RENAN DE LUCA AVILA

Time series forecasting via machine learning using signal processing and exogenous features

São Paulo
2025

RENAN DE LUCA AVILA

Time series forecasting via machine learning using signal processing and exogenous features

Revised version

Dissertation presented to Escola Politécnica da Universidade de São Paulo to obtain Master's degree in Computer Engineering.

Concentration area:

Computer Engineering

Supervisor:

Doctor Glauber de Bona

São Paulo
2025

Catalogação-na-publicação

Avila, Renan
Time series forecasting via machine learning using signal processing and exogenous features / R. Avila -- São Paulo, 2025.
151 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.APRENDIZAGEM PROFUNDA 2.PREVISÃO (ANÁLISE DE SÉRIES TEMPORAIS) I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

ACKNOWLEDGMENTS

I would like to thank my Father, Homero Farias Avila, who very closely oriented, followed, and supported me throughout my entire life, playing a key role in my academic path. I thank my mother, Alexandra Martins Buzatto, for the fundamental and everlasting care, especially during my early childhood. I thank my Professor Glauber de Bona for the excellent conduct of my supervision, who, in addition to providing motivation, knowledge, and technical orientation to carry out this research, also supported and bore with me until the end. I also thank each Professor at this and other Schools I have been through, who helped me to build, in greater or lesser degree, foundations upon which I think today, which allowed me to have autonomy to build my own ideas, among them: Professor Dione Mari Morita, Professor Fernando Akira Kurokawa, Professor Rodrigo Leite, Professor Renzo Campano, Professor Simone Motta, Professor Daniel Medeiros, Professor Edmilson Motta, Professor Antônio Pedrine and Professor Flávio Cipparrone.

ABSTRACT

AVILA, R. d. L. Time series forecasting via machine learning using signal processing and exogenous features. 2025. Dissertation (Master's) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2025.

Time series forecasting has long been an important topic of research, predating the advent of machine learning. Early statistical models focused on predicting the target series based solely on its own past. More recently, it has been shown that taking multiple variables related to the context of the target as inputs to the model, and extracting new features from them—for instance, with signal decomposition and dimensionality reduction techniques—can improve prediction accuracy. However, as more data are added to the model, it becomes more difficult for training to converge to a global optimum, and as the data are more aggressively reduced, the likelihood of losing valuable information increases.

To address this trade-off, this work investigates how a single-input state-of-the-art model that leverages time series decomposition for feature extraction (CEEMDAN-LSTM) can make use of exogenous features to improve its accuracy while balancing training complexity issues. The investigation used a total of four data sets from different domains and led to two main results: the proposal of a new architecture, X-CEEMDAN-LSTM, which includes exogenous features as inputs and outperforms its single-input version; and the evaluation of different relevance metrics, with two proposed scores, in order to select the most significant exogenous features for the input, reducing the dimensionality of the data set while preserving the best results for the X-CEEMDAN-LSTM model.

Keywords – LSTM, CEEMDAN, neural networks, time series, forecasting, exogenous features, feature selection, feature extraction.

RESUMO

AVILA, R. d. L. **Previsão de séries temporais via aprendizado de máquina utilizando processamento de sinais e variáveis exógenas**. 2025. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2025.

A previsão de séries temporais é um tema de pesquisa de grande relevância, anterior ao surgimento do aprendizado de máquina. Modelos estatísticos tradicionais concentram-se na previsão da série-alvo exclusivamente a partir de seus próprios valores passados. Nos últimos anos, entretanto, demonstrou-se que a inclusão de múltiplas variáveis relacionadas ao contexto da série-alvo como entradas do modelo, bem como a extração de novos atributos a partir delas — por exemplo, por meio de técnicas de decomposição de sinais e redução de dimensionalidade — pode melhorar a acurácia preditiva. Contudo, à medida que mais dados são adicionados ao modelo, torna-se mais difícil que o treinamento convirja para um ótimo global; por outro lado, quanto mais agressiva for a redução dos dados, maior o risco de perda de informações relevantes.

Para lidar com esse *trade-off*, este trabalho investiga como um modelo de entrada única de última geração que utiliza decomposição de séries temporais para extração de atributos (CEEMDAN-LSTM) pode incorporar variáveis exógenas para melhorar sua acurácia, equilibrando os desafios de complexidade do treinamento. A investigação utilizou quatro conjuntos de dados de diferentes domínios e resultou em duas contribuições principais: a proposta de uma nova arquitetura, X-CEEMDAN-LSTM, que inclui variáveis exógenas como entradas e supera sua versão de entrada única; e a avaliação de diferentes métricas de relevância — incluindo duas propostas neste trabalho — para selecionar as variáveis exógenas mais significativas, reduzindo a dimensionalidade do conjunto de dados sem comprometer o desempenho do modelo X-CEEMDAN-LSTM.

Palavras-chave – LSTM, CEEMDAN, redes neurais, séries temporais, previsão, variáveis exógenas, seleção de atributos, extração de atributos.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

LSTM - Long short term memory

CEEMDAN - Complete ensemble empirical mode decomposition with adaptive noise

IMF - Intrinsic mode function

PCA - Principal component analysis

ANN - Artificial Neural Network

RNN - Recurrent Neural Network

BOVESPA - Bolsa de Valores de São Paulo

API - Application Programming Interface

OHLC - Open, High, Low, Close [,Volume]

CONTENTS

PART I

INTRODUCTION

1 PROBLEM AND MOTIVATION

Prediction is one of the main drivers of human knowledge. Science itself may be understood as a consequence of humanity's efforts to anticipate the future. Physics, for example, seeks to explain how nature works by experimenting and observing past phenomena; since nature appears not to change over time, the same principles can be applied to understand how the future will behave. Mathematics, similarly, not only helps explain natural phenomena but also predicts abstract and complex patterns that may not even exist in nature.

Engineering, in turn, is traditionally concerned with applying established knowledge to new contexts, generating additional knowledge as a result—ultimately creating new futures for humankind. The humanities and social sciences also rely on prediction: economics models how markets behave, psychology studies how individuals act, and political science analyzes how societies evolve. Although each field focuses on different aspects of human experience, they share a common objective: to understand the present well enough to anticipate what will come next.

In this context, since the future is uncertain, it is natural to expect a field of science dedicated exclusively to studying uncertainty: Statistics, a discipline of immense value.

Unlike the previously mentioned fields, electronic computing is a comparatively recent area of study that brings together Mathematics, Engineering, Physics, and Statistics. Although it draws from each of these areas and addresses problems that rely on all of them, computing differs in that it has become one of the most powerful forces in shaping the world and influencing human behavior within short periods of time.

Consequently, electronic computing is intrinsically related to predicting the future. Every prediction concerns an observed phenomenon, and its observation over time defines a time series. The predictability of a given phenomenon may be understood in terms of the amount of entropy involved. For example, predicting phenomena governed directly by a small set of well-understood physical or chemical laws—such as the motion of an

accelerating body or a chemical equilibrium reaction—is, at first sight, easier than predicting the chaotic motion of a double pendulum or the behavior of a quantum particle. The latter depend on far more variables and conditions than the former.

Entropy and uncertainty reduce predictability. This is where skepticism about using past observations to predict the future becomes meaningful (Hume, 2007), since we may not be accounting for all relevant variables, including the possibility that the laws of nature themselves may change over time.

Acknowledging the inherent uncertainty of the future, and faced with the complexity of the universe's phenomena, this work focuses on human-scale circumstances in time and space. Its aim is to modestly advance the frontiers of our limited knowledge in forecasting worldly phenomena close to human reality, based on past observations, thereby producing useful information for anticipating the near future so that humankind may plan its path, even though determining such a future is not up to us.

1.1 Financial Domain

An example of complex phenomena are the human interactions, which happen in multiple layers and make human behaviors and its consequences very hard to predict, such as opinions and social behaviors.

So, the future is also shaped by human relationships itself: Politics, ethics, and philosophy deeply impact the future of humankind. One of the great impacts is related to the management of productivity sources that power humankind's evolution, the economy.

The economy is a complex organization system that distributes resources and productivity among humankind, since resources and productivity are limited and the demand for them is variable over future time, then naturally there came up the famous system of markets where there is offer and demand for products and services. Today, one of the forms of this system is the stock market.

1.1.1 Motivation

Predicting stock market prices has been a matter of study for a period as long as the stock market itself and will always be. The economy changes as the technology evolves and also has deep relations to how the humankind politics globally and locally cycles. So does the stock market, as a consequence of all these factors. That is the main reason

why this subject of predicting the stock market has kept and will keep itself among the top most studied problems throughout all time, as long as there is offer and demand for anything that is hard to tell its value.

Value means more than just how much something costs; it actually means how much something costs in the context of someone's interest. For example, a hammer may have no value for a painter but a great value for a blacksmith. In other words, anything that may be owned by someone may have different value depending on how one plans to use it and for what purpose.

This applies to the stock market and explains why stocks are hard to tell their values, keeping in mind the efficient market hypothesis, which has a special comment in Appendix B. Although one can check the price of any stock on any exchange at anytime, the price does not match the value. The value is mostly related to the future, rather than the price itself. The price at a moment is a consequence of the past. That means, for argument's sake, how a company or country used its resources to change the world, providing a new service, technology or product, which in turn depends on how the people involved made decisions and how all the other companies or countries reacted to these decisions.

At this point, the importance of the external context arises over the internal interests. The surroundings of a company or country in the evolution of humankind have a great impact on its own decisions. The context is exogenous, and all this exogenous information that explains the value of something or even the complexity of a chaotic natural phenomenon must be known to relate it to a previous behavior.

Although the financial domain is the first motivation of this work, every time series has its own peculiarities, and a good model should be domain-agnostic, thus the work takes into account data from other domains. The history of the financial domain is detailed in the next section.

1.1.2 History

Financial time series forecasting is a study subject that has been developed and researched mostly since the beginning of the 19th century, and is still a hot topic when it comes to researching nowadays. This study field originated from the statistical and stochastic processes analysis, Yule (1927) proposed the autoregressive models which is based on autocorrelations within a certain signal, then it was enhanced in 1951 by adding a moving average component (Whittle, 1951), creating the autoregressive moving aver-

age model (ARMA). The next great contribution to the field is Holt's linear exponential smoothing model (Holt, 2004), next to it comes Box and Jenkins (1976) with autoregressive integrated moving average model (ARIMA).

Box and Jenkins have made great contributions not only with model development but also with books that popularized all the previous work along with their own innovations. And this was a mark in the study field. After that, time series forecasting became even more researched and great advances came.

Among these advances, there was the study of heteroscedasticity by Engle (1982), who created the ARCH (autoregressive conditional heteroscedasticity) category of models in 1982, worth a Nobel prize. Another Nobel prize in the field was awarded to Granger, who has published work with Engle on studies after causality and cointegration theories (Engle and Granger, 1987). These theories started to study correlation with exogenous time series, investigating how the information of certain phenomena described by a time series can influence other phenomena.

Meanwhile Engle and Granger were at the apex of their research fields, Werbos was researching the subject of neural networks, a "dead" subject in computer science, at least for that moment. It was given as dead by Minsky and Papert (1969) when they pointed out the limitations of neural networks that used the neuron (called perceptron) proposed by Rosenblatt (1958). The limitations were the linearity and difficulty of the optimization algorithms for multilayer networks. These limitations were surpassed by Werbos (1990) in his work, although he had already researched them in his thesis (Werbos and John, 1974), he proposed the backpropagation optimization algorithm to find the right weights in a multilayer neural network. With the backpropagation algorithm, neural networks were then back "alive". This algorithm is still today the most used algorithm for training neural networks.

Since then, neural networks have increasingly gained interest. The first well-known network architecture is called the multilayer perceptron (MLP), which is a chain of neurons. After that a great field of study around the different possible architectures sparked in the scientific community. As a result, several well-known applications of neural networks arose, and as a specific part of those applications, in which nonlinear sequential behavior is a great concern, the time series prediction became a great research target. Due to the intrinsic sequential characteristic of time series, recurrent neural networks emerged as a natural solution, since recurrence embeds sequential characteristics. Although sequences are addressed, MLPs and simple RNNs that undergo several iteration steps may change

the behavior of neurons depending on the slice of data being optimized; thus only part of the data is learned and this part comes from the last iteration steps, which means that the most recent data in the learning process is better learned when compared to the old data (first iteration steps). That is a memory problem.

Aiming to solve this inequality in the optimization process, researchers started to develop neural network architectures with memory capacity, that implies an architecture with areas that are not always affected by all iteration steps. To choose rightly when to update certain neurons Hochreiter and Schmidhuber (1997) proposed an architecture with gates as filters in his Long-Short-Term Memory (LSTM) architecture for neural networks. Then LSTMs have been successfully applied to the field of time series prediction (Gers et al., 2001) with time window approaches, and also have shown great success in the field of natural language processing (NLP). Surprisingly, natural language has non-linearities and is sequential, making it a fertile field for LSTMs.

Throughout time, LSTMs have been combined with other prediction models to forecast time series, ranging from machine learning to statistical models, such as convolutional neural networks (CNN) (Huang et al., 2017), support vector machines (SVM) (Li et al., 2018) and ARIMA (Wang and Lou, 2019). In parallel to the hybridization of models and new architecture proposals, the preprocessing techniques for data handling were studied in depth to feed networks with information in an optimal structure or format.

The most well-known preprocessing techniques for time series have a common root idea of decomposition that relies on the Fourier transform, which for a given time series generates a set of other time series components representing different frequency behaviors; the original time series may be rebuilt using the components. Similar techniques include wavelet decomposition, principal component analysis (PCA), and more recently, empirical mode decomposition (EMD). The empirical mode decomposition was proposed by N.E. Huang et al (1998), different from the latter ones, the EMD works with the whole series length and not rolling windows, as well as uses a computationally simpler method to generate decompositions. The EMD technique stood as a solid approach and was further improved in the complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN), proposed by Torres et al. (2011).

The CEEMDAN technique and the LSTM networks are used in the work taken as a starting point for this investigation, based on Cao's CEEMDAN-LSTM (Cao et al., 2019). Although Cao's CEEMDAN-LSTM model overcame all baselines, it only deals with autocorrelation, which means the prediction model only uses as input the history

of the target series for prediction. It does not take into account relevant exogenous features. Today's market price fluctuations are influenced by several factors other than the actual historical performance itself. A sign for this is shown in Zhen's ACommNN model (Yang et al., 2021), which takes into account cross-regional series for index forecasting, in other words, modern models look at the market movements in a more global manner, with several exogenous features, although this last work lacks the use of decomposition technique.

1.2 Objective

We aim to develop a new prediction model that uses exogenous features to enhance state-of-the-art accuracy for one-step ahead target time series prediction.

Firstly, we will start by adapting the Jian Cao's CEEMDAN-LSTM model to handle time series with additional exogenous variables as input.

Secondly, we will evaluate relevance metrics for filter-based exogenous features to use to achieve optimal results.

1.3 Organization of this work

This work is the result of two **time** phases of research and development marked, respectively, by the first and second objectives.

The first phase tackles the problem of adapting a state-of-the-art model to handle multiple exogenous features as input. It is a consequence of the motivation around the financial field and of reviewing the literature in the area of time series forecasting. This led to the discovery of a paper that featured a preprocessing approach linked with a particular data flow architecture that utilizes traditional LSTM models, namely the CEEMDAN-LSTM. However, the literature shows that the incorporation of exogenous variables can enhance the performance of the model, and the exploration of adding exogenous time series to the CEEMDAN-LSTM model through architectural changes while preserving key aspects of signal processing plays an important role in this first phase.

The second phase tackles the problem of selecting the exogenous features to input. The primary objective of this stage is to identify a precise filter-based feature selection technique by assessing relevance metrics already solid in the literature to select the most beneficial exogenous features for each data set. Additionally, this phase aims to examine

the model's ability to enhance forecasting accuracy across diverse data sets in comparison to the standard CEEMDAN-LSTM model, which lacks exogenous features.

The findings and methods related to these phases are organized in the following **textual** main parts:

1. **First part**: the present introduction, aiming to give **context** around the problem and the **objective**; then a summary of related work, to highlight parallel state-of-the-art works that are being done in the same field of literature; subsequently, there is a section of base theory to bear in mind theoretical concepts that will be used in the method.

2. **Second part**: explains the **method** as well as the motivations to take each approach, detailing every step necessary to reproduce our experiments, and explaining what we expect to achieve in the results. It is subdivided into the previously mentioned phase 1 and phase 2.

3. **Third part**: shows the **results** both in the first phase of the work and in the second phase, how we executed the experiments, and how we interpret each of them and what it means in the context of science we are in, as well as what should be next. It is subdivided into the previously mentioned phase 1 and phase 2.

4. **Fourth part**: shows **discussions** about the results, a summary of the work and the pavings for future research.

2 RELATED WORK

The present research has Jian Cao's paper as a reference, but in this chapter we will present the main recent and successful models for time series prediction. All of them contain decomposition techniques and use neural models, including the use of exogenous or correlated time series.

2.1 Other works that use CEEMDAN and LSTM

Jian Cao published in 2019 a work entitled "Financial time series forecasting model based on CEEMDAN and LSTM" (Cao et al., 2019), which was taken as a starting point for this work and will hereinafter be called the CEEMDAN-LSTM model. The model is summarized in a data flow and a neural architectures.

Cao used an old idea from the signal processing field of decomposing the input signal into components, which may be seen as a feature extraction technique. The individual components then become the input data for the neural model as seen in Figure 1.

The process of decomposing the signal facilitates the model fitting during the training phase since the data tend to have a similar magnitude and frequency throughout its whole length.

The neural architecture used in Cao's work is a double-layered LSTM model, followed by a double dense layer with a custom regularization algorithm in the end, as seen in Figure 1.

The forecast data in Cao's work are index data for forecasting with the new proposed model, ranging from American to Asian indexes, which showed greater accuracy than compared baselines, such as SVM and ARIMA. The source of data taken was Yahoo Finance.

The same model as published by Jian Cao was used by Lin et al. (2021) to predict the same index prices as Jian Cao did, but this time Lin introduced the comparison to other

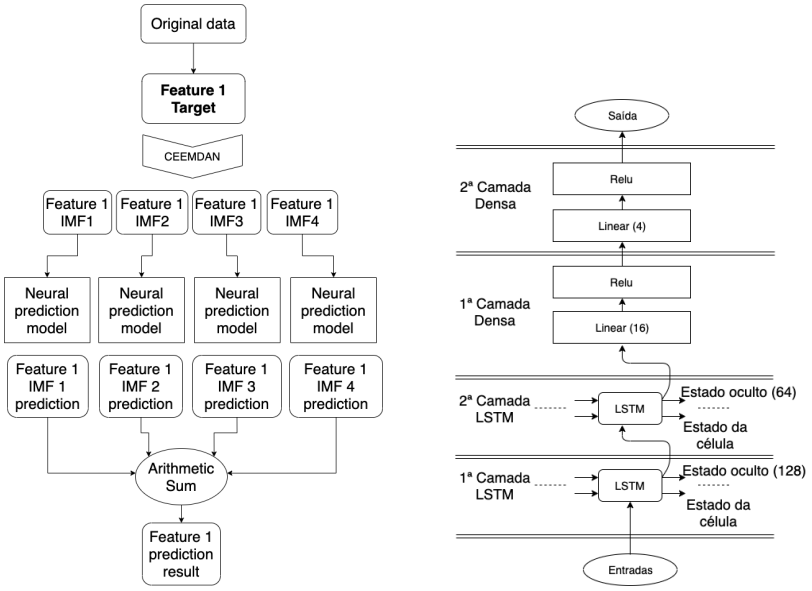


Figure 1: a) CEEMDAN-LSTM data flow proposed by Cao in 2019; b) CEEMDAN-LSTM neural architecture proposed by Cao in 2019

models such as SVM, Elman network, Wavelet Neural Networks (WAV), and also each of these models combined with their respective decomposition switched to CEEMDAN. Since each model is evaluated with a different loss function, in order to have a fair comparison, Lin used the MCS test (model confidence set) proposed by Hansen et al. (2011). Lin's results show that the CEEMDAN-LSTM model is still optimal.

The same author, Lin, more recently in 2022, also used the same model, but this time to predict the realized volatility of stock indices (Lin et al., 2022). High frequency Realized volatility is a highly noisy and nonlinear signal, often more noisy than the price itself, also volatility is a way to measure risk, which can be further combined with price forecasting in order to achieve better results with lower risk, but this last future work is not addressed by Lin. Lin used the architecture shown in Figure 2.

2.2 Other works that use decomposition and neural models

A particular approach taken by Emeksiz and Tan (2022) brings two different decomposition methods together with a BPNN model. They propose a hybrid use of Ensemble Empirical Mode Decomposition Adaptive Noise (CEEMDAN), Local Mean Decomposition (LMD), Hurst and Backpropagation Neural Network (BPNN). Emeksiz introduces the LMD to address the highest frequency IMF, since it has the most noisy behavior of all

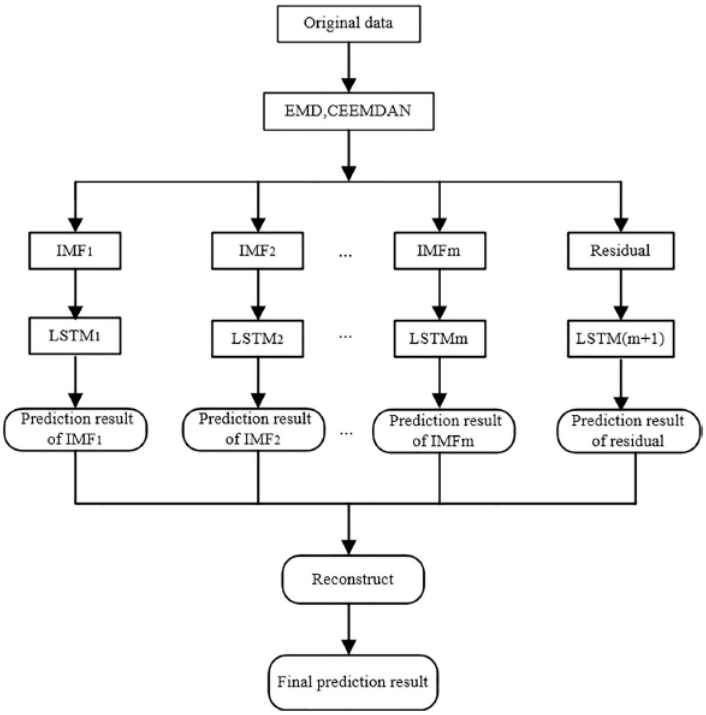


Figure 2: Architecture used by Lin, same as Cao. Source:(Lin et al., 2021)

IMFs, then the result of LMD undergoes Hurst analysis. Hurst analysis shows stochastic behavior of signals; the main objective is to predict the Hurst exponent (Hurst, 1951). Hurst exponent is related to the fractional Brownian motion (Mandelbrot and Van Ness, 1968). The Hurst analysis ends up separating a series into micro, meso and macro behaviors which are all fed into neural models as the other IMFs. The author uses wind speed data, and his architecture is shown in Figure 3.

Other spectral decomposition methods for time series, besides the EMD-based ones, are also widely used. Wavelet decomposition is commonly adopted in index prediction, Zhang et al. (2017) shows that forecasting a time series based on its components rather than the original series itself improves precision. The work forecasts financial index data using traditional ARIMA models.

Parvini et al. (2022) proposed another work that used wavelet decomposition to predict gold, oil, index, and crypto prices with the LSTM neural network. Figure 4. The author uses Discrete Wavelet Transform (DWT) and db3 (Daubechies) wavelet to remove noise and decompose. The authors chose to employ the method in a simulation to buy and sell these assets, and in the end the result showed that the model works better with S&P500.

There is also another decomposition method called *variational mode decomposition*

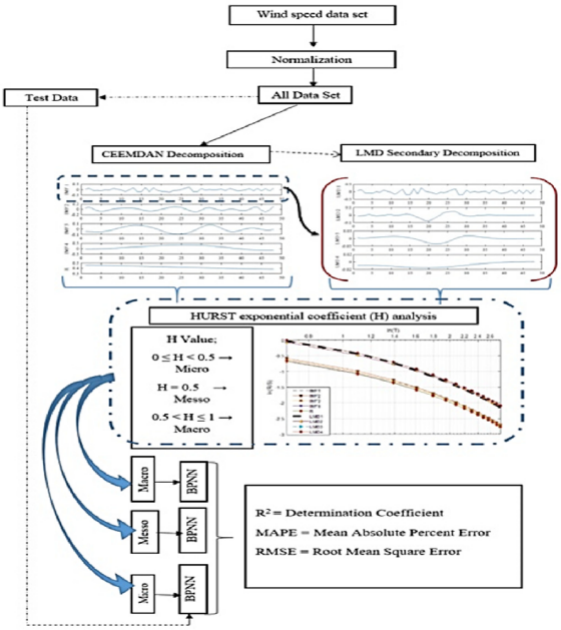


Figure 3: Emeksiz's proposed architecture. Source:(Emeksiz and Tan, 2022)

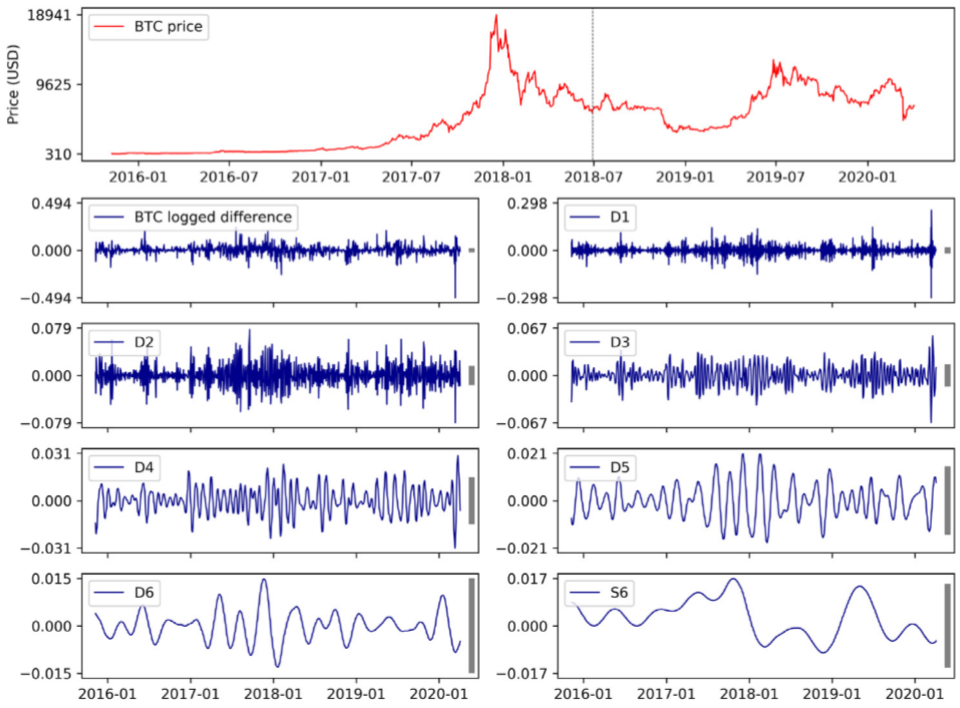
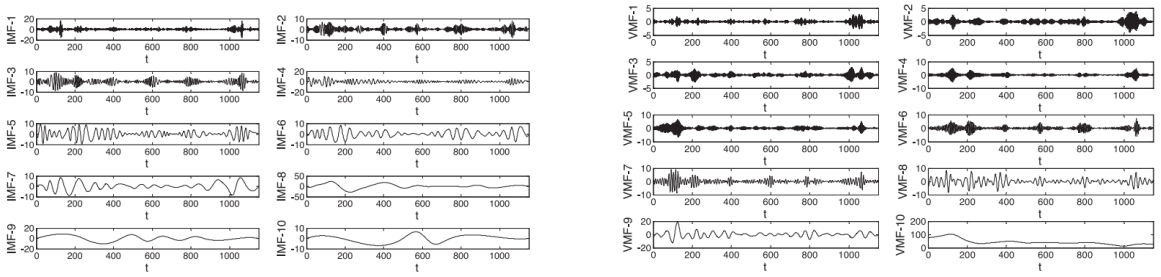


Figure 4: Parvinis's decomposition with wavelets. Source:(Parvini et al., 2022)

(VMD) which is compared to EMD in Giannakis (2019), and together with the particular generalized regression neural network model outperforms the EMD method. The work forecasts energy and oil prices, and the comparison between EMD and VMD for an example proposed by the author is shown in Figure 5.



((a)) ((b))

Figure 5: a) California electricity EMD decomposition; b) California electricity VMD decomposition (Giannakis, 2019)

2.3 Other works that use exogenous features

Jiawen researched an Infinite Hidden Markov model to forecast realized volatility of oil markets based on two references for oil price: the west Texas intermediate crude oil and the brent oil, which comes from northern Europe. This approach evaluates that adding an exogenous information, such as international oil price reference, to a local information-fed model provides an enhancement in prediction accuracy.

A hidden Markov model (HMM) problem occurs when one supposes that the system is a Markov process with unknown parameters, from which it is only possible to observe external variables, so the objective is to derive the Markov process parameters from the observed variable. A Markov process describes a system that changes its state solely based on its previous state, shown in Figure 6 as variable x, and also the observed variable y can only depend on the state of x at any given instant t.

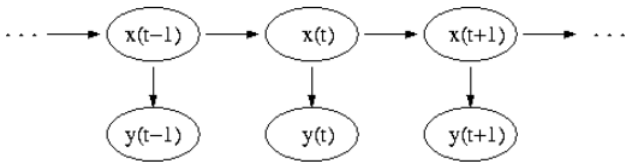


Figure 6: Hidden Markov Model Diagram. Source:Wiki

The Infinite Markov Model (IHMM) is an extension that makes it possible to not need to know the number of different states the system has *a priori* (Beal et al., 2001).

It turns out that with a guessed large enough amount of states and Dirichlet parameters guessing it is possible to tell the expected number of states and the starting state.

Jiawen then uses IHMM to fit a heterogeneous autoregressive model (HAR) (Corsi and Renò, 2012), which can be written as a general linear regression $y_t = \alpha x_t + u_t$, with u_t as a normal distribution. The exogenous input is used in the HAR model by extending the dependency of S and P and crude oil exogenous data. Jiawen's results show that crude oil exogeneity improves forecasting, but the same is not true for the S and P.

Exogenous data are also explored by Saha and Lopez (2020) to forecast energy prices in the next day. The author uses demand and temperature as exogenous data to predict the price of energy at each hour of the following day, because it is well known that energy is affected by temperature and demand. Before feeding the model, the time series data of the energy price undergo a natural logarithm transformation to reduce the peak presence. The architecture used uses an LSTM to learn endogenous past trends and a CNN to learn the exogenous trends independently; the outputs of both models are then subject to a stacking filter, also using convolution, which is shown in Figure 7.

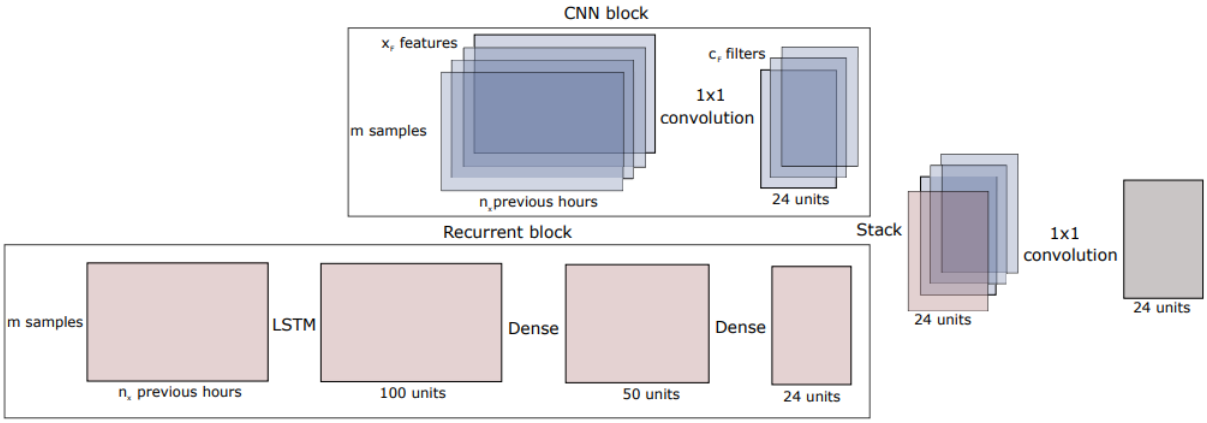


Figure 7: Saha's architecture, the upper block takes the exogenous features.

Closer to our aim in this work, Guo and Lin (2018) proposed a multivariate LSTM (MV-LSTM), which claims to tackle the problem of dealing with exogenous variables in the same LSTM. If an LSTM is fed with endogenous and exogenous data, all the information will be mixed up within the hidden state, based on that Tian proposes an LSTM architecture with separate hidden states for each input variable. A good concern in that work is the interpretability; Since each variable has its own hidden state, the influence of each variable in the final result can also be measured, a side output of MV-LSTM is the importance of each variable, generating weights.

In Figure 8, it is shown how the MV-LSTM works for a two-variable input, where H

is the representation of the hidden state.

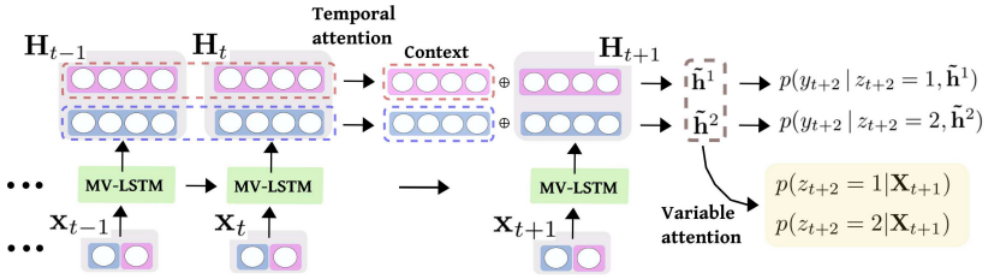


Figure 8: MV-LSTM Architecture.

The attention mechanism used here is based on a probabilistic mixture of different proposed authors, as in Equation 2.1, which takes into account conditional probabilities, where x stands for the input, y stands for the output, z stands for a newly introduced random variable, and N is the number of variables.

$$p\left(y_{T+1} \mid \mathbf{X}_T\right) = \sum_{n=1}^{N} p\left(y_{T+1}, z_{T+1} = n \mid \mathbf{X}_T\right) = \sum_{n=1}^{N} p\left(y_{T+1} \mid z_{T+1} = n, \mathbf{X}_T\right) p\left(z_{T+1} = n \mid \mathbf{X}_T\right)$$
$$= \underbrace{\sum_{n=1}^{N-1} p\left(y_{T+1} \mid z_{T+1} = n, \mathbf{X}_T^n\right) p\left(z_{T+1} = n \mid \mathbf{X}_T\right)}_{\text{Exogenous part}} + \underbrace{p\left(y_{T+1} \mid z_{T+1} = N, \mathbf{Y}_T\right) p\left(z_{T+1} = N \mid \mathbf{X}_T\right)}_{\text{Autoregressive part}}, \tag{2.1}$$

An important aspect of Tian's paper are the tests applied to the baseline data: Augmented Dickey-Fuller (AD-Fuller) and Kwiatkowski-Philips Schmidt Shin (KPSS) tests to determine the necessity of differentiating time series. The data sets used are on meteorological and energy subjects.

An important model that takes into account exogenous features and has been extensively researched is the nonlinear autoregressive network with exogenous inputs (NARX), which in fact is an adaptation of statistical NARX models to neural networks (Diaconescu, 2008). The NARX network is a common multilayer perceptron with a feedback loop connecting the output to the input as seen in Figure 9, although it is designed to be used with exogenous data, within time series the exogenous data are usually the tapped delay lines of the same time series.

