ordinal encoding was applied to variables with a clear, ordered relationship among categories, such as "month" in the *Air* data set (*e.g.,* "January" - "December"). This method preserves the inherent order of these categories by assigning them integer values, allowing models to interpret the numerical relationships correctly. On the other hand, one-hot encoding was utilised for nominal variables, which lack any natural ordering among categories, such as "Rain Today" in the *Weather* data set (*e.g.,* "yes" and "no"). This technique creates binary columns for each category, indicating their presence or absence, thus preventing models from mistakenly assuming an ordinal relationship. By employing these encoding strategies, the transformed data accurately reflects the underlying structure of the categorical variables, enhancing the learning ability of all experimented models.

3.1.3.3 Standardisation

Continuous variables are scaled with standardisation to ensure the range of data lies between a smaller range for the model to learn better for all data sets. DNN models generally do not have the constraint of expecting features to have a certain distribution, so transforming the distribution is not necessary but it does allow the algorithms to converge faster and minimise the weight of any variable with extreme values. Standardisation is particularly important for training LSTM models, since standardisation on the data centres the noise from trend reverse signals [125] and prevents activation functions to saturate. The data is standardised by:

$$x_{scaled} = \frac{x - \mu}{s} \tag{3.9}$$

where μ is the mean and s is the standard deviation of the data.

3.1.3.4 Outliers

Outliers are defined as data points that deviate significantly from the main data distribution [79]. Identifying them can be challenging, but statistical methods help in detection. This study employs the standard deviation method after data standardisation and normalisation, as it is most effective for normally distributed data [65]. Outliers are detected using z-scores, where any data point exceeding three standard deviations from the mean is classified as an outlier—a common threshold in practice.

3.1.3.5 Resampling Classification Data

As discussed in section 3.1.2, the *Weather* data set used in this study is severely imbalanced (class distribution 1:0.26), meaning that the number of instances in one class (*i.e.,* rained next day) significantly outweighs those in the other class (*i.e.,* did not rain next day). This imbalance can cause the model to favour the majority class during training, as it is exposed to far more examples of that class. To address this issue, the *Weather* data set was resampled using the Synthetic Minority Over-sampling Technique (SMOTE) [24] to achieve a balanced class distribution. SMOTE works by generating synthetic samples for the minority class by interpolating between existing minority class instances. This method helps in avoiding overfitting, which can occur with simple duplication of samples, and allows the model to learn a more generalisable decision boundary. By balancing the classes, SMOTE can improve the model performance on the minority class, leading to a more robust and fairer classifier.

3.1.4 Benchmark Models

For bench-marking, the prediction results of AttnLSTM, AttnED, and FocusLearn are compared against selected baselines, including state-of-the-art interpretable models and commonly used strong-performing methods for multivariate time series tasks. These models are selected for their relevance in interpretability, widespread adoption, or performance reputation.

- **Neural Additive Model (NAM)** [4]. Chosen because FocusLearn extends this architecture; serves as a natural baseline for assessing improvements in predictive performance as well as to maintain interpretability.
- **Scalable Polynomial Additive Model (SPAM)** [42]. SPAM is selected as a more recent interpretable model that improves on NAM by including feature interactions, providing a benchmark for predictive performance. It is worth to note here that only feature importance and not feature curves can be provided by SPAM as explanations.
- **Long Short-Term Memory (LSTM)** [58]. Serves as a widely used deep learning baseline for time series, allowing us to evaluate whether FocusLearn can match or surpass standard DNN performance. LSTM is not interpretable.
- **Extreme Gradient Boosted Trees (XGBoost)**. This is another popular standard approach[5] – it is generally accepted that, although a single tree can be interpretable, interpretability degrades rapidly as the number of trees grows. Both NAM and SPAM compare themselves with XGBoost and therefore we do too.

Detailed descriptions of the benchmark models are provided in the following chapter. XGBoost by default can handle missing data, and these missing data do not need to be imputed where other models do. However, to ensure a fair comparison between XGBoost and and other models, we also evaluate a version of XGBoost (XGBoost-imputed) where missing data are imputed in the same manner as the other models using the techniques as mentioned in section 3.1.3.1.

3.1.5 Training Procedures

For all data sets, 80% of the data is used as the training set, 20% of the training set is used as the validation set, and 20% of the data is reserved as the unseen testing set. To ensure a fair comparison, all models are hyper-tuned. Hyper-parameters are hyper-tuned using Ray-tune[6] with the validation set. More specifically, we use

[5]XGBoost: `https://xgboost.readthedocs.io/en/stable/`
[6]Ray-tune `https://www.ray.io/`

Table 3.2: **Hyper-parameters to be tuned for all models with their settings**.

	Hyper-parameters	Search Space Type	Search Range/Values
NAM	Learning Rate	Log uniform	$[1e^{-5}, 1.0]$
	Dropout Rate	Log uniform	[0.01, 1.0]
	Batch Size	Integer	32, 64, 128, 256, 512
	Hidden Sizes	List of integers	[64, 32], [128, 64], [256, 128], [512, 216]
SPAM	Learning Rate	Log uniform	$[1e^{-5}, 1.0]$
	Dropout Rate	Log uniform	[0.01, 1.0]
	Batch Size	Integer	32, 64, 128, 256, 512
LSTM	Learning Rate	Log uniform	$[1e^{-5}, 1.0]$
	Dropout Rate	Log uniform	[0.01, 1.0]
	Batch Size	Integer	32, 64, 128, 256, 512
	Activation	Categorical	ReLu, Sigmoid, Tanh
	Number of hidden unit	Integer	32, 64, 128, 256, 512
XGBoost	Number of estimators	Integer	1, 2, 4, 8, 10, 20, 50 100, 200, 250, 500, 1,000
	Max Depths	Integer	2, 5, 10, 20, 25, 50 100, 200, 500, 1,000, 2,000
	Subsample	Log uniform	[0.0, 1.0]
	Learning Rate	Log uniform	$[1e^{-5}, 1.0]$
AttnLSTM	Learning Rate	Log uniform	$[1e^{-5}, 1.0]$
	Dropout Rate	Log uniform	[0.01, 1.0]
	Batch Size	Integer	32, 64, 128, 256, 512
	Activation	Categorical	ReLu, Sigmoid, Tanh
	Number of hidden unit	Integer	32, 64, 128, 216, 512
AttnED	Learning Rate	Log uniform	$[1e^{-5}, 1.0]$
	Dropout Rate	Log uniform	[0.01, 1.0]
	Dropout rate in the encoder LSTM	Log uniform	$[1e^{-5}, 1.0]$
	Dropout rate in the decoder LSTM	Log uniform	$[1e^{-5}, 1.0]$
	Batch Size	Integer	32, 64, 128, 256, 512
	Activation	Categorical	ReLu, Sigmoid, Tanh
	Number of hidden unit	Integer	32, 64, 128, 216, 512
FocusLearn	Learning Rate	Log uniform	$[1e^{-5}, 1.0]$
	Dropout rate in the RNN	Log uniform	[0.01, 1.0]
	Dropout rate in the MNN	Log uniform	[0.01, 1.0]
	Dropout rate in the final output	Log uniform	[0.01, 1.0]
	Batch Size	Integer	32, 64, 128, 256, 512
	Number of hidden unit in the RNN	Integer	32, 64, 128, 256, 512
	Number of hidden units in the MNN	List of integers	[64, 32], [128, 64], [256, 128], [512, 216]

Log-uniform distributions were used to sample values over several orders of magnitude, which is more appropriate for scale-sensitive hyper-parameters such as learning rate and dropout rate.

the Asynchronous Successive Halving Algorithm scheduler [85] for better parallelism, with the Optuna [7] search algorithm. All models are trained with the Adam optimisation technique, and 200 epochs with early call-back to avoid over-fitting. Hyper-tuned parameters for each model are listed in Table 3.2.

3.1.6 Reproducibility Strategy

Reproducibility is a fundamental principle of scientific research and is essential for ensuring the credibility and utility of machine learning experiments. In this study, a reproducibility strategy is implemented to enable independent verification of the findings and to facilitate future extensions of the work.

The reproducibility strategy adopted here follows the recommendations on how to create reproducible AI [54] by UK Reproducible Network [8], in terms of data, source code, methods, and experiments. These recommendations emphasise the importance of transparency, accessibility, and methodological rigour in empirical machine learning studies.

3.1.6.1 Data Access and Pre-processing Protocols

The data sets used in this study are publicly available and fully cited in the relevant sections. Pre-processing pipelines, including imputation of missing values, standardisation of feature distributions, outlier detection and removal, and class balancing through resampling, are implemented in reproducible scripts shared in the repository. These steps are described in detail in section 3.1.3, ensuring full transparency in data preparation.

3.1.6.2 Code Design and Organisation

Besides of having an open repository for the source code on GitHub, the codebase developed for this study is also structured to promote clarity, reusability, and

[7] Optuna search algorithm `https://optuna.org/`
[8] UK Reproducible Network: `https://www.ukrn.org/artificial-intelligence/`

ease of maintenance. All key components – such as data pre-processing, model architecture definitions, training routines, evaluation procedures, and visualisation - are implemented using modular Python scripts organised into well-separated files and folders. Each module encapsulates a specific functionality, making the codebase easy to navigate and extend.

Functions and classes were used extensively to abstract and isolate key procedures. For example, each model architecture is defined as a class, allowing consistent initialisation and training logic across different experiments. Data loaders, training loops, and evaluation routines are also implemented as reusable functions or class methods to avoid code duplication and improve readability.

Descriptive docstrings and inline comments are included to facilitate understanding and replication by other researchers. The code is developed with readability and transparency in mind, following standard Python practices and naming conventions.

To further support reproducibility, all relevant parameters (*e.g.,* learning rates, batch sizes, epochs) are defined at the top of each script or within the main execution functions. This design choice ensures that experiments can be re-run with minimal manual intervention or configuration.

3.1.6.3 Training and Evaluation Procedure

To maintain consistency and fairness in model comparison, all baseline and proposed models are trained using identical protocols. Each model is evaluated over five independent runs, with different random initialisations using the same optimal hyper-parameters identified through hyper-parameters tuning. Performance metrics reported in this study represent the mean and standard deviation across these trials.

The same training and test splits are maintained across models to ensure consistency in evaluation. In addition, all training logs, including hyper-parameter settings and system utilisation metrics, are retained to support auditability.

3.2 How can the prediction results of an LSTM be improved for multivariate time series problems?

LSTM models are well-suited for sequential data and have demonstrated significant success in various predictive tasks due to their ability to capture long-term dependencies and temporal dynamics [61, 63, 121, 136]. Sequential data refers to those data where the order of the element matters, such that the position of each data point within the sequence is important and are typically indexed by time or position to reflect temporal or logical progression. A common example is the time series data. Multivariate time series, in particular, present greater complexity than univariate time series due to the intricate dependencies and interactions among multiple variables over time. Consequently, an LSTM-based model needs to be more sophisticated to effectively manage these complexities.

3.2.1 How does the performance of an attention-based LSTM compare to a stand-alone LSTM in multivariate time series forecasting?

We want to explore combining LSTM with an **attention mechanism (AttnLSTM)**, taking inspiration from other work [29, 86, 168]. Different to these works, we will be using a self-attention mechanism [144] as self-attention can selectively focus on parts of the inputs, and this selective focus allows LSTM to prioritise relevant information and potentially enhance predictive accuracy [70]. By integrating the self-attention into the LSTM, we aim to improve the ability of LSTM to capture important temporal patterns and dependencies within multivariate time series data. Therefore, it is anticipated that AttnLSTM will deliver better predictive performance than a stand-alone LSTM due to its ability in handling complex multivariate time series data.

Self-attention was introduced as a core mechanism in the Transformer architecture [11], and there are many reasons to why we are not exploring transformers themselves directly. Although transformers have demonstrated strong performance across various domains, including time series analysis, they are not the focus of this study due to their computational complexity and resource demands. Unlike LSTMs, which process inputs sequentially, transformers rely on self-attention mechanisms that compute dependencies across all time steps simultaneously. This results in quadratic time and memory complexity with respect to sequence length, making training significantly more computationally expensive. Additionally, transformers require larger data sets to generalise effectively and often involve extensive hyperparameter tuning, further increasing their complexity. Furthermore, from our initial research, it was found that complicated models are more difficult to explain. Additionally, simple feature importance XAI methods, such as SHAP and LIME, might be even more prone to being unfaithful to the model in such complex scenarios [133], as these methods often fail to capture the intricate relationships and temporal dependencies within the data. Therefore, to maintain a balance between predictive and explanation performance, we opted for an LSTM-based approach combined with an attention mechanism, which provides a more manageable level of complexity while still enhancing the predictive performance.

3.2.2 Can an attention-based encoder-decoder predict better than an attention-based LSTM?

The most important reason transformers are not investigated in this work is their tendency to struggle with capturing temporal dependencies in long sequences [162]. Unlike RNNs, which process data sequentially and inherently retain ordering through recurrence, transformers rely on self-attention mechanisms that require explicit positional encodings to represent sequence order. This lack of an inductive bias toward temporal structure can make transformers less effective at modelling long-range dependencies, especially in cases where important information is distributed across distant time steps. Additionally, transformers store attention scores for every pair of time steps, thus creating a memory bot-

tleneck and limiting the ability of transformers in handling very long sequences. On the other hand, RNNs have gating mechanisms that can help them retain long-range dependencies. Moreover, transformers operate within a fixed input window, meaning that once the context length is exceeded, earlier information is discarded, whereas RNNs can in principle maintain a persistent memory state through recurrence and gating mechanisms, making them better suited for tasks that require learning from arbitrarily long contexts. Existing literature suggests that integrating an attention mechanism into an RNN-based encoder-decoder architecture can address this limitation by combining the strengths of sequential modeling with attention-driven flexibility [41, 146, 160, 171]. The attention mechanism enables the model to capture long-term dependencies and subtle patterns more effectively, particularly when combined with an encoder-decoder system. This combination has been shown to be useful for identifying irregular patterns, trends, and seasonal variations in real-world time series data.

Taking inspiration from these works [41, 146, 160, 171], we will explore the capabilities of an **attention-based encoder-decoder architecture (AttnED)**. Unlike these studies, we utilise LSTMs in both the encoder and decoder, leveraging their strengths in handling sequential data, with self-attention incorporated between the encoder and decoder. This addition of self-attention enhances the LSTM's ability to focus on relevant parts of the input sequence. By integrating attention into the encoder-decoder framework, we aim to further improve the predictive performance of LSTMs, particularly in capturing complex temporal dependencies and interactions within multivariate time series data. AttnED is expected to outperform both LSTM and AttnLSTM while requiring fewer computational resources than transformers.

3.3 Can an interpretable DNNs be designed for multivariate time series problems by leveraging feature selection and MNN?

Choosing between post-hoc and ante-hoc methods in any domain can be a difficult task. Post-hoc methods can only approximate their underlying model, which makes explanations less faithful. On the other hand, ante-hoc approaches are often criticised for their lack of flexibility and usability, as well as their trade-off between accuracy and interpretability [116]. NAM [4] is a fully interpretable ante-hoc DNN that can produce visualisations that are the exact depiction of how NAM arrived at a prediction. These visualisations are easy to interpret, making it an attractive choice for many real-world applications. However, from our initial phases of the research, it was found that the predictive performance of NAM is less than desirable, often performing worse than traditional statistical methods, such as an HMM.

Therefore, is it possible to design an interpretable DNNs, that offers the same level explanation of NAM, but can outperform NAM in terms for predictive performance, as well as matching the predictive performance of those non-interpretable models? We have named this approach as FocusLearn. The architecture of FocusLearn is inspired by Modular Neural Networks, specifically with having each individual neural network attend to a single feature. The outputs of these individual neural networks are then linearly combined together, as in additive models such as NAM. In additive models, including NAM, the family of neural networks are summed directly to produce the final prediction and this linear combination does not involve learned weights to ensure the contribution of each feature to the final prediction remains independent of any scaling factors.

It is found through our experiments with NAM , that learning DNNs in the linear space tends to result in a poor predictive performance. This is also the reason why it is proposed in SPAM to learn the higher-order feature interaction by using a tensor rank decomposition of polynomials. Therefore, we are leveraging insights gained from the AttnLSTM and AttnED experiments as inspiration for

FocusLearn by introducing an RNN to learn the temporal dependencies in the data and an attention mechanism to drive for feature selection, as learning only the most salient features has been shown to be a useful method of improving interpretability and allocating learning capacity more effectively [21, 53].

Through this synthesis of **modular architecture**, **linear computation**, and **attention-driven feature selection**, FocusLearn endeavours to set a new standard for interpretable DNNs with improved predictive capabilities, whilst maintaining interpretability on the same level with NAM. The entirety of the inference path of FocusLearn is interpretable since the computation for prediction is simply a series of linear combinations of interpretable networks. Predictive performance is improved, not by making the inference step nonlinear, but from a better selection of features using attention to learn the modular networks during training and to ignore features that have no contribution to the model.

3.4 Ablation Experiments of FocusLearn

FocusLearn consists of several components (discussed in detail in section 4.1.3), each of which may play a crucial role in the overall performance of FocusLearn. To evaluate the contribution of each component, we will conduct a series of ablation experiments. These experiments will involve removing or modifying individual components of FocusLearn, allowing us to compare the performance of the full FocusLearn against various ablated versions. The results will be evaluated using the evaluation metrics as introduced in section 3.1.1, ensuring that the contribution of each component is quantitatively assessed. This process will help to demonstrate the importance of each component and also identify potential limitations and areas for improvement in the architecture of FocusLearn.

3.5 Summary

This chapter has presented the full methodological pipeline adopted in this research, including model evaluation, data selection, pre-processing, benchmarking,

and the motivation and development of the proposed model – FocusLearn.

We began by outlining the evaluation metrics used for assessing model performance. Metrics were chosen to suit both regression and classification tasks – such as SMAPE, WAPE, and MASE for regression, and Accuracy, F1 score, and AUC for classification. Each metric was selected for its specific strengths and was discussed in the context of what aspect of performance it measures. Together, they provide a comprehensive picture of model behaviour under various evaluation criteria.

Next, we described the data sets used in the analysis, emphasising their diversity across domains, scales, complexity, and class distribution. The complementarity of these data sets allows for a robust evaluation across a range of prediction challenges. While the selected data sets are sufficient to validate the model's effectiveness under different conditions, limitations in data set scale and dimensionality were acknowledged. Larger or more complex data sets will be explored in future work to further extend the scope of this study.

We then discussed the key pre-processing techniques applied, including strategies for handling missing data, encoding categorical features, standardisation, outlier removal, and class imbalance mitigation via resampling. These steps were necessary to ensure data quality and consistency before model training and evaluation.

Following this, we introduced the benchmark models against which our proposed method was compared. These included both traditional and state-of-the-art models commonly used for interpretable and predictive time series modelling. We also detailed the training procedures and measures taken to ensure reproducibility, including consistent random seed usage and repeated trials with fixed optimal hyperparameters.

We then described the experimental exploration that led to the development of FocusLearn. This began with testing whether augmenting LSTM with attention mechanisms could enhance predictive performance on multivariate time series tasks. We further examined whether an attention-based encoder-decoder archi-

tecture could outperform attention-augmented LSTM models. Insights gained from these experiments informed the development of FocusLearn, a model that integrates interpretability, modular neural networks, and feature selection.

Finally, we presented ablation experiments designed to assess the contribution of each component within FocusLearn. These experiments isolate the role of attention, feature selection, and modularity, providing insight into which aspects of the model architecture most significantly influence performance.

Chapter 4

Results

This section presents the new Neural Network architectures developed and the outcomes and findings derived from a comprehensive analysis of the conducted experiments. The results show that FocusLearn[1] outperformed interpretable models, Neural Additive Model (NAM) and Scalable Polynomial Additive Model (SPAM), on all data sets. Sometimes FocusLearn even outperformed non-interpretable models, LSTM, attention-based LSTM (AttnLSTM), attention-based encoder-decoder (AttnED), and non-interpretable XGBoost. We discuss the prediction results of AttnLSTM, AttnED, and FocusLearn in comparison with the benchmark models on time series data first, then we discuss the series of ablation experiments conducted on FocusLearn.

[1] GitHub Repository: `https://github.com/qisuqi/FocusLearn/tree/main`

4.1 Architecture Designs

4.1.1 AttnLSTM - Attention-based LSTM

The integration of self-attention and LSTM enables the model to selectively focus on specific parts of the input sequence, a process that mirrors the human biological system's efficient information processing strategy. In essence, the attention mechanism in the AttnLSTM model functions akin to human cognitive processes, where information is not processed all at once, but instead attention is directed to distinct and relevant portions of data [106]. Following many existing literature [29, 86, 168], the workflow of the AttnLSTM involves the initial processing of inputs through the attention mechanism before the LSTM as shown in Figure 4.1. Different to these literature, the specific attention mechanism used in AttnLSTM is self-attention [144].

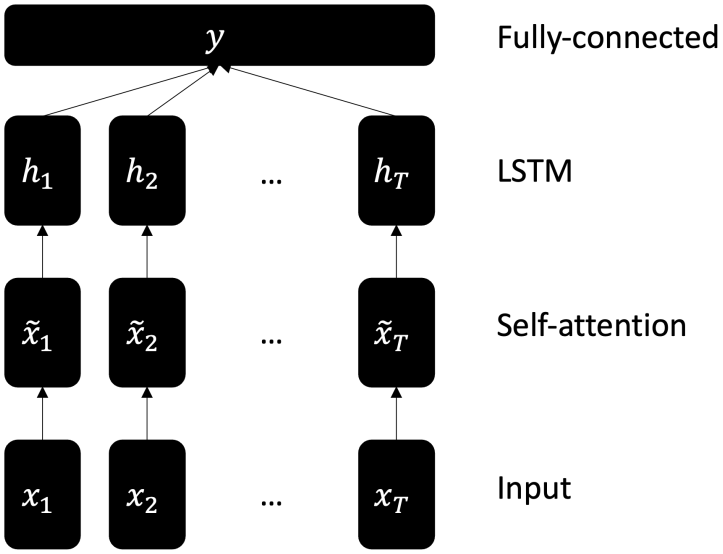


Figure 4.1: **AttnLSTM Architecture**.

The reason why self-attention is chosen over other attention mechanisms is because a variable in the multivariate time series not only depends on its past values, but also on other variables at the same time step. Self-attention mechanisms are better capable at capturing complex temporal dependencies, both short-term and long-term, as shown in the Transformer model. Most importantly, self-attention

provides some interpretability into which parts of the input sequence are most relevant for predicting future values at each time step.

This architecture of AttnLSTM as shown in Figure 4.1 allows the attention mechanism to play a significant role in shaping the initial representations of the input sequence. AttnLSTM takes inputs as a sequence of time-ordered feature vectors of length T, $x = x_1, x_2, \ldots, x_T$, where each x_t is the observations at timestep t for N features. The first stage of the AttnLSTM is to apply self-attention mechanism across the entire sequence to learn weighted dependencies between timesteps. For each timestep, t, AttnLSTM computes a contextual representation $\tilde{x}_t$ as a weighted sum of all input vectors:

$$\tilde{x}_t = \sum_{s=1}^{T} \alpha_{t,s} x_s \tag{4.1}$$

where $\alpha_{t,s}$ is the attention weight that measures the influence of input x_s on the representation at time t. The resulting sequence of attention-enhanced representations is denoted: $\tilde{x} = \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_T$. Subsequently, the LSTM layer learns the attention weighted sequence, $\tilde{x}$, refining the ability of the model to selectively attend to relevant temporal dependencies within the data. Consequently enhancing the capability of the model to handle long-term dependencies effectively. The outputs of the LSTM – the final hidden state, h_T, is passed to a fully-connected layer to map the learned representation to the prediction target.

The use of attention mechanisms in conjunction with LSTMs serves the dual purpose of preventing the model from assigning disproportionate weight to specific input features, while facilitating the establishment of meaningful correlations between input and target features [155].

4.1.2 AttnED - Attention-based Encoder-Decoder

In AttnED[2], LSTMs are used in both the encoder and the decoder, with a self-attention in between the encoder and the decoder, as shown in Figure 4.2. This

[2]Using AttnED with SHAP for predicting and explaining HAid usage has been submitted as a paper and accepted at the AI4H workshop of SITIS2022 [135].

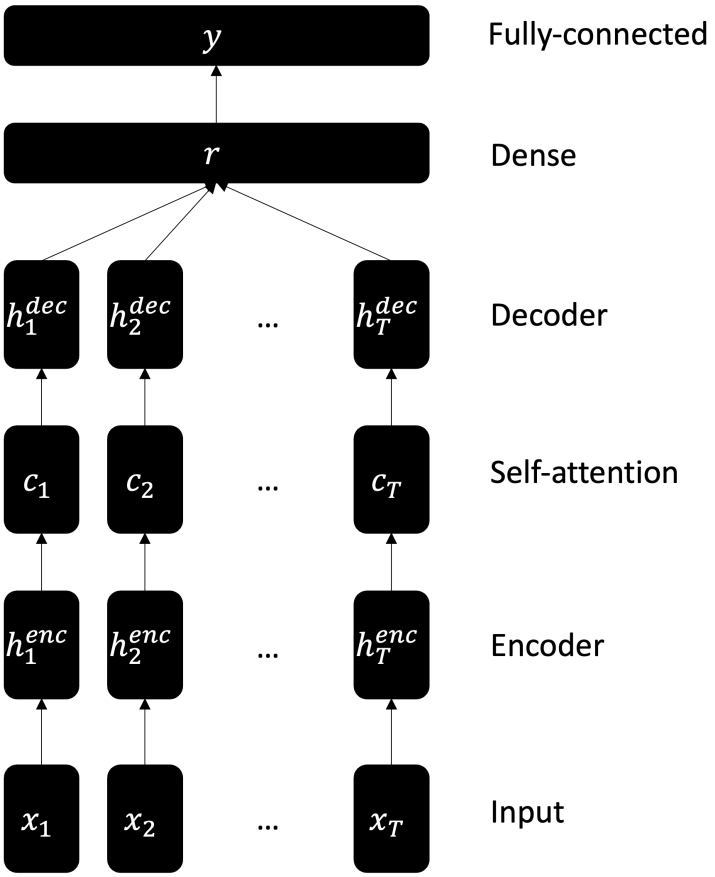


Figure 4.2: **AttnED Architecture**.
The encoder uses an LSTM to encode the input sequences into a vector with fixed dimensionality, while the decoder uses another LSTM to decode the target sequences from the fixed vectors.

architecture is largely similar to Yuan *et al.* [160] (*i.e.,* stand-alone LSTMs are used in both the encoder and the decoder with an attention mechanism in between the encoder and the decoder). However, it differs in the specific implementation of the attention mechanism, where Yuan *et al.* employed a global attention but a self-attention is employed in this work. As mentioned in section 4.1.1, self-attention is chosen in this work because they are better capable at capturing complex temporal dependencies, both short-term and long-term.

AttnED also takes inputs as a sequence of time-ordered feature vectors of length T, $x = x_1, x_2, \ldots, x_T$, where each x_t is the observations at timestep t for N features. The Encoder-LSTM in AttnED is used to learn the time dependencies in the input multivariate time series data and returns the hidden state vectors,

i.e., $H^{enc} = h_1^{enc}, h_2^{enc}, \ldots, h_T^{enc}$, where each h_t^{enc} represents the encoding of the input at time t. A self-attention mechanism is then applied to the hidden state vectors of the Encoder-LSTM, H^{enc}, to learn how each feature of the multivariate time series depends on other features:

$$c = \sum_{t=1}^{T} \alpha_{t,s} h_t^{enc} \tag{4.2}$$

where c is the context vector that captures a weighted combination of the encoder states. A Decoder-LSTM then generates forecasting values using learned context vector, c, as the inputs and produces a sequence of hidden states, $H^{dec} = h_1 dec, h_2 dec, \ldots, h_T dec$. Following the Decoder-LSTM, a dense layer is used to fully connect the hidden states to learn complex relationships among the features extracted from previous layers, followed by another fully-connected layer that generates the final predictions.

This means that the difference between AttnED and AttnLSTM is that rather than obtaining attention weights directly from the input sequences as in AttnLSTM, the attention weights are obtained from the hidden state vectors of the encoder LSTM in AttnED. Applying the attention mechanism to the hidden states rather than directly to the inputs could offer several key advantages, particularly in sequence-based tasks. One primary reason is that hidden states encapsulate a richer representation of the input data [134], as they are derived from intermediate layers of a neural network that have already processed and transformed the raw inputs. This should allow the attention mechanism to focus on more abstract and meaningful features, rather than the potentially noisy and less informative raw inputs. Furthermore, using hidden states should enable the model to attend to different aspects of the input sequence dynamically, based on the context provided by previous and subsequent elements in the sequence [11, 155]. This dynamic contextualisation should enhance the ability of AttnED to capture long-range dependencies in the data [11]. By contrast, applying attention directly to the inputs as in AttnLSTM was not exploit these contextual relationships as effectively, potentially limiting the performance of the model. Therefore, attention mechanisms applied to hidden states provide a more robust and contextually

aware means of selectively emphasising relevant information, hopefully leading to more accurate outputs.

