Eötvös Loránd University

Faculty of Informatics

Department of Data Science and Engineering

Pseudo-time series data augmentation with non GANs strategy

Supervisor:
Nouar Ikrame
PhD Candidate

Author:
Caliskan Enes
Computer Science MSc

Budapest, 2025

Contents

Chapter 1

Introduction

In the era of big data, the ability to collect, analyze and utilize information has become essential to technological advancement and decision-making. Time series data, which represents data points collected or recorded at specific time intervals, is critical because it can be applied in various fields, including finance, healthcare and environmental science. Analyzing time series data allows for identifying trends, patterns, and anomalies, which aids predictive modeling and strategic planning [1].

Despite their value, acquiring high-quality, diverse and reliable time series data remains challenging due to data scarcity, privacy constraints and inconsistent sampling. Synthetic data generation has emerged as a promising solution to these challenges by replicating the statistical properties of real-world data while preserving privacy.

Pseudo-time series refers to artificially generated sequential data that mimics the statistical properties and patterns of structured sequential data [2]. These artificial sequences are designed to capture and reproduce the underlying distributional characteristics and relationships present in the source data. Synthetic data is not a replacement for real data [3]. Instead, it aims to complement real-world information by providing a practical means to overcome data limitations.

There are various ways to generate pseudo-time series from original data. While Generative Adversarial Networks(GANs) have demonstrated success in image generation, their application to time series remains limited. Time series data present unique challenges, such as temporal coherence and variable-length sequences. While GANs have struggled with these issues, especially without specialized architectures, even alternative approaches, such as Bayesian methods, require compromises, such as resampling to fixed lengths.

Most GANs, especially the traditional ones [4], are optimized for spatial data and require extensive training and tuning. They often fail to accurately capture multimodal distributions or temporal uncertainty, leading to synthetic data that lack structural fidelity [5] or usefulness in downstream tasks—moreover, their "black box" nature privacy risk [6].

This thesis proposes a Bayesian generative framework for time series synthesis, which directly models uncertainty, integrates prior knowledge, and produces interpretable posterior distributions [7]. This probabilistic approach offers a fast and robust alternative to GAN-based solutions, particularly in low-data regimes, by leveraging Bayesian statistical principles to generate synthetic sequences that capture essential characteristics of the original data.

Since Bayesian models are based on Bayes' theorem [8], this approach allows the integration of prior information into the model, improving the accuracy and reliability of the generated data. Bayesian models can be computationally more efficient than GANs, as they do not require the extensive training cycles and computational power typically associated with GAN-based approaches. This efficiency makes Bayesian methods more accessible and feasible for various applications.

The creation process involves preprocessing the existing pseudo-time series dataset, defining the model and variables, updating priors with observed data, and using Bayes' theorem to obtain the posterior distribution. We then used trace and posterior predictive methods to generate time series samples, postprocessed and evaluated through various metrics, analyzed by distinct classifiers.

Chapter 2 will give the theoretical background for this research and related works. Chapter 3 will provide a comprehensive description of the time series dataset used and the methods to preprocess and generate time series with Bayesian modeling, along with postprocessing techniques that aim to enhance the performance of these generated sequences. This chapter will also introduce evaluation metrics that will be put into use. Chapter 4 will describe the results and evaluation metrics used to assess the quality of the time series. Section 4.3 will further analyze the limitations of synthesized time series with various classifiers and offer a new perspective on its shortcomings. Finally, chapter 6 will explain the advantages and limitations of the Bayesian approach and conclude the research.

Chapter 2

Theoretical background

This chapter provides an overview of the key theoretical concepts and mathematical foundations that underpin time series analysis and synthesis techniques. The following sections introduce fundamental principles that are relevant to various approaches in the field.

- **Pseudo-time series:** Pseudo-time series are data sequences that exhibit structural or statistical dependencies across ordered elements but are not indexed by chronological time. Unlike true time series, where each observation is tied to a specific timestamp and temporal continuity is inherent, pseudo-time series are sequences where the ordering reflects a logical or spatial structure rather than actual time.