Although NARX recurrent networks are mostly used with tapped delay lines, there are different initiatives that try to take into account real exogenous data, using other series rather than the target one, Agung (2022) used NARX with open, high, low and

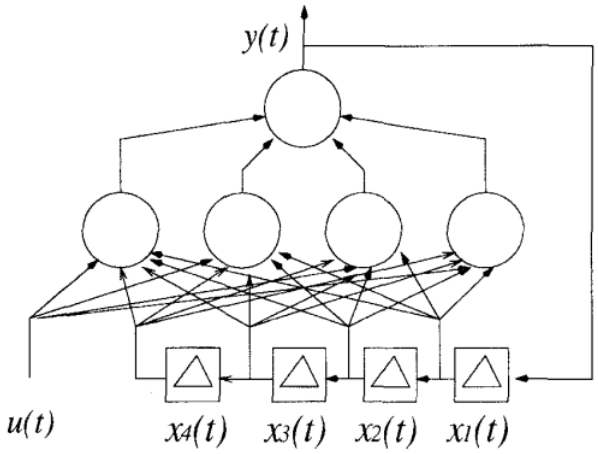


Figure 9: NARX Architecture with 4 tapped-delay lines input, (Lin et al., 1996)

close prices for stock index prediction and also used other technical indicators (moving average and momentum, for example). The architecture used in Wiseto's paper is the one in Figure 10.

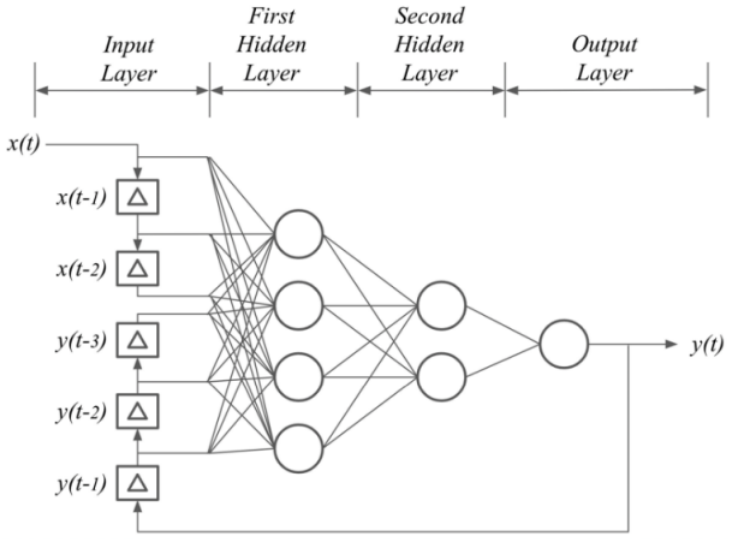


Figure 10: NARX NN with Two Hidden Layers (Ezzeldin & Hatata, 2018)

In Figure 10 the series y represents the target prediction series and the series x represents another exogenous input, such as the moving average. Since the architecture of NARX network is a multilayer perceptron, it is possible to include as many input parameters as desired.

When dealing with exogenous features one should always ask whether the so thought exogenous features are indeed exogenous or not, or how good the extra information in an exogenous features is. In several cases, features behaviors overlap and thus do not bring new and efficient information for predicting the target feature. Similarity metrics are the

Dynamic Time Warping Methods	**Full Name**
DTW	Dynamic Time Warping
DDTW	Derivative DTW
IDTW	Incremental DTW
WDTW	Weighted DTW
Edit Distance Methods	**Full Name**
EDR	Edit Distance on Real Sequence
ERP	Edit Distance with Real Penalty
LCSS	Longest Common Subsequence
TWED	Time Warping Edit Distance
Other methods	**Full Name**
ALoT	Alignment of Textures
AMSS	Angular Metric for Shape Similarity
FSTE	Fast Time Series Evaluation
MSM	Move-Split-Merge
MVM	Minimal Variance Matching
SpADe	Spatial Assembling Distance
Swale	Sequence Weighted Alignment Model

Table 1: Main elastic similarity measures for exogenous features (Christen et al., 2022).

state-of-the-art method, the higher the similarity, the better the exogenous features.

Some works study how similarity metrics can be better used to select exogenous features, in the field of time series analysis, Christen et al. (2022) summarizes in Table 1 the main similarity metrics, such as distance and dynamic comparison, to evaluate exogenous feature quality, presenting requirements for a robust significance analysis method around exogenous feature selection.

Christen also differs two types of metrics, the elastic ones and the lock-step ones. Lock-step measures take into account only the matching time points between two series, while elastic or dynamic ones also take into account time shift. This is shown in Figure 11, where simple lock-step euclidean distance is compared to dynamic time warping.

For instance, correlation factor is a traditional lock-step metric, while dynamic time warping is the most widely used when dealing with elastic metrics.

DTW compares all points in the target feature combined to all the other points in the exogenous feature, which then leads to a matrix representing all distances between all combination of 2 points, one in each series. The objective then is to find the shortest distance from start to end points constrained by the fact that no distance is allowed to cross lines with each other.

A good visual representation of DTW is presented in Figure 12, where the binary

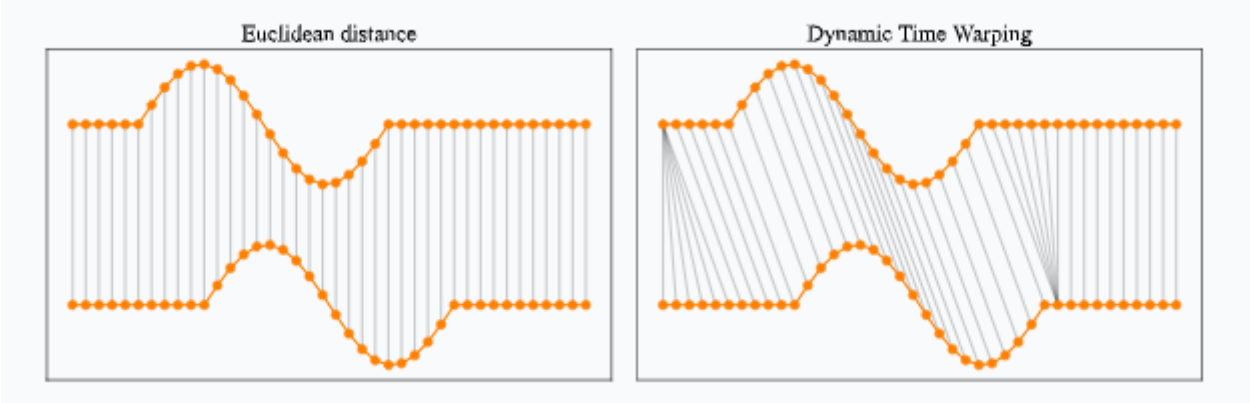


Figure 11: Dynamic time warping time shift resistant distances comparison to euclidean distance. Source: rtavenar

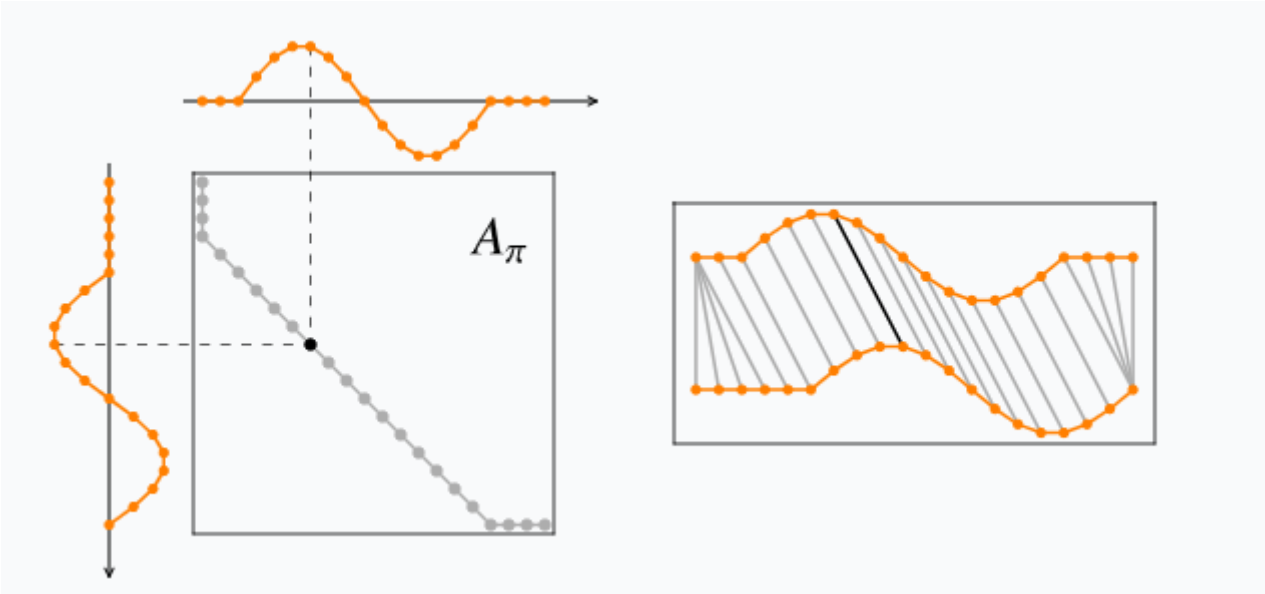


Figure 12: Visual comparison of distances between two time shifted series through DTW. Source: rtavenar

matrix A_π, defined in Equation 2.2, represents the dot product point by point with the target series in the x axis and the exogenous series in the y axis.

$$(A_\pi)_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \pi \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The left side of Figure 2.2 has only the highlights of the nonzero elements in gray, the other elements are zero. The nonzero elements represent a time-matching point between both series.

Christen et al. (2022) shows that these methods for analyzing the quality of exogenous features are not yet robust enough in terms of alignment with flexibility in amplitude

warping, and not yet robust enough when dealing with outliers and missing data, which is very common in large data sets. For that reason, DTW and similar methods are the state-of-the-art method for evaluating quality of exogenous time series, but there is still room for developing better methods. Anyway, for sure these methods are needed to be considered when dealing with exogenous features.

3 BASE THEORY

Before going into the model details, as well as discussing the data flow architecture, we will present some basic principles and concepts about the forecasting subject, particularly focused on time series.

3.1 Time Series

A time series is a set of sequential data in which order matters over time, also referred to as signal. Let us define it as a vector z, in which $z_t \in \mathbb{R}$ represents the value of the series at the instant t. And t can also be real, but for the sake of simplification, we are going to assume $t \in \mathbb{N}$ as Natural.

Time series can be discrete or continuous. Continuous series are mostly modeled rather than observed, and discrete series are generally collections of observations or measures for different time instants along a continuous series. Discrete series are the most common and are the standard taken in this study.

These observations can be taken on a periodic basis, such as a constant time interval between each pair of measures, which is called a univariate time series, but can also be taken at any time with no respect to a constant delta, which is called a multivariate time series.

Another important classification of time series regards its stationarity, and time series can be classified as either stationary or nonstationary. Stationarity in this context means that the behavior of a time series does not change over time, the behavior of a time series is described by its statistical properties, such as mean and variance. That means a stationary time series would, for instance, develop itself along time around a constant mean with an expected and well known variance. On the other hand, for example non-stationary time series can behave exponentially exploding or even be a mix of different stationary parts.

In traditional statistics, the most common premise about a time series for building a predictive model is that it be stationary. But nature is not generally stationary, so one of the most common transformations to make a non-stationary discrete series a stationary one is to subtract each data point from its previous value as detailed in 3.1, known as the return equation.

$$\Delta z_t = z_t - z_{t-1} \tag{3.1}$$

In case of non-stationary resulting returns, the difference can be made once again, obtaining the second series of returns 3.2.

$$\Delta^2 z_t = \Delta(z_t - z_{t-1}) = z_t - 2z_{t-1} + z_{t-2} \tag{3.2}$$

In most of the situations, the second return will be enough to get a stationary series.

3.2 Time series forecasting

3.2.1 Problem definition

A time series prediction problem involves making a forecast or prediction of future values of a time series based on its past values. A time series is a set of data points collected at regular intervals over time. In a time-series prediction problem, the goal is to use historical data to create a model that can accurately predict future values of the time series. The present research aims to improve the accuracy of state-of-the-art forecasting.

Mathematically, a time series can be represented as a sequence of $n \in \mathbb{Z}$ observations, mathematically represented in Equation 3.3:

$$z = z_1, z_2, ..., z_t \tag{3.3}$$

where z_t is the value of the time series at time t. The goal of time series prediction is to estimate the value of y at time $t + 1$, denoted as z_{t+1}, based on the values observed up to time t.

A time series prediction model typically involves a set of parameters that are used to estimate the relationship between the past values of the time series and future values. The model, be it statistical or machine learning, can be expressed as a function, as seen

in Equation 3.4:

$$z_{(t+1)} = f(z_1, z_2, ..., z_t; \theta) \tag{3.4}$$

where f is the prediction function that represents the model that receives a historical input and outputs the prediction for the next value. θ are the model parameters that need to be estimated, it can be several parameters, thus it is abstracted into θ.

The accuracy of a prediction model is evaluated by comparing the predicted values with the actual values of the time series. Metrics for evaluating accuracy are MSE, MAPE, and RMSE, for example, which measures the difference between the predicted values and the actual values in different ways. The goal of the prediction model is to minimize the error metric of choice and to make accurate forecasts of future values of the time series.

We add to the classical problem of forecasting a certain time series the problem of forecasting time series based not only on its history but in phenomenon related variables history, here called as exogenous features. The problem is then adapted to Equation 3.5 that deals with multiple series ${z_i}^j$ of observations and tries to predict a set of the same series of observations, where j is the number of total series, including exogenous ones, and k_m represents the indexes of the target prediction features, $k_m < j, m \in \mathbb{Z}$.

$$(z^{k_1}, z^{k_2}, ..., z^{k_m}) = f({z_1}^1, {z_2}^1, ..., {z_t}^1, {z_1}^2, {z_2}^2, ..., {z_t}^2, ..., {z_1}^j, {z_2}^j, ..., {z_t}^j; \theta) \tag{3.5}$$

3.2.2 Exogenous series

The term *exogenous* comes from initial statistical definition of an explanatory variable that is not correlated to the prediction error in a linear regression model, in contrast to *endogenous* variables. But lately this term has been used to describe a variable different than the target prediction series, and this is the use of this term in this work. The target series in turn, is the one to be predicted and where the result interest lies on.

Since there are multiple series considered, the research problem can be classified as multivariate time series forecasting, although multivariate can also mean non-constant consecutive time step intervals. However, it is important to notice that in this work we are not interested in predicting multiple features, we are interested in predicting a single target feature based on input of multiple exogenous features.

3.2.3 Statistical predictive models

The first statistical models to predict time series were made by stochastic processes, processes that are ruled by probabilistic laws. Among the classics books of statistics for prediction we have (Box et al., 2012) which focuses on Box-Jenkins first approaches, and there we find references for the well-known statistical predictive models. A mathematical definition of a stochastic process is when z is actually a random variable.

There are two types of models: the parametric and non-parametric ones. A parametric model is one in which there are a limited number of parameters and is most commonly used with an *a priori* hypothesis. The nonparametric ones are instead not limited in parameters and generally do not suppose anything about the data in an *a priori* manner.

An interesting characteristic classified among stochastic processes is the self-dependence, or autocorrelation. This happens when the model value z_t has a dependency on the previous value z_{t-1} or a set of previous values. When there is autocorrelation, the series is called a Markovian series.

A well-known example of a parametric model is linear regression, which predicts a continuous outcome based on a linear relationship between input variables and a dependent variable. Another example is logistic regression, which is used for binary outcomes by modeling the probability of an event occurring using a logistic function. Time series forecasting models, such as AutoRegressive Moving Average (ARMA) and AutoRegressive Integrated Moving Average (ARIMA), also belong to this category. The ARMA model combines autoregressive (AR) and moving average (MA) components to model stationary time series data. ARIMA, an extension of ARMA, includes an integrated (I) component that helps model non-stationary data by differencing it to achieve stationarity. These models are widely used in time series analysis for predicting future values based on past observations.

On the other hand, non-parametric models do not assume a specific functional form and are more flexible in modeling data with complex relationships. An example of a non-parametric model is the k-nearest neighbors (KNN) algorithm, which predicts outcomes based on the values of nearby data points, making it adaptable to various types of data. Another widely used non-parametric model is decision trees, which split data into subsets based on feature values and recursively make predictions for each subset, without requiring a fixed parametric form. Other non-parametric models, like random forests and support vector machines (SVM), are powerful tools in machine learning that build on decision trees but improve prediction accuracy through ensemble methods and non-linear decision

boundaries. Non-parametric models can be highly powerful as they don't make strict assumptions about the data, though they often require larger datasets and can be more computationally intensive.

3.3 Feature selection

Feature selection is a critical preprocessing step in time-series prediction, with the aim of enhancing the interpretability of the model, reducing computational costs, and improving predictive accuracy by identifying the most relevant exogenous variables. Unlike feature selection for static datasets, time series feature selection presents unique challenges due to temporal dependencies, non-stationarity, and lagged relationships. Recent advances in this domain leverage statistical, information theory, and machine learning-based approaches to optimize feature selection.

Feature selection methods can be broadly categorized into filter, wrapper, embedded, and hybrid approaches, each with distinct theoretical underpinnings such as in Lai et al. (2023) and in Yang et al. (2023).

3.3.1 Methods

All feature selection methods can fit into three main categories Yang et al. (2023), these are detailed below. It is worth to mention that mixing up the three is possible and is called a hybrid method.

3.3.1.1 Filter

Filter methods assess the relevance of the features independently of the predictive model, relying on statistical or information-theoretic metrics. Common approaches reviewed by Yang et al. (2023) and Christen et al. (2020) include:

- **Correlation-Based Selection**: Pearson's correlation and Spearman rank correlation measure linear and monotonic relationships between features and the target variable.
- **Mutual Information and Entropy-Based Methods**: Information-theoretic criteria, such as Mutual Information (MI) and Conditional Entropy, quantify nonlinear dependencies.

- **Dynamic Time Warping (DTW)**: A traditional metric that accounts for temporal misalignments, widely used in feature selection but limited in relevance quantification Berndt and Clifford (1994).
- **Forward Angular Relevance Metric (FARM)**: A recently proposed metric improving similarity measures by incorporating local and global influence assessments, as in Christen et al. (2023).

3.3.1.2 Wrapper

Wrapper methods iteratively evaluate subsets of features using a predictive model. These methods are computationally intensive as they rely on training and testing models for different subsets of characteristics to identify the most effective combinations. And for this reason they are not model-agnostic, they actually depend on the specific model. Some common wrapper techniques include the following.

- **Forward Selection**: Starts with an empty set and iteratively adds the most relevant features based on model performance.
- **Backward Elimination**: Begins with all features and systematically removes the least relevant ones to improve performance.
- **Recursive Feature Elimination (RFE)**: Repeatedly builds a model, ranks features by importance, and eliminates the least relevant ones in a stepwise manner.

Despite their accuracy, wrapper methods can be computationally expensive, making them impractical for large datasets. However, they offer significant advantages in discovering feature interactions and optimizing selection.

3.3.1.3 Embedded

Embedded methods integrate feature selection within the model training process. Regularization techniques like LASSO (L1-penalty) and Ridge Regression (L2-penalty) impose sparsity constraints, effectively eliminating irrelevant features. Tree-based models such as Random Forest and XGBoost leverage feature importance scores derived from split criteria.

3.3.2 Mechanisms

The challenges in time series feature selection can relate to temporal dependence, i.e. features that exhibit autocorrelation, requiring selection methods that account for lag effects and cross-dependencies; also can relate to non-stationarity, because the feature relevance may vary over time, necessitating adaptive selection strategies; or event multicollinearity and redundancy, some time series can be very correlated and bring no additional information, for this reason Principal Component Analysis (PCA), Independent Component Analysis (ICA), and feature clustering techniques help mitigate redundancy.

Some Deep Learning models for time series leverage attention mechanisms, such as the temporal fusion transformer, which together with recurrence in the neural network may achieve better performance and interpretability (Koya and Roy, 2024) than LSTMs and standard transformers.

Another widely used mechanism for feature selection is the use of relevance metrics, which can be used in a model and domain-agnostic filter-based selection method. Evaluating the relevance of exogenous features is essential to improve the accuracy of the forecast and prevent useless computation. State-of-the-art relevance metrics are listed below.

- **DTW (Traditional Metric for Similarity)**: Measures distance-based similarity rather than predictive relevance.
- **FARM (Recent Metric for Relevance)**: Quantifies predictive impact while compensating for time warping.
- **Euclidean Distance**: A simple distance metric used to quantify differences between feature vectors.
- **Error Metrics (e.g. RMSE, MAPE)**: Evaluate how much deviation occurs between predicted and actual values, useful for ranking exogenous features based on their contribution to prediction errors. It is interesting to notice that the RMSE and the Euclidean distance measure the same behavior but have different values.
- **Derivative Dynamic Time Warping (DDTW)**: An extension of DTW that considers first-order derivatives to improve time-series alignment by capturing trends and slopes.
- **Correlation**: Traditional Pearson correlation measures the statistical dependency between an exogenous feature and the target variable, often used for initial feature ranking.

Feature selection effectiveness is assessed using performance metrics (e.g., RMSE, MAE) and feature interpretability. Cross-validation and ablation techniques adapted for time series, such as walk-forward validation or exhaustive accuracy comparison methods can ensure reliable evaluation.

3.4 Machine learning

Within artificial intelligence models, there are two main approaches to how to train the model, and it depends on the objective and on the data shape itself. One can use a supervised learning algorithm, which prerequisites are the data to be previously valued, and the unsupervised learning, which will define the value of the data.

The supervised method requires the data output to be labeled with correct values, the values that the model should predict, be it classes or be it numbers.

The unsupervised approach is mostly used when the data is hard to label or is not originally labeled, so that the objective is to group certain similar data and find meaning for each discovered group.

In this work, we use neural networks with a supervised learning algorithm.

The artificial intelligence models are also able to learn patterns, similarly to humans, there are two main types of learning: supervised and unsupervised learning.

On the one hand, supervised learning is analogous to learning with a teacher, who can tell what is correct and what is wrong so that the student can target what is right and avoid what is wrong. The same occurs with neural models, where the output is compared to a certain selected data that represent a part of the full data that must be predicted.

On the other hand, unsupervised learning occurs when one does not have a teacher. In this way, one must use other techniques to progress. An example is when one gets hurt, he or she learns that the action that caused the wound is not to be repeated. That is an external stimulus that our body senses has evolved to feel and work in this way to prevent us from getting hurt. The same stimulus occurs when a treat is given to a dog that behaved well to positively reinforce the good action. There are artificial intelligence models that work very similarly, and this is called reinforcement learning, which is an unsupervised learning method. Still, there are other unsupervised methods, such as clustering, which works by grouping data with similar behavior.

3.4.1 Supervised learning

Our main focus here is supervised learning, which is the most widely adopted within neural networks for time series prediction.

In this section the basics of supervised learning, the mostly used technique for neural networks, is described.

3.4.1.1 Split data set

As mentioned, the supervised learning uses a teacher to know what is correct and what is wrong prior to learning. This teacher, in fact, is a sample of the full target behavior to be predicted, usually a sample of the full data set. This sample will be used by the model to learn the behavior that it represents hoping that the behavior of the sample can be applied to the behavior of the whole data set. That sample is called *train data set*.

Besides the train data set, there is also the *validation data set*, because the behavior of the train sample may not represent very well the whole data set, so the model performance is compared during the training phase with the performance on the validation throughout the learning process. This step is important because it is possible to determine whether the model is overfitting to the specific characteristics of the train data set or if it has some degree of generality. Overfitting is discussed further at the *regularization* section.

Finally, the model is tested against the *test data set*, which is another part of the whole data set that is never been used in anyway during the training phase, that is the data set from which performance of the model will be measured.

Usually the training data set is the the biggest of the three data sets, ranging from 60% to 90% of the whole data set, depending of the data size, the rest is the test data set. 10% to 30% part of the train phase is commonly used as validation.

It is also important to note that all the train, validation and test data sets are used in the model's output domain. These output data set may or may not be the same as the input data set.

3.4.1.2 Training

The train consists in adjusting the weights of the neural model in order to minimize the error between the predicted and the target result. The target result comes from using the train data set in a supervised manner.

A single train step is to show the model the input, make it produce its output and compare its output to the expected result from the corresponding output data set (train data set, for example); after comparing, it is needed to adjust the weights of the model towards reducing the target error measure.

The weight adjustment can be done in two ways: adjust after each output production or adjust after a certain amount of outputs.

Adjusting after each output is called on-line learning, and adjusting after a couple of outputs is called batch learning. The main difference between these two methods is how one calculates the error measure that needs to be minimized. With only one output, the error is the arithmetic difference between two numbers. With multiple outputs, one can select, for example, root mean square or mean average percentage errors to be calculated from a sequence of outputs.

The size of the batch is commonly chosen to be the full train data set. Once the model has completed predicting all inputs once, then the training phase has advanced an *epoch*, that epoch does not depend on the batch size.

3.4.1.3 Gradient descent

Gradient descent algorithm is also known as stochastic gradient descent or stochastic approximation, is attributed to Augustin-Louis Cauchy for suggesting it in 1847, began being used in nonlinear optimization with Haskell Curry in 1944.

Gradient descent is an iterative optimization method that is used to minimize an objective function. In the case of neural networks with supervised learning, this objective function is the chosen error function, which calculates how far the result of neural net is towards the target or expected value.

For a given differentiable and convex function section, the gradient descent takes steps towards the local minima from an arbitrary starting point in the following way, as of Equation 3.6:

$$x_n := x_{n-1} - \eta \nabla E(x_{n-1}) \tag{3.6}$$

Here η is the learning rate, which is a chosen hyperparameter and represents how far the iteration steps will go from its respective last iteration step. And x_n is the result of the gradient descent method, which aims to be the value where the error function E is

minimum in the end. Finally, x_0 is the starting point, generally an arbitrary x-axis point for the convex error function. An example of the gradient descent steps is shown in Figure 13.

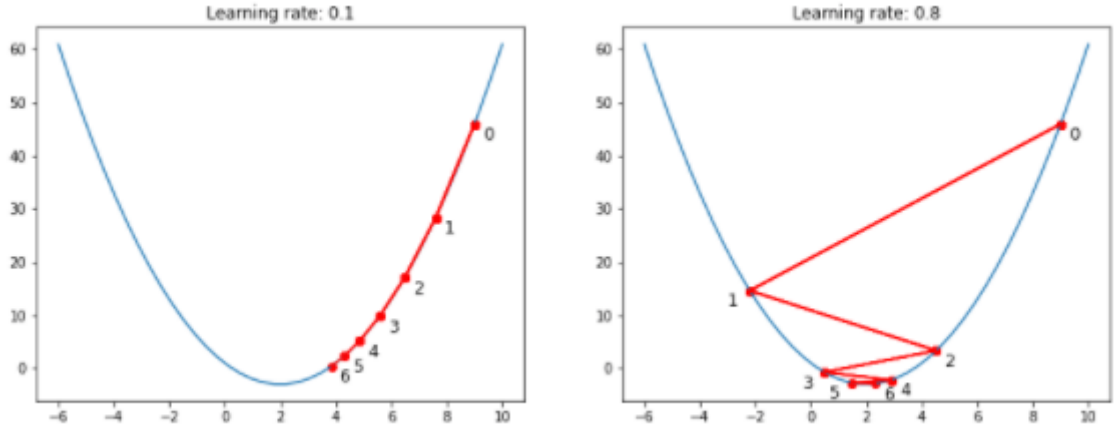


Figure 13: Gradient descent example with 10 iteration steps. Source:TowardsDataScience

The gradient descent method resembles the Newton method for finding roots of a function, the main difference being that Newton's method aims the target as the zero of a function result and gradient descent aims the target as the minimum of the function.

3.4.1.4 Back propagation

The backpropagation method uses the gradient descent algorithm to optimize the weights in each neuron of a neural net. In Figure 14 we see a neuron that belongs to a network and is identified by j, it has m inputs denoted by y_i, which is a function of n and ranges from y_0 to y_m. The bias is adopted as the input y_0 that is taken as $+1$.

Equation 3.7 describes the output of the neuron j in this context, it uses the definition of a neuron and sums the weighted inputs which is then passed through an activation function φ.

$$y_j(n) = \varphi_j \left(\sum_{i=0}^{m} w_{ji}(n) y_i(n) \right) \tag{3.7}$$

The most important step in backpropagation algorithm is the derivative chain rule that makes the difference between the target and the result propagate to neurons weights, this is shown in Equation 3.8.

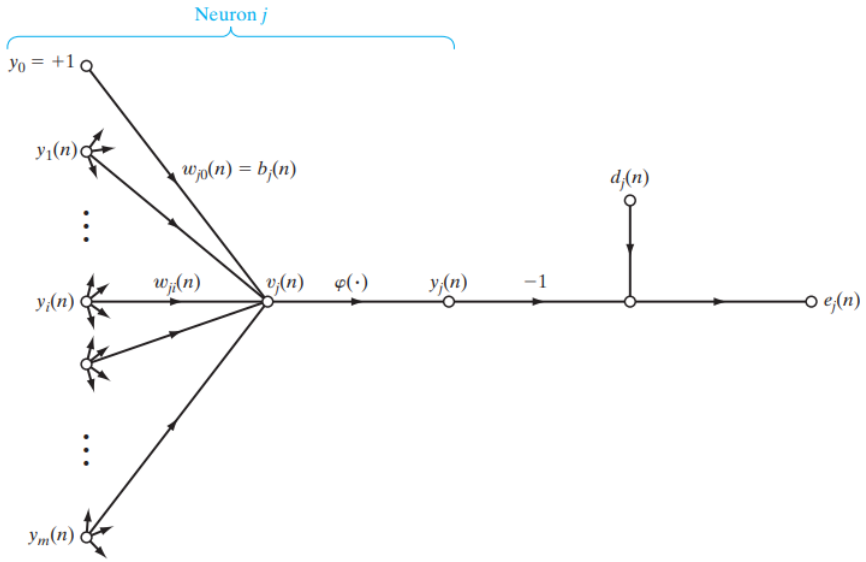


Figure 14: Neuron scheme for backpropagation algorithm (Haykin, 2009)

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial w_{ji}(n)} \tag{3.8}$$

At Figure 14, the $e_j(n)$ is the error signal for a single neuron and $E(n)$ is the error for the whole neural network, there are several means of calculating $E(n)$ such as MSE (mean squared error).

$$e_j(n) = d_j(n) - y_j(n) \tag{3.9}$$

In equation 3.9 the individual error for a single neuron j is defined as the difference between the *ith* and respective element of the expected output vector d in the supervised learning context, and $y_i(n)$ is the actual resulting output of the neuron j.

In order to calculate the error for the whole net, one must combine all the individual errors with a chosen metric, MSE in our case. The interesting part of choosing MSE as an error metric for the whole network is that its derivative against $e_j(n)$ depends on $e_j(n)$ itself.

$$E(n) = \text{MSE} = \frac{1}{N} \sum_{i=1} (e_i)^2 \tag{3.10}$$

$$\frac{\partial E(n)}{\partial e_j(n)} \propto e_j(n) \tag{3.11}$$

Equation 3.11 shows that the derivative is proportional to the error itself, since the constant factor can be absorbed by the η learning rate in the end, we do not need to keep track of the 1/2 that will appear in the derivative.

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.12}$$

Equation 3.12 shows that the derivative of the individual error signal in terms of the neuron output does not depend on the expected output vector and thus ends up resulting in the -1 for reference convenience.

The derivative of the output in terms of each weight comes from equation 3.7 using the chain rule, as in equation 3.14.

$$\frac{\partial y_j(n)}{\partial w_{ji}(n)} = y_i(n).\varphi'_j\left(\sum_{i=0}^{m} w_{ji}(n)y_i(n)\right) \tag{3.13}$$

Thus, substituting the partial derivatives 3.11, 3.14, and 3.12 into the chain rule equation 3.8, we have:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n).y_i(n).\varphi'_j\left(\sum_{i=0}^{m} w_{ji}(n)y_i(n)\right) \tag{3.14}$$

Observing the gradient descent of Equation 3.6, one can see that the delta to change the weight w_{ji} is the following:

$$\Delta w_{ji}(n) = -\eta\frac{\partial E(n)}{\partial w_{ji}(n)} \tag{3.15}$$

And this concludes the step of the bckpropagation optimization algorithm, this step can be repeated until the error satisfies the stopping criteria, a common stopping method is to set a certain number of repetitions (epochs) with no enhancement in error.

This technique to stop the repetitions when no enhancement is measured is called early stopping, which prevents the optimization method from overfitting the weights towards the expected response vector, which ends up making the model predict a specific pattern rather than the goal of predicting a generic pattern.

3.4.2 Neural networks

Commonly known as artificial neural networks these models arose in order to try to simulate human brain's mechanism to compute. Unlike standard digital computers, the brain works in a non-linear and parallel manner. The brain has a high density of neurons that perform tasks, such as pattern recognition and perception, many times faster than conventional digital computers, because of this different architecture.

In addition to increasing information processing throughput with the human brain architecture, it also has the capacity to create and modify itself to build rules that work better than others, which is what we call experience. This is an adaptation ability that makes our brain the most complex information processing system we know.

The definition of a neural network follows as Haykin (2009), which was adapted from Aleksander and Morton (1990).

> "A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity to store experiential knowledge and make it available for use. It resembles the brain in two respects:
>
> 1. The network acquires knowledge from its environment through a learning process.
> 2. The strengths of the interneuron connection, known as synaptic weights, are used to store the acquired knowledge."

The widely adopted neuron models try to replicate the human-neuron interaction, the first to work on this idea was Rosenblatt (1958), the so-called perceptron model came as a result.

3.4.2.1 Neuron

The most common neuron model is also called Rosenblatt's perceptron (Rosenblatt, 1958), which in turn is based on the McCulloch and Pitts (1943) model for the human neuron.

$$y_k = \varphi(b_k + \sum_{j=1}^{m} w_{kj} x_j) \tag{3.16}$$

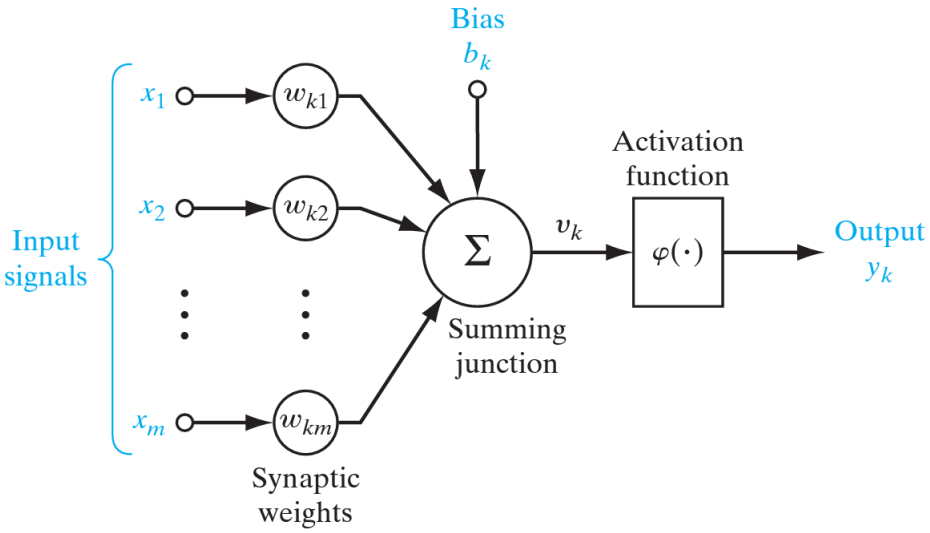


Figure 15: Artificial neuron model (Haykin, 2009)

Equation 3.16 summarizes the math of the *kth* neuron model inside a network, which has inputs ranging from x_1 to x_m and w_{kj} as weights (learning and specialization result), bias b_k for more flexibility in optimization, and φ as the activation function, which introduces nonlinearity, all this results in the output y_k.