4.1.3 FocusLearn - An Interpretable DNN for Multivariate Time Series Problems

In this section, we introduce a novel interpretable model named FocusLearn[3] for time series analysis. FocusLearn seeks to outperform NAM in terms of predictive ability, and match the predictive performance of DNNs, while retaining the commendable interpretability of NAM, in order to offer a transparent understanding of the model's decision process, particularly regarding the specific impact of each feature, contributing to increased interpretability and accountability.

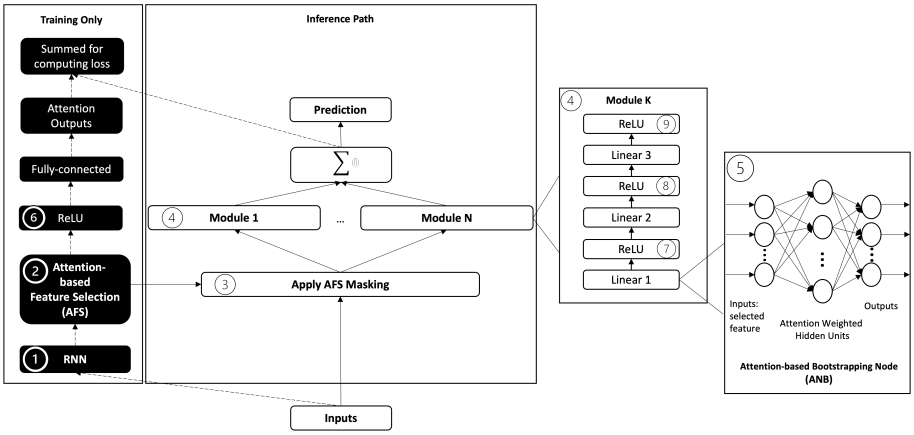


Figure 4.3: **FocusLearn Architecture**.
FocusLearn consists of two main paths (training and inference) with five main components: a ① *Recurrent Neural Network (RNN)* (see section 4.1.3.1), an ② *Attention-based Feature Selection (AFS)* (see section 4.1.3.2), and a group of ④ *interpretable Modular Neural Networks (MNNs)* that learns top n input features selected by the AFS (see section 4.1.3.3). The ⑤ *Attention-based Node Bootstrapping (ANB)* in each *module*'s first layer is weighted by the *AFS*'s attention weights (see section 4.1.3.3.1). The outputs of MNNs are then summed together and serve as predictions. All components are required for training, and to maintain the interpretability of FocusLearn, only components in the *inference path* box are used during the inference step (see section 4.1.3.7).

[3]FocusLearn has been accepted at the IJCNN 2024 conference, doi: `10.1109/IJCNN60899.2024.10651481`

4.1.3.1 Recurrent Neural Network

Inputs are first processed by the RNN component, shown in ① in Figure 4.3, which learns the time-dependencies in the input data. RNNs are a key model for time series analysis due to their ability to capture the non-linearity of time series data and model complex patterns. Their recurrent connections allow RNNs to retain information from previous time steps, enabling the model to recognise and learn meaningful temporal dependencies in sequential data.

The RNN component of FocusLearn, by default an LSTM network, can be easily replaced by alternative architectures like GRU or BLSTM for enhanced flexibility (see ablation experiments in section 4.3.3). Raw inputs, x_t, are processed by the LSTM through a series of gating mechanisms as shown in Equations (4.3)[58], where x_t is the raw input sub-sequence at current timestep t, and i_t, f_t, and o_t are the input, forget, and output gates, respectively. w_a and w_h are the weights and b is the bias.

$$\begin{cases} f_t = \sigma(w_a x_t + w_h h_{t-1} + b) \\ i_t = \sigma(w_a x_t + w_h h_{t-1} + b) \\ s_t = \tanh(w_a x_t + w_h h_{t-1} + b) \\ c_t = f_t \bigodot c_{t-1} + i_t \bigodot s_t \\ o_t = \sigma(w_x x_t + w_h h_{t-1} + b) \\ h_t = \tanh(c_t) \bigodot o_t \end{cases} \tag{4.3}$$

To address challenges related to vanishing or exploding gradients and the instability of hidden representations resulting from the dynamic nature of state updates in the RNN, layer normalisation is applied to the RNN outputs. Layer normalisation [10] operates by independently normalising the activations of each neuron across the RNN outputs by re-centering and re-scaling the activation values to achieve zero mean and unit variance for each neuron.

In practice, the layer normalisation statistics are computed over all the hidden units in the same layer as in Equations (4.4) and (4.5).

$$\mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \tag{4.4}$$

$$\sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^t - \mu^t)^2} \tag{4.5}$$

where μ and σ are the normalisation terms, H is the number of hidden units in a layer, and a^t is the vector representation of the summed inputs in the recurrent layer. The a_t is computed from the current input x_t and the previous hidden state h_{t-1} as in Equation (4.6), where w_h and w_x are the weights for h_{t-1} and x_t, respectively.

$$a^t = w_h h_{t-1} + w_x x_t \tag{4.6}$$

Batch normalisation [66] and layer normalisation [10] are two methods for preventing the problem of vanishing or exploding gradients and unstable hidden representations due to the highly dynamic nature of state updates in an RNN. In batch normalisation, summed inputs to each hidden unit are normalised over all training layers. However, this may become problematic with RNNs because the statistics of the entire batch may not be representative of the unique characteristics of each time step. This can lead to sub-optimal normalisation, especially in scenarios where the sequence lengths vary. Layer normalisation made improvements upon this by only computing the normalisation statistics over all the hidden units in the same layer and only depends on the summed inputs to a layer at the current time step. Therefore, by maintaining a stable distribution of activation values, layer normalisation facilitates the effective propagation of gradients through the network during training, as well as at stabilising the hidden state dynamics in RNNs [10].

4.1.3.2 Attention-based Feature Selection

The Attention-based Feature Selection (AFS) component of FocusLearn is inspired by the Multi-head attention [144]. In the original version of the Multi-head attention, each attention head attends to an input sub-sequence as a way to reduce the computational complexity and $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ can be linearly projected with different projections to d_q and d_v dimensions, respectively.

However, as we wish to use the Multi-head attention mechanism in FocusLearn as a feature selection tool, we cannot have each head attend to an input sub-sequence. This is because using the attention weights of the multi-head attention would not be indicative of importance of a particular feature, given that different weights assigned to different parts of the input data are used in each head [87]. Therefore, instead of making each head attend to a sub-sequence of the input, the multi-head attention is modified here so each attention head computes the attention weight of each input feature independently. The details of the modifications are discussed in the following section.

4.1.3.2.1 Applying Attention-based Feature Selection in FocusLearn

Given the number of input features, N, and the number of hidden states of the RNN component, s, the dimension of the final states calculated from the RNN component, H, is $N \times s$. Each hidden state, H_i, where $i \in 1, \cdots, N$, corresponds to each input feature, x_i. N number of attention heads is then applied to each H_i as shown in Figure 4.4. Mathematically, for each feature i, the attention head computes the attention head output, AH_i, as shown in Equation (4.7), where $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_q \times d_v}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_k \times d_v}$, and $\boldsymbol{W}_i^V \in \mathbb{R}^{d_q \times d_v}$ are the projections of attention head-specific weights for $\boldsymbol{Q}_i$, $\boldsymbol{K}_i$, $\boldsymbol{V}_i$, respectively, and $\boldsymbol{Q}_i$, $\boldsymbol{K}_i$, $\boldsymbol{V}_i$ are the query, key, and value matrices derived from H_i of x_i.

$$AH_i = \text{Attention}(\boldsymbol{Q}_i\boldsymbol{W}_i^Q, \boldsymbol{K}_i\boldsymbol{W}_i^K, \boldsymbol{V}_i\boldsymbol{W}_i^V) \tag{4.7}$$

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [AH_1, AH_2, ..., AH_N]\boldsymbol{W}_{AH} \tag{4.8}$$

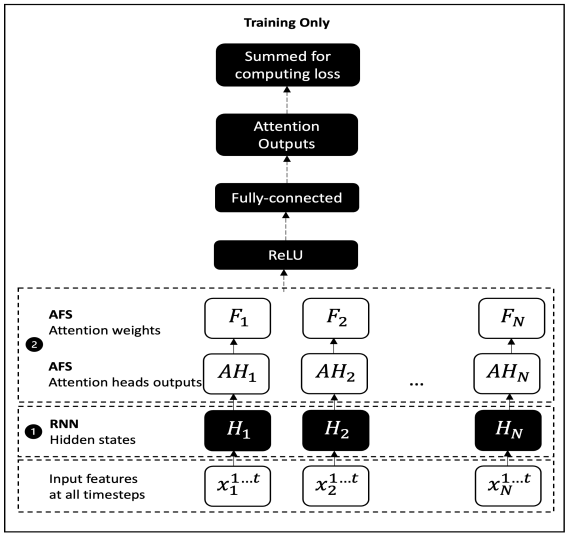


Figure 4.4: **Applying AFS in FocusLearn**.
AFS is component ② in Figure 4.3. Each attention head focuses on each hidden state of every feature. The attention weights generated by all attention heads across all timesteps are then averaged to determine the overall attention weight for each feature.

In Equation (4.8), $\boldsymbol{W}_{AH} \in \mathbb{R}^{(AH \cdot d_v) \times d_q}$ is the attention weight that linearly combines concatenated outputs from all attention heads. However, as previously mentioned, $\boldsymbol{W}_{AH}$ would not be indicative of importance of a particular feature. Therefore, we are utilising the attention head outputs, AH, to compute the attention weights instead. To do this, we first specify that the embedding dimension of the attention mechanism to be the number of input features, N, to ensure each AH_i has the same dimensionality as the inputs, *i.e.,* (L, B, N), where L is the sequence length, and B is the batch size. The sequence length refers to the number of timesteps for the RNN to compute which is the number of observations in the input.

Next, an adaptive feature-wise pooling operation is employed to compute the overall attention weight, w_i, for each feature. This is done by averaging the attention heads output at timestep t, AH_i^t, over the L (sequence length) dimension and calculated as shown in Equation (4.9). This strategy allows AFS to capture complex dependencies and interactions among features, as well as identifies the relative importance of each feature, allowing FocusLearn to effectively focus on the most salient aspects of the data for subsequent processing stages.

$$\boldsymbol{w}_i = \frac{1}{L}\sum_{t=1}^{L} AH_i^t \tag{4.9}$$

An adaptive feature-wise pooling operation was chosen because the adaptive nature allows flexibility in handling inputs with different dimensions, and only the output length needs to be pre-defined, which is one here.

The output of the AFS component is obtained by applying the softmax function to the computed average attention weights, w_i, for normalisation as shown in Equation (4.10). F is feature position-aware, meaning that each F_i corresponds to each input feature x_i as shown in Figure 4.4. This therefore means that F_i allows us to select the top $\boldsymbol{n}$ features directly from the original input at component ③ Select Features in Figure 4.3 on page 72, which is described in detail in the following section.

$$F_i = \text{softmax}(w_i) \tag{4.10}$$

4.1.3.2.2 Selecting Top n Features

As discussed in the previous section, F is feature position-aware so that each F_i corresponds to a specific input feature x_i. To identify and select the top n features from the original input, we follow a systematic approach as follows:

Extract Indices: We begin by extracting the indices of the top n features that exhibit the largest F values. For instance, if the largest F values correspond to features at indices [1, 4, 6], we select these indices for further processing.

Create a Mask: Next, a binary mask is created based on these indices. This mask indicates which features should be retained and which should be ignored. For example, if our input features are indexed from 0 to 10, the mask would be: [0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0], where 1s correspond to the selected features (indices 1, 4, and 6) and 0s indicate features that are to be excluded for further consideration.

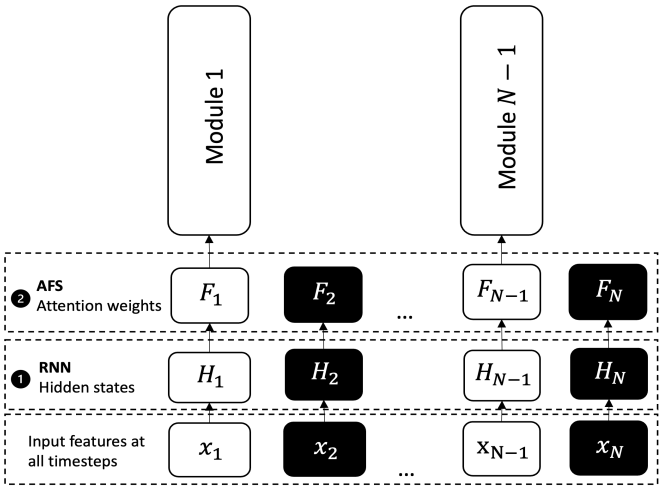


Figure 4.5: **Selecting top n features**.
Showing an example where F_2 and F_n have the smallest values, meaning that feature x_2 and x_n are to be masked out. Modular Neural Networks (MNNs) are then only constructed for feature x_1 and x_{n-1}.

Select Features Using the Mask: Finally, we apply the mask to the original input, effectively filtering out the less important features. By multiplying the original input by the mask, we retain only the selected features. For example, if the original input consists of 10 features, applying the mask results in retaining features 1, 4, and 6 while setting the values of all other features to zero.

This masking operation ensures that only the top n features influence the model's predictions and gradient calculations, effectively filtering the feature space. This process effectively filters the feature space, allowing FocusLearn to concentrate on the most relevant attributes.

Given n selected features, n Modular Neural Networks (MNNs) are then constructed as illustrated in Figure 4.5. The details of MNNs are discussed in the following section. In FocusLearn, the parameter n can either be pre-defined based on prior knowledge or tuned as a hyper-parameter to optimise model performance. This flexibility in setting n enables a tailored approach, accommodating various data sets and modelling requirements. To ensure that gradients are propagated adequately through the MNNs (discussed in section 4.1.3.3), F is sent through a Rectified Linear Unit (ReLU) and a fully-connected layer so that the loss of the RNN-AFS components can be calculated (discussed in section 4.1.3.6).

4.1.3.3 Modular Neural Networks of FocusLearn

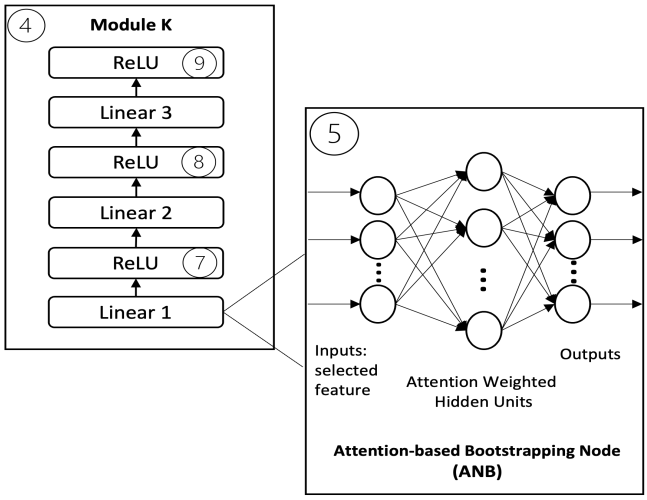


Figure 4.6: **The MNNs component of FocusLearn**.
MNN is component ④ in Figure 4.3. Each module in MNNs learns each selected input feature independently with ANB (component ⑤ in Figure 4.3) acts as a bridge between the AFS and the MNN component.

Each module in the MNNs component ④ of FocusLearn consists of a single selected input feature trained independently by a collection of linear computation and ReLU activations as shown in Figure 4.6. Training each selected input feature independently ensures the impact of each feature on the prediction is independent of other features. Thus, this component methodologically belongs to the GAM family, following Equation (4.11), where y is the target variable, β is the intercept, g is the link function, $\boldsymbol{x} = x_1, x_2, ..., x_n$ is the input (or the selected input in the case of FocusLearn), and each f_i is a univariate shape function with $\mathbb{E}(f_i) = 0$.

$$g(\mathbb{E}(y)) = \beta + f_1(x_1) + f_2(x_2) + \cdots f_n(x_n) \tag{4.11}$$

The MNNs component renders FocusLearn interpretable because each univariate shape function is parameterised by a neural network through a series of linear transformations, as done with NAM. This means that the weights and biases of the neural network are served as the parameters that define the behaviour of the univariate shape function. Each neural network, through its layers and neurons, maps inputs to outputs, and the values of these parameters are learned during training to approximate the desired shape function.

Adding ReLU (⑦, ⑧, and ⑨ in Figure 4.6) into this series of linear transformations also introduces non-linearity [126] to FocusLearn, particularly with a feature's current value and its past value. The ability to capture non-linearity is paramount in time series analysis, where the temporal dependencies and dynamic patterns inherent in the data necessitate the ability of FocusLearn, particularly the MNNs component, to learn complex representations. The inherent non-linearity introduced by ReLU empowers FocusLearn to discern and leverage intricate structures in the input-output mapping, leading to improved predictive performance and a more comprehensive understanding of the underlying temporal dynamics.

4.1.3.3.1 Attention-based Node Bootstrapping

The main difference between the MNNs component of FocusLearn and NAM is that each feature is initially weighted using the attention weights F in FocusLearn, which will be shown to improve predictive performance considerably. To achieve this, we introduce Attention-based Node Bootstrapping (ANB) to each module (see Figure 4.3⑤) to learn the weights based on the inputs factored by the attention weights and shifted by a bias. For each module, Xavier Initialisation [51] initialises the biases (to zeros) and weights at each layer. The previously-computed attention weights for the selected features, $\boldsymbol{F}$, are then multiplied by the initialised weights, $\boldsymbol{W}_{\text{init}}$, as in Equation (4.12). Finally, the ReLU activation function is applied to each ANB for each scalar input x shifted by a bias, b, as shown in Equation (4.13). ANB is applied in the first layer of each module (Linear 1 in Figure 4.3④), so each F_i can be used directly. These individual networks are then trained jointly with the RNN and AFS components using back-propagation, to ensure the MNNs components are learned rather than discovered by chance.

$$\boldsymbol{W}_{\text{mod}_i} = \boldsymbol{W}_{\text{init}_i} * F_i \tag{4.12}$$

$$ANB(x) = \text{ReLU}((x - b) * \boldsymbol{W}_{\text{mod}}) \tag{4.13}$$

ReLU was chosen for two reasons. Firstly, due to its aptitude for capturing non-linear and intricate relationships within the data. Secondly, due to its efficacy in mitigating the vanishing gradient problem [51, 52], a pervasive issue in deep learning optimisation. By allowing positive gradients to pass unimpeded during backpropagation for positive input values, ReLU helps prevent the exponential decay of gradients encountered in traditional activation functions, thereby facilitating more effective weight updates and ultimately enhancing the convergence of the model.

4.1.3.4 Explanation Visualisations of FocusLearn

As with NAM, the interpretability of FocusLearn comes from visualising the shape functions for each modular network. Since each selected feature is learned independently, the shape functions show the exact contribution of a module (*i.e.,* feature) to a prediction, and the relationship between $f_i(x_i)$ and x_i can be plotted to visualise and describes exactly how FocusLearn computes a prediction.

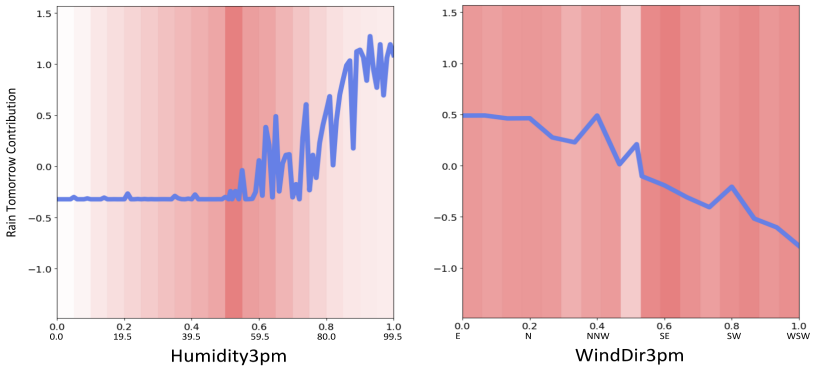


Figure 4.7: **An example of the graphs learned by FocusLearn for Weather**. Predicting whether it will rain tomorrow (classification) on the *Weather* data set. These plots show two features selected by AFS– Humidity at 3pm and Wind direction at 3pm, where selected features with normalised and original values are on the x-axis, and contribution towards the prediction are on the y-axis.

An example of the visualisation is shown in Figure 4.7, with the full explanation visualisations of FocusLearn provided in Appendix B. In the visualisations, each shape function for each feature is plotted with its corresponding normalised data density, as in NAM. Feature values are on the x-axis, and feature contributions are on the y-axis. The shape functions are presented using a blue line,

allowing us to see the contribution of each feature varies with different feature values. The normalised data density are presented in the form of red bars, where a darker red denotes a higher data density. This allows us to see if the model had an adequate amount of data to learn the appropriate shape functions.

Taking Figure 4.7 as an example, the shape function plot for Humidity at 3pm shows that humidity from 60 to 99% at 3pm today will likely lead to *rain* tomorrow. The start of the upward trend of the contribution of humidity occurs in a region of high data density. The plot for Wind Direction at 3pm shows higher chances of rain tomorrow when there is a northerly wind today, compared with lower chances of rain tomorrow when the wind direction changes to a more southerly wind today.

4.1.3.5 Learning Rate Scheduler

Since FocusLearn strongly resembles the Transformer architecture, we are employing the Cosine Annealing learning rate warm-up scheduler [91] to further accelerate convergence, stabilise the training, and improve generalisation for the Adam optimisation algorithms [90]. Unlike traditional approaches of setting the learning rate as a constant or in a decreasing order, the learning rate warm-up technique progressively increases the learning rate from zero to the specified value in the initial training iterations. This strategy proves particularly beneficial for adaptive optimisation algorithms like Adam, which leverage bias correction factors. The warm-up mitigates potential issues associated with higher variance in the first training iterations, ensuring a smoother optimisation trajectory. Additionally, the application of layer normalization techniques in the RNN component, while essential for stabilising activations, can sometimes lead to very high gradients during the initial iterations. The combination of these techniques contributes to a more stable and efficient training process for FocusLearn.

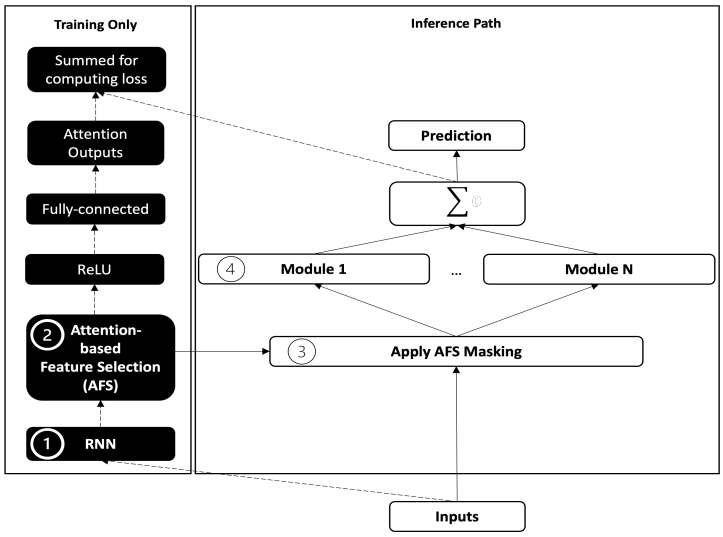


Figure 4.8: **Zoomed in view of two paths of FocusLearn**.

4.1.3.6 Loss Functions

To ensure the network gradients are flowing through all components of FocusLearn and improve FocusLearn's convergence during training, the RNN ①-AFS ② and MNNs components are trained jointly, with each one aiming to minimise the Mean Squared Error Loss for regression tasks and Binary Cross-Entropy with Logits Loss for classification tasks as in Equation (4.14), where y is the target value, $\tilde{y_i}$ is the predicted value, and N is the number of data points.

$$\text{Loss} = \begin{cases} \frac{1}{N}\sum(y_i - \tilde{y_i})^2 \ \textit{for regression} \\ \\ -\frac{1}{N}\sum_{i=1}^{N}(y_i * log(\tilde{y_i}) + (1 - y_i) * log(1 - \tilde{y_i})) \ \textit{for classification} \end{cases} \tag{4.14}$$

As Figure 4.8 shows, there are two paths for the inputs to traverse: training and inference. In the training box (left), there are a ReLU and a fully-connected layers implemented after the AFS component. The *ReLU* and the *Fully-connected* layers are added to ensure the gradients are back-propagating back to the inputs and the RNN ①-AFS ② components are learned during the training phase. The

loss of the produced *Attention Outputs* are summed together with the loss of the *Modules Outputs* as the final loss for FocusLearn to minimise, as in equation 4.15.

$$\text{Loss}_{\text{FocusLearn}} = \text{Loss}_{\text{RNN-AFS}} + \text{Loss}_{\text{mod}} \tag{4.15}$$

4.1.3.7 Dual-Paths of FocusLearn

All components of FocusLearn are required in the training phase, but only the inference path (right) is required at inference time. To ensure the explainability of FocusLearn, the inference path includes by construction only the MNN component. This means that at inference time components linked by dashed arrows in the *training only* box of Figure 4.8 are omitted. Therefore, the RNN ①-AFS ② components exclusively handle feature selection during training. Then once the top n features are selected, FocusLearn becomes as explainable as NAM, as it no longer relies on the excluded components for inference.

4.2 Prediction Results On Time Series Data

In this study, XGBoost is considered interpretable when the number of trees remains below 50. When the tuned model exceeds this threshold, it is treated as a black-box model due to the cumulative complexity of multiple interacting trees, which makes it infeasible to extract meaningful, human-understandable rules or feature effects. The consequence of this is twofold. First, comparisons against such high-tree-count XGBoost models serve as a benchmark for predictive performance against a strong but non-interpretable baseline. Second – and more critically – if the proposed interpretable model (FocusLearn) is able to match or outperform this complex version of XGBoost, it provides evidence that strong prediction performance does not have to come at the expense of interpretability. This reinforces the practical value of the proposed model in settings where both accuracy and transparency are essential.

Table 4.1: **Test set average results comparison of FocusLearn with other methods on four data sets for time series regression and classification.**
Means and standard deviations are reported from 5 independent training and evaluation runs, each with different random initialisations using the same optimal hyper-parameters (see Appendix C.1) for each task. For regression tasks, lower results are better (↓). For classification tasks, higher results are better (↑). Regression results are ranked by the SMAPE and classification results are ranked by Accuracy.

		Regression ↓				
		SMAPE	WAPE	MASE	Training Time	Training Time %
Air	AttnED	**0.2547***	**0.1578****	1.1918*	14,619.98	1,220.26
	FocusLearn	0.2640**	0.1637**	**0.1654****	**1,198.10**	100.00
	XGBoost-No Impute(50)	0.2704**	0.1706**	0.1706**	20.92	1.75
	AttnLSTM	0.2852**	0.1887**	1.4250*	4,260.25	355.58
	XGBoost-Impute(50)	0.2969**	0.1965**	1.4841**	20.92	1.75
	LSTM	0.2935*	0.1772**	0.1774**	1,289.57	107.63
	NAM	0.5221*	0.4206*	0.4166*	3,598.02	300.31
	SPAM	0.5231^{+}	0.2778^{+}	0.2765^{+}	1,571.26	131.15
Energy	AttnLSTM	**0.3530***	**0.4421***	1.4065^{+}	96.45	155.97
	FocusLearn	0.3704*	0.4557*	**0.5503***	119.64	193.47
	AttnED	0.4630*	0.70*	2.2266*	290.10	469.11
	SPAM	0.4853*	0.5486	0.6557	70.20	113.52
	XGBoost-No Impute(10)	0.7633**	1.0533**	3.3509**	0.06	0.10
	NAM	1.3801^{+}	1.4320^{+}	1.7856^{+}	157.11	254.06
	LSTM	1.5022**	0.8922**	2.8511**	**61.84**	100.00
		Classification ↑				
		Accuracy	F1	AUC	Training Time	Training Time %
EEG	**FocusLearn**	**0.8203***	**0.8131***	**0.8203***	**24.28**	100.00
	AttnED	0.7047^{+}	0.0890^{+}	0.5240^{+}	195.65	805.81
	XGBoost-No Impute(5)	0.6959**	0.6169*	0.6824**	0.02	0.08
	NAM	0.5430**	0.0115**	0.5020**	136.56	562.44
	SPAM	0.5000*	0.0159*	0.5002*	115.61	476.15
	LSTM	0.5000**	0.2667^{+}	0.5**	53.53	220.47
	AttnLSTM	0.3906^{+}	0.1149^{+}	0.4795*	89.90	370.26
Weather	AttnLSTM	**0.8624****	0.5494*	0.7022*	1,770.23	266.42
	AttnED	0.8614**	0.5394*	0.6961*	4,452.03	670.02
	FocusLearn	0.8518**	**0.7857***	**0.8008****	1,192.00	179.39
	XGBoost-No Impute(5)	0.8473*	0.5508*	0.7009*	0.23	0.03
	XGBoost-Impute(5)	0.8468*	0.5583*	0.7046**	0.21	0.03
	LSTM	0.8419**	0.4592*	0.6509**	**664.46**	100.00
	SPAM	0.8369**	0.4305^{+}	0.6444*	1,231.40	185.32
	NAM	0.8169**	0.2442^{+}	0.5687*	1,552.91	233.71

Interpretable model. Non-interpretable model.
**: Variance across experiments is < 0.01.
*: Variance across experiments is < 0.05.
$^{+}$: Variance across experiments is > 0.05.

