History Preservation Approach to Aid in Multivariate Time Series Forecasting

T.K.S Mendis

2024

History Preservation Approach to Aid in Multivariate Time Series Forecasting

T.K.S Mendis
Index No : 19000962

Supervisor: Dr. M.I.E. Wickramasinghe
Co-Supervisor: Mr. M.A.P.P. Marasinghe

May 2024

Submitted in partial fulfilment of the requirements of the B.Sc. (Honours) in Computer Science Final Year Project

Declaration

I certify that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: T.K.S Mendis

..... Date: 24/09/2024

This is to certify that this dissertation is based on the work of T.K.S Mendis under my supervision. The thesis has been prepared according to the format stipulated and is of an acceptable standard.

Principle Supervisor's Name: Dr. M.I.E. Wickramasinghe

..... Date: 29/09/2024

Co-Supervisor's Name: Mr. M.A.P.P. Marasinghe

..... Date: 27/09/2024

Abstract

Multivariate time series forecasting involves utilizing past and present information to predict future outcomes, enabling informed decision-making. Multivariate time series forecasting holds considerable importance across various domains such as finance, healthcare, and weather forecasting. In multivariate time series forecasting, the accuracy and efficiency of forecasts hinge on how effectively the method captures relationships, correlations, and dependencies from past data. This task is particularly challenging due to the complexity introduced by multiple correlated time series variables. Moreover, significant historical events leave notable impacts on time series data, offering valuable insights that are beneficial if leveraged appropriately. However, existing models in literature make forecasts based on the generalized scenario and lack exploration into investigating and extracting insights from these significant historical events. This study proposes a history preservation technique aimed at identifying and utilizing these events within the forecasting process.

This history preservation method uses anomaly detection to identify historically significant events. Then through a data augmentation method, this information is integrated into the original times series dataset. This augmented dataset is then used in forecasting. Experiments were conducted with this method using three machine-learning models and two datasets with two different anomaly detection methods. Results indicated that although there was a decrease in efficiency, accuracy notably improved over the baseline in eight out of a possible twelve instances. Furthermore, five out of the eight improved instances exhibited accuracy improvements exceeding 15% over the baseline. Therefore this approach allows the model to forecast based not only on the generalized scenario but also on significant events that deviate from the norm, leading to greater accuracy in forecasting.

Preface

The primary focus of this study is to explore history preservation as a means to enhance the accuracy and efficiency of multivariate time series forecasting. This method incorporates anomaly detection to identify historically significant events, followed by a data augmentation process to integrate this anomaly information into the original time series dataset. The augmented dataset is then utilized for multivariate time series forecasting.

To the best of my knowledge, the approach presented in this study has not been previously applied to aid in multivariate time series forecasting. Finally, with the continuous guidance and supervision of my supervisor and co-supervisor, we believe that the conclusions drawn in this study represent new contributions to the body of knowledge in multivariate time series forecasting.

Acknowledgement

I extend my heartfelt gratitude to my supervisor, Dr. Manjusri Wickramasinghe, and my co-supervisor, Mr. Pasindu Maransighe, whose expertise and guidance were invaluable throughout my research journey.

I am deeply thankful to my examiners, Dr. H N D Thilini and Mr. T N B Wijethilake, for their timely and insightful feedback, as well as to the evaluators present at the evaluations for their constructive feedback, which greatly contributed to the improvement of my study.

Additionally, I would like to express my appreciation to Mr. Isuru Nanayakkara, Mr. Roshan Abeyweera, and my colleagues at the Cognitive Systems and Time Series (COTS) Research Group at UCSC for their unwavering support throughout my research endeavors.

Finally, I am profoundly grateful to my parents and my sister for their constant love and encouragement, which have been a source of strength and motivation on this journey.

Contents

List of Figures

List of Tables

List of Acronyms

Chapter 1

Introduction

1.1 Introduction & Background

The progression from past to present shapes the future, and understanding this progression offers an advantage in mastering it. Forecasting is the study of the past and present to know the future. A common forecasting approach involves predicting one variable's future values based on its historical patterns alone. For instance, forecasting gold prices considering only its past price trends. While this method can produce accurate forecasts, it overlooks other significant factors. Additional variables, such as economic indicators like unemployment rates or import levels, can enhance forecasting accuracy by providing more comprehensive insights into price dynamics. To illustrate, let's explore the relationship between unemployment rates, import levels, and gold prices in the United States.

The historical movements in gold prices, as depicted in Figure 1.1, provide valuable insights for forecasting its future trajectory. Analysis of past gold price data alone reveals noticeable patterns that can inform forecasts. However, incorporating historical movements in unemployment rates and import levels alongside gold price data, as shown in Figure 1.2, produces more comprehensive and robust information. In the context of gold price dynamics, the observed increase in prices can be attributed to external factors, such as the global economic downturn. During periods of economic uncertainty, investors tend to store wealth in stable assets, with gold being a primary choice. By considering additional variables, such as unemployment rates and import levels, one can explain the fluctuations in gold prices

in correlation with broader economic conditions, particularly during downturns.

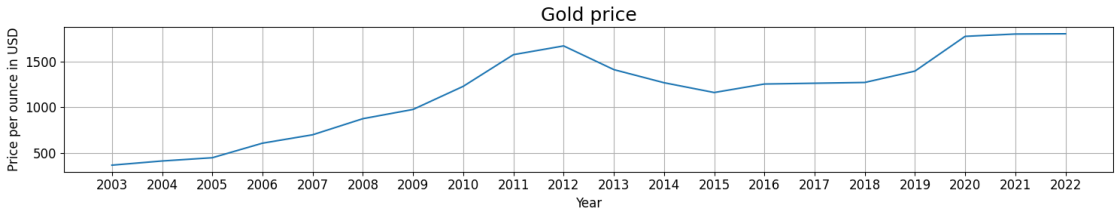


Figure 1.1: Price behavior of ounce of gold

Figure 1.2 highlights a notable similarity between the periods of 2009-2012 and 2020-2021, albeit with some temporal differences, as the 2009 downturn spanned a longer duration. This similarity is attributable to both periods being characterized by economic downturns. Such similar events offer valuable insights for forecasting, as they allow us to leverage past experiences to understand current circumstances better and make better forecasts. Utilizing multivariate time series analysis enables a more precise identification of these similar events, enhancing our ability to extract relevant information for forecasting purposes.

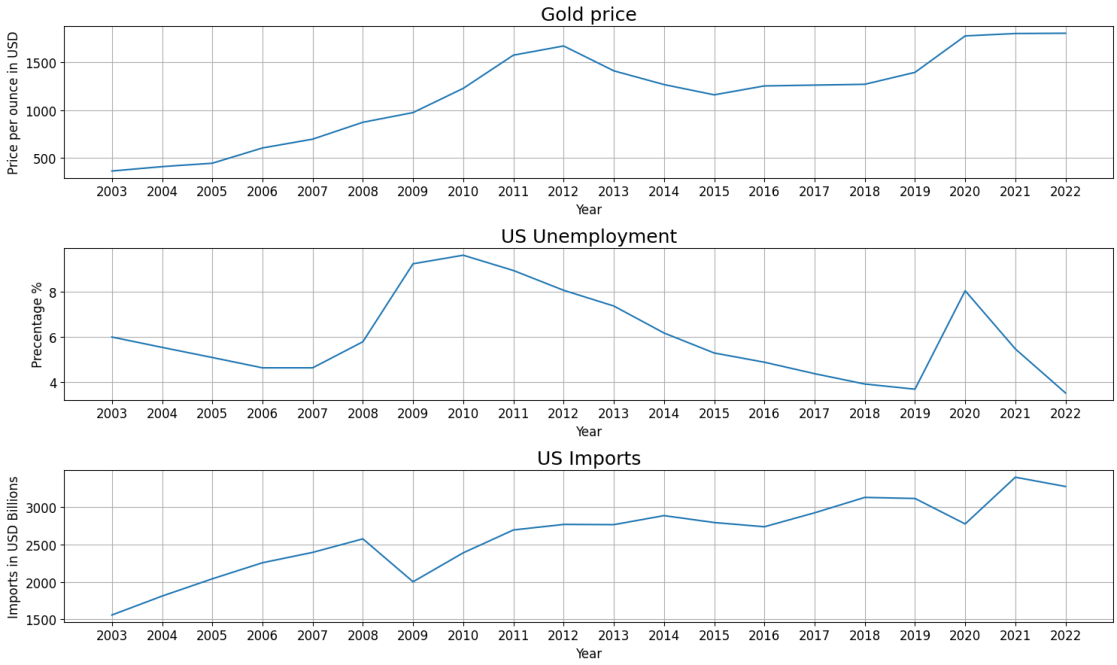


Figure 1.2: Behavior of gold prices with unemployment and imports of the United States

The variables discussed, namely gold price, unemployment, and imports, are all examples of time series data. Time series data consists of sequential observations

collected over time and arranged in chronological order. Typically, such data is gathered at regular intervals, maintaining a fixed sampling frequency. However, in certain cases, irregular sampling frequencies may occur due to variations in data collection methods or the presence of missing values.

When discussing time series, it's important to introduce the inherent components of time series: Trend, Seasonality, and Residuals [1].

The trend represents the long-term movement of a time series, which can exhibit either increasing or decreasing patterns over time. While trends are often discussed in broad time scales, they may also alternate between increasing and decreasing, or even plateau for extended periods.

Seasonality refers to recurring similar behaviors in time series data at regular intervals. For instance, the demand for chocolates on Valentine's Day demonstrates a seasonal behavior. Identifying and understanding such patterns can significantly aid in forecasting.

Residuals are the remaining components of a time series after removing both the trend and seasonality. Ideally, these residuals should exhibit no correlation with each other. However, if correlations persist, it suggests that some information inherent in the data has not been sufficiently removed during preprocessing [2]. These residuals pose challenges for accurate forecasting.

Forecasting is predicting future outcomes with minimal errors using historical data and relevant domain knowledge. Time series forecasting is the same with sequential data. In time series forecasting, the goal is to predict the next value within the series for the upcoming sampling interval. A fundamental categorization in time series forecasting determines the type of data utilized in the forecasting process: univariate and multivariate time series forecasting.

In univariate time series forecasting, forecasts rely solely on the time-ordered sequence data of a single variable, which is also the variable being forecasted. Conversely, multivariate time series forecasting involves forecasting either a single variable or multiple variables by considering multiple related variables, all of which are time series themselves.

Multivariate time series forecasting is a crucial area of research in statistics and Machine Learning (ML) that has significant implications in various fields such as

finance, economics, healthcare, weather forecasting, and more. With the increasing availability of large-scale data and the rapid advancements in computational methods, the ability to accurately predict future values of multivariate time series has become an essential tool for decision-making in various domains.

Currently in literature hybrid models like the Exponential Smoothing-Recurrent Neural Network (ES-RNN) [3] and combination models based on the Light Gradient-Boosting Machine (LightGBM) are performing the best[4], [5]. Hybrid models that use both statistical methods, as well as ML methods, are performing well because each method accounts for the other's weakness[5]. Whereas LightGBM utilizes a set of methods and combines them for the multivariate time series forecasting effort. This has proved to be quite useful as four out of the top five performers of the M5 competition used the LightGBM technique[4]. The value of combination models lies in the fact that different models used can focus on different aspects of the time series[4].

The accuracy and efficiency of multivariate time series forecasting models are influenced by the model's ability to infer relationships, correlations, and dependencies from past data of different time series. An all-encompassing term that can be used for this activity is history preservation.

History preservation involves effectively capturing the past behavior of a given time series in a meaningful manner to support forecasting efforts. Even Moving Average (MA) methods utilize this concept, albeit in a basic form. MA forecasting calculates the next value by averaging the most recent values within a specified window size, thereby preserving and utilizing recent historical data for future predictions. In multivariate time series forecasting, capturing the behavior of multiple time series over extended periods for effective history preservation remains an area of ongoing research. When executed correctly, history preservation can enhance the efficiency and accuracy of forecasts.

Furthermore, it is rather difficult to preserve history in multivariate time series forecasting owing to the complexity introduced by multiple correlated time series data. This complexity can reduce the accuracy and efficiency of multivariate time series forecasting. This study explored the current models of multivariate time series forecasting and adapted them for history preservation to improve the accu-

racy of multivariate time series forecasting. Accuracy improvement was achieved, however efficiency of the forecasting effort was affected slightly.

1.2 Problem Significance and Contribution

The ability to accurately forecast the future is essential for making informed decisions in a variety of domains such as financial forecasting [6], weather prediction [7], electricity demand forecasting [8] and sales forecasting [4]. The more accurate the forecasts the better a decision maker can utilize them to prepare for the future.

In forecasting, understanding the historical context as well as the current state of the situation is of utmost importance since the future evolves from the present influenced by the past. This future depends not only on the past behavior of the forecasted variable but also on the interplay with other related variables. The essence of forecasting lies in the recognition that history is not a mere collection of isolated data points, but rather a complex web of interconnected events and patterns spanning over multiple variables. These events and patterns shape the evolving nature of the system. Preserving the history of events and patterns in data will enable us to capture the evolving nature of the system which will allow for effective and accurate forecasts.

This research aims to make contributions to the field of multivariate time series forecasting by delving deeper into the intricate relationship between historical data, the current state, and the evolving nature of the forecasted system, by developing a technique that effectively captures the complex dynamics at play when it comes to preservation of recurrent significant historical events. This study seeks to enhance the accuracy and efficiency of multivariate time series forecasting techniques, ultimately enabling better decision-making and more precise forecasts in various domains and applications.

1.3 Research Aim

Improve the accuracy and efficiency of multivariate time series forecasting.

1.4 Research Questions

RQ1. What are the techniques used in literature for multivariate time series forecasting?

RQ2. How can we adopt multivariate time series forecasting techniques to preserve history?

RQ3. How preservation of recurrent significant historical events aid in improving the accuracy and efficiency of multivariate time series forecasting?

1.5 Research Objectives

RO1.1. Determine the aspects that contribute to the performance of different multivariate time series forecasting techniques.

RO2.1. Analyze the recurrence of significant historical events and whether they are generalizable.

RO2.2. Develop a technique to capture the characteristics of recurrent significant historical events to aid in multivariate time series forecasting.

RO3.1. Examine the relationship between preservation of recurrent significant historical events and the accuracy of forecasts.

RO3.2. Study the influence of preservation of recurrent significant historical events on the efficiency of multivariate time series forecasting models.

1.6 Scope

- Exploring the techniques used in multivariate time series forecasting and the aspects that contribute to the performance of multivariate time series forecasting.
- Explore history preservation techniques used in literature for multivariate time series forecasting and evaluate their effectiveness.
- Develop a technique for preservation of recurrent significant historical events. Use the developed technique in multivariate time series forecasting and evaluate its effectiveness.
- Examine how preservation of recurrent significant historical events aid in

improving the accuracy and efficiency of multivariate time series forecasting.

1.7 Methodology and Approach

The research methodology adheres to the Scientific Method in developing a technique for history preservation, intended for application in multivariate time series forecasting. This technique utilizes anomaly detection and data augmentation to retain and leverage historical data. Further discussion on the research design can be found in Chapter 3.

1.8 Summery

This chapter introduces the research study, outlining its research problem, aim, questions, objectives, and scope. Additionally, it provides an overview of multivariate time series forecasting. Following this introduction, Chapter 2 delves into the Literature Review, while Chapter 3 offers a detailed account of the Research Design. Implementation and results are discussed in Chapter 4, with Chapter 5 presenting the research findings, conclusions, limitations, and implications for future research.

Chapter 2

Literature Review

Time series forecasting plays a crucial role in numerous real-world applications, financial market predictions [9], energy demand forecasting [10], and environmental monitoring [11] are such applications. As the volume and complexity of time series data continue to grow, there is an increasing demand for accurate and efficient forecasting as it has become an essential tool for decision-making in various domains.

Time series forecasting can be categorized as univariate and multivariate. In univariate, only a single time series attribute is used, and in multivariate more than one time series attribute is used for the forecasting. Due to more attributes being available in multivariate forecasting, more information is available but in the same way, it is more complex. This review, provides a comprehensive overview of the current landscape of multivariate time series forecasting, highlighting recent developments, challenges, and opportunities in the field. The review begins by discussing traditional approaches to time series forecasting, including Recurrent Neural Network (RNN), and Convolutional Neural Networks (CNN). These foundational techniques have laid the groundwork for more advanced methodologies and continue to serve as building blocks for ensemble methods and hybrid models although these foundational models are inefficient in capturing long-range dependencies.

In terms of an advanced model, the review delves into Transformer-based architectures, initially designed for natural language processing tasks, which have been adapted to handle sequential data, offering improved performance in capturing

long-range dependencies and spatial relationships. Attention mechanisms, a key component of transformer models, enable models to focus on relevant information within the input sequences, enhancing both accuracy and interoperability. Single models have strengths and limitations and different models have different strengths and limitations. This fact can be exploited.

Thus the review explores the emergence of hybrid and ensemble methods as a powerful strategy for improving forecast accuracy and robustness. By combining multiple models, hybrid and ensemble methods mitigate the limitations of individual models and offer more reliable forecasts. Hybrid models combine elements of different architectures whereas ensemble models use techniques like bagging and boosting to combine weak learners.