A close look at equation 3.16 leads to the autoregressive model (AR), which is a particular case of the neuron model when φ is the identity function and when x_i takes successive temporal meaning. The only difference between AR model and the Neuron model is the activation function which brings nonlinearity, this behavior is what makes the neural net able to be optimized for a wider range of phenomena.

Theorem 1. *Universal approximation Theorem: a single-layered sigmoid neural net is able to approximate any continuous function.*

The universal approximation theorem was first shown by Cybenko (1989) for sigmoid feed forward nets. In addition, the theorem was proven for all other neural network architectures, including recurrent (Schäfer and Zimmermann, 2006) and convolutional networks (Zhou, 2020). It has also been proven that any non-polynomial activation function also maintains universality Leshno et al. 1992.

Although only 1 hidden layer is needed to be universal, multiple layers can ease the convergence of weights to faster and better solutions . In other words, shallow networks are universal and deep networks are too, but when dealing with the parameter training of the whole network, it has been figured out that deep networks can do the same work with

much lower number of training parameters, and the equivalent shallow network would need to have an infeasibly large hidden layer.

The activation function is responsible for bringing nonlinearity to the neuron model, this means turning the output behavior variable in order to benefit a certain kind of expected values. This is similar to filtering in the signal processing area: one of the most traditional filter functions is the *Heaviside* function.

It is also the nonlinearity of activation functions that make it possible to arbitrarily increase the number of layers in a neural net, otherwise any combination of multiple linear weights and functions in each layer could be described by a single linear layer.

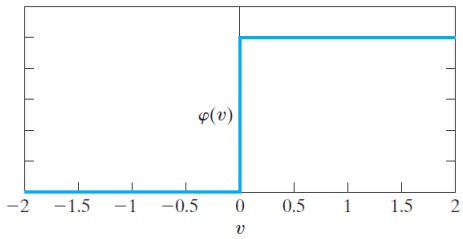


Figure 16: Heaviside activation function. Source:(Haykin, 2009)

This Figure 16 activation function lets only positive outputs pass through and blocks negative ones, it ranges from 0 to 1 on the y axis (counter domain), and uses the entire x axis for its inputs (domain), then transforms a range $[-\infty; +\infty]$ of values into a set of $\{0; 1\}$. This is discontinuous, but if Heaviside had a continuous version it could be the Sigmoid function.

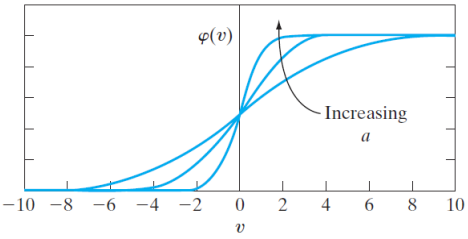


Figure 17: Sigmoid activation function. Source:(Haykin, 2009)

$$\varphi(x) = \frac{1}{1 + e^{-ax}}, x \in \mathbb{R} \tag{3.17}$$

The function in Figure 17 transforms a range $[-\infty; +\infty]$ of values into a set of $[0; 1]$. And it can be shaped to be more or less smooth with its parameter a in Equation 3.18. Note that the Heaviside function is a particular case of the Sigmoid function when the parameter a tends to infinity.

This format of equation 3.18 is interesting because its derivative can be easily computed recursively, which is convenient for programming purposes.

$$\varphi'(x) = \varphi(x)(1 - \varphi(x)), x \in \mathbb{R} \tag{3.18}$$

There are many other activation functions, but for clarity, they will be mentioned when needed.

3.4.2.2 MLP

The multilayer perceptron (MLP) is the result of chained neurons in a directed graph format, which is called *feedforward*.

In Figure 21 a small case of an MLP is shown, which has only two hidden layers.

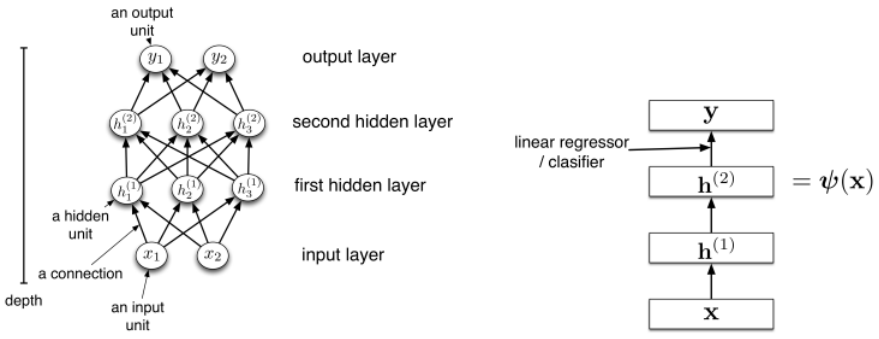


Figure 18: 2 layer MLP. Source:University of Toronto (Haykin, 2009)

The outputs y of this MLP can be computed as follows:

$$\begin{aligned} h_i^{(1)} &= \varphi^{(1)}\left(\sum_j w_{ij}^{(1)} x_j + b_i^{(1)}\right) \\ h_i^{(2)} &= \varphi^{(2)}\left(\sum_j w_{ij}^{(2)} h_j^{(1)} + b_i^{(2)}\right) \\ y_i &= \varphi^{(3)}\left(\sum_j w_{ij}^{(3)} h_j^{(2)} + b_i^{(3)}\right) \end{aligned} \tag{3.19}$$

Where:

$h_i^{(l)} \rightarrow$ represents the output of the hidden unit i in the layer l.

$\varphi^{(l)} \rightarrow$ represents the activation function of the layer l.

$b_i^{(l)} \rightarrow$ represents the bias of the unit i in layer l.

$w_{ij}^{(l)} \rightarrow$ represents the weight between the unit i in layer l and the unit j in layer $l-1$.

3.4.2.3 RNN

RNN stands for recurrent neural networks, and its key feature compared to MLP is that the result of an iteration takes part in the input of the next iteration. This is the feedback loop connection, which is very different from traditional feedforward neural nets. The RNN architecture diagram of a single step can be seen in Figure 19 and Equation 3.20.

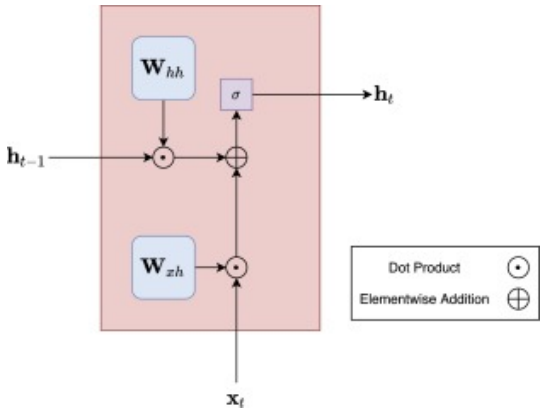


Figure 19: Vanilla RNN. Source:Sciencedirect (Haykin, 2009)

Where:

$\mathbf{W}_{hh} \rightarrow$ represents hidden weight matrix.

$\mathbf{W}_{xh} \rightarrow$ represents input to the hidden weight matrix.

$$\mathbf{h}_t = \sigma\left(\mathbf{h}_{t-1}\mathbf{W}_{hh} + \mathbf{x}_t\mathbf{W}_{x\leftarrow\leftarrow}\right) \tag{3.20}$$

Recursive neural networks can deal well with sequential nature data because most of sequential data depend on previous steps in a certain way. So RNNs can be used to predict series in a synchronous way and in an asynchronous way, the synchronicity varies if the step t of the input series predicts the same step t of the target series. But one can also predict only a certain desired point in the series, the last point in most of the cases. These use cases are described in Figure 20.

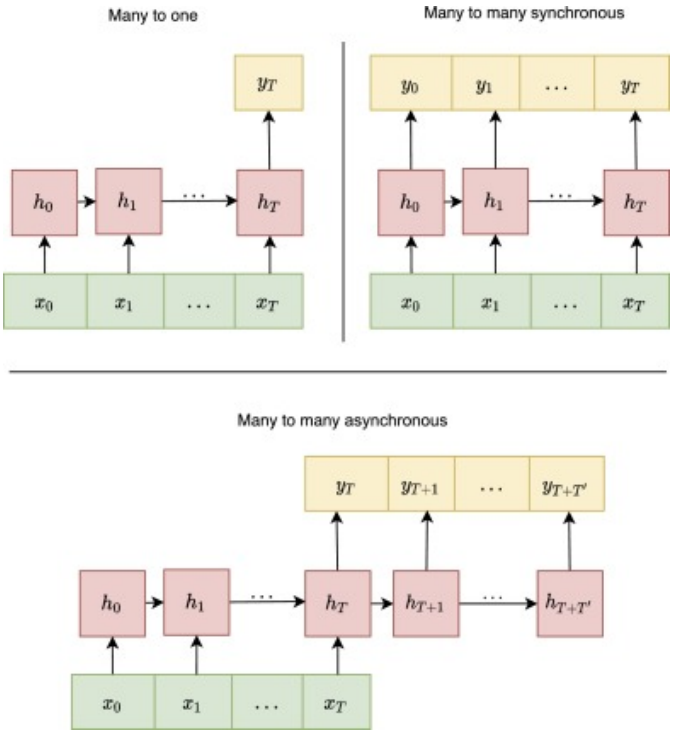


Figure 20: Vanilla RNN use cases for predicting series. Source:`https://www.sciencedirect.com/topics/computer-science/recurrent-neural-network`, Sciencedirect (Haykin, 2009)

3.4.2.4 LSTM

The architecture of long-short-term memory neural networks was first proposed by Sepp Hochreiter (1997). It is a variation of recurrent neural networks that deal with the problem of exploding gradients and long-term "forgetting", typical problems of recurrent neural networks.

The LSTM architecture contains 3 gates, the Input gate, the Forget gate, and the output gate.

1. **Input gate**: lets the new information pass to be stored in the cell state.
2. **Forget gate**: discards the information that was stored in the cell state.
3. **Output gat**e: provides the activation for the final output for the LSTM cell at a certain iteration for the time step t.

The equations for each of the LSTM gates are as follows:

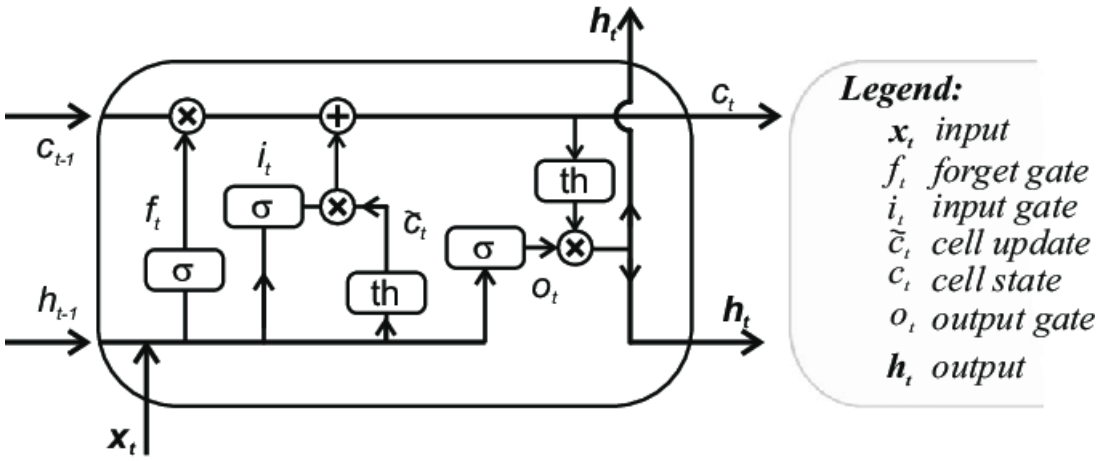


Figure 21: LSTM architecture diagram. Source:`https://www.researchgate.net/figure/LSTM-cell-with-its-internal-structure_fig4_332766508`, (Haykin, 2009)

$$
\begin{aligned}
i_t &= \sigma\left(w_i\left[h_{t-1}, x_t\right] + b_i\right) \\
f_t &= \sigma\left(w_f\left[h_{t-1}, x_t\right] + b_f\right) \\
o_t &= \sigma\left(w_o\left[h_{t-1}, x_t\right] + b_o\right)
\end{aligned}
\tag{3.21}
$$

The equations for the cell state updating are the following:

$$
\begin{aligned}
\tilde{c_t} &= \tanh\left(w_c\left[h_{t-1}, x_t\right] + b_c\right) \\
c_t &= f_t * c_{t-1} + i_t * \tilde{c_t} \\
h_t &= o_t * \tanh\left(c^t\right)
\end{aligned}
\tag{3.22}
$$

Where:

$i_t \rightarrow$ represents the input gate.

$f_t \rightarrow$ represents the forget gate.

$o_t \rightarrow$ represents the output gate.

$\sigma \rightarrow$ represents the sigmoid function.

$w_x \rightarrow$ weight for the respective gate neurons (x).

$h_{t-1} \rightarrow$ output of the previous LSTM block(at timestamp $t-1$).

$x_t \rightarrow$ input at the current timestamp.

$b_x \rightarrow$ biases for the respective gates (x).

$c_t \rightarrow$ cell state (memory) at timestamp (t).

$\tilde{c}_t \rightarrow$ represents a candidate for the cell state at timestamp (t).

The LSTM architecture is recursive and thus works well for sequential data, but what makes it better than vanilla RNNs are the gates that make it possible to select data to keep and to discard during training. These gates fix the problem of long forgetting that happens in Vanilla RNNs: when the sequential data are long, the oldest data (first data that is learned) is forgotten because the new values change the weights with no chance to keep old learned behavior.

3.5 Signal Decomposition

Signal decomposition is a technique that describes a complex signal as a combination of different simpler components, such that complex analysis can be done with signals that approximate good *a priori* hypothesis such as stationarity and periodicity.

Signal decomposition is also part of the signal processing techniques, which starts around Fourier and is necessary for a good understanding of the general objective in decomposing signals. To clarify that we have the Appendix A, that shows more detail about the classic signal processing techniques.

Here the main focus will be on the EMD method and its variations because of its capability to decompose without any a priori condition about the signal, although there are many other important methods, such as Fourier, Wavelets and PCA, that need to be studied in order to fully understand what surrounds the EMD-based methods. These other methods have also been used along with neural models for prediction (Wang and Lou, 2019).

3.5.1 EMD

The EMD method stands for: empirical mode decomposition and was first published by N.E. Huang et al (1998). The main objective of this method is to convert a complex and non-stationary signal into several components of linear and stationary signals.

The maxima and minima are defined as the points in which the first derivative of the signal changes to the opposite: if the derivative was positive before that point and negative after that point, then it is classified as a local maxima, otherwise it is a minima.

The components generated by the EMD method are called IMFs, intrinsic mode functions, and must obey 2 conditions:

1. The number of extreme values and zero-crossings must be equal or differ at the most by one.

2. The mean value of the envelope constructed by the local maxima and minima is zero at any point.

3.5.1.1 Mathematical steps

Let us call k as the component level, d_k is the IMF of the k-th level and r_k is the residue of the k-th level component.

1. With $k = 0$, connect with spline interpolation each of the local maxima u and each of the local minima f of the signal. One could use an akima-like spline or a traditional cubic spline.

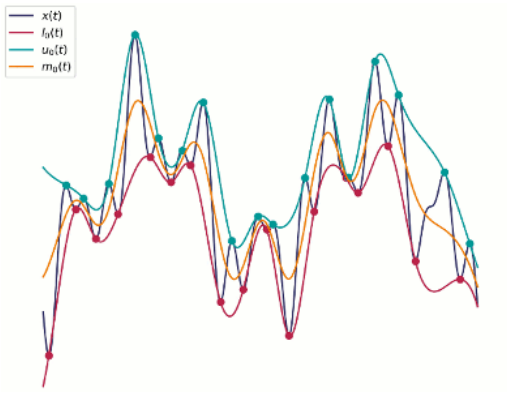


Figure 22: Maxima u, minima f, signal x and average m [Source: `https://upload.wikimedia.org/wikipedia/en/4/49/Emd_example_lowres.gif`]

2. Obtain the mean of the maxima and minima envelopes. $m = (u + f)/2$

3. Obtain the IMF candidate: $d_{k+1} = r_k - m$. r_0 is the original signal itself.

4. Check whether d_{k+1} is an actual IMF.

If d_{k+1} is an IMF (according to the 2 criteria cited before), calculate $r_k = Z(t) - \sum_{i=1}^{k+1}$ and then repeat the process since step 2 with r_k as original signal and proceed to the next k value.

If not, repeat the process from Step 2 with the same d_{k+1} as the original signal.

5. Keep getting different IMFs until either a maximum number of IMFs is reached or until a certain signal metric, such as power, reaches a desired value or even when the residue becomes monotonic.

The process of repeating steps until all values have been decomposed is called sifting. And by the nature of this method's construction, the original signal can be recomposed by adding up all the obtained IMFs: $Z(t) = r_K + \sum_{i=1}^{K} d_k$, where K is the maximum level mode.

3.5.1.2 Mode mixing

Mode mixing is a phenomenon that often occurs in the EMD decomposition when the original signal is intermittent. It causes the IMFs to include frequency behaviors of other adjacent levels, rather than only their own level.

The most studied example is available at `https://laszukdawid.com/blog/2014/07/11/mode-mixing-in-emd/#ref4` and was shown by Wu et al. (2011). Daubechies is one of the most prominent authors in the field of wavelets; there are even wavelet formats with their name.

Let us take the following original signal, $s(t) = \sin(t) + b\sin(\omega t)$, which is a sum of two sines of different frequencies and amplitudes. When adding sinusoidal signals with very different frequencies we can decompose each behavior into a different IMF, as can be seen in Figure 23.

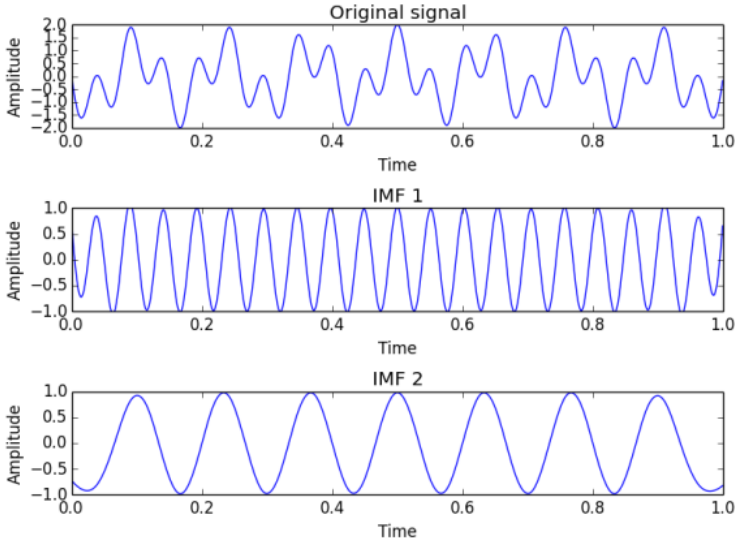


Figure 23: Very different frequencies sines addition. [Source: `https://laszukdawid.com/blog/2014/07/11/mode-mixing-in-emd/#ref4`]

This clear separation is not observed when it comes to signals with very close frequencies, especially when beating occurs, as seen in Figure 24.

In Figure 24, we observe that the first IMF is almost equal to the original signal and

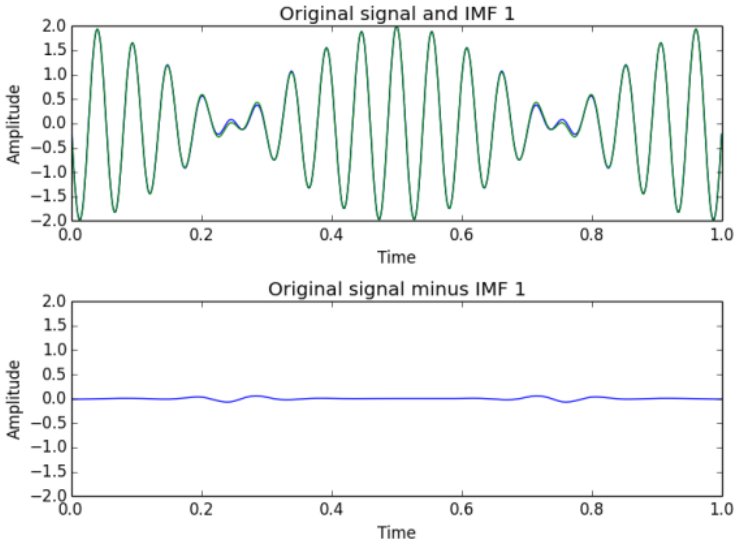


Figure 24: Very similar frequencies sines addition. [Source: `https://laszukdawid.com/blog/2014/07/11/mode-mixing-in-emd/#ref4`]

the first residue has almost no power but shows some occasional disturbances. That is the mode-mixing phenomenon.

3.5.2 EEMD

EEMD method stands for ensemble empirical mode decomposition. It aims to solve the problem of mode mixing that occurs with the EMD method.

EEMD uses noise to assist in mode mixing solving.

3.5.2.1 Mathematical steps

Let $w^i(t)$ be a zero mean noise signal, $i = 1, 2, ...I$. Let $E(.)$ be the EMD decomposition operator of the process explained in the EMD subsection. Let d_k be the each level's IMF candidate (mode).

1. Create a noise added signal $Z^i(t) = Z(t) + w^i(t)$ for each $i = 1, 2, ...I$.

2. Decompose each of the $Z^i(t)$ for $i = 1, 2, ...I$ by EMD. Resulting in the $d_k^{(i)}$ for each i-th noise realization.

3. The actual k-th EEMD mode will be the average of each noise added signals from the same k level: $\bar{d_k} = \frac{1}{I}\sum_{i=1}^{I} d_k^{(i)}$.

4. Calculate the residue as in EMD, $r_k = r_{k-1} - d_k$ and repeat all the steps using r_k

as original signal. Until one of the same stop criterion is reached as in EMD.

Since the $\bar{d}_k$ does not disconsider the added noise, thus the recomposition of the original signal is not kept perfect as in EMD. The hypothesis here is that if we do many realization, that means getting I to a huge value, all the signals will tend to cancel themselves in the end, at least almost cancel. Enough to get negligible errors.

So we can state that $Z(t) \cong r_K + \sum_{i=1}^{K} \bar{d}_k$, where K is the maximum level mode, since it is approximate then the method is incomplete.

This method solves the mode mixing problem in a partial manner and introduces the negligible error.

3.5.2.2 CEEMD

CEEMD stands for complete ensemble empirical mode decomposition. And aims to solve the problem of the incompleteness in the EEMD method, while keeping the same mode mixing problem solution.

The main idea for the completeness is to add an opposite noise for each noise realization, making it 2 separate EEMD decompositions and then arithmetically averaging both.

The only change here, when compared to EEMD is how the IMF candidate is calculated: $\bar{d}_k = \frac{1}{2I} \sum_{j=1}^{2} \sum_{i=1}^{I} d_{j,k}^{(i)}$.

3.5.3 CEEMDAN

The CEEMDAN method stands for complete ensemble empirical mode decomposition with adaptive noise. This method was proposed by Torres et al. (2011) and takes into account noise addition to fully prevent mode mixing (phenomenon that causes components to mix frequency behaviors). This was an improvement over the EEMD (ensemble empirical mode decomposition) method proposed by Wu and Huang (2009). EEMD tried to solve the mode mixing problem between near-IMF frequencies by using sampling the sifting process of EMD, but was not successful in some special cases depending on the signal, especially when dealing with spiky signals such as numerical Dirac's delta, or impulses. The EMD in turn was the original idea of applying splines and curve envelopes with maxima and minima, and then subtracting the result from the previous signal, which is called the sifting process, proposed by (N.E. Huang et al, 1998) in 1998.

An example of a price signal purely decomposed via CEEMDAN is shown in Figure 25. In which we realize that the lower the IMF level, the higher the frequency it relates to, and also the higher the IMF level, the lower the frequency of the component, which is caused by the nature of the sifting process, which first extracts the high-frequency patterns leaving the long trends as residue. The frequencies generally are not constant in a single component, but mostly have similar magnitude in the same component and vary little along the time. The main benefit given by CEEMDAN technique is the recomposition: It enables us to recompose the original signal by just arithmetically adding the components, which is simple and fast, yet effective to describe different time-frequency patterns.

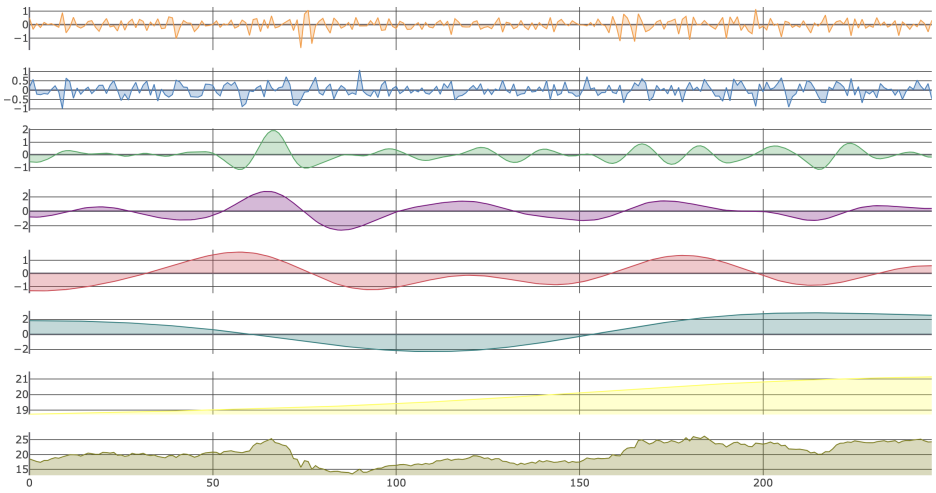


Figure 25: Example of CEEMDAN decomposition for the closing price signal of PETR4 from February 6th 2018 to February 11th of 2019. Legend from top to bottom: IMF1, IMF2, IMF3, IMF4, IMF5, IMF6, residue and original data. Horizontal axis represents time in days, and vertical axis represents IMF magnitude. Area colored for visualization purposes.

The main contribution of this method is the noise adaptation.

3.5.3.1 Mathematical steps

Let us call $E_j(.)$ the j-th mode of EMD. We call the standard deviation of a white Gaussian noise ε_0. We will also describe the different realizations of the white Gaussian noise with zero mean and unit variance as w^i, with $i \in \mathbb{Z}, 0 < i < I$, and I is the maximum number of realizations (number of components of the EMD method). And $Z(t)$ is the time series.

1. Calculate: $Z(t) - \varepsilon_0 \cdot w^i(t)$

2. EMD is used to decompose the original $Z(t)$, which will result in a certain number of components; let us call this number I. And calculate the first mode:

$IMF_1(t) = \frac{1}{I} \cdot \sum_{i=1}^{I} imf_1^i(t)$

imf_1^i is the first mode of each realization.

3. Find the first residue: $r_1(t) = Z(t) - IMF_1(t)$

4. Evaluate: $r_1(t) + \varepsilon_1 \cdot E_1(w^i(t))$, $i = 1, 2, ...I$ until the first EMD mode. IMF_2 will be calculated as the following average: $IMF_2(t) = \frac{1}{I} \cdot \sum_{i=1}^{I} E_1(r_1(t) + \varepsilon_1 \cdot E_1(w^i(t)))$

5. The k-th residue is: $r_k(t) = r_{k-1}(t) - IMF_k(t)$

6. Evaluate $r_1(t) + \varepsilon_1 \cdot E_1(w^i(t))$, $i = 1, 2, ...I$ until the first EMD mode and define the $(k+1)$-th mode:

$IMF_{k+1}(t) = \frac{1}{I} \cdot \sum_{i=1}^{I} E_1(r_k(t) + \varepsilon_k \cdot E_k(w^i(t)))$

7. Here the process becomes recursive: In order to get the next k steps 5 and 6 must be repeated. The stop condition is when the residue becomes a monotonic function.

The original signal may be expressed in terms of IMFs in the following manner:

$Z(t) = \sum_{k=1}^{K} IMF_k(t) + r_k(t)$

Here, $r_k(t)$ is the final residue and K is the total number of modes in the decomposition.

The CEEMDAN method is complete, thus we can recompose the signal by arithmetically adding each of the components and the residue.

3.6 Signal interpolation

Signal interpolation is the process of approximating an originally continuous function given only samples of this function (Caruso and Quarta, 1998). Although it approximates the original function, the resulting function ends up containing all interpolated data points so that this process is not classified as an approximation or curve fitting. In most of the use cases of interpolation, one has discrete observation data points of a certain phenomenon and needs to recover the general behavior of the original continuous function that describes that phenomenon.

3.6.1 Polynomial interpolation

In polynomial interpolation, the approximation is simple and just needs a $(n-1)$-th degree polynomial to interpolate a set of n data points (Gasca and Sauer, 2001).

Describing a generic polynomial as $p_n(t) = \sum_{i=0}^{n-1} a_i x^i$, one can use each of the coordinates of the data points by substitution and end up having an equation system $n \times n$, where the variables are the coefficient of the polynomial.

In order to prevent scaling the degree of the polynomial with the number of data points, one can use the interpolation by parts, fitting a polynomial periodically for a given amount of sequential data.

Linear interpolation is as simple as connecting each data point by a line. It must not be confused with the linear regression, which uses least sum of squares in order to find the optimal solution for a line that is the closest to all the data points.

3.6.2 Splines

Splines are nth degree piecewise polynomial interpolation that is continuously differentiable $n - 1$ times, it results in smooth curves that tend to approximate continuous behavior. The most common kind of spline is the cubic one.

But there is also a valuable and different method, that is, the Akima spline. The main difference between them is that the cubic spline takes into account the whole set of data points when defining the cubic polynomial interpolation, and the Akima spline takes into account only the vicinity of the target data points and adapts accordingly in a limited window.

Also, any interpolation method can be used as a predictive model for time series when the resulting interpolated function is extrapolated for the next unknown data points.

3.6.2.1 Cubic

Natural cubic splines are a piecewise interpolation method available in `https://towardsdatascience.com/numerical-interpolation-natural-cubic-spline-52c1157b98ac`, consequently it uses low-degree polynomials so that the fitting behaves more smoothly instead of spiky, that forces the resulting function to be twice differentiable. The requirements that must be met are the following.

- Between each pair of spline interpolated data points there is a segment that joins them together, which is a third-degree polynomial.
- Right at the data points, the first (slope) and second (bending moment) derivatives must be continuous.

- At the end-points, the second derivative must be zero, which occurs naturally.

The steps to find the cubic spline function given a set of n points $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ equally spaced, which means that it can be understood as a univariate time series, are the following:

1. The spline function can be represented as `https://www.rajgunesh.com/resources/downloads/numerical/cubicsplineinterpol.pdf`

$$S(x) = \begin{cases} s_1(x) & \text{if} \quad x_1 \leq x < x_2 \\ s_2(x) & \text{if} \quad x_2 \leq x < x_3 \\ & \vdots \\ s_{n-1}(x) & \text{if} \quad x_{n-1} \leq x < x_n \end{cases} \tag{3.23}$$

   s_i must be a third degree polynomial of the form:

$$s_i(x) = a_i (x - x_i)^3 + b_i (x - x_i)^2 + c_i (x - x_i) + d_i \mid i \in \mathbb{N}, i < n \tag{3.24}$$

2. The first and second derivatives of the third degree polynomials are described as follows:

$$s_i'(x) = 3a_i (x - x_i)^2 + 2b_i (x - x_i) + c_i \mid i \in \mathbb{N}, i < n \tag{3.25}$$

$$s_i''(x) = 6a_i (x - x_i) + 2b_i \mid i \in \mathbb{N}, i < n \tag{3.26}$$

3. Since the interpolation has to satisfy each of the data points, we can say that $S(x_i) = y_i \mid i \in \mathbb{N}, i < n$.

4. Focusing on a certain interval between two points x_i and x_{i+1}, we can say $S(x_i) = s_i(x_i)$, because the full spline function is gathered by parts of smaller third degree functions. Substituting these finding s in Equation 3.26 we have:

$$\begin{aligned} y_i &= s_i(x_i) \\ y_i &= a_i (x_i - x_i)^3 + b_i (x_i - x_i)^2 + c_i (x_i - x_i) + d_i \\ y_i &= d_i \end{aligned} \tag{3.27}$$

5. Each smaller third-degree function must join each other at the data points in order for the full spline function to have the first and second derivative continuous, so $s_i(x_i) = s_{i-1}(x_i)$ and $s_i'(x_i) = s_{i-1}'(x_i)$ and $s_i''(x_i) = s_{i+1}''(x_i)$.

6. With these information and some math we can achieve: Assuming $h = x_{i+1} - x_i$ is a constant, the univariate distance between consecutive point in the considered

time series, and $M_i = s_i''(x_i)$ is the second derivative of the *ith* segment.

$$\begin{aligned} a_i &= \frac{M_{i+1} - M_i}{6h} \\ b_i &= \frac{M_i}{2} \\ c_i &= \frac{y_{i+1} - y_i}{h} - \left(\frac{M_{i+1} + 2M_i}{6}\right) h \\ d_i &= y_i \end{aligned} \tag{3.28}$$

7. This can be rewritten in the form of a matrix, considering that $M_1 = M_n = 0$ for the aforementioned border condition to be satisfied.

$$\begin{bmatrix} 4 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix} \tag{3.29}$$

Each of the coefficients can be determined once each M_i is known, since h is a constant given directly by the x values of the data points. The result from Equation 3.29 is unique.

3.6.2.2 Akima

Akima spline is computationally faster than cubic spline, since it only uses the neighbourhood points.

The Akima spline interpolation is calculated very similarly to the natural cubic spline but with different restrictions by the following steps (Akima, 1970):

1. Instead of considering a segment between consecutive points, in Akima it is considered a set of 5 points for each of the given points: the two points before, the two points after and the considered point in the center in order to calculate the slope of the curve.

$$slope_i = \frac{|m_{i+1} - m_i|\, m_{i-1} + |m_{i-1} - m_{i-2}|\, m_i}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|} \tag{3.30}$$

Where i is the index of the considered point, and m_i is the slope $\frac{y_{i+1}-y_i}{x_{i+1}-x_i}$.

2. The third-degree polynomial between each pair of consecutive points is determined by the coordinates of the two points along with the slopes at each of them.

3. In iorder to calculate the slope at the border points, we must estimate 2 more points to extend the set for each end point.

4. So the cubic function is the unique cubic polynomial that satisfies $s(x_i) = y_i$, $s(x_{i+1}) = y_{i+1}$, $s'(x_i) = slope_i$ and $s'(x_{i+1}) = slope_{i+1}$

For the border points, since akima's spline need 2 adjacent points and at the border we do not have so, it is needed to estimate two points after the last one. Let the last point be (x_3, y_3), the previous points be (x_1, y_1) and (x_2, y_2) and the 2 more points we need to estimate (x_4, y_4) and (x_5, y_5).

- Suppose that these points lie on a curve described by: $y = a_0 + a_1(x - x_3) + a_2(x - x_3)^2$. Where a_i is constant.

- Suppose that $x_5 - x_3 = x_4 - x_2 = x_3 - x_1$, this way we can find x_4 and x_5, and consequently y_4 and y_5 substituting x_i in the second degree curve above.

Since Akima uses a local method it is computationally faster, but also fits less smoothly as in Figure 26 .