Note: The number in parentheses after XGBoost's name is the optimal number of trees for it; XGBoost is deemed as non-interpretable if the number of trees exceeds 50. Data are pre-processed differently for XGBoost. More specifically, XGBoost supports missing values by default, whereas missing values are imputed for training with other models. Therefore, to ensure a fair comparison with other models, missing values are imputed for training with XGBoost-Impute if there are missing values.
Since XGBoost requires significantly less training time than neural network models, it is consistently the fastest; therefore, training time percentage are also reported which is relative to the fastest neural network model.

4.2.1 AttnLSTM usually predicts better than an LSTM

Hypothesis: It is hypothesised that combining LSTM with attention mechanisms will result in a time series model that outperforms a stand-alone LSTM in terms of prediction performance measured by metrics such as Accuracy for classification problems and Mean Absolute Percentage Error (MAPE) for regression problems.

Table 4.1 shows that AttnLSTM significantly outperformed LSTM in the *Energy* and *Weather* data sets across all evaluation metrics. In particular, AttnLSTM achieved best Symmetric Mean Absolute Percentage Error (SMAPE) and Weighted Absolute Percentage Error (WAPE) in the *Energy* data set, and the best accuracy in the *Weather* data set. On the other hand, LSTM is the worst performer in the *Energy* data set in terms of SMAPE, where AttnLSTM outperformed LSTM in SMAPE by 76.5%, 50.4% in WAPE, and 50.7% in MASE. LSTM is also one of the bottom performing model in the *Weather* data set, with AttnLSTM outperformed LSTM in accuracy by 2.4%, 16.4% in F1 score, and 7.3% in AUC.

In the *Air* data set, the difference in SMAPE and WAPE between AttnLSTM and LSTM is minimal, with AttnLSTM achieved a better SMAPE than LSTM (by 2.8%), and LSTM achieved a better WAPE than AttnLSTM (by 6.1%). However, LSTM achieved a far better MASE than AttnLSTM by 87.55%. Examining the prediction visualisations in Figure 4.9, predicted values of LSTM also seem to align better with the true values than those of AttnLSTM.

Table 4.1 shows that AttnLSTM not only achieved a much worse performance than LSTM in the *EEG* data set, it is also the worst performer amongst all models. It is worth to note that LSTM only achieved 50% accuracy and AUC, suggesting that LSTM is performing at a level of random choice, which is typically considered very poor performance, especially for a binary classification task. An accuracy of 50% suggests that LSTM is correct half of the time, and this is equivalent to random guessing in a binary classification scenario. An AUC of 50% suggests that LSTM has no discrimination capacity between two classes. With a lower accuracy,

F1, and AUC score, it shows that AttnLSTM performs even below random choice. This means that AttnLSTM is consistently making incorrect predictions with the *EEG* data set, which is worse than guessing randomly.

Overall it can be said that AttnLSTM can achieve a better predictive performance than LSTM, but in specific contexts. Notably, AttnLSTM emerged as the best performer in the larger binary classification task (*Weather* has 137,910 instances), suggesting its ability to leverage the attention mechanism effectively when substantial data is available. However, AttnLSTM struggled with the smaller binary classification task (*EEG* has 14,892 instances), where its performance was less consistent. This could be due to the fact that attention mechanism needs more data, with more diverse information, to learn the complex patterns in the data. Also, as attention mechanism selectively focus on the most relevant parts of the input sequence, attention mechanism would become more beneficial when there is a lot of information to sift through. Furthermore, the *EEG* data set contains highly correlated input variables (correlation matrix shown in Appendix D.3.1), making it challenging for AttnLSTM to extract meaningful patterns, leading to its poor performance. The *Weather* data set, being large and relatively structured, allowed AttnLSTM to leverage its attention mechanism effectively, achieving the highest accuracy among all models.

In regression tasks, AttnLSTM also showed a stronger capability overall; although LSTM outperformed AttnLSTM in the *Air* data set in terms of MASE, the difference was minimal in terms of SMAPE and WAPE. This indicates that AttnLSTM may possess better generalisation and robustness in handling diverse data scenarios, particularly in larger or more complex data sets. Furthermore, the *Energy* data set exhibits strong seasonality, as indicated by its time series decomposition plot (Appendix D.1.1), where the seasonal component shows pronounced fluctuations with frequent ups and downs. This variability may have allowed AttnLSTM 's attention mechanism to effectively capture important patterns by selectively focusing on relevant past time steps, leading to significant performance improvements. In contrast, the seasonal pattern in the *Air* data set (Appendix D.2.1) is much smoother, with more gradual variations over time. This relative smoothness may have reduced the advantage of the attention mech-

anism, resulting in AttnLSTM performing less effectively in *Air* compared to its performance in *Energy*.

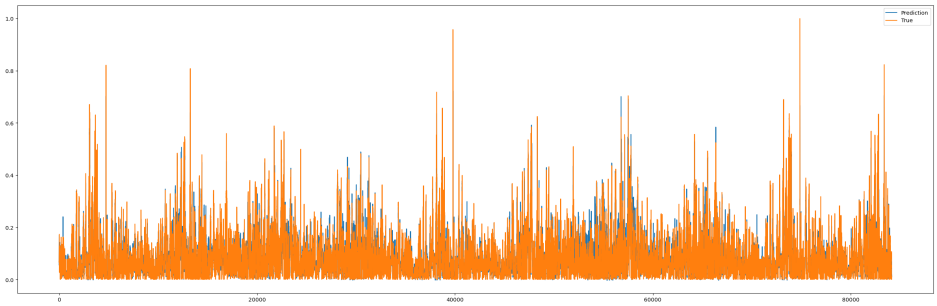
The findings indicate that AttnLSTM is generally a superior model in predictive performance, especially in large data sets, supporting the hypothesis. However, its effectiveness is strongly influenced by data characteristics. In data sets with limited data points and smoother seasonal patterns, LSTM may still be more reliable.

4.2.2 AttnED sometimes predicts better than AttnLSTM

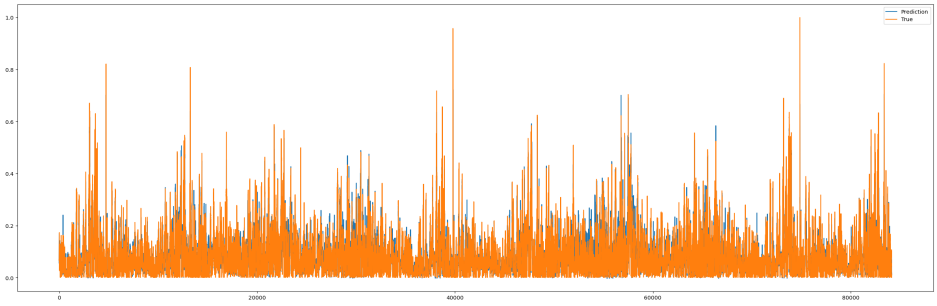
Hypothesis: It is hypothesised employing an LSTM to first capture the data representations and temporal dependencies, followed by an attention mechanism to learn from the hidden states may be more effective than using an attention mechanism initially to process the inputs, followed by an LSTM, in terms of prediction performance measured by metrics such as Accuracy for classification problems and Mean Absolute Percentage Error (MAPE) for regression problems.

The experimental results presented in Table 4.1 indicate that AttnED outperformed AttnLSTM in both the *Air* and *EEG* data sets across all evaluated metrics. In particular, AttnED achieved the best results in SMAPE and WAPE metrics for the *Air* data set, demonstrating its effectiveness in handling large regression data sets. Furthermore, in the *EEG* data set, a smaller classification data set, AttnED continued to deliver a better performance than AttnLSTM, with AttnLSTM being the worst performer in the *EEG* data set, suggesting that the architecture of AttnED is well-suited for various types of data.

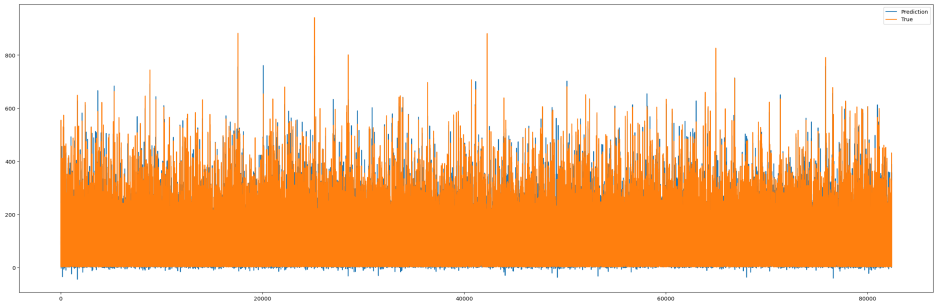
Although AttnLSTM outperformed AttnED in the *Weather* data set, the observed differences in performance metrics were minimal, where the difference in Accuracy was only 0.12%, with F1 and AUC metrics showing differences of 1.8% and 0.87%, respectively. Table 4.1 shows that AttnLSTM also outperformed AttnED in the *Energy* data set, however, the prediction visualisations in Figure 4.10 shows that AttnLSTM's predicted time series appears quite flat and fails to capture the peaks, whereas AttnED, despite overestimating the peaks, demonstrates a better ability to reflect the overall upward and downward patterns of the data.



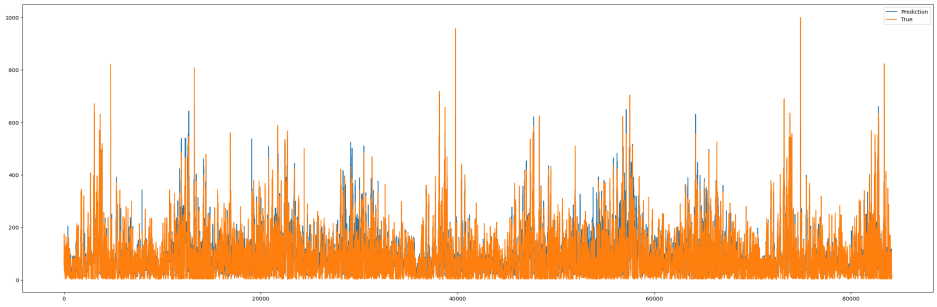
(a) AttnED



(b) FocusLearn

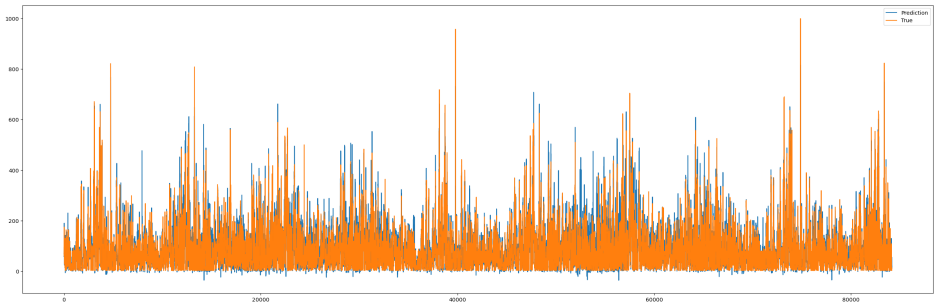


(c) XGBoost-No Impute

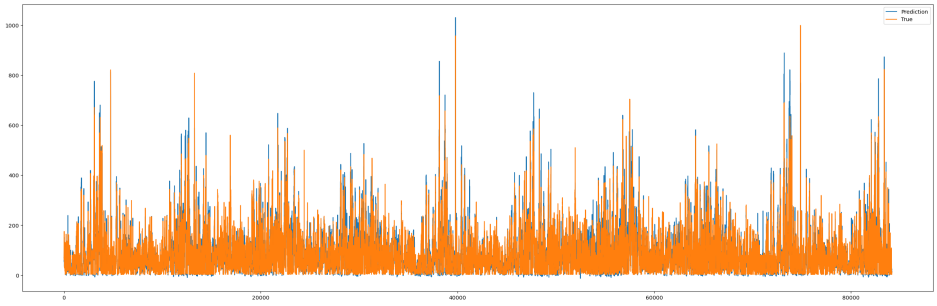


(d) AttnLSTM

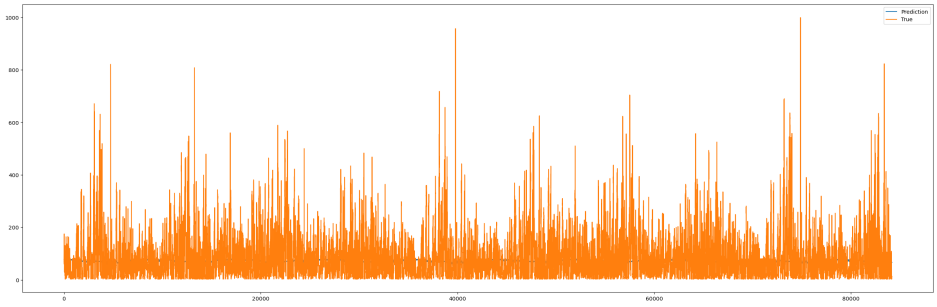
Figure 4.9: **Prediction Visualisations for the *Air* Data Set.** X-axis represents predicted timesteps and y-axis represents feature values. True future time series are in orange and predicted time series are in blue. Data are pre-processed differently for XGBoost. More specifically, XGBoost supports missing values by default, whereas missing values are imputed for training with other models. Therefore, to ensure a fair comparison with other models, missing values are imputed for training with XGBoost-Impute if there are missing values.



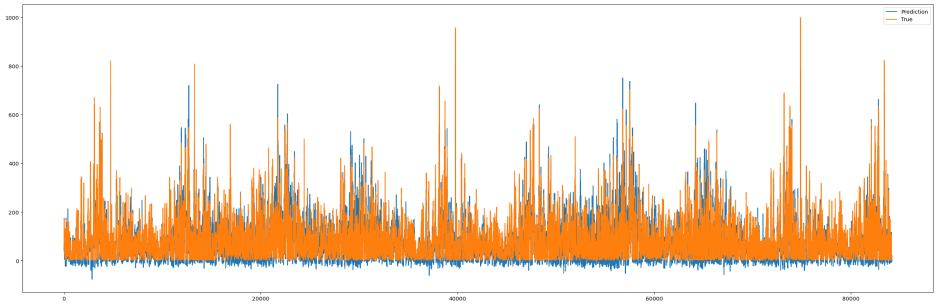
(e) XGBoost-Impute



(f) LSTM

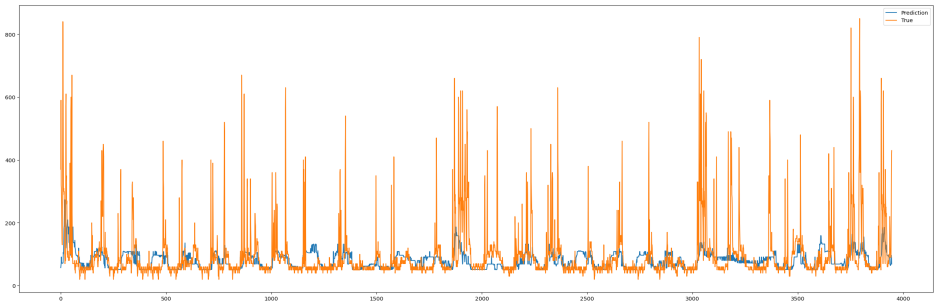


(g) NAM

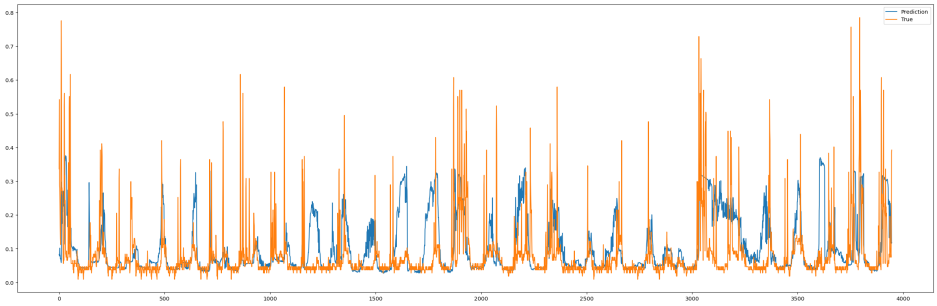


(h) SPAM

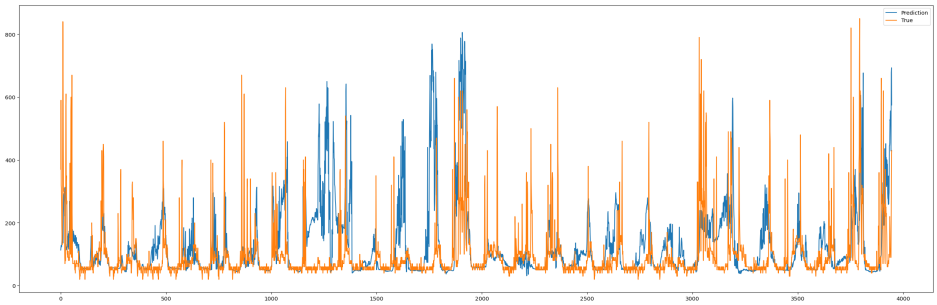
Figure 4.9: **Prediction Visualisations for the *Air* Data Set (cont.).** X-axis represents predicted timesteps and y-axis represents feature values. True future time series are in orange and predicted time series are in blue. Data are pre-processed differently for XGBoost. More specifically, XGBoost supports missing values by default, whereas missing values are imputed for training with other models. Therefore, to ensure a fair comparison with other models, missing values are imputed for training with XGBoost-Impute if there are missing values.



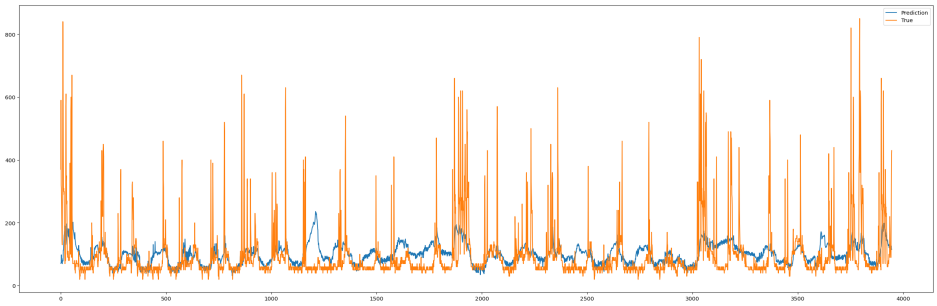
(a) AttnLSTM



(b) FocusLearn

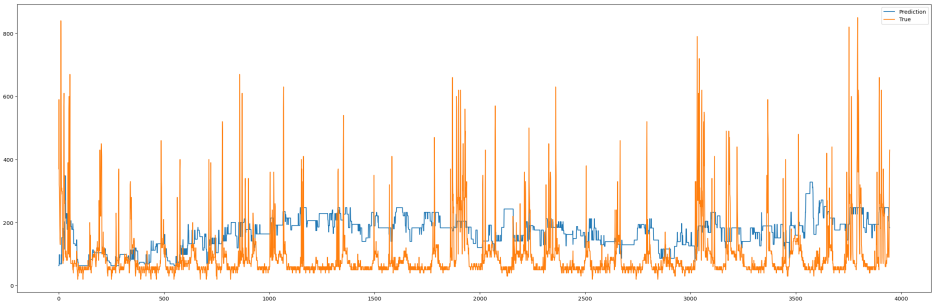


(c) AttnED

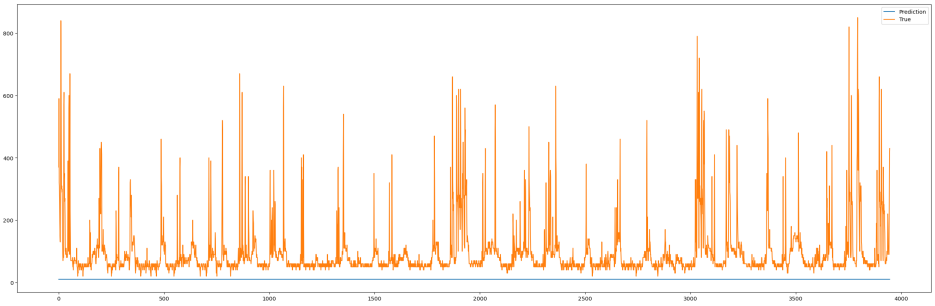


(d) SPAM

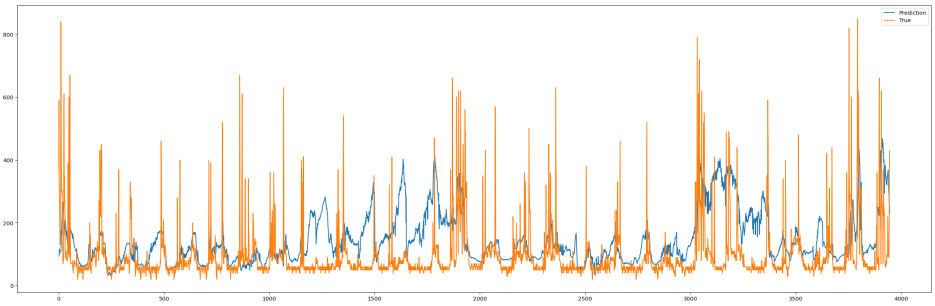
Figure 4.10: **Prediction Visualisations for the *Energy* Data Set.** X-axis represents predicted timesteps and y-axis represents feature values. True future time series are in orange and predicted time series are in blue.



(e) XGBoost-No Impute



(f) NAM



(g) LSTM

Figure 4.10: **Prediction Visualisations for the *Energy* Data Set (cont.).** X-axis represents predicted timesteps and y-axis represents feature values. True future time series are in orange and predicted time series are in blue.

Despite the fact that AttnED consistently ranking among the top three performers across all data sets in terms of SMAPE (for regression tasks) and Accuracy (for classification tasks), AttnED's predictive performance on other metrics is less impressive. The *EEG* data set, being noisy and with highly correlated input variables, presented challenges for both AttnLSTM and AttnED. In particular, AttnED achieved an F1 score of just 0.0890, placing it among the lowest-performing models in terms of F1 score. In the context of a binary classification task, such a low F1 score suggests that AttnED struggled to distinguish between the classes and was essentially making random guesses. Similarly, AttnLSTM also exhibits inconsistent performance. For instance, AttnLSTM is the best performer in the *Energy* data set (small regression task) and the *Weather* data set (large classification task), it faced challenges in the *EEG* data set (small classification tasks), where it was the lowest performer among the models evaluated in terms of Accuracy and AUC.

The experimental results suggest that employing an LSTM to first capture the data representations and temporal dependencies, followed by an attention mechanism to learn from the hidden states, as in AttnED, may be more effective than using an attention mechanism initially to process the inputs, followed by an LSTM, as in AttnLSTM. This configuration of AttnED appears more advantageous, as demonstrated by its prediction results compared to that of AttnLSTM, because the LSTM is inherently well-suited for modelling sequential data and capturing long-term dependencies, which are important for tasks involving time series. By first allowing the LSTM to learn these dependencies, the attention mechanism can then focus more effectively on the most relevant hidden states, refining the understanding of the model of key information in the input sequence. In contrast, applying attention to the inputs before the LSTM may limit the ability of the model to fully exploit temporal patterns, as the attention mechanism might not prioritise the most important temporal relationships before the LSTM has processed the sequence.

4.2.3 FocusLearn can predict better than interpretable models

Hypothesis: It is hypothesised that the proposed interpretable time series model will achieve better prediction performance than state-of-the-art interpretable models, such as NAM, as measured by metrics such as Accuracy for classification problems and Mean Absolute Percentage Error (MAPE) for regression problems.

Table 4.1 indicates that the predictive performance of FocusLearn substantially and consistently exceeds that of both interpretable models, NAM and SPAM, across all evaluation metrics in both regression and classification tasks. Notably, NAM and SPAM often rank among the worst-performing models, particularly in the *Air* and *Weather* data sets. As shown in Figure 4.9 and Figure 4.10, FocusLearn captures patterns and trends in the *Air* and *Energy* data sets more effectively, with predicted values closely aligning to ground truth, whereas NAM and SPAM fail to learn meaningful patterns, resulting in almost flat-line predictions. This suggests that NAM and SPAM may struggle with capturing the complex temporal dependencies present in these data sets, which FocusLearn handles more effectively.

XGBoost is deemed as an interpretable model when its optimal number of trees remains below 50. Table 4.1 shows that XGBoost meets this criterion for the *Energy*, *EEG*, and *Weather* data sets, where the optimal number of trees is 10, 5, and 5, respectively. Despite its interpretability in these cases, FocusLearn consistently outperforms XGBoost across all data sets and evaluation metrics, with the performance gap being especially clear in the *Energy* and *EEG* data sets. Specifically, FocusLearn improves the SMAPE score by 51%, the WAPE score by 56%, and the MASE score by 85% compared to interpretable XGBoost in the *Energy* data set. The inability of XGBoost to capture the fluctuations in the *Energy* data set, as evidenced in Figure 4.10, suggests that tree-based models may struggle with non-stationary time series that exhibit rapid short-term variations. Similarly, in the *EEG* data set, FocusLearn achieves the best overall performance, even surpassing DNN time series models, highlighting its effectiveness in learning from noisy sequential data.

Although XGBoost achieves results close to FocusLearn in the *Weather* data set, particularly in accuracy (FocusLearn: 0.8518, XGBoost-No Impute: 0.8473, and XGBoost-Impute: 0.8468), FocusLearn significantly outperforms XGBoost in F1 and AUC scores. FocusLearn improves the F1 score by 30% compared to XGBoost-No Impute and by 29% compared to XGBoost-Impute, while achieving a 12% improvement in AUC for both variants. Given that the *Weather* data set was resampled to address class imbalance (original class distribution: 1:0.26), the superior performance of FocusLearn in AUC suggests that it is better at distinguishing between classes under imbalanced conditions.

There are two variants of XGBoost (XGBoost-No Impute and XGBoost-Impute) evaluated here because XGBoost supports missing values by default, whereas missing values are imputed for training with other models. Therefore, missing values are imputed for training with XGBoost-Impute, if there are missing values, to ensure a fair comparison with other models. In the *Weather* data set, results show that the difference in predictive performance between XGBoost-Impute and XGBoost-No Impute is also minimal, suggesting that XGBoost's built-in missing value handling is robust for this data set.

Beyond predictive accuracy, FocusLearn also demonstrates faster computational efficiency because it excludes the RNN-AFS components during inference and uses a reduced set of features. It consistently achieves faster computation times than NAM across all data sets and is faster than SPAM in the *Air*, *EEG*, and *Weather* data sets, making it the fastest neural network for the *Air* and *EEG* data sets. In the *Energy* data set, FocusLearn is 24% faster than NAM, though SPAM remains the fastest by 41%. Similarly, in the *Weather* data set, FocusLearn outperforms NAM by 23% and SPAM by 3.2%. These results indicate that FocusLearn not only improves predictive performance over NAM, SPAM, and XGBoost but also achieves faster inference times, making it a practical choice for real-time applications.

The performance differences across data sets can be attributed to their unique characteristics. The *Energy* data set exhibits strong seasonality, as indicated by a clear daily pattern in its time series decomposition plots (Appendix D.1.1), and high variance, evidenced by large fluctuations in observed values (Appendix D.1.2).

Tree-based models like XGBoost may struggle with such data due to their limited ability to capture long-term dependencies. The *EEG* data set is characterised by noisy sequential data, where the presence of redundant or irrelevant features makes feature selection crucial, as shown by the correlation matrix shown in Appendix D.3.1, which can obscure meaningful patterns. The *Weather* data set presents challenges due to class imbalance and missing data, where FocusLearn 's advantage in learning class distributions and handling missing values effectively results in higher F1 and AUC scores. The poor performance of NAM and SPAM across data sets suggests that these models may be less capable of learning complex, non-linear relationships in sequential data.

In summary, FocusLearn consistently outperforms interpretable models (NAM, SPAM, and interpretable XGBoost) across all data sets, both in predictive accuracy and computational efficiency, further confirming the proposed hypothesis. The analysis suggests that FocusLearn is particularly well-suited for data sets with high variance, strong temporal dependencies, and imbalanced class distributions, where traditional tree-based models and simpler interpretable neural networks may struggle. In particular, combining the attention mechanism with LSTMs for feature selection, as hypothesised, has proven beneficial for the *EEG* data set, where FocusLearn achieved the best performance. These findings emphasise the need for further research into bridging interpretability and performance in time series forecasting models.

4.2.4 FocusLearn can match the performance of non-interpretable models

Hypothesis: It is hypothesised that the proposed interpretable time series model will match or achieve better prediction performance than traditional DNN-based time series models, such as LSTM, as measured by metrics such as Accuracy for classification problems and Mean Absolute Percentage Error (MAPE) for regression problems.