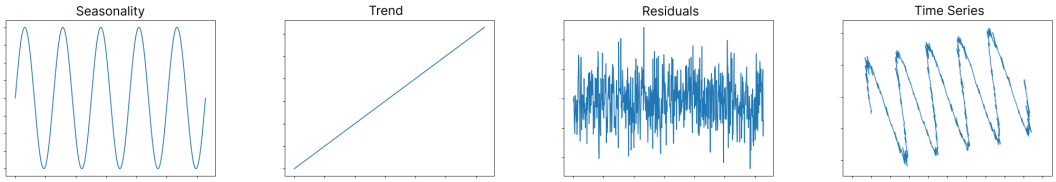
This review is structured as follows. Section 2.1 presents time series forecasting and the various aspects of the subject matter. Section 2.2 presents the Multivariate Time Series Forecasting and its aspects. Section 2.3 discusses the traditional deep-learning architectures of time series forecasting. Section 2.4 summarizes and analyses the current literature on Multi/ Ensemble Architectures. Section 2.5 discusses the evaluation metrics used in multivariate time series forecasting. Section 2.6 and 2.7 analyze the current works and discuss potential future work that can be done in the field and conclude the review.

2.1 Time Series Forecasting

This section covers time series and forecasting, which are used in various fields like finance and weather forecasting to aid in decision making. Time series are sequences of data collected over time, by analyzing time series patterns and trends can be identified. Forecasting entails predicting future values based on past data using methods based on statistics and machine learning. Understanding these concepts is important for making decisions and predictions in many areas.

2.1.1 Time Series

A time series is a sequence of data observed over time and ordered chronologically. Typically data in a time series is gathered in constant intervals so most time series



(a) Seasonality (b) Trend (c) Residuals (d) Time Series

Figure 2.1: Seasonality, Trend, and Residuals of a Time Series

data has a fixed sampling frequency. In some instances, however, due to how the data is being gathered and missing values, there can be time series with irregular sampling frequencies.

The discussion of time series necessitates the introduction of inherent components of a time series. They are Trend, Seasonality, and Residuals [1]. The trend is the long-term movement of a time series, this can be either increasing or decreasing over time. Although the trend is talked about in rather large time scales, it does not always either increase or decrease, they switch between the two and it can even plateau for extended periods. Seasonality is the presence of similar behaviors in the time series data at specific regular intervals. For example the rise of demand for winter wear during the winter season of the year. These types of behaviors can be highly useful when it comes to predicting the future. Residuals are what is left when Trend and Seasonality are processed out of the time series. Ideally, these residuals shall not be correlated with each other, if not that suggests that there is some information left in the data that has not been removed in the preprocessing [2]. These residuals make forecasting difficult. Figure 2.1 illustrates the three components seasonality, trend, and residuals. The time series depicted in 2.1d is made up of the Seasonality, Trend, and Residuals shown in figures 2.1a, 2.1b, and 2.1c.

2.1.2 Time series forecasting

Forecasting is the act of predicting the future with as few errors as possible given information about historical data and domain knowledge that may be useful. Time series forecasting is the same however when forecasting time series the next possible value for the next sampling interval is forecasted. When forecasting time series, a fundamental categorization affects what data is used in the forecasting effort.

They are univariate and multivariate time series forecasting.

Univariate time series is where the forecasting is done only with the use of a single variable's time-ordered sequence data and this variable itself shall be forecasted. No other data shall be integrated into this forecasting. Let's take an example, let's say the price of fuel needs to be forecasted and for that forecasting effort only the historical pricing data of fuel shall be utilized, this is univariate time series forecasting.

Multivariate time series forecasting is where the forecasting effort takes into account multiple related variables which themselves are time series. As an example let's again consider the fuel price forecasting as before, in the multivariate scenario not only the historical pricing data of fuel but also other related variables such as price data of crude oil, inflation rate, dollar rate and such shall be considered. This provides the forecaster with more information to base the final forecast which can lead to better accuracy in the forecasted value.

2.1.3 Methods of Time Series Forecasting

In time series forecasting, there are two main ways to make predictions: statistical based methods and machine learning. Statistical methods use historical data patterns and math models, while machine learning methods use Artificial Neural Networks (ANN) that learn from data to make predictions.

Statistical based methods of time series forecasting are characterized by the use of mathematical representations of time series behavior to forecast the future. In Statistical based methods, a model will produce the same output for the same input every time. There are many Statistical based methods of time series forecasting including Auto Regressive (AR), MA, Exponential Smoothing (ES), and Auto Regressive Integrated Moving Average (ARIMA).

In machine learning methods ANNs are used for the forecasting effort. ANNs consist of neurons and weights that connect neurons to each other. These weights are capable of capturing information for the task it is trained on. By tuning these weights information about the behavior of the time series can be captured and can be later retrieved to be used in forecasting [12]. There are many machine learning methods in time series forecasting including recurrence-based, convolution-based,

attention-based, and boosting-based techniques. Then there are multi/ ensemble methods that combine multiple machine learning models or machine learning models and Statistical based models to achieve more accurate forecasts [4], [5].

Over the years it has been shown that machine-learning methods have better performance than Statistical based methods. The potential of the machine learning methods was first demonstrated in[5], where the highest accuracy method was based on a hybrid approach utilizing Exponential Smoothing a Statistical based method, and RNNs a machine learning method for the forecasting effort[3]. The second highest accuracy method used a machine learning method called gradient boosting for the forecasting effort[13]. Even though these are based on machine learning methods they are built upon Statistical based techniques. Hence the authors of [5] have concluded that there are possibilities for improvements of pure machine learning methods in forecasting.

Indeed performance improvements of pure machine learning methods can be seen in [4] where top-performing techniques were all pure machine learning methods and they performed better than all the statistical benchmarks used in the study. The main machine learning technique used is the gradient boosting technique implemented in the LightGBM framework. Four out of five of the top performers used the LightGBM for their forecasting effort, the top performer is also in this group of four. Machine learning based techniques have shown potential in the last few years and this review aims to focus on machine learning methods of multivariate time series forecasting.

2.2 Multivariate Time Series Forecasting

Compared to univariate time series forecasting there are additional aspects to consider when it comes to multivariate time series forecasting due to the introduction of multiple related variables. Some of these are how to select the most suitable variables for the forecasting effort, how to effectively cross learn to capture relationships between variables, and how interpretability can be useful for decision making in the context of multiple variables. Such aspects and more shall be discussed in this section.

2.2.1 Variable Selection

Multivariate time series forecasting, as the name suggests, deals with more than one time-ordered variable. Hence during the forecasting effort, the correlations between these time series are exploited to aid and increase the accuracy of the forecasts. The correlations discussed here represent the dependencies of time series and how they are influenced by each other [14]. These time series are influenced by each other at different scales hence some time series may or may not aid in the forecasting effort. The choice of variables used in the time series is key to achieving accurate forecasts while maintaining efficiency. This selection is mainly based on the domain knowledge of the forecaster. Automatic variable selection based on the effectiveness of that variable in the forecast is a useful area of research.

2.2.2 Forecasting Horizon

The forecasting horizon refers to the span of time into the future a prediction is made, it varies depending on the specific problem, but it is typically chosen based on the useful period of the forecast. It is important to note that the forecasting horizon influences the reliability and accuracy of the forecasts. Generally, the farther into the future the forecast, the greater the error in the predictions. The forecast horizon should be taken into consideration when choosing the model. There are two types of forecasting horizons when it comes to time series forecasting. They are point forecasting and multi-horizon forecasting.

Point Forecasting

Point forecasting is a method used in time series forecasting to make a single prediction of a future value using past observations or patterns in data. One limitation of point forecasting is that it assumes a constant and unchanging underlying process generating the time series data. This may not be the case in many real-world situations. Point forecasting can be used to forecast multi-horizon forecasts by utilizing the already forecasted values as inputs. This type of multi-horizon forecasting introduces error propagation through the forecasted sequence due to error stacking [1].

Multi-Horizon Forecasting

Instead of predicting a single future value at a specific point in time, multi-horizon forecasting forecasts the future values at multiple points in time over a specified forecasting horizon of a time series. Within multi-horizon forecasting, it's common to observe a decline in forecast accuracy with the expansion of the forecasting horizon [15], [16].

2.2.3 Interpretability

Interpretability in time series forecasting refers to the capacity to comprehend and explain the underlying factors and mechanisms that are driving the forecasts produced by a model. Interpretability allows decision-makers to understand the influences that contributed to the forecast which will be useful in assessing the reliability of the forecast [17]. The importance of interpretability is further increased in multivariate time series forecasting because of the multiple variables, as understanding how variables affect each other can assist in decision-making. There are several techniques to increase the interpretability of time series forecasting models. Model selection, feature engineering, and visualization relating to pre-model interpretability. Error and sensitivity analysis concerned with the model interpretability. With these techniques, factors influencing the forecast and model behavior can be identified allowing for improving model accuracy as well as decision-making [17].

Interpretability with Attention

The attention mechanism keeps track of the significance of each time step to the forecasting in the input time series. The weights assigned by the attention mechanism can be analyzed to understand the contribution of each feature and the importance of each time step in the input sequence for the forecast. This helps to recognize patterns and relationships in the data influencing the forecast. This mechanism is one of the predominant techniques used to achieve interpretability [18]–[21].

2.2.4 Cross Learning

When it comes to multivariate time series forecasting, it is important to consider dependencies between the different time series as they aid in forecasting. Dependencies can arise between time series due to many factors such as cause-and-effect relationships, common underlying trends, and interactions between time series. When it comes to dependencies there are two main perspectives to consider, one the Local perspective and the other Global perspective [14]. Figure 2.2 depicts the difference between the local and global perspectives of dependency consideration.

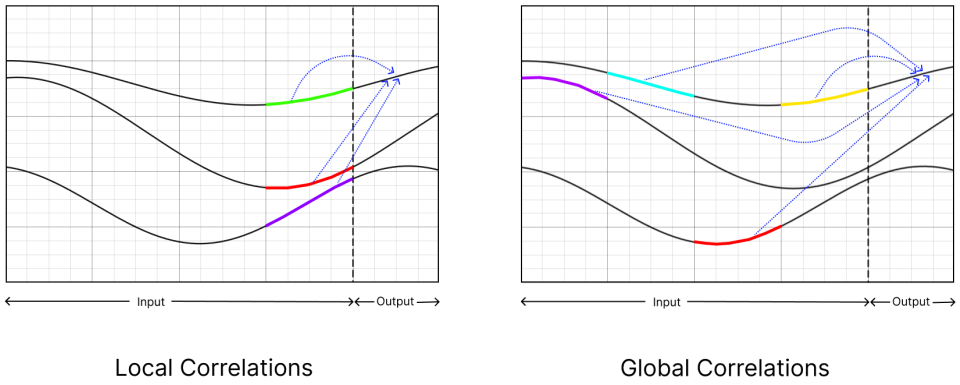


Figure 2.2: Time Series Correlations

Methods like ES [22], Gaussian processes [14], Convolutional attention mechanisms [23], and Graph Neural Networks (GNN) [24] have been used for these dependency considerations where they focussed on both perspectives to aid in achieving accurate forecasts.

Local Perspective

Local dependency in multivariate time series forecasting refers to the dependency between the current value of a time series and its most recent past values as well as the most recent past values of other related time series within a certain window of time [14]. This concept is related to the idea of short-term memory in forecasting and is important to consider to accurately model the complex relationships between multiple time series.

Global Perspective

Global dependency in multivariate time series forecasting refers to the long-term dependencies between multiple time series [14]. This type of dependency is concerned with the overall behavior of the time series and the common factors that influence it.

2.2.5 Linearity

In multivariate time series forecasting, the relationship between the response variable and the predictor variables over time can be linear, where the response variable is assumed to be a linear combination of the predictor variables and can be expressed as a system of linear equations, or non-linear, where the model allows for a more complex relationship between the response variable and the predictor variables [2].

The choice between linear and non-linear multivariate time series models depends on the nature of the data and the problem being analyzed. An example of a linear multivariate time series model is the Vector Autoregression (VAR) model [2] and examples of non-linear multivariate time series models include neural network models like the RNN model, Long Short-Term Memory (LSTM) model, and GNN model.

2.3 Traditional Deep Learning Architectures for Time Series Forecasting

This section discusses traditional time series forecasting architectures. Current literature is largely outgrown from using these traditional architectures alone in time series forecasting. These models have now become the building blocks of multi/ ensemble methods which are currently the state of the art [4], [5].

2.3.1 Recurrence Based Architectures

Recurrence based architectures have an internal hidden state that is a representation of history that the model has seen so far. How this internal state is updated

gives rise to three different models.

Recurrent Neural Network

As the name suggests RNN is a subclass of neural networks and one of its main abilities is that it can handle sequential data by leveraging its internal memory. It is composed of a set of interconnected processing units which are called neurons that are arranged in a temporal sequence. Each neuron in an RNN receives inputs from the previous neuron in the sequence, as well as from the current input, and produces an output that is fed into the next neuron in the sequence.

The key feature of RNNs is their ability to maintain a hidden state vector that represents the network's internal memory. This memory contains the summary of information that was introduced to the model in prior steps. The hidden state vector is updated recursively based on the current input and the previous hidden state, using a set of learnable parameters that are optimized during training. The updated hidden state is then used to make predictions or classifications for the current time step. This hidden state solves the problem of feed-forward Neural Network (NN), which is that feed-forward NNs cannot keep track of past dependencies[1], [25].

Even though RNNs solve some problems of feed-forward NNs it introduces new ones. The main challenges of training RNNs are the vanishing and exploding gradient problem, which can occur when the gradients used to update the parameters become very small or very large. This problem makes it difficult for the network to handle long-term dependencies in the data. Several variants of RNNs have been developed, including the LSTM and Gated Recurrent Unit (GRU) architectures, which use more sophisticated mechanisms for the gates in the architectures to better handle long-term dependencies and avoid the problems mentioned above.

Long Short-Term Memory

To handle the vanishing gradient and exploding gradient problems LSTM was introduced. LSTM also has the ability to better handle long-range dependencies. These improvements are achieved by introducing a cell state. This cell state keeps track of long-standing information. This memory state is governed by a set of

gates. Which are the update gate, forget gate, and output gate. Forget controls which information shall be preserved or forgotten in the cell state. Update gate controls the new information from which the cell state is updated. Some authors refer to this gate as the input gate as well. Output gate controls the output of a cell state which shall be used in the next cell state. One of the key features of LSTMs is their ability to selectively remember or forget information over long periods of time. This is achieved by the regulation of the flow of information by the gates. This allows the cell to maintain important information over long periods of time [1], [25].

Gated Recurrent Unit

GRU is a simplification of LSTM. This simplification came about to decrease the computational cost of LSTM. This simplification reduced the number of gates from three in LSTM to two in GRU. The gates of the GRU are the update gate and the relevance gate. The candidate state is computed by the GRU to be used by the gates to update the current cell/ memory state. Update gate considers the candidate state and decides whether to update the cell state or not with the candidate state. The prior cell state is required to compute the candidate state for the current cell state updates. The relevance gate decides the relevancy of the prior state in this computation. Compared to LSTMs, the GRU has fewer parameters and is, therefore, faster to train and more computationally efficient. However, LSTMs are generally considered more powerful and flexible, especially for tasks involving long-term dependencies [1].

2.3.2 Convolution Based Architectures

Convolution based architectures have been mainly used in classification problems, the same is true in time series classification[26]. Convolutional layers which are arranged sequentially make up the structure of the CNN. A sliding filter and dot product are used to obtain the output of layers used in this architecture. This is called a convolution of the input and the filter. One of the main features of the CNN is its ability to identify patterns in the input data [6].

Temporal Convolutional Networks (TCN) [27] are a notable example of convo-

lutional based architectures. TCNs use causal convolutions to prevent information from the future from affecting the past. Furthermore, like RNNs TCNs can use an input sequence of any length to generate an output of the same length. It is infeasible to consider a large receptive field with simple convolutions as it requires deep networks or large filters. Dilated convolutions were introduced to address this issue. Dilated convolutions allow for an exponentially large receptive field [28]. Fig.2.3 illustrates causal convolution with dilation.

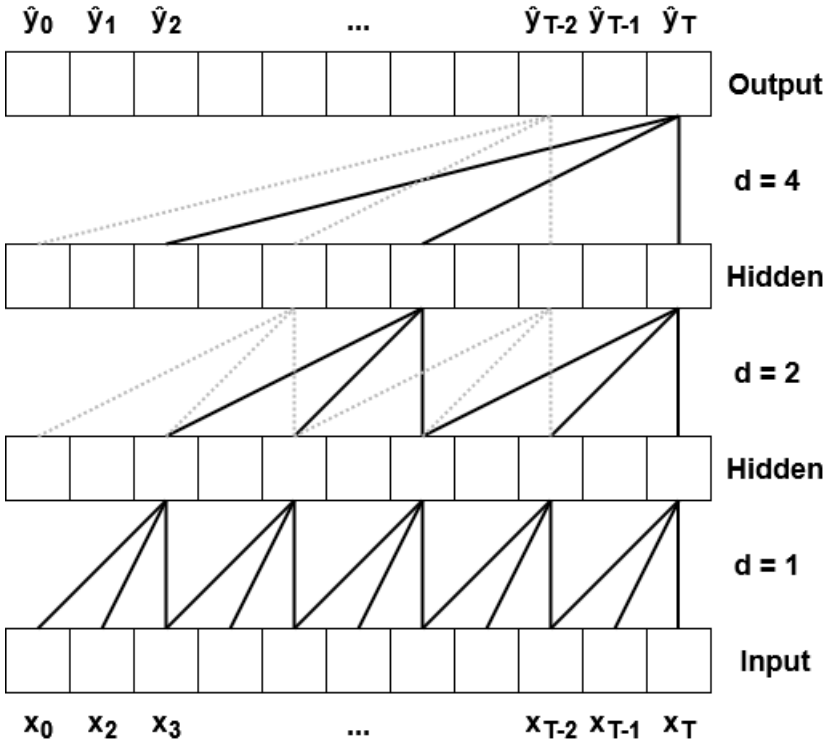


Figure 2.3: A dilated causal convolution with dilation factors d = 1, 2, 4 and filter size k = 3.