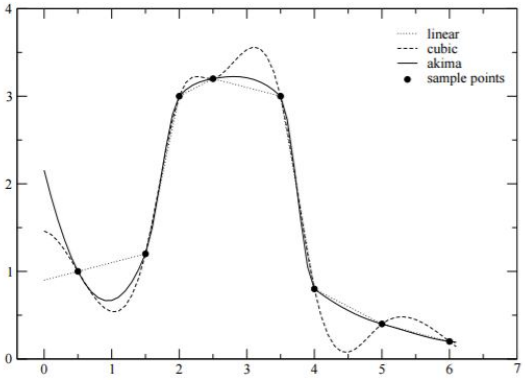


Figure 26: Cubic and Akima splines comparison. [Source: `https://laszukdawid.com/blog/2014/07/11/mode-mixing-in-emd/#ref4`]

PART II

METHOD

Reminding the target problem of this research and also the related work, the most successful one step ahead forecasting time series initiatives involve a context designed pre-processing for feature extraction and a careful model training over the extracted information. In order to improve the current state of the art, we decided to add exogenous features into consideration, this makes it necessary to evolve the model architecture as well as the pre-processing mechanism so that the exogenous information improve accuracy, instead of messing the convergence process during training.

This research takes two steps into widening state-of-the-art frontiers.

First step is to make exogenous data work within current successful models, allowing any number of exogenous time series to be fed into the target feature predictor model. This step will be denoted as the phase 1.

And the second step is to find a way to select best exogenous features by assessing different state-of-the-art relevance metrics, in order to prevent unnecessary and useless data to flow into parameter tuning and training. This step will be denoted as phase 2.

4 PHASE 1 METHOD: MULTIPLE INPUT ARCHITECTURE

The main goal for phase 1 is to find a way to consider multiple exogenous features as input for the single feature CEEMDAN-LSTM model, keeping a single target feature as output for prediction. In this pursuit, the data set and data preprocessing method is first defined.

Then, since a key factor that makes it more difficult to consider multiple input is the signal decomposition part, a new data flow architecture is proposed in order to handle the multiple new features obtained by the decomposition of each exogenous feature. Along with the data flow an optimal mixed prediction model architecture is proposed, which is necessary to handle features of different shapes along with prediction model limitations.

Since valuable explanatory information is added from exogenous features, the expected result is an improvement in accuracy when comparing the new proposed model X-CEEMDAN-LSTM to the original CEEMDAN-LSTM in forecasting the same data set and target feature.

4.1 Data

The considered market data source is public through `https://github.com/ranaroussi/yfinance` and focuses on the Brazilian stock market.

The time series features considered are:

1. **Open**: the stock price at the first successful trade in a day;
2. **High**: the greatest stock price traded during the whole trading day;
3. **Low**: the least stock price traded during the whole trading day;
4. **Close**: the stock price at the last successful trade in a day;

5. **Volume**: the total amount in shares of a certain stock traded during the whole trading day.

The features considered in the Yahoo Finance API are all the adjusted historical prices. Since corporate actions such as cash dividends, stock splits and reverse splits may cause an unpredictable discontinuity for the price time series, we take the adjusted curve into consideration. It is not of our interest to predict corporate actions neither their influence over the price.

The studied data ranges from the very beginning of 2018 to the end of 2022, all in a daily basis. That is: the time step between two consecutive points in the considered time series is a trading day, and in this case we make it univariate (we do not consider the weekends and holidays gaps). The considered stocks are the 10 top liquidity in target the period, the ones with highest amount of traded financial volume, in the Brazilian exchange B3.

Although the considered time step makes the time series interday, for the sake of standardizing: all times are considered in the time of Brasilia GMT-3.

The time ranges used are from the January 1st of 2018 until December 31st of 2022, for the following Brazilian equities: PETR4, VALE3, BOVA11, ITUB4, BBDC4, B3SA3, BBAS3, ABEV3, MGLU3, VVAR3.

4.1.1 Preprocessing

The raw data are given in price values, so it generally ranges from few Brazilian Reais to few hundred Reais. For the optimization method, backpropagation, on the neural model converge with less epochs and to make all of the different signals actually comparable we first make a Mathematical transformation on each signal, prior to feeding the model as in 4.1.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

The minmax adjust makes all time series signals values be always between 0 and 1. In 4.1 the x' is the new transformed value, x is the value of a data point in an intrinsic mode function, x_{max} is the greatest value among the whole series and x_{min} is the least.

4.1.2 Time windows

For each of the IMF series resulting from the CEEMDAN decomposition, a series of windows of sequential values is generated. An IMF series of length n yields a series with $n - \Delta t + 1$ windows, where Δt is the window length, and this series of windows is used as input. For example, if the original series is $[1, 2, 3, 4, 5, 6, 7]$ and the window size is 3, the resulting series of windows will be $[[1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6], [5, 6, 7]]$. Since the training method is supervised and the objective is to predict one step ahead in the time series, the corresponding labels for training this example data set would be $[1, 2, 3] -> 4$, $[2, 3, 4] -> 5$ and so on.

The window length Δt is treated as a hyperparameter and is adjusted according to the relative frequency of each series. For example, low-frequency series have time windows of greater length and high-frequency series have lesser window length because, as in Cao et al. (2019), the higher the frequency of a trend, the less dependency of long-past information, as well as the lower the frequency trend, the more dependency of long past information. The input tensor for the neural network is bi-dimensional of size $m \cdot \Delta t$, where m is the number of input features.

4.1.3 Data subsets

We divide the data into training and test sets after preprocessing, with the most recent 10% of data representing the test set, and the following most recent 20% of data representing the training data set. The training data set is used to allow hyperparameter tuning without affecting test results. This is represented in Figure 27.

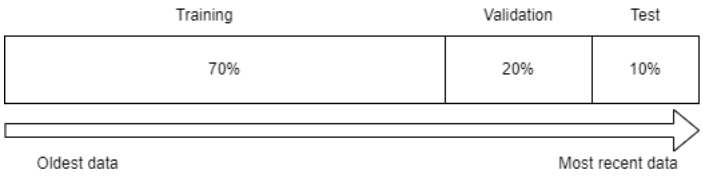
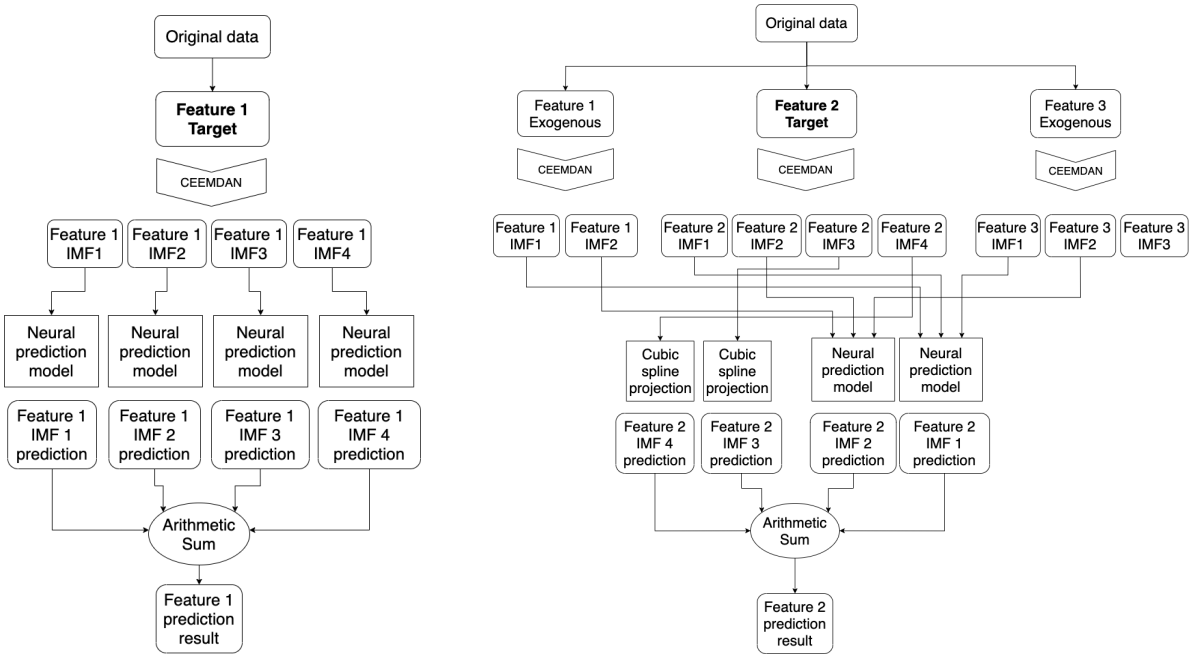


Figure 27: Data set separation.

The choice to test on most recent data is made upon the hypothesis that the most recent market behaviors are more likely to represent the short term future than the older market behavior, as stated in Cao et al. (2019). This choice does not contrast the common cyclic behavior since the considered time period is four times a year cycle. We are aware that this may hide some cyclic behavior of greater period.



((a)) Cao's data flow architecture.

((b)) Our proposed data flow architecture.

Figure 28: Cao's Architecture (a) deals with a single input feature and ours (b) deals with multiple input features. Both result in a single feature prediction.

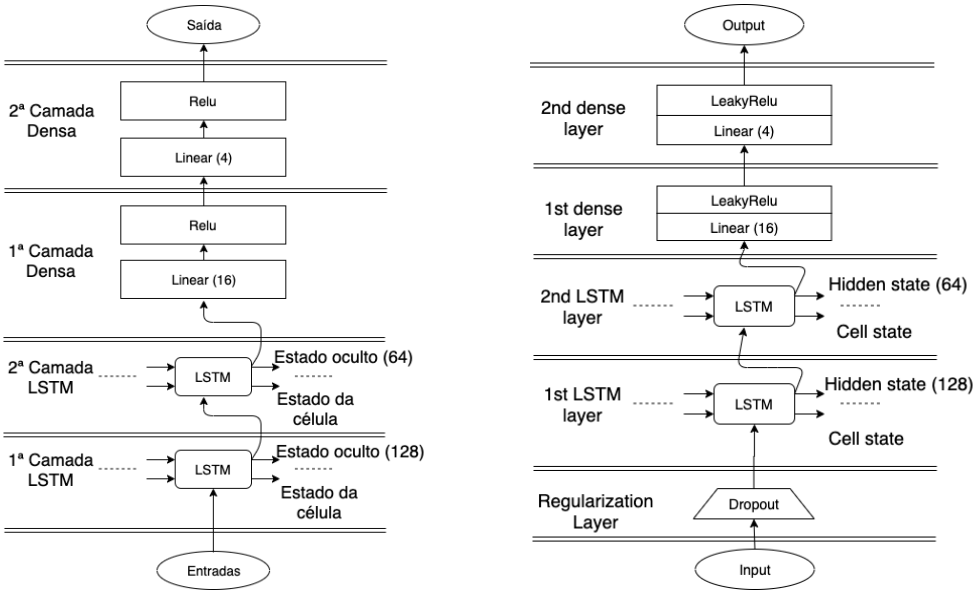
4.2 Model, Data flow architectures and training process

The phase 1 method focused on expanding Cao's model and method to handle exogenous inputs, expecting the model to become more accurate by introducing more data. And this was proved right, as long as we used the correct data flow architecture and the correct neural architecture, we were able to come up with better results when compared to the original single feature input and single feature output model of Cao.

We built a new data flow, as seen in Figure 28, taking as the starting point Cao's model (Cao et al., 2019) CEEMDAN-LSTM for single feature input to single feature output, in order to achieve multi-feature input to single feature output.

Our neural model is seen in the b item of Figure 29, on the other hand, Cao's original is seen in the same figure but in item a. Ours is almost the same as Cao's, except for the activation functions used in the dense layers and the presence of a dropout layer. Cao's neural architecture uses ReLu as activation function and ours LeakyRelu to deal with the data growth and provide more convergence stability for a general learning rate. We also have a dropout layer in the input, which we used to prevent the model from overfitting.

Both this model and this data flow architecture were proposed in BRACIS 2020



((a)) Cao's neural architecture. ((b)) Our proposed neural architecture.

Figure 29: Neural architecture comparison

conference, the greatest Brazilian conference on artificial intelligence (de Luca Avila and De Bona, 2020). At that time, the step further was to deal with the additional exogenous features and, by doing this with the architecture shown in Figure 28, we achieved an average improvement of 21% in accuracy when compared to Cao's single feature input.

4.2.1 Neural Vs. Spline prediction models

As we have described in Figure 28, our proposed architecture uses two different types of prediction models: the neural and the spline projection ones. This approach arose once the longest trends IMFs tend to be more flat than the high frequency IMFs, which are spiky with periods near to the time step magnitude. This makes the long trends a lot easier to predict than the high frequency components. So, instead of wasting computing power and time to train a neural network to predict a simple signal, we rather use the spline projection.

The spline projection was an option since the CEEMDAN decomposition is generated by applying the spline envelopes. So that the maxima and minima tend to approximate more each time the sifting process generates a new IMF, making the residue (the component with the longest period of all) an actual spline curve in the limit when the stop condition for CEEMDAN provides an infinite number of components.

The training of the model is unique for each of the IMF components after decom-

position and pre-processing, each of them undergoes a different hyperparameter tuning, which ends up in a different number of training epochs. The optimization algorithm and loss function are the most traditional ones: backpropagation with Adam optimizer and mean squared error (MSE) loss function.

4.2.2 Experiment definition

The experiment starts with a certain model architecture setup, defined model architectures are:

- Jian Cao's original CEEMDAN-LSTM (single input and single output)
- Proposed model X-CEEMDAN-LSTM with hybrid prediction models: splines for greater IMFs and neural model for lower IMFs (multiple input and single output)
- Proposed model X-CEEMDAN-LSTM with only neural models (multiple input and single output)
- Proposed model X-CEEMDAN-LSTM with only neural models (multiple input and single output) with Time Warping input selection.

After the experiment's model definition, the next step is to set the parameters and hyperparameters. The initial parameters for each of the neuron's weights are set at zero and the very initial hyperparameters are taken from Cao's work and tuned for lowering the time spent on training phase and maximize accuracy.

Once the hyperparameter tuning phase is done, the experiment consists on decomposing, training the model and predicting the test data 10 times for each of the equities considered. It is important to reproduce the experiment several times because of the random nature of CEEMDAN decomposition, causing the same full prediction process to result in different accuracies.

4.2.3 Experiment evaluation

The metrics taken as standards to evaluate the model's accuracies are MAPE and MSE.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i} \tag{4.2}$$

$$MSE(y) = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n} \tag{4.3}$$

These specific metrics were chosen for two main reasons: MSE is the most commonly used metric for training and convergent the model; MAPE is great for comparing the final results, since the magnitude of the resulting signal may vary depending on the price of each stock. High-priced stocks usually vary more than low-priced ones, thus the MSE metric alone would not be enough to compare accuracies among different stocks.

Since initial neural net weights are random and the CEEMDAN decomposition also uses a white noise component, multiple repetitions of the exact same experiment can lead to slightly different results. This in turn is the main reason for repeating the experiment several times and picking an average outcome for each series of experiments.

In this way, each experiment is repeated 10 times and the MSE and MAPE are averaged, as well as the standard deviation is measured to see whether the random part of the method is leading to unpredictable results.

Another important part of the method evaluation is to use different stocks of different natures and different behaviors. The results for each of the chosen stocks are then averaged using MAPE to check whether the model is capable of handling signals of great abundance.

5 PHASE 2 METHOD: DATA SET EXPANSION, ABLATION TEST AND FEATURE SELECTION

The main goal for phase 2 is to establish a general feature selection filter-based method, based on the assessment of different relevance metrics, that leads to the best accuracies across all four different data sets using the same X-CEEMDAN-LSTM data flow and training process setup as in phase 1. Bear in mind that the X-CEEMDAN-LSTM is designed to predict one-step ahead time series, and the feature selection aims to optimize accuracy and computational cost towards the optimal accuracy.

For the feature selection method to be generic in the scope of this work, it needs to be validated against all 4 datasets and provide the best feature combinations without the need to train and evaluate the model with all possible feature combinations. But for finding the best selection method it is necessary to actually know what the best feature combinations are and try to somehow relate the relevance metrics to them. For this reason, the X-CEEMDAN-LSTM model will be trained and evaluated using the four data sets with all possible combinations of exogenous features, always providing the target feature as input; this process will be called the exogenous feature ablation test.

Since there are four considered datasets and each of them have 4 different exogenous features, the total number of experiments is the size of the power set of four times 4, times 2 because we also test the vanilla LSTM without CEEMDAN for benchmarking, we do not test the model with splines for simplicity), which yields 128 experiments. Each experiment is repeated five times with different random seeds for noise and initial weights, and the results shown are given by the average. So, the actual total number of experiments is 640 experiments.

Depending on the dataset, each experiment takes from 5 minutes up to 1.5 hours, so the total expected time for running an entire set of experiments ranges from about 50 hours up to 40 days. During the development phase, this process might need to be done a few times. Thus, a parallel way of experiments is needed at scale. Since there

are large datasets the RAM memory can be a bottleneck when parallelizing, as well as the computational processing power if there are too many processes in the same machine, these limits lead to the need of a multi-machine scalable time series data processing and training framework, preferably in a cloud environment for launching preprocessing and training process instances.

5.1 Data

The data sets were majorly expanded both in size lengths and in getting datasets from other domains when comparing to phase 1. The finance data set was greatly expanded in length and kept both focusing on PETR4 as a liquid name example and Solutions Data Services as a source (`https://dataservices.btgpactualsolutions.com/`) to extract interday candles data from the Brazilian exchange stocks, since Yahoo Finance, the source used in phase 1, showed inconsistencies across this research when handling corporate events in its time series.

Besides the financial domain, three new datasets were also tested, among them one from solar battery energy load domain and another one from Switzerland wind quality data, both acquired in partnership with University of Lucerne, and the last one from transformer (the electrical equipment) load, which is public and widely used in the literature. Figure 30 shows a wide view of the phase 2 datasets for a rough visual comparison, as well as Table 2 shows the basic statistical description of the four datasets. The datasets behave very differently, being the financial one the only one without a potential stationary long-term shape because of its almost monotonic increase across time in some features, the other ones are mostly seasonal and have a stationary shape, all of them varying a lot in magnitude and deviation, although the magnitude differences do not appear in Figure 30 and Table 2 because they were taken after min-max scaling.

The exogenous features and the target feature were chosen based on rough knowledge about each dataset domain. Each data set had its own number of exogenous variables, but for simplicity and standardization, only four exogenous features were chosen for evaluation for each data set.

The ETTh1 data set monitors an electricity transformer from a region of a province of China including oil temperature and variants of load from July 2016 to July 2018 at an hourly frequency. The aim with this dataset was to predict the oil temperature (OT) inside the transformer equipment based on its own history but also based on the high

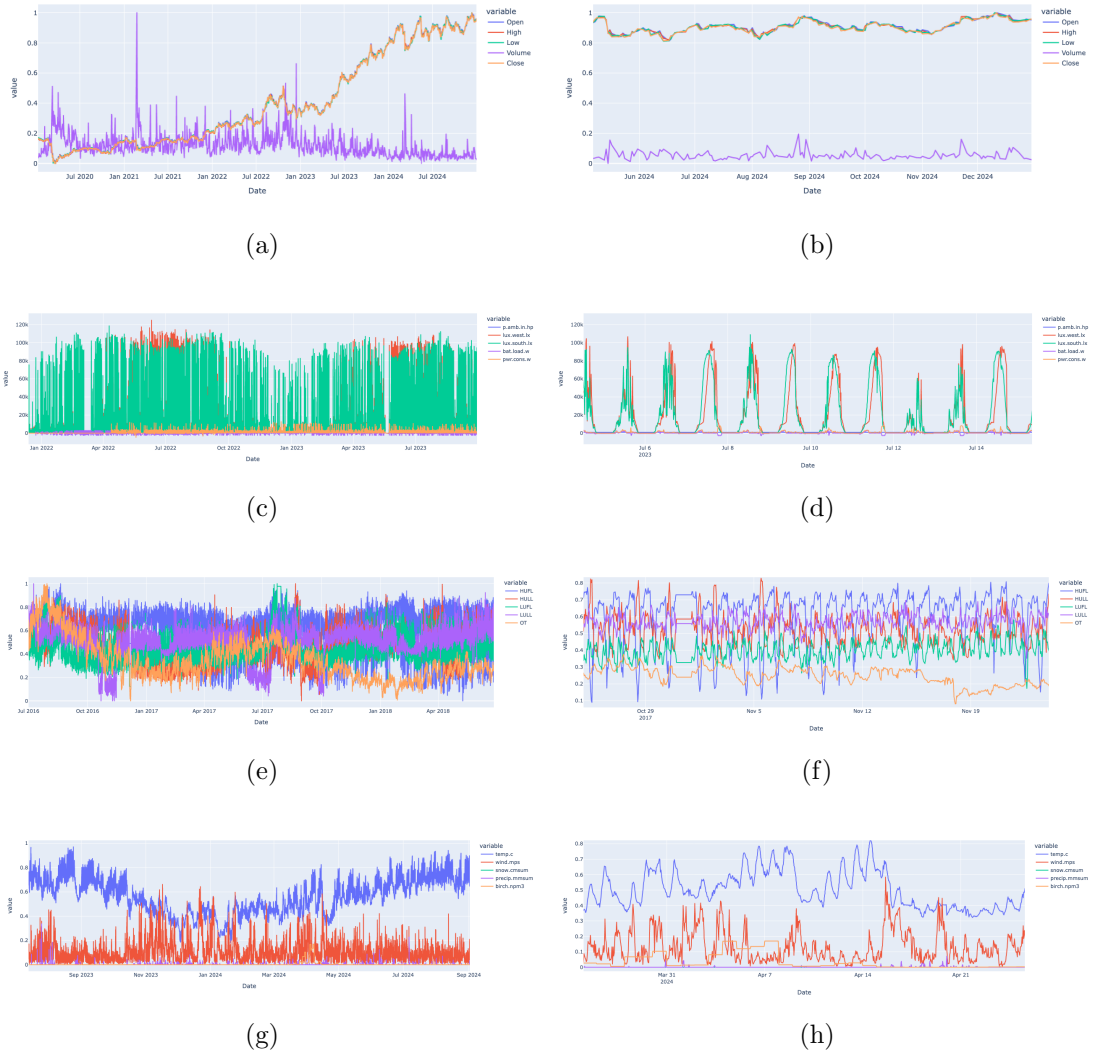


Figure 30: Raw datasets graphs from phase 2 after min-max scaling: (a) Finance macro; (b) Finance example window magnified; (c) Energy load macro; (d) Energy load example window magnified; (e) Wind macro; (f) Wind example window magnified;

Dataset	Feature	count	mean	std	25%	50%	75%
ETTh1	HUFL	17420	0,648999821	0,15248639	0,615598715	0,679180164	0,744207133
	HULL	17420	0,470628264	0,137346487	0,369401473	0,468459981	0,567585733
	LUFL	17420	0,439196976	0,12022568	0,361655996	0,415135233	0,496902737
	LULL	17420	0,504399404	0,135737338	0,462078343	0,531129736	0,586144459
	OT	17420	0,347488801	0,171041315	0,22049634	0,308982368	0,442410215
Finance	Open	1235	0,402996355	0,300828275	0,151364036	0,304781461	0,689938398
	High	1235	0,399426273	0,302813265	0,144258934	0,301113064	0,689806678
	Low	1235	0,4001806	0,299891656	0,15070713	0,301119623	0,690041249
	Volume	1235	0,120736317	0,07863882	0,067283323	0,103624436	0,152297046
	Close	1235	0,398224214	0,300101452	0,146138092	0,301345816	0,686220012
Energy Load	p.amb.in.hp	62840	0,96493257	0,120143187	0,97593361	0,980082988	0,98340249
	lux.west.lx	62840	0,088791608	0,174108031	8,30302E-06	0,001754555	0,085924442
	lux.south.lx	62840	0,113519454	0,214491993	8,73731E-06	0,001635784	0,094892034
	bat.load.w	62840	0,470307745	0,132064608	0,407197597	0,46283166	0,463029001
	pwr.cons.w	62840	0,316130044	0,076011305	0,277703789	0,296268886	0,316738569
Wind	temp.c	303912	0,513150283	0,151829729	0,396551724	0,511494253	0,624521073
	wind.mps	303912	0,10842024	0,088340334	0,046511628	0,085271318	0,142118863
	snow.cmsum	303912	0,000141891	0,006285507	0	0	0
	precip.mmsum	303912	0,001138684	0,008136447	0	0	0
	birch.npm3	303912	0,005417341	0,035196146	0	0	0

Table 2: Raw datasets basic statistical description after min-max scaling.

useful load, high useless load, low useful load and low useless load (HUFL, HULL, LUFL and LULL).

The financial data set shows the daily candle aggregates from PETR4 stock name trading data from January 2020 to October 2024. The aim is to predict the closing price at the end of the day based on its own history and the exogenous additional data of opening, high, and low price values, as well as the trading volume.

The energy load data set monitors the power of a house solar battery in Switzerland from January 2022 to December 2023 in 5-minute periods. The goal of this data set is to predict the power consumption (pwr.cons.w) of the battery based on its own history and exogenous features such as solar illumination from different directions (lux.west.lx and lux.south.lx), ambient pressure (p.ambient.in.hp), and battery load (bat.load.w) as the battery is used. This data set is the only one out of the four that contains missing data; each missing data point is replaced with zero values for all variables.

The wind data set covers wind quality and weather metrics from Meteosuisse, the Swiss meteorological station. The data set used was a time-sampled one because its size was too large for a computationally reasonable preprocessing and training time; despite the fact that the original data set covers data in 1-minute time periods, the sampled data set covers data in 5-minute periods, only considering the most recent value for sampling. The goal is to predict the birch pollen in the air (birch.npm3), because it is related to allergic respiratory symptoms, based on the average temperature (temp.c), wind speed (wind.mps), snow conditions (snow.cmsum) and rain conditions (precip.mmsum).

All data sets are made publicly available for research reproduction purposes in the repository `https://github.com/avilarenan/FARM_LSTM.git` on Github.

5.2 Experimenting framework

This framework is a collateral result of phase 2 that is made publicly available at `https://github.com/avilarenan/PyHeta` for others to enable scale in time series experiments. The framework architecture is shown in Figure 33, it has an embedded UI WEB screen also partially shown in Figure 34 that works for uploading new raw datasets, setting parameters, and launching new data processing as well as training processing (parts 1, 2 and 9 of Figure 33).

The core of the framework relies on two processor pieces of code, the data processor and the train processor, both must obey a step-based code design and are represented by two Python files and the example code structure can be seen in Figure 31. Each step has its inputs and produces its outputs, which are then chained to next step inside the same processor code. Once data processing is finished, its output is saved in a data base such as PostgreSQL and is ready to be used in the next training process part.

Each processor application is triggered by an instruction received from a message queue application, such as RabbitMQ. Once started, each processor will begin its own function pipelining at code level. According to Figure 31. Starting by the data processor the functions are the following:

- **Simple Data Processing** scaling, splitting and windowing.
- **Signal generation or decomposition** new feature generation based on raw dataset or feature decomposition (i.g. CEEMDAN)
- **Relevance metrics calculations** Relevance measures are able to be done both in raw data (default) and in scaled or decomposed data.
- **Final data aggregation** Coupling all processed features in a single input data frame and in a single expected output data frame.

In the subsequent step, the training processor takes action, with the following steps at code level as in Figure 31.

- **Training** Model training throughout epochs, undergoing auto-learning rate tuning and early stopping, other hyperparameters are fixed for all datasets.

- **Evaluation** Model inference with accuracy metrics.
- **Interpretability** Optional step for analyzing model behavior based on inputs.
- **Recomposition & descaling** Final step for aggregating previous steps and generating final accuracy metrics.

The data processor saves its output to a relational DB such as PostgreSQL and

Both processors are run based on Python processes under a worker manager Python application and produce logs which are streamlined to an S3 for debugging purposes, this enables multiprocessing parallelization and multi-machine parallelization in a cloud environment. For example, the experiments related to phase 2 of this work were all run using 12 m7i.xlarge machines from AWS at top scale, which took about a week to produce the final results.

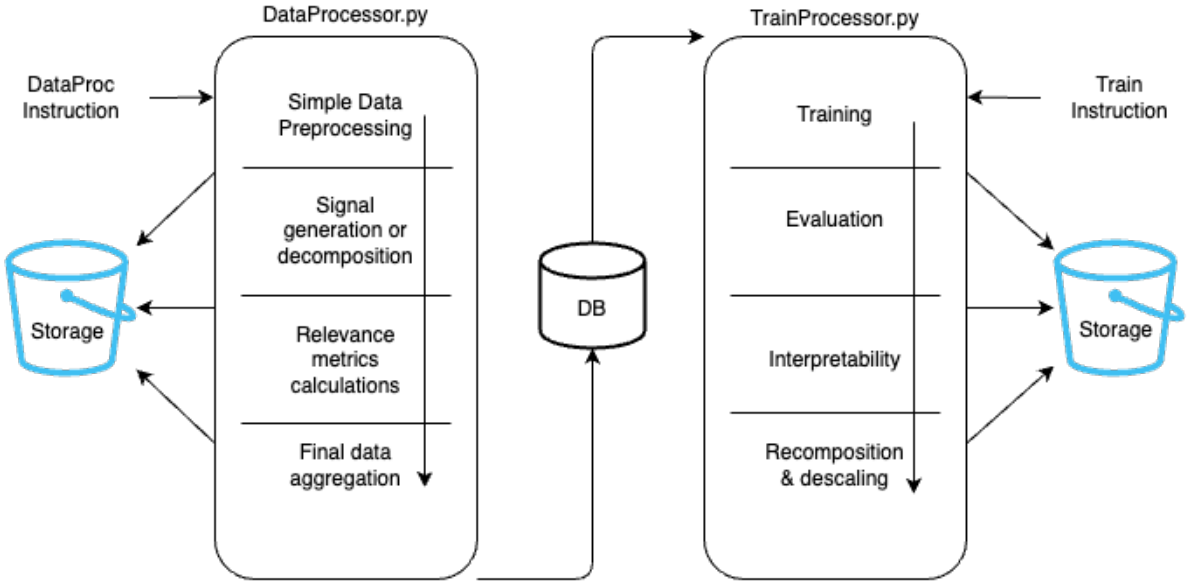


Figure 31: Developed Scalable data processing and training Python Application Framework code structure example

5.3 Experiment definition

The model used is X-CEEMDAN-LSTM, the same developed in phase 1, described in Figure 28 item b. It is important to mention that this model uses the same IMF levels for feeding each IMF LSTM predictor as input because it was shown in phase two that this data flow provides better accuracy than the one with all components fed to the model as input.

An experiment consists of two steps: data processing and model training. The data processing undergoes min-max scaling, then an optional decomposition, and afterwards relevance metrics are calculated, and everything is joined and prepared in a final data set for training input. This data processing step must be performed only once for each combination of data pre-processing parameters; these parameters are described in the table 3.

In this phase an ablation study will be conducted in order to evaluate feature importance and relevance metric's scores. The ablation will test the combinations of the power set of four domain-selected exogenous features as input, with and without CEEMDAN decomposition and all four data sets.

The Dataset, target_feature, SKIP_CEEMDAN and ablation_features hyperparameters are needed for the data processing step, all the rest is necessary for the training phase. Since the data processing and training phases are separate from each other, it is possible for one to reuse the same processed dataset with a different set of hyperparameters for the training phase several times, eliminating unnecessary duplicated computational cost.

For the training phase, the hyperparameters n_epochs_by_imf, n_epochs_stop_by_imf, model_by_imf and window vary according to the IMF component level that is currently predicted. A single experiment, when using the CEEMDAN decomposition, predicts each of the components separately and recomposes them at the end. And each component has a different behavior over time, for example, the component series became smoother as the IMF level grows, up to the residue, which has a non-stationary behavior, different from all other components. For this reason, the hyperparameters are specialized for them. Specifically for the model_by_imf hyperparameter the MLP model is chosen for the residue because of this non-stationary behavior that makes it difficult to predict by an LSTM.

Hyper Parameter	Value	Type	Meaning
DATASET	Variable according to ablation instance	String	Dataset to use in the experiment
target_feature	Variable according to ablation instance	String	Target time series to predict as output
SKIP_CEEMDAN	Variable according to ablation instance	Boolean	Whether to decompose each of the input series with CEEMDAN
ablation_features	Variable according to ablation instance	List of Strings	Subset of exogenous features
train_frac	0.7	Float	Fraction to split the train and test dataset in chronological order
windows	IMF0:2, IMF1:3, IMF2:4, IMF3:4, Residue:6	Int	Window length with which to prepare inputs for neural network, its value depends on the component level
learning_rate	0.001	Float	Initial learning rate for the optimization algorithm
optim_algorithm	adam	String	Optimization algorithm, which is always Adam
batch_size	64	Int	Batch size for training the neural networks
n_epochs_by_imf	IMF0:1000, IMF1:1000, IMF2:1000, IMF3:500, IMF4:500, Residue:10000	Int	Maximum number of epochs to use during training, its value depends on the component level
n_epochs_stop_by_imf	IMF0:5, IMF1:10, IMF2:10, IMF3:10, IMF4:10, Residue:50	Int	Maximum number of strike epochs before early stopping
model_by_imf	IMF0:LSTM, IMF1:LSTM, IMF2:LSTM, IMF3:LSTM, IMF4:LSTM, Residue:MLP	String	Neural network model to use for prediction. its value depends on the component level
min_delta_stop	0	Float	Tolerance value to consider a strike epoch in the early stopping algorithm
repetitions	5	Int	Number of repetitions to run the same experiment

Table 3: Experiment hyperparameters.

Next, the training process takes into account the optimization iteration of the model through epochs, when it happens to be a decomposed dataset from CEEMDAN, a different model is trained for each of the components, just as in Figure 28 item b. The evaluation is then made component-wise, as well as an optional interpretability extraction. The final step is to recompose each component result by arithmetically summing them in case of CEEMDAN decomposed data set, after all descaling is needed and is done with the same scaling parameters used in the data processing step. The processed data sets can be reused.

Since both the data processing step and the training step have initialization randomness involved: for example, the random seed of the white noise used in CEEMDAN decomposition or even the random seed in the weight initialization of the neural networks, each pair of data processing and model training is repeated 5 times, and the results shown across this work are an average of them.

The final result of the experiment is then registered in a result data base which is then used for comparison and analysis relating the training hyperparameters and data set used.

The loss function for training neural networks is RMSE, and the hyperparameters for the experiments are summarized in the table 3.

5.4 Feature selection analysis

As mentioned in the base theory of this work, there are three main approaches to feature selection in time series: filter, wrapper, and embedded.

The filter is based on statistical tests that can be taken prior to model training or evaluation and are typically used with a certain threshold above or below which features are discarded.

Wrapper is based on testing the model with several combinations of features and evaluating its accuracy, since the number of possible combinations can grow exponentially, there are incremental step techniques that allow finding the potentially best combination of features without spending too much on computational resources. Although it can assess the exact accuracy and provide the best combinations of input features, the computation cost and the model-specific dependency make it a niche choice.

The embedded method relies on building elements that select the most relevant fea-

tures during the training process of a given machine learning model. Lasso regression and decision trees algorithms, as well as attention-based neural networks, can be examples of embedded methods of feature selection. The drawback here is mainly the lack of interpretability as the number of features increases.

For this work, the focus will be mainly on the filter, based on metrics calculations and selection before running the model, and consequently being a model-agnostic method. However, we will also explore wrapper methods by analyzing improvements in accuracy for given models.

The relevance metrics explained in the base theory are calculated for each of the exogenous features against the target feature, according to each different data set, shown in Table 18. In addition, the effect of the decomposition of CEEMDAN combined with exogenous series will be further evaluated.

Feature selection rule based on relevance metrics

- **FARM_global**: selects the feature with the maximum FARM_global.
- **Correlation**: selects the feature with the highest correlation.
- **FastDTW**: selects the feature with the minimum FastDTW distance.
- **FastDDTW**: selects the feature with the minimum FastDDTW distance.
- **Euclidean**: selects the feature with the minimum Euclidean distance.

The selected relevance metrics are the state-of-the-art ones: correlation, DTW, DDTW and their Fast versions, the baseline euclidean distance and the FARM global metric; the last one aims to fix alignment issues that happen with DTW as explained in the base theory of this work. The calculated relevance metrics are shown in Table 18. The only modification on the metrics values will be to take the absolute value of correlation, because we are interested in both negative or positive correlations, since neural networks can abstract the absolute value and may improve accuracy when using an exogenous feature in both cases.