The results presented in Table 4.1 demonstrate that FocusLearn outperforms non-interpretable models (LSTM, AttnLSTM, AttnED, and non-interpretable

XGBoost) across all evaluation metrics in the *EEG* data set, and attains comparable performance in the *Air*, *Energy*, and *Weather* data sets, with FocusLearn ranks second in terms of SMAPE in the *Air* and *Energy* data sets and ranks third in terms of Accuracy in the *Weather* data set.

In the regression tasks involving *Air* and *Energy* data sets, FocusLearn obtained the best performance in terms of MASE compared to all models, while also demonstrating the second-best performance in SMAPE and MAPE. Importantly, the difference in the performance between FocusLearn and the top-performing model (AttnED for *Air* and AttnLSTM for *Energy*) in terms of SMAPE and MAPE metrics is negligible, with only a 3.5% difference in SMAPE and a 3.6% difference in WAPE compared to AttnED in the *Air* data set, and a 4.6% difference in SMAPE and a 3.0% difference in MAPE compared to AttnLSTM in the *Energy* data set. The prediction visualisations for the *Energy* data set in Figure 4.10 show that AttnLSTM tends to underestimate the peaks in the future time series, whereas AttnED consistently overestimates these peaks. FocusLearn, however, can strike a balance between the two sometimes, demonstrating that FocusLearn can sometimes capture the variations without significant over- or underestimation. Furthermore, it is shown that FocusLearn can handle both time series with strong seasonality, as in *Energy*, and with smoother seasonality, as in *Air*.

In classification tasks with *EEG* and *Weather*, FocusLearn achieved the best performance across all evaluation metrics in *EEG*, demonstrating that FocusLearn can handle high-dimensional and inherently noisy data where LSTM, AttnLSTM, and AttnED struggled. In the *Weather* data set, FocusLearn ranked third in terms of Accuracy, closely following AttnLSTM and AttnED, but FocusLearn achieved the best F1 and AUC scores compared to all models by a large margin. More specifically, FocusLearn yields an increase of 30% in F1 and an increase of 12% in AUC compared to AttnLSTM, and FocusLearn yields an increase of 31% in F1 and an increase of 13% in AUC compared to AttnED. The difference in terms of Accuracy between AttnLSTM (the best performer for the *Weather* data set) and FocusLearn is also minimal with only a 1.2% difference between the two.

Finally, FocusLearn also often has a faster computation time compared to the non-interpretable models, with FocusLearn being the fastest neural network model in the *Air* and *EEG* data sets. Although LSTM is the fastest neural network in both *Energy* and *Weather* data sets, it is also the worst performing model in the *Energy* data sets and achieved less than 50% F1 score in the *Weather* data set. While FocusLearn is 48% slower than LSTM in the *Energy* data set and 44% slower than LSTM in the *Weather* data set. Compared to all models, FocusLearn ranks as the third fasted neural network in the *Energy* data set and ranks as the second fasted neural network in the *Weather* data set.

Overall, FocusLearn can match the prediction results of non-interpretable DNN models, and can also outperform them on all metrics sometimes (*i.e., EEG*). FocusLearn also can have a faster computation time compared to non-interpretable DNN models (*i.e.,* AttnLSTM and AttnED).

4.2.5 Summary

Overall, prediction results presented in Table 4.1 along with prediction visualisations for the regression data sets (*Air* and *Energy*) in Figure 4.9 and Figure 4.10 show that FocusLearn consistently ranks among the top three performers across all data sets and metrics, and can capture the up-and-down pattern in the future time series. In contrast, the performance of other DNN models, such as LSTM, AttnLSTM, and AttnED, fluctuates significantly across data sets, underscoring FocusLearn's reliable performance. This consistency is particularly noteworthy given that FocusLearn is an interpretable model, challenging the common notion that interpretable models may sacrifice predictive accuracy. The poor predictive performance of other interpretable models, such as NAM and SPAM, is evident in this study, as they consistently rank among the worst performers across all data sets and metrics. For regression tasks, FocusLearn's predictions are also much more closely aligned with the true values than NAM and SPAM, as demonstrated in Figure 4.9 and Figure 4.10, further confirming its superior predictive power.

Finally, in the case of the *EEG* data set, FocusLearn not only outperforms all other models (both interpretable and non-interpretable) across every metric,

but these models, particularly DNN models, perform exceptionally poorly. For example, AttnED achieved an F1 score as low as 0.0890, LSTM achieved an AUC of just 0.5, and AttnLSTM achieved an accuracy as low as 0.3906. These results emphasise the strong performance of FocusLearn, especially when other models struggle. Additionally, FocusLearn is either the fastest DNN model or close to it, with only XGBoost outperforming it in terms of training time.

4.3 Ablation Experiments on FocusLearn

In this section, we will discuss a series of ablation experiments conducted on FocusLearn to further evaluate its performance. These experiments are designed to demonstrate the effectiveness of various novel ideas introduced in the development of FocusLearn – Attention-based Feature Selection (AFS), Attention-based Node Bootstrapping (ANB), Dual-Paths, as well as the choice of different RNNs and the choice of n (number of features to select). By systematically testing these innovations under different conditions, we aim to provide a deeper understanding of their impact on the overall accuracy, robustness, and generalisation capabilities of FocusLearn. The results obtained from these experiments will offer valuable insights into the strengths and potential limitations of the proposed methods, thereby contributing to a more comprehensive assessment of FocusLearn's ability as a high performance interpretable time series model.

4.3.1 How effective is Attention-based Feature Selection in improving the predictive performance of FocusLearn?

The Attention-based Feature Selection (AFS) component (see component ② in Figure 4.3 on page 72) drives the feature selection in FocusLearn, as learning only the most salient features has been shown to be a useful method of improving interpretability and allocating learning capacity more effectively [21, 53]. Therefore, to validate the effectiveness of the proposed AFS, the effect of FocusLearn with and without feature selection is examined.

Table 4.2: **Test set average results comparison of the AFS component of FocusLearn on four data sets for time series regression and classification.**
Two experiments are compared for each data set – using AFS for feature selection and without any feature selection where all available features are used. For regression tasks, lower results are better (↓). For classification tasks, higher results are better (↑). n represents the number of features.

Regression ↓					
		SMAPE	WAPE	MASE	Training Time
Air	FocusLearn - n = 10	**0.2640**	**0.1637**	**0.1654**	1,198.10
	FocusLearn - n = 16	0.2754	0.1721	0.1723	2,750.10
Energy	FocusLearn - n = 10	**0.3704**	**0.4557**	**0.5503**	119.64
	FocusLearn - n = 29	0.5994	0.6729	0.7925	97.39
Classification ↑					
		Accuracy	F1	AUC	Training Time
EEG	FocusLearn - n = 10	**0.8203**	**0.8131**	**0.8203**	24.28
	FocusLearn - n = 14	0.7853	0.75	0.7852	91.36
Weather	FocusLearn - n = 10	**0.8518**	**0.7857**	**0.8008**	1,192.00
	FocusLearn - n = 21	0.8442	0.4726	0.6567	899.42

The performance of n = 10 (reported in the result section 4.2) and using all available features in the inputs (no feature selection) are compared for each data set (*Air*, *EEG*, *Energy*, and *Weather*) using the evaluation metrics as in section 3.1.1 (*e.g.,* SMAPE, MASE, and WAPE for regression tasks, and Accuracy, F1, and AUC for classification tasks, as well as the training time). The *Air* data set has 16 features, *Energy* has 29 features, *EEG* has 14 features, and *Weather* has 21 features. Optimal hyper-parameters for all data sets selected during hyper-tuning are used for this experiments, and trained for 100 epochs with the Adam optimiser and early call-backs to avoid over-fitting.

Table 4.2 shows that the application of feature selection using AFS in our study led to a significant enhancement in the prediction performance, with the n = 10 experiments consistently outperforming n = all available features in all evaluation metrics for both regression and classification tasks. This marked improvement underscores the effectiveness of AFS, which efficiently reduced the feature space, eliminated irrelevant attributes, and emphasised the most important variables. By focusing FocusLearn's learning process on the most informative features, AFS is able to enhance the predictive performance of FocusLearn. These findings validate the utility of feature selection in refining machine learning models and highlight its potential to improve outcomes in similar research contexts.

4.3.2 How effective is Attention-based Node Bootstrapping in improving the predictive performance of FocusLearn?

In FocusLearn, each feature is initially weighted using the attention weights in the Modular Neural Networks (MNNs) component (see component ④ in Figure 4.3). This is done by applying Attention-based Node Bootstrapping (ANB) to each module (see component ⑤ in Figure 4.3) to learn the weights based on the inputs factored by the attention weights obtained from the AFS component. To test how effective ANB is in improving the predictive performance of FocusLearn, prediction results of FocusLearn with and without ANB are compared using all data sets. As the MNNs component is inspired by NAM, we are also comparing the predictive performance of FocusLearn with ANB and FocusLearn with *exp-centred* hidden unit (ExU), as in NAM [4]. Optimal hyper-parameters for all data sets selected during hyper-tuning are used for this experiments, and trained for 100 epochs with the Adam optimiser and early call-backs to avoid over-fitting.

While ExU was proposed [4] to overcome the limitation of that standard neural networks in NAM, it failed to model the fluctuations in 1-dimensional prediction arrays. Using ExU, the weights are learnt in the logarithmic space with inputs shifted by a bias. Given a scalar input, x, each hidden unit using an active function, f, computes the output, $h(x)$, as shown in Equation (4.16), where w and b are the weight and bias parameters.

$$h(x) = f(e^{w} * (x - b)) \tag{4.16}$$

Table 4.3 shows that FocusLearn with ANB achieved the substantially better prediction results (*i.e.,* lowest of the regression evaluation metrics and highest of the classification evaluation metrics) compared to all experiments for all data sets. Also, FocusLearn with ExU tends to perform the worst, with the only exception in the *Energy* data set. One reason why FocusLearn with ExU performed worse could be attributed to the inadequacy of the ExU in capturing the intricacies of jagginess and fluctuations within the 1-dimensional prediction array, despite the fact that it was designed to do so. Conversely, ANB demonstrates a better ca-

Table 4.3: **Test set average results comparison of the ANB component of FocusLearn on four data sets for time series regression and classification.**
Three experiments are compared for each data set – using ANB in FocusLearn, without using ANB in FocusLearn, and using ExU as proposed in [4] in FocusLearn. For regression tasks, lower results are better (↓). For classification tasks, higher results are better (↑).

Regression ↓					
		SMAPE	MASE	WAPE	Training Time
Air	FocusLearn - With ANB	**0.2640**	**0.1637**	**0.1654**	1,198.10
	FocusLearn - Without ANB	0.5767	0.3816	0.3816	1,605.80
	FocusLearn - With ExU	0.5970	0.4780	0.4791	2,641.17
Energy	FocusLearn - With ANB	**0.3704**	**0.4557**	**0.5503**	119.64
	FocusLearn - With ExU	0.5329	0.5967	0.7411	76.00
	FocusLearn - Without ANB	0.5576	0.6244	0.7762	48.40
Classification ↑					
		Accuracy	F1	AUC	Training Time
EEG	FocusLearn - With ANB	**0.8203**	**0.8131**	**0.8203**	24.28
	FocusLearn - Without ANB	0.6886	0.6257	0.6885	60.05
	FocusLearn - With ExU	0.5983	0.2923	0.4213	59.84
Weather	FocusLearn - With ANB	**0.8518**	**0.7857**	**0.8008**	1,192.00
	FocusLearn - Without ANB	0.6629	0.5574	0.6629	2,082.41
	FocusLearn - With ExU	0.5519	0.2291	0.5519	1,293.99

pability in discerning changes within the 1-dimensional prediction array, thereby yielding more accurate predictions. As can be seen, the proposed ANB is effective at improving the performance of FocusLearn.

4.3.3 How does FocusLearn perform with different RNN models?

The RNN component of FocusLearn (see component ① in Figure 4.3) is responsible for learning the time-dependencies in the input data. The RNN component (by default an LSTM network) can easily employ alternative architectures like GRU or BLSTM for enhanced flexibility. Therefore, this ablation experiment aims to test how would FocusLearn perform with these alternative architectures instead of an LSTM. Optimal hyper-parameters for all data sets selected during hyper-tuning are used for this experiments, and trained for 100 epochs with the Adam optimiser and early call-backs to avoid over-fitting.

Table 4.4 shows that using an LSTM in the RNN component of FocusLearn achieved the best prediction results for all data sets across all evaluation metrics. The reason of why using an LSTM outperformed using GRU and BLSTM in FocusLearn could be due to the fact that LSTMs are specifically designed for handling long-term dependencies in the input sequences. This means that LSTMs

Table 4.4: **Test set average results comparison of using different RNN models in the RNN component of FocusLearn on four data sets for time series regression and classification.**

Three experiments are compared for each data set – using LSTM in the RNN component, using GRU in the RNN component, and using BLSTM in the RNN component. For regression tasks, lower results are better (↓). For classification tasks, higher results are better (↑).

Regression ↓					
		SMAPE	MASE	WAPE	Training Time
Air	FocusLearn - LSTM	**0.2640**	**0.1637**	**0.1654**	1,198.10
	FocusLearn - BLSTM	0.2661	0.1696	0.1687	2,650.73
	FocusLearn - GRU	0.2831	0.1742	0.1729	2,457.91
Energy	FocusLearn - LSTM	**0.3704**	**0.4557**	**0.5503**	119.64
	FocusLearn - GRU	0.3726	0.4643	0.5810	177.68
	FocusLearn - BLSTM	0.3750	0.4572	0.5646	164.14
Classification ↑					
		Accuracy	F1	AUC	Training Time
EEG	FocusLearn- LSTM	**0.8203**	**0.8131**	**0.8203**	24.28
	FocusLearn - GRU	0.7730	0.7581	0.7730	46.81
	FocusLearn - BLSTM	0.7623	0.7650	0.7623	53.85
Weather	FocusLearn - LSTM	**0.8518**	**0.7857**	**0.8008**	1,192.00
	FocusLearn - GRU	0.7837	0.7656	0.7837	1,355.57
	FocusLearn - BLSTM	0.7665	0.7348	0.7665	1,136.16

are able to capture historical context over many timesteps in the data, whereas GRUs and BLSTMs might only capture recent patterns. Although GRUs tend to be less computationally expensive to train than LSTMs because GRUs combine forget and input gates into a single update gate, the results here have shown that GRUs might not have enough capacity to handle complex dependencies in the data, thereby leading to a poorer predictive performance than LSTMs. BLSTMs tend to perform the worst as shown in Table 4.4, where BLSTMs only marginally outperformed GRU in the *Air* data set by only 6.0% in SMAPE, 2.6% in WAPE, and 2.4% in MASE. One reason for this could be due to the fact that BLSTMs are more suited for tasks where context from both past and future is important (*e.g.,* language translation or speech recognition), as BLSTMs process sequences in both forward and backward directions. In time series analysis, unidirectional context (past-to-future) is often sufficient, and the backward pass in BLSTMs may not provide an advantage.

4.3.4 How does FocusLearn perform with different inference paths?

In addition to the previously discussed AFS and ANB, our study introduces another novel approach – Dual-Paths, which is to have distinct training and inference paths (see section 4.1.3.7). Techniques to boost the learning capability of FocusLearn can be implemented in the training path, whilst the inference path includes by construction only the MNNs component, to ensure that FocusLearn is as explainable as NAM at inference time.

Table 4.5: **Test set average results comparison of the Dual-Paths component of FocusLearn on four data sets for time series regression and classification.**
Two experiments are compared for each data set – using Dual-Paths and without using Dual-Paths in FocusLearn. For regression tasks, lower results are better (↓). For classification tasks, higher results are better (↑).

Regression ↓					
		SMAPE	MASE	WAPE	Training Time
Air	FocusLearn (with Dual-Paths)	**0.2640**	**0.1637**	**0.1654**	1,198.10
	FocusLearn (without Dual-Paths)	0.8840	0.9198	0.9203	2,947.35
Energy	FocusLearn (with Dual-Paths)	**0.3704**	**0.4557**	**0.5503**	119.64
	FocusLearn (without Dual-Paths)	0.9718	1.3900	1.7165	40.05
Classification ↑					
		Accuracy	F1	AUC	Training Time
EEG	FocusLearn (with Dual-Paths)	**0.8203**	**0.8131**	**0.8203**	24.28
	FocusLearn (without Dual-Paths)	0.7984	0.7755	0.7984	86.05
Weather	FocusLearn (with Dual-Paths)	**0.8518**	**0.7857**	**0.8008**	1,192.00
	FocusLearn (without Dual-Paths)	0.777	0.7550	0.7777	1,372.33

In this ablation experiment, we ignore the interpretability requirement, to explore whether the predictive performance of a variant of FocusLearn that uses all components at inference (*e.g.,* selected inputs → RNN → AFS → apply AFS masking → MNNs → summing → prediction), improves compared to the original version of FocusLearn where the selected inputs are only processed by the MNNs at inference. Optimal hyper-parameters for all data sets selected during hyper-tuning are used for this experiments, and trained for 100 epochs with the Adam optimiser and early call-backs to avoid over-fitting.

Table 4.5 shows that by adopting the Dual-Paths method, we not only make FocusLearn more interpretable at inference, but also improves its predictive performance is also improved when compared with using all components at inference. In particular, by using the Dual-Paths method, FocusLearn demonstrates a much superior predictive performance in regression tasks. In the *Air* data set, Fo-

cusLearn with Dual-Paths achieves an increase of 70% in SMAPE, an increase of 82% in WAPE, and an increase of 82% in MASE compared to FocusLearn without Dual-Paths. In the *Energy* data set, FocusLearn with Dual-Paths achieves an increase of 62% in SMAPE, an increase of 67% in WAPE, and an increase of 68% in MASE compared to FocusLearn without Dual-Paths. This improvement not only underscores the potential of separate learning from making inference within the current study but also suggests its potential applicability across a wider range of other domains.

4.3.5 How does FocusLearn perform with different number of features?

The final ablation experiment conducted is to compare the predictive performance of FocusLearn with different number of selected features (*e.g.,* different number of MNNs to construct). For the *Energy* and *Weather* data sets, the evaluation metrics are compared across 4 experiments - $n = 5$, $n = 10$ (reported in the result section 4.2), $n = 15$, and $n = 20$. For the *Air* data set, the evaluation metrics are compared across $n = 5$, $n = 10$, and $n = 15$, since there are only 16 features in the *Air* data set. Lastly, for the *EEG* data set, the evaluation metrics are compared across $n = 5$ and $n = 10$, since there are only 14 features in the *EEG* data set. Optimal hyper-parameters for the data sets selected during hyper-tuning are used for all experiments, and trained for 100 epochs with the Adam optimiser and early call-backs to avoid over-fitting.

Table 4.6 shows that the performance of varying the choice of n remained relatively consistent across different values of n for regression tasks. There is no clear winner out of these experiments. Taking the *Energy* data set for an example, that the difference in the SMAPE value between $n = 15$ and $n = 10$ is 0.02, the difference in the MASE value between $n = 15$ and $n = 20$ is 0.05, the difference in the WAPE value between $n = 5$ and $n = 20$ is 0.01, and the difference in the training time between $n = 10$ and $n = 5$ is approximately 400 seconds.

The minimal impact of different values of n on the overall performance of FocusLearn for regression tasks suggests that the choice of n is a robust method.

Table 4.6: **Test set average results comparison of different number of n to choose in FocusLearn on four data sets for time series regression and classification.**
Several experiments are compared for each data set by varying the number of selected features. For regression tasks, lower results are better (↓). For classification tasks, higher results are better (↑).

Regression ↓					
		SMAPE	MASE	WAPE	Training Time
Air	FocusLearn - n = 5	**0.2521**	0.1660	1.2536	1,680.09
	FocusLearn - n = 10	0.2640	**0.1637**	**0.1654**	1,198.10
	FocusLearn - n = 15	0.2595	0.1669	1.2603	3,402.07
Energy	FocusLearn - n = 5	0.3578	**0.4412**	0.5283	512.71
	FocusLearn - n = 10	0.3704	0.4557	0.5503	**119.64**
	FocusLearn - n = 15	**0.3545**	0.4438	**0.5137**	336.24
	FocusLearn - n = 20	0.3675	0.4590	0.5672	378.56
Classification ↑					
		Accuracy	F1	AUC	Training Time
EEG	FocusLearn - n = 5	0.7819	0.7499	0.6542	27.41
	FocusLearn - n = 10	**0.8203**	**0.8131**	**0.8203**	24.28
Weather	FocusLearn - n = 5	0.7672	0.7336	0.7272	759.20
	FocusLearn - n = 10	**0.8518**	**0.7857**	**0.8008**	1,192.00
	FocusLearn - n = 15	0.7805	0.7639	0.7825	1,219.58
	FocusLearn - n = 20	0.7596	0.7176	0.7596	1,310.30

The *Air* data set has 16 features. The *Energy* data set has 29 feature. The *EEG* data set has 14 features. The *Weather* data set has 21 features.

As a result, whether n is pre-defined or hyper-tuned, FocusLearn is likely to perform effectively, simplifying the implementation of FocusLearn and reducing the need for extensive parameter tuning. This robustness can be considered an advantage, as it implies that FocusLearn is less sensitive to the number of selected features, hence the number of MNNs, and may be more easily generalised across different tasks or data sets without the need for fine-tuning n.

On the other hand, one potential limitation that can be drawn from the finding that the performance of FocusLearn remains relatively consistent regardless of the value of n for the regression tasks is the possibility that FocusLearn may not fully leverage the flexibility of this parameter. If n does not significantly impact the results, it could indicate that FocusLearn is not sensitive to the number of MNNs. It is worth to reiterate here that this experiment does not invalidate the effectiveness of AFS – in this experiment, features are still selected using AFS, we only consider the impact of how many features to choose on the predictive performance of FocusLearn.

It is suspected that the reason to why FocusLearn is less sensitive to different values of n in regression tasks could be due to the data sets investigated in this study are still relatively small compared to some real-world time series data. We

anticipate that FocusLearn will be more sensitive to different values of n with data sets that have a significant number of features to be selected from.

For classification tasks, using n = 10 shows a clear improvement compared to other n values. This might imply that the sensitivity of FocusLearn with respect to the number of MNNs is task dependent. In regression tasks, where the output is continuous, FocusLearn appears to be more robust to variations in different values of n. However, in classification tasks, where the output is discrete and decision boundaries are more defined, changes in n significantly impacts predictive performance, suggesting that it is more influential in tasks requiring clear distinctions between classes.

4.4 Discussion

This research addresses the gap in the current landscape of methods tailored for explaining time series data, where some methods can only handle either regression or classification tasks, while others are confined to post-hoc interpretability (trying to understand the model after it made decisions). The relative inflexibility and often sub-optimal predictive performance of interpretable models are some of the key factors contributing to the greater popularity of post-hoc methods in practical applications. It is shown in this study that the predictive performance of interpretable models, such as Neural Additive Model (NAM) [4] and Scalable Polynomial Additive Model (SPAM) [42], and interpretable eXtreme Gradient Boost (XGBoost) are not comparable with Deep Nerual Networks (DNNs), such as Long Short-Term Memory (LSTM) [58], Attention-based LSTM (AttnLSTM), and Attention-based Encoder-Decoder (AttnED). The popularity of post-hoc methods arises from the fact they can be applied across a variety of predictive models, because obtaining a good predictive performance is still one of the most important objectives for machine learning models.

However, the inherent approximation limitation of post-hoc methods, where explanations can vary significantly between different approaches and can often produce misleading explanations [97] highlights critical challenge. Therefore, the

main aim of this study was to develop an interpretable model that is capable of delivering both high predictive performance and explanations for both regression and classification tasks.

FocusLearn is a novel interpretable modular neural network with attention mechanism as feature selection for time series analysis. The goal of FocusLearn is to improve the predictive performance compared to NAM, and, if possible, to match the predictive performance of the non-interpretable DNN models. Furthermore, the architecture proposed in this work is explicitly designed to handle both classification and regression tasks within a unified framework. This is not merely a switch in loss function, but a demonstration of architectural flexibility and robustness across different task formulations. This dual capability is particularly valuable in real-world applications where both types of predictive tasks frequently co-exist, such as healthcare or sensor-based systems.

Our evaluation in Table 4.1 shows that FocusLearn not only outperforms NAM on all metrics by a large margin, but does the same with SPAM. In fact, its predictive performance matches that of non-interpretable XGBoost, LSTM, AttnLSTM, and AttnED, and can even outperform them in some cases. Our results also show a consistent predictive performance of FocusLearn across both classification and regression tasks, and that it generalises well on unseen data. Furthermore, FocusLearn exhibits significantly faster computation time compared to NAM on all data sets and SPAM on the *Air*, *EEG*, and *Weather* data sets, with FocusLearn being the fastest neural network in the *Air* and *EEG* data sets.

FocusLearn consists of two main paths (training and inference) with five main components: a *Recurrent Neural Network (RNN)* (see section 4.1.3.1), an *Attention-based Feature Selection (AFS)* (see section 4.1.3.2), and a group of *interpretable Modular Neural Networks (MNNs)* that learns the top n input features as selected by the AFS (see section 4.1.3.3). The *Attention-based Node Bootstrapping (ANB)* in each *module*'s first layer is weighted by the *AFS*'s attention weights (see section 4.1.3.3.1). The outputs of MNNs are then summed together and serve as predictions. All components are required for training, and to maintain the interpretability of FocusLearn, only components in the *inference path* box are used during the inference step (see section 4.1.3.7)

Several novel ideas are introduced in FocusLearn, including AFS, ANB, and Dual-Paths. In AFS, Multi-head Attention is modified to select most relevant features from the original input, while ANB acts as a bridge between AFS and each module by weighing each selected feature with its corresponding attention weights. With Dual-Paths, techniques to boost the learning capability of FocusLearn can be implemented in the training path, whilst the inference path includes by construction only the MNNs component, to ensure that FocusLearn is as explainable as NAM at inference time. It is worth to note here that creating a separate inference path is not the same as creating a surrogate model for the RNN-AFS components (or the training path) of FocusLearn. Surrogate models, in terms of XAI techniques, are created after the original prediction model already made a decision. Taking SHAP for examples, predictions need to be produced first with a prediction model, then SHAP can be used to explain why the prediction model made certain decisions. In FocusLearn, the training path is only responsible for learning which features to select, and the inference path is responsible for making the decision.

Before designing the architecture of FocusLearn, several experiments were conducted to identify the key factors of an effective prediction model for multivariate time series data. These experiments helped inform the design of FocusLearn by providing insights into the essential components for achieving accurate predictions, which is crucial for ensuring that FocusLearn performs well in its predictive tasks. From the initial phase of this research, it became evident that a stand-alone LSTM is a promising candidate to be investigated further as it showcased superior predictive performance in univariate settings. Therefore, to understand what techniques can be used to improve upon the predictive performance of a stand-alone LSTM with multivariate time series data, we experimented with two DNN architectures – AttnLSTM and AttnED. Previous research has shown that incorporating attention mechanism into LSTMs can improve their prediction performance because attention mechanism can selectively focus on parts of the inputs, and this selective focus allows LSTM to prioritise relevant information in the inputs [41, 44, 86, 160, 168]. Experiment results in Table 4.1 have shown that AttnLSTM and AttnED can indeed outperform a stand-alone LSTM on both regression and classification tasks. Experiments results also show that AttnED can

outperform AttnLSTM in some of the tasks. This could be attributed to the fact that the attention mechanism is applied to the hidden states rather than directly to the inputs, since hidden states can encapsulate a richer representation of the input data [134] and the attention mechanism can focus on more abstract and meaningful features.

If the attention mechanism enables LSTMs to selectively focus on specific parts of the input sequence, it can also be utilised as a feature selection tool. Learning only the most salient features has been shown to be a useful method of improving interpretability and allocating learning capacity more effectively [21, 53]. Experiment results in section 4.3.1 show that utilising AFS as a feature selection tool in FocusLearn also enhances its predictive performance – using AFS outperforms not using AFS on all data sets across all evaluation metrics. To utilise the attention weights further, each module in the MNNs component of FocusLearn is initially weighted using their corresponding attention weights using ANB. Experiment results in section 4.3.2 show that ANB is also effective at improving the prediction results of FocusLearn, with using ANB consistently outperforming not using ANB or using ExU (as proposed in NAM) across all evaluation metrics for both regression and classification tasks.

The RNN component of FocusLearn, by default an LSTM network, can easily employ alternative architectures like Gated Recurrent Unit (GRU) or Bidirectional LSTM (BLSTM) for enhanced flexibility. Experiment results presented in section 4.3.3 show that using an LSTM in the RNN component of FocusLearn proves to be an advantageous decision, as using LSTM achieved the best prediction results for all data sets across all evaluation metrics compared with using GRU and BLSTM in the RNN component.

To ensure the explainability of FocusLearn, the inference path includes by construction only the MNN component. This means that at inference time the RNN-AFS components exclusively handle feature selection during training. Once the top n features are selected, FocusLearn becomes as explainable as NAM, as it no longer relies on the components in the training path for inference. Experiment results in section 4.3.4 show that by adopting this Dual-Paths method, FocusLearn not only becomes as explainable as NAM but the prediction results

of FocusLearn are also improved when compared with using all components at inference.