  This concept enables the application of traditional time series analysis techniques to non-temporal data. For example, sequences derived from shape contours [9] or gene expression profiles [10, 11, 12] can be treated as time series, even though their indices do not represent real time. The utility of pseudo-time series lies in their ability to model meaningful progression or patterns in domains where temporal ordering is absent but sequential structure existent and informative.

- **Bayes' theorem:** Bayesian modeling, which is based on Bayes' theorem [8], forms an important framework for many time series synthesis techniques. This theorem offers a systematic method for updating probabilities as new evidence becomes available [13]. Bayes' theorem can be expressed with the following formula:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

The components of this formula represent:

- $P(A|B)$: The posterior probability—the probability of event A occurring given that B is true.
- $P(B|A)$: The likelihood—the probability of event B occurring given that A is true.
- $P(A)$: The prior probability—the initial probability of event A before considering evidence B.
- $P(B)$: The evidence—the probability of observing evidence B under all possible hypotheses.

In the context of time series analysis, events A and B typically represent hypotheses and observed data, respectively. This formulation allows analysts to make predictions based on previously acquired data, which is essential for various probabilistic approaches to time series modeling [13]. The ability to update beliefs systematically as new evidence emerges makes Bayesian methods particularly valuable for sequential data analysis.

- **Bayesian Modeling for Time Series:** Bayesian modeling offers a probabilistic framework for sampling synthesized time series. By combining priors with observed evidence, Bayesian models generate posterior distribution over model parameters that have been specified. Once the posterior distribution over parameters is obtained, it can generate new data via the posterior predictive distribution, ensuring that the synthesized samples reflect the learned structure and uncertainty. This is particularly valuable when dealing with limited data [14].

- **Hierarchical Modeling:** Hierarchical Bayesian models set up parameters that work at multiple levels, letting values change across groups while keeping structured relationships between patterns. This works really well for datasets with nested structures where observations might share some features but still have their own distinct characteristics. These approaches can capture both the bigger group trends and the smaller variations within groups, which makes them great for analyzing complex data where you care about both the overall patterns and the local differences.

- **Gaussian Random Walk (GRW):** A Gaussian Random Walk is a stochastic process where each point represents a noisy increment from the previous one. GRWs are effective at capturing smooth, drifting behavior that is common in real-world time series. Their flexibility makes them suitable for modeling latent trends in data, particularly when gradual changes over time need to be represented without imposing rigid structural constraints.

Numerous studies have been conducted on generating pseudo-time series using different techniques, especially GANs. These attempts to better understand the structure and synthesis of the time series are worth mentioning and exploring. In addition to the generation process, we examine evaluation metrics and analysis methods that provide deeper insight into the quality and characteristics of time series generated through Bayesian modeling.

Sanchez-Castillo et al. [11] propose creating pseudo-time series data to be used in the Bayesian framework for the inference of gene regulatory networks. They compare the performance of the proposed method with previous approaches using synthetic data.

Forestier et al. [15] propose a framework for generating synthetic time series using weighted averaging under Dynamic Time Warping (DTW), aiming to augment sparse datasets. Their methods are evaluated on the University of California Riverside (UCR) Time Series Archive [16], where they demonstrate improved classification performance on 56 out of 85 datasets, no change on 7, and a decline on 22, highlighting the potential and limitations of synthetic augmentation in time series learning.

Salmi and Seixas [9] explore the pseudo-time series generation using GANs. Their approach creates synthetic time series data that faithfully replicates the original data's patterns, trends and features while making compelling arguments about evaluation metrics to compare original data to generated ones.

Iwana et al. [17] offer comprehensive taxonomy and empirical evaluation, valuable insights and practical guidelines for researchers and practitioners in the field of time series analysis.

Liu et al. [12] accurately reconstruct cell cycle pseudo-time series, provide valuable insights into cellular processes, and hold promise for further applications in developmental biology and disease research.

Frühwirth-Schnatter et al. [18] provides an adaptable method for identifying temporal dependencies in data. These models are beneficial in domains where a nuanced understanding of uncertainty is crucial, thanks to their probabilistic nature. Computational method advances keep making them more practical and efficient, opening doors to wider use and acceptance.