The approaches for analyzing relevance metrics for single and multiple exogenous features will be taken as follows, the relevance metrics will be measured using raw data

sets relating the target and the exogenous feature.

- Single exogenous - Top score: the top accuracy results will be matched against best selected features for each relevance metric and a top relevance metric score will be calculated. The metrics will be ranked by the most feature selection success ratio.
- Single exogenous - Improvement score: an improvement score will be calculated for each metric based on how close the improvement for each feature of a metric is to the actual accuracy improvement when adding that feature to the input.
- Multi exogenous - Top score: the top accuracy results will be matched against the best selected groups of features for each relevance metric, and a top relevance metric score will be calculated based on how many times the metric selects a feature that is present in the input among the top accuracy experiments.
- Multi exogenous - Improvement score: an improvement score will be calculated based on how close the respective metric improvement for each feature in all experiments that contain that feature is to the actual average accuracy improvement for each feature addition in all experiments that contain that feature.

The main goal of this phase 2 work is to find a rule-based filter feature selection method that is able to achieve the best accuracy for most of the data sets tested prior to model training. This will be done by comparing the relevance metrics of table 18 and 19 with the results of the ablation test for all combinations of 4 exogenous characteristics for each of the 4 datasets present in Tables 14, 17, 15 and 16, calculating the top and improvement scores described above. This will allow the analysis of what are the best relevance metrics to use in the rule-based filter.

5.4.1 Relevance metric top score

The metric top score is proposed and aims to show how good a relevance metric is in the task of predicting what features one should use in order to achieve the best accuracy results. In other words, the top score metric shows how likely a feature with high relevance according to a metric is to show up in the actual best accuracy results.

5.4.1.1 Natural language definition

The process for calculating the top score for multiple exogenous features is the following. The single exogenous feature case is a special case from the definition.

1. Do the ablation tests for every feature combination for all data sets.
2. Select the top N accuracy results using up to K exogenous features as input for each data set.
3. Calculate relevance metrics for each pair of target and exogenous feature for each relevance metric.
4. Select the top features according to each relevance metric.
5. The metric top score will be calculated as the sum of the number of times a top feature selected by the relevance metric appears in the set of input features for the selected top N accuracy results using up to K exogenous features divided by n with n varying from 1 to N, for each data set.

5.4.1.2 Mathematical definition

Let D be the set of datasets, indexed by d, where $d \in D$. Let F be the set of all exogenous features, indexed by f, where $f \in F$. Let K be the number of exogenous features used in a certain experiment.

Define the accuracy function of a feature subset $S \subseteq F$ with $|S| = K$, $0 < K < |F|$, on dataset d as:

$$\mathrm{A}(S, d)$$

This function returns the model accuracy when using the subset S as input features on dataset d.

The ablation test consists of computing $\mathrm{A}(S, d)$ for all possible feature subsets $S \subseteq F$ with $|S| = K$, $0 < K < |F|$, across all datasets $d \in D$.

For each dataset d, let $\mathcal{S}_d^*$ be the set of the top N feature subsets achieving the highest accuracy:

$$\mathcal{S}_d^* = \{S_1, S_2, \ldots, S_N\} \quad \text{where} \quad \mathrm{A}(S_i, d) \text{ is among the top } N \text{ for dataset } d$$

where each S_i satisfies $|S_i| \leq K$.

Let M be the set of relevance metrics, indexed by m. Each metric m assigns a relevance score between a feature f and the target variable in dataset d:

$$R_m(f, d)$$

where $R_m(f, d)$ quantifies the relevance of feature f to the target variable in dataset d.

For each relevance metric m, we define the top-ranked features in dataset d as:

$$F_{d,m}^{\dagger} = \{f \in F \mid R_m(f, d) \text{ is the P best in dataset } d\}$$

where $F_{d,m}^{\dagger}$ contains the most relevant features according to metric m for dataset d and P is the selected number of considered most relevant features according to the previously described feature selection rule. For the sake of simplification, P will be chosen as 1 for this work's results.

Define an indicator function:

$$\mathbb{I}(f \in S_i)$$

which equals 1 if feature f is part of the feature subset S_i and 0 otherwise.

For a given relevance metric m, the metric top score is computed as:

$$\text{TopScore}_m = \frac{1}{|D|} \sum_{d \in D} \sum_{n=1}^{N} \sum_{f \in F_{d,m}^{\dagger}} \mathbb{I}(f \in S_n)$$

where:

- $|D|$ is the total number of datasets, - The inner sum iterates over the top N feature subsets for each dataset, - The summation over f counts how many times features selected by relevance metric m appear in the selected top N accuracy results, - The division by n normalizes the score for varying n from 1 to N.

Thus, TopScore_m represents the average frequency with which features ranked highly by metric m also appear in the best-performing feature subsets across datasets.

5.4.2 Relevance metric improvement score

The relevance improvement score is proposed and its motivation is to see how good a relevance metric is in the task of predicting how much relative accuracy improvement there will be when a certain exogenous feature is added, compared to the worst accuracy result.

5.4.2.1 Natural language definition

The process for calculating the improvement score for multiple exogenous features is the following. The single feature case is a special case.

1. Do the ablation tests for every feature combination for all data sets.
2. Select all the accuracy results for each data set using N exogenous feature as input where a certain exogenous feature is present. Normalize accuracy values between 0 and 100 for each data set. Calculate the average improvement in accuracy related to the presence of a feature in the input compared to the worst accuracy result among the selected ones by summing the accuracy improvement of each experiment where the feature is in the input set and dividing by the number of different experiments.
3. Calculate relevance metrics for each pair of target and exogenous feature for each relevance metric. Normalize all metrics between 0 and 100 for each data set. This will lead us to the metric improvement for each feature.
4. Evaluate the difference between feature average improvement in accuracy and the feature metric improvement for each metric and each feature. Take the absolute values of these differences. Sum the absolute differences together and divide by the number of features. This will lead us to an intermediate score which is between 0 and 1 and the closer to zero the better the metric. Take the absolute difference between this intermediate metric and one, in order to invert the behavior, ending up with: the highet the score the better the metric, because the least the error in predicting the accuracy improvement for a given feature.

5.4.2.2 Mathematical definition

Perform ablation tests for every combination of exogenous features $S \in 2^F$, training the model $\mathcal{M}(S, T)$ on each dataset D_i, where D_i represents a different dataset. For each

experiment, evaluate the model's accuracy $A(S, D_i)$, where $A(S, D_i)$ denotes the accuracy of the model when trained with the feature set S on dataset D_i.

Normalize the accuracy scores $A(S, D_i)$ for each dataset D_i such that the accuracy values are scaled between 0 and 100:

$$A_{\text{norm}}(S, D_i) = 100 \cdot \frac{A(S, D_i) - A_{\min}(D_i)}{A_{\max}(D_i) - A_{\min}(D_i)}$$

where $A_{\min}(D_i)$ and $A_{\max}(D_i)$ are the minimum and maximum accuracy values, respectively, across all feature combinations in D_i.

For each feature f_i, calculate the average improvement in accuracy across all experiments where f_i is present in the feature set. This can be done by summing the normalized accuracy improvements of all experiments with f_i in the input and then dividing by the number of such experiments:

$$\text{AvgImprovement}(f_i) = \frac{1}{|S_i|} \sum_{S \in S_i} (A_{\text{norm}}(S, D_i) - A_{\text{norm}}(S \setminus \{f_i\}, D_i))$$

where S_i is the set of all experiments that include f_i.

For each pair of exogenous feature f_i and target T, compute a relevance metric $R_m(f_i, T, D_i)$ for each metric $m \in M$, where M is the set of relevance metrics. Normalize these metrics to be between 0 and 100 for each dataset D_i and **reverse the distance-based relevance metrics** values by subtracting them from 100, since the best distance-based relevance metrics are the lowest. This gives you a standardized metric improvement, the higher the better, for each feature f_i, as follows:

$$R_{m,\text{norm}}(f_i, T, D_i) = 100 \cdot \frac{R_m(f_i, T, D_i) - R_{m,\min}(D_i)}{R_{m,\max}(D_i) - R_{m,\min}(D_i)}$$

where $R_{m,\min}(D_i)$ and $R_{m,\max}(D_i)$ represent the minimum and maximum values of relevance metrics $R_m(f_i, T, D_i)$ across all features for each dataset D_i.

For each exogenous feature f_i, evaluate the difference between the feature's average accuracy improvement and its relevance metric improvement. This will be done for each metric $m \in M$ as follows:

$$D_m(f_i) = |\text{AvgImprovement}(f_i) - R_{m,\text{norm}}(f_i, T, D_i)|$$

For each metric $m \in M$, sum the absolute differences across all features f_i, quantifying

the average error in predicting the accuracy improvement of a certain metric:

$$E_m = \frac{1}{n} \sum_{f_i \in E} D_m(f_i)$$

where n is the total number of exogenous features $|F|$.

The value E_m gives us an intermediate score for each metric m, which reflects how bad the metric corresponds to the observed accuracy improvement for each feature. The lower the score, the better the metric in predicting the accuracy improvement, since it means the error decreases. To invert the behavior, we compute the final score for each metric by subtracting S_m from 1:

$$\text{ImprovementScore}_m = 1 - E_m$$

The final ranking of metrics $m \in M$ is based on the final score ImprovmentScore_m. The metric with the highest $\text{ImprovementScore}_m$ is considered the best for determining the relevance of exogenous features to predict the target time series T.

PART III

RESULTS

6 PHASE 1 RESULTS

The first results of this research were published on BRACIS 2020 (de Luca Avila and De Bona, 2020), these results explored how one could add the capability to deal with and learn from exogenous features when talking about the CEEMDAN-LSTM model, which was limited to a single feature input and a single feature output.

At that point we were able to create and propose a new architecture that used mixed models, for the components with high frequency trends, we used neural models for prediction, and for the long trends components we used spline projection. One of the conclusion points analyzed how the threshold for model switching behaved for the top ten liquidity Brazilian stocks, and we found an optimum threshold at the IMF2, including the edge. Each neural prediction model was fed with the IMF2 of each input signal, just by reasoning that the same level IMFs across the exogenous signals should have similar frequencies and thus the model would be able to capture similar patterns easier, but with no testing using other signals neither broad justification.

The resulting overall accuracy for the final signal prediction, which was still the closing price, but now with open, high, low, close and volume as inputs, led to an improvement of 27.234% in the average MAPE accuracy in the top ten Brazilian tickers, when compared to the original CEEMDAN-LSTM model. The details can be found in the original article reference (de Luca Avila and De Bona, 2020).

6.1 Training phase

For this result to be achieved we had to go through hyperparameter calibration, which is shown in Table 4 for the training process. That research used two different prediction models, the LSTM model and the spline projection model, and we also analyzed how each of the IMFs behaved when predicted by each of the prediction models. Our hypothesis was that neural models should do better in high-frequency components, such as IMF0 and the near, and on the other hand, the spline projection models should do better when

dealing with long-term trends, such as the residue and the like. But the main concern when dealing with multiple prediction models is to find the optimal component level threshold that minimizes the final prediction error. And this was also done; it is shown in Figure 32.

Table 4: Hyperparameters

Target feature (close) IMF		1	2	3
CEEMDAN	Noise Scale	0.15	0.15	0.15
	Learning Rate	2e-4	8e-4	8e-4
	Max. Epochs	2500	2000	1500
	Window size	2	2	3
	Patience	20	20	20
	Dropout probability	0.2	0.2	0.2

6.2 Optimal threshold

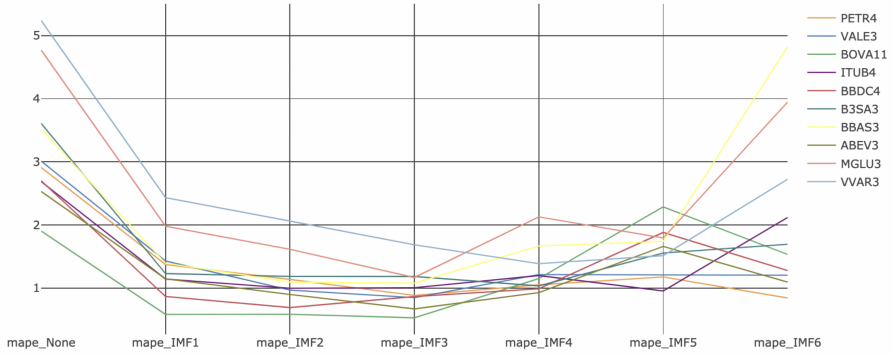


Figure 32: MAPE accuracy by tested IMF threshold. For example: in x axis the mape_IMF3 means that the tested models uses neural prediction models for the IMFs up to 3 including the edge, and the other IMFs are predicted with spline projection.

The results shown in Figure 32 show that the optimal MAPE may be around IMF2 or IMF3. So we investigated all the 4 variations of the model for these 2 thresholds. The model variations are the following: LSTM-CEEMDAN-SPLINE, XLSTM-CEEMDAN-SPLINE, CEEMDAN-LSTM and X-CEEMDAN-LSTM. The X addition before CEEMDAN stands for Exogenous, the SPLINE addition after the LSTM means that we use spline projections to predict greater level IMFs (long trends).

6.3 Experiments

The results are shown in Table 5, with the best values in bold.

Testing these two thresholds in IMF3 and IMF2 in Table 5, we repeated each experiment since decomposition, training up to the actual prediction 10 times for each of the tickers and each of the model variations. The results show the average and the std in these 10 experiments. The results are not always the same because CEEMDAN adds random noise during the sifting process, so each decomposition may vary a little.

6.4 Result summary

The results are summarized and averaged across the tickers in Table 6, and the values refer to the percentage of improvement (how much the average MSE and MAPE has decreased or how much the average standard MSE and MAPE has decreased). This table shows that the best accuracy improvement result in MAPE was to use the IMF2 threshold with exogenous features and spline projections for the higher level IMFs, giving 27.234% improvement.

Although the IMF3 threshold has given a greater improvement in MSE, since we are comparing different tickers with different magnitude absolute price values, the MAPE metric is more adequate for measuring improvement, and so the relevant advance comes with IMF2, given the higher MAPE enhancement.

Table 5: Comparison results

	Threshold at IMF3				**Threshold at IMF2**			
	MSE		**MAPE**		**MSE**		**MAPE**	
Metric	**AVG**	**STD**	**AVG**	**STD**	**AVG**	**STD**	**AVG**	**STD**
Symbol	**LSTM-CEEMDAN-SPLINE**							
PETR4	0.140	**0.006**	**0.930**	**0.055**	0.155	0.015	1.063	0.069
VALE3	**0.347**	0.101	0.996	**0.134**	0.340	0.125	1.098	0.085
BOVA11	**0.302**	**0.046**	0.653	**0.048**	0.331	0.095	0.672	0.038
ITUB4	**0.051**	**0.007**	1.047	**0.038**	0.107	0.019	1.085	0.083
BBDC4	**0.093**	0.015	0.887	0.088	0.151	0.018	0.933	**0.075**
B3SA3	**0.272**	**0.015**	1.217	**0.048**	0.337	0.042	1.343	0.085
BBAS3	0.178	0.048	1.184	0.083	0.222	0.043	1.271	**0.048**
ABEV3	**0.051**	0.006	**0.678**	0.077	0.070	**0.005**	0.928	**0.037**
MGLU3	**0.628**	0.058	1.464	0.065	0.942	**0.040**	1.963	0.062
VVAR3	**0.010**	**0.001**	1.948	**0.076**	0.019	0.003	2.107	0.159
Symbol	**XLSTM-CEEMDAN-SPLINE**							
PETR4	**0.122**	0.024	0.957	0.109	0.103	0.006	0.972	0.030
VALE3	0.356	**0.099**	**0.995**	0.193	0.297	0.121	1.056	0.083
BOVA11	0.305	0.101	**0.572**	0.064	0.311	0.077	0.610	0.032
ITUB4	0.067	0.019	1.040	0.058	0.098	0.010	**0.969**	**0.032**
BBDC4	0.110	0.032	0.841	0.084	0.105	**0.010**	**0.823**	0.085
B3SA3	0.299	0.050	**1.077**	0.114	0.277	0.032	1.132	0.106
BBAS3	**0.161**	**0.020**	**1.038**	0.053	0.202	0.039	1.156	0.058
ABEV3	0.053	**0.005**	0.708	0.138	0.071	0.007	0.860	0.068
MGLU3	0.717	0.121	**1.193**	**0.049**	1.152	0.178	1.726	0.074
VVAR3	0.015	0.003	**1.710**	0.230	0.018	0.003	2.040	0.337
	Models without splines:							
	MSE				**MAPE**			
Metric	**AVG**		**STD**		**AVG**		**STD**	
Symbol	**LSTM-CEEMDAN**							
PETR4	0.247		0.153		1.269		0.165	
VALE3	0.567		0.403		1.297		0.280	
BOVA11	1.523		1.146		0.910		0.269	
ITUB4	0.119		0.063		1.280		0.242	
BBDC4	0.201		0.128		1.490		0.418	
B3SA3	0.673		0.567		1.569		0.347	
BBAS3	0.660		0.478		1.918		0.547	
ABEV3	0.106		0.049		0.984		0.197	
MGLU3	1.103		0.419		1.850		0.333	
VVAR3	0.085		0.097		2.828		1.077	
Symbol	**XLSTM-CEEMDAN**							
PETR4	0.232		0.154		1.480		0.565	
VALE3	1.433		1.412		1.290		0.228	
BOVA11	2.390		3.336		1.385		0.655	
ITUB4	0.280		0.242		1.222		0.170	
BBDC4	0.564		0.621		2.304		0.923	
B3SA3	6.011		5.410		2.486		1.535	
BBAS3	3.973		3.099		3.213		1.436	
ABEV3	0.120		0.059		1.458		0.475	
MGLU3	2.489		1.841		2.605		0.790	
VVAR3	0.123		0.146		2.887		0.904	

Table 6: Average improvements relative to CEEMDAN-LSTM (%) over top ten liquidity Brazilian equities

	Architectural modification improvement relative to vanilla CEEMDAN-LSTM (%)				
Threshold	**IMF3**		**IMF2**		**N/A**
Metric	**spline**	**x+spline**	**spline**	**x+spline**	**x**
mse_avg	60.934	**65.233**	53.686	62.285	-145.209
mse_std	**94.595**	89.910	89.369	92.432	-266.306
mape_avg	26.695	27.054	21.672	**27.234**	-35.755
mape_std	**76.961**	68.464	76.471	76.797	-135.948

7 PHASE 2 RESULTS

Phase 2 uses the same X-CEEMDAN-LSTM model as in Phase 1, as well as its training and testing methods. However, more data sets are considered and there is an ablation test, which overall increases the complexity of experimentation.

Once the results of the ablation test have been done, the analysis and evaluation of different state-of-the-art relevance metrics begin. The goal is to find, in an objective way, what is the best general relevance metric upon which to make decisions about cutting off less relevant exogenous time series. And for this assessment the two proposed scores in the method are used, and visual comparisons are made in the Appendix C.

7.1 Ablation test

Considering the four datasets, each with four distinct exogenous features, use of CEEMDAN decomposition or not, and the five experiment repetitions before mentioned in the method, the actual total number of experiments in the ablation test reaches 640 experiments, which is only possible by using the further detailed proposed experimental framework. The development took months and the final result took about a week.

Figures 38, 39, 40 and 41 show the intermediate components and belong to the Appendix of this work due to the space they take. And Figure 42 shows the prediction results for vanilla LSTM, without the use of CEEMDAN. All figures were taken with random feature combination just as examples of forecasting results.

The full accuracies are shown in tables 14, 15, 16 and 17 summarizing the accuracy results for the ablation tests and these tables are kept in the Appendix for the space they take.

dataset	ENERGY_LOAD	ETTh1	FINANCE	METEOSUISSE
target_feature	pwr.cons.w	OT	Close	birch.npm3
actual_best_feature_ceemdan	**p.amb.in.hp**	**LULL**	Low	**temp.c**
ceemdan_rmse	**75.7868497**	**0.1478283**	24.5695808	**26.3536735**
actual_best_feature_no_ceemdan	lux.south.lx	HULL	**High**	precip.mmsum
no_ceemdan_rmse	1451.864147	175.8252575	**23.5081446**	1593.450294
selected_by_FARM_global	lux.south.lx	**LULL**	Low	**temp.c**
selected_by_correlation	lux.south.lx	HULL	Low	**temp.c**
selected_by_FastDTW	bat.load.w	LUFL	**High**	precip.mmsum
selected_by_FastDDTW	**p.amb.in.hp**	**LULL**	Open	snow.cmsum
selected_by_Euclidean	bat.load.w	LUFL	Low	snow.cmsum

Table 7: Feature selection for each relevance metric.

dataset	rank	features	NoFeatures	target_feature	final_rmse_test	ceemdan
ENERGY_LOAD	1	['p.amb.in.hp']	1	pwr.cons.w	75.7868497	True
ETTh1	1	['LULL']	1	OT	0.1478283	True
FINANCE	1	['High']	1	Close	23.5081446	False
METEOSUISSE	1	['temp.c']	1	birch.npm3	26.3536735	True

Table 8: Best result from ablation tests for each data set using only a single exogenous feature.

7.2 Feature selection for single exogenous time series

The accuracy results used for this analysis are those in tables 14, 15, 16, 17 filtering the rows where the number of input features is 1.

The relevance metrics in Table 18 are used for selection, and each metric is interpreted as having a monotonically increasing relationship with the improvement in RMSE.

Both tables are merged on the basis of the dataset, target feature, and exogenous feature used as input. Then the analysis in Table 7 is made possible.

Applying the simple selection rule mentioned in the method of this phase, table 7 is obtained in a straightforward manner. And Tables 8 and 9 represent the particular case when the top score is calculated using a single exogenous feature and the accurate result of top 1, which is completely consistent with the results of the simple selection rule in Table 7.

In table 7, the feature names in bold are those correctly selected by the relevance metrics. The correct best feature is selected considering the best RMSE among the models with and without CEEMDAN. Notice that using the CEEMDAN decomposition technique improves RMSE for 3 of the 4 datasets. The only case where vanilla X-LSTM outperforms CEEMDAN is for the finance dataset, most probably because of the bad performance of the X-CEEMDAN-LSTM model in predicting the residue of nonstationary time series under the same selected hyperparameters that work best for other data sets.

Dataset	Metric	Exogenous Feature	Top_1_Occurrence	Metric_Top_Score
ETTh1	**Correlation**	HULL	0	0
ETTh1	**Euclidean**	LUFL	0	0
ETTh1	**FARM Global**	LULL	1	1
ETTh1	**FastDDTW**	LULL	1	**1**
ETTh1	**FastDTW**	LUFL	0	0
Energy Load	**Correlation**	lux.south.lx	0	0
Energy Load	**Euclidean**	bat.load.w	0	0
Energy Load	**FARM Global**	lux.south.lx	0	0
Energy Load	**FastDDTW**	p.amb.in.hp	1	**1**
Energy Load	**FastDTW**	bat.load.w	0	0
Finance	**Correlation**	Low	0	0
Finance	**Euclidean**	Low	0	0
Finance	**FARM Global**	Low	0	0
Finance	**FastDDTW**	Open	0	0
Finance	**FastDTW**	High	1	**1**
Meteosuisse	**Correlation**	temp.c	1	**1**
Meteosuisse	**Euclidean**	snow.cmsum	0	0
Meteosuisse	**FARM Global**	temp.c	1	**1**
Meteosuisse	**FastDDTW**	snow.cmsum	0	0
Meteosuisse	**FastDTW**	precip.mmsum	0	0

Table 9: Metric top score based on number of occurrences of feature in experiment input feature set in the top result using only a single exogenous feature.

From table 7: Euclidean distance had the worst metric performance because it did not select any correct exogenous feature; correlation and FastDTW had medium metric performance because both selected 1 correct feature out of the 4 datasets; FARM global and FastDDTW had the best metric performance because both selected 2 correct features out of the 4 datasets.

Since there are relevance metrics tied in the Top score for this single exogenous input analysis, an analysis of RMSE improvement compared to metric improvement is shown in Table 20, this will allow us to untie the metrics. Table 20 is converted to a visual comparison of successive normalized percentage improvements related to each metric and RMSE in Figure 43. Improvements for distance metrics and RMSE are always considered towards low values, for FARM global and correlation the improvements are considered towards high values.

For example, looking at the column RMSE for CEEMDAN True, which is the best absolute RMSE compared to CEEMDAN False as in Table 7, in the item (a) of Figure 43 it is possible to understand that the most significant feature for accuracy improvement is LULL, followed by LUFL and HULL with smaller but significant importance. The same overall behavior is seen for the FARM metric and the FastDDTW metric, but the difference of improvement suggested by FastDDTW from LUFL to LULL is too large when

Data Set	ENERGY_LOAD	ETTh1	FINANCE	METEOSUISSE
farm_global score	28.14	**95.04**	**61.54**	74.96
correlation score	34.38	57.25	51.14	**80.36**
FastDTW distance score	38.78	79.97	51.3	21.99
FastDDTW distance score	**59.9**	80.04	51.03	36.65
Euclidean distance score	36.12	82.13	51.33	22.7

Table 10: Relevance metric improvement score for single exogenous input. Best results in bold.

compared to the actual improvement when considering RMSE. Compared to FastDDTW, the FARM metric has a proportionality closer to the RMSE.

In order to bring this visual comparison to a numerical one, the following **metric improvement score calculation** process is established:

1. Starting with the improvement values from Table 20, normalize improvements between 0 and 100%

2. Calculate the improvement differences between each metric column to the best RMSE column for each feature. For the Finance data set, the best RMSE column is with CEEMDAN False, and for the rest of the data sets the best RMSE column is with CEEMDAN True.

3. Take the absolute values of the differences and the sum within the data sets for each metric.

4. Since the worst-case scenario would be a metric that results in a difference of 100% for all features, the maximum possible value for the sum of differences for 4 features would be 400. So we divide all results by 400 and make them percentage values, from which we subtract 1 and take the absolute value again.

5. Finally, obtain the metric score; the closer to 100 the better, and the closer to zero the worse.

The resulting scores are shown in Table 10 and the best results are in bold. The FARM metric is the best metric according to this score for the Finance and ETTh1 data sets; the correlation metric is the best for the Meteosuisse data set; FastDDTW distance score is the best metric for the energy load data set; FastDTW and Euclidean distance metrics were not the best in any case.

A comparison of the results of Tables 7 and 10 raises the following observations:

- The FARM global metric is the best for the ETTh1 data set in both analyses.
- The FARM global metric correctly selects the Meteosuisse dataset best feature, but loses in score for the correlation, although both are very close to each other.
- The FARM global metric is the best for the finance data set according to Table 10, but it selects the wrong feature as the best in Table 7. This is due to the FARM metric recognizing a less strong relevance for the Open feature compared to other metrics, which turned out to be true, although all other metrics put Open, High, Low almost in the same relevance level.
- The correlation metric is the best for the Meteosuisse data set in both analyses.
- FastDDTW distance metric is the best for the Energy Load data set in both analysis.
- FastDTW distance metric is the best for the Finance dataset in Table 7, but FARM global is the best for the Finance data set in Table 10.

It is worth mentioning that the results for the Finance data set are divergent because it is the only data set out of the 4 that contains nonstationary monotonic trend, so it is expected to be different. Also, the Volume feature for the Finance data set brings significant improvements in forecasting accuracy and none of the metrics managed to recognize its value.

After all, even under these circumstances, the Farm global metric outperformed the other ones in most cases. A second best metric would be the Fast DDTW based on the previous listed findings.

7.3 Feature selection for multiple exogenous time series

As a starting point, let us see the overall best results in Table 11. First important observation is that the three best results overall include exogenous features, which means that using exogenous features improved accuracies for all data sets. This is a positive finding for the main hypothesis of this work: the use of exogenous features might enhance model's learning.

Looking from the CEEMDAN decomposition perspective, it is worth mentioning that the overall results improved significantly when using CEEMDAN for three out of four

dataset	rank	features	NoFeatures	target_feature	final_rmse_test	ceemdan
ENERGY_LOAD	**1**	['p.amb.in.hp', 'bat.load.w']	2	pwr.cons.w	74.2627627	True
ENERGY_LOAD	**2**	['p.amb.in.hp']	1	pwr.cons.w	75.7868497	True
ENERGY_LOAD	**3**	['bat.load.w']	1	pwr.cons.w	325.4334619	True
ETTh1	**1**	['LULL']	1	OT	0.1478283	True
ETTh1	**2**	['LUFL']	1	OT	0.3815923	True
ETTh1	**3**	['LUFL', 'LULL']	2	OT	0.3850976	True
FINANCE	**1**	['Low', 'Volume']	2	Close	0.7410814	False
FINANCE	**2**	['Open', 'High', 'Low']	3	Close	0.766522	False
FINANCE	**3**	['High', 'Low', 'Volume']	3	Close	0.9109977	False
METEOSUISSE	**1**	['temp.c']	1	birch.npm3	26.3536735	True
METEOSUISSE	**2**	['wind.mps']	1	birch.npm3	34.5296584	True
METEOSUISSE	**3**	['snow.cmsum']	1	birch.npm3	35.7643445	True

Table 11: Three best results from ablation tests for each data set. Rank column represents the best as 1, second best as 2 and third best as 3.

data sets. The Finance data set is the only one which had better results without using CEEMDAN decomposition technique, this is due to the monotonic long period and non stationary trend not present in the other data sets. This trend makes it difficult for model's like LSTM to learn without having proper hyperparameters set. As described in the method, the hyperparameters are set as the same for all datasets for sake of standardization, and this is why the results for Finance data set in phase 2 of this work may differ from the ones in previous phases.

Looking to the relevance values in Table 18, and consequently to the average rank of features in Table 19, we should expect to see LUFL and HULL for the ETTh1 dataset, lux.west.lx and lux.south.lx for the Energy Load data set, Low and High for the Finance data set, and precip.mmsum and snow.cmsum for Meteosuisse data set. Compared to the best results in Table 11 we can notice that the average rank is not a good predictor. So we should try to find which individual metric rank is the best predictor of the top results in RMSE. The number of top results considered are arbitrarily chosen, here three is the chosen number.

Table 12 presents the metric_top_score which is calculated by summing the number of occurrences of the feature in the top N experiments divided by N, for example: LULL feature occurs 1 time in the top 1, 1 time in the top 2 and 2 times in the top 3, which yields $1 + 1/2 + 2/3 = 2.17$. Based on Table 12, the best metric for features that appear in the top results for each data set are the following:

- For ETTh1 data set are **FARM** and **FastDDTW**
- For the Energy Load data set it is the **FastDDTW**
- For the Finance data set **correlation**, **FARM** and **Euclidean** are tied at the top

Dataset	Metric	Exogenous Feature	Top_3_Occurrence	Top_2_Occurrence	Top_1_Occurrence	Metric_Top_Score
ETTh1	**FARM Global**	**LULL**	**2**	**1**	**1**	**2.17**
ETTh1	**FastDDTW**	**LULL**	**2**	**1**	**1**	**2.17**
ETTh1	**FastDTW**	**LUFL**	2	1	0	1.17
ETTh1	**Euclidean**	**LUFL**	2	1	0	1.17
ETTh1	**Correlation**	**HULL**	0	0	0	0.0
Energy Load	**FastDDTW**	**p.amb.in.hp**	**2**	**2**	**1**	**2.67**
Energy Load	**FastDTW**	**bat.load.w**	2	1	1	2.17
Energy Load	**Euclidean**	**bat.load.w**	2	1	1	2.17
Energy Load	**Correlation**	**lux.south.lx**	0	0	0	0.0
Energy Load	**FARM Global**	**lux.south.lx**	0	0	0	0.0
Finance	**Correlation**	**Low**	**3**	**2**	**1**	**3.0**
Finance	**FARM Global**	**Low**	**3**	**2**	**1**	**3.0**
Finance	**Euclidean**	**Low**	**3**	**2**	**1**	**3.0**
Finance	**FastDTW**	**High**	2	1	0	1.17
Finance	**FastDDTW**	**Open**	1	1	0	0.83
Meteosuisse	**Correlation**	**temp.c**	**1**	**1**	**1**	**1.83**
Meteosuisse	**FARM Global**	**temp.c**	**1**	**1**	**1**	**1.83**
Meteosuisse	**FastDDTW**	**snow.cmsum**	1	0	0	0.33
Meteosuisse	**Euclidean**	**snow.cmsum**	1	0	0	0.33
Meteosuisse	**FastDTW**	**precip.mmsum**	0	0	0	0.0

Table 12: Metric top score based on number of occurrences of feature in experiment input feature sets among the top results.

- For the Meteosuisse data set **correlation** and **FARM** are the best

FARM metric is the one that has the overall best score for the considered data sets because it achieves best score for 3 out of 4 data sets, followed by FastDDTW and Correlation tied in second place, and Euclidean distance at third. The other metrics did not achieve significant scores. In order to untie and confirm the best overall metrics, besides this static analysis we will also create scores for how good the metric predicts the improvement size in RMSE test.

The accuracy improvement analysis for multiple exogenous features as input will have a similar approach as in the single exogenous feature **metric improvement score calculation process** described before; however, it will need the following adaptation: the final feature improvement score will be the sum of the improvement scores in all scenarios for each input number of features where the considered feature happens to be in the input divided by the number of times it happens to be in the input. This change will take place in step 1 of the **improvement metric score calculation process** described in the single exogenous feature selection process.

The number of occurrence of a feature follows the Newton's binomial, for example, for single feature input, each feature happens to be in the input only once; for 2 features input, each feature happens to be in the input three times; for 3 features input, each feature happens to be in the input three times; for 4 features input each feature happens to be in the input only once again. The resulting normalized improvement scores are seen in table 21. The same table is visually shown for better comparison in Figure 44.

Based on Table 21, the metric improvement scores between 0 and 100 are calculated

		Improv. Score			
CEEMDAN		**True**	**True**	**False**	**True**
Metric	**#NF**	**Energy Load**	**ETTh1**	**Finance**	**Meteosuisse**
correlation	**1**	34.38	57.25	51.14	**80.37**
FARM	**1**	28.14	**95.04**	**61.54**	74.97
fastdtw	**1**	38.78	79.97	51.3	21.99
fastddtw	**1**	**59.9**	80.04	51.03	36.64
Euclidean	**1**	36.12	82.14	51.34	22.7
correlation	**2**	35.56	75.77	22.82	**87.4**
FARM	**2**	29.32	71.67	**33.27**	83.12
fastdtw	**2**	39.38	**91.8**	22.98	38.55
fastddtw	**2**	**61.08**	51.82	23.03	36.09
Euclidean	**2**	36.72	89.63	23.1	39.27
correlation	**3**	31.36	60.89	74.58	66.95
FARM	**3**	25.12	86.55	**84.97**	**81.48**
fastdtw	**3**	37.47	85.78	74.73	41.28
fastddtw	**3**	**56.04**	66.7	74.46	32.08
Euclidean	**3**	34.81	**92.75**	74.77	40.59

Table 13: Improvement scores for all metrics and number of features, with CEEMDAN based on best results

and obtained in Table 13.

The results shown in Table 13 provide some key observations.

- FARM metric is the most consistent and outstanding for the Finance data set. We can see in Figure 44 item (b) that this is because FARM is the only metric able to predict that the Open variable is not as meaningful as other High and Low features, which matches the best results in RMSE.

- FastDDTW is the most consistent metric for the Energy Load data set, because it has the best improvement score for 1, 2 and 3 features as input. Also FastDDTW outstands other metrics significantly for the Energy load data set. We can see in Figure 44 item (c) that FastDDTW is the only metric that points toward p.amb.in.hp as the best metric, and this is proven correct since the results when using CEEMDAN technique for the Energy Load data set match.

- FARM, FastDTW and Euclidean have the best scores for the ETTh1 data set. This can be explained by the high overall score values of all metrics for ETTh1, meaning that this behavior is related to the data set itself and not to a specific metric.