One observation made from evaluation results in section 4.3.5 is that the performance of FocusLearn by varying the numbers of selected features, n, remained relatively consistent for regression tasks, but $n = 10$ (reported in Table 4.1) showed a significant improvement over other choices of n for classification tasks. This finding might imply that the sensitivity of FocusLearn with respect to the number of different n (consequently how many MNNs to construct) is task dependent. In regression tasks, where outputs are continuous, FocusLearn appears to be more robust with the choice of n. This however might also indicate that FocusLearn may not fully leverage the flexibility of this parameter. On the other hand, in classification tasks, where the output is discrete and decision boundaries are more defined, changes in n significantly impact predictive performance, suggesting that it is more influential in tasks requiring clear distinctions between classes.

Chapter 5

Application of FocusLearn in Hearing Loss Case

The research conducted in this study was initially motivated by the SMART BEAR EU research project [1]. Particularly with the need for transparent decision-making processes in AI models for sensitive domains, such as health care. SMART BEAR aims to develop a smart big data platform that will provide evidence-based personalised support for several pressing healthcare issues faced by the ageing EU societies, including hearing loss, cardiovascular diseases, and mental health. SMART BEAR aims to analyse data continuously collected from integrated heterogeneous sensors, and assistive medical and mobile devices, in order to provide evidence needed for personalised support and interventions, towards promoting healthy and independent living. SMART BEAR is currently tested in large-scale pilots involving 5,000 elderly participants from five countries in the EU: France, Greece, Italy, Romania, and Portugal.

[1] Research Number: 857172. Website: `https://www.smart-bear.eu/`. CORDIS Project Summary Website: `https://cordis.europa.eu/project/id/857172/results`

Our participation in the SMART BEAR EU project focused on hearing loss (HLoss), where the question to be answered is to **predict future hearing aids (HAids) usage and find out what features influence the patients to use their HAids more.** Clinicians can then adjust the interventions based on the explanations.

To date, there are still not enough data from SMART BEAR to conduct analysis that will provide meaningful results. Finding alternative and similar data that are also HLoss-related was challenging. *OtiReal* [33] is a HLoss data set that closely resembles SMART BEAR data, but does not contain important variables such as overall HAid satisfaction and manual adjustment of HAid volume/program. This data set was collected during the EVOTION EU research project [2], a precursor to the SMART BEAR project, which focused only on HLoss.

Although the original aim was to utilise all available features collected through the SMART BEAR, data limitations of the *OtiReal* data set restricted the analysis to acoustic features only, which were therefore used as the basis for predicting future HAids usage over a week and explaining which acoustic features would influence the patients to use their HAids more.

5.1 Hearing Loss Data

OtiReal is a **sampled real** data set collected from 98 participants wearing Oticon A/S Internet-connected HAids between June and December 2019 [33]. All participants were allocated with a pair of Oticon Opn™ HAids that were connected to the Oticon ON™ smartphone app for remote control with HearingFitness™ [81] feature enabled. The commercially available HearingFitness™ program activates automatic logging of sound data every 60s from the HAids microphones together with timestamps indicating when the HAids were turned on and connected to Bluetooth [33]. Given the real-world nature of the OtiReal data, no personal information (*e.g.,* age and gender) were provided due to privacy issues. Table 5.1 summarises the variables with their description and type. Sound pres-

[2]Research number: 727521. Website: `https://h2020evotion.eu/`

sure level (SPL), sound modulation level (SML), and signal-noise-ratio (SNR) are continuous acoustic variables, and sound class (*sClass*) classifies the momentary sound environment into four categories: quiet, noise, speech, and speech in noise. Noise floor (Nf) is a continuous variable that measures the output of the HAid in operation with no input signal.

Table 5.1: **List of variable types and descriptions for the OtiReal data set**.

Variable Name	Type	Description
ID	Integer	Identifier
hSide	Categorical	HAid side: left or right of worn hearing aid
SPL	Continuous	Sound pressure level (0-10KHz)
SML	Continuous	Sound modulation level (0-10KHz)
SNR	Continuous	Signal-to-noise ratio (0-10KHz)
Nf	Continuous	Noise floor (0-10KHz)
sClass	Categorical	Sound environment: quiet, noise, speech, and speech in noise
hUptime	Continuous	Hearing aid up time (seconds)
Timestamp	ISO 8601	Time of the logged record

SPL, SML, and SNR are derived acoustic features computed from momentary sound waves recorded by calibrated HAid microphone at ear level. These features are all estimated in a broadband frequency range between 0-10kHz and expressed in decibel units. SPL is the level output estimate from a low-pass infinite impulse-response filter with a time constant of 63ms, where the filter smooths rapid fluctuations in the signal to provide a measure of overall sound level. SML is then derived as the difference between the top and bottom trackers of the SPL. The top tracker of the SPL, or the peak detector, is implemented with a fast dynamic attack time of 1-5s and a slow release of time of 30ms. Whereas the bottom tracker of the SPL, or the valley detector, is the reverse of this. SNR is the difference between the bottom tracker and the immediate SPL. The sound class variable classifies the momentary sound environment into four categories by a proprietary HAid algorithm using SPL, SML, and SNR values.

5.1.1 Pre-processing the Hearing Loss data

5.1.1.1 Calculate Hearing Aid Usage

OtiReal does not contain actual daily hearing aid (HA) usage logs, so usage is inferred from the Timestamp variable. To estimate the total HAid usage in sec-

onds per day per patient, usage intervals are computed based on consecutive timestamps. The term Distance, D, is defined as the difference in seconds between two consecutive timestamps, t_i and t_{i+1}. Following an assumption discussed during the viva, if the time difference between consecutive measurements exceeds 10 minutes (600 seconds), it is presumed that the hearing aid was not worn during that interval, and such gaps are excluded from the total usage calculation.

OtiReal does not contain actual daily HAid usage, so this is calculated from the Timestamp variable. To calculate the total HAid usage in seconds per day per patient, the usage interval is calculated first from the Timestamp variable. The term Distance, D, is defined first as the difference in seconds between two consecutive timestamps, t_i and t_{i+1}, of when the measurements are taken as in Equation (5.1). If the time difference between t_i and t_{i+1} exceeds 10 minutes (600 seconds), it is presumed that the HAid was not worn during that interval, and such gaps are excluded from the total usage calculation. such that:

$$D = t_{i+1} - t_i, \tag{5.1}$$

The maximum distance, D_{max}, between two consecutive timestamps was set to 600 seconds in this analysis by the domain experts of EVOTION. In other words, two consecutive measurements m_i and m_{i+1} taken at t_i and t_{i+1} respectively, are considered to belong to the same interval, u_t, if and only if the distance, D, between the two timestamps is less or equal D_{max}, *i.e.,* $|D| \leq D_{max}$. Consequently, m_{i+1} is considered to belong to the subsequent interval, u_{t+1}, if the distance is greater than 600 seconds. The Usage Interval Duration, d_t, of each u_t is then calculated as:

$$d_t = t_e - t_s, \tag{5.2}$$

where t_e is the last (maximum) timestamp of u_t and t_s is the first (minimum) timestamp of u_t. Finally, the HAid Usage, h_t, per day is calculated by taking $d_1 + d_2 + \cdots + d_n$ for all intervals in a day. For other variables to be transformed

with a daily frequency, the average of each variable is taken for each interval for each participant.

5.1.1.2 Other Pre-processing Steps

After calculating the HAid usage for the *OtiReal* data set, it is then pre-processed in the same manner as the four time series data sets, as discussed in section 3.1.3, following an identical sequence of steps to ensure consistency across all data sets. These steps include handling missing data, encoding categorical features, standardisation and normalisation, and removing outliers. There are 23.69% of missing values in the *OtiReal* data set, referring to absent feature entries (*e.g.,* SPL, SML, SNR) rather than missing time steps, as the data are collected at irregular intervals.

The missing values in the *OtiReal* are more than twice as those in the *Weather* data set (9.83%) and more than 24-fold increase compared to the *Air* data set (0.98%).

Features *ID* and *Timestamp* are excluded from modelling in the *OtiReal* data set because these two features are not informative for prediction. The *ID* feature serves only as a unique, pseudonymised identifier for each participant and does not carry any meaningful pattern related to the outcome. Similarly, while the *Timestamp* reflects the temporal order of the recordings, the relevant temporal information is already captured in the time-dependent features. Including these variables would risk introducing noise or allowing the model to overfit to non-generalisable ordering artifacts. This results in the *OtiReal* data set only containing seven features. Therefore, a sliding window with lagged features approach is also applied during pre-processing to enhance the temporal context and improve model performance. Specifically, data from the current timestep, t_i, and the next timestep, t_{i+1}, are used to predict the HAid usage at t_{i+2}. Incorporating lagged features allows the model to leverage both immediate past and current information to better capture short-term dependencies between timesteps. It also enhances the ability of the model to predict future HAid usage more accurately, as the short-term temporal dynamics are explicitly included in the predictive process. In total, there are 16 features including calculated HAid usage.

Table 5.2: **Comparing predictive performance on the healthcare domain uses cases**. Means and standard deviations are reported from 5 independent training and evaluation runs, each with different random initialisations using the same with optimal hyper-parameters. Appendix C.2 summarises the hyper-parameters and their optimal corresponding settings for each task. For regression tasks, lower results are better (↓). Regression results are ranked by the SMAPE. Training time percentage is calculated with respect to the fastest neural network model.

		SMAPE	WAPE	MASE	Training Time	Training Time %
OtiReal	LSTM	**0.1809***	**0.0666***	1.2694^{+}	92.77	390.45
	FocusLearn	0.2105*	0.0806**	1.4829	96.30	405.30
	SPAM	0.2951^{+}	0.1249*	2.3075^{+}	119.73	503.91
	XGBoost-Impute(100)	0.3265**	0.1468**	2.701**	12.96	50.55
	AttnED	0.5278*	0.3336*	1.3587*	49.56	208.59
	AttnLSTM	0.5339*	0.3259*	1.3273*	23.76	100.00
	NAM	0.8543*	0.6661	12.3317	238.95	1,005.68
	XGBoost-No Impute(100)	0.9574**	0.6846**	**1.0840****	1.29	5.43

Interpretable model. Non-interpretable model.
**: Variance across experiments is < 0.01.
*: Variance across experiments is < 0.05.
$^{+}$: Variance across experiments is > 0.05.

Note: The number in parentheses after XGBoost's name is the optimal number of trees for it; XGBoost is deemed as non-interpretable if the number of trees exceeds 50. Data are pre-processed differently for XGBoost. More specifically, XGBoost supports missing values by default, whereas missing values are imputed for training with other models. Therefore, to ensure a fair comparison with other models, missing values are imputed for training with XGBoost-Impute.
Since XGBoost requires significantly less training time than neural network models, it is consistently the fastest; therefore, training time percentage are also reported which is relative to the fastest neural network model.

5.2 Prediction and Explanation Results On Hearing Loss Uses Case

5.2.1 Prediction Results

As Table 5.2 shows, FocusLearn outperforms all interpretable models, including NAM and SPAM, on all evaluation metrics. Most notably, FocusLearn substantially outperforms NAM by 75% in SMAPE, 88% in WAPE, and 88% in MASE.

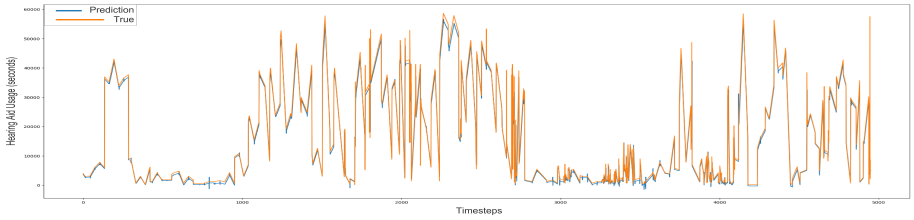
Table 5.2 also shows FocusLearn can match the predictive performance of non-interpretable models – LSTM, AttnLSTM, AttnED, and non-interpretable XGBoost. FocusLearn and LSTM achieved the lowest SMAPE (FocusLearn: 0.2105 and LSTM: 0.1809) and WAPE (FocusLearn: 0.0806 and LSTM: 0.0666) but did not achieve a good MASE result (FocusLearn: 1.4829 and LSTM: 1.2694). While XGBoost-No Impute achieved the lowest MASE (1.0840) error, its SMAPE (0.9574) and WAPE (0.6846) errors are the worst when comparing with other models, and indicate that imputation may be beneficial even for XGBoost in

some cases. More specifically, FocusLearn outperformed XGBoost-No Impute by 78% in SMAPE and 84% in WAPE. Although XGBoost-Impute performed better than XGBoost-No Impute as shown in Table 5.2, FocusLearn also outperformed XGBoost-Impute across all evaluation metrics by 36% in SMAPE, 45% in WAPE, and 45% in MASE.

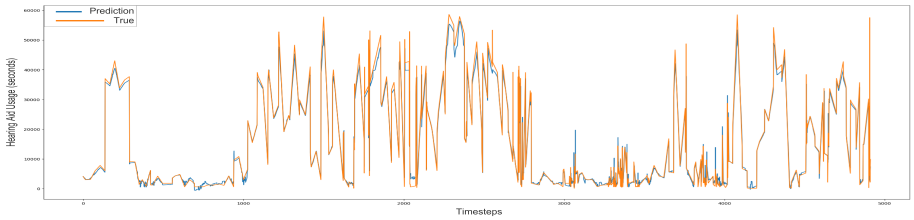
Both SMAPE and WAPE are percentage-based error estimators and focus on the accuracy of percentage errors, whereas MASE assess the ability of a model relative to a simple benchmark model. Therefore, given the discrepancy in results depending on the choice of metric, a visual inspection of the prediction results would be helpful to decide which model performed better in the *OtiReal* data set. While metrics are essential for quantitative evaluation, the ultimate goal is to achieve predictions that align with the underlying pattern and behaviour in the data.

The evaluation of the models on the *OtiReal* data set was conducted using SMAPE, WAPE, and MASE – the same metrics applied to the benchmark data sets (*e.g., Air*, *Energy*, *Weather*, and *EEG*). While these metrics are appropriate for regression tasks and offer robust summaries of predictive accuracy, their interpretation requires additional nuance in a healthcare context. For instance, SMAPE and WAPE provide relative error measures, which are useful for understanding overall model deviation, but do not distinguish between under- and over-prediction, which may have different consequences in practice. In a clinical decision support scenario, underestimating non-adherence to HAid usage may be more problematic than overestimating it. MASE provides scale-independent interpretability, but again does not capture the temporal or behavioural significance of errors in patient-related predictions. Thus, while these metrics offer a consistent basis for technical comparison, their implications in the healthcare setting should be contextualised. Future iterations of the model could benefit from incorporating task-specific evaluation criteria or outcome-sensitive cost functions to better align quantitative performance with clinical relevance.

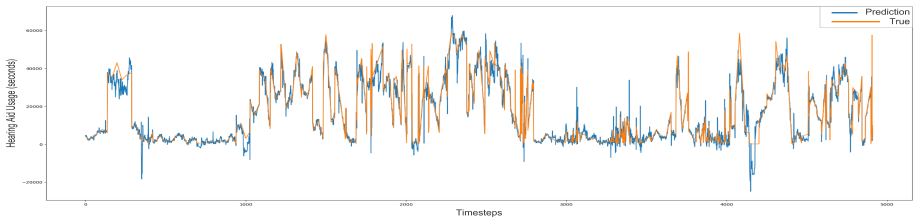
Visualisations in Figure 5.1 show that FocusLearn and LSTM are better than both XGBoost-Impute and XGBoost-No Impute at capturing the pattern and trend in the *OtiReal* data set as the predicted values are much better aligned



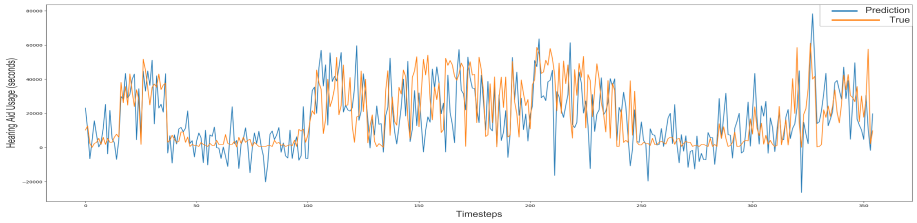
(a) LSTM



(b) FocusLearn



(c) XGBoost-Impute



(d) XGBoost-No Impute

Figure 5.1: **Prediction Visualisations.**
X-axis represents predicted timesteps and y-axis represents feature values. *Note: data are pre-processed differently for XGBoost. More specifically, XGBoost supports missing values by default, whereas missing values are imputed for training with FocusLearn and LSTM. Therefore, to ensure a fair comparison, missing values are imputed for training with XGBoost-Impute.*

to the true values. The figures also show that the magnitude of the true/target values vary significantly in the *OtiReal* data set and both SMAPE and WAPE are better at comparing models in this situation than MASE. More specifically, while MASE provides a robust, scale-independent measure of overall model performance, it has limitations when applied to data sets with extended flat regions. In such cases, models that generate volatile predictions around flat true values may still achieve a low MASE, since the absolute deviations are small and average out over time. This results in a discrepancy when quantitative evaluation suggests strong predictive performance, but visual inspection reveals instability and poor alignment with the true time series. Such behaviour was observed for XGBoost-No Impute on the *OtiReal* data set, where it achieved the best MASE score, but visual inspection in Figure 5.1 of the prediction series suggests that it did not align as well as FocusLearn and LSTM. This observation highlights the importance of combining quantitative metrics with qualitative visualisation when evaluating time series forecasts, as no single metric can fully capture the perceptual or application-specific quality of predictions.

FocusLearn also has a notably faster computational time than the interpretable models, being 20% faster than SPAMand 60% faster than NAM. It is also very close to the training time of LSTM.

5.2.2 Explanation Results

Figure 5.2 shows the 10 shape functions learned by FocusLearn for the *OtiReal* data set to predict future HAid usage. These shape plots are discussed from left to right and top to bottom as follows.

The shape plots for sound modulation level at timestep t_i (*SML-i*), Noise Floor at both timesteps (*Nf-ti* and *Nf-t(i+1)*), and Signal-to-noise ratio at both timesteps (*SNR-ti* and *SNR-t(i+1)*) follow a similar pattern in contributing to the model prediction. Patients with median *SML*, *Nf*, and *SNR* values sensed by the HAids tend to use their HAids more in the future. In terms of *SML*, patients will be more likely to use their HAids more in the future if *SML* values sensed by the HAids are around 15 decibels, and are less likely to use their HAids more

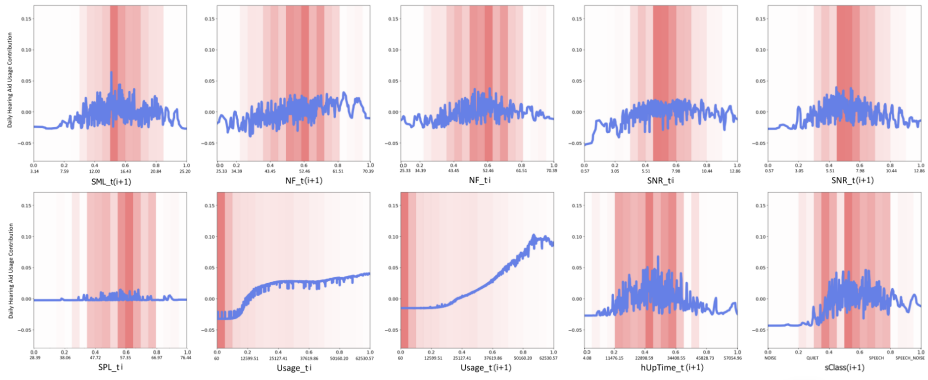


Figure 5.2: **Graphs learned by FocusLearn for OtiReal**
Predicting future HAid usage (regression) on the *OtiReal* data set. These plots show top 10 features selected by the AFS component in the FocusLearn, where selected features with normalised and original values are on the x-axis, and daily future HAid usage prediction contribution are on the y-axis. In OtiReal, two timesteps of data are transformed and used as inputs, such that we are using data at t_i and t_{i+1} to predict HAid usage at t_{i+1}.

in the future if *SML* values sensed by the HAids are either less than 10 or more than 23 decibels. For *Nf*, if the HAids sense the *Nf* value at around 50 decibels, then these patients will be more likely to use their HAids more in the future. For *SNR*, patients will be more likely to use their HAids more in the future if *SNR* values sensed by the HAids are around 6 decibels. HAid Up Time ($hUpTime$) also follows a similar pattern with these variables, where if the HAids have been activated for around 23,000 seconds (or 6.3 hours), then these patients will be most likely to use their HAids more in the future.

Comparing with other variables, Sound Pressure Level at timestep t_{i+1}(*SPL-t(i+1)*) does not contribute much to predicting future HAids usage. This would warrant a further investigation with audiologists as *SML* and *SNR* are derived from *SPL*. *SML* is the difference between peak and valley detector of the *SPL* and *SNR* is the difference between the valley detector and the immediate *SPL*.

As expected, the shape plots for *Usage* show that patients with higher daily HAids usage are more likely to have higher future daily HAids usage and patients with lower daily HAids usage are more likely to have lower future daily HAids usage. The difference in the shape function (blue line) in *Usage-ti* and *Usage-t(i+1)* shows that the maximum contribution to future HAids usage for *Usage-ti* is lower than the maximum contribution for *Usage-t(i+1)*. This means that patients are more likely to have an even higher future daily HAids usage if they use their

HAids for a sufficiently long period in a day continuously (for at least 2 days in this case).

The most rapid increase in future HAids usage contribution at *Usage-ti* occurs when HAids usage is above 10,000 seconds (or 3 hours), this means this is the minimum HAids usage that a patient needs to have to be able to have a higher future HAids usage. Whereas the most rapid increase in future HAids usage contribution at *Usage-t(i+1)* occurs when HAids usage is above 20,000 seconds (or 6 hours).

The sound class (*sClass*) classifies momentary sound environment into four categories by a proprietary HAids algorithm using *SML*, *SNR*, and *SPL* values. Its plot shows that if the sound class is at noise or quiet setting, then these patients are less likely to use their HAids more in the future, whereas if the sound class is at speech setting, then these patients are more likely to use their HAids more in the future. This could be an interesting observation to the audiologists as this might show that those patients who are active and use their HAids to assist their HLoss will tend to continuously use their HAids in the future, and on the other hand those patients who might not be familiar with their HAids or not actively use their HAids are less likely to continuously use their HAids in the future. It may also indicate that socialisation could help increase HAids usage.

Chapter 6

Conclusion

There is a gap in the current literature where the majority of eXplainable AI (XAI) methods tailored for explaining time series data are confined to handling either regression or classification tasks, while other explanation methods tend to steer towards to post-hoc interpretability (trying to understand the model after decisions have been made), which are often criticised for low fidelity [3], instability [50], and even inaccuracy [88, 120], since they only approximate their underlying model. Furthermore, current interpretable models tend to have a substantially inferior predictive performance when compared to Deep Neural Networks (DNNs). Therefore, models such as the Neural Additive Models (NAM) have not been extensively applied to the time series domain, despite the ability of NAM to produce explanation graphs that precisely describe the reasoning behind the decisions.

This research bridged the gaps in the current literature by introducing an interpretable DNN designed for time series analysis, named FocusLearn. The goal of FocusLearn was to improve predictive performance compared to NAM and yet retain its interpretability. Experiments on two regression and two classification data sets (see Table 4.1), as well as one healthcare use case (see Table 5.2), have shown that FocusLearn not only outperforms NAM (and its variant SPAM)

on all metrics, but it can match the predictive performance of non-interpretable models Long Short-Term Memory (LSTM), attention-based LSTM (AttnLSTM), and attention-based encoder-decoder (AttnED), and even outperforming them in some cases. Even more importantly, FocusLearn was consistently among the top performers, while other DNNs performed quite poorly in some data sets. This was achieved without requiring substantially more computational resources. Indeed, on some data sets, FocusLearn was the fastest DNN.

6.1 Summary of Key Findings

6.1.1 AttnLSTM and AttnED can perform better than a stand-alone LSTM

The development of FocusLearn began with a recognition that Recurrent Neural Networks (RNNs), specifically LSTM [58], are suitable tools for time series analysis due to their ability to capture long-term dependencies and temporal dynamic [61, 63, 121, 136]. However, a stand-alone LSTM would need additional assistance, such as attention mechanism, to be able to manage the complexities presented by multivariate time series due to the intricate dependencies and interactions among multiple variables over time. To this end, we experimented with two model architectures to improve upon the predictive performance of a stand-along LSTM. These are AttnLSTM and AttnED. Experimental results demonstrated that AttnLSTM and AttnED can indeed improve LSTM's ability in handling complex data in some cases, showing that attention mechanism can be a useful tool in DNNs. Consequently, one of the sub-objectives of this study – experiment with time series DNN models that help to improve predictions compared to a stand-alone LSTM – has been successfully achieved.

6.1.2 FocusLearn can perform better than interpretable models and match the performance of non-interpretable models

FocusLearn consists of five main components: an RNN to learn the temporal dependencies in the data (see section 4.1.3.1), an Attention-based Feature Selection (AFS) to select features while also suppressing redundant features (see section 4.1.3.2), and a group of interpretable Modular Neural Networks (MNNs) that learns the top n input features selected by the AFS (see section 4.1.3.3). The Attention-based Node Bootstrapping (ANB) in each module's first layer is additionally weighted by the AFS 's attention weights (see section 4.1.3.3.1). The outputs of MNNs are then summed together and serve as predictions, as in additive models such as NAM.

Another novel idea introduced in FocusLearn is Dual-Paths, which is to have two distinct training and inference-paths. The inference path of FocusLearn includes by construction only the MNNs component. Results from the abalation experiment (see section 4.3.4) have shown that having distinct training and inference paths not only made FocusLearn interpretable, but the predictive performance also improved.

Experiment results therefore have shown that FocusLearn has met one of the main objectives set out in this research, which is effectively predicting time series data for both regression and classification tasks. More specifically, FocusLearn also meets the sub-objectives. In terms of prediction, FocusLearn outperforms state-of-the-art interpretable model, such as NAM, SPAM, and interpretable XGBoost, as well as matches the performance of time series DNN model, such as LSTM, AttnLSTM, AttnED, and non-interpretable XGBoost, even outperforming them in some cases. More importantly, FocusLearn demonstrates consistently strong performance and maintains reliable and efficient training time, in contrast to other models.

In terms of explanations, FocusLearn also has met the other main objective which is to provide explanations. In particular, FocusLearn meets the last sub-

objective such that FocusLearn does not require post-hoc methods to provide explanations, as was demonstrated with the HLoss data set.

6.2 Contributions to the Field

The primary contribution of this research lies in the development of FocusLearn, which draws inspiration from experiments conducted in this study. In particular, that the predictive performance of a stand-alone LSTM can be improved by incorporating a self-attention mechanism. This is because self-attention allows LSTM to selectively focus on the most relevant part of the input sequence. Furthermore, the prediction results can also be improved by using an LSTM to learn the input data representations and the time dependencies in the data to allow the attention mechanism to focus on more abstract and meaningful features, rather than the potentially noisy and less informative raw inputs.

FocusLearn also draws inspiration from the principles of modularity [19, 143], additive models [4, 42], and the idea that interpretability can be enhanced through linearity [17]. By structuring the model as a network of individual modular units, which can be summed together in line with additive models, this approach creates a flexible yet interpretable framework. Modular networks, by their very nature, offer greater interpretability compared to traditional DNNs, addressing a key challenge in the field of XAI. However, interpretability so far has come at the cost of performance. To overcome this, we introduced several novel ideas that were inspired by the experiments discussed in this work, namely AFS, ANB, and Dual-Paths, allowing FocusLearn to maintain high performance while still offering transparent explanations of its predictions. This work significantly contributes to the XAI domain by presenting a novel model that not only performs well but also provides interpretable insights for both regression and classification tasks, filling a critical gap in the current literature.

The importance of this work is underscored by the growing need for AI models that are both powerful and transparent. In many applications, particularly in high-stakes environments like healthcare and finance, the ability to explain how

and why a model makes certain decisions is as crucial as the accuracy of the predictions themselves. This research bridges the gap between performance and interpretability, offering a new approach that could influence future developments in the field, making AI systems more trustworthy and widely applicable.

Furthermore, the potential applications of Dual-Paths extend far beyond the initial scope we can currently foresee. While initially designed for FocusLearn, Dual-Paths is not inherently tied to this specific model, suggesting its broader utility across a range of different models and contexts. This versatility opens up opportunities for its deployment in various domains where there is a need for models that are not only accurate but also interpretable. By leveraging Dual-Paths, practitioners in diverse fields can achieve potentially high levels of predictive performance while maintaining the transparency and interpretability of their models, making it a valuable tool across multiple domains.

6.3 Limitations of the Study

A significant limitation of this study stems from computational constraints, which had a substantial impact on the scope and depth of the research. This limitation not only restricted the diversity of models evaluated but also may have biased the findings toward models that are computationally less intensive. Future work would benefit from access to more robust computational resources, allowing for a more comprehensive evaluation of a broader range of models and data sets, thereby providing a more thorough understanding of their relative strengths and limitations.