2.3.3 Attention Based Architectures

Attention was first introduced by Bahdanau et al. 2016 [29]. The attention mechanism allows a network to focus on specific events in the past which increases the effective look-back window size of the models[25]. Furthermore, with the attention mechanism, models can be relieved of the burden of encoding input sequences to fixed-size vectors because it uses the whole input sequence as a look-up space. Transformer is a notable model that uses a variation of attention.

The Transformer is a neural network architecture for sequence-to-sequence learning that was introduced in a paper by Vaswani et al. 2017 [30]. Transformer is a model that outperforms the RNN regarding long-term history preservation. One of the main factors for this is the fact that RNNs are not parallelizable due to their inherent sequential nature. Transformer overcomes this issue by relying exclusively on an attention mechanism for history preservation[30].

The self-attention mechanism is the core of the Transformer, which allows the model to compute representations of each component in the input sequence by attending to other elements in the sequence. Self-attention is calculated by computing the dot product between a query vector and key vectors for each element in the input sequence. The scores obtained are normalized using a softmax function and utilized to weight a value vector for each element. Self-attention mechanism outputs the weighted sum of the value vectors, which represents the attended representation for each element. This self-attention mechanism allows the transformer to avoid the recurrent structure of the RNN and reduce the signal travel lengths within the NN to the theoretical limit of O(1) [31]. The transformer aids in temporal dependency tracking as it focuses on both global and local contexts of time series [23].

The Transformer also introduces positional encoding, which allows the model to incorporate the position of each element in the input sequence into its computation. This involves adding a set of sinusoidal functions to the input sequence, with different frequencies and offsets for each position in the sequence.

There are models like the Informer which focuses on reducing the time and memory complexity of the vanilla transformer's self-attention mechanism and improving the efficiency of the forecasts [31]. Then there are models like the Temporal Fusion Transformer that utilize recurrent layers for local dependencies and self-attention for long-term dependencies utilizing the strengths of both [15].

2.4 Multi/ Ensemble Architectures

Ensemble models in multivariate time series forecasting involve the amalgamation of forecasts generated by multiple models. The fundamental idea underlying ensemble models are the accounting of individual models' strengths and weaknesses

in way of combining, resulting in a more accurate and reliable forecast [32]. In multivariate time series forecasting, considering the multitude of variables, this approach is particularly valuable.

Ensemble models involve using various algorithms, such as ARIMA, regression, and machine learning approaches, to produce a set of forecasts, which then are combined using techniques such as weighted averaging or stacking to produce a final ensemble forecast. Another approach is to train multiple models on different data subsets, such as different time periods or subsets of input variables. These models can then be combined using techniques such as bagging or boosting to generate an ensemble forecast. Some examples are the AdaBoost algorithm [33] and the utilization of stacking as their ensembling method [34].

Ensemble models provide several benefits over individual models, including enhanced accuracy, reduced overfitting, and improved robustness to changes in the underlying data. However, implementing ensemble models can be more complicated than using individual models, and it may require more computational resources for training and running the models [32].

2.5 Evaluation Metrics

Multivariate time series forecasting involves predicting multiple time series variables simultaneously. Here are some common evaluation metrics for multivariate time series forecasting:

1. Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE): MAE and RMSE can be calculated for each of the individual variables separately. These metrics measure the average absolute and squared differences between the actual and predicted values for each variable.
2. Mean Absolute Percentage Error (MAPE) and Symmetric Mean Absolute Percentage Error (SMAPE): MAPE and SMAPE can be calculated for each variable. These metrics measure the average percentage difference between the actual and predicted values for each variable.
3. Mean Absolute Scaled Error (MASE): Can be calculated for each variable. MASE scales the error based on the in-sample MAE from the naive forecast method, making it independent of the data scale. A scaled error less than

one indicates a better forecast than the average one-step naive forecast, while a value greater than one suggests a worse forecast [35].

4. Mean Square Error (MSE) and R-squared (R2): MSE is the average of the squared errors between the actual and predicted values across all variables. R2 measures the proportion of the total variance in all the variables that are explained by the model.
5. Mean Directional Accuracy (MDA): MDA can be used for time series forecasting to measure the percentage of correct directional predictions across all variables.

2.6 Discussion

Multivariate time series forecasting is a complex and crucial task with applications ranging from finance to environmental science. Insights into the various approaches and methodologies used in forecasting help in decision-making.

Traditional deep learning architectures, such as RNNs, LSTMs, and GRUs, have been fundamental in time series forecasting with their ability to maintain internal memory which allows them to capture complex patterns and dependencies [1], [25]. However, these models face challenges in capturing long-term dependencies and efficiently processing temporal information due to vanishing and exploding gradient problems.

Attention-based architectures, particularly transformers, have emerged as alternatives to RNNs for capturing global temporal dependencies and spatial relationships [30], [31]. Substituting recurrent connections in favor of self-attention mechanisms, transformers alleviate the vanishing and exploding gradient problem and offer more parallelizable training, leading to faster convergence and improved scalability. However, their reliance on large-scale pretraining and extensive computational resources may limit their applicability in resource-constrained environments. Due to the heavy resource needs of transformers models like the Informer were introduced. Informer optimizes the transformer architecture for time series forecasting by reducing computational complexity with a technique called sparse attention [31]. Furthermore, transformers have positional encodings which are not ideal for time series data. Temporal Fusion Transformer replaces these encodings

with a sequence-to-sequence layer as this layer is more suitable for time series data [15]. A potential area of research here is introducing sparse attention found in the Informer to the Temporal Fusion Transformer to ascertain how efficiency and accuracy are affected.

Single models like RNNs and transformers have limitations and when these limitations are better mitigated more robust forecasts are produced. Hybrid/ Ensemble methods, which combine multiple models, have gained traction for their ability to enhance accuracy and robustness in multivariate time series forecasting [32]–[34]. These enhancements are gained by leveraging the diversity of individual models, hybrid/ ensemble methods can mitigate the limitations of a single model with a combination of one or more other models and offer more reliable forecasts. However, the effectiveness of hybrid/ensemble methods depends on the diversity and quality of constituent models, as well as the choice of aggregation technique, which may introduce additional complexity and computational overhead. As exemplified by the Hybrid ES-RNN [22] model which performs well in forecasting by extracting contextual information from related time series, utilizing separate tracks for context and forecasting. Although ES-RNN faces challenges in scalability and interpretability with a large number of time series.

Moving forward, future research could focus on continued efforts in developing hybrid and ensemble methods that strike a balance between accuracy and computational efficiency. Which will prove crucial for advancing the field of multivariate time series forecasting.

2.7 Conclusion and Future Work

This review has explored various methodologies and approaches for multivariate time series forecasting, highlighting the strengths, limitations, and advancements in the field. Multivariate time series forecasting plays a pivotal role in diverse domains, including finance, healthcare, climate modeling, and more. Leveraging deep learning architectures, ensemble methods, and innovative modeling techniques, researchers have made significant strides in improving forecast accuracy and robustness.

Traditional deep learning architectures such as RNNs and CNNs have laid

the groundwork for modeling sequential data [1], [26]. However, challenges in capturing long-term dependencies and processing spatial relationships have led to the development of more sophisticated architectures like transformers [30].

Ensemble methods have emerged as effective strategies for combining forecasts from multiple models to mitigate individual weaknesses and enhance overall performance [32].

Recent advancements in hybrid models, transformer-based architectures, and GNNs have pushed the boundaries of multivariate time series forecasting [15], [22], [23], [31]. By incorporating contextual information, handling spatial dependencies, and improving scalability and interpretability, these models offer promising solutions to complex forecasting challenges.

In current literature, while multivariate time series forecasting models do capture the historical behavior of time series data for forecasting, there is a notable absence of explicit consideration of significant historical events. These events, despite their potential to profoundly impact the environment, are not integrated into forecasting efforts to gain insights and enhance accuracy. This represents a gap in the existing methodologies, as leveraging significant historical events could provide valuable opportunities for exploitation in forecasting tasks.

Multivariate time series forecasting continues to evolve rapidly, driven by advancements in deep learning, ensemble methods, and innovative modeling techniques. Through interdisciplinary collaboration and methodological innovation, researchers can uncover new opportunities and address complex forecasting problems, to further advance the field.

Chapter 3

Research Design

3.1 Research Approach

The research methodology adheres to the Scientific Method in developing a technique for history preservation through anomaly detection, intended for application in multivariate time series forecasting. This technique seeks to retain and leverage historical data, with the goal of enhancing the accuracy and efficiency of forecasting.

3.1.1 Datasets

During the study, two datasets were used for experimentation. One of these datasets named Primary Economic Dataset [36] was created for this study. The other was a dataset that was already available named Beijing Multi-Site Air-Quality Data Dataset [37].

The main motivation that prompted the creation of the Primary Economic Dataset was that obtaining multivariate time series data is difficult. There are quality univariate datasets but such multivariate data are not present. This dataset covers five decades' worth of data depicting the behavior of key economic indicators outlined by the Federal Reserve Bank of St. Louise. Furthermore, this dataset also includes global surface temperature and atmospheric CO2 concentration to cover the relationship between economy and nature.

The second dataset is the Beijing Multi-Site Air-Quality Data Dataset which tracks six air pollutants alongside six other meteorological variables [38] over four

years. This dataset will allow exploration into how natural chaotic systems [39] evolve.

Both datasets were utilized for both training the models and making forecasts within the scope of this study. These forecasts served as the basis for assessing the efficacy of the History Preservation method proposed by this study in multivariate time series forecasting.

3.1.2 Models

Three models that were identified to perform well during the literature review phase were used in the study. These models were used in assessing the efficacy of the History Preservation method outlined below.

Light Gradient-Boosting Machine

LightGBM utilizes gradient boosting, a methodology that amalgamates multiple weak learners, often represented by decision trees, to formulate a predictive model. These decision trees are generally shallow and are incrementally developed to minimize a predefined loss function. The selection of LightGBM for this study was motivated by its prominence as one of the leading models for multivariate time series forecasting within the current research landscape [4].

Long Short-Term Memory (LSTM)

Recurrence-based architectures represent the prevailing methods in time series forecasting. Techniques such as RNN, LSTM, and GRU are NN models designed to retain historical information internally for forecasting purposes. Among these, LSTM stands out as the most accurate [1], benefiting from its inherent compatibility with time series data and demonstrated performance in prior research [3], [40]. Hence, it was selected for this study due to its well-established efficacy. While hybrid models incorporating LSTM alongside other methods are common, this study opted for a straightforward LSTM implementation to distinctly evaluate the effectiveness of the History Preservation method detailed below.

The study utilized two variants of LSTM: one focusing on a single lag, where only the data point immediately preceding the forecast is considered, as described

here; and the other, detailed in the subsequent Multi Lag LSTM section.

Multi Lag LSTM

The Multi Lag LSTM shares similarities with the LSTM model discussed in the preceding section. However, a key distinction lies in the Multi Lag LSTM approach, where multiple data points preceding the forecast can be incorporated. This inclusion of additional historical data provides the model with a richer information set for forecasting purposes.

3.1.3 Anomaly Detection

An anomaly, as defined by Hawkins, refers to an observation that deviates significantly from other observations to the extent that it raises suspicions about whether it was generated by a different mechanism [41]. These anomalies can arise from various factors, including but not limited to, financial fraud, security breaches, and terrorist activities [42]. Therefore, identifying these anomalies is crucial as it can provide insights and aid in addressing future scenarios where similar events may occur.

While some anomalies may be attributed to noise, errors, or undesired data, which are typically not useful, it's often advisable to remove such anomalies to improve data quality and create a cleaner dataset for anomaly detection. However, certain anomalies carry valuable information about their occurrence. In recent years, especially in time series analysis, researchers have increasingly focused on identifying and studying anomalies themselves to gain insight into the underlying mechanisms behind their occurrence [43]. Before looking into detecting anomalies, it's important to consider three different types of anomalies.

Anomaly Types

Point anomalies are characterized by one data point being anomalous than the rest of the data. Whereas contextual anomalies are defined when an anomaly is considered to be anomalous in a specific context. The last of the anomaly types are collective anomalies where a set of data points are anomalous compared to the rest of the data [42]. Figure 3.1 illustrates these anomalies in terms of time series.

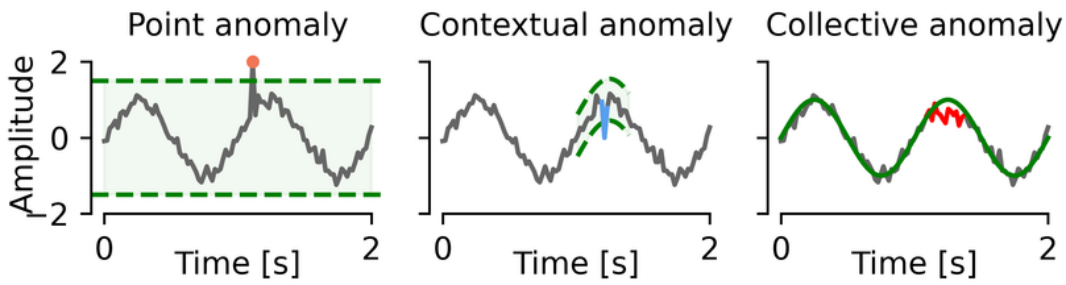


Figure 3.1: Three anomaly types [44].

Anomaly Detection Methods

Understanding the underlying mechanisms of anomalies relies on effectively identifying and analyzing anomalies. Insights gained from the analysis of anomalies can inform decision-making processes in many domains, enabling better handling of similar events in the future [42]. There are many ways of anomaly detection ranging from statistical methods to deep learning methods [45].

Time series decomposition has proven to be effective in anomaly detection in numerous studies [46], [47]. This technique involves breaking down a time series into three main components: seasonal, trend, and residual components [2]. Anomalies, by their very nature, deviate from the normal behaviour exhibited by the data, making them less likely to be captured in the seasonal or trend components of time series decomposition. Consequently, residue-based anomaly detection becomes feasible [48].

Seasonal and Trend decomposition using Loess (STL), or Seasonal and Trend decomposition using Loess, is a straightforward statistical method used for time series decomposition [49]. STL breaks down time series data into seasonal, trend, and residual components as depicted in Figure 3.2. In this context, particular attention is directed towards the residual component, as it serves to pinpoint anomalies within the time series data. Anomalies are identified as data points that exhibit deviations exceeding a multiple of the standard deviation. The focus on the residual component stems from the effectiveness of the models, described in the previous section, in adequately capturing the seasonality and trend within the time series data. Additionally, anomalies typically manifest as deviations from the established norms, rather than conforming to seasonal or trend patterns.

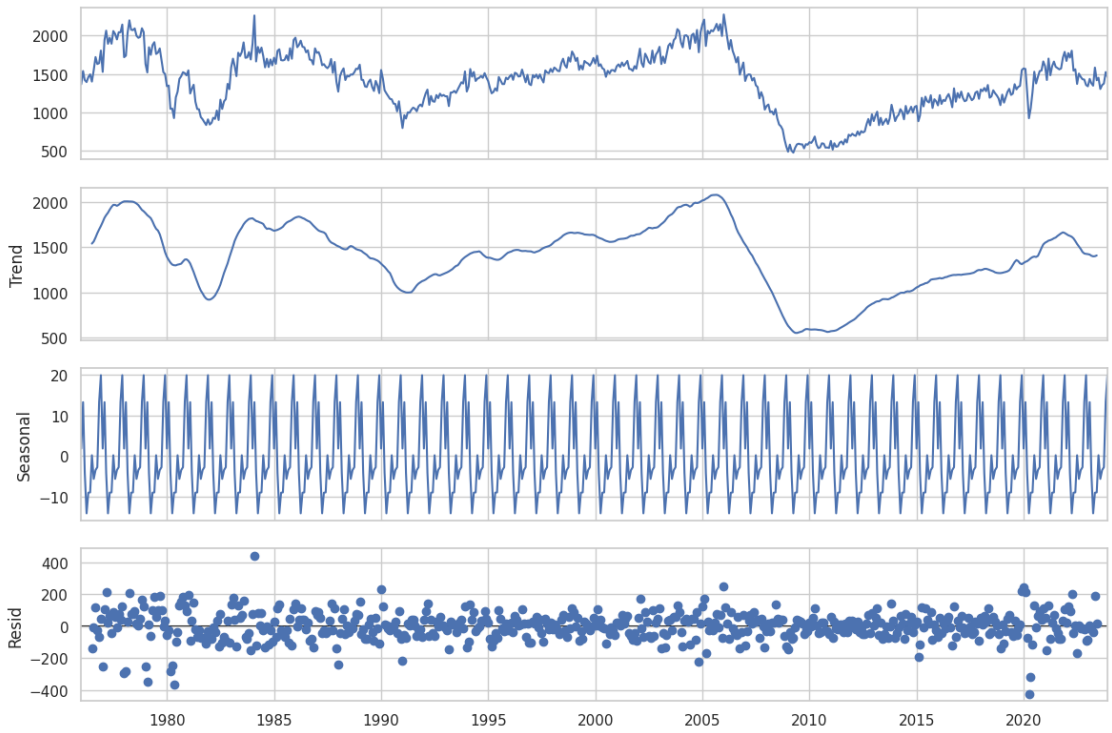
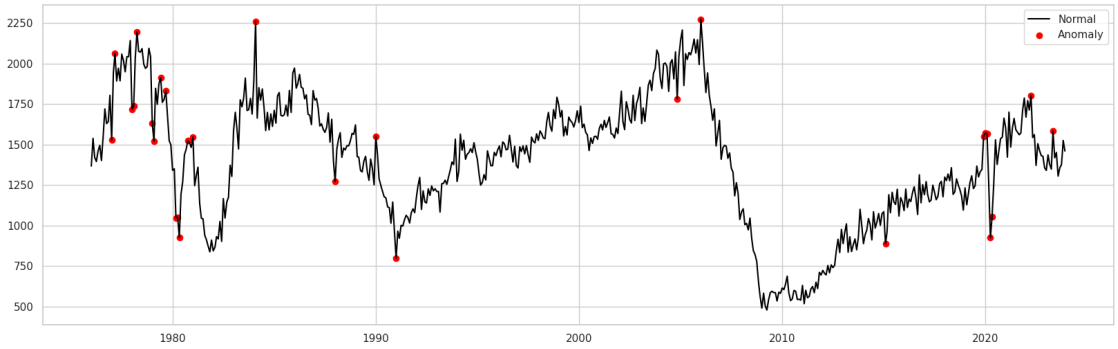