Kottas et al. [19] work significantly contributes to the field of time series analysis by introducing a robust and flexible framework that leverages from Bayesian non-parametric methods. The transformation approach and hierarchical modeling enhance the model's ability to capture complex temporal dependencies, setting a strong foundation for future advancements.

Yadav et al. [20] proposed a novel GAN-based architecture, Multivariate Time-Series TGAN (MTS-TGAN), specifically designed to address the shortcomings of existing GAN models in generating high-quality multivariate time-series data. MTS-TGAN extends TimeGAN by adding feature selection layer to remove noise and enhance feature representation, improving the model's ability to capture intricate relationships in multivariate datasets. The model was evaluated using qualitative and quantitative metrics such as Train Real Test Synthetic (TRTS) and Train Synthetic Test Real (TSTR), along with MAE and MSLE scores. Additionally, the work by Yadav et al. validates the need for a model-specific evaluation strategy, aligning with our objective to measure the effectiveness of Bayesian methods as a non GAN-based alternative.

Fouladgar et al. [21] investigate time series generation challenges due to the increasing need for models that can produce dependable time series. Their work aims to overcome these problems by introducing new sensitivity metrics specific to time series data. Four sensitivity metrics are developed in their work: Max Short-Term Sensitivity (MSS), Max Long-Term Sensitivity (MLS), Average Short-Term Sensitivity (ASS) and Average Long-Term Sensitivity (ALS). The emphasis placed on sensitivity metrics emphasizes how crucial it is to assess model stability in time-series contexts, as this can help guide strategies for creating pseudo-time series. Examining how Bayesian models compare against GANs in terms of producing consistent, comprehensible data augmentations can be helpful by combining a variety of sensitivity metrics. Including short- and long-term sensitivity assessments would allow the model to demonstrate robustness, contributing to an evaluation framework for pseudo-time series data.

Driemel et al. [22] developed distance measures specifically designed for comparing time series data, such as DTW and the Fréchet distance. These metrics are valuable for evaluating the quality of synthesized time series by measuring how closely the generated sequences match reference data. Because of its ability to adapt to local data changes and irregular sampling, the Fréchet distance, in particular, has become increasingly popular for clustering time series. They contributed significantly to this field by proposing the first $(1 + \varepsilon)$-approximation algorithms for k-center and k-median clustering under the Fréchet distance. To provide a solution that works well with irregularly sampled data, their algorithms identify cluster centers that minimize the maximum or sum of Fréchet distances to each time series in the data set. By utilizing the Fréchet distance as an evaluation metric, how well can the generated time series align with real data in terms of overall shape and class-specific patterns be measured.

Wang and Koniusz [23] address the limitations of DTW by proposing Time-Weighted Dynamic Time Warping (TDTW), a DTW variant that incorporates time weighting. TDTW applies larger weights to more recent data points, making it well-suited for applications that prioritize the latest observations, such as stock price analysis and climate trend forecasting. Li et al. [24] demonstrated that TDTW achieves superior clustering and classification accuracy on datasets where recent data is more relevant, outperforming conventional DTW and several other DTW variants. The work by Wang and Koniusz offers valuable insights into time-sensitive similarity measurement. By incorporating TDTW as an evaluation metric, we can assess whether synthetic series generated by Bayesian models preserve the general structure and the importance of recent data points. This metric could enhance the robustness of the evaluation framework, aligning with your objective to create a realistic pseudo-time series with temporal dependencies reflective of real-world data.

Chapter 3

Methods

In this chapter, the methods used to generate pseudo-time series from actual time series data are presented in detail. The principal methodology that is used for this paper is based on Bayes' theorem, and results are achieved by creating a Gaussian model in model's framework. Pym3 is a Python library for time series forecasting and analysis. It provides a range of tools and models for working with time series data, including model training and evaluation methods. Combining the power of Pym3 with Bayesian estimation will allow for fast time series generation.