Furthermore, some state-of-the-art methods, such as transformers and Temporal Fusion Transformer (TFT) [87], were not included in the experiments or used as baseline models due to their substantial computational resource requirements. These models are known for their impressive performance, but they typically demand high computational power and extensive memory, which exceeded the capabilities available for this study. Additionally, transformers and TFT tend to excel only when trained on large data sets [165], which are often difficult to

obtain or access, especially in specialised domains, such as the HLoss use case investigated in this study. Consequently, while these methods are highly regarded in the field, their exclusion from this study was necessary due to practical constraints, potentially limiting the exploration of certain advanced techniques and their comparative evaluation against the models experimented with in this study.

That being said, the need for high-performance computing environments to run these models effectively suggests that their deployment may be limited to well-funded research institutions or organisations with significant computational infrastructure. This limitation challenges the accessibility and scalability of these methods, particularly in resource-constrained settings where such infrastructure is not available. Therefore, while these models may demonstrate strong performance under ideal conditions, their reliance on extensive computational resources could hinder their practical adoption and limit their usability in broader, more diverse contexts. On the other hand, the computation time of FocusLearn is much faster than interpretable models (NAM and SPAM), and sometimes even faster than non-interpretable models (LSTM, AttnLSTM, AttnED, and non-interpretable XGBoost) as shown in Table 4.1 and Table 5.2.

As a limitation of this work, we did not consider multi-class time series classification. This is a potential area of future research and it could be possible to extend FocusLearn and other developed models to multi-class classification with the right implementation. Another limitation of FocusLearn is that it is not able to provide a higher-order feature interaction like in SPAM. However, this was not of a focus for this work where the objective of this research is to develop an interpretable model for time series that can outperform NAM in terms of predictive performance and provides similar explanations.

Furthermore, experiment results from section 4.3.5 shows that a potential limitation of FocusLearn is that there is a possibility that FocusLearn may not fully leverage the flexibility of n (number of features to select) because the performance of FocusLearn remains relatively consistent regardless of the value of n for regression tasks. If n does not significantly impact the results, it could indicate that FocusLearn is not sensitive to the number of MNNs. It is hypothesised that FocusLearn's reduced sensitivity to variations in n may be attributed

to the relatively small size of the data sets examined in this study compared to larger real-world time series data, such as that collected by SMART BEAR. SMART BEAR aimed to gather a total of 265 variables encompassing patient demographics, biological, environmental, and behavioural characteristics [64]. We expect that FocusLearn will exhibit greater sensitivity to different values of n when applied to data sets with a substantial number of features to select from.

In addition to computational limitations, this study is constrained by the relatively small size and limited diversity of the evaluated data sets. These constraints may limit the generalisability of our findings to broader real-world time series scenarios, particularly those with high feature dimensionality, irregular sampling, or domain-specific complexity.

The final limitation of this work is the absence of comprehensive evaluations for the XAI methods used, specifically with NAM, SPAM, and FocusLearn. Although there are currently no universally accepted objective metrics for evaluating XAI methods, it is imperative to explore and apply existing metrics, such as Rosenfeld's set [118], to these methods. Implementing such objective metrics in future studies could provide a more robust and comprehensive evaluation, combining both objective and subjective validation, which is essential for establishing the trustworthiness and generalisability of FocusLearn across different applications and domains.

6.4 Future Research and Directions

Related work on AFS, MFS and TabNet shares some similarities with the proposed FocusLearn. However, FocusLearn is distinct from them because attention weights are learned from a recurrent network in FocusLearn and used for weighted feature selection. FocusLearn also has flexibility in using any RNNs. FocusLearn was designed to be closely related to architectures with modularity, such as additive models. A common assumption of such models such as GAM and NAM is feature independence. This has been proved to be too strong an assumption in certain areas of application such as time series, as indicated by the

poor predictive performance of such models in the experiments reported in this paper. FocusLearn tackles this problem by using AFS for feature selection during training. Variations of NAM on the other hand, such as SPAM, seek to tackle the same problem by allowing interaction terms. This, however, may increase the complexity of the explanations provided by these models. As future work, we shall evaluate this and other trade-offs, such as the number of modules to use, in practice, since it is likely that the answers to these questions will depend on the characteristics of the specific application domain and quantitative expert evaluation.

6.4.1 Feature interaction

In comparison with SPAM, a given application may require the use of higher-order feature interaction, not available in FocusLearn. Although providing higher-order feature interaction as in SPAM is not in the scope of FocusLearn, we shall investigate the possibility of the attention mechanism creating such features for implementation into a single network module. Finally, we shall continue to investigate the practical value of the explanations provided, how they may inform model intervention, and the interactions that may exist between attention-based explanations of RNNs and interpretable modular neural networks.

6.4.2 Number of modules in FocusLearn

With regards to the number of modules to use, we shall continue to investigate into FocusLearn's sensitivity to variations in n (number of features to select). Specifically, it will be important to explore whether FocusLearn can fully leverage the flexibility of n with large data sets that have a wide range of features to select from. This future work would be possible when SMART BEAR data becomes available, or any other similar large, high-dimensional real data can be sourced. Nevertheless, it is important to acknowledge that pursuing this future work might present significant challenges.

6.4.3 Subjective and objective XAI evaluation

Another area for future work is to explore and apply both subjective and objective XAI evaluation metrics to provide a robust and comprehensive evaluation of the explanations of FocusLearn. Although there are currently no universally accepted objective metrics for evaluating XAI methods, it is imperative to explore and apply existing metrics, such as Rosenfeld's set [118], to these methods. Besides the Rosenfeld's set [118], other methods that can be taken into consideration include evaluating the fidelity (truthfulness and faithfulness) of the explanations [157], the robustness of the explanations [7], as well as the sensitivity of the explanations [14]. However, as there are currently no universally accepted objective metrics for evaluating XAI methods, these evaluation methods will need to be thoroughly examined to carefully assess their advantages and disadvantages. Nevertheless, this is an essential step to take for establishing the trustworthiness of FocusLearn across different applications and domains.

6.4.4 Extending from time series to NLP

As this work has a focus on multivariate time series analysis, another potential area for future research is to extend FocusLearn to other domains with tabular data. Additionally, given that FocusLearn utilises RNNs and an attention mechanism during training, there is potential to extend its application beyond time series modelling to other domains, such as natural language processing (NLP), in future work. Text data can be viewed as sequential data, where the meaning of a word is influenced by its surrounding context within a sentence. This sequential nature is why models like Transformers, RNNs, and attention mechanisms are widely employed in the NLP domain. Exploring this direction is promising, as extending FocusLearn to provide explanations for NLP models might significantly enhance our understanding and interpretability of complex language models, making it a valuable area for future research.

6.5 Ethical Consideration

This research contributes to the growing field of interpretable machine learning, with the goal of making predictive models more transparent and explainable. Interpretability is particularly important in domains such as healthcare and decision support systems, where black-box or non-interpretable models can lead to consequences that are difficult to justify or audit.

By prioritising interpretability in model design, this work supports ethical principles such as accountability, transparency, and fairness. The ability to generate explanation graphs and feature importance insights allows stakeholders – including domain experts and end users – to understand, critique, and potentially contest the predictions made by the model. This is especially relevant in high-stakes applications, where decisions may affect individuals' health, safety, or access to resources.

While this study does not directly involve human subjects or personal data, the models developed here could be deployed in settings where ethical considerations must be addressed. Future work could include human-centred evaluation of explanations to ensure they are not only technically sound but also meaningful and actionable for decision-makers.

6.6 Final Remarks

The ultimate goal of this research is to develop an interpretable time series model that can predict better than state-of-the-art interpretable models, such as Neural Additive Model (NAM) [4] and Scalable Polynomial Additive Model (SPAM) [42], and match the predictive performance of non-interpretable models, such as Long Short-term Memory (LSTM) [58] and its subsequent advancements aimed at improving its predictive performance on multivariate time series data, as well as be able to provide explanations without relying on post-hoc methods. Experiments on two regression, *Air* and *Energy*, and two classification data sets, *EEG* and *Weather*, as well as one hearing loss use case; *OtiReal* have shown that Fo-

cusLearn, the developed interpretable time series model has met these objectives. This research therefore bridges the gap between predictive performance and interpretability, offering a new approach that could influence future developments in the field, making AI systems more trustworthy and widely applicable.

The main idea behind FocusLearn draws inspiration from the principles of modularity [19, 143], additive models [4, 42], and the idea that interpretability can be enhanced through linearity [17]. However, the actual development and refinement of the approach were significantly guided by insights gained from experimental results, namely the experiment on attention-based LSTM (AttnLSTM) and attention-based encoder-decoder (AttnED). From these experiments, we have developed several novel ideas to improve the predictive performance of FocusLearn: Attention-based Feature Selection (AFS), Attention-based Node Bootstrapping (ANB), and Dual-Paths. Results from ablation experiments have shown that AFS and ANB are highly effective at improving the predictive performance of FocusLearn, and by adopting the Dual-Paths method, not only the predictive performance of FocusLearn improves, but FocusLearn is also as explainable as NAM at inference.

The limitations of developed models are discussed in section 6.3, which include computational constraints, or that FocusLearn may not fully leverage the flexibility of n (number of features to select). These limitations will be addressed in future research. One promising avenue is to extend the application of FocusLearn beyond time series modelling to natural language processing, given the similarities between time series and text data.

I believe strongly in the potential of this work, as the findings encourage the use of AI systems in domains that currently do not use AI systems, and would greatly benefit from these technological improvements. I am hopeful that this research will spark further exploration and creativity in the field of XAI, especially given the growing importance of XAI highlighted in recent policies like the EU AI Act. As we push the boundaries of XAI, the insights from this research underscore our ongoing quest for knowledge and the exciting opportunities that lie ahead.

Bibliography

[1] Kjersti Aas, Martin Jullum, and Anders Løland. "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values". In: *Artificial Intelligence* 298 (Sept. 2021), p. 103502. ISSN: 00043702. DOI: 10.1016/j.artint.2021.103502.

[2] Ginanjar Abdurrahman and Mukti Sintawati. "Implementation of xgboost for classification of parkinson's disease". In: *Journal of Physics: Conference Series* 1538.1 (May 2020), p. 012024. ISSN: 1742-6588. DOI: 10.1088/1742-6596/1538/1/012024.

[3] Amina Adadi and Mohammed Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2870052.

[4] Rishabh Agarwal et al. "Neural additive models: Interpretable machine learning with neural nets". In: *Advances in Neural Information Processing Systems* 34 (2021).

[5] David Alvarez Melis and Tommi Jaakkola. "Towards Robust Interpretability with Self-Explaining Neural Networks". In: *Advances in Neural Information Processing Systems* 31 (2018).

[6] David Alvarez-Melis and Tommi S Jaakkola. "Towards robust interpretability with self-explaining neural networks". In: *Advances in neural information processing systems* 31 (2018).

[7] David Alvarez-Melis and Tommi S. Jaakkola. "On the Robustness of Interpretability Methods". In: *arXiv: 1806.08049* (June 2018).

[8] Sercan Ö. Arik and Tomas Pfister. "TabNet: Attentive Interpretable Tabular Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.8 (Aug. 2021), pp. 6679–6687. DOI: 10.1609/aaai.v35i8.16826.

[9] Emre Ates et al. "Counterfactual Explanations for Multivariate Time Series". In: *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*. IEEE, May 2021, pp. 1–8. ISBN: 978-1-7281-5934-8. DOI: 10.1109/ICAPAI49758.2021.9462056.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *arXiv: 1607.06450* (July 2016).

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv: 1409.0473* (Sept. 2014).

[12] Leonard E Baum. "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes". In: *Inequalities* 3.1 (1972), pp. 1–8.

[13] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181.

[14] Umang Bhatt, Adrian Weller, and José M. F. Moura. "Evaluating and Aggregating Feature-based Model Explanations". In: *arXiv: 2005.00631* (May 2020).

[15] H. Bourlard and N. Morgan. "Continuous speech recognition by connectionist statistical methods". In: *IEEE Transactions on Neural Networks* 4.6 (1993), pp. 893–909. ISSN: 10459227. DOI: 10.1109/72.286885.

[16] Leo Breiman. "Random Forest". In: *Machine Learning* 45.1 (2001). ISSN: 08856125. DOI: 10.1023/A:1010933404324.

[17] Trenton Bricken et al. "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning". In: *Transformer Circuits Thread* (2023).

[18] Nadia Burkart and Marco F Huber. "A Survey on the Explainability of Supervised Machine Learning". In: *J. Artif. Int. Res.* 70 (May 2021), pp. 245–317. ISSN: 1076-9757. DOI: 10.1613/jair.1.12228.

[19] Terry Caelli, Guan Ling, and Wilson Wen. "Modularity in neural computing". In: *Proceedings of the IEEE* 87.9 (1999), pp. 1497–1518. ISSN: 00189219. DOI: 10.1109/5.784227.

[20] Luis Candanedo. *Appliances Energy Prediction*. 2017.

[21] Lin Cao et al. "A Multiattention-Based Supervised Feature Selection Method for Multivariate Time Series". In: *Computational Intelligence and Neuroscience* 2021 (July 2021), pp. 1–10. ISSN: 1687-5273. DOI: 10.1155/2021/6911192.

[22] T. Catfolis and K. Meert. "Implementing empirical modelling techniques with recurrent neural networks". In: *Proceedings Eighth IEEE International Conference on Tools with Artificial Intelligence*. IEEE Comput. Soc. Press, 1996, pp. 434–435. ISBN: 0-8186-7686-8. DOI: 10.1109/TAI.1996.560746.

[23] Kitty S. Chan, Jinnet B. Fowles, and Jonathan P. Weiner. "Review: Electronic Health Records and the Reliability and Validity of Quality Measures: A Review of the Literature". In: *Medical Care Research and Review* 67.5 (Oct. 2010), pp. 503–527. ISSN: 1077-5587. DOI: 10.1177/1077558709359007.

[24] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: (June 2011). DOI: 10.1613/jair.953.

[25] Ke Chen. "Deep and Modular Neural Networks". In: *Springer Handbook of Computational Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 473–494. DOI: 10.1007/978-3-662-43505-2{_}28.

[26] Kehai Chen et al. "Syntax-Directed Attention for Neural Machine Translation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). ISSN: 2374-3468. DOI: 10.1609/aaai.v32i1.11910.

[27] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785.

[28] Ziyue Chen et al. "Detecting the causality influence of individual meteorological factors on local PM2.5 concentration in the Jing-Jin-Ji region". In: *Scientific Reports* 7.1 (Jan. 2017), p. 40735. ISSN: 2045-2322. DOI: 10.1038/srep40735.

[29] Li-Chen Cheng, Yu-Hsiang Huang, and Mu-En Wu. "Applied attention-based LSTM neural networks in stock prediction". In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec. 2018, pp. 4716–4718. ISBN: 978-1-5386-5035-6. DOI: 10.1109/BigData.2018.8622541.

[30] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv: 1406.1078* (June 2014).

[31] Kyunghyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *arXiv: 1409.1259* (Sept. 2014).

[32] Kyu Sung Choi, Seung Hong Choi, and Bumseok Jeong. "Prediction of IDH genotype in gliomas with dynamic susceptibility contrast perfusion MR imaging using an explainable recurrent neural network". In: *Neuro-Oncology* 21.9 (Sept. 2019), pp. 1197–1209. ISSN: 1522-8517. DOI: 10.1093/neuonc/noz095.

[33] Jeppe H. Christensen et al. "The everyday acoustic environment and its association with human heart rate: evidence from real-world data logging with hearing aids and wearables". In: *Royal Society Open Science* 8.2 (Feb. 2021), rsos.201345. ISSN: 2054-5703. DOI: 10.1098/rsos.201345.

[34] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *arXiv: 1412.3555* (Dec. 2014).

[35] Michael P. Clements and Grayham E. Mizon. "Empirical analysis of macroeconomic time series". In: *European Economic Review* 35.4 (May 1991), pp. 887–917. ISSN: 00142921. DOI: 10.1016/0014-2921(91)90042-H.

[36] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (Sept. 1995). ISSN: 0885-6125. DOI: 10.1007/BF00994018.

[37] Susanne Dandl et al. "Multi-Objective Counterfactual Explanations". In: 2020, pp. 448–469. DOI: 10.1007/978-3-030-58112-1.

[38] Yanzhuo Ding et al. "Visualizing and Understanding Neural Machine Translation". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2017, pp. 1150–1159. DOI: 10.18653/v1/P17-1106.

[39] Derek Doran, Sarah Schulz, and Tarek R Besold. "What does explainable AI really mean? A new conceptualization of perspectives". In: *arXiv:1710.00794* (2017).

[40] Mengnan Du, Ninghao Liu, and Xia Hu. "Techniques for Interpretable Machine Learning". In: *Communications of the ACM* 63.1 (2019), pp. 68–77. DOI: 10.1145/3359786.

[41] Shengdong Du et al. "Multivariate time series forecasting via attention-based encoder–decoder framework". In: *Neurocomputing* 388 (May 2020), pp. 269–279. ISSN: 09252312. DOI: 10.1016/j.neucom.2019.12.118.

[42] Abhimanyu Dubey, Filip Radenovic, and Dhruv Mahajan. "Scalable Interpretability via Polynomials". In: *arXiv: 2205.14108* (May 2022).

[43] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211. DOI: 10.1207/s15516709cog1402.

[44] Chenyou Fan et al. "Multi-Horizon Time Series Forecasting with Temporal Attention Learning". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2527–2535. ISBN: 9781450362016. DOI: 10.1145/3292500.3330662.

[45] Zheng-gang Fang et al. "Application of a data-driven XGBoost model for the prediction of COVID-19 in the USA: a time-series study". In: *BMJ Open* 12.7 (July 2022), e056685. ISSN: 2044-6055. DOI: 10.1136/bmjopen-2021-056685.

[46] Zicheng Fang, Peng Wang, and Wei Wang. "Efficient Learning Interpretable Shapelets for Accurate Time Series Classification". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, Apr. 2018, pp. 497–508. ISBN: 978-1-5386-5520-7. DOI: 10.1109/ICDE.2018.00052.

[47] Leif Feddersen and Catherine Cleophas. "Hierarchical Neural Additive Models for Interpretable Demand Forecasts". In: *arXiv: 2404.04070* (Apr. 2024).

[48] Dingfei Ge, Narayanan Srinivasan, and Shankar M Krishnan. "Cardiac arrhythmia classification using autoregressive modeling". In: *BioMedical Engineering OnLine* 1.1 (Dec. 2002), p. 5. ISSN: 1475-925X. DOI: 10.1186/1475-925X-1-5.

[49] Wendong Ge et al. "An Interpretable ICU Mortality Prediction Model Based on Logistic Regression and Recurrent Neural Networks with LSTM units." In: *AMIA Annual Symposium Proceedings*. Vol. 2018. American Medical Informatics Association, 2018, pp. 460–469.

[50] Amirata Ghorbani, Abubakar Abid, and James Zou. "Interpretation of Neural Networks Is Fragile". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 3681–3688. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33013681.

[51] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[52] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[53] Ning Gui, Danni Ge, and Ziyin Hu. "AFS: An Attention-Based Mechanism for Supervised Feature Selection". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 3705–3713. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33013705.

[54] Odd Erik Gundersen, Yolanda Gil, and David W. Aha. "On Reproducible AI: Towards Reproducible Research, Open Science, and Digital Scholarship in AI Publications". In: *AI Magazine* 39.3 (Sept. 2018), pp. 56–68. ISSN: 0738-4602. DOI: 10.1609/aimag.v39i3.2816.

[55] Amine Haboub, Hamza Baali, and Abdesselam Bouzerdoum. "Multichannel Signal Classification Using Vector Autoregression". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal*

Processing (ICASSP). IEEE, May 2020, pp. 1021–1025. ISBN: 978-1-5090-6631-5. DOI: 10.1109/ICASSP40776.2020.9054144.

[56] Sridhar S Hannenhalli and Robert B Russell. "Analysis and prediction of functional sub-types from protein sequence alignments". In: *Journal of molecular biology* 303.1 (2000), pp. 61–76. DOI: 10.1006/jmbi.2000.4036.

[57] Trevor Hastie and Robert Tibshirani. "Generalized additive models for medical research". In: *Statistical methods in medical research* 4.3 (1995), pp. 187–196. ISSN: 0883-4237. DOI: 10.1214/ss/1177013604.

[58] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[59] Wei-Chiang Hong. "Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting". In: *Neural Computing and Applications* 21.3 (Apr. 2012). ISSN: 0941-0643. DOI: 10.1007/s00521-010-0456-7.

[60] Tomas Hrycej. *Modular Learning in Neural Networks: A Modularized Approach to Neural Network Classification*. 1st. John Wiley \& Sons, Inc., 1992.

[61] Yuxiu Hua et al. "Deep learning with long short-term memory for time series prediction". In: *IEEE Communications Magazine* 57.6 (2019), pp. 114–119. DOI: 10.1109/MCOM.2019.1800155.

[62] Siteng Huang et al. "DSANet: Dual Self-Attention Network for Multivariate Time Series Forecasting". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 2129–2132. ISBN: 9781450369763. DOI: 10.1145/3357384.3358132.

[63] Yang Huang, Chiun-Hsun Chen, and Chi-Jui Huang. "Motor Fault Detection and Feature Extraction Using RNN-Based Variational Autoencoder". In: *IEEE Access* 7 (2019). ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2940769.

[64] Eleftheria Iliadou et al. "Profiling Hearing Aid Users through Big Data Explainable Artificial Intelligence Techniques". In: *Frontiers in Neurology* (2022). DOI: 10.3389/fneur.2022.933940.

[65] Ihab F Ilyas and Xu Chu. *Data cleaning*. ACM, 2019.

[66] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv: 1502.03167* (Feb. 2015).

[67] Aya Abdelsalam Ismail et al. "Benchmarking deep learning interpretability in time series predictions". In: *Advances in neural information processing systems* 33 (2020), pp. 6441–6452.

[68] Taku Ito, Keiichi Ochiai, and Yusuke Fukazawa. "C-LIME: A Consistency-Oriented LIME for Time-Series Health-Risk Predictions". In: *Knowledge Management and Acquisition for Intelligent Systems*. Vol. 12280. 2021, pp. 58–69. DOI: 10.1007/978-3-030-69886-7.

[69] Joseph D Janizek, Pascal Sturmfels, and Su-In Lee. "Explaining explanations: Axiomatic feature interactions for deep networks". In: *Journal of Machine Learning Research* 22 (2021), pp. 1–54.

[70] Ran Jing. "A Self-attention Based LSTM Network for Text Classification". In: *Journal of Physics: Conference Series* 1207 (Apr. 2019), p. 012008. ISSN: 1742-6588. DOI: 10.1088/1742-6596/1207/1/012008.

[71] Wonkeun Jo and Dongil Kim. "Neural additive time-series models: Explainable deep learning for multivariate time-series prediction". In: *Expert Systems with Applications* 228 (Oct. 2023), p. 120307. ISSN: 09574174. DOI: 10.1016/j.eswa.2023.120307.

[72] Argyro Kampouraki, George Manis, and Christophoros Nikou. "Heartbeat Time Series Classification With Support Vector Machines". In: *IEEE Transactions on Information Technology in Biomedicine* 13.4 (July 2009), pp. 512–518. ISSN: 1089-7771. DOI: 10.1109/TITB.2008.2003323.

[73] Kathan Kashiparekh et al. "ConvTimeNet: A Pre-trained Deep Convolutional Neural Network for Time Series Classification". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2019, pp. 1–8. ISBN: 978-1-7281-1985-4. DOI: 10.1109/IJCNN.2019.8852105.

[74] Shruti Kaushik et al. "AI in Healthcare: Time-Series Forecasting Using Statistical, Neural, and Ensemble Architectures". In: *Frontiers in Big Data* 3 (2020), p. 4. ISSN: 2624-909X. DOI: 10.3389/fdata.2020.00004. URL: https://www.frontiersin.org/article/10.3389/fdata.2020.00004.

[75] Minkyu Kim, Hyun-Soo Choi, and Jinho Kim. "Higher-order Neural Additive Models: An Interpretable Machine Learning Model with Feature Interactions". In: *arXiv: 2209.15409* (Sept. 2022).

[76] Louis Kirsch, Julius Kunze, and David Barber. "Modular networks: Learning to decompose neural computation". In: *Advances in neural information processing systems* 31 (2018).

[77] Maxim S. Kovalev and Lev V. Utkin. "A robust algorithm for explaining unreliable machine learning survival models using the Kolmogorov–Smirnov bounds". In: *Neural Networks* 132 (Dec. 2020), pp. 1–18. ISSN: 08936080. DOI: 10.1016/j.neunet.2020.08.007.

[78] Maxim S. Kovalev, Lev V. Utkin, and Ernest M. Kasimov. "SurvLIME: A method for explaining machine learning survival models". In: *Knowledge-Based Systems* 203 (Sept. 2020), p. 106164. ISSN: 09507051. DOI: 10.1016/j.knosys.2020.106164.

[79] Max Kuhn and Kjell Johnson. *Feature engineering and selection: A practical approach for predictive models*. CRC Press, 2019.

[80] Manish Kumar and Thenmozhi M. "Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest". In: *SSRN Electronic Journal* (2006). ISSN: 1556-5068. DOI: 10.2139/ssrn.876544.

[81] Ariane Laplante-Lévesque et al. "Data-Driven Hearing Care With Hearing FitnessTM: Oticon Shares Openly Its Innovative Vision and Roadmap". In: *Copenhagen, Denmark: Oticon A/S* (2017), pp. 1–7.

[82] Ryno Laubscher. "Time-series forecasting of coal-fired power plant reheater metal temperatures using encoder-decoder recurrent neural networks". In: *Energy* 189 (Dec. 2019), p. 116187. ISSN: 03605442. DOI: 10.1016/j.energy.2019.116187.

[83] Logan Lebanoff, Kaiqiang Song, and Fei Liu. "Adapting the Neural Encoder-Decoder Framework from Single to Multi-Document Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018, pp. 4131–4141. DOI: 10.18653/v1/D18-1446.

[84] Jiwei Li et al. "Visualizing and Understanding Neural Models in NLP". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*. Association for Computational Linguistics, 2016, pp. 681–691. DOI: 10.18653/v1/N16-1082.

[85] Liam Li et al. "A System for Massively Parallel Hyperparameter Tuning". In: *arXiv: 1810.05934* (Oct. 2018).

[86] Youru Li et al. "EA-LSTM: Evolutionary attention-based LSTM for time series prediction". In: *Knowledge-Based Systems* 181 (Oct. 2019), p. 104785. ISSN: 09507051. DOI: 10.1016/j.knosys.2019.05.028.

[87] Bryan Lim et al. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting". In: *International Journal of Forecasting* 37.4 (Oct. 2021), pp. 1748–1764. ISSN: 0169-2070. DOI: 10.1016/J.IJFORECAST.2021.03.012.

[88] Zachary C. Lipton. "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." In: *Queue* 16.3 (2018), pp. 31–57.

[89] Gang Liu and Jiabao Guo. "Bidirectional LSTM with attention mechanism and convolutional layer for text classification". In: *Neurocomputing* 337 (Apr. 2019), pp. 325–338. ISSN: 09252312. DOI: 10.1016/j.neucom.2019.01.078.

[90] Liyuan Liu et al. "On the Variance of the Adaptive Learning Rate and Beyond". In: *arXiv: 1908.03265v4* (Aug. 2019).

[91] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *arXiv: 1608.03983* (Aug. 2016).

[92] Yin Lou et al. "Accurate intelligible models with pairwise interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 623–631. ISBN: 9781450321747. DOI: 10.1145/2487575.2487579.

[93] Yin Lou et al. "Accurate intelligible models with pairwise interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, Aug. 2013, pp. 623–631. ISBN: 9781450321747. DOI: 10.1145/2487575.2487579.

[94] Simon M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Proceedings of the 31st international conference on neural information processing systems*. 2017, pp. 4768–4777.

[95] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *arXiv: 1508.04025* (Aug. 2015).

[96] Ričards Marcinkevičs and Julia E Vogt. "Interpretability and explainability: A machine learning zoo mini-tour". In: *arXiv: 2012.01805* (2020).

[97] Joao Marques-Silva and Xuanxiang Huang. "Explainability Is Not a Game". In: *Commun. ACM* 67.7 (2024), pp. 66–75.

[98] Masayoshi Mase, Art B. Owen, and Benjamin Seiler. "Explaining black box decisions by Shapley cohort refinement". In: *arXiv: 1911.00467* (Nov. 2019).

[99] Colin Mccormack. "Adaptation of Learning Rule Parameters Using a Meta Neural Network". In: *Connection Science* 9.1 (Mar. 1997), pp. 123–136. ISSN: 0954-0091. DOI: 10.1080/095400997116775.

[100] Karim El Mokhtari, Ben Peachey Higdon, and Ayşe Başar. "Interpreting Financial Time Series with SHAP Values". In: *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*. 2019, pp. 166–172. DOI: 10.5555/3370272.3370290.

[101] Christoph Molnar. *Interpretable machine learning*. Lulu.com, 2020.

[102] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. "Explaining machine learning classifiers through diverse counterfactual explanations". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 607–617. ISBN: 9781450369367. DOI: 10.1145/3351095.3372850.