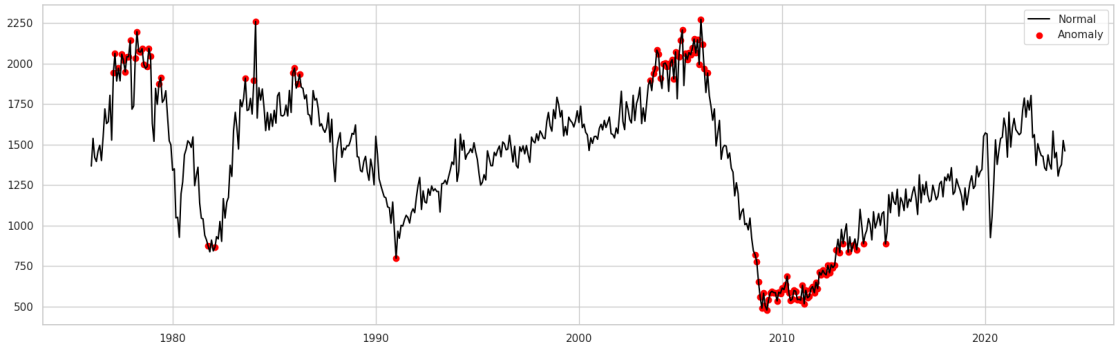
Figure 3.2: Decomposed Trend, Seasonality, and Residual components of HOUST variable of Primary Economic Dataset.

Isolation Forest is an anomaly detection method known for its linear time complexity [50]. Isolation trees exploit the inherent characteristic of anomalies, which is their rarity and distinctiveness, making them relatively easy to separate. Due to its linear time complexity, Isolation Forest serves as the foundation for many anomaly detection methods in the literature [51], [52]. It employs a contamination factor to specify the proportion of data points considered as anomalies within the dataset. This contamination factor serves as a configurable parameter. Notably, the anomalies detected by the Isolation Forest method are distinct from those detected by the STL method, as described in Section **??**. This distinction is depicted in figure 3.3.

Anomaly detection serves as the tool in this study for identifying significant events within time series data. Subsequently, these identified events are incorporated into the History Preservation method outlined later. The study utilized the mentioned statistical point anomaly detection techniques known for their simplicity and efficiency.



(a) Anomalies detected from STL method.



(b) Anomalies detected from Isolation Foreset method.

Figure 3.3: Anomaly detection of HOUST variable of the Primary Economic Dataset using STL (above) and Isolation Forest (below) methods.

3.2 History Preservation with Anomaly Detection

In this study, History Preservation is delineated as the process of capturing significant historical events and leveraging insights derived from them to enhance forecasting accuracy. However, to implement this approach, a method for identifying these significant historical events must be established.

During historically significant events, notable changes occur within the environment, often reflected as anomalous occurrences in time series data. Such anomalous events induce drastic shifts in the behavior and movement of related time series, thereby presenting an opportunity for beneficial insights. Consequently, in this study, anomalous events are defined as historically significant events. To identify these anomalous events, established anomaly detection methods in time series analysis are employed. Subsequently, the detected anomalies are incorporated into the forecasting process to facilitate learning to gain insights.

In this study, a data augmentation technique is employed to incorporate detected anomalies into the forecasting process. Anomalies are identified for each time series within the dataset based on various criteria utilizing the previously mentioned anomaly detection methods in section 3.1.3. Subsequently, each detected anomaly is assigned a score, thereby generating an anomaly score time series for each variable in the dataset. These individual anomaly score time series are then amalgamated into a single anomaly time series based on time. This aggregated anomaly time series is thereafter integrated into the original dataset as a new time series variable.

This data augmentation and scoring methodology serves to emphasize significant historical events that exert an impact across all time series variables. Moreover, it ensures that even events with relatively lesser impact are captured, albeit to a lesser extent. This approach enables the learning and forecasting of events based on their significance, rather than solely categorizing them as anomalies.

These augmented datasets are then utilized in the modeling process to facilitate learning and forecasting. The learning and forecasting tasks are executed using the three models outlined in Section 3.1.2. The evaluation plan is detailed in Section

3.3 to gauge the effectiveness of the proposed History Preservation method. Figure 3.4 depicts a high-level design of Forecasting with History Preservation.

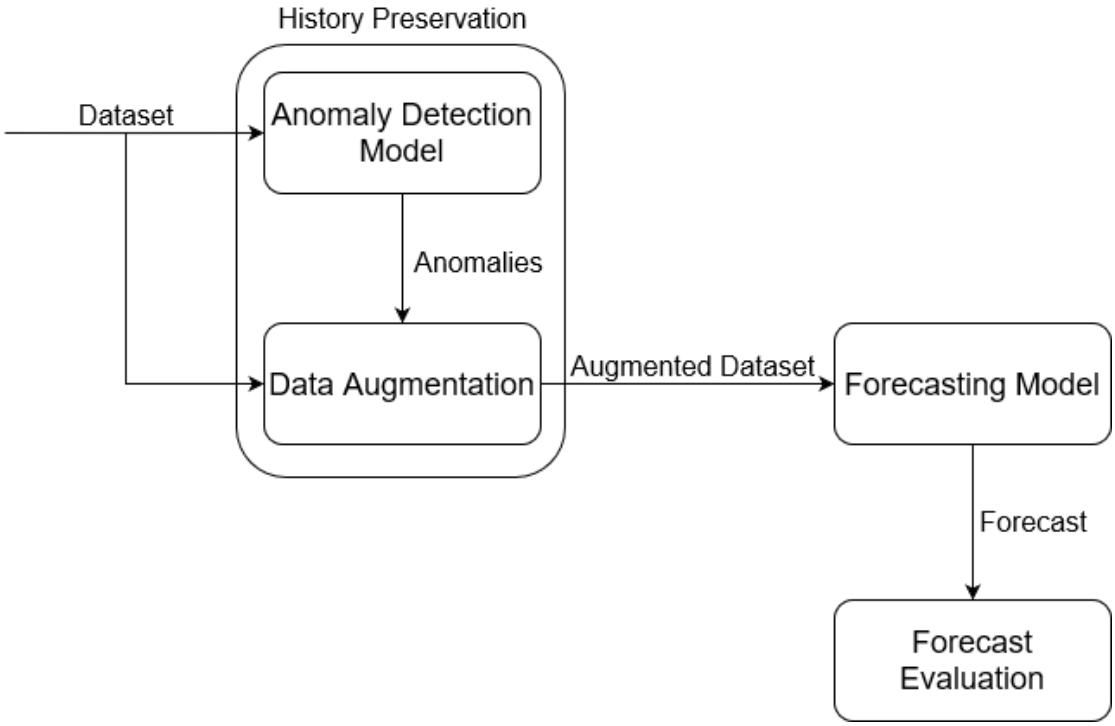


Figure 3.4: High-level design of Forecasting with History Preservation.

3.3 Evaluation Plan

An experiment suit was designed to evaluate the effectiveness of the proposed History Preservation method of this study. This experiment suit covers the impact of the proposed History Preservation method across models and datasets considering different anomaly methods used.

Experiment Name Format

[model]_[dataset]_[anomaly_method]

Models

- LightGBM
- LSTM
- Multi Lag LSTM

Datasets

- Primary - Primary Economic Dataset
- Beijing - Beijing Multi-Site Air-Quality Data Dataset

Anomaly Methods

- No data augmentation - anom_0
- Three anomaly detection methods derived from STL (anom_1, anom_2, anom_3)
- Three anomaly detection methods derived from Isolation Forests (iso_trs_0.1, iso_trs_0.2, iso_trs_0.3)

3.3.1 Metrics

Multivariate time series forecasting involves predicting multiple time series variables simultaneously. SMAPE is used as the evaluation metric for this study to evaluate the efficacy of the proposed history preservation method. SMAPE can be calculated for each individual variable and it measure the average percentage difference between the actual and predicted values for each variable.

Average of SMAPE are used to get a global understanding of the effectiveness of the proposed method of history preservation. Furthermore SMAPE is used to create the Weighted Symmetric Mean Absolute Percentage Error (WSMAPE) which allows another single evaluation score to be obtained.

SMAPE and WSMAPE

SMAPE is defined as follows. Where n is the number of data points, A_t is the actual value and F_t is the forecast value.

$$\text{SMAPE} = \frac{100}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2} \tag{3.1}$$

With the above SMAPE, WSMAPE shall be produced by introducing a weighting mechanism.

$$\text{WSMAPE} = \sum_{i=1}^{m} w_i \ * \quad \text{SMAPE}_i \tag{3.2}$$

The m above denotes the number of variables forecast in the multivariate time series forecast. Further w_i denotes the weight given for the i^{th} variable and SMAPE_i denotes the SMAPE of the i^{th} variable.

The weighting mechanism used in this study is based on the variance of variables. Where w_i is proportional to the variance of the i^{th} variable considered.

The WSMAPE defined here was inspired by the Weighted Root Mean Squared Scaled Error (WRMSSE) used in the M5 competition [4].

3.3.2 Environment

All experiments are conducted on Google Collaboratory with the following specifications.

- CPU: Intel Xeon CPU @2.20 GHz
- RAM: 13GB
- GPU: NVIDIA Tesla K80
- VRAM: 12GB DDR5

The language of implementation is Python in Collaboratory notebooks. As the file storage, Google Drive is used.

3.4 Research Tools

- LaTeX with overleaf for writing.
- SemanticScholar for tracking papers.
- Google Scholar and Semantic Scholar to search for papers.
- Google Drive, Docs, and Sheets for taking notes and compiling papers.
- Grammarly as a writing aid.
- TurnitIn as the plagiarism checker.

Chapter 4

Implementation and Results

4.1 Dataset Analysis

4.1.1 Primary Economic Dataset

As mentioned in section 3.1.1 due to the unavailability of quality multivariate time series data a new dataset was created for the purposes of this study. This dataset includes monthly data from 1976 to the end of 2023. Data in this dataset contains key economic indicators, global surface temperature, and atmospheric CO2 concentration [36].

Data Sources

- Federal Reserve Bank of St.Louise [53] (Variables 1 - 13)
- International Monetary Fund [54] (Variable 14)
- London Bullion Market Association [55] (Variables 15 - 16)
- National Centers for Environmental Information [56] (Variable 17)

Variables

1. Unemployment Rate (UNRATE)
2. Personal Consumption Expenditures: Chain-type Price Index (PCEPI)
3. Personal Consumption Expenditures Excluding Food and Energy (Chain-Type Price Index) (PCEPILFE)
4. Real Personal Income (RPI)

5. Personal Saving (PMSAVE)
6. Light Weight Vehicle Sales: Autos and Light Trucks (ALTSALES)
7. Industrial Production: Total Index (INDPRO)
8. Capacity Utilization: Total Index (TCU)
9. New Privately-Owned Housing Units Authorized in Permit-Issuing Places: Total Units (PERMIT)
10. New Privately-Owned Housing Units Started: Total Units (HOUST)
11. Labor Force Participation Rate (CIVPART)
12. Consumer Price Index for All Urban Consumers: All Items in U.S. City Average (CPIAUCSL)
13. Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average (CPILFESL)
14. Atmospheric CO2 Concentrations (AtmCO2)
15. Gold Price (goldMP)
16. Silver Price (silverMP)
17. Global Surface Temperature (globalTemp)

4.1.2 Beijing Multi-Site Air-Quality Data Dataset

This dataset contains the hourly readings of six air pollutants and six other relevant meteorological variables. The data is collected from twelve national air quality monitoring centers in Beijing, China [37]. There are around 35000 datapoints per monitoring center. Data is gathered from March 1st, 2013 to February 28th, 2017. Variables are as follows.

Air pollutant concentrations

- PM2.5 - Particulate Matter with diameter less than 2.5 μm
- PM10 - Particulate Matter with diameter less than 10 μm
- SO2 - Sulfur Dioxide
- NO2 - Nitrogen Dioxide
- CO - Carbon Monoxide
- O3 - Ozone

Meteorological variables

- TEMP - Air Temperature
- PRES - Pressure
- DEWP - Dew Point
- RAIN - Precipitation
- WD - Wind Direction
- WSPM - Wind Speed

4.1.3 Preprocessing

Before conducting experiments, it is crucial to preprocess the data into a format that the model can accept. This preprocessing entails various tasks such as handling missing values, removing duplicates, scaling numerical features, and encoding categorical data. Effective data preprocessing is critical for ensuring the quality and reliability of results, as well as optimizing model performance.

Primary Economic Dataset

The creation of the dataset draws from the four data sources outlined in Section 4.1.1. Python, Pandas, and NodeJS are utilized for this task. The resulting dataset comprises monthly data spanning from 1976 to 2023.

For sourcing key economic indicators, the Federal Reserve Bank of St.Louise serves as the data repository. Federal Reserve Bank of St.Louise offers an Application Programming Interface (API) for data retrieval, and the fred-py-api library [57] is employed to interface with this API. Utilizing the API, data spanning from 1976 to 2023 for the variables(1-13) mentioned in Section 4.1.1 is obtained. Subsequently, the retrieved data is merged into a single Comma Separated Values (CSV) file based on the date attribute. This process results in the creation of a CSV file with data on economic indicators.

For the Atmospheric CO2 Concentrations, the IMF provides a downloadable CSV file containing this data. Initially, the original dataset contained numerous variables that were irrelevant to the scope of this study. Therefore, all such extraneous variables were discarded, retaining only the 'Date', 'Unit', and 'Value' columns.

Given that the dataset included values from various units for a single date, filtering was performed to isolate data with the parts per million unit. Thereafter, the 'Unit' column was dropped from the dataset. Finally, data spanning from 1976 to 2023 was filtered and saved into a CSV file.

For gold and silver prices, data from the London Bullion Market Association was utilized. Although LBMA did not provide downloadable files or an API, the data was publicly accessible. To extract this data from the website, a JavaScript script was developed within NodeJS. This script scraped the data from the London Bullion Market Association website and stored it in two separate CSV files, one for gold prices and another for silver prices. The data extracted consisted of daily prices spanning from 1969 to 2023. Following this extraction, the next phase involved transforming the daily data into monthly data using Python and Pandas. This transformation entailed selecting the first data point of each month to represent that month's data. Then, the dataset was filtered to include data from 1976 to 2023. This process resulted in the creation of two CSV files with gold and silver price data.

For global temperature data, the National Centers for Environmental Information has a downloadable CSV file. This file was obtained from the National Centers for Environmental Information and filtered to extract data spanning from 1976 to 2023. The resulting dataset was saved into a CSV file.

Upon creation of the individual files for gold and silver prices, atmospheric CO2 concentrations, global temperature, and key economic indicators, they were merged based on their date attributes to form the final Primary Economic Dataset for this study. This dataset did not contain any missing values and was normalized during use. This dataset covers the last five decades of economic data and other factors that impact the economy, allowing us to gain insights into historically significant events that happened during that time.

Beijing Multi-Site Air-Quality Data Dataset

The preprocessing of this dataset was partly inspired by the paper by Li et al [58].

First, a station was chosen at random out of the 12 available. Then some of the unnecessary columns shall be dropped like the column with the station name.

Since wind direction is categorical it is dropped as well.

There are missing values in the dataset that need to be handled, for this the K-Nearest Neighbors imputation is used. The number of neighbors considered here is five and the weights of neighbors are uniform.

Numerical values are on different scales. This is not ideal as it can lead to variables having different levels of influence on the model during training. Hence these numeric data are normalized.

4.1.4 Data Augmentation

Data augmentation via anomaly detection serves as a crucial component of the proposed History Preservation method. This process hinges on the detection and scoring of anomalies, leading to the creation of individual anomaly time series for each variable within the dataset. Afterward, these anomaly time series are aggregated to form an additional variable in the dataset. For anomaly detection two methods used are the STL and Isolation Forest methods.

Seasonal and Trend decomposition using Loess (STL)

Seasonal and Trend decomposition using Loess is a methodology designed to decompose time series data into three distinct components: seasonal, trend, and residual. As detailed in Section **??**, the residual component is specifically leveraged for anomaly detection in this study. Within the context of this research, a data point is classified as anomalous if its corresponding residual component exceeds twice the standard deviation of the overall residual component.

Upon detection of an anomaly, it is scored using three distinct methods tailored for STL, as outlined in Section 3.3.