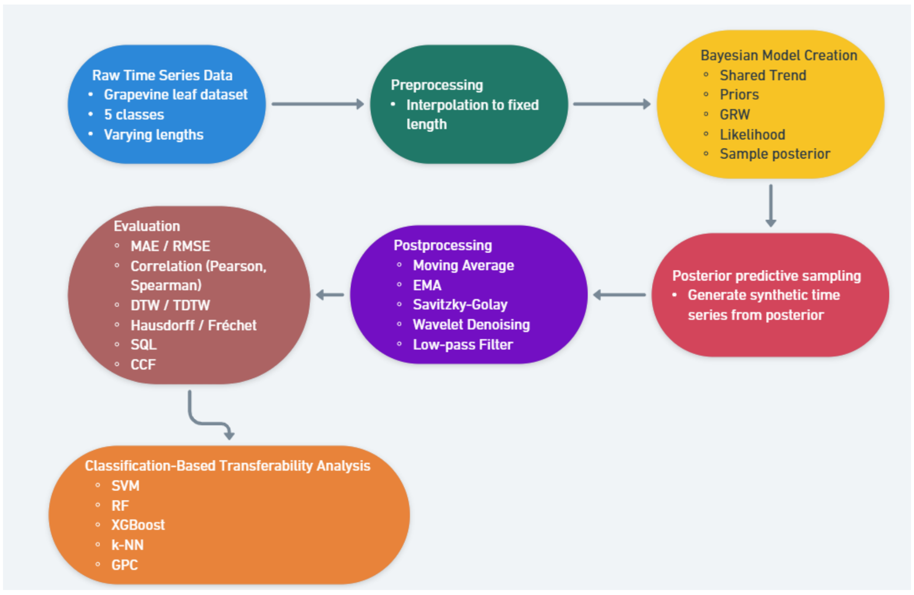


Figure 3.1: An overview of the Bayesian time series synthesis pipeline

To better illustrate the structure and flow of the methodology used in this study, Figure 3.1 provides a visual overview of the entire pipeline. The process begins with preprocessing raw grapevine time series data, followed by Bayesian modeling using PyMC3. Synthetic time series are then generated from the posterior predictive distribution. Several postprocessing techniques are applied to reduce noise and fluctuations. The final outputs are evaluated using a diverse set of metrics, and their transferability is assessed using multiple classification algorithms.

3.1 Time series dataset

The dataset for generating pseudo-time series is based on transforming grapevine leaves into data points, research that Seixas and Horváth have made [25]. Data points represent the distance from the stem to the contour of the leaf. Then, these points are taken to form a time series. The dataset contains five different classes; each class is named in Hungarian after the grapevine leaves produced from the tree: “Cabernet Franc," “Kékfrankos," “Sárgamuskotály," “Szürkebarát" and “Tramini."

In total, 377 time series samples exist within the five classes. Samples from different classes show different characteristics, as shown in Figure 3.2, and they are distributed as follows: Cabernet Franc contains 95 samples, Kékfrankos contains 73 samples, Sárgamuskotály contains 96 samples, Szürkebarát contains 89 samples and Tramini contains 24 samples.