[103] W. James Murdoch, Peter J. Liu, and Bin Yu. "Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs". In: *arXiv:1801.05453* (Jan. 2018).

[104] Nguyet Nguyen. "Hidden Markov Model for Stock Trading". In: *International Journal of Financial Studies* 6.2 (2018). ISSN: 2227-7072. DOI: 10.3390/ijfs6020036.

[105] An-Phi Nguyen et al. "MonoNet: enhancing interpretability in neural networks via monotonic features". In: *Bioinformatics Advances* 3.1 (Jan. 2023). ISSN: 2635-0041. DOI: 10.1093/bioadv/vbad016.

[106] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. "A review on the attention mechanism of deep learning". In: *Neurocomputing* 452 (Sept. 2021), pp. 48–62. ISSN: 09252312. DOI: 10.1016/j.neucom.2021.03.091.

[107] Oliver Roesler. *EEG Eye State*. July 2023. DOI: 10.24432/C57G7J.

[108] Boris N Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *arXiv: 1905.10437* (2019).

[109] Rui Pedro Paiva and António Dourado. "Interpretability and learning in neuro-fuzzy systems". In: *Fuzzy Sets and Systems* 147.1 (Oct. 2004), pp. 17–38. ISSN: 0165-0114. DOI: 10.1016/J.FSS.2003.11.012.

[110] Leonardos Pantiskas, Kees Verstoep, and Henri Bal. "Interpretable Multivariate Time Series Forecasting with Temporal Attention Convolutional Neural Networks". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2020, pp. 1687–1694. ISBN: 978-1-7281-2547-3. DOI: 10.1109/SSCI47803.2020.9308570.

[111] Lawrence Rabiner and Biinghwang Juang. "An introduction to hidden Markov models". In: *IEEE ASSP Magazine* 3.1 (1986), pp. 4–16. DOI: 10.1109/MASSP.1986.1165342.

[112] Lawrence R. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (Feb. 1989), pp. 257–286. ISSN: 1558-2256. DOI: 10.1109/5.18626.

[113] Filip Radenovic, Abhimanyu Dubey, and Dhruv Mahajan. "Neural Basis Models for Interpretability". In: *arXiv: 2205.14120* (May 2022).

[114] Khairan Rajab, Firuz Kamalov, and Aswani Kumar Cherukuri. "Forecasting COVID-19: Vector Autoregression-Based Model". In: *Arabian Journal for Science and Engineering* 47.6 (June 2022), pp. 6851–6860. ISSN: 2193-567X. DOI: 10.1007/s13369-021-06526-2.

[115] Suresh Ramakrishnan et al. "Forecasting Malaysian exchange rate using machine learning techniques based on commodities prices". In: *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*. IEEE, July 2017, pp. 1–5. ISBN: 978-1-5090-3035-4. DOI: 10.1109/ICRIIS.2017.8002544.

[116] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. Association for Computing Machinery, 2016, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

[117] Thomas Rojat et al. "Explainable Artificial Intelligence (XAI) on TimeSeries Data: A Survey". In: *arXiv:2104.00950* (Apr. 2021).

[118] Avi Rosenfeld. "Better Metrics for Evaluating Explainable Artificial Intelligence". In: *20th International Foundation for Autonomous Agents and Multiagent Systems (AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, 2021, pp. 45–50. DOI: 10.5555/3463952.3463962.

[119] Andrew Ross, Isaac Lage, and Finale Doshi-Velez. "The neural lasso: Local linear sparsity for interpretable explanations". In: *Workshop on Transparent and Interpretable Machine Learning in Safety Critical Environments, 31st Conference on Neural Information Processing Systems*. 2017.

[120] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x.

[121] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." In: *Proceedings of the Annual Conference of International Speech Communication Association*. 2014.

[122] Cedric Schockaert, Reinhard Leperlier, and Assaad Moawad. "Attention Mechanism for Multivariate Time Series Recurrent Model Interpretability Applied to the Ironmaking Industry". In: *arXiv:2007.12617, 2020* (July 2020).

[123] Mike Schuster and Kuldip K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. ISSN: 1053587X. DOI: 10.1109/78.650093.

[124] Pavel Senin and Sergey Malinchik. "SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model". In: *2013 IEEE 13th International Conference on Data Mining*. IEEE, Dec. 2013, pp. 1175–1180. ISBN: 978-0-7695-5108-1. DOI: 10.1109/ICDM.2013.52.

[125] Akhil Sethia and Purva Raut. "Application of LSTM, GRU and ICA for stock price prediction". In: *Information and Communication Technology for Intelligent Systems*. Springer, 2019, pp. 479–487. DOI: 10.1007/978-981-13-1747-7{_}46.

[126] Wenling Shang et al. "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units". In: *arXiv: 1603.05201* (Mar. 2016).

[127] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games 2.28*. 1953, pp. 307–317.

[128] AMANDA J. C SHARKEY. "Modularity, Combining and Artificial Neural Nets". In: *Connection Science* 9.1 (Mar. 1997), pp. 3–10. ISSN: 0954-0091. DOI: 10.1080/095400997116702.

[129] Garima Sharma et al. "Comparative Analysis of Supervised Models for Diamond Price Prediction". In: *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, Jan. 2021. ISBN: 978-1-6654-1451-7. DOI: 10.1109/Confluence51648.2021.9377183.

[130] Tao Shen et al. "DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). ISSN: 2374-3468. DOI: 10.1609/aaai.v32i1.11941.

[131] Liang Shi, Chen Qian, and Feng Guo. "Real-time driving risk assessment using deep learning with XGBoost". In: *Accident Analysis & Prevention* 178 (Dec. 2022), p. 106836. ISSN: 00014575. DOI: 10.1016/j.aap.2022.106836.

[132] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, 2017, pp. 3145–3153. DOI: 10.5555/3305890.3306006.

[133] Dylan Slack et al. "Fooling LIME and SHAP". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 180–186. ISBN: 9781450371100. DOI: 10.1145/3375627.3375830.

[134] Hendrik Strobelt et al. "LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 667–676. ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2744158.

[135] Qiqi Su and Eleftheria Iliadou. "Predicting and Explaining Hearing Aid Usage Using Encoder-Decoder with Attention Mechanism and SHAP". In: *2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, Oct. 2022, pp. 308–315. ISBN: 978-1-6654-6495-6. DOI: 10.1109/SITIS57111.2022.00053.

[136] Md Arif Istiake Sunny, Mirza Mohd Shahriar Maswood, and Abdullah G. Alharbi. "Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model". In: *2nd Novel Intelligent and Leading*

Emerging Sciences Conference (NILES). 2020, pp. 87–92. DOI: 10.1109/NILES50944.2020.9257950.

[137] Vinitra Swamy et al. "MultiModN—multimodal, multi-task, interpretable modular networks". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. New Orleans, LA, USA: Curran Associates Inc., 2024.

[138] Kenichi Tatsumi et al. "Crop classification of upland fields using Random forest of time-series Landsat 7 ETM+ data". In: *Computers and Electronics in Agriculture* 115 (July 2015), pp. 171–179. ISSN: 01681699. DOI: 10.1016/j.compag.2015.05.001.

[139] Cécile Trottet et al. "Modular Clinical Decision Support Networks (MoDN)—Updatable, interpretable, and portable predictions for evolving clinical environments". In: *PLOS Digital Health* 2.7 (July 2023), e0000108. ISSN: 2767-3170. DOI: 10.1371/journal.pdig.0000108.

[140] Michael Tsang et al. "Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability." In: *Advances in Neural Information Processing Systems* 31 (2018).

[141] Nishant Uniyal et al. "Ultrasound RF Time Series for Classification of Breast Lesions". In: *IEEE Transactions on Medical Imaging* 34.2 (Feb. 2015), pp. 652–661. ISSN: 0278-0062. DOI: 10.1109/TMI.2014.2365030.

[142] Lev V. Utkin, Egor D. Satyukov, and Andrei V. Konstantinov. "SurvNAM: The machine learning survival model explanation". In: *Neural Networks* 147 (Mar. 2022), pp. 81–102. ISSN: 08936080. DOI: 10.1016/j.neunet.2021.12.015.

[143] Sergio Varela-Santos and Patricia Melin. "A new modular neural network approach with fuzzy response integration for lung disease classification based on multiple objective feature optimization in chest X-ray images". In: *Expert Systems with Applications* 168 (Apr. 2021), p. 114361. ISSN: 09574174. DOI: 10.1016/j.eswa.2020.114361.

[144] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv:1706.03762* (June 2017).

[145] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015, pp. 3156–3164. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298935.

[146] John Violos et al. "Self-Attention based encoder-Decoder for multistep human density prediction". In: *Journal of Urban Mobility* 2 (Dec. 2022), p. 100022. ISSN: 26670917. DOI: 10.1016/j.urbmob.2022.100022.

[147] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR". In: *Harv. JL \& Tech.* 31 (2017), p. 841.

[148] Jingyuan Wang et al. "Deep Fuzzy Cognitive Maps for Interpretable Multivariate Time Series Prediction". In: *IEEE Transactions on Fuzzy Systems* 29.9 (Sept. 2021), pp. 2647–2660. ISSN: 1063-6706. DOI: 10.1109/TFUZZ.2020.3005293.

[149] Yan Wang and Yuankai Guo. "Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost". In: *China Communications* 17.3 (Mar. 2020), pp. 205–221. ISSN: 1673-5447. DOI: 10.23919/JCC.2020.03.017.

[150] Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, May 2017, pp. 1578–1585. ISBN: 978-1-5090-6182-2. DOI: 10.1109/IJCNN.2017.7966039.

[151] Adam White and Artur d'Avila Garcez. "Measurable Counterfactual Local Explanations for Any Classifier". In: *arXiv:1908.03020* (Aug. 2019).

[152] Beate Wild et al. "A graphical vector autoregressive modelling approach to the analysis of electronic diary data". In: *BMC Medical Research Methodology* 10.1 (Dec. 2010), p. 28. ISSN: 1471-2288. DOI: 10.1186/1471-2288-10-28.

[153] Xuefeng Xi, Zhou Pi, and Guodong Zhou. "Global Encoding for Long Chinese Text Summarization". In: *ACM Transactions on Asian and Low-*

Resource Language Information Processing 19.6 (Nov. 2020), pp. 1–17. ISSN: 2375-4699. DOI: 10.1145/3407911.

[154] Xiaoxia Xiong, Long Chen, and Jun Liang. "A New Framework of Vehicle Collision Prediction by Combining SVM and HMM". In: *IEEE Transactions on Intelligent Transportation Systems* 19.3 (2018), pp. 699–710. DOI: 10.1109/TITS.2017.2699191.

[155] Kelvin Xu et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *Proceedings of Machine Learning Research*. PMLR, 2015, pp. 2048–2057.

[156] Lexiang Ye and Eamonn Keogh. "Time series shapelets". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*. 2009, p. 947. ISBN: 9781605584959. DOI: 10.1145/1557019.1557122.

[157] Chih-Kuan Yeh et al. "On the (in) fidelity and sensitivity of explanations". In: *Advances in neural information processing systems* 32 (2019).

[158] Lean Yu, Xun Zhang, and Shouyang Wang. "Assessing Potentiality of Support Vector Machine Method in Crude Oil Price Forecasting". In: *EURASIA Journal of Mathematics, Science and Technology Education* 13.12 (Nov. 2017). ISSN: 13058223. DOI: 10.12973/ejmste/77926.

[159] Xiang Yu et al. "Novel hybrid multi-head self-attention and multifractal algorithm for non-stationary time series prediction". In: *Information Sciences* 613 (Oct. 2022), pp. 541–555. ISSN: 00200255. DOI: 10.1016/j.ins.2022.08.126.

[160] Yuan Yuan et al. "Using An Attention-Based LSTM Encoder–Decoder Network for Near Real-Time Disturbance Detection". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 1819–1832. ISSN: 1939-1404. DOI: 10.1109/JSTARS.2020.2988324.

[161] Lotfi A. Zadeh. "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3.1 (1973), pp. 28–44. ISSN: 0018-9472. DOI: 10.1109/TSMC.1973.5408575.

[162] Ailing Zeng et al. "Are Transformers Effective for Time Series Forecasting?" In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.9 (June 2023), pp. 11121–11128. ISSN: 2374-3468. DOI: 10.1609/aaai.v37i9.26317.

[163] Pengyu Zeng et al. "Muformer: A long sequence time-series forecasting model based on modified multi-head attention". In: *Knowledge-Based Systems* 254 (Oct. 2022), p. 109584. ISSN: 09507051. DOI: 10.1016/j.knosys.2022.109584.

[164] Bo Zhang et al. "A novel Encoder-Decoder model based on read-first LSTM for air pollutant prediction". In: *Science of The Total Environment* 765 (Apr. 2021), p. 144507. ISSN: 00489697. DOI: 10.1016/j.scitotenv.2020.144507.

[165] Hao Zhang et al. "A temporal fusion transformer for short-term freeway traffic speed multistep prediction". In: *Neurocomputing* 500 (Aug. 2022), pp. 329–340. ISSN: 09252312. DOI: 10.1016/j.neucom.2022.05.083.

[166] Peter G. Zhang. "Time series forecasting using a hybrid ARIMA and neural network model". In: *Neurocomputing* 50 (Jan. 2003), pp. 159–175. ISSN: 0925-2312. DOI: 10.1016/S0925-2312(01)00702-0.

[167] Shuyi Zhang et al. "Cautionary tales on air-quality improvement in Beijing". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2205 (Sept. 2017), p. 20170457. ISSN: 1364-5021. DOI: 10.1098/rspa.2017.0457.

[168] Xuan Zhang et al. "AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction". In: *IOP Conference Series: Materials Science and Engineering* 569.5 (July 2019), p. 052037. ISSN: 1757-8981. DOI: 10.1088/1757-899X/569/5/052037.

[169] Zhongheng Zhang. "Missing data imputation: focusing on single imputation." In: *Annals of translational medicine* 4.1 (Jan. 2016), p. 9. ISSN: 2305-5839. DOI: 10.3978/j.issn.2305-5839.2015.12.38.

[170] Binggui Zhou et al. "Interpretable Temporal Attention Network for COVID-19 forecasting". In: *Applied Soft Computing* 120 (May 2022), p. 108691. ISSN: 15684946. DOI: 10.1016/j.asoc.2022.108691.

[171] Haoyi Zhou et al. “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.12 (May 2021), pp. 11106–11115. ISSN: 2374-3468. DOI: 10.1609/aaai.v35i12.17325.

Appendix A

Hidden Markov Model and N-Beats for Univariate Time Series

A.1 Hidden Markov Model

The observation sequence, O, is the given time series. In this analysis, the probability of the observation (i.e. the emission probability B) is assumed to be the Gaussian distribution, such that $b_j(v(t)) = \mathcal{N}\,(v(k) = v(t),\, \mu_i,\, \sigma_i)$ where μ_{ij} and σ_j are the mean and variance of the distribution for each state, $h(j)$, respectively. Therefore, the parameters of the HMM are modified to $\lambda = (A, \mu, \sigma, \pi)$ and can now be learned and adjusted by solving the Learning Problem of the fundamental questions that can be answered by an HMM, which is given the observation sequence, O, adjust the parameters λ to maximise $P(O|\lambda)$. Although there is no known optimal algorithm for estimating the model parameters given any fi-

nite number of observation sequence [112], we can however choose the parameters that maximise $P(O|\lambda)$ locally using an iterative approach. The Baum-Welch algorithm [12] is one of the most popular iterative approaches. The Baum-Welch algorithm is a special case of the Expectation-Maximisation (EM) algorithm and the Forward and Backward algorithms are applied at the Expectation step.

The model is trained on the known observation sequence $O = v(1)...v(T)$ with the number of hidden states set to five after grid search. The most probable next observation symbol, $v(T+1)$, can be predicted using the Forward Algorithm. This is done by constructing two HMM models.

For the first HMM model, the potential $v(T+1)$ is searched from the known $O = v(1)...v(T)$. The likelihood function $P[v(1)v(2)...v(T)v(T+1)|\lambda]$ is then calculated, and the $v(T+1)$ that gives the maximum likelihood under the model can be estimated as the future single observation symbol, named $v(T+1)_{HMM1}$. There is, however, a limitation when only searching for $v(T+1)$ from the known $O = v(1)...v(T)$, such as when the next real observation symbol never appeared in the known $O = v(1)...v(T)$.

A second HMM model therefore is constructed to overcome this limitation by training on the fraction of change between two consecutive known observation symbols, such that,

$$f_j = \frac{v(j+1) - v(j)}{v(j)}. \tag{A.1}$$

Similar steps to the first HMM is then followed. The likelihood function $P[f_1 f_2 ... f_T f_{T+1}|\lambda]$ is calculated and the fraction of change that gives the maximum likelihood function is estimated as the future fraction of change. The output of the second HMM is calculated as:

$$v(T+1)_{HMM2} = v(T) * (1 + f_{T+1}). \tag{A.2}$$

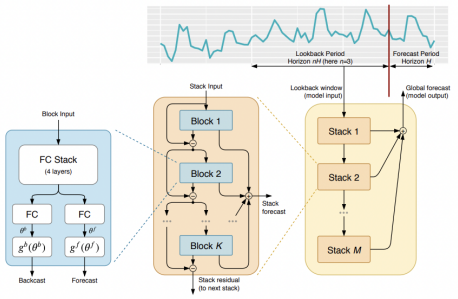


Figure A.1: N-Beats Architecture Proposed by Oreshkin *et al.* [108]

The final future observation symbol $v(\tilde{T}+1)$ is calculated by taking the average of the outputs of two HMM models, such that,

$$v(\tilde{T}+1) = \frac{v(T+1)_{HMM1} + v(T+1)_{HMM2}}{2}. \tag{A.3}$$

The results of either using the real observation symbols or the predicted observation symbols when predicting the next observation symbols are also compared.

A.2 N-Beats

N-Beats is a deep neural architecture proposed by Oreshekin *et al.* [108] that focuses on solving univariate time series problems, with interpretability being one of its desirable properties. N-Beats is composed of several building blocks that can be stacked together with doubly residuals, where one residual branch does backcast, the other forecast, and the input of the next block is constructed by subtracting the backcast of the output from the previous block. Oreshkin *et al.* [108] further defined two configurations of the architecture, namely generic and interpretable, where the interpretable architecture consists of two specifically defined blocks, trend and seasonality, in order to decompose the time series and make the outputs interpretable.

As shown in Figure A.1, the architecture of Interpretable N-Beats consists of two blocks, namely Trend Block and Seasonality Block, which are doubly

residual stacked on top of each other. The basis of each block is made up of a fully connected network followed by a backward, g_l^b, and a forward, g_l^f basis layer. The fully connected network has 4 fully connected layers and ReLU non-linearity that produces two predictors of expansion coefficients, backward θ_l^b and forward θ_l^f. The backward and forward basis layers then take their respective expansion coefficients and produce the backcast $\hat{x}_l$ and forecast $\hat{y}_l$ outputs by projecting the coefficients internally on the set of functions. Forecasts are then aggregated in a hierarchical fashion.

In the Trend Block, g_l^b and g_l^f are constrained to be a polynomial function of a small degree, p, that varies slowly across the forecast window. This function is trying to mimic the behaviour of trend, which is a monotonic function, or at least a slowly varying function [108]. When p is set low (e.g., 2 or 3), it forces the trend forecast to mimic the trend [108]. Therefore, p is set equal to 2 in this analysis. In the Seasonality Block, g_l^b and g_l^f are constrained to be a Fourier series in order to mimic the characteristics of seasonality, which is a regular, cyclical, recurring fluctuation [108].

Appendix B

Explanation Results of FocusLearn for Time Series Data

B.1 Explanations for the Air Data Set

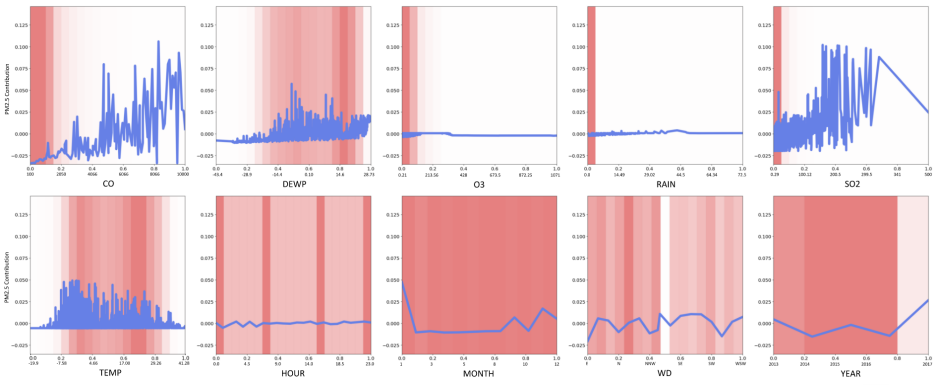


Figure B.1: Graphs learned by FocusLearn for Air.
in predicting future $PM_{2.5}$ values (regression) on the *Air* data set. These plots show top 10 features selected by the AFS component in the FocusLearn, where selected features with normalised and original values are on the x-axis, and future $PM_{2.5}$ values prediction contribution are on the y-axis.

The shape function plot for Carbon Monoxide (CO) shows an overall positive correlation with large variation. Similarly for Sulfur Dioxide ($SO2$), there is a positive correlation until the sudden drop when the value of $SO2$ exceeds 320 $\mu g/mg^3$. Notice how simply sampling the input to the LSTM to obtain such correlations would not be informative or feasible in practice with an increased number of input variables.

The plot for Dew Point Temperature ($DEWP$) shows that when $DEWP$ is at around 0 degree Celsius, predicted future $PM_{2.5}$ values tend to be the highest. The contribution of $DEWP$ towards predicting future $PM_{2.5}$ values then drops at around 7 degree Celsius and increases again at around 12 degree Celsius.

Ozone Concentration ($O3$), $Rain$, $HOUR$ contribute almost nothing towards predicting future $PM_{2.5}$ values, and the contribution of $O3$ even becomes negative with high $O3$ values (*e.g.,* higher $O3$ values leads to lower future $PM_{2.5}$ values).

The plot for Temperature ($TEMP$) shows that when temperature is in the range of -8 and 28 degrees Celsius, future $PM_{2.5}$ values are predicted to be the highest, whereas when the temperature is above 28 and below -8 degree Celsius, this will lead to a low future $PM_{2.5}$ values. This observation is in line with many literature studying correlation between $PM_{2.5}$ and season such that there is a negative correlation in summer and autumn and positive correlation in spring and winter [28]. The plot for $MONTH$ also confirms this observation, such that future $PM_{2.5}$ values are predicted to be the highest in January, followed by November and September.

The plot for Wind Direction (WD) shows that future $PM_{2.5}$ values are predicted to be the lowest when there is a Eastern wind, and future $PM_{2.5}$ values are predicted to be the highest when the wind changes from a northerly to southerly wind (*i.e.,* normalised value at around 0.5 on the x-axis). When the wind direction changes from Southwestern wind to West-southwestern wind, it is also likely leads to a lower future $PM_{2.5}$ value. Lastly, the plot for $YEAR$ shows that more recent data (*i.e.,* data from 2016 onward) have more impact on the model in predicting higher future $PM_{2.5}$ values. Whereas there is a drastic decrease in contribution of $YEAR$ towards predicting future $PM_{2.5}$ values in the years between 2013 and 2014.

B.2 Explanations for the Energy Data Set

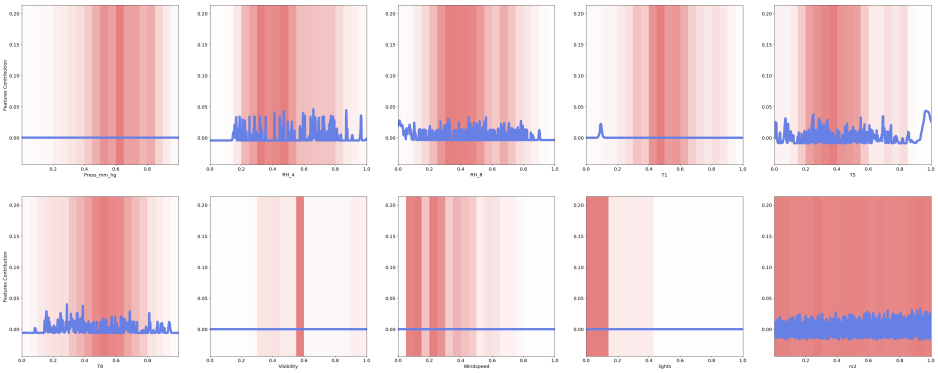


Figure B.2: Graphs learned by FocusLearn for Energy.
Predicting future appliance energy values (regression) on the *Energy* data set. These plots show top 10 features selected by the AFS component in the FocusLearn, where selected features with normalised and original values are on the x-axis, and future energy values prediction contribution are on the y-axis.

Figure B.2 presents the shape functions that show the exact depiction of each feature contribution to the model predicting energy uses in the house. Notably, pressure measured in $mmHg$ (*pressure_mm_hg*), *Viability*, and *Wind speed* measured at the Chiveres weather station, as well as energy consumption of light fixtures (*light*) within the household, exhibit negligible contribution to the model prediction, as evidenced by their flat blue lines indicating a constant shape function with a contribution equal to zero. Features such as humidity in the office (*RH4*), humidity in bedroom 2 (*RH8*), temperature in the bathroom (*T5*), temperature in bedroom 2 (*T8*), and temperature in the living room (*T2*) demonstrate fluctuating contributions above zero, albeit without significant values that would markedly influence the model to predict higher appliance energy usage.

From these observations, it can be inferred that data sourced from the Chiveres weather station, encompassing pressure, viability, wind speed, and dew point, exert minimal impact on household appliance energy consumption. Conversely, variations in humidity and temperature across different rooms within the household significantly influence appliance energy usage patterns.

B.3 Explanations for the EEG Data Set

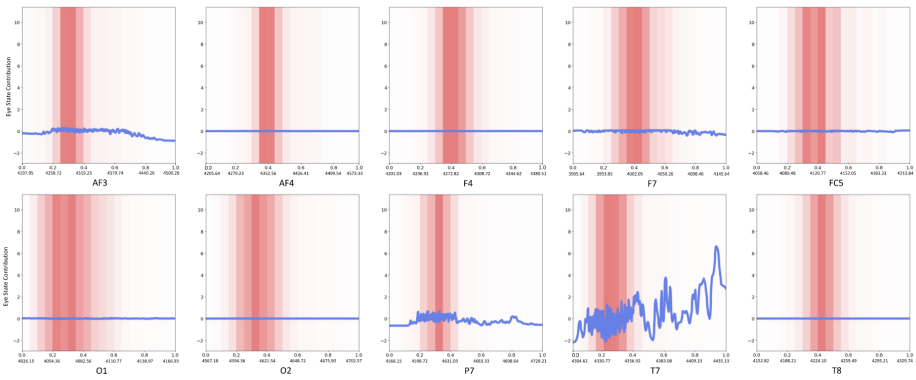


Figure B.3: Graphs learned by FocusLearn for EEG.
Predicting eye state (classification) on the *EEG* data set. These plots show top 10 features selected by the AFS component in the FocusLearn, where selected features with normalised and original values are on the x-axis, and contribution towards the prediction are on the y-axis.

Figure B.3 shows that electrodes placed at the *AF4*, *F4*, *F7*, *FC5*, *O1*, *O2*, and *T8* regions have almost no contribution towards FocusLearn in predicting eye state from the EEG, and *AF3* and *P7* have minimal contribution towards the model prediction. This means that electrodes placed at these regions do not contribute much to FocusLearn in predicting the eye state.

Figure B.3 shows that the electrode placed on the *T7* region is the most important one in predicting the eye state. Its plot shows that higher *T7* values means it is more likely that the eyes are open (*i.e.,* label 1), and lower *T7* values means it is more likely that the eyes are closed (*i.e.,* label 0). Interestingly, *T7* is a feature that is neither the most correlated to the outcome nor the one with the highest multicollinearity. Meaning that FocusLearn managed to capture the non-linear relationship in the underlying data.

B.4 Explanations for the Weather Data Set

The shape function plot for Humidity at 3pm (*Humidity3pm*) shows that humidity from 60 to 99% at 3pm today will likely lead to *rain* tomorrow. The start of the upward trend of the contribution of humidity occurs in a region of high data

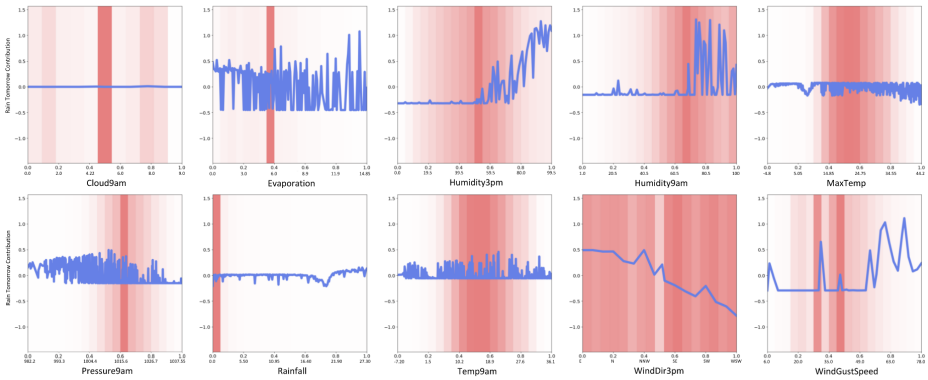


Figure B.4: Graphs learned by FocusLearn for Weather.
Predicting whether it will rain tomorrow (classification) on the *Weather* data set. These plots show top 10 features selected by the AFS component in the FocusLearn, where selected features with normalised and original values are on the x-axis, and contribution towards the prediction are on the y-axis.

density. The plot for Wind Direction at 3pm (*WindDir3pm*) shows higher chances of rain when there is a northerly wind compared with lower chances of rain when the wind direction changes to a more southerly wind.