- anom_1
  - Anomaly val = a_v : (currently $a_v = 5$)
  - Non anomaly val = a_n : (currently a_n = -5)
  - No augmentation hence if an anomaly is detected that data point gets the value of a_v else the value is a_n.
- anom_2
  - Anomaly val = a_v : (currently $a_v = 5$)

  - Non anomaly val = a_n : (currently a_n = -5)
  - std_{resid} = standard deviation of residual values (from stl decomposition)
  - Anomaly score augmentation = $(|y_i - std_{resid}|/std_{resid}) * a_v$
- anom_3
  - Anomaly val = a_v : (currently a_v = 5)
  - Non anomaly val = a_n : (currently a_n = -5)
  - std_{resid} = standard deviation of residual values (from stl decomposition)
  - Anomaly score augmentation = $(|y_i - std_{resid}|) * a_v$

Isolation Forests

Isolation Forests represent an anomaly detection technique that identifies anomalies based on a contamination factor. This factor delineates the proportion of the dataset expected to be anomalous. It's important to note that Isolation Forests detect anomalies distinct from those identified by the STL method. In this study, three contamination factors were employed to enable three levels of anomaly detection.

- iso_trs_0.1
  - Contamination factor = 0.1
  - Anomaly val = a_v : (currently a_v = 5)
  - Non anomaly val = a_n : (currently a_n = -5)
  - If an anomaly is detected that data point gets the value of a_v else the value is a_n.
- iso_trs_0.2
  - Contamination factor = 0.2
  - Anomaly val = a_v : (currently a_v = 5)
  - Non anomaly val = a_n : (currently a_n = -5)
  - If an anomaly is detected that data point gets the value of a_v else the value is a_n.
- iso_trs_0.3
  - Contamination factor = 0.3
  - Anomaly val = a_v : (currently a_v = 5)
  - Non anomaly val = a_n : (currently a_n = -5)

– If an anomaly is detected that data point gets the value of a_v else the value is a_n.

Anomaly Time Series Aggregation

In both of these methods, after scoring anomalies for each time series variable in the dataset, a collection of anomaly time series is generated, corresponding to each variable. These anomaly time series are subsequently aggregated. The chosen aggregation method in this study is summation. Therefore, all time series are aligned based on date and summed accordingly. This process yields the aggregated anomaly time series, which is then integrated into the original dataset.

The aggregation method is represented by Equation 4.1. In this equation, $AgreTS^d$ denotes the value at date d of the aggregated anomaly time series, while $AnomTS_i^d$ represents the value at date d of the i^{th} anomaly time series. And n denotes the number of time series variables in the original dataset.

$$AgreTS^d = \sum_{i=0}^{n} AnomTS_i^d \tag{4.1}$$

4.2 Models

The following models were implemented according to the descriptions provided below, and the experiments outlined in Section 3.3 were executed using these models. It's worth noting that once these models are defined, they remain unchanged throughout the study. This approach is adopted because the study primarily focuses on evaluating the effects of History Preservation through data augmentation, necessitating consistency in the models utilized.

4.2.1 LightGBM

The model definition for LightGBM draws inspiration from the winning model of the M5 competition [4]. However, certain parameters have been adjusted to better suit the requirements of this study. The parameters are as follows:

```
lgb_params = {'boosting_type': 'gbdt',
```

```
'objective': 'regression',
'tweedie_variance_power': 1.1,
'metric': 'rmse',
'subsample': 0.5,
'subsample_freq': 1,
'learning_rate': 0.015,
'num_leaves': 3,
'min_data_in_leaf': 10,
'feature_fraction': 0.5,
'max_bin': 100,
'n_estimators': 3000,
'boost_from_average': False,
'verbose': -1,
'seed' : 1995}
```

4.2.2 LSTM

The LSTM utilized in this study is defined as follows: it comprises a single LSTM layer with 50 neurons, followed by a dense layer serving as the output layer. The model is then trained for 50 epochs, with a batch size of 72, utilizing the MAE loss function and employing the Adam optimizer. The lag of this LSTM model is set to one.

4.2.3 Multi Lag LSTM

The Multi Lag LSTM model bears significant similarity to the LSTM model defined in Section 4.2.2, with the primary distinction lying in the lag parameter. In this variant, the lag is defined to be 3, indicating that the model considers the previous three data points for each forecast.

4.3 Experiments: Primary Economic Dataset

4.3.1 LightGBM

Baseline Experiments

Experiment name : LightGBM_primary_anom_0

Anomaly Method : anom_0 (No data augmentation)

Metric	Value
avg_SMAPE	12.50059
WSMAPE	12.64186

Table 4.1: Metrics for LightGBM with no data augmentation for Primary Economic dataset

Experiments with History Preservation

It's notable that both anomaly detection methods, STL and Isolation Forest, yielded improvements over the baseline performance when applied to the primary economic dataset. Specifically, in terms of accuracy enhancement measured by WSMAPE, STL demonstrated an improvement of **10.36%**, while Isolation Forest exhibited an improvement of **5.75%**.

Anomaly Method: STL

Metric	Baseline	anom_1	anom_2	anom_3
avg_SMAPE	12.50059	12.19176	**12.1**	13.31529
WSMAPE	12.64186	**11.33156**	11.76063	15.82842

Table 4.2: Metrics for LightGBM with STL data augmentation for Primary Economic dataset

Anomaly Method: Isolation Forests

Metric	Baseline	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
avg_SMAPE	12.50059	12.38706	**12.32352**	12.38118
WSMAPE	12.64186	12.73511	11.96875	**11.91537**

Table 4.3: Metrics for LightGBM with Isolation Forest data augmentation for Primary Economic dataset

4.3.2 LSTM

Baseline Experiments

Experiment name : LSTM_primary_anom_0

Anomaly Method : anom_0 (No data augmentation)

Metric	Value
avg_SMAPE	12.36765
WSMAPE	10.52907

Table 4.4: Metrics for LSTM with no data augmentation for Primary Economic dataset

Experiments with History Preservation

History preservation through STL did not result in an improvement over the baseline accuracy for the LSTM model when applied to the primary economic dataset. However, in contrast, when utilizing Isolation Forest for history preservation, there was a significant improvement in accuracy, with an enhancement of **17.71%** measured by WSMAPE.

Anomaly Method: STL

Metric	Baseline	anom_1	anom_2	anom_3
avg_SMAPE	**12.36765**	13.01647	13.42	13.19765
WSMAPE	**10.52907**	10.99233	11.12231	11.99993

Table 4.5: Metrics for LSTM with STL data augmentation for Primary Economic dataset

Anomaly Method: Isolation Forests

Metric	Baseline	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
avg_SMAPE	12.36765	13.02412	**11.61824**	11.86882
WSMAPE	10.52907	11.8469	10.06988	**8.66456**

Table 4.6: Metrics for LSTM with Isolation Forest data augmentation for Primary Economic dataset

4.3.3 Multi Lag LSTM

Baseline Experiments

Experiment name : multilagLSTM_primary_anom_0

Anomaly Method : anom_0 (No data augmentation)

Metric	Value
avg_SMAPE	3.17412
WSMAPE	3.60416

Table 4.7: Metrics for multilagLSTM with no data augmentation for Primary Economic dataset

Experiments with History Preservation

It's evident that the Multi Lag LSTM model already exhibits notable performance compared to both LightGBM and single lag LSTM. Additionally, with the incorporation of history preservation through both STL and Isolation Forest methods, the Multi Lag LSTM model demonstrates further improvements over the baseline accuracy measured by WSMAPE. Specifically, STL resulted in an accuracy enhancement of **17.22%**, while Isolation Forest yielded an improvement of **16.53%**.

Anomaly Method: STL

Metric	Baseline	anom_1	anom_2	anom_3
avg_SMAPE	3.17412	3.07647	**2.85588**	3.62882
WSMAPE	3.60416	3.15175	**2.9834**	3.72916

Table 4.8: Metrics for multilagLSTM with STL data augmentation for Primary Economic dataset

Anomaly Method: Isolation Forests

Metric	Baseline	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
avg_SMAPE	3.17412	**2.64**	3.55059	3.01529
WSMAPE	3.60416	**3.00847**	4.42669	3.45007

Table 4.9: Metrics for multilagLSTM with Isolation Forest data augmentation for Primary Economic dataset

4.4 Experiments: Beijing Multi-Site Air-Quality Data Dataset

4.4.1 LightGBM

Baseline Experiments

Experiment name : LightGBM_beijing_anom_0

Anomaly Method : anom_0 (No data augmentation)

Metric	Value
avg_SMAPE	23.48182
WSMAPE	1.36373

Table 4.10: Metrics for LightGBM with no data augmentation for Beijing dataset

Experiments with History Preservation

History Preservation with both STL and Isolation Forest did not yield any accuracy improvement over baseline with the Beijing Dataset using LightGBM.

Anomaly Method: STL

Metric	Baseline	anom_1	anom_2	anom_3
avg_SMAPE	**23.48182**	24.02727	24.04273	24.69182
WSMAPE	**1.36373**	1.53443	1.52986	2.24453

Table 4.11: Metrics for LightGBM with STL data augmentation for Beijing dataset

Anomaly Method: Isolation Forests

Metric	Baseline	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
avg_SMAPE	**23.48182**	24.24	24.16364	24.11636
WSMAPE	**1.36373**	1.67819	1.59637	1.61633

Table 4.12: Metrics for LightGBM with Isolation Forest data augmentation for Beijing dataset

4.4.2 LSTM

Baseline Experiments

Experiment name : LSTM_beijing_anom_0

Anomaly Method : anom_0 (No data augmentation)

Metric	Value
avg_SMAPE	19.74182
WSMAPE	16.90485

Table 4.13: Metrics for LSTM with no data augmentation for Beijing dataset

Experiments with History Preservation

For the Beijing Dataset, employing history preservation with STL using LSTM did not result in any accuracy improvement over the baseline. However, when utilizing history preservation with Isolation Forest, there was an improvement in accuracy, with an enhancement of **4.63%** measured by WSMAPE.

Anomaly Method: STL

Metric	Baseline	anom_1	anom_2	anom_3
avg_SMAPE	**19.74182**	21.19818	21.22909	22.09818
WSMAPE	**16.90485**	17.41566	16.95142	17.61226

Table 4.14: Metrics for LSTM with STL data augmentation for Beijing dataset

Anomaly Method: Isolation Forests

Metric	Baseline	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
avg_SMAPE	**19.74182**	20.19	20.52545	21.38636
WSMAPE	16.90485	16.88637	17.27897	**16.12286**

Table 4.15: Metrics for LSTM with Isolation Forest data augmentation for Beijing dataset

4.4.3 Multi Lag LSTM

Baseline Experiments

Experiment name : multilagLSTM_beijing_anom_0

Anomaly Method : anom_0 (No data augmentation)

Metric	Value
avg_SMAPE	4.88091
WSMAPE	4.00758

Table 4.16: Metrics for multilagLSTM with no data augmentation for Beijing dataset

Experiments with History Preservation

It's evident that with Multi Lag LSTM, both history preservation methods, STL and Isolation Forest, led to improvements in accuracy measured by WSMAPE. Specifically, STL resulted in an enhancement of **39.71%**, while Isolation Forest yielded an improvement of **31.37%**.

Anomaly Method: STL

Metric	Baseline	anom_1	anom_2	anom_3
avg_SMAPE	4.88091	4.23273	4.33636	**3.65636**
WSMAPE	4.00758	4.53384	3.38652	**2.41604**

Table 4.17: Metrics for multilagLSTM with STL data augmentation for Beijing dataset

Anomaly Method: Isolation Forests

Metric	**Baseline**	**iso_trs_0.1**	**iso_trs_0.2**	**iso_trs_0.3**
avg_SMAPE	4.88091	4.25364	**3.06727**	5.39
WSMAPE	4.00758	7.3688	**2.75048**	6.96481

Table 4.18: Metrics for multilagLSTM with Isolation Forest data augmentation for Beijing dataset

Chapter 5

Analysis and Conclusions

5.1 Introduction

This study investigates the utility of history preservation to aid multivariate time series forecasting in terms of accuracy and efficiency. Anomaly detection and data augmentation are utilized as the history preservation method in the study. This chapter covers the conclusions derived from the study and discusses the implications for future research.

5.2 Discussion

When a model learns to forecast, it generally learns to predict based on common patterns or the generalized scenario. However, when an event deviates from this norm, the model often fails to perform accurately. These deviations typically correspond to significant events with considerable impact, and if forecasted correctly, they can offer substantial benefits. To address this issue, history preservation is employed to capture these critical scenarios where notable changes occur.

Conventional models struggle to forecast significant events because they are primarily trained to forecast generalized patterns. This study demonstrates that history preservation methods based on anomaly detection can identify these significant events, leading to improved forecasting accuracy when this new information is integrated. The proposed technique detects significant events through anomaly detection and incorporates this information into the dataset via data augmentation, as outlined in section 4.1.4. As a result, during model training, the model

becomes aware of significant events, gaining insight into these deviations from the generalized scenario. This additional context allows the model to forecast based not only on generalized scenarios but also on significant events.

The study employed two different anomaly detection methods to identify significant events, as described in section 3.1.3. By integrating this anomaly-based information into the training process, the forecasting models achieved greater accuracy, specifically in scenarios involving significant deviations from expected patterns.

Using STL-based history preservation, out of the six scenarios represented by three models and two datasets, only three instances demonstrated improved accuracy with history preservation. Among the three scenarios that did not show improvement, the poor performance of LightGBM can be attributed to its inability to effectively handle periodic time series. For instance, the 'RAIN' attribute in the Beijing dataset exhibits periodic behavior, and rain is known to influence air quality dynamics [59], [60]. On the other hand, LSTM and Multi Lag LSTM demonstrate proficiency in handling the 'RAIN' attribute in their forecasting efforts. However, LSTM fails to improve upon the baseline accuracy, mainly due to its consideration of a single data point for forecasting. This limitation restricts its ability to encounter anomalies and leverage anomaly information for forecasting purposes. In contrast, Multi Lag LSTM considers multiple data points for forecasting, enabling it to better utilize anomaly information, leading to improved performance, as evidenced in Table 5.1.

Dataset	**LightGBM**	**LSTM**	**Multi Lag LSTM**
Primary	10.36%	Not improved	17.22%
Beijing	Not improved	Not improved	39.71%

Table 5.1: With anomaly detection using STL, accuracy improvement over baseline of the three models used in this study.

Isolation Forest-based history preservation also encompasses six instances, comprising two datasets with three models. Out of these six instances, five exhibited improvement over the baseline, as shown in Table 5.2. The single instance that did not demonstrate improvement was LightGBM with the Beijing dataset, which is due to the periodic 'RAIN' attribute in the Beijing dataset. However, with Iso-

lation Forest-based history preservation, LSTM improved over the baseline. This improvement can be attributed to the fact that the Isolation Forest method detects anomalies that are more closely grouped together compared to STL, which detects anomalies that are more spaced apart. This distinction can be observed in Figure 3.3 in Section 3.1.3. The clustered nature of anomalies allows LSTM more opportunities to encounter anomalies, even with its consideration of only a single prior data point for forecasting. Similarly, as seen before, Multi Lag LSTM improves over the baseline by considering multiple data points. Just as forecasting benefits from considering more prior data points, the history preservation method proposed in this study also lends itself to improvement with increased consideration of prior data points.

Dataset	**LightGBM**	**LSTM**	**Multi Lag LSTM**
Primary	5.75%.	17.71%	16.53%
Beijing	Not improved	4.63%	31.37%

Table 5.2: With anomaly detection using Isolation Forest, accuracy improvement over baseline of the three models used in this study.

The proposed history preservation technique affects the efficiency of multivariate time series forecasting. For evaluating the efficiency of multivariate time series forecasting, it's essential to consider both the time and memory complexity of the specific model used for the forecast. This serves as the baseline. In this study, the baseline involves forecasting with the three models: LightGBM, LSTM, and Multi Lag LSTM, without employing the history preservation method proposed in this study.

With the incorporation of history preservation, an additional process is introduced. History preservation is achieved through anomaly detection and data augmentation. For anomaly detection, STL and Isolation Forest methods are utilized. Data augmentation involves integrating an additional time series attribute into the dataset used for forecasting. Due to the anomaly detection and data augmentation processes, there is an increase in time complexity, as well as memory complexity, due to the introduction of more data into the dataset. Consequently, overall efficiency in multivariate time series forecasting decreases with history preservation.

This decrease in efficiency is primarily dependent on the anomaly detection and data augmentation methods employed. However, in this study, the use of simple anomaly detection methods and data augmentation techniques helps to minimize the decrease in efficiency.

5.3 Conclusions About Research Questions

The first research question was to identify the techniques utilized in multivariate time series forecasting. Through a comprehensive literature survey, various techniques were identified, and three machine learning models were selected for further investigation in the study.

The second research question sought to adapt identified techniques for history preservation. To achieve this, the study proposes a history preservation technique that uses anomaly detection to identify significant events and integrates this anomaly information into datasets through data augmentation, as detailed in section 4.1.4. By doing so, the model, during training, becomes aware of the significant events in the data, gaining insights from these deviations. This approach allows the model to forecast based not only on the generalized scenario but also on significant events that deviate from typical patterns, leading to greater accuracy in forecasting.

The third research question focused on evaluating the efficacy of the history preservation technique, gauged in terms of accuracy and efficiency in multivariate time series forecasting. To accomplish this, experiments were conducted using two anomaly detection methods, three machine learning models, and two datasets. Results indicated that although there was a slight decrease in efficiency, accuracy notably improved over the baseline in eight out of a possible twelve instances. Furthermore, five out of the eight improved instances exhibited accuracy improvements exceeding 15% over the baseline.