- For Meteosuisse data set, correlation does a good job in for 1 and 2 features but is outperformed by FARM when there are 3 input features. It is worth mentioning

that the correlation and FARM have significantly better scores for all numbers of features when compared to other metrics. This means that distance based metrics do not fit the Meteosuisse data set, most probably due to the long periods of zero values in some features, as shown in Figure 30.

Gathering both the improvement score results from Table 13 and the top score results from Table 12, it is possible to notice that FARM outstands other metrics overall. FARM is responsible for 5 out of the 12 best improvement scores from Table 13 and 3 best top score out of 4 data sets from Table 12. Tied in second place the FastDDTW and correlation metrics come in, both responsible for 3 out 12 best improvement scores from table 13 and 2 best top score out of 4 data sets from Table 12.

7.4 Experimenting framework

The experimenting framework design described in the method of phase 2 drives the general division of data processing and model training for both processes to be possible in parallel. However, the actual building of the application structure needed to be created.

The Figure 33 show the application level topology of the framework. It relies on databases such as PostgreSQL for relational data or S3 for file type data, also a key component of the framework is the message queue, such as RabbitMQ or Kafka and the user interface is built using streamlit.

The message queue is the main horizontal scalability enabler, and it works with the publisher and consumer concept. The framework UI provides data processing and training instructions for the queues, which in turn may have multiple consumers, which are materialized as Python processes in multiple servers.

The data flow of the Figure 33 is the following:

1. The user provides a raw data set in CSV or parquet format on the user interface shown in Figure 33 item (a) which saves the data in a relational data base.

2. The user selects the data processing hyperparameters in the UI shown in Figure 34 item (a) and hits the button to launch the data processing with the selected hyperparameters. And the data processing parameters are sent to the server where the back-end of the UI is hosted.

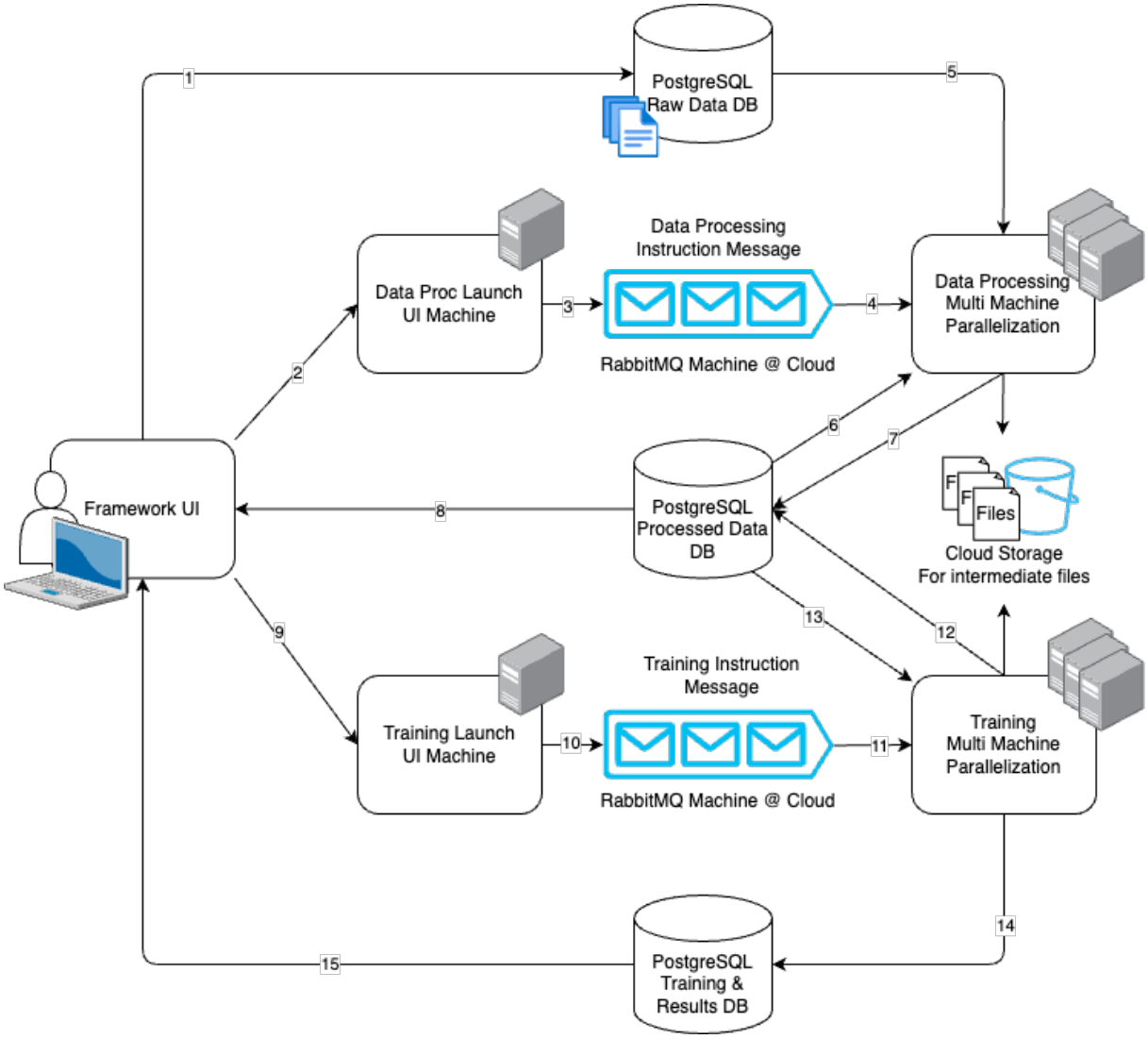


Figure 33: Developed Scalable data processing and training Python Application Framework architecture

3. The back-end server processes the parameters for data processing, creates individual instructions for each experiment, and sends them to the message queue.

4. Data processing workers consume the instructions in parallel from the message queue.

5. The data processing workers retrieve the correct raw data from the data base for the experiment it is handling.

6. Data processing workers query the processed database to prevent duplicated processing. This behavior can be overridden to force overwriting results.

7. The resulting processed data that underwent scaling, windowing, decomposition and relevance metrics calculation according to the chosen data processing parameters are saved in the database and are ready to be used for training. Temporary files generated in intermediate steps of data processing are saved in cloud storage for debugging purposes.

8. The user sees and inspects the processed data sets in the user interface as in Figure 34 item (b).

9. The user selects the parameters for the training process, including the related processed data, which are sent to the UI backend.

10. The server converts the training parameters into training instructions for each experiment and sends to the message queue.

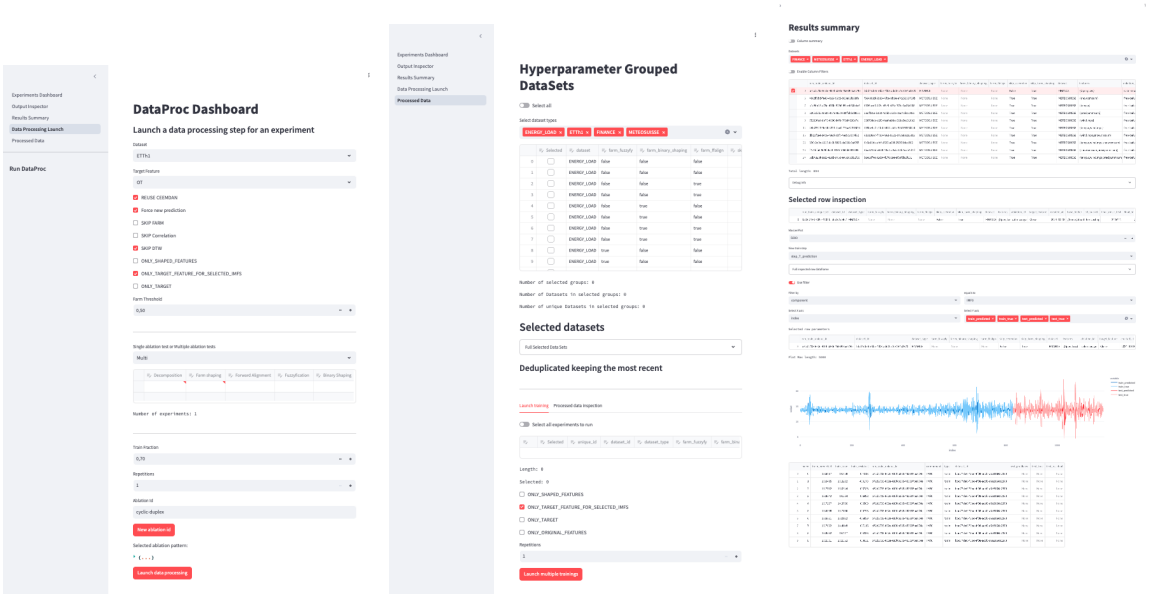
11. Training workers consume the training instructions in parallel from the message queue.

12. Training workers query the data base for the corresponding processed data set.

13. Training workers retrieve the corresponding experiment processed data set.

14. Training workers run the training process and save in the data base the final results such as accuracy metrics and prediction graphs. The training workers also save in cloud storage intermediate files for later debugging.

15. The user sees and inspects the final results and graphs of the training as shown in Figure 34 item (c). The user is able to inspect each experiment results individually and a summarized result table, the source of Tables 14, 15, 16 and 17.



(a) (b) (c)

Figure 34: Developed Python framework UI screenshots of, from left to right, data processing launch page, training launch page and results summary page

This large-scale experimentation framework is made publicly available at `https://github.com/avilarenan/PyHeta`. It is compatible with any PyTorch model and any time series data set, being flexible to even changing the process steps as described in Figure 31.

PART IV

CONCLUSION & FUTURE WORK

8 OVERVIEW

This work is composed of two main phases, each providing its own results, which are motivated in a chained fashion.

The initial motivation for this work is to look for improvements in the forecast accuracy of time series through machine learning. It all started with a main focus on the financial domain and was further expanded. Thus, the first phase starts after a bibliography review in the field of time-series forecasting, which resulted in finding a paper that presented a preprocessing technique coupled with a specific data-flow architecture, leveraging traditional LSTM models, the CEEMDAN-LSTM.

But, considering a well-known hypothesis which states that the use of exogenous variables may improve performance of models, the investigation of exogenous time-series addition to the proposed model started. During this search, it was found that with a data flow architecture, which is proposed in phase 1 of this work and called X-CEEMDAN-LSTM, specifically designed to enable exogenous features to be used to predict the target feature, it is possible to improve performance.

However, not every exogenous feature is capable of adding relevant information for the model to improve forecasting; the exogenous features of the first phase were chosen by previous knowledge about the finance data set domain. In this context, it is necessary to have a method to decide whether an exogenous feature is worth adding as input for the model. The actual trade-off regards the balance between data size increasing and the consequent increase in the complexity of training, thus more difficulty in converging to an optimal solution, and the addition of valuable data that may bring new helpful information for prediction of target feature.

This trade-off was the main motivation for the qualification process of this Master degree work. During the qualification, the focus was on testing whether a chosen relevance metric for time series similarity could help in choosing exogenous features, which ended up being made based on two different approaches, the previous knowledge about the

domain-driven one and all the possible exogenous features that passed a metric threshold one. At that point, the length of the finance time series dataset was slightly expanded and benchmark prediction models other than LSTM were also tested. The preliminary results showed that choosing features based on simple rules derived from a single relevance metric is not enough.

Since a single relevance metric threshold combination of exogenous feature input did not provide a general improvement, it was necessary to test more metrics and with different datasets, in order to experiment with different behaviors. That is where the second phase starts, motivated by the need to test the model in scale with larger datasets and with all the possibilities for a group of exogenous features.

In order to produce experiments at scale, a side consequence of this work was a Python framework to launch data processing and model training. This made it possible to generate results for hundreds of experiments using large datasets. However, the main goal of this second phase was to find an accurate filter-based feature selection method through relevance metric evaluation to choose the most useful exogenous features for each data set. And also evaluate the capability of the model to improve forecasting accuracy when dealing with different datasets when compared to the vanilla CEEMDAN-LSTM without exogenous features.

9 CONCLUSION

The first phase of this research laid the foundation for integrating exogenous features into time series forecasting models, particularly by enhancing the CEEMDAN-LSTM architecture. Traditional LSTM-based approaches often rely solely on past values of the target series for predictions, limiting their ability to incorporate external influences. To address this limitation, this phase introduced a generalized CEEMDAN-LSTM framework capable of handling multiple exogenous inputs while preserving the advantages of empirical mode decomposition.

A key methodological contribution of this phase was the hybrid modeling approach, where different components of the decomposed time series were handled through separate predictive mechanisms. High-frequency components, which tend to exhibit nonlinear and volatile patterns, were processed using deep learning techniques, while lower-frequency trends were projected using spline interpolation. This combination allowed the model to capture both short-term fluctuations and long-term dependencies more effectively than conventional approaches.

The experimental results on ten of the most liquid Brazilian stocks confirmed that the proposed approach improved prediction accuracy compared to the baseline CEEMDAN-LSTM model. Specifically, the proposed framework achieved a **27.23%** reduction in MAPE (Mean Absolute Percentage Error), demonstrating the importance of incorporating exogenous information into financial time series forecasting. Furthermore, the identification of an optimal IMF threshold at IMF2 provided valuable insights into the role of decomposition in feature extraction. These findings set the stage for the subsequent research phases, where feature selection methodologies were further refined to enhance predictive performance.

The second and final phase of this research focused on developing a generalizable feature selection method that could be applied across different data sets while maintaining high forecasting accuracy. Given the limitations of using an arbitrary relevance metric to select features, this phase aimed to evaluate multiple relevance metrics to provide a

more holistic method to choose the right relevance metric for analyzing the importance of exogenous features.

A critical component of this phase was the implementation of an extensive ablation study, where all possible combinations of exogenous features were tested across four distinct datasets. This large-scale experimentation required the development of a scalable Python-based framework capable of efficiently processing, training, and evaluating a wide range of feature subsets. The framework was designed to handle increasing data complexity, allowing for systematic exploration of feature selection methods in high-dimensional spaces.

Results from this phase showed the evaluation of multiple relevance metrics across an ablation test with all possible feature combination in a considered set of 4 exogenous features for each data set. This led to conclusion that the results selected with FARM metric mostly outperformed other metrics, showing that its capability of considering local relevance can produce better overall results, regardless of the raw data shape. FastDDTW and correlation are worth mentioning because they outperformed FARM in a few scenarios. Specifically, models trained with carefully selected exogenous features using the best relevance metric exhibited higher accuracy, validating the importance of a wide assessment of multiple relevance metrics. Furthermore, the study provided evidence that while exogenous features can enhance predictive accuracy, their selection must be optimized to balance model complexity and interpretability.

Overall, this phase reinforced the importance of feature selection in time series forecasting, particularly in scenarios where exogenous information plays a crucial role. The developed methodology offers a robust foundation for future research, enabling the integration of more advanced selection techniques, domain knowledge, and automated feature ranking approaches.

With these findings, this research contributes to the growing field of machine learning for time series forecasting, demonstrating that including exogenous features and choosing the right relevance metric for feature selection can significantly enhance predictive performance and save computational power.

10 FUTURE WORK

While this research has demonstrated significant advancements in time series forecasting with exogenous feature selection, several directions remain open for further investigation:

- **Expanding the dataset scope:** The current study was limited to four datasets, each with specific characteristics. Future research should include a broader range of datasets, covering different time series domains (e.g., energy, climate, finance, and healthcare) to validate the generalizability of the feature selection methodology.
- **Exploring additional predictive models**: While the research primarily focused on LSTM-based architectures, alternative deep learning models such as transformers, attention-based models, and hybrid architectures should be investigated. These models have demonstrated superior performance in recent time series forecasting studies.
- **Enhancing feature selection techniques**: The current feature selection method relies on relevance metrics, but state-of-the-art techniques such as deep feature extraction, autoencoders, and reinforcement learning-based selection could further refine the process. Methods such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) could be employed to enhance feature interpretability.
- **Investigating dimensionality reduction strategies**: High-dimensional input spaces pose a challenge to model convergence and computational efficiency. Future studies should explore principal component analysis (PCA), t-SNE, UMAP, and feature pruning techniques to optimize model training and inference time.
- **Automating hyperparameter tuning**: The choice of IMF thresholds, window sizes, and model hyperparameters significantly influences prediction accuracy. Future research should implement Bayesian optimization, evolutionary algorithms, or neural architecture search (NAS) to automate hyperparameter tuning.

- **Integrating domain knowledge into feature selection**: Current methods are primarily data-driven. Incorporating domain expertise and causality analysis could refine the selection process, ensuring meaningful relationships between chosen exogenous features and target variables.
- **Cloud-based large-scale experimentation**: Future work should explore distributed training approaches, federated learning, and real-time feature selection in cloud environments to enhance performance across large-scale datasets.
- **Time series pre-processing with local relevance-based shaping techniques**: Future studies should investigate approaches that selectively highlight the most relevant sections of exogenous time series data while attenuating less meaningful portions. This would allow for more efficient and interpretable model training by focusing on the most impactful time periods for predicting the target variable.

By addressing these directions, future research can further refine feature selection methodologies and advance the state-of-the-art in time series forecasting with exogenous data integration.

BIBLIOGRAPHY

Ignatius Wiseto Agung. Input parameters comparison on narx neural network to increase the accuracy of stock prediction. *JOURNAL OF INFORMATICS AND TELECOMMUNICATION ENGINEERING*, 6(1):82–90, 2022. ISSN 2549-6255. doi: 10.31289/jite.v6i1.7158. URL http://ojs.uma.ac.id/index.php/jite/article/view/7158.

Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM*, 1970.

Matthew Beal, Zoubin Ghahramani, and Carl Rasmussen. The infinite hidden markov model. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2001/file/e3408432c1a48a52fb6c74d926b38886-Paper.pdf.

Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pages 359–370, 1994.

George E Box and Gwilym M Jenkins. Time series analysis, control, and forecasting. *San Francisco, CA: Holden Day*, 3226(3228):10, 1976.

George E. P. Box, Gwylim M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. A JOHN WILEY & SONS, INC., 4 edition, 2012. ISBN 9780470272848. doi: 10.4135/9781412986366.

Jian Cao, Zhi Li, and Jian Li. Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A*, 519:127–139, 2019. ISSN 0378-4371. doi: 10.1016/j.physa.2018.11.061. URL https://doi.org/10.1016/j.physa.2018.11.061.

C. Caruso and F. Quarta. Interpolation methods comparison. *Computers Mathematics with Applications*, 35(12):109–126, 1998. ISSN 0898-1221. doi: https://doi.org/10.1016/S0898-1221(98)00101-1. URL https://www.sciencedirect.com/science/article/pii/S0898122198001011.

Ramón Christen, Luca Mazzola, Alexander Denzler, and Edy Portmann. Distance metrics for evaluating the use of exogenous data in load forecasting. In Davide Ciucci, Inés Couso, Jesús Medina, Dominik Ślezak, Davide Petturiti, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 469–482, Cham, 2022. Springer International Publishing. ISBN 978-3-031-08974-9.

Ramón Christen, Luca Mazzola, Alexander Denzler, and Edy Portmann. Exogenous data for load forecasting: A review. *Proceedings of the 12th International Joint Conference on Computational Intelligence (IJCCI 2020)*, pages 489–500, 2020. doi: 10.5220/0010213204890500.

Ramón Christen, Luca Mazzola, Alexander Denzler, and Edy Portmann. Exogenous data in forecasting: Farm - a new measure for relevance evaluation. *arXiv*, 2304.11028v2, 2023. URL `https://arxiv.org/abs/2304.11028`.

Fulvio Corsi and Roberto Renò. Discrete-time volatility forecasting with persistent leverage effect and the link with continuous-time volatility modeling. *Journal of Business & Economic Statistics*, 30(3):368–380, 2012. doi: 10.1080/07350015.2012.663261. URL `https://doi.org/10.1080/07350015.2012.663261`.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL `https://doi.org/10.1007/BF02551274`.

Renan de Luca Avila and Glauber De Bona. Financial time series forecasting via ceemdan-lstm with exogenous features. In Ricardo Cerri and Ronaldo C. Prati, editors, *Intelligent Systems*, pages 558–572, Cham, 2020. Springer International Publishing. ISBN 978-3-030-61380-8.

Eugen Diaconescu. The use of narx neural networks to predict chaotic time series. *Wseas Transactions on computer research*, 3(3):182–191, 2008.

Cem Emeksiz and Mustafa Tan. Multi-step wind speed forecasting and hurst analysis using novel hybrid secondary decomposition approach. *Energy*, 238:121764, 2022. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2021.121764. URL `https://www.sciencedirect.com/science/article/pii/S0360544221020120`.

Robert F. Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 1982. ISSN 00129682. doi: 10.2307/1912773.

Robert F Engle and Clive WJ Granger. Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276, 1987.

Mariano Gasca and Thomas Sauer. On the history of multivariate polynomial interpolation. In C. Brezinski and L. Wuytack, editors, *Numerical Analysis: Historical Developments in the 20th Century*, pages 135–147. Elsevier, Amsterdam, 2001. ISBN 978-0-444-50617-7. doi: https://doi.org/10.1016/B978-0-444-50617-7.50007-0. URL `https://www.sciencedirect.com/science/article/pii/B9780444506177500070`.

Felix Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. pages 669–676, 08 2001. doi: 10.1007/3-540-44668-0_93.

Dimitrios Giannakis. Data-driven spectral decomposition and forecasting of ergodic dynamical systems. *Applied and Computational Harmonic Analysis*, 47(2):338–396, 2019.

Tian Guo and Tao Lin. Multi-variable lstm neural network for autoregressive exogenous model, 2018. URL `https://arxiv.org/abs/1806.06384`.

Peter R. Hansen, Asger Lunde, and James M. Nason. The model confidence set. *Econometrica*, 79(2):453–497, 2011. doi: https://doi.org/10.3982/ECTA5771. URL `https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA5771`.

Simon Haykin. *Neural Networks and Learning Machines*. P. Hall, third edit edition, 2009. ISBN 9780131471399.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735.

Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2003.09.015. URL `https://www.sciencedirect.com/science/article/pii/S0169207003001134`.

Qiongxia Huang, Riqing Chen, Xianghan Zheng, and Zhenxing Dong. Deep sentiment representation based on CNN and LSTM. *Proceedings - 2017 International Conference on Green Informatics, ICGI 2017*, pages 30–33, 2017. doi: 10.1109/ICGI.2017.45.

David Hume. *A Treatise of Human Nature: A Critical Edition*. Oxford University Press, New York, 2007.

H. E. Hurst. Long-term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers*, 116(1):770–799, 1951. doi: 10.1061/TACEAT.0006518.

Sinan Rasiya Koya and Tirthankar Roy. Temporal fusion transformers for streamflow prediction: Value of combining attention with recurrence. *Journal of Hydrology*, 637: 131301, 2024.

H. Lai, T. Chen, and X. Zhou. Composition of feature selection for time-series prediction with deep learning. *Neural Computing and Applications*, 35:257–273, 2023. doi: 10.1007/s00521-023-01234-5.

Xin Li, Xianzhong Long, Guozi Sun, Geng Yang, and Huakang Li. Overdue prediction of bank loans based on LSTM-SVM. *Proceedings - 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, SmartWorld/UIC/ATC/ScalCom/CBDCo*, pages 1859–1863, 2018. doi: 10.1109/SmartWorld.2018.00312.

Tsungnan Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996. doi: 10.1109/72.548162.

Yu Lin, Yan Yan, Jiali Xu, Ying Liao, and Feng Ma. Forecasting stock index price using the ceemdan-lstm model. *The North American Journal of Economics and Finance*, 57: 101421, 2021. ISSN 1062-9408. doi: https://doi.org/10.1016/j.najef.2021.101421. URL `https://www.sciencedirect.com/science/article/pii/S1062940821000553`.

Yu Lin, Zixiao Lin, Ying Liao, Yizhuo Li, Jiali Xu, and Yan Yan. Forecasting the realized volatility of stock price index: A hybrid model integrating ceemdan and lstm. *Expert Systems with Applications*, 206:117736, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117736. URL `https://www.sciencedirect.com/science/article/pii/S095741742201017X`.

Nicolas Maloumian. Unaccounted forms of complexity: A path away from the efficient market hypothesis paradigm. *Social Sciences Humanities Open*, 5(1):100244, 2022. ISSN 2590-2911. doi: https://doi.org/10.1016/j.ssaho.2021.100244. URL `https://www.sciencedirect.com/science/article/pii/S2590291121001406`.

Benoit B. Mandelbrot and John W. Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM Review*, 10(4):422–437, 1968. doi: 10.1137/1010093. URL `https://doi.org/10.1137/1010093`.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 1943. ISSN 00074985. doi: 10.1007/BF02478259.

Marvin Minsky and Seymour Papert. Perceptron: an introduction to computational geometry. *The MIT Press, Cambridge, expanded edition*, 1969.

N.E. Huang et al. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proc. R. Soc. Lond. A*, 454:903–995, 1998. ISSN 13645021. doi: 10.1098/rspa.1998.0193. URL `papers3://publication/uuid/B74763A9-34E1-446C-8677-521F1BEFF53A`.

Navid Parvini, Mahsa Abdollahi, Sattar Seifollahi, and Davood Ahmadian. Forecasting bitcoin returns with long short-term memory networks and wavelet decomposition: A comparison of several market determinants. *Applied Soft Computing*, 121:108707, 2022. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2022.108707. URL `https://www.sciencedirect.com/science/article/pii/S1568494622001673`.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958. ISSN 0033295X. doi: 10.1037/h0042519.

Dipanwita Saha and Felipe Lopez. A deep learning forecaster with exogenous variables for day-ahead locational marginal price, 2020. URL `https://arxiv.org/abs/2010.06525`.

Anton Maximilian Schäfer and Hans Georg Zimmermann. Recurrent neural networks are universal approximators. In Stefanos D. Kollias, Andreas Stafylopatis, Włodzisław Duch, and Erkki Oja, editors, *Artificial Neural Networks – ICANN 2006*, pages 632–640, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-38627-8.

Jurgen Schmidhuber Sepp Hochreiter. LONG SHORT-TERM MEMORY. *Neural Computation*, 9(8):1–32, 1997.

Patney Rakesh Kumar Singh Pushpendra, Joshi Shiv Dutt and Saha Kaushik. The fourier decomposition method for nonlinear and non-stationary time series analysis. *Proc. R. Soc.*, 2017. doi: 10.1098/rspa.2016.0871.

Mar??a E Torres, Marcelo A Colominas, Gast??n Schlotthauer, and Patrick Flandrin. A complete ensemble empirical mode decomposition with adaptive noise. *ICASSP, IEEE*

International Conference on Acoustics, Speech and Signal Processing - Proceedings, pages 4144–4147, 2011. ISSN 15206149. doi: 10.1109/ICASSP.2011.5947265.

Zheng Wang and Yuansheng Lou. Hydrological time series forecast model based on wavelet de-noising and ARIMA-LSTM. *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, (Itnec): 1697–1701, 2019. doi: 10.1109/ITNEC.2019.8729441.

Paul Werbos and Paul John. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. 01 1974.

Paul J. Werbos. Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 1990. ISSN 15582256. doi: 10.1109/5.58337.

Peter Whittle. *Hypothesis testing in time series analysis*, volume 4. Almqvist & Wiksells boktr., 1951.

Hau-Tieng Wu, Patrick Flandrin, and Ingrid Daubechies. One or two frequencies? the synchrosqueezing answers. *Advances in Adaptive Data Analysis*, 3:29–39, 04 2011. doi: 10.1142/S179353691100074X.

Zhaohua Wu and Norden E. Huang. Ensemble empirical mode decomposition: A noise-assisted data analysis method. *Advances in Adaptive Data Analysis*, 2009. ISSN 17935369. doi: 10.1142/S1793536909000047.

X. Yang, J. Zhao, and P. Li. A comprehensive survey on recent feature selection methods for mixed data: Challenges, solutions, and future directions. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1–18, 2023. doi: 10.1109/TKDE.2023.1234567.

Zhen Yang, Jacky Keung, Md Alamgir Kabir, Xiao Yu, Yutian Tang, Miao Zhang, and Shuo Feng. Acomnn: Attention enhanced compound neural network for financial time-series forecasting with cross-regional features. *Applied Soft Computing*, 111:107649, 2021. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2021.107649. URL `https://www.sciencedirect.com/science/article/pii/S1568494621005706`.

G. Udny Yule. On a Method of Investigating Periodicities in Disturbed Series. *Philosophical Transactions of the Royal Society of London*, 226(Series A):167–298, 1927. URL `https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.1927.0007`.

Keyi Zhang, Ramazan Gençay, and M. Ege Yazgan. Application of wavelet decomposition in time-series forecasting. *Economics Letters*, 158:41–46, 2017. ISSN 0165-1765. doi: https://doi.org/10.1016/j.econlet.2017.06.010. URL https://www.sciencedirect.com/science/article/pii/S0165176517302379.

Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, 48(2):787–794, 2020. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2019.06.004. URL https://www.sciencedirect.com/science/article/pii/S1063520318302045.

Alexandra Gabriela Ţiţan. The efficient market hypothesis: Review of specialized literature and empirical research. *Procedia Economics and Finance*, 32:442–449, 2015. ISSN 2212-5671. doi: https://doi.org/10.1016/S2212-5671(15)01416-1. URL https://www.sciencedirect.com/science/article/pii/S2212567115014161. Emerging Markets Queries in Finance and Business 2014, EMQFB 2014, 24-25 October 2014, Bucharest, Romania.

APPENDIX A – SIGNAL PROCESSING

Some of the honor mentions that needed to be studied along with the developed study.

A.1 Fourier

Fourier transform approximates a given signal as an arithmetic sum of sines and cosines of different amplitude and frequency, often called harmonics. An example of a classic Fourier decomposition is the square wave, shown in 35.

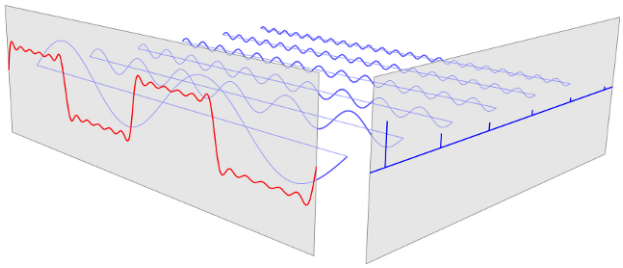


Figure 35: Square wave Fourier decomposition [Source: Wiki, Tex.StackExchange]

The traditional Fourier series expansion of generic continuous signal $Z(t)$ that is defined in the domain of $t_1 \leq t \leq t_1 + T_0$, let us turn it into a univariate discrete and periodic time series, extending it by: $Z_{T_0}(t) = \sum_{k=-\infty}^{\infty} Z(t - kT_0)$, this way $Z(t) = Z_{T_0}(t) \cdot \omega(t)$.

$\omega(t)$ is defined as follows:

$$\omega(t) = \begin{cases} 1 & \text{if } t_1 \leq t \leq t_1 + T_0 \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

Meanwhile, continuous time Fourier decomposition is defined as follows:

$$Z_{T_0}(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos(k\omega_0 t) + b_k \cos(k\omega_0 t)) \tag{A.2}$$

Where $\omega_0 = 2\pi rad \cdot s^{-1}$, $a_0 = \frac{1}{T_0}\int_{t_1}^{t_1+T_0} Z_{T_0}(t)dt$, $a_k = \frac{2}{T_0}\int_{t_1}^{t_1+T_0} Z_{T_0}(t)\cos(k\omega_0 t)dt$, $b_k = \frac{2}{T_0}\int_{t_1}^{t_1+T_0} Z_{T_0}(t)\sin(k\omega_0 t)dt$.

A.1.1 FFT

The above mentioned Fourier transform is intended for continuous data, but here we are dealing with discrete data, for which there is a Discrete Fourier Transform (DFT) algorithm. It works as the following equation adapted from *quick reference*:

$$Z_{DFT}(t) = \frac{1}{\sqrt{N}}\sum_{t=0}^{N-1} Z(t)e^{\frac{-2\pi itk}{N}}, k \in \mathbb{N} \mid k < N \tag{A.3}$$

Where N is the length of the discrete signal $Z(t)$. DFT provides a linear and invertible transformation. The Fast Fourier transform arose in order to enhance the computational performance of DFT. FFT takes a particular case of DFT when the signal has length that is power of 2.

Take $N = 2m$ and substitute in the DFT formula, then the transformation can be divided into odd and even parts:

$$Z_{FFT}(t) = \frac{1}{\sqrt{N}}\sum_{n=0}^{m-1} Z(2t)\omega_m^{tk} + \frac{\omega_N}{\sqrt{N}}\sum_{n=0}^{m-1} Z(2t+1)\omega_m^{tk} \tag{A.4}$$

Where $\omega_N = e^{-\frac{2\pi i}{N}}$. This last form derived from the particular case where $N = 2m$ is the Fast Fourier Transform, and the computational benefit is that it has a recursive form, which is computationally efficient to calculate.

A.1.2 FDM

Over time, the main restriction of using Fourier transform to extract components of the signal was the signal being stationary and linear. But Singh Pushpendra and Kaushik (2017) proposed a method, that relates to empirical mode decomposition, to use Fourier to decompose nonlinear and non stationary signals with the FDM (Fourier decomposition method).

After some manipulation with the complex representation of sines and cosines, the general complex mathematical description of the for each of the Fourier Intrinsic Bands (FIBs) is as follows:

$$FIB_i = Re\{a_i(t)e^{j\theta_i(t)}\} = Re\{\sum_{k=N_{i-1}+1}^{N_i} c_k e^{jk\omega_0 t}\}, i \in \mathbb{N} \mid i \leq M \tag{A.5}$$

With $N_0 = 0$ and $N_M = \infty$, and $\theta_i(t) = i\omega_0 t$. The sum of the real part and imaginary part of the above equation result in signal termed analytical FIB. The number of FIBs is defined increasing the N value and using the following set of stop criteria:

$$N_{i-1} + 1 \leq N_i \leq \infty, a_i(t) \geq 0, \omega_i(t) = \frac{d\theta_i(t)}{dt} \geq 0, \forall t \tag{A.6}$$

The mathematical algorithmic steps to decompose a given signal $Z(t)$, without linearity and stationary restrictions, with the Fourier decomposition method (Singh Pushpendra and Kaushik, 2017) are the following. The DT-FDM algorithmic summary (LTH-FS) to obtain AFIBFs, for $i = 1, ..., M$ with $N0 = 0, NM = (N/21)$ (or $NM = (N1)/2$ if N is odd) is as follows:

1. Obtain $Z_{FFT}(k) = FFT(Z(t))$
2. Set $AFIBF_i = \sum_{k=N_{i-1}+1}^{N_i} Z_{FFT}(k)e^{\frac{j2\pi kn}{N}} = a_i(n)e^{j\theta_i(n)}$
3. Obtain the maximum value of N_i such that $N_{i-1}+1 \leq N_i \leq \frac{N}{2}-1$ and phase $\theta_i(n)$ of $AFIBF_i$ is a monotonically increasing function, that is, $\omega_i(n) = \frac{\theta_i(n+1)-\theta_i(n-1)}{2} \geq 0$,.

The advantage of the FDM method is that it is theoretically based on Fourier, different from EMD that is empirical.

A.2 Hilbert spectrum

The **Hilbert Spectrum** is a time-frequency representation of a signal obtained through the **Hilbert-Huang Transform (HHT)**, which is particularly suited for analyzing nonstationary and nonlinear signals. It provides a detailed depiction of how the instantaneous frequency content of a signal evolves over time, offering a high-resolution alternative to classical spectral analysis techniques such as the Fourier transform.

A.2.1 Definition

The Hilbert spectrum is derived by first decomposing a signal $x(t)$ into a set of **intrinsic mode functions (IMFs)** using **empirical mode decomposition (EMD)**.

Each IMF represents a mono-component oscillatory mode with well-defined instantaneous frequency characteristics. Once the signal is decomposed, the **Hilbert Transform** is applied to each IMF to obtain its **analytical signal**, from which the **instantaneous frequency** and **instantaneous amplitude** can be extracted.