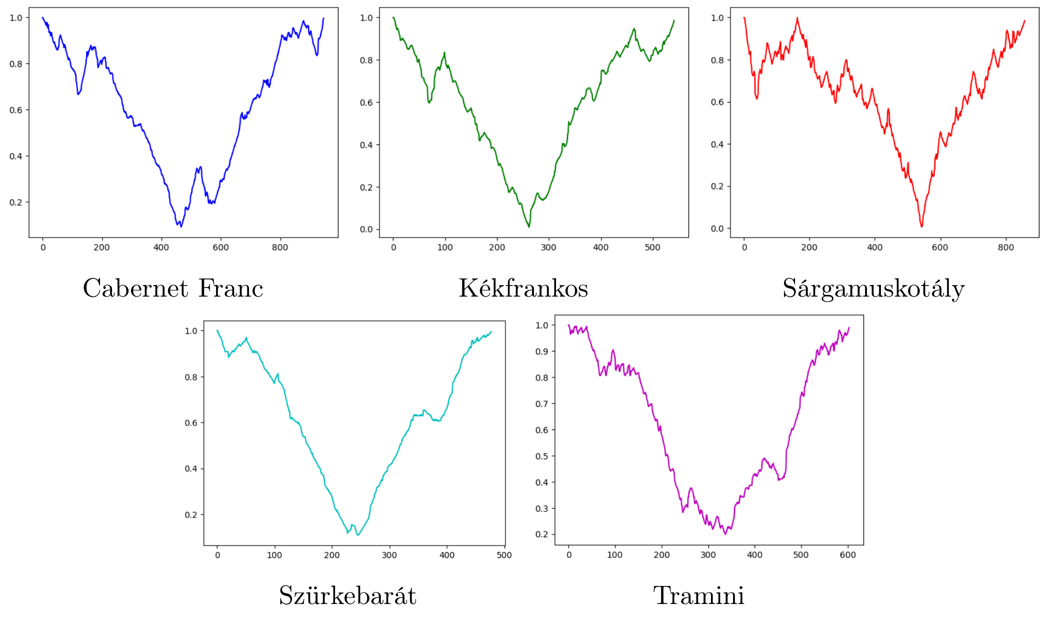


Figure 3.2: Pseudo-time series samples picked randomly from each class

To better understand the dataset used and the characteristics of the time series within the same class, two samples with the same length (time series with the same number of data points) from the Sárgamuskotály class have been used. These two samples were picked randomly, with the importance of being from the same class. Characteristics and patterns of time series within the same class were mentioned before, and this feature can be seen more clearly in Figure 3.3.

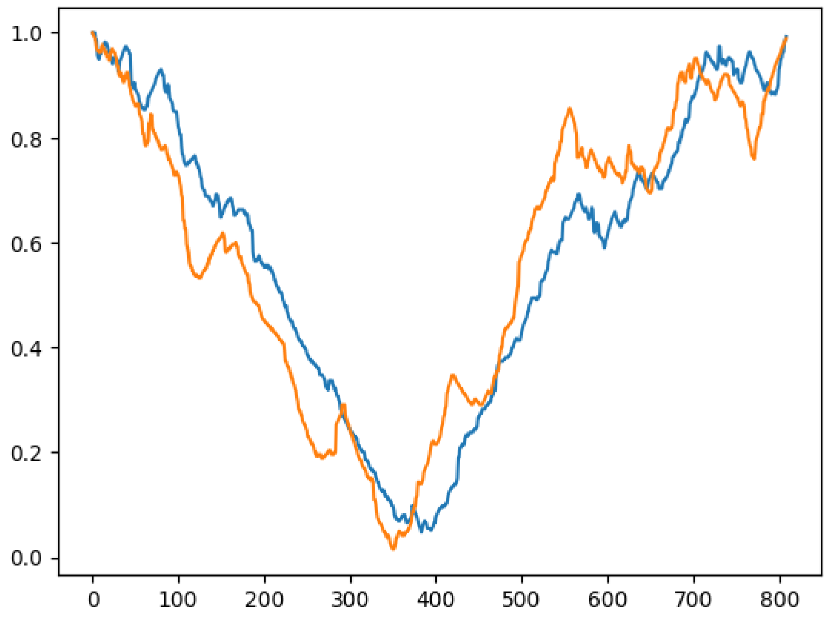


Figure 3.3: Two pseudo-time series samples from Sárgamuskotály class

3.2 Data preprocessing

To prepare the dataset for Bayesian model creation, it was necessary to normalize the time series to ensure uniformity in length across all samples. This step was crucial because the original dataset consisted of time series with varying numbers of data points. To achieve this, a resampling method using interpolation was applied. First, the shortest time series in the dataset was identified, and its length was designated as the target length. Each time series was then interpolated to match this length by linearly spacing both its original indices and the new target indices, allowing for smooth scaling of the data points. This interpolation process effectively resampled the data to the desired uniform length, creating a standardized dataset suitable for model training.

After normalization, two arrays were created to structure the data for model input: an index array and an observed values array. The index array replicated the uniform indices for all samples, ensuring that each series shared the same structural framework. In contrast, the observed values array contained the resampled series values. This preprocessing step provided a consistent and structured dataset, enabling the subsequent Bayesian model to analyze and generate pseudo-series data effectively.