5.4 Conclusions About Research Problem

This study explores significant historical events and how capturing information from them can aid multivariate time series forecasting. The identification of these

events was accomplished using anomaly detection methods in time series analysis. By augmenting the data with insights derived from these historical events, the accuracy of multivariate time series forecasting improved. However, this increase in accuracy came at the expense of a slight decrease in forecasting efficiency. Thus, history preservation based on anomaly detection enhances the accuracy of multivariate time series forecasting, but with a trade-off in efficiency. This approach allows models to forecast based not only on generalized scenarios but also on significant events that deviate from generalized scenarios, leading to more accurate forecasts.

5.5 Limitations and Implications for Further Research

The proposed history preservation technique demonstrates that when a model is trained with information about significant historical events, it can forecast based on not only generalized scenarios but also these significant events, leading to improved forecasting accuracy. To identify these significant events, the study used simple statistical anomaly detection methods. However, these methods do not fully capture the complexity and diversity of significant historical events.

To enhance the proposed history preservation technique, more sophisticated anomaly descriptors that can better represent the intricacies of these events should be explored. Heuristic-based approaches that leverage autoencoders, variational autoencoders, and generative adversarial networks could be promising areas for further study. Moreover, additional research is required to determine how these advanced anomaly descriptors can be seamlessly integrated into the forecasting process to further boost accuracy and improve the overall performance of multivariate time series forecasting.

Another area for further exploration is the optimization of the forecasting models themselves. Throughout the study, the models remained unchanged, providing a stable baseline for the research. However, there's a considerable opportunity to refine these models in combination with improving the history preservation technique. This could involve exploring various model architectures, tuning hyperparameters,

or experimenting with ensemble methods to achieve better predictive performance.

Bibliography

[1] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. T. Lora, "Deep learning for time series forecasting: A survey," *Big data*, 2020.

[2] R. J. Hyndman and G. Athanasopoulos, "Forecasting: Principles and practice," 2013.

[3] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting*, 2020.

[4] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions," *International Journal of Forecasting*, 2022.

[5] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, pp. 54–74, 2020.

[6] A. Borovykh, S. M. Bohté, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv: Machine Learning*, 2017.

[7] A. Faisal, A. Rahman, M. T. M. Habib, A. H. Siddique, M. Hasan, and M. Khan, "Neural networks based multivariate time series forecasting of solar radiation using meteorological data of different cities of bangladesh," *Results in Engineering*, 2022.

[8] Y. Mu, M. Wang, X. Zheng, and H. Gao, "An improved lstm-seq2seq-based forecasting method for electricity load," in *Frontiers in Energy Research*, 2023.

[9] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning : A systematic literature review: 2005-2019," *ArXiv*, vol. abs/1911.13288, 2019. [Online]. Available: `https://api.semanticscholar.org/CorpusID:208513396`.

[10] I. Ghalehkhondabi, E. Ardjmand, G. R. Weckman, and W. A. Young, "An overview of energy demand forecasting methods published in 2005–2015," *Energy Systems*, vol. 8, pp. 411–447, 2017. [Online]. Available: `https://api.semanticscholar.org/CorpusID:156838253`.

[11] P. R. P. G. Hewage, A. Behera, M. Trovati, *et al.*, "Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station," *Soft Computing*, vol. 24, pp. 16 453–16 482, 2020. [Online]. Available: `https://api.semanticscholar.org/CorpusID:218994062`.

[12] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," *Renewable & Sustainable Energy Reviews*, vol. 74, pp. 902–924, 2017.

[13] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, "Fforma: Feature-based forecast model averaging," *International Journal of Forecasting*, 2020.

[14] Y. Cheng, C. Guo, K. Chen, *et al.*, "A pattern discovery approach to multivariate time series forecasting," *ArXiv*, vol. abs/2212.10306, 2022.

[15] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *ArXiv*, vol. abs/1912.09363, 2019.

[16] C. Fan, Y. Zhang, Y. Pan, *et al.*, "Multi-horizon time series forecasting with temporal attention learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2527–2535, ISBN: 9781450362016. DOI: 10.1145/3292500.3330662. [Online]. Available: https://doi.org/10.1145/3292500.3330662.

[17] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, 2019.

[18] Z. Li, J. He, H. Liu, and X. Du, "Combining global and sequential patterns for multivariate time series forecasting," *2020 IEEE International Conference on Big Data (Big Data)*, pp. 180–187, 2020.

[19] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, "A memory-network based solution for multivariate time-series forecasting," *ArXiv*, vol. abs/1809.02105, 2018.

[20] L. Pantiskas, K. Verstoep, and H. E. Bal, "Interpretable multivariate time series forecasting with temporal attention convolutional neural networks," *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1687–1694, 2020.

[21] S. B. Roy, M. Yuan, Y. Fang, and M. K. Sett, "Spatio-temporal attention with symmetric kernels for multivariate time series forecasting," *2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 21–26, 2022.

[22] S. Smyl, G. Dudek, and P. Pełka, "Contextually enhanced es-drnn with dynamic attention for short-term load forecasting," *ArXiv*, vol. abs/2212.09030, 2022.

[23] L. Huang, F. Mao, K. Zhang, and Z. Li, "Spatial-temporal convolutional transformer network for multivariate time series forecasting," *Sensors (Basel, Switzerland)*, vol. 22, 2022.

[24] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[25] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A*, vol. 379, 2020.

[26] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585, 2016.

[27] C. S. Lea, R. Vidal, A. Reiter, and G. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *ECCV Workshops*, 2016. [Online]. Available: `https://api.semanticscholar.org/CorpusID:12414640`.

[28] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *ArXiv*, vol. abs/1803.01271, 2018.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[30] A. Vaswani, N. M. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *NIPS*, 2017.

[31] H. Zhou, S. Zhang, J. Peng, *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *ArXiv*, vol. abs/2012.07436, 2020.

[32] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, 2018.

[33] F. Liu, M. Cai, L. Wang, and Y. Lu, "An ensemble model based on adaptive noise reducer and over-fitting prevention lstm for multivariate time series forecasting," *IEEE Access*, vol. 7, pp. 26 102–26 115, 2019.

[34] E. F. Urbinate, L. K. Felizardo, and E. Del-Moral-Hernandez, "Deep learning stacking for financial time series forecasting: An analysis with synthetic and real-world time series," *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2022)*, 2022.

[35] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, pp. 679–688, 2006.

[36] K. Mendis, *Primary-economic-dataset*, `https://github.com/KM-drago/primary-economic-dataset.git`.

[37] S. Chen, *Beijing Multi-Site Air-Quality Data*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5RK5G, 2019.

[38] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, "Cautionary tales on air-quality improvement in beijing," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, 2017.

[39] R. Buizza, "Chaos and weather prediction january 2000," 2002.

[40] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, 2018.

[41] D. M. Hawkins, "Identification of outliers," in *Monographs on Applied Probability and Statistics*, 1980. [Online]. Available: `https://api.semanticscholar.org/CorpusID:128775129`.

[42] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, 15:1–15:58, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:207172599.

[43] A. Bl'azquez-Garc'ia, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys (CSUR)*, vol. 54, pp. 1–33, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211075794.

[44] P. Yan, A. Abdulkadir, P.-P. Luley, *et al.*, "A comprehensive survey of deep transfer learning for anomaly detection in industrial time series: Methods, applications, and directions," *IEEE Access*, vol. 12, pp. 3768–3789, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259837291.

[45] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: A comprehensive evaluation," *Proc. VLDB Endow.*, vol. 15, pp. 1779–1797, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250331153.

[46] J. Gao, X. Song, Q. Wen, P. Wang, L. Sun, and H. Xu, "Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks," *ArXiv*, vol. abs/2002.09545, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211258817.

[47] J. Hochenbaum, O. Vallis, and A. Kejariwal, "Automatic anomaly detection in the cloud via statistical learning," *ArXiv*, vol. abs/1704.07706, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:6096711.

[48] A. González-Muñiz, I. D. Blanco, A. A. C. Vega, D. García-Pérez, and D. Pérez, "Two-step residual-error based approach for anomaly detection in engineering systems using variational autoencoders," *Comput. Electr. Eng.*, vol. 101, p. 108 065, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250316513.

[49] R. B. Cleveland, "Stl : A seasonal-trend decomposition procedure based on loess," 1990. [Online]. Available: https://api.semanticscholar.org/CorpusID:64570714.

[50] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:6505449.

[51] P.-F. Marteau, S. Soheily-Khah, and N. Béchet, "Hybrid isolation forest - application to intrusion detection," *ArXiv*, vol. abs/1705.03800, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:27977659.

[52] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:208016559.

[53] *Federal reserve bank of st. louise*, USA. [Online]. Available: https://fred.stlouisfed.org/.

[54] *International monetary fund.* [Online]. Available: https://climatedata.imf.org/.

[55] *The independent precious metals authority*. [Online]. Available: https://www.lbma.org.uk/.

[56] *National centers for environmental information*, USA. [Online]. Available: https://www.ncei.noaa.gov/.

[57] P. Zachary Spar, *Fred-py-api*, https://github.com/zachspar/fred-py-api.

[58] D. Li, J. Liu, and Y. Zhao, "Prediction of multi-site pm2.5 concentrations in beijing using cnn-bi lstm with cbam," *Atmosphere*, vol. 13, no. 10, 2022, ISSN: 2073-4433. DOI: 10.3390/atmos13101719. [Online]. Available: https://www.mdpi.com/2073-4433/13/10/1719.

[59] Z. Liu, L. Shen, C. Yan, J. Du, Y. Li, and H. Zhao, "Analysis of the influence of precipitation and wind on pm2.5 and pm10 in the atmosphere," *Advances in Meteorology*, vol. 2020, pp. 1–13, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:225431711.

[60] X. Feng and S. Wang, "Influence of different weather events on concentrations of particulate matter with different sizes in lanzhou, china.," *Journal of environmental sciences*, vol. 24 4, pp. 665–74, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:43325902.

Appendices

Appendix A

Results

Results tables are on the next page onwards.

EXPERIMENT_NAME	anom_0	anom_1	anom_2	anom_3	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
UNRATE	33.93	34.49	34.4	35.15	34.95	34.35	34.55
PCEPI	1.08	1.25	1.24	1.73	1.23	1.21	1.22
PCEPILFE	1.02	1.02	1.01	1.41	1.01	1.0	0.99
RPI	0.24	0.25	0.26	1.09	0.27	0.25	0.27
PMSAVE	15.76	15.32	15.35	14.61	15.27	15.27	15.03
ALTSALES	14.1	14.44	13.61	17.48	14.21	14.56	15.02
INDPRO	13.71	13.75	13.63	12.81	12.68	13.28	13.18
TCU	16.35	15.98	16.03	15.33	14.91	15.47	15.87
PERMIT	10.72	11.35	11.64	18.48	13.92	11.98	12.02
HOUST	15.48	12.09	12.88	15.95	12.99	12.92	12.84
CIVPART	21.04	21.26	21.17	21.82	21.34	21.29	21.08
CPIAUCSL	2.8	2.76	2.8	3.0	2.85	2.83	2.9
CPILFESL	3.2	2.37	2.39	3.45	2.43	2.42	2.38
AtmCO2	12.1	12.11	12.32	12.9	12.53	12.35	12.27
goldMP	9.11	10.48	9.6	11.78	11.24	10.77	10.45
silverMP	34.01	29.97	27.44	30.77	30.93	30.49	31.52
globalTemp	7.86	8.37	9.93	8.6	7.82	9.06	8.89

Table A.1: LightGBM - Primary Economic Dataset - Individual SMAPE

EXPERIMENT_NAME	anom_0	anom_1	anom_2	anom_3	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
PM2.5	4.18	4.99	4.92	4.73	5.19	5.28	5.06
PM10	3.79	4.47	4.63	4.66	4.66	4.59	4.52
SO2	6.18	5.51	5.64	5.83	5.81	5.62	5.68
NO2	2.69	2.96	2.95	3.79	2.99	2.98	3.08
CO	1.26	1.37	1.34	2.04	1.51	1.42	1.46
O3	12.12	15.46	15.37	16.63	16.47	16.22	15.4
TEMP	14.62	11.77	11.59	13.18	12.08	11.86	12.05
PRES	0.82	0.89	0.91	1.66	1.0	0.93	0.97
DEWP	12.64	16.98	17.17	18.21	16.97	16.87	17.03
RAIN	194.52	194.38	194.44	194.44	194.4	194.42	194.38
WSPM	5.48	5.52	5.51	6.44	5.56	5.61	5.65

Table A.2: LightGBM - Beijing Multi-Site Air-Quality Data Dataset - Individual SMAPE

EXPERIMENT_NAME	anom_0	anom_1	anom_2	anom_3	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
UNRATE	19.12	22.86	28.6	30.44	25.83	21.24	19.51
PCEPI	2.09	2.82	1.71	3.04	3.0	1.98	2.91
PCEPILFE	2.68	4.72	7.6	2.55	4.27	5.34	3.51
RPI	2.74	2.9	2.14	3.86	2.25	2.64	2.23
PMSAVE	22.21	22.42	21.72	20.75	25.64	24.59	22.82
ALTSALES	25.76	24.21	25.69	27.6	25.54	22.9	28.63
INDPRO	13.34	12.5	17.4	13.3	14.46	12.91	16.75
TCU	10.81	19.36	13.24	8.46	13.21	9.35	10.29
PERMIT	13.07	11.5	9.02	14.99	16.04	10.84	9.22
HOUST	8.59	9.97	12.97	9.76	8.13	8.77	7.71
CIVPART	48.79	46.35	47.7	46.39	39.67	39.44	39.89
CPIAUCSL	2.98	2.63	2.53	2.37	2.3	1.75	2.4
CPILFESL	4.27	4.21	3.91	3.42	3.49	4.4	6.25
AtmCO2	3.07	3.44	4.08	3.84	4.67	3.12	3.92
goldMP	8.38	14.27	8.9	12.29	13.49	11.08	7.31
silverMP	13.76	9.15	12.75	12.34	9.9	9.01	10.27
globalTemp	8.59	7.97	8.18	8.96	9.52	8.15	8.15

Table A.3: LSTM - Primary Economic Dataset - Individual SMAPE

EXPERIMENT_NAME	anom_0	anom_1	anom_2	anom_3	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
PM2.5	27.33	29.05	27.89	28.33	27.53	26.57	28.21
PM10	28.33	28.7	28.57	28.01	28.84	29.12	29.19
SO2	19.02	23.84	23.76	22.59	22.56	27.24	26.51
NO2	26.2	27.2	26.6	25.81	25.72	25.01	25.47
CO	29.86	30.86	29.93	31.34	29.84	30.5	28.17
O3	34.34	34.38	35.0	33.92	33.45	34.39	34.32
TEMP	4.18	4.19	4.27	4.24	4.24	4.31	4.27
PRES	1.88	1.87	1.85	1.86	1.87	1.99	1.91
DEWP	2.64	2.63	2.64	2.61	2.85	2.62	2.86
RAIN	11.82	18.89	21.47	32.86	13.39	12.47	22.81
WSPM	31.56	31.57	31.54	31.51	31.8	31.56	31.53

Table A.4: LSTM - Beijing Multi-Site Air-Quality Data Dataset - Individual SMAPE

EXPERIMENT_NAME	anom_0	anom_1	anom_2	anom_3	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
UNRATE	2.42	2.52	3.22	3.99	3.59	4.77	1.92
PCEPI	3.36	5.76	2.89	3.92	1.79	2.39	2.47
PCEPILFE	2.74	1.69	2.85	4.7	2.02	3.46	5.43
RPI	2.42	2.24	3.03	4.27	6.11	4.43	1.8
PMSAVE	7.03	2.89	3.24	4.48	1.27	3.37	3.75
ALTSALES	3.81	2.74	2.49	1.83	1.9	5.97	5.04
INDPRO	2.95	1.94	4.39	6.14	2.78	3.63	3.91
TCU	3.06	1.7	3.84	4.25	3.75	4.57	1.59
PERMIT	2.16	1.99	2.76	2.76	3.41	4.42	2.62
HOUST	4.46	4.09	3.05	4.22	2.74	4.68	4.4
CIVPART	2.96	5.63	2.12	1.87	1.68	2.71	1.84
CPIAUCSL	2.76	1.98	2.07	1.91	1.56	2.41	2.09
CPILFESL	2.53	3.17	1.76	3.5	4.31	3.56	2.07
AtmCO2	3.14	3.6	3.0	3.19	3.13	2.04	1.66
goldMP	3.93	3.62	3.57	4.82	1.64	3.74	2.39
silverMP	2.94	2.15	2.77	1.74	1.56	2.48	3.35
globalTemp	1.29	4.59	1.5	4.1	1.64	1.73	4.93

Table A.5: Multi Lag LSTM - Primary Economic Dataset - Individual SMAPE

EXPERIMENT_NAME	anom_0	anom_1	anom_2	anom_3	iso_trs_0.1	iso_trs_0.2	iso_trs_0.3
PM2.5	6.81	2.8	2.55	4.44	1.77	3.85	6.18
PM10	5.14	6.12	4.07	2.85	1.97	2.4	11.35
SO2	3.08	4.95	4.61	4.1	2.47	3.9	4.64
NO2	7.74	2.1	3.97	5.47	6.25	3.36	6.73
CO	6.14	4.07	4.17	2.53	4.97	2.45	4.7
O3	6.96	1.4	8.88	4.34	4.07	1.81	2.4
TEMP	2.08	1.94	4.29	1.93	2.93	2.68	3.29
PRES	1.52	5.14	2.32	2.14	10.38	3.09	9.4
DEWP	5.98	3.32	2.25	3.28	4.72	4.54	1.01
RAIN	1.96	10.18	3.29	2.89	2.37	2.6	7.3
WSPM	6.28	4.54	7.3	6.25	4.89	3.06	2.29