Mathematically, the analytic signal corresponding to an IMF $c_i(t)$ is given by:

$$z_i(t) = c_i(t) + j\mathcal{H}[c_i(t)] \tag{A.7}$$

where $\mathcal{H}[c_i(t)]$ is the **Hilbert Transform** of $c_i(t)$. The instantaneous amplitude $a_i(t)$ and the instantaneous frequency $\omega_i(t)$ are computed as:

$$a_i(t) = \sqrt{c_i^2(t) + \mathcal{H}[c_i(t)]^2} \tag{A.8}$$

$$\omega_i(t) = \frac{d}{dt}\left(\tan^{-1}\left(\frac{\mathcal{H}[c_i(t)]}{c_i(t)}\right)\right) \tag{A.9}$$

The **Hilbert Spectrum**, denoted as $H(\omega, t)$, is then constructed by plotting the instantaneous frequency $\omega_i(t)$ against time with amplitude $a_i(t)$ as an intensity measure:

$$H(\omega, t) = \sum_{i=1}^{N} a_i(t)\delta(\omega - \omega_i(t)) \tag{A.10}$$

where N is the number of IMFs.

A.2.2 Example

The first plot in Figure 36 represents a composite signal containing multiple frequency components.

- The waveform exhibits varying oscillation speeds, indicating dynamically changing frequency content.
- Higher frequency oscillations appear in some sections, while others display slower variations.
- This confirms that the signal is non-stationary, meaning that its frequency content changes over time.

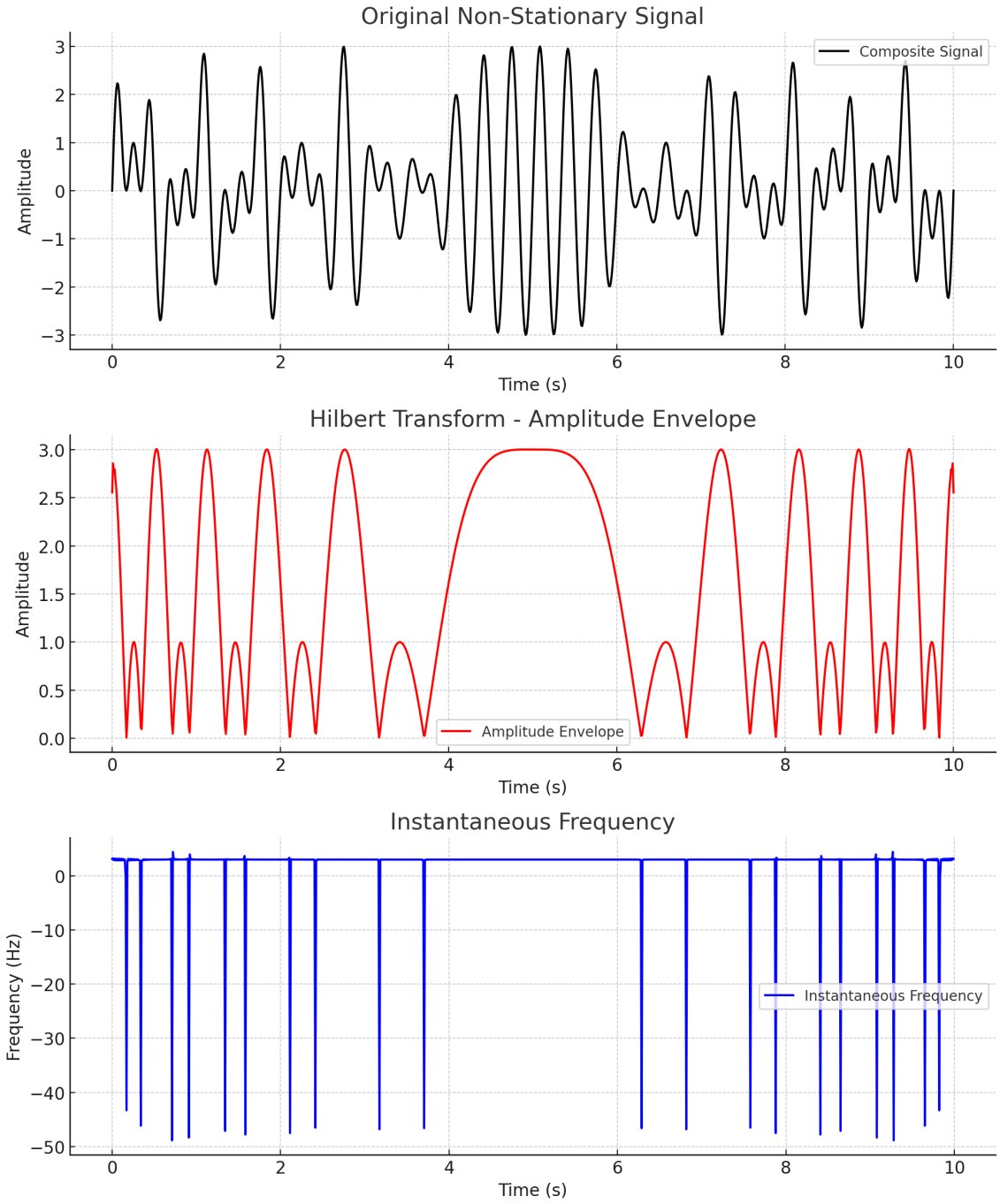


Figure 36: Arbitrary example signal for demonstration of HHT.

- Classical Fourier analysis would struggle to fully capture these transient frequency changes.

The second plot in Figure 36 shows the amplitude envelope extracted from the analytic signal.

- The red curve represents the instantaneous amplitude of the signal.
- It highlights variations in signal intensity over time.
- High envelope amplitude corresponds to high-energy portions of the signal, while low amplitude regions indicate quieter sections.
- This information is valuable in applications such as speech processing, biomedical signal analysis (e.g., EEG/ECG), and mechanical fault detection.

The third plot in Figure 36 presents the instantaneous frequency obtained via the Hilbert Transform.

- The blue curve shows how the dominant frequency evolves over time.
- Unlike Fourier-based methods that provide only a global frequency representation, the Hilbert transform reveals time-resolved frequency variations.
- Some sections exhibit high instantaneous frequencies, corresponding to rapid oscillations, while other regions have lower frequencies.
- The smooth transitions in frequency confirm the nonstationary nature of the signal.

A.2.3 Summary

Unlike classical spectral methods, the Hilbert transform provides a fine-grained view of frequency changes over time. It also features the amplitude envelope, which captures variations in intensity, making it useful for analyzing transient phenomena. And finally, it handles non-Stationary Behavior, in other words, it can accurately represent a signal composed of dynamically evolving components with multiple local frequencies, in contrast to a single global frequency approach.

The Hilbert transform is useful in various fields, such as analyzing brain waves (EEG) or heart signals (ECG) in biomedical engineering; detecting mechanical faults in rotating

machinery for structural health monitoring; understanding time-dependent oscillations in earthquake data for seismology and geophysics fields; examining pitch and tone variations in spoken language for speech and audio processing.

In summary, the Hilbert Transform and Hilbert Spectrum provide an adaptive time-frequency representation of signals, enabling precise analysis of nonstationary processes. By extracting instantaneous amplitude and frequency information, these tools offer a significant advantage over traditional Fourier-based methods, making them highly relevant in various scientific and engineering applications.

A.3 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used to reduce the complexity of high-dimensional data while retaining the essential patterns. The idea is to transform the data into a new set of orthogonal axes, called principal components, which capture the maximum variance in the data. These principal components are linear combinations of the original variables, sorted in descending order based on the variance they explain. PCA is typically used for exploratory data analysis, pattern recognition, and noise reduction.

Mathematically, given a dataset X of shape $n \times p$, where n is the number of observations and p is the number of features, PCA begins by standardizing the data (if necessary) to have zero mean and unit variance. Next, the covariance matrix C of the data is computed as:

$$C = \frac{1}{n-1} X^T X$$

The next step is to compute the eigenvalues and eigenvectors of this covariance matrix. Eigenvectors represent the directions (or principal components), while eigenvalues indicate the amount of variance captured by each component. The eigenvector with the largest eigenvalue corresponds to the first principal component, and so on.

To project the data onto the principal components, the original data X is multiplied by the matrix of eigenvectors. The resulting projection will have a reduced dimensionality depending on how many principal components you choose to keep. Formally, the data projection onto the first k principal components is given by:

$$Z = XW_k$$

where W_k is the matrix of the first k eigenvectors. The new dataset Z has reduced dimensions (from p to k), with the largest amount of variance captured by the first few components.

A.4 Wavelets

Wavelets are derived from the Fourier transform but take into account a limited rolling time window.

A certain signal $Z(t)$ can be described as the sum of the orthogonal wavelet basis functions, weighted by the wavelet transform coefficients.

$$Z(t) = \sum_{j,k} \gamma(j,k)\psi_{j,k}(t) \tag{A.11}$$

Where ψ represents a generical wavelet basis function, that is defined as $\psi_{j,k}(t) = 2^{\frac{j}{2}}\psi\left(2^j t - k\right)$. Some wavelet examples are the following:

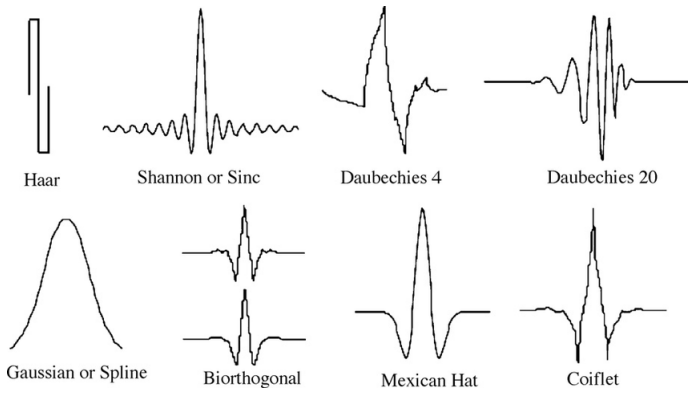


Figure 37: Examples of Wavelet basis functions (Source: Alan Godfrey)

To obtain the resulting function of a certain wavelet when resizing it horizontally and rolling it horizontally along x axis we follow the *reference*.

The wavelet $\psi_{a,b}$ is defined according to its scale a and location b.

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{A.12}$$

The parameter j, k can be discretized as:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{a_0^m}}\psi\left(\frac{t - nb_0a_0^m}{a_0^m}\right) \tag{A.13}$$

Which in turn, substituting $a_0 = 2$ and $b_0 = 1$ as a dyadic grid scale, which is one of the simplest and most efficient forms of choosing parameters such that they leads to:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t - k2^j}{2^k}\right) \tag{A.14}$$

$$\psi_{j,k}(t) = 2^{-j/2}\psi\left(2^{-j}t - k\right) \tag{A.15}$$

APPENDIX B – EFFICIENT MARKET

The efficient market hypothesis states that the information of stocks and market products in general is well distributed, so well that no one could take advantage of the market in terms of returns (Ţiţan, 2015).

It has 3 forms: the weak hypothesis, which accounts for the distribution of information about all the historical prices and returns across the entire market; the semi-strong hypothesis, which accounts for the distribution of public information such as the factors that influence a certain sector (e.g. climate data for agriculture or quarterly results and fundamentalist data); and finally the last form that is the strong efficient market hypothesis, which considers that even the private information is also well distributed so that every market player can tell exactly all about a certain company or product.

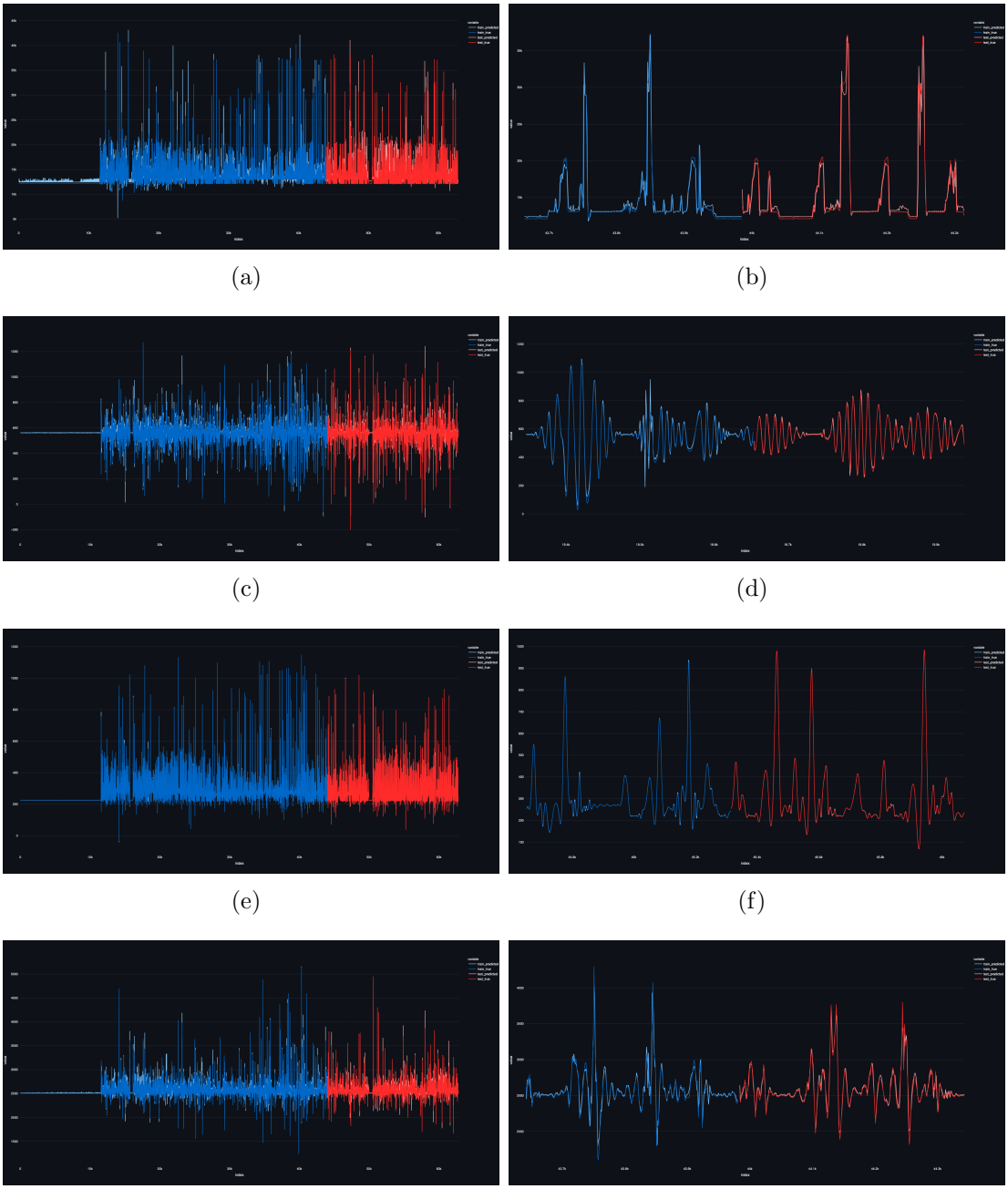
Although that is a concern and a well known hypothesis, Maloumian (2022) argues that forecasting is indeed possible. We bring the exact text of his paper as mention for justification for the research goal of predicting market behavior.

"*The link between information flows or shocks and price movements is at best only correlative, not causal. Forecasting is therefore in essence not impossible; markets could potentially offer profit opportunities un- related to inefficiency or even to what is defined as value. Finally, the clear emergence of observed patterns suggests that these forms, in particular Elliott waves, constitutes the outcome of the complexity at work. They should be considered as such with a renewed interest since they could provide a better appraisal of market risks and price behaviors.*" - excerpt from Maloumian (2022).

APPENDIX C – PHASE 2 ADDITIONAL TABLES AND FIGURES

Here all intermediate CEEMDAN components predictions for each data set of phase 2 is shown for the sake of completeness in Figures 38, 39, 40 and 41.

The tables 14, 15, 16 and 17 that summarize the accuracy results for the ablation tests are also available.



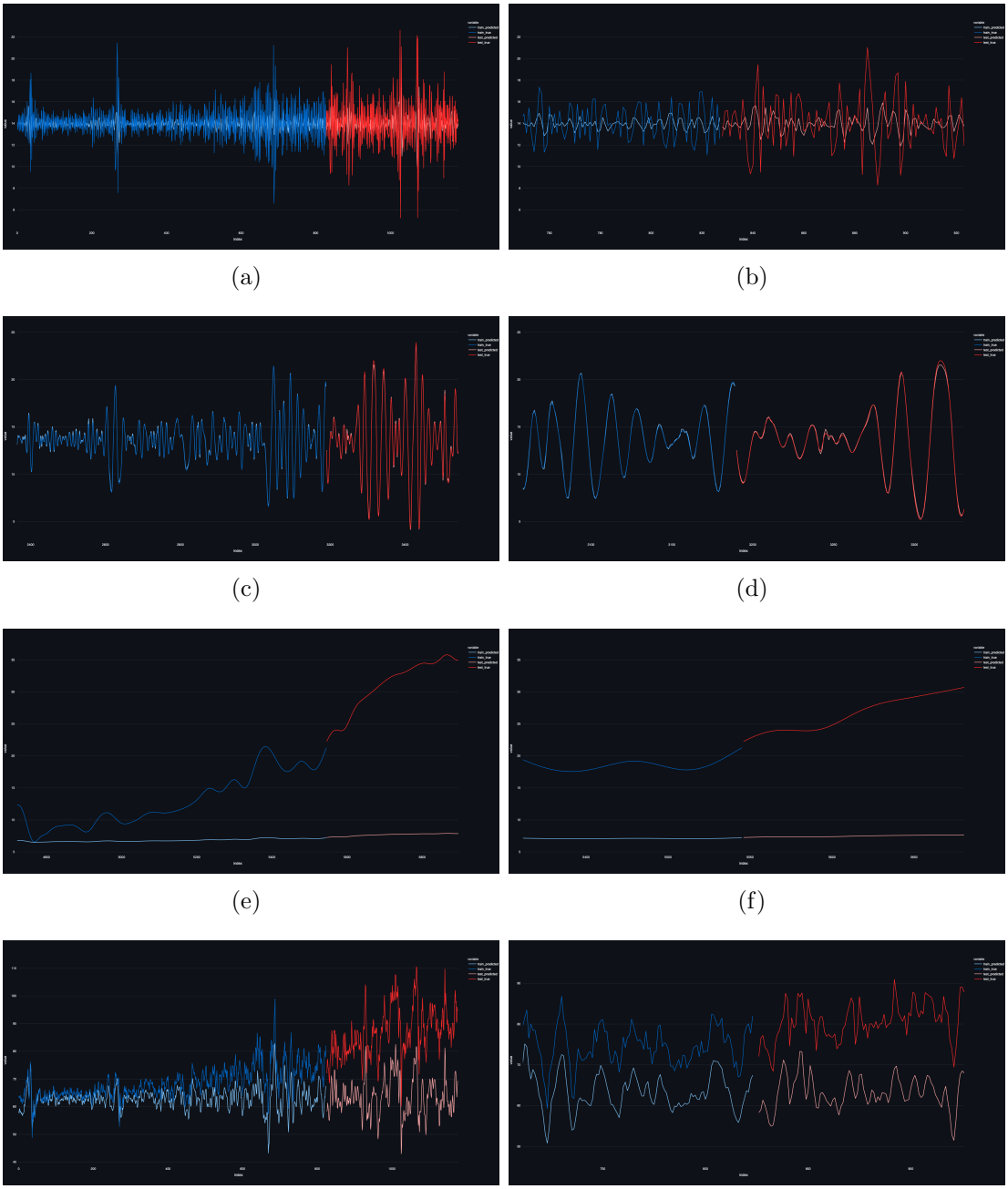
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 38: Example of energy Load dataset target feature prediction vs truth values experiment result: (a) IMF0 full resampled length; (b) IMF0 train/test limit window; (c) IMF2 full resampled length; (d) IMF2 train/test limit window; (e) Residue full resampled length; (f) Residue train/test limit window;



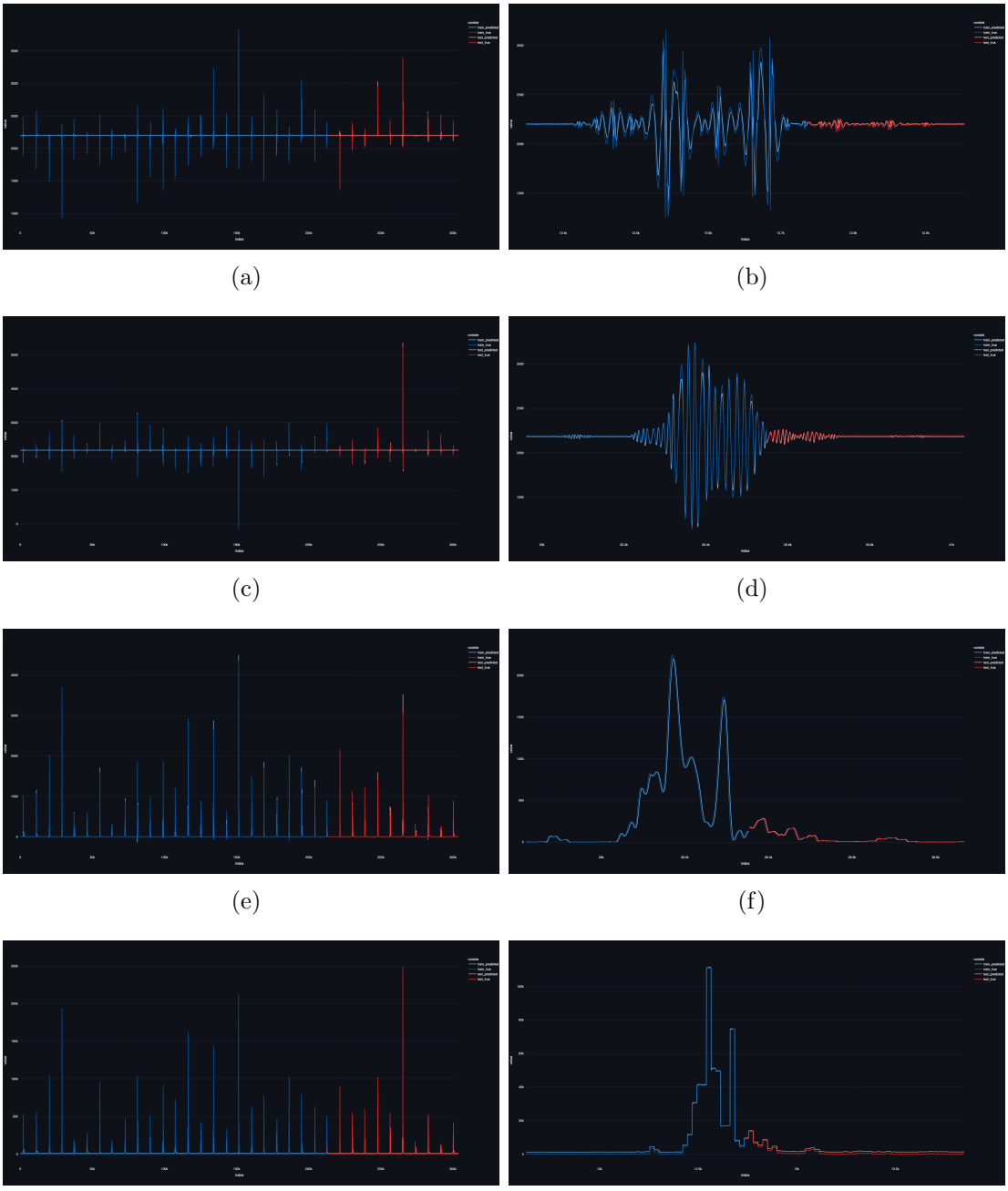
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 39: Example Finance dataset target feature prediction vs truth values experiment: (a) IMF0 full resampled length; (b) IMF0 train/test limit window; (c) IMF2 full resampled length; (d) IMF2 train/test limit window; (e) Residue full resampled length; (f) Residue train/test limit window;



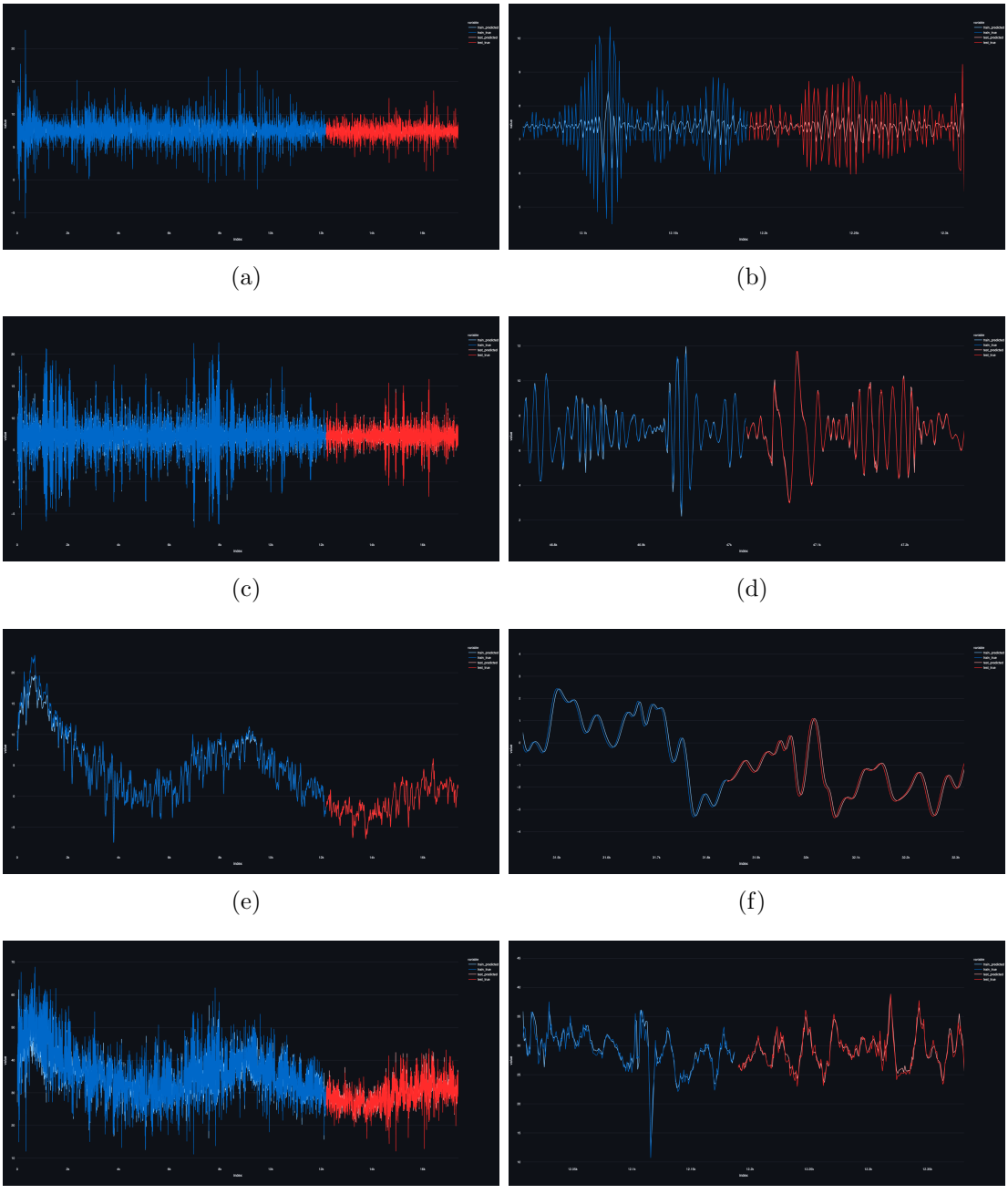
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 40: Example Wind dataset target feature prediction vs truth values experiment: (a) IMF0 full resampled length; (b) IMF0 train/test limit window; (c) IMF2 full resampled length; (d) IMF2 train/test limit window; (e) Residue full resampled length; (f) Residue train/test limit window;



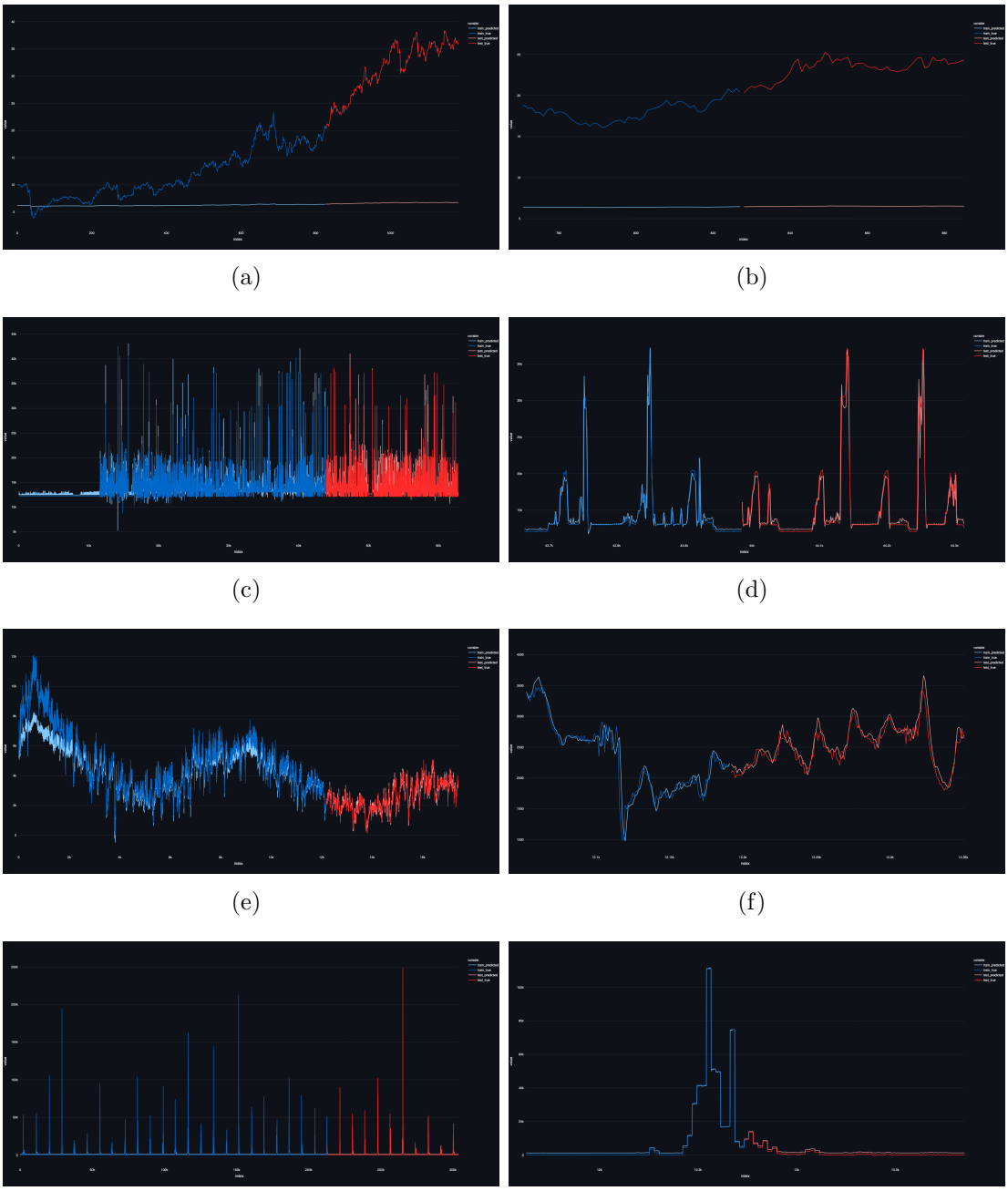
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 41: Example ETTh1 dataset target feature prediction vs truth values experiment: (a) IMF0 full resampled length; (b) IMF0 train/test limit window; (c) IMF2 full resampled length; (d) IMF2 train/test limit window; (e) Residue full resampled length; (f) Residue train/test limit window;



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 42: Example from, top to bottom, Finance, Energy Load, ETTh1 and Wind datasets target features prediction vs truth values experiment without using CEEMDAN for decomposition: (a) Finance full resampled length; (b) Finance train/test limit window; (c) Energy Load full resampled length; (d) Energy Load train/test limit window; (e) Wind full resampled length; (f) Wind train/test limit window;

features	NoFeatures	target_feature	ceemdan	final_rmse_test	final_mape_test	final_mae_test	final_rmse_train	final_mape_train	final_mae_train
{""}	0	pwr.cons.w	TRUE	6248,627971	0,0206412	5609,01309	26499,4661	0,2461666	21229,17642
{bat.load.w}	1	pwr.cons.w	TRUE	325,4334619	2,1386913	176,868945	290,0119817	0,4987163	166,9004976
{lux.south.lx}	1	pwr.cons.w	TRUE	6530,261523	0,0165253	3519,515889	5799,568451	0,0155542	3290,538503
{lux.west.lx}	1	pwr.cons.w	TRUE	7354,513073	0,0229459	4243,374714	6601,680822	0,0218752	4009,427282
{p.amb.in.hp}	1	pwr.cons.w	TRUE	75,7868497	**0,0150693**	**39,3175547**	66,9732692	**0,0140071**	**36,3465207**
{lux.south.lx,bat.load.w}	2	pwr.cons.w	TRUE	6702,032084	0,0164825	3549,941599	5985,270914	0,0157926	3379,703212
{lux.west.lx,bat.load.w}	2	pwr.cons.w	TRUE	7613,189194	0,0212483	4055,41121	6816,223192	0,0198689	3774,859335
{lux.west.lx,lux.south.lx}	2	pwr.cons.w	TRUE	7300,048347	0,0233756	3971,900758	6506,955164	0,022088	3723,649614
{p.amb.in.hp,bat.load.w}	2	pwr.cons.w	TRUE	**74,2627627**	0,0161193	42,2252631	**66,6762412**	0,0152114	39,6087574
{p.amb.in.hp,lux.south.lx}	2	pwr.cons.w	TRUE	6833,843967	0,0179605	3613,518879	6076,046603	0,0167988	3358,473753
{p.amb.in.hp,lux.west.lx}	2	pwr.cons.w	TRUE	7113,628782	0,0202175	3949,64379	6355,320427	0,0190544	3693,682617
{lux.west.lx,lux.south.lx,bat.load.w}	3	pwr.cons.w	TRUE	7621,34557	0,0184212	3986,826905	6681,779415	0,0172741	3716,323117
{p.amb.in.hp,lux.south.lx,bat.load.w}	3	pwr.cons.w	TRUE	6815,294873	0,0173293	3589,614144	5973,811546	0,0161529	3326,723015
{p.amb.in.hp,lux.west.lx,bat.load.w}	3	pwr.cons.w	TRUE	6807,304647	0,0164358	3608,106421	5932,549744	0,0151871	3320,645736
{p.amb.in.hp,lux.west.lx,lux.south.lx}	3	pwr.cons.w	TRUE	7635,214105	0,020352	3958,333289	6735,606822	0,0192734	3713,540747
{p.amb.in.hp,lux.west.lx,lux.south.lx,bat.load.w}	4	pwr.cons.w	TRUE	6748,67358	0,0190981	3607,829844	6050,79829	0,0181145	3395,7346
{""}	0	pwr.cons.w	FALSE	1474,481297	0,0394546	620,7474708	1323,998797	0,0415793	606,0041993
{bat.load.w}	1	pwr.cons.w	FALSE	1467,215286	0,0394291	620,5787739	1326,340672	0,0435029	633,6216882
{lux.south.lx}	1	pwr.cons.w	FALSE	1451,864147	0,0386324	612,7347213	1315,560138	0,0417264	615,2229652
{lux.west.lx}	1	pwr.cons.w	FALSE	1463,196408	0,0368101	589,4442347	1321,89551	**0,0392898**	**584,0012972**
{p.amb.in.hp}	1	pwr.cons.w	FALSE	1477,158053	0,0423061	661,0126657	1317,086929	0,0430755	641,8364471
{lux.south.lx,bat.load.w}	2	pwr.cons.w	FALSE	1455,01265	0,0391965	617,2692133	1324,023183	0,0443397	651,3770361
{lux.west.lx,bat.load.w}	2	pwr.cons.w	FALSE	1455,499991	0,0385143	607,1286951	1327,162747	0,0439779	644,0927998
{lux.west.lx,lux.south.lx}	2	pwr.cons.w	FALSE	1453,344003	0,0384044	608,0718108	1323,500367	0,0425053	626,7292285
{p.amb.in.hp,bat.load.w}	2	pwr.cons.w	FALSE	1469,812214	0,0423487	658,6534578	1329,203853	0,0461093	679,2268679
{p.amb.in.hp,lux.south.lx}	2	pwr.cons.w	FALSE	1460,066844	0,0422459	662,194013	1316,55505	0,0440112	652,4791009
{p.amb.in.hp,lux.west.lx}	2	pwr.cons.w	FALSE	1498,948171	0,0412968	651,170755	1331,395282	0,0400957	598,8529564
{lux.west.lx,lux.south.lx,bat.load.w}	3	pwr.cons.w	FALSE	1465,514849	**0,0349143**	**561,0650322**	1329,522154	0,0416271	590,1792081
{p.amb.in.hp,lux.south.lx,bat.load.w}	3	pwr.cons.w	FALSE	1518,743051	0,0438661	688,3716077	1351,012807	0,0448531	663,0901998
{p.amb.in.hp,lux.west.lx,bat.load.w}	3	pwr.cons.w	FALSE	**1449,946228**	0,0392495	618,9113873	1322,776388	0,0440829	650,8913316
{p.amb.in.hp,lux.west.lx,lux.south.lx}	3	pwr.cons.w	FALSE	1450,66259	0,0399574	627,415239	**1313,314657**	0,0427842	633,3683808
{p.amb.in.hp,lux.west.lx,lux.south.lx,bat.load.w}	4	pwr.cons.w	FALSE	1450,481127	0,0376414	595,0353749	1323,531714	0,0429564	632,6658701