3.3 Model creation

This section defines the probabilistic model used for synthesizing time series data. The Bayesian framework is applied with uncertainty in the parameters within the time series data in mind. The model is structured to capture global patterns shared across all time series and individual-specific variations unique to each time series.

3.3.1 Shared trend

A key element of this model is the shared latent factor, which represents common underlying patterns across all samples in the dataset. The data exhibit certain structural similarities that can be captured through probabilistic modeling. Each sample is unique in its characteristics, but displays glimpses of shared patterns within its class structure. We focus on capturing these patterns to generate synthetic samples of increased quality and representativeness.

In this model, the shared trend is defined as a latent variable, meaning it is not directly observed in the data but inferred from it. The shared trend is modeled as a normal distribution:

$$shared\_trend \sim N(0, 1)$$

The normal distribution has a mean of 0 and a standard deviation of 1. This prior choice indicates that we initially expect the shared trend to have a mean of 0 and be spread out. This allows the model to learn any common trend in the data, such as a global upward or downward trend, which the individual scaling and offset parameters can adjust.

3.3.2 Priors for slope and intercept

In Bayesian modeling, the first step is to define *priors* for the parameters. These priors represent our beliefs about the parameters before we observe any data. For this model, we define two parameters for each time series: the *slope* b_i and the *intercept* a_i. These parameters allow each time series to scale and shift the shared global trend, reflecting that each series can follow a different pattern while still sharing a common underlying trend.

The priors for the intercept and slope are chosen to be normally distributed:

$$a_i \sim N(0, 1), \quad b_i \sim N(1, 0.5)$$

The prior for a_i, the intercept, has a mean of 0 and a standard deviation of 1. This suggests that, before observing the data, we expect the intercept of each series to be around zero, with some flexibility to shift upward or downward. For b_i, the slope, we choose a prior with a mean of 1 and a standard deviation of 0.5, reflecting the belief that most time series will have a moderate upward trend but with room for variability.

These priors help the model adjust to the data. By specifying the priors, we give the model a framework to learn the posterior distributions of these parameters, conditioned on the observed data.

3.3.3 Gaussian Random Walk

To capture the unique patterns in each time series, we introduce a Gaussian Random Walk (GRW). A random walk is a statistical model where the value at each time point is the sum of the previous value and a random noise component. In this case, the random walk captures the evolving, unique patterns specific to each time series that are not shared across the dataset.

GRW assumes that the random noise added at each step follows a normal distribution, which means that the changes from one-time point to the next are Gaussian distributed. The GRW is defined as:

$$\text{unique\_trend}_i \sim \text{GRW}(\sigma = 0.1, \text{shape} = T)$$

Where T is the number of time points in each series.This component allows each time series to have a trend that can deviate from the global shared trend. The GRW flexibly captures these individual variations across sequences.

3.3.4 Likelihood

Once the priors for the parameters are defined, we can model the *likelihood* of the observed data, given the parameters. In the Bayesian framework, the likelihood function represents the probability of observing the data, given a particular set of parameters. It quantifies how well a set of parameters explains the observed data.

In this case, the observed data y_i (the actual time series) is modeled as normally distributed, with the mean equal to the combination of the shared trend, the individual scaling and offset parameters, and the unique random walk trend. The likelihood is expressed as:

$$y_i \sim N(u_i, \sigma = 0.05)$$

where u_i is the combined prediction for the i-th time series, incorporating the shared trend, intercept, slope and unique trend:

$$u_i = a_i + b_i \cdot \text{shared\_trend} + \text{unique\_trend}_i$$

The term $\sigma = 0.05$ represents the standard deviation of the residuals, which captures the noise or errors in the observed data. Using this likelihood function, the model can learn the best-fitting values for the intercept, slope, and unique trends, while accounting for the uncertainty in the data.