The plot for the Fraction of sky obscured by cloud (oktas) at 9am (*Cloud9am*) shows that this feature contributed nothing in FocusLearn predicting whether or not it will rain tomorrow.

The plot for Class A pan evaporation (mm) in the 24 hours to 9am *Evaporation* shows that the relationship between Evaporation and future rain fall is rather complicated. This is as expected as although both evaporation and precipitation (rainfall) are connected within the water cycle, their relationship depends on several factors including global climate phenomena and local factors. Nevertheless, this shape plot would be helpful to meteorologists to discover the detailed pattern between *Evaporation* and *Rain*. For example, in this specific case, when *Evaporation* reaches 13mm, then this will most likely lead to rain tomorrow. This complicated relationship can also be observed with Atmospheric pressure (hpa) reduced to mean sea level at 9am (*Pressure9am*) and Temperature (degrees Celsius) at 9am (*Temp9am*), and how atmospheric pressure and temperature affects the chance of raining the next day also depends on other factors.

The plot for Humidity at 9am (*Humidity9am*) follows a similar pattern to

Humidity3pm, with a drastic increase in the chance of raining tomorrow when the humidity is above 60%. Therefore, when humidity at 9am is above 60%, it will most likely to rain tomorrow, and when humidity at 9am is less than 60%, it will most likely to not rain tomorrow.

The plot for Maximum Temperature in degrees Celsius (*MaxTemp*) shows that higher maximum temperature means there is a less chance of raining tomorrow, and there is a drop in chance of raining tomorrow when the maximum temperature is above 5 degree Celsius.

The plot for The amount of rainfall recorded for the day in mm (*Rainfall*) shows that, as expected, the more rainfall detected on the recorded day, more likely to rain tomorrow. There is a drop in chance of raining tomorrow when rainfall detected on the recorded day is above 17mm, and this might warrant a further investigation with the meteorologist.

The plot for The speed (km/h) of the strongest wind gust in the 24 hours to midnight (*WindGustSpeed*) shows that when the speed of the strongest wind is at around 70km/h, it is more likely to rain tomorrow. The chance of raining tomorrow then drop when the speed of the strongest wind is above 70km/h. Whereas when the speed of the strongest wind is less than 49 km/h, it is more likely to not rain tomorrow.

Appendix C

Best Hyper-parameters for the Prediction Models

C.1 On Time Series Data

Table C.1: Optimal Hyper-Parameters for the four time series data sets.

	Hyper-parameters	Air	Energy	EGG	Weather
FocusLearn	Initial Learning Rate	0.0008	0.0058	0.0021	0.0003
	Dropout rate in the RNN	0.0349	0.3584	$8.6209e^5$	0.1992
	Dropout rate in the MNN	0.0142	0.0140	0.0005	0.8378
	Dropout rate in the final output	0.0513	0.0579	0.0011	0.1851
	Batch Size	64	512	512	64
	Number of hidden unit in the RNN	64	64	64	1,026
	Number of hidden units in the MNN	[256, 128]	[256, 128]	[32, 16]	[512, 256]
NAM	Learning Rate	0.0122		0.0007	0.0002
	Dropout Rate	0.0405		0.0107	0.0218
	Batch Size	512		256	256
	Hidden Sizes	[512, 256]		[64, 32]	[256, 128]
SPAM	Learning Rate	0.1902		0.0009	
	Dropout Rate	0.0265		0.0590	
	Batch Size	256		64	
LSTM	Learning Rate	0.0082		0.7330	0.0369
	Dropout rate	0.0567		0.2340	0.1285
	Batch Size	64		128	64
	Number of hidden unit	128		128	256
XGBoost	Number of estimators	200		50	1,000
	Max Depths	50		5	5
	Subsample	0.8069		0.3804	0.3191
	Learning Rate	0.7600		$2.8467e^5$	0.4725

C.2 On Healthcare Domain Use Case

Table C.2: Optimal Hyper-Parameters for the Hearing Loss domain use case.

	Hyper-parameters	OtiReal
FocusLearn	Initial Learning Rate	0.0130
	Dropout rate in the RNN	0.0318
	Dropout rate in the MNN	0.1450
	Dropout rate in the final output	0.0339
	Batch Size	256
	Number of hidden unit in the RNN	256
	Number of hidden units in the MNN	[256, 128]
NAM	Learning Rate	0.0065
	Dropout Rate	0.4875
	Batch Size	64
	Hidden Sizes	[128, 64]
SPAM	Learning Rate	0.268
	Dropout Rate	0.1072
	Batch Size	64
LSTM	Learning Rate	0.0260
	Dropout rate	0.1153
	Batch Size	256
	Number of hidden unit	64
XGBoost	Number of estimators	1,000
	Max Depths	100
	Subsample	0.2605
	Learning Rate	0.3370

Appendix D

Time Series Data Analysis

D.1 Energy Time Series Data Analysis

D.1.1 Energy Time Series Data Decomposition

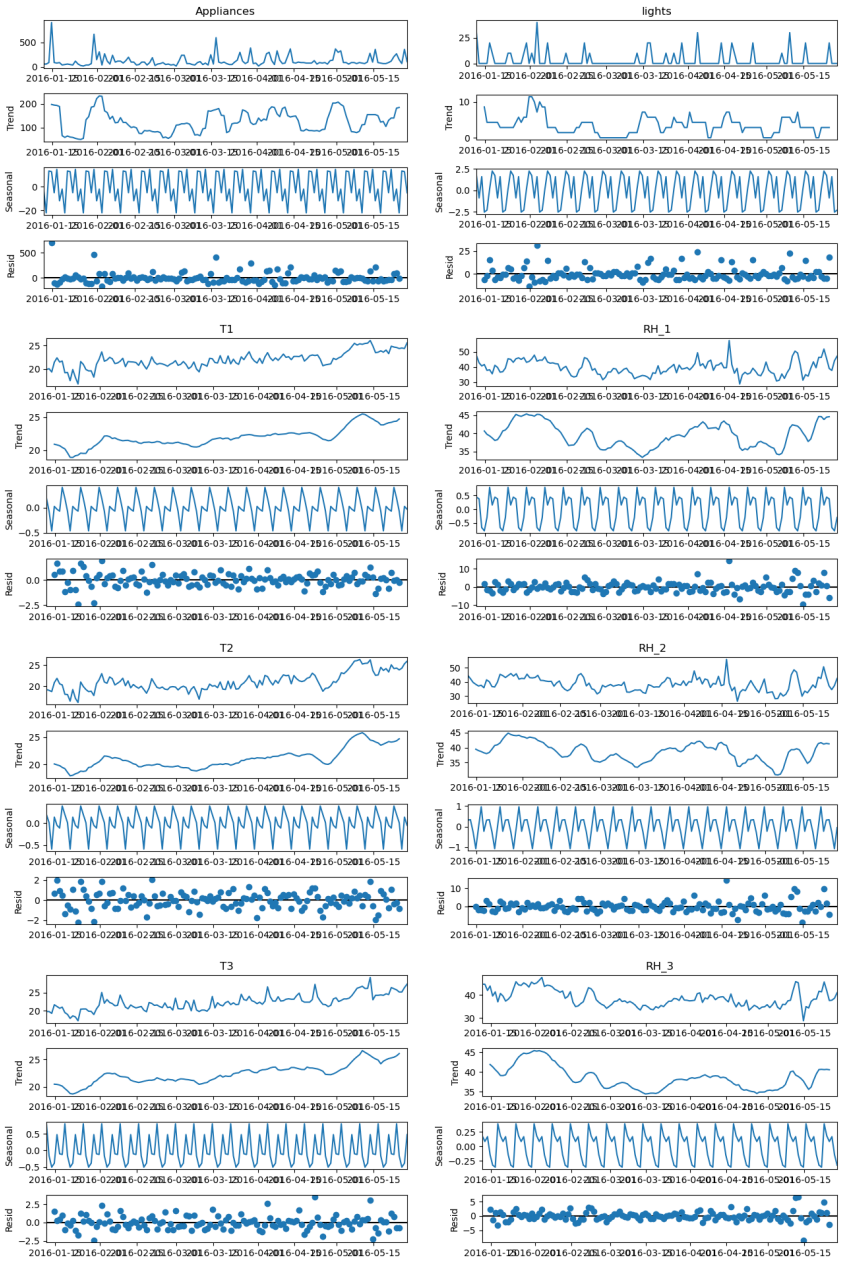


Figure D.1: Energy Time Series Data Decomposition Graphs I

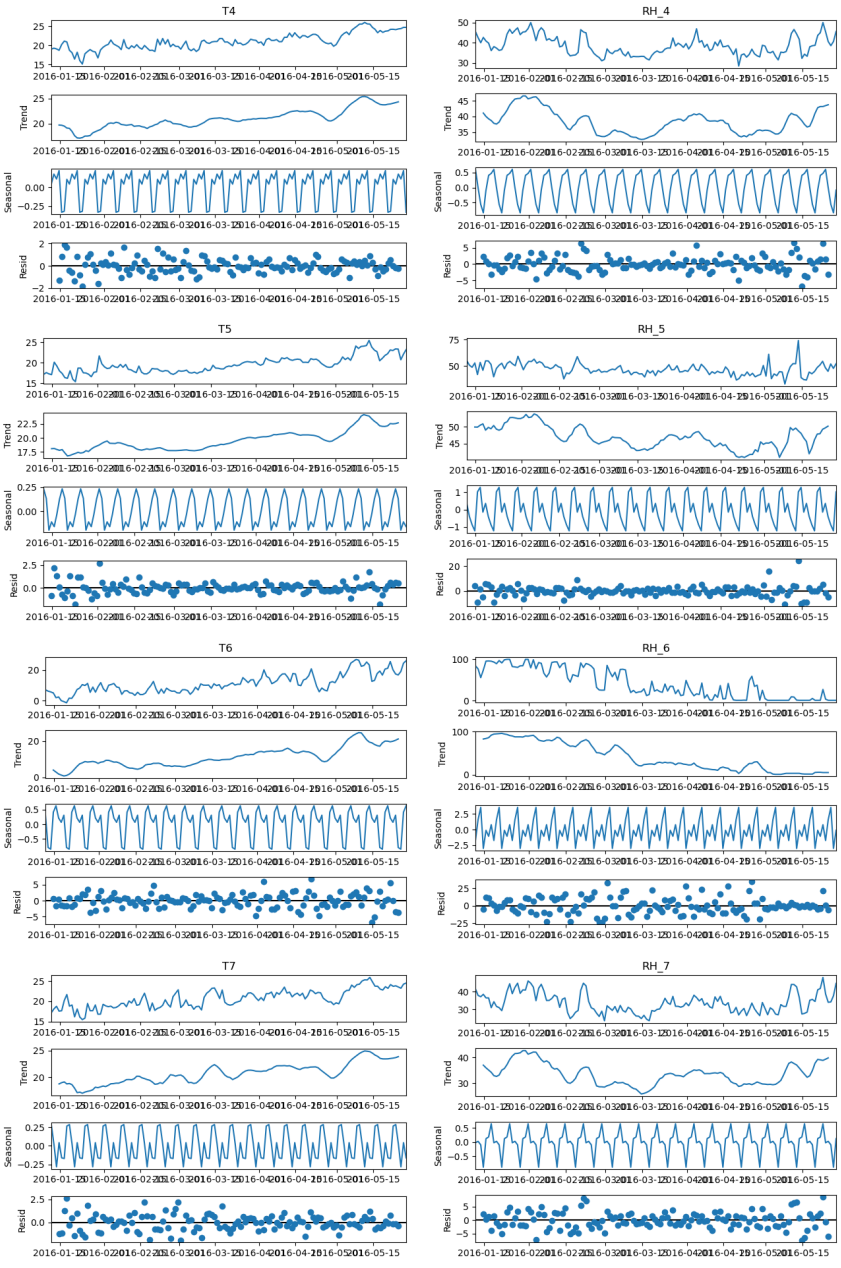


Figure D.2: Energy Time Series Data Decomposition Graphs II

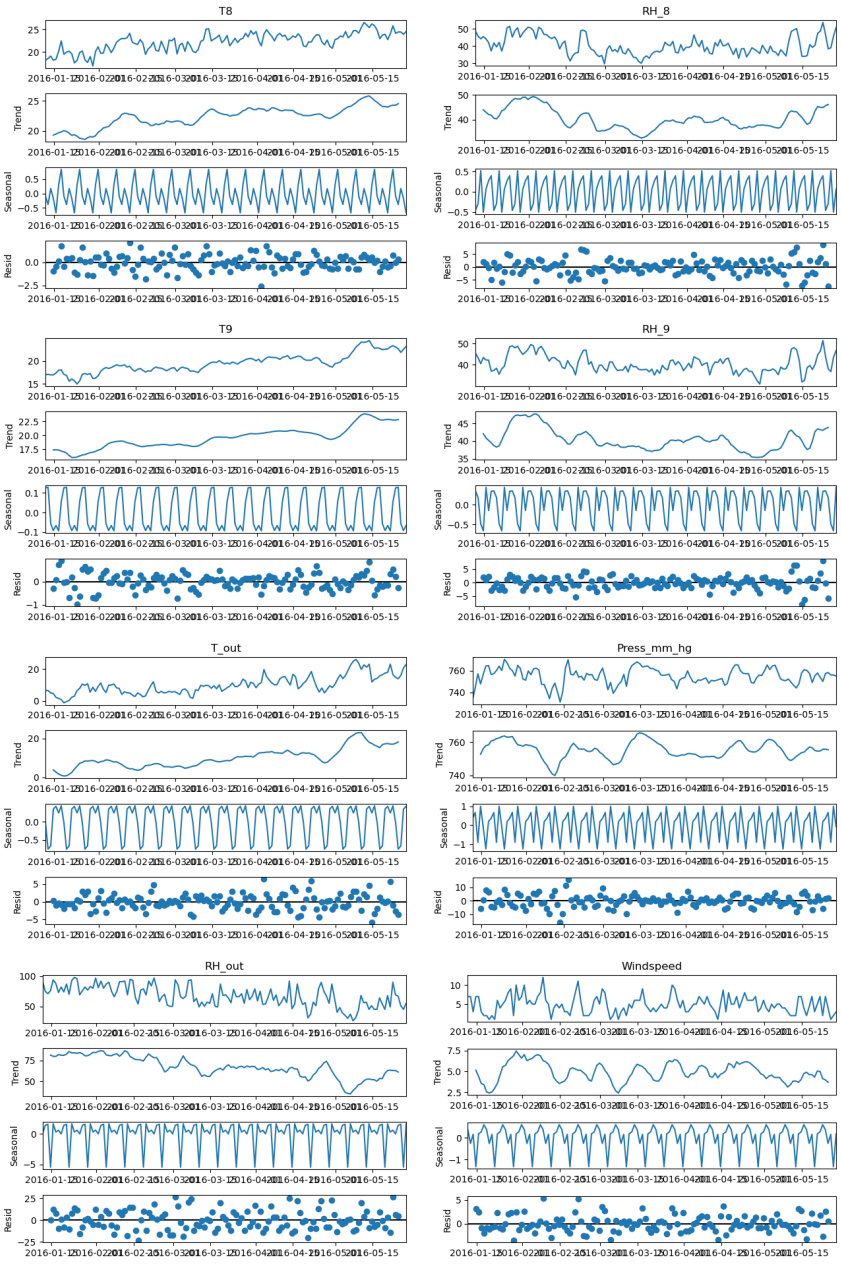


Figure D.3: Energy Time Series Data Decomposition Graphs III

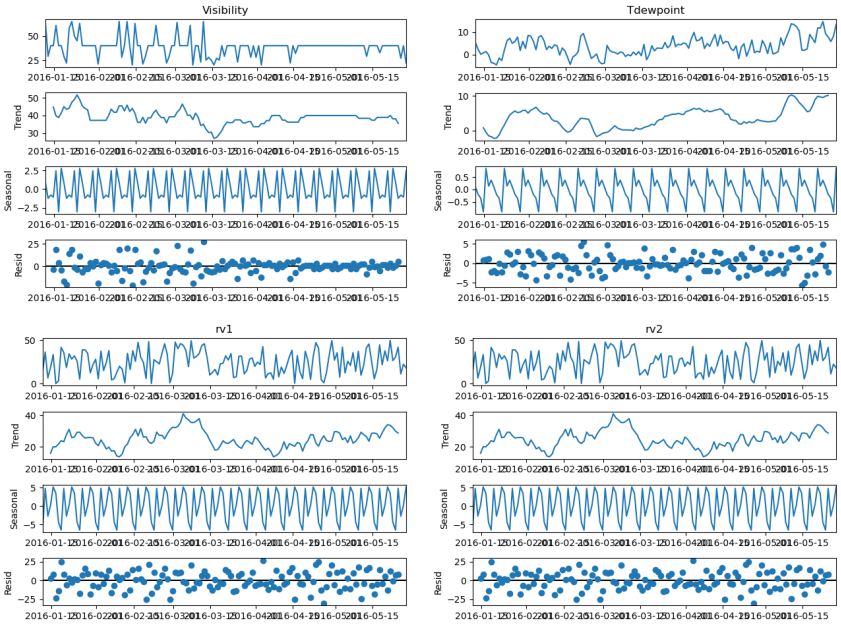


Figure D.4: Energy Time Series Data Decomposition Graphs IV

D.1.2 Energy Time Series Data Rolling Standard Deviation

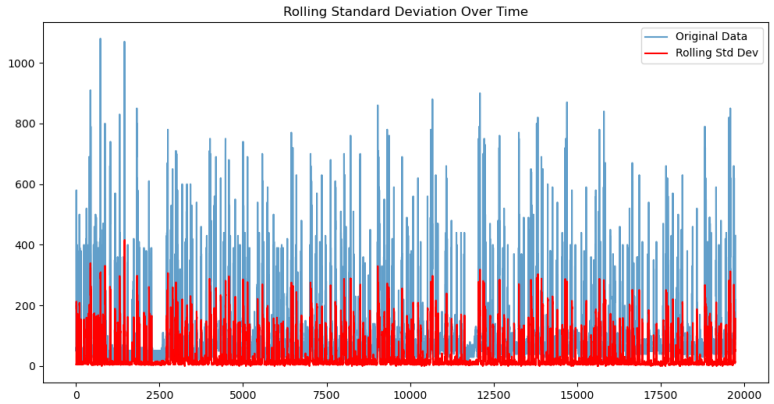


Figure D.5: Energy Time Series Data Rolling Standard Deviation over Time

D.2 Air Time Series Data Analysis

D.2.1 Air Time Series Data Decomposition

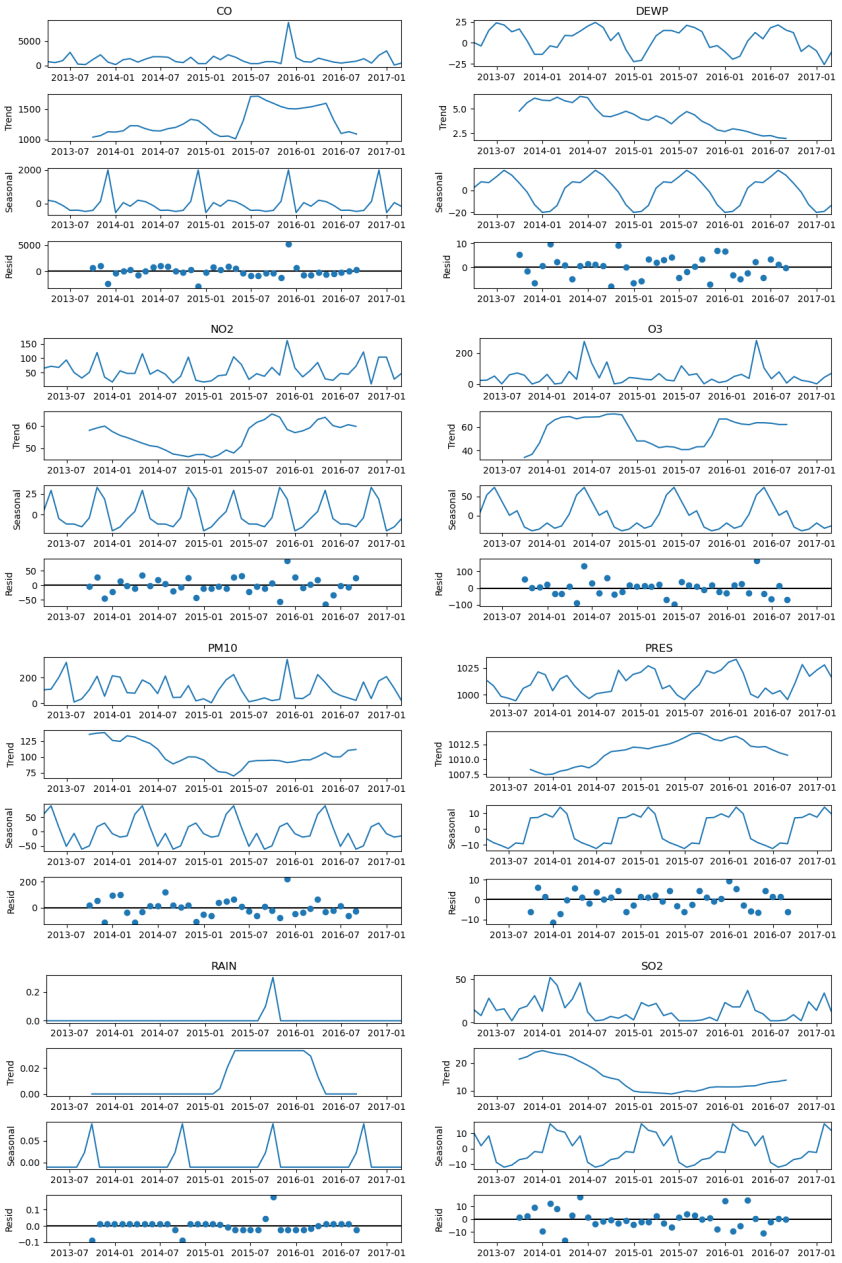


Figure D.6: Air Time Series Data Decomposition Graphs I

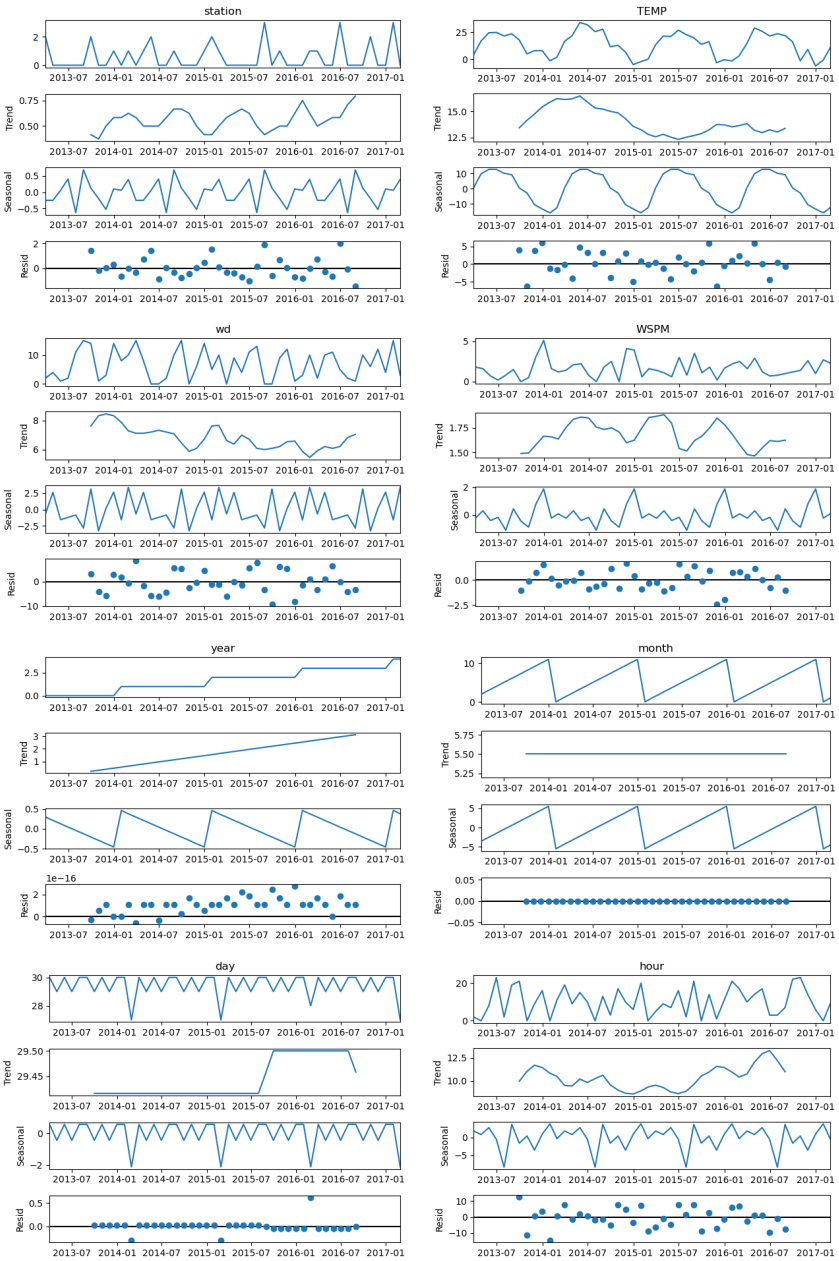


Figure D.7: Air Time Series Data Decomposition Graphs II

D.2.2 Air Time Series Data Rolling Standard Deviation

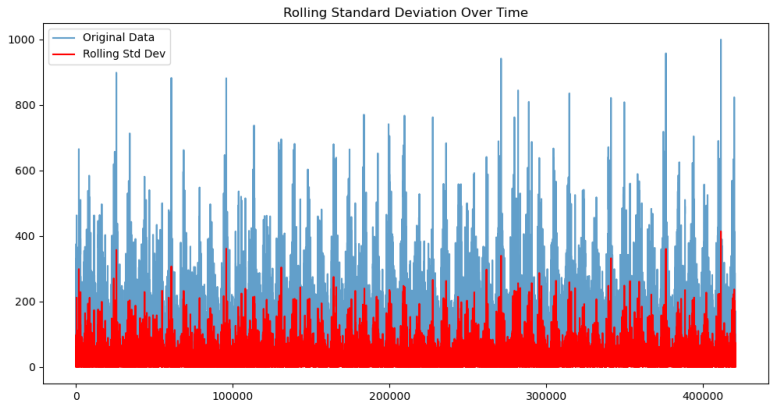


Figure D.8: Air Time Series Data Rolling Standard Deviation over Time

D.3 EEG Time Series Data Analysis

D.3.1 EEG Time Series Data Correlation

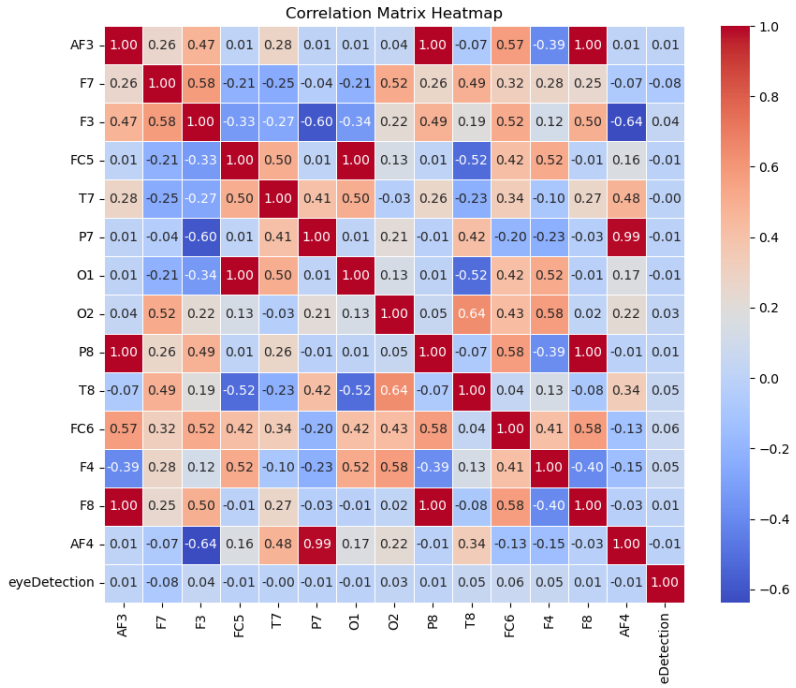


Figure D.9: EEG Time Series Data Correlation Graph