Table 14: Energy load dataset ablation test

features	NoFeatures	target_feature	ceemdan	final_rmse_test	final_mape_test	final_mae_test	final_rmse_train	final_mape_train	final_mae_train
{""}	0	OT	TRUE	652,866428	**0,0151576**	650,5979667	1512,889187	0,1297442	1132,269518
{HUFL}	1	OT	TRUE	1,3515607	0,0324812	1,0042698	2,2847638	0,0446259	1,6540526
{HULL}	1	OT	TRUE	0,5188568	0,0616422	0,3947363	1,6995358	0,1616655	1,2652183
{LUFL}	1	OT	TRUE	0,3815923	0,0158013	0,2867735	0,6230673	**0,0225943**	0,4523407
{LULL}	1	OT	TRUE	**0,1478283**	0,0704172	**0,1099908**	**0,2538072**	0,1015384	**0,1833576**
{HUFL,HULL}	2	OT	TRUE	1,2893805	0,0312643	0,9584429	2,1653729	0,0427963	1,5661409
{HUFL,LUFL}	2	OT	TRUE	1,225376	0,0315899	0,9058926	2,1151585	0,0442215	1,5258186
{HUFL,LULL}	2	OT	TRUE	4,4123939	0,1120934	3,7353122	6,5169903	0,1488785	4,8520062
{HULL,LUFL}	2	OT	TRUE	0,4703241	0,0533454	0,3493511	0,779745	0,0666585	0,5614146
{HULL,LULL}	2	OT	TRUE	0,5700948	0,0878009	0,4310374	1,4584939	0,1640623	1,0401129
{LUFL,LULL}	2	OT	TRUE	0,3850976	0,0165633	0,2857806	0,6279969	0,0237036	0,4535655
{HUFL,HULL,LUFL}	3	OT	TRUE	1,3595189	0,0368624	1,0115551	2,221076	0,0481688	1,6067242
{HUFL,HULL,LULL}	3	OT	TRUE	1,4123243	0,03221	1,0494295	2,4331012	0,0445785	1,7511202
{HUFL,LUFL,LULL}	3	OT	TRUE	1,3894624	0,0342052	1,0317445	2,3514569	0,0471045	1,6949481
{HULL,LUFL,LULL}	3	OT	TRUE	0,4534478	0,152777	0,3406384	1,2407936	0,1035427	0,8715021
{HUFL,HULL,LUFL,LULL}	4	OT	TRUE	1,3593062	0,0311829	1,0192247	3,3774807	0,0605833	2,4121984
{""}	0	OT	FALSE	**163,0098236**	**0,0531609**	**115,5113975**	418,1292573	**0,0589106**	284,3625468
{HUFL}	1	OT	FALSE	177,5384659	0,0627109	130,3057434	449,1385117	0,0634323	313,4397482
{HULL}	1	OT	FALSE	175,8252575	0,0559853	125,3432752	478,8142965	0,0721873	349,3800883
{LUFL}	1	OT	FALSE	179,2987138	0,0951741	128,7016358	642,949187	0,0835123	410,4019956
{LULL}	1	OT	FALSE	222,1668438	0,078368	170,2668435	**335,4563323**	0,0589954	**254,5089785**
{HUFL,HULL}	2	OT	FALSE	184,6793512	0,0691929	135,263484	853,2133984	0,1113563	534,9773179
{HUFL,LUFL}	2	OT	FALSE	1018,090682	0,2812836	848,3632471	2810,236636	0,7124876	2144,378921
{HUFL,LULL}	2	OT	FALSE	188,0098573	0,0630223	141,4215656	528,3461083	0,0820953	399,391709
{HULL,LUFL}	2	OT	FALSE	177,5540469	0,0579181	126,9760601	623,9584577	0,0928822	419,5102416
{HULL,LULL}	2	OT	FALSE	191,6875948	0,0619239	141,9588664	719,8613578	0,100953	508,9021437
{LUFL,LULL}	2	OT	FALSE	165,8810464	0,0550416	117,2517976	503,2369175	0,0684754	341,715614
{HUFL,HULL,LUFL}	3	OT	FALSE	182,6479013	0,0725752	132,9109339	814,6303722	0,1048897	511,0344372
{HUFL,HULL,LULL}	3	OT	FALSE	209,4192803	0,0949186	164,6282015	808,2721337	0,1550836	543,8104622
{HUFL,LUFL,LULL}	3	OT	FALSE	182,2978682	0,0610604	136,6325706	548,6588133	0,083363	409,0055121
{HULL,LUFL,LULL}	3	OT	FALSE	212,6176428	0,0726572	161,5620306	371,1150909	0,0670874	291,1774633
{HUFL,HULL,LUFL,LULL}	4	OT	FALSE	206,5768939	0,0767814	161,6748532	825,041566	0,1135799	558,9065221

Table 15: ETTh1 dataset ablation test

features	NoFeatures	target_feature	ceemdan	final_rmse_test	final_mape_test	final_mae_test	final_rmse_train	final_mape_train	final_mae_train
{""}	0	Close	TRUE	65,2262303	**0,0098941**	53,0284588	220,7824728	0,1457784	156,4581144
{High}	1	Close	TRUE	25,9904793	0,4107939	25,396457	7,67359	0,1004144	6,1314273
{Low}	1	Close	TRUE	24,5695808	0,407513	23,9601417	6,6641996	0,0883742	5,1329214
{Open}	1	Close	TRUE	26,817086	0,4481704	26,2462029	8,0339726	0,1120876	6,5613125
{Volume}	1	Close	TRUE	24,8440434	0,3968807	24,2334038	6,5242766	0,0818646	4,9461596
{High,Low}	2	Close	TRUE	27,747245	0,457166	27,1885686	8,7320402	0,1247916	7,3429113
{High,Volume}	2	Close	TRUE	32,660684	0,5855735	31,8383486	12,4923604	0,2088967	11,2770369
{Low,Volume}	2	Close	TRUE	27,128905	0,4726409	26,5825538	8,3989886	0,1243757	7,0187478
{Open,High}	2	Close	TRUE	31,5822182	0,5627359	30,8326142	11,054623	0,1771894	9,7163175
{Open,Low}	2	Close	TRUE	24,5440871	0,3941448	24,0314108	7,4378134	0,1003225	6,1020565
{Open,Volume}	2	Close	TRUE	23,6009898	0,3646458	23,1174329	7,2235523	0,0956479	5,9711977
{High,Low,Volume}	3	Close	TRUE	34,0466543	0,6330928	33,3670568	12,6169533	0,2147967	11,3204799
{Open,High,Low}	3	Close	TRUE	22,8849881	0,3473345	22,3553385	**6,2819632**	**0,0761204**	**4,8825368**
{Open,High,Volume}	3	Close	TRUE	25,3388207	0,4241616	24,7223412	6,6969651	0,0873733	5,091862
{Open,Low,Volume}	3	Close	TRUE	25,5576859	0,4005353	25,0745486	8,286516	0,1140338	7,0738436
{Open,High,Low,Volume}	4	Close	TRUE	**22,7624068**	0,352022	**22,2881913**	7,0033568	0,0925115	5,8045048
{""}	0	Close	FALSE	24,8840502	3,5616321	24,4503823	6,9939019	0,8530972	5,5526567
{High}	1	Close	FALSE	23,5081446	2,7877397	23,0491381	5,9809593	0,5685633	4,490157
{Low}	1	Close	FALSE	25,0378154	3,6643468	24,5871768	6,8410917	0,7939386	5,3321674
{Open}	1	Close	FALSE	26,237075	4,7048448	25,8224362	8,0549283	1,3034247	6,7635605
{Volume}	1	Close	FALSE	25,1613071	3,7551605	24,7228986	7,0767716	0,8787668	5,6285102
{High,Low}	2	Close	FALSE	27,4624552	4,5474459	27,1149934	7,9706318	1,0388547	6,3537992
{High,Volume}	2	Close	FALSE	1,1409459	0,0267374	0,8886215	0,6146358	0,045971	0,4665387
{Low,Volume}	2	Close	FALSE	**0,7410814**	0,0175541	**0,5396181**	0,6541634	0,0510867	0,5168239
{Open,High}	2	Close	FALSE	24,1938659	3,1461	23,7812482	6,9582047	0,8591182	5,6123136
{Open,Low}	2	Close	FALSE	25,0500887	3,676406	24,6175776	7,1292498	0,902626	5,7162249
{Open,Volume}	2	Close	FALSE	25,3903432	3,9298136	24,9563795	7,2557123	0,9431868	5,825741
{High,Low,Volume}	3	Close	FALSE	0,9109977	0,0212489	0,6856021	**0,5361769**	**0,0391537**	**0,4043551**
{Open,High,Low}	3	Close	FALSE	0,766522	**0,0175511**	0,554529	0,5778794	0,0442673	0,4491269
{Open,High,Volume}	3	Close	FALSE	26,8856315	4,0678725	26,5816109	8,675352	1,4253858	7,4141946
{Open,Low,Volume}	3	Close	FALSE	27,4940623	4,5934473	27,1925312	8,9976067	1,5996767	7,7295017
{Open,High,Low,Volume}	4	Close	FALSE	24,4268483	3,2714091	23,9830382	6,613839	0,7316024	5,1292876

Table 16: Finance dataset ablation test

features	NoFeatures	target_feature	ceemdan	final_rmse_test	final_mape_test	final_mae_test	final_rmse_train	final_mape_train	final_mae_train
{""}	0	birch.npm3	TRUE	17654,93996	0,0108749	14681,30596	76885,81368	0,1770473	54307,46391
{precip.mmsum}	1	birch.npm3	TRUE	39,8471857	0,0010799	9,6139134	45,0912936	0,0011993	10,7256396
{snow.cmsum}	1	birch.npm3	TRUE	35,7643445	0,0014609	12,5925685	40,19657	0,0015718	13,5990058
{temp.c}	1	birch.npm3	TRUE	**26,3536735**	0,0009289	8,4602146	**27,4456933**	**0,0009851**	**8,9913921**
{wind.mps}	1	birch.npm3	TRUE	34,5296584	0,0011773	10,3970776	37,6057413	0,0012792	11,3325263
{snow.cmsum,precip.mmsum}	2	birch.npm3	TRUE	45,9077602	0,0013433	11,4162654	53,0092455	0,0014898	12,7003192
{temp.c,precip.mmsum}	2	birch.npm3	TRUE	40,7940281	0,0011826	11,4841246	46,1909737	0,0012808	12,4828045
{temp.c,snow.cmsum}	2	birch.npm3	TRUE	49,0070194	0,0011271	9,3765547	55,1749752	0,0012987	10,8128216
{temp.c,wind.mps}	2	birch.npm3	TRUE	35,9005492	0,0010121	9,1606442	39,2494983	0,0011131	10,0756971
{wind.mps,precip.mmsum}	2	birch.npm3	TRUE	45,7796032	0,0028016	25,7396018	50,8691704	0,0029075	26,7827958
{wind.mps,snow.cmsum}	2	birch.npm3	TRUE	48,7580136	0,0032213	28,7081625	54,1786885	0,0033382	29,8267646
{temp.c,snow.cmsum,precip.mmsum}	3	birch.npm3	TRUE	54,0341556	0,0022906	19,7140999	59,6350865	0,0024437	21,0800553
{temp.c,wind.mps,precip.mmsum}	3	birch.npm3	TRUE	43,7597177	0,0010093	9,0290031	50,1296944	0,0011399	10,2262884
{temp.c,wind.mps,snow.cmsum}	3	birch.npm3	TRUE	46,8444221	0,0027535	25,0090828	53,4492997	0,002888	26,2874462
{wind.mps,snow.cmsum,precip.mmsum}	3	birch.npm3	TRUE	48,5519154	0,0012495	11,0635271	56,9285696	0,001413	12,5696155
{temp.c,wind.mps,snow.cmsum,precip.mmsum}	4	birch.npm3	TRUE	44,2993772	**0,0009092**	**8,2152112**	50,8296862	0,0010433	9,4522101
{""}	0	birch.npm3	FALSE	**1564,61847**	0,9173623	821,2621794	**1725,622528**	0,9203173	831,2218376
{precip.mmsum}	1	birch.npm3	FALSE	1593,450294	0,9184304	869,6266319	1748,686017	0,9211932	879,2276783
{snow.cmsum}	1	birch.npm3	FALSE	1687,245884	0,9073602	503,9842279	1892,601582	0,9127418	524,0601131
{temp.c}	1	birch.npm3	FALSE	1742,294695	0,9228984	1115,348395	1877,473761	0,9249307	1114,589989
{wind.mps}	1	birch.npm3	FALSE	1628,526201	0,9192169	906,7464005	1774,162636	0,9218716	917,9065819
{snow.cmsum,precip.mmsum}	2	birch.npm3	FALSE	1671,531531	0,9211562	1006,616495	1823,354193	0,923451	1014,438128
{temp.c,precip.mmsum}	2	birch.npm3	FALSE	1693,58478	**0,9057001**	460,4906907	1880,290041	**0,9116399**	486,0352652
{temp.c,snow.cmsum}	2	birch.npm3	FALSE	1762,530147	0,9234291	1145,027478	1897,030065	0,9253867	1141,69671
{temp.c,wind.mps}	2	birch.npm3	FALSE	1851,298144	0,924761	1266,045139	1970,847911	0,9264801	1262,320058
{wind.mps,precip.mmsum}	2	birch.npm3	FALSE	1947,28466	0,9232718	1144,952542	2135,431176	0,9256002	1163,524561
{wind.mps,snow.cmsum}	2	birch.npm3	FALSE	1864,00346	0,9244524	1262,728139	1981,461269	0,9261861	1270,255802
{temp.c,snow.cmsum,precip.mmsum}	3	birch.npm3	FALSE	1568,579597	1,0009106	**110,1751295**	1779,732179	0,9749137	**125,4786427**
{temp.c,wind.mps,precip.mmsum}	3	birch.npm3	FALSE	2250,610785	0,9312498	1620,220873	2406,390448	0,9323979	1621,116621
{temp.c,wind.mps,snow.cmsum}	3	birch.npm3	FALSE	1737,425028	0,9222311	1097,216195	1869,432784	0,924352	1097,776638
{wind.mps,snow.cmsum,precip.mmsum}	3	birch.npm3	FALSE	1690,882173	0,9215209	1034,267652	1839,937744	0,9237408	1043,796042
{temp.c,wind.mps,snow.cmsum,precip.mmsum}	4	birch.npm3	FALSE	1729,773002	0,9230759	1101,361138	1870,554061	0,9251536	1103,026033

Table 17: Wind dataset ablation test

Dataset	Target	Exogenous	correlation	FARM_global	DTW distance	FastDTW distance	FastDDTW distance	DDTW distance	MSE	Euclidean
Finance	**Close**	**Open**	0,999525606	0,668167869	8,49244831	6,151753756	**3,553553427**	**3,549565901**	0,000108887	0,3667097
Finance	**Close**	**High**	0,999784655	0,795645309	**6,486430603**	**4,344263956**	3,757299522	3,75672244	0,000047900	0,2432214
Finance	**Close**	**Low**	**0,999809453**	**0,79592955**	7,20576307	4,958074742	3,629750406	3,629392453	**0,000038141**	**0,217035613**
Finance	**Close**	**Volume**	-0,437937973	0,489800796	386,5463648	362,9366553	35,36997358	35,1288865	0,193820153	15,47151864
Energy Load	**pwr.cons.w**	**p.amb.in.hp**	0,02114368	0,363567029	NA	40554,89415	**662,0751742**	NA	0,440770328	166,4271835
Energy Load	**pwr.cons.w**	**lux.west.lx**	0,195586468	0,578459267	NA	**16223,92056**	1139,912452	NA	0,082596745	72,04428804
Energy Load	**pwr.cons.w**	**lux.south.lx**	**0,265334641**	**0,592908657**	NA	16245,34101	1280,252824	NA	0,084182964	72,73278111
Energy Load	**pwr.cons.w**	**bat.load.w**	0,251231154	0,573436953	NA	8715,336729	1539,972331	NA	**0,041945334**	**51,34047906**
ETTh1	**OT**	**HUFL**	0,059916472	0,471834071	2107,155749	3979,843387	503,4706728	462,961652	0,140287872	49,43495447
ETTh1	**OT**	**HULL**	**0,224353831**	0,497973139	**1414,940264**	2498,887102	429,8967204	405,0196427	0,052739353	30,3103865
ETTh1	**OT**	**LUFL**	0,118836249	0,51179053	1467,772908	**2186,361085**	418,7141915	398,7610233	**0,047230114**	**28,68359444**
ETTh1	**OT**	**LULL**	0,067455316	**0,516089237**	1735,562298	3035,166208	**277,1209036**	**261,380772**	0,069165958	34,7112516
Wind	**birch.npm3**	**temp.c**	**0,026349192**	**0,069637885**	NA	144959,6377	3445,833437	NA	0,281802085	292,6483134
Wind	**birch.npm3**	**wind.mps**	-0,012234092	0,065519026	NA	27579,22742	7054,631345	NA	0,019728428	77,43194397
Wind	**birch.npm3**	**snow.cmsum**	-0,003467746	0,000073644	NA	1388,896492	**120,4671872**	NA	**0,001307637**	**19,93505711**
Wind	**birch.npm3**	**precip.mmsum**	-0,014264063	0,010648158	NA	**1345,325518**	376,6412221	NA	0,001331443	20,1157006

Table 18: Original relevance metrics for all combinations of features when compared to respective targets, notice all values range from 0 to 1.

dataset	target_feature	exogenous_feature	rank_correlation	rank_FARM_global	rank_FastDTW	rank_FastDDTW	rank_Euclidean	average_rank
ETTh1	**OT**	**LUFL**	2	2	**1**	2	**1**	1.6
ETTh1	**OT**	**HULL**	**1**	3	2	3	2	2.2
ETTh1	**OT**	**LULL**	3	**1**	3	**1**	3	2.2
ETTh1	**OT**	**HUFL**	4	4	4	4	4	4.0
Energy Load	**pwr.cons.w**	**lux.west.lx**	3	2	2	2	2	2.2
Energy Load	**pwr.cons.w**	**lux.south.lx**	**1**	**1**	3	3	3	2.2
Energy Load	**pwr.cons.w**	**bat.load.w**	2	3	**1**	4	**1**	2.2
Energy Load	**pwr.cons.w**	**p.amb.in.hp**	4	4	4	**1**	4	3.4
Finance	**Close**	**Low**	**1**	**1**	2	2	**1**	1.4
Finance	**Close**	**High**	2	2	**1**	3	2	2.0
Finance	**Close**	**Open**	3	3	3	**1**	3	2.6
Finance	**Close**	**Volume**	4	4	4	4	4	4.0
Wind	**birch.npm3**	**precip.mmsum**	2	3	**1**	2	2	2.0
Wind	**birch.npm3**	**snow.cmsum**	4	4	2	**1**	**1**	2.4
Wind	**birch.npm3**	**temp.c**	**1**	**1**	4	3	4	2.6
Wind	**birch.npm3**	**wind.mps**	3	2	3	4	3	3.0

Table 19: Exogenous features ranks, ranked from 1 as the best to 4 the worse feature for each metric.

dataset	target_feature	exogenous_feature	ceemdan	rmse_improvement	farm_improvement	correlation_improvement	FastDTW distance_improvement	FastDDTW distance_improvement	Euclidean_improvement
ENERGY_LOAD	pwr.cons.w	bat.load.w	False	0.67%	57.73%	1088.21%	78.51%	0.00%	69.15%
ENERGY_LOAD	pwr.cons.w	lux.west.lx	False	0.95%	59.11%	825.04%	60.00%	25.98%	56.71%
ENERGY_LOAD	pwr.cons.w	lux.south.lx	False	1.71%	63.08%	1154.91%	59.94%	16.87%	56.30%
ENERGY_LOAD	pwr.cons.w	p.amb.in.hp	False	0.00%	0.00%	0.00%	0.00%	57.01%	0.00%
ETTh1	OT	LUFL	False	19.30%	8.47%	98.34%	45.06%	16.83%	41.98%
ETTh1	OT	HULL	False	20.86%	5.54%	274.44%	37.21%	14.61%	38.69%
ETTh1	OT	LULL	False	0.00%	9.38%	12.58%	23.74%	44.96%	29.78%
ETTh1	OT	HUFL	False	20.09%	0.00%	0.00%	0.00%	0.00%	0.00%
FINANCE	Close	Low	False	4.57%	62.50%	128.30%	98.63%	89.74%	98.60%
FINANCE	Close	High	False	10.40%	62.44%	128.29%	98.80%	89.38%	98.43%
FINANCE	Close	Open	False	0.00%	36.42%	128.23%	98.31%	89.95%	97.63%
FINANCE	Close	Volume	False	4.10%	0.00%	0.00%	0.00%	0.00%	0.00%
METEOSUISSE	birch.npm3	snow.cmsum	False	3.16%	0.00%	0.00%	99.04%	98.29%	93.19%
METEOSUISSE	birch.npm3	precip.mmsum	False	8.54%	14358.96%	311.34%	99.07%	94.66%	93.13%
METEOSUISSE	birch.npm3	wind.mps	False	6.53%	88867.23%	252.80%	80.97%	0.00%	73.54%
METEOSUISSE	birch.npm3	temp.c	False	0.00%	94460.16%	659.84%	0.00%	51.16%	0.00%
ENERGY_LOAD	pwr.cons.w	bat.load.w	True	95.58%	57.73%	1088.21%	78.51%	0.00%	69.15%
ENERGY_LOAD	pwr.cons.w	lux.west.lx	True	0.00%	59.11%	825.04%	60.00%	25.98%	56.71%
ENERGY_LOAD	pwr.cons.w	lux.south.lx	True	11.21%	63.08%	1154.91%	59.94%	16.87%	56.30%
ENERGY_LOAD	pwr.cons.w	p.amb.in.hp	True	98.97%	0.00%	0.00%	0.00%	57.01%	0.00%
ETTh1	OT	LUFL	True	71.77%	8.47%	98.34%	45.06%	16.83%	41.98%
ETTh1	OT	HULL	True	61.61%	5.54%	274.44%	37.21%	14.61%	38.69%
ETTh1	OT	LULL	True	89.06%	9.38%	12.58%	23.74%	44.96%	29.78%
ETTh1	OT	HUFL	True	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
FINANCE	Close	Low	True	8.38%	62.50%	128.30%	98.63%	89.74%	98.60%
FINANCE	Close	High	True	3.08%	62.44%	128.29%	98.80%	89.38%	98.43%
FINANCE	Close	Open	True	0.00%	36.42%	128.23%	98.31%	89.95%	97.63%
FINANCE	Close	Volume	True	7.36%	0.00%	0.00%	0.00%	0.00%	0.00%
METEOSUISSE	birch.npm3	snow.cmsum	True	10.25%	0.00%	0.00%	99.04%	98.29%	93.19%
METEOSUISSE	birch.npm3	precip.mmsum	True	0.00%	14358.96%	311.34%	99.07%	94.66%	93.13%
METEOSUISSE	birch.npm3	wind.mps	True	13.34%	88867.23%	252.80%	80.97%	0.00%	73.54%
METEOSUISSE	birch.npm3	temp.c	True	33.86%	94460.16%	659.84%	0.00%	51.16%	0.00%

Table 20: Improvement values for each metric and RMSE from test data set for a single exogenous series as input.

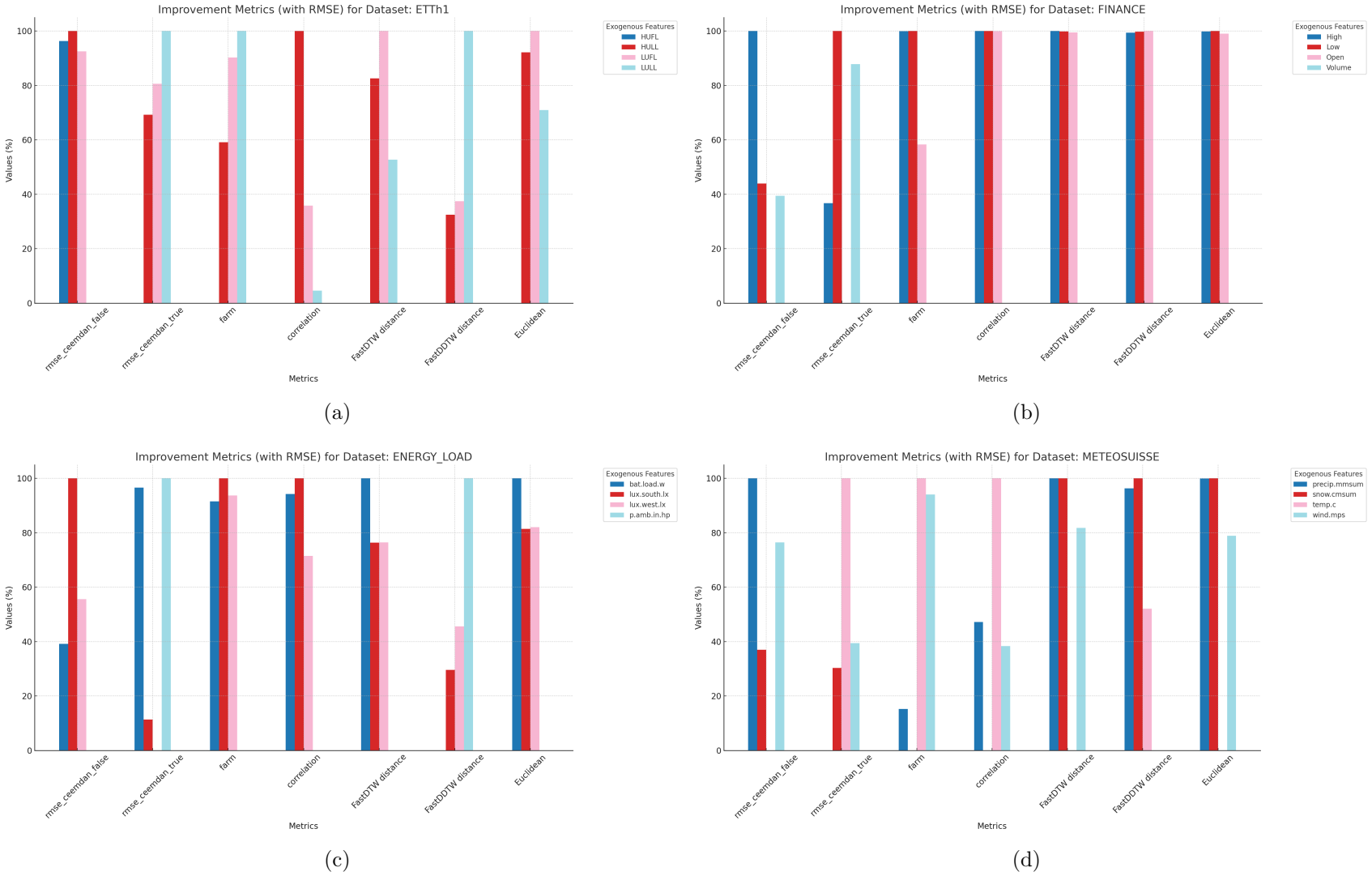


Figure 43: Normalized percentage improvement of metrics and RMSE for each group of dataset. Single exogenous input case. Figure items meanings follow graph titles.

dataset	feature	correlation	FARM_global	fastdtw	fastddtw	Euclidean	imp_no_ceemdan_1	imp_no_ceemdan_2	imp_no_ceemdan_3	imp_ceemdan_1	imp_ceemdan_2	imp_ceemdan_3
finance	**open**	99.95	58.27	99.5	100.0	99.02	0.0	0.0	0.54	0.0	97.56	100.0
finance	**high**	100.0	99.91	100.0	99.36	99.83	100.0	46.11	100.0	36.78	0.0	23.95
finance	**low**	100.0	100.0	99.83	99.76	100.0	43.95	45.14	97.72	100.0	100.0	21.98
finance	**volume**	0.0	0.0	0.0	0.0	0.0	39.42	100.0	0.0	87.79	68.41	0.0
energy_load	**p.amb.in.hp**	0.0	0.0	0.0	100.0	0.0	0.0	0.0	22.63	100.0	100.0	98.32
energy_load	**lux.west.lx**	71.44	93.7	76.42	45.57	82.01	55.2	34.82	100.0	0.0	0.0	0.97
energy_load	**lux.south.lx**	100.0	100.0	76.35	29.58	81.41	100.0	100.0	0.0	11.32	14.88	0.0
energy_load	**bat.load.w**	94.22	91.51	100.0	0.0	100.0	39.31	80.3	1.04	96.57	95.41	100.0
etth1	**hufl**	0.0	0.0	0.0	0.0	0.0	96.3	0.0	100.0	0.0	0.0	0.0
etth1	**hull**	100.0	59.06	82.57	32.5	92.16	100.0	99.01	0.0	69.18	94.86	97.62
etth1	**lufl**	35.83	90.29	100.0	37.44	100.0	92.5	3.46	89.45	80.58	100.0	100.0
etth1	**lull**	4.58	100.0	52.67	100.0	70.95	0.0	100.0	1.15	100.0	32.18	94.49
meteosuisse	**temp.c**	100.0	100.0	0.0	52.04	0.0	0.0	97.44	17.93	100.0	100.0	46.64
meteosuisse	**wind.mps**	38.31	94.08	81.73	0.0	78.92	76.43	0.0	0.0	39.41	73.64	100.0
meteosuisse	**snow.cmsum**	0.0	0.0	99.97	100.0	100.0	36.98	100.0	100.0	30.26	0.0	0.0
meteosuisse	**precip.mmsum**	47.18	15.2	100.0	96.31	99.93	100.0	96.07	24.76	0.0	62.27	30.02

Table 21: Normalized improvement values for multiple exogenous inputs compared to relevance metrics improvements. For example, imp_no_ceemdan_1 means results for improvement in RMSE_test when CEEMDAN is false and the number of input features is 1.

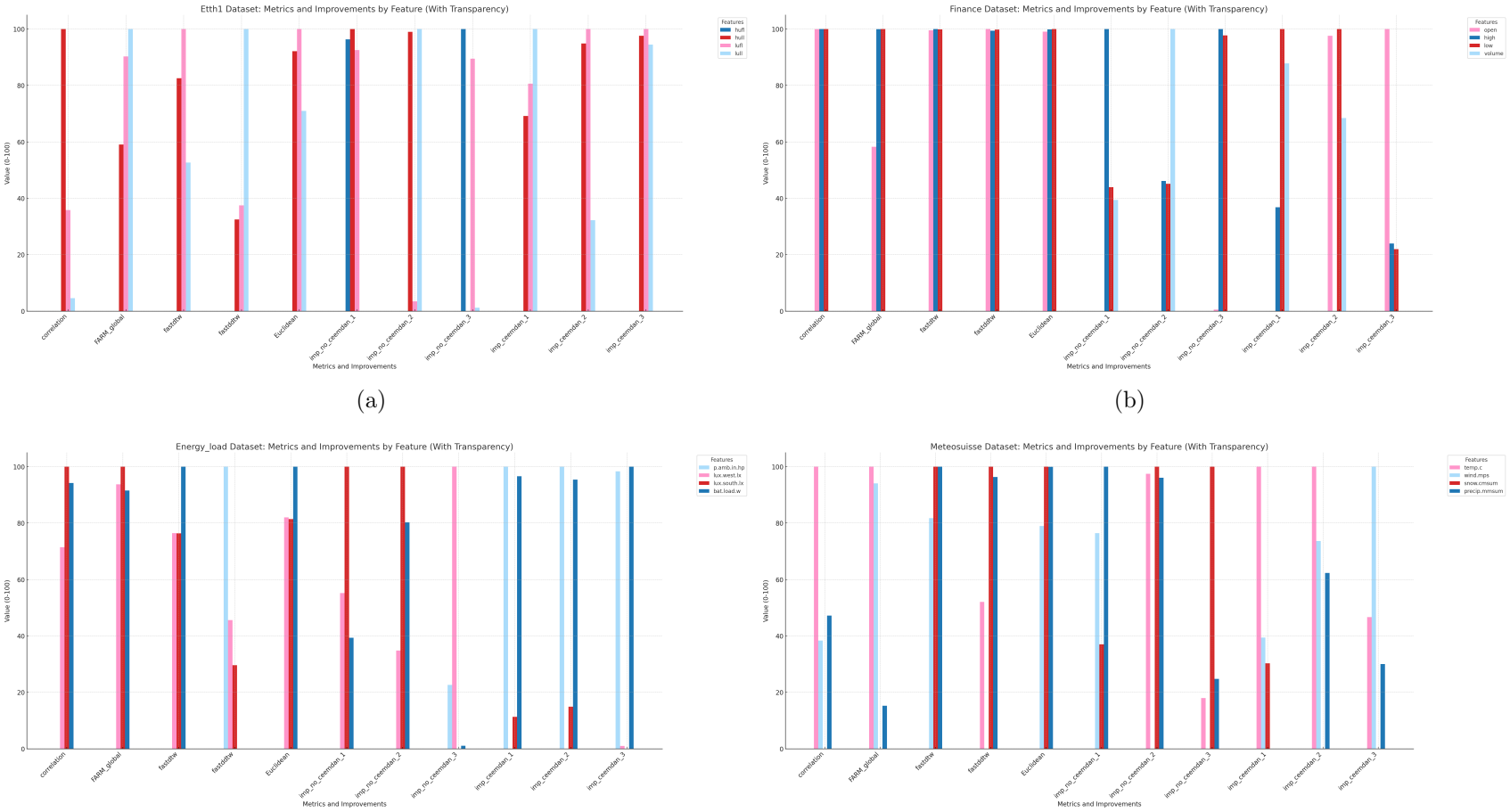


Figure 44: Normalized percentage improvement of metrics and RMSE for each group of dataset. Multiple exogenous input case. Number of features as zero and number of features as 4 do not provide improvement when varying features. Figure items meanings follow graph titles.