3.3.5 Posterior Distribution

The Bayesian approach combines prior information with the observed data to compute the *posterior distribution* of the parameters. After considering the observed data, the posterior distribution represents our updated beliefs about the parameters. This is the core of the Bayesian approach, as it allows us to incorporate prior knowledge (in the form of priors) and update that knowledge based on the data. By drawing samples from the posterior distribution, we can derive a range of possible values for the intercept, slope, shared patterns, and unique variations. These posterior samples allow us to quantify the uncertainty in the model parameters. The posterior distribution is calculated using the Bayes theorem [13].

3.3.6 Posterior Predictive Distribution and Data Generation

Once we have drawn samples from the posterior distribution, we can use the *posterior predictive distribution* to generate new synthetic data. The posterior predictive distribution allows us to simulate new data points based on the parameters learned from the observed data. This is crucial for generating synthetic samples that mimic the characteristics of the original dataset. Using the posterior predictive approach, we draw samples that incorporate both the estimated parameter values and their associated uncertainty, resulting in realistic synthetic data that preserves the statistical properties of the original observations. The posterior predictive distribution can be represented mathematically as:

$$y_{\text{new}} \sim P(y|\theta_{\text{posterior}})$$

This formulation expresses that new data points (y_{new}) are drawn from a probability distribution conditioned on the posterior distribution of parameters ($\theta_{\text{posterior}}$). This mathematical framework allows us to generate synthetic data that inherits both the structural patterns and the inherent variability present in the original observations.

3.4 Postprocessing of synthesized time series

After generating time series data using Bayesian modeling, the next step involves applying postprocessing techniques to refine the samples. While Bayesian methods allow us to capture the underlying patterns and uncertainties within the data, the generated time series can still exhibit unwanted fluctuations and noise. These fluctuations are often due to various factors, such as the stochastic nature of the model, parameter uncertainty, or the inherent randomness in the data generation process.

Time series data often contains the underlying trend or pattern and noise in real-world scenarios [26]. The challenge is distinguishing between the actual signal and the noise so that the resulting time series can more accurately reflect the underlying process. These techniques enhance overall model performance. Also, for qualitative metrics, it will make interpretability easier, as raw generated time series can be challenging to interpret due to erratic fluctuations.

3.4.1 Moving Average

The MA is one of the simplest techniques to smooth time series data by reducing random fluctuations and highlighting the underlying patterns, as used by Raudys et al. [27] in their work. This method computes the average of a fixed number of data points within a sliding window, replacing the central point of the window with this average. The MA effectively reduces noise in the data by averaging short-term variations. The smoothing effect depends on the window size (k): a larger window produces more smoothing but may obscure finer details. Despite its simplicity, the MA has limitations, including a lag effect where the smoothed series trails behind rapid changes in the data. Mathematically, the MA at time t is defined as:

$$\mathrm{MA}_t = \frac{1}{k} \sum_{i=t-k+1}^{t} x_i \tag{3.1}$$

where k is the window size, t is the current time step, and x_i are the data points. This technique is beneficial for identifying long-term patterns in noisy time series data. To apply this technique to our synthesized time series, convolution operation was used, which efficiently computes the weighted sum of data points within the sliding window. The generated time series is smoothed by convolving it with a uniform kernel, where each kernel element is $1/k$. This kernel ensures that all data points within the window are given equal weight in the averaging process.

3.4.2 Exponential Moving Average

EMA is a weighted smoothing technique that assigns exponentially decreasing weights to older data points [28]. Unlike the standard MA, which gives equal weight to all points within the sliding window, the EMA prioritizes recent data, making it more responsive to short-term changes while reducing noise. This feature makes the EMA particularly effective for series with dynamic patterns, as it captures more recent fluctuations without completely discarding older patterns.

The EMA is computed recursively, with each new value being a combination of the current data point and the previous EMA value. Mathematically, the EMA is expressed as follows:

$$\mathrm{EMA}_t = \alpha \cdot x_t + (1-\alpha) \cdot \mathrm{EMA}_{t-1} \tag{3.2}$$

where:

- x_t is the current data point,
- EMA_{t-1} is the EMA value at the previous time step,
- α is the smoothing factor, which determines the weight of the current data point.