Table A.6: Multi Lag LSTM - Beijing Multi-Site Air-Quality Data Dataset - Individual SMAPE

Appendix B

Code Listings

B.1 Evaluation

Listing B.1: Main Evaluation

```
# %%
def smape(actual, predicted) -> float:

    # Convert actual and predicted to numpy
    # array data type if not already
    if not all([isinstance(actual, np.ndarray),
                isinstance(predicted, np.ndarray)]):
        actual, predicted = np.array(actual), np.array(predicted)

    return round(
        np.mean(
            np.abs(predicted - actual) /
            ((np.abs(predicted) + np.abs(actual))/2)
        )*100, 2
    )

# %%
def evaluation_main(evaluation_dict, predicted_attribute, y_test, y_pred_val):
  evaluation_dict[predicted_attribute] = {}
  evaluation_dict[predicted_attribute]['MAE'] = mean_absolute_error(y_test,
    y_pred_val)
  evaluation_dict[predicted_attribute]['MSE'] = mean_squared_error(y_test,
    y_pred_val)
  evaluation_dict[predicted_attribute]['MAPE'] = mean_absolute_percentage_error(
    y_test, y_pred_val)
  evaluation_dict[predicted_attribute]['SMAPE'] = smape(y_pred_val, y_test)
  return evaluation_dict

# %%
def avg_evals(eval_type, evaluation_dict):
  eval_type_avg = 0
  for k, v in evaluation_dict.items():
    eval_type_avg = eval_type_avg + v[eval_type]
  return eval_type_avg / (len(evaluation_dict.items()))
```

Listing B.2: WSMAPE

```
# %%
import pandas as pd
from collections import defaultdict
from sklearn.preprocessing import MinMaxScaler, normalize, StandardScaler
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,\
    mean_squared_error

# %%
datasets = ['primary','beijing']
anoms = ['anom_0','anom_1','anom_2','anom_3','iso_trs_0.1','iso_trs_0.2','
    iso_trs_0.3']
```

```
11
12 #variables
13 DATASET = "beijing" # primary | beijing
14 ANOM_TYPE = "anom_0" # anom_0 | anom_1 | anom_2 | anom_3 | iso_trs_0.1 | iso_trs_0
       .2 | iso_trs_0.3
15 MODEL = "multilagLSTM" # LightGBM | LSTM | multilagLSTM
16
17 datasets = {"primary":"primary_",
18                 "beijing":"Beijing_data/beijing_"
19                 }
20
21 anom_dict = {"anom_0":"dataset_anom_0.csv",}
22
23 dataset_path = "/content/drive/MyDrive/Research/code/FINAL_DATASET/"
24
25 file_path = dataset_path + datasets[DATASET] + anom_dict[ANOM_TYPE]
26
27 SMAPE_path = '/content/drive/MyDrive/Research/code/RESULTS/'+MODEL+'/'+MODEL+'_'+
       DATASET+'_SMAPE.csv'
28
29 # %%
30 smape_df = pd.read_csv(SMAPE_path)
31
32 # %%
33 df = pd.read_csv(file_path)
34
35 # %%
36 df = df.iloc[:,1:]
37
38 df_normalized = normalize(df)
39 df_processed_numeric = pd.DataFrame(df_normalized,columns=df.columns)
40
41 # %%
42 variance_dict = defaultdict()
43 co = list(df_processed_numeric.columns)
44 # co.remove('date')
45 var_sum = 0
46 for i in co:
47     variance_dict[i] = df_processed_numeric[i].var(axis=0)
48     var_sum = var_sum + df_processed_numeric[i].var(axis=0)
49
50 # %%
51 li = []
52 for j in range(0,7):
53   WSMAPE = 0
54   exp_name = smape_df.iloc[j]["EXPERIMENT_NAME"]
55   for i in co:
56     WSMAPE = WSMAPE + (variance_dict[i]/var_sum)*smape_df.iloc[j][i]
57   li.append(round(WSMAPE,5))
58   print(exp_name, round(WSMAPE,5))
```

B.2 Anomaly Detection Methods

B.2.1 Seasonal and Trend decomposition using Loess

Listing B.3: STL based anomaly detecting

```
1 # %%
2 %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/
      imports_installs_prereq.ipynb
3 %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/preprocessing.
      ipynb
4 %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/grapher.ipynb
5 %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/evaluations.ipynb
6
7 installAll()
8
9 # %%
```

```
from sklearn.model_selection import TimeSeriesSplit
from sktime.performance_metrics.forecasting import MeanAbsoluteScaledError

from sklearn.preprocessing import MinMaxScaler, normalize, StandardScaler
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,\
  mean_squared_error

from collections import defaultdict

from IPython.display import display

import numpy as np
import pandas as pd
pd.set_option('display.max_rows', 15)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

import matplotlib.pyplot as plt
from datetime import datetime
from datetime import timedelta
from pandas.plotting import register_matplotlib_converters
from mpl_toolkits.mplot3d import Axes3D

from statsmodels.tsa.stattools import acf, pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
register_matplotlib_converters()
from time import time
import seaborn as sns
sns.set(style="whitegrid")

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.covariance import EllipticEnvelope

import warnings
warnings.filterwarnings('ignore')

from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.dates as mdates

RANDOM_SEED = np.random.seed(0)

# %%
def std_from_resid(residVals):
  std_arr = np.array(residVals)
  nan_indices = np.where(np.isnan(std_arr))
  std_arr = np.delete(std_arr, nan_indices)
  return np.std(std_arr)

# %%
def anomaly_detection_stl_decomp(final_dataset, attribute, anomaly_dict,
    anomaly_val, non_anomaly_val, augment_anomalies_flag = False,  decompose_model
    ='additive'):

  decompose_series = final_dataset[[attribute]]

  result = seasonal_decompose(decompose_series,model=decompose_model)

  THRESHOLD = std_from_resid(result.resid.values)*2

  x = result.resid.index
  y = result.resid.values

  datesX = []
  for i in range(len(x)):
    if y[i] > THRESHOLD or y[i] < -THRESHOLD:
      if augment_anomalies_flag:
        augmented_anomaly_val = int((abs(y[i] - std_from_resid(result.resid.values
    ))/std_from_resid(result.resid.values))*anomaly_val)
        # augmented_anomaly_val = int((abs(y[i] - std_from_resid(result.resid.
    values)))*anomaly_val)
      else:
```

```
          augmented_anomaly_val = anomaly_val
        datesX.append(augmented_anomaly_val)
      else:
        datesX.append(non_anomaly_val)
    anomaly_dict[attribute] = list(datesX)

# %%
def multivariate_aligned_anomalies_sum(anomaly_val, non_anomaly_val,
    augment_anomalies_flag = False):
    final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/
      final_dataset.csv',parse_dates=[0], index_col=0)
    anomaly_dict = defaultdict()
    for i in final_dataset.columns:
      anomaly_detection_stl_decomp(final_dataset, i, anomaly_dict,anomaly_val,
      non_anomaly_val, augment_anomalies_flag, decompose_model='additive')

    anomaly_df = pd.DataFrame.from_dict(anomaly_dict)
    anomaly_df_transposed = anomaly_df.T
    alignments = anomaly_df_transposed.sum()
    return np.array(list(alignments))

# %%
def multivariate_aligned_anomalies_inidividual(anomaly_val, non_anomaly_val,
    augment_anomalies_flag = False):
    final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/
      final_dataset.csv',parse_dates=[0], index_col=0)
    anomaly_dict = defaultdict()
    for i in final_dataset.columns:
      anomaly_detection_stl_decomp(final_dataset, i, anomaly_dict,anomaly_val,
      non_anomaly_val, augment_anomalies_flag, decompose_model='additive')

    anomaly_df = pd.DataFrame.from_dict(anomaly_dict)
    return anomaly_df

# %%
def get_multivariate_aligned_anomalies_inidividual():
    return multivariate_aligned_anomalies_inidividual(anomaly_val, non_anomaly_val,
      augment_anomalies_flag)

def get_multivariate_aligned_anomalies_sum():
    return multivariate_aligned_anomalies_sum(anomaly_val, non_anomaly_val,
      augment_anomalies_flag)


# %%
def dataset_augmentaiton_with_anomalies(final_dataset, anomaly_val,
    non_anomaly_val, anomaly_integration_flag = 1, sum_anomaly_flag = 1,
    both_anomalies_and_anomaly_sum = 0, augment_anomalies_flag = True):
    if anomaly_integration_flag:
      if both_anomalies_and_anomaly_sum:
        anomalies_inidividual =  get_multivariate_aligned_anomalies_inidividual()
        for i in final_dataset.columns[1:]:
          # print(anomalies_inidividual[i])
          column_name = i + '_anomalies'
          # print(column_name)
          final_dataset[column_name] = anomalies_inidividual[i]

        events = get_multivariate_aligned_anomalies_sum()
        final_dataset['_events'] = events

      else:
        if sum_anomaly_flag:
          events = get_multivariate_aligned_anomalies_sum()
          final_dataset['_events'] = events
        else:
          anomalies_inidividual =  get_multivariate_aligned_anomalies_inidividual()
          for i in final_dataset.columns[1:]:
            # print(anomalies_inidividual[i])
            column_name = i + '_anomalies'
            # print(column_name)
            final_dataset[column_name] = anomalies_inidividual[i]


```

```
# %%
final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/final_dataset.csv',parse_dates=[0], index_col=0)

# %%
RAIN_TEST_RATIO = 0.7
anomaly_val = 5
non_anomaly_val = -5

final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/final_dataset.csv')

anomaly_integration_flag = 0
sum_anomaly_flag = 1
both_anomalies_and_anomaly_sum = 0
augment_anomalies_flag = False


dataset_augmentaiton_with_anomalies(final_dataset, anomaly_val, non_anomaly_val, anomaly_integration_flag, sum_anomaly_flag, both_anomalies_and_anomaly_sum, augment_anomalies_flag)

# %%
# there are 4 definitions for anomaly score with the name format anom_x , x = {1,2,3}

final_dataset.to_csv("/content/drive/MyDrive/Research/code/FINAL_DATASET/primary_dataset_anom_0.csv", index=False)
```

B.2.2 Isolation Forest

Listing B.4: Isolation Foreset based anomaly detecting

```
# %%
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.covariance import EllipticEnvelope
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from collections import defaultdict


# %%
anomaly_dict = defaultdict()
anomaly_val = 5

outliers_fraction = float(.3)

final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/final_dataset.csv',parse_dates=[0], index_col=0)

dataset = final_dataset[[variable]]

for i in final_dataset.columns:
  dataset = final_dataset[[i]]

  scaler = StandardScaler()
  np_scaled = scaler.fit_transform(dataset.values.reshape(-1, 1))
  data = pd.DataFrame(np_scaled)

  model =  IsolationForest(contamination=outliers_fraction)
  model.fit(data)

  anomaly_dict[i] = model.predict(data)

# %%
anomaly_df = pd.DataFrame.from_dict(anomaly_dict)

```

```
# %%
anomaly_df = anomaly_df * -5

# %%
anomaly_df['UNRATE'].value_counts()

# %%
anomaly_df_transposed = anomaly_df.T
alignments = anomaly_df_transposed.sum()
events = np.array(list(alignments))
final_dataset['_events_isolation_trees'] = events

# %%
final_dataset.to_csv("/content/drive/MyDrive/Research/code/FINAL_DATASET/
    primary_dataset_isolation_trees_0.3.csv", index=True)
```

B.3 Models

B.3.1 LightGBM

Listing B.5: Automater for LightGBM

```
# %%
from google.colab import drive
drive.mount('/content/drive')

# %%
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/
    imports_installs_prereq.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/preprocessing.
    ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/grapher.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/evaluations.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/
    anomaly_dectection_stl_decomposition.ipynb

installAll()

# %%
import numpy as np
import pandas as pd
import wandb
import matplotlib.pyplot as plt
from sklearn.model_selection import TimeSeriesSplit
from sktime.performance_metrics.forecasting import MeanAbsoluteScaledError

from sklearn.preprocessing import MinMaxScaler, normalize, StandardScaler
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,\
  mean_squared_error

from collections import defaultdict
import os

# %%
datasets = ['primary','beijing']
anoms = ['anom_0','anom_1','anom_2','anom_3','iso_trs_0.1','iso_trs_0.2','
    iso_trs_0.3']

#variables
TRAIN_TEST_RATIO = 0.7
# DATASET = "primary" # primary | beijing
# ANOM_TYPE = "anom_0" # anom_0 | anom_1 | anom_2 | anom_3 | iso_trs_0.1 |
    iso_trs_0.2 | iso_trs_0.3
MODEL = "LightGBM"

datasets = {"primary":"primary_",
                "beijing":"Beijing_data/beijing_"
                }
```

```
42
43 anom_dict = {"anom_0":"dataset_anom_0.csv",
44                 "anom_1":"dataset_anom_1.csv",
45                 "anom_2":"dataset_anom_2.csv",
46                 "anom_3":"dataset_anom_3.csv",
47                 "iso_trs_0.1":"dataset_isolation_trees_0.1.csv",
48                 "iso_trs_0.2":"dataset_isolation_trees_0.2.csv",
49                 "iso_trs_0.3":"dataset_isolation_trees_0.3.csv",
50                 }
51
52 dataset_path = "/content/drive/MyDrive/Research/code/FINAL_DATASET/"
53
54 for DATASET in datasets:
55   for ANOM_TYPE in anoms:
56     EXPERIMENT_NAME = MODEL + '_' + DATASET + '_' + ANOM_TYPE
57
58     file_path = dataset_path + datasets[DATASET] + anom_dict[ANOM_TYPE]
59
60     final_dataset = pd.read_csv(file_path)
61
62     final_dataset.drop(columns=["RAIN"], inplace=True)
63
64     %run /content/drive/MyDrive/Research/code/FINAL_DATASET/lightGBM.ipynb
```

Listing B.6: LightGBM

```
1  # %%
2  %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/
       imports_installs_prereq.ipynb
3  %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/preprocessing.
       ipynb
4  %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/grapher.ipynb
5  %run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/evaluations.ipynb
6
7  installAll()
8
9  # %%
10 from sklearn.model_selection import TimeSeriesSplit
11 from sktime.performance_metrics.forecasting import MeanAbsoluteScaledError
12
13 from sklearn.preprocessing import MinMaxScaler, normalize, StandardScaler
14 from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,\
15   mean_squared_error
16
17 from collections import defaultdict
18
19 from IPython.display import display
20
21 import numpy as np
22 import pandas as pd
23 pd.set_option('display.max_rows', 15)
24 pd.set_option('display.max_columns', 500)
25 pd.set_option('display.width', 1000)
26
27 import matplotlib.pyplot as plt
28 from datetime import datetime
29 from datetime import timedelta
30 from pandas.plotting import register_matplotlib_converters
31 from mpl_toolkits.mplot3d import Axes3D
32
33 from statsmodels.tsa.stattools import acf, pacf
34 from statsmodels.tsa.statespace.sarimax import SARIMAX
35 register_matplotlib_converters()
36 from time import time
37 import seaborn as sns
38 sns.set(style="whitegrid")
39
40 from sklearn.preprocessing import StandardScaler
41 from sklearn.decomposition import PCA
42 from sklearn.cluster import KMeans
43 from sklearn.covariance import EllipticEnvelope
44
45 import warnings
```

```
warnings.filterwarnings('ignore')

from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.dates as mdates

RANDOM_SEED = np.random.seed(0)

# %%
def std_from_resid(residVals):
  std_arr = np.array(residVals)
  nan_indices = np.where(np.isnan(std_arr))
  std_arr = np.delete(std_arr, nan_indices)
  return np.std(std_arr)

# %%
def anomaly_detection_stl_decomp(final_dataset, attribute, anomaly_dict,
    anomaly_val, non_anomaly_val, augment_anomalies_flag = False,  decompose_model
    ='additive'):

  decompose_series = final_dataset[[attribute]]

  result = seasonal_decompose(decompose_series,model=decompose_model)

  THRESHOLD = std_from_resid(result.resid.values)*2

  x = result.resid.index
  y = result.resid.values

  datesX = []
  for i in range(len(x)):
    if y[i] > THRESHOLD or y[i] < -THRESHOLD:
      if augment_anomalies_flag:
        augmented_anomaly_val = int((abs(y[i] - std_from_resid(result.resid.values
    ))/std_from_resid(result.resid.values))*anomaly_val)
        # augmented_anomaly_val = int((abs(y[i] - std_from_resid(result.resid.
    values)))*anomaly_val)
      else:
        augmented_anomaly_val = anomaly_val
      datesX.append(augmented_anomaly_val)
    else:
      datesX.append(non_anomaly_val)
  anomaly_dict[attribute] = list(datesX)

# %%
def multivariate_aligned_anomalies_sum(anomaly_val, non_anomaly_val,
    augment_anomalies_flag = False):
  final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/
    final_dataset.csv',parse_dates=[0], index_col=0)
  anomaly_dict = defaultdict()
  for i in final_dataset.columns:
    anomaly_detection_stl_decomp(final_dataset, i, anomaly_dict,anomaly_val,
    non_anomaly_val, augment_anomalies_flag, decompose_model='additive')

  anomaly_df = pd.DataFrame.from_dict(anomaly_dict)
  anomaly_df_transposed = anomaly_df.T
  alignments = anomaly_df_transposed.sum()
  return np.array(list(alignments))

# %%
def multivariate_aligned_anomalies_inidividual(anomaly_val, non_anomaly_val,
    augment_anomalies_flag = False):
  final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/
    final_dataset.csv',parse_dates=[0], index_col=0)
  anomaly_dict = defaultdict()
  for i in final_dataset.columns:
    anomaly_detection_stl_decomp(final_dataset, i, anomaly_dict,anomaly_val,
    non_anomaly_val, augment_anomalies_flag, decompose_model='additive')

  anomaly_df = pd.DataFrame.from_dict(anomaly_dict)
  return anomaly_df

# %%
def get_multivariate_aligned_anomalies_inidividual():
```

```
    return multivariate_aligned_anomalies_inidividual(anomaly_val, non_anomaly_val,
      augment_anomalies_flag)

def get_multivariate_aligned_anomalies_sum():
    return multivariate_aligned_anomalies_sum(anomaly_val, non_anomaly_val,
      augment_anomalies_flag)


# %%
def dataset_augmentaiton_with_anomalies(final_dataset, anomaly_val,
    non_anomaly_val, anomaly_integration_flag = 1, sum_anomaly_flag = 1,
    both_anomalies_and_anomaly_sum = 0, augment_anomalies_flag = True):
    if anomaly_integration_flag:
        if both_anomalies_and_anomaly_sum:
            anomalies_inidividual =  get_multivariate_aligned_anomalies_inidividual()
            for i in final_dataset.columns[1:]:
                # print(anomalies_inidividual[i])
                column_name = i + '_anomalies'
                # print(column_name)
                final_dataset[column_name] = anomalies_inidividual[i]

            events = get_multivariate_aligned_anomalies_sum()
            final_dataset['_events'] = events

        else:
            if sum_anomaly_flag:
                events = get_multivariate_aligned_anomalies_sum()
                final_dataset['_events'] = events
            else:
                anomalies_inidividual =  get_multivariate_aligned_anomalies_inidividual()
                for i in final_dataset.columns[1:]:
                    # print(anomalies_inidividual[i])
                    column_name = i + '_anomalies'
                    # print(column_name)
                    final_dataset[column_name] = anomalies_inidividual[i]


# %%
final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/
    final_dataset.csv',parse_dates=[0], index_col=0)

# %%
RAIN_TEST_RATIO = 0.7
anomaly_val = 5
non_anomaly_val = -5

final_dataset = pd.read_csv('/content/drive/MyDrive/Research/code/FINAL_DATASET/
    final_dataset.csv')

anomaly_integration_flag = 0
sum_anomaly_flag = 1
both_anomalies_and_anomaly_sum = 0
augment_anomalies_flag = False


dataset_augmentaiton_with_anomalies(final_dataset, anomaly_val, non_anomaly_val,
    anomaly_integration_flag, sum_anomaly_flag, both_anomalies_and_anomaly_sum,
    augment_anomalies_flag)

# %%
# there are 4 definitions for anomaly score with the name format anom_x , x =
    {1,2,3}

final_dataset.to_csv("/content/drive/MyDrive/Research/code/FINAL_DATASET/
    primary_dataset_anom_0.csv", index=False)
```

B.3.2 LSTM

Listing B.7: Automater for LSTM

```
# %%
from pandas import read_csv

# %%
from google.colab import drive
drive.mount('/content/drive')

# %%
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/
    imports_installs_prereq.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/preprocessing.
    ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/grapher.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/evaluations.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/
    anomaly_dectection_stl_decomposition.ipynb

installAll()

# %%
datasets = ['primary','beijing']
anoms = ['anom_0','anom_1','anom_2','anom_3','iso_trs_0.1','iso_trs_0.2','
    iso_trs_0.3']

#variables
TRAIN_TEST_RATIO = 0.7
# DATASET = "primary" # primary | beijing
# ANOM_TYPE = "anom_0" # anom_0 | anom_1 | anom_2 | anom_3 | iso_trs_0.1 |
    iso_trs_0.2 | iso_trs_0.3
MODEL = "LSTM"

datasets = {"primary":"primary_",
                "beijing":"Beijing_data/beijing_"
                }


anom_dict = {"anom_0":"dataset_anom_0.csv",
                "anom_1":"dataset_anom_1.csv",
                "anom_2":"dataset_anom_2.csv",
                "anom_3":"dataset_anom_3.csv",
                "iso_trs_0.1":"dataset_isolation_trees_0.1.csv",
                "iso_trs_0.2":"dataset_isolation_trees_0.2.csv",
                "iso_trs_0.3":"dataset_isolation_trees_0.3.csv",
                }

dataset_path = "/content/drive/MyDrive/Research/code/FINAL_DATASET/"

for DATASET in datasets:
  for ANOM_TYPE in anoms:
    EXPERIMENT_NAME = MODEL + '_' + DATASET + '_' + ANOM_TYPE

    file_path = dataset_path + datasets[DATASET] + anom_dict[ANOM_TYPE]

    dataset = read_csv(file_path, header=0, index_col=0)

    dataset.drop('RAIN', axis=1, inplace=True)

    %run /content/drive/MyDrive/Research/code/LSTM/multiVariableLSTM.ipynb
```

Listing B.8: LSTM

```
# %%
# prepare data for lstm
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_csv
from pandas import DataFrame
```

```
from pandas import concat
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from collections import defaultdict


def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
        n_vars = 1 if type(data) is list else data.shape[1]
        df = DataFrame(data)
        cols, names = list(), list()

        for i in range(n_in, 0, -1):
                cols.append(df.shift(i))
                names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
        #
        for i in range(0, n_out):
                cols.append(df.shift(-i))
                if i == 0:
                        names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
                else:
                        names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars
    )]

        agg = concat(cols, axis=1)
        agg.columns = names

        if dropnan:
                agg.dropna(inplace=True)
        return agg

# %%
evaluation_dict = defaultdict()




for predicted_variable in dataset.columns:

  if '_' in predicted_variable:
    continue


  values = dataset.values


  values = values.astype('float32')

  scaler = MinMaxScaler(feature_range=(0, 1))
  scaled = scaler.fit_transform(values)

  reframed = series_to_supervised(scaled, 1, 1)


  columns_to_drop = [*range(dataset.shape[1], dataset.shape[1]*2, 1)]



  predicted_variable_location = dataset.shape[1] + dataset.columns.get_loc(
    predicted_variable)
  columns_to_drop.remove(predicted_variable_location)

  reframed.drop(reframed.columns[columns_to_drop], axis=1, inplace=True)
  print(reframed.head())

  values = reframed.values

  n_train_mask = int(dataset.shape[0] * TRAIN_TEST_RATIO)


```

```
    train = values[:n_train_mask, :]
    test = values[n_train_mask:, :]

    train_X, train_y = train[:, :-1], train[:, -1]
    test_X, test_y = test[:, :-1], test[:, -1]

    train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
    test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
    print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)


    model = Sequential()
    model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
    model.add(Dense(1))
    model.compile(loss='mae', optimizer='adam')

    history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data
      =(test_X, test_y), verbose=2, shuffle=False)


    yhat = model.predict(test_X)
    test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))

    inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
    inv_yhat = scaler.inverse_transform(inv_yhat)
    inv_yhat = inv_yhat[:,0]

    test_y = test_y.reshape((len(test_y), 1))
    inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
    inv_y = scaler.inverse_transform(inv_y)
    inv_y = inv_y[:,0]

    evaluation_main(evaluation_dict, predicted_variable, inv_y, inv_yhat)

# %%
avg_MAE = avg_evals('MAE', evaluation_dict)

# %%
avg_MSE = avg_evals('MSE', evaluation_dict)

# %%
avg_MAPE = avg_evals('MAPE', evaluation_dict)

# %%
avg_SMAPE = avg_evals('SMAPE', evaluation_dict)

# %%
def eval_dict_for_specific_evals(evaluation_dict, eval_metric):
  eval_dict = defaultdict()
  eval_dict['EXPERIMENT_NAME'] = EXPERIMENT_NAME + "-" + eval_metric
  for k,v in evaluation_dict.items():
    # key = k + '_' + eval_metric
    value = v[eval_metric]
    eval_dict[k] = value
  return eval_dict

# %%
import os

# %%
def put_evals_to_csv(evaluation_dict_for_metric, column_names, csv_path):
  if os.path.exists(csv_path):
    eval_csv = pd.read_csv(csv_path)
    new_data_df = pd.DataFrame.from_dict(evaluation_dict_for_metric, orient='index
    ').T
    csvdf = pd.concat([eval_csv, new_data_df], ignore_index=True)
    csvdf.to_csv(csv_path, index=False)
  else:
    df = pd.DataFrame.from_dict(evaluation_dict_for_metric, orient='index').T
    df.to_csv(csv_path, index=False)

# %%
EVALS = ['MAE','MSE','MAPE','SMAPE']
```

```
dataset_columns = list(dataset.columns)

for EVAL_CONSIDERED in EVALS:
  eval_dict_for_metric = eval_dict_for_specific_evals(evaluation_dict,
    EVAL_CONSIDERED)
  csv_path = '/content/drive/MyDrive/Research/code/RESULTS/' + MODEL + '/' + MODEL
     + '_' + DATASET + '_'+ EVAL_CONSIDERED +'.csv'
  put_evals_to_csv(eval_dict_for_metric, dataset_columns, csv_path)

# %%
#average valus of evals

avg_evals_dict = {
    "EXPERIMENT_NAME": EXPERIMENT_NAME + "-aggregated",
    "avg_MAE":avg_MAE,
    "avg_MSE":avg_MSE,
    "avg_RMSE":avg_RMSE,
    "avg_MAPE":avg_MAPE,
    "avg_SMAPE":avg_SMAPE
}

average_eval_column_names = list(avg_evals_dict.keys())
csv_path = '/content/drive/MyDrive/Research/code/RESULTS/' + MODEL + '/' + MODEL +
     '_' + DATASET +'_average.csv'

put_evals_to_csv(avg_evals_dict, average_eval_column_names, csv_path)
```

B.3.3 Multi Lag LSTM

Listing B.9: Automater for Multi Lag LSTM

```
# %%
from pandas import read_csv

# %%
from google.colab import drive
drive.mount('/content/drive')

# %%
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/
    imports_installs_prereq.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/preprocessing.
    ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/grapher.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/includes/evaluations.ipynb
%run /content/drive/MyDrive/Research/code/FINAL_DATASET/
    anomaly_dectection_stl_decomposition.ipynb

installAll()

# %%
datasets = ['primary','beijing']
anoms = ['anom_0','anom_1','anom_2','anom_3','iso_trs_0.1','iso_trs_0.2','
    iso_trs_0.3']

#variables
TRAIN_TEST_RATIO = 0.7
# DATASET = "primary" # primary | beijing
# ANOM_TYPE = "anom_0" # anom_0 | anom_1 | anom_2 | anom_3 | iso_trs_0.1 |
    iso_trs_0.2 | iso_trs_0.3
MODEL = "multilagLSTM"

datasets = {"primary":"primary_",
                "beijing":"Beijing_data/beijing_"
                }

anom_dict = {"anom_0":"dataset_anom_0.csv",
                "anom_1":"dataset_anom_1.csv",
                "anom_2":"dataset_anom_2.csv",
                "anom_3":"dataset_anom_3.csv",
```

```
                "iso_trs_0.1":"dataset_isolation_trees_0.1.csv",
                "iso_trs_0.2":"dataset_isolation_trees_0.2.csv",
                "iso_trs_0.3":"dataset_isolation_trees_0.3.csv",
                }

dataset_path = "/content/drive/MyDrive/Research/code/FINAL_DATASET/"

for DATASET in datasets:
  for ANOM_TYPE in anoms:
    EXPERIMENT_NAME = MODEL + '_' + DATASET + '_' + ANOM_TYPE

    file_path = dataset_path + datasets[DATASET] + anom_dict[ANOM_TYPE]

    dataset = read_csv(file_path, header=0, index_col=0)
    dataset.drop(columns=["RAIN"], inplace=True)

    %run /content/drive/MyDrive/Research/code/LSTM/multiVariableMultilagLSTM.ipynb
```

Listing B.10: Multi Lag LSTM

```
# %%
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM


def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
 n_vars = 1 if type(data) is list else data.shape[1]
 df = DataFrame(data)
 cols, names = list(), list()

 for i in range(n_in, 0, -1):
  cols.append(df.shift(i))
  names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

 for i in range(0, n_out):
  cols.append(df.shift(-i))
  if i == 0:
    names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
  else:
    names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]

 agg = concat(cols, axis=1)
 agg.columns = names

 if dropnan:
  agg.dropna(inplace=True)
 return agg

# %%
evaluation_dict = defaultdict()


for predicted_variable in dataset.columns:

  if '_' in predicted_variable:
    continue

  values = dataset.values


  values = values.astype('float32')

  scaler = MinMaxScaler(feature_range=(0, 1))
```

```
    scaled = scaler.fit_transform(values)




    n_months = 3
    n_features = dataset.shape[1]

    reframed = series_to_supervised(scaled, n_months, 1)



    columns_to_drop = [*range(dataset.shape[1]*n_months, dataset.shape[1]*n_months +
       dataset.shape[1], 1)]
    predicted_variable_location = dataset.shape[1]*n_months + dataset.columns.
      get_loc(predicted_variable)
    columns_to_drop.remove(predicted_variable_location)


    reframed.drop(reframed.columns[columns_to_drop], axis=1, inplace=True)
    print(reframed.head())


    values = reframed.values

    TRAIN_TEST_RATIO = 0.7

    n_train_mask = int(dataset.shape[0] * TRAIN_TEST_RATIO)


    train = values[:n_train_mask, :]
    test = values[n_train_mask:, :]

    n_obs = n_months * n_features
    train_X, train_y = train[:, :n_obs], train[:, -n_features]
    test_X, test_y = test[:, :n_obs], test[:, -n_features]
    print(train_X.shape, len(train_X), train_y.shape)

    train_X = train_X.reshape((train_X.shape[0], n_months, n_features))
    test_X = test_X.reshape((test_X.shape[0], n_months, n_features))
    print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)


    model = Sequential()
    model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
    model.add(Dense(1))
    model.compile(loss='mae', optimizer='adam')

    history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data
      =(test_X, test_y), verbose=2, shuffle=False)



    yhat = model.predict(test_X)
    test_X = test_X.reshape((test_X.shape[0], n_months*n_features))

    inv_yhat = concatenate((yhat, test_X[:,  -(n_features-1):]), axis=1)
    inv_yhat = scaler.inverse_transform(inv_yhat)
    inv_yhat = inv_yhat[:,0]

    test_y = test_y.reshape((len(test_y), 1))
    inv_y = concatenate((test_y, test_X[:,  -(n_features-1):]), axis=1)
    inv_y = scaler.inverse_transform(inv_y)
    inv_y = inv_y[:,0]



    # evaluations
    evaluation_main(evaluation_dict, predicted_variable, inv_y, inv_yhat)

# %%
avg_MAE = avg_evals('MAE', evaluation_dict)

```

```
# %%
avg_MSE = avg_evals('MSE', evaluation_dict)

# %%
avg_MAPE = avg_evals('MAPE', evaluation_dict)

# %%
avg_SMAPE = avg_evals('SMAPE', evaluation_dict)

# %%
def eval_dict_for_specific_evals(evaluation_dict, eval_metric):
  eval_dict = defaultdict()
  eval_dict['EXPERIMENT_NAME'] = EXPERIMENT_NAME + "-" + eval_metric
  for k,v in evaluation_dict.items():
    # key = k + '_' + eval_metric
    value = v[eval_metric]
    eval_dict[k] = value
  return eval_dict

# %%
import os

# %%
def put_evals_to_csv(evaluation_dict_for_metric, column_names, csv_path):
  if os.path.exists(csv_path):
    eval_csv = pd.read_csv(csv_path)
    new_data_df = pd.DataFrame.from_dict(evaluation_dict_for_metric, orient='index
    ').T
    csvdf = pd.concat([eval_csv, new_data_df], ignore_index=True)
    csvdf.to_csv(csv_path, index=False)
  else:
    df = pd.DataFrame.from_dict(evaluation_dict_for_metric, orient='index').T
    df.to_csv(csv_path, index=False)

# %%
EVALS = ['MAE','MSE','MAPE','SMAPE']
dataset_columns = list(dataset.columns)

for EVAL_CONSIDERED in EVALS:
  eval_dict_for_metric = eval_dict_for_specific_evals(evaluation_dict,
    EVAL_CONSIDERED)
  csv_path = '/content/drive/MyDrive/Research/code/RESULTS/' + MODEL + '/' + MODEL
     + '_' + DATASET + '_'+ EVAL_CONSIDERED +'.csv'
  put_evals_to_csv(eval_dict_for_metric, dataset_columns, csv_path)

# %%
#average valus of evals

avg_evals_dict = {
    "EXPERIMENT_NAME": EXPERIMENT_NAME + "-aggregated",
    "avg_MAE":avg_MAE,
    "avg_MSE":avg_MSE,
    "avg_RMSE":avg_RMSE,
    "avg_MAPE":avg_MAPE,
    "avg_SMAPE":avg_SMAPE
}

average_eval_column_names = list(avg_evals_dict.keys())
csv_path = '/content/drive/MyDrive/Research/code/RESULTS/' + MODEL + '/' + MODEL +
     '_' + DATASET +'_average.csv'

put_evals_to_csv(avg_evals_dict, average_eval_column_names, csv_path)
```