The smoothing factor α is calculated as:

$$\alpha = \frac{2}{s+1} \tag{3.3}$$

where s is the span, a parameter that controls the degree of smoothing. A smaller span results in a larger α, making the EMA more responsive to recent changes. Conversely, a larger span provides greater smoothing at the cost of responsiveness.

In this study, we applied exponential smoothing with a span of 10 to reduce fluctuations in the generated samples while preserving their essential characteristics.

3.4.3 Savitzky-Golay Filter

The Savitzky-Golay filter is a polynomial smoothing technique that reduces noise in time series data while preserving higher-order features like peaks and valleys [29]. Unlike simple averaging methods, which may overly smooth the data and distort its shape, the Savitzky-Golay filter fits a polynomial of a specified degree to the data points within a sliding window. It replaces the center point of the window with the value from the fitted polynomial. This approach ensures that the overall structure of the data is maintained while smoothing out random fluctuations.

Mathematically, the filter applies a least-squares polynomial fit to each data window. For a data point y_t, the smoothed value is expressed as:

$$y_t = \sum_{i=-m}^{m} c_i x_{t+i} \tag{3.4}$$

Where m is half the window size (window_length = $2m$+1), c_i are the coefficients of the polynomial determined by the least-squares fit, and x_{t+i} are the original data points within the window. The degree of the polynomial (p) determines the level of flexibility in capturing the data's shape, with higher degrees allowing more complex patterns to be preserved.

In this study, we applied a Savitzky-Golay filter with a polynomial of degree 3 and a window length of 11 to smooth the generated samples, effectively preserving important features while reducing noise.

3.4.4 Wavelet Denoising

Wavelet denoising is a powerful technique for reducing noise in time series data while preserving time-domain and frequency-domain information [30]. Unlike traditional smoothing methods that operate exclusively in the time or frequency domain, wavelet denoising decomposes a time series into multiple scales using wavelet transforms. This allows for localized analysis and processing of the signal at different resolutions, making it especially effective for handling non-stationary data.

Wavelet denoising involves three main steps: decomposition, thresholding and reconstruction. First, the time series is decomposed into wavelet coefficients representing the data at different scales (frequency components). Noise, typically characterized by high-frequency components, is suppressed by applying a threshold to the wavelet coefficients. Finally, the denoised time series is reconstructed from the modified coefficients.

The generated time series is processed with the Discrete Wavelet Transform (DWT) using the Daubechies 4 wavelet (db4) and a decomposition level of 3. Soft thresholding with a threshold value of 0.2 is applied to the wavelet coefficients. The denoised series is reconstructed, ensuring its length matches the original series.

3.4.5 Low-pass Filtering

Low-pass filtering is a technique used to remove high-frequency noise from time series data while preserving the lower-frequency components that are typically associated with the actual signal [31]. This is especially useful when the time series contains random fluctuations or short-term irregularities that do not reflect the underlying trend. Low-pass filtering works by attenuating (reducing) the high-frequency components of the signal, which are often considered to be noise, while allowing the lower-frequency components, which typically correspond to the actual signal, to pass through.

The Low-pass filter can be implemented in various ways, but one of the most common methods is using the Butterworth filter, which is designed to have a flat frequency response in the passband. This means that it does not introduce any distortion to the signal in the frequency range of interest.

In this study, a low-pass Butterworth filter is applied to the generated time series to remove high-frequency noise. The filter design incorporates specific cutoff frequency and order parameters to achieve optimal noise reduction while preserving the signal's essential characteristics. A zero-phase filtering technique is implemented, which performs both forward and backward passes through the data, thereby ensuring that the phase of the signal remains undistorted throughout the filtering process.

3.5 Evaluation

One of the most challenging aspects of working with synthetic time series is evaluating their quality and performance. In their article, Stenger et al. [32] create a high-standard discussion about quantifying the evaluation of these series. Pseudo-time series must preserve the appearance of sequential patterns and statistical properties while ensuring the synthetic data remains useful for practical applications. Yadav et al. [20] mention that there is no standardized set of metrics for evaluating the generated data for time series. As a result, various approaches have been proposed to assess the quality of the generated time series.

To evaluate the performance and quality of the generated time series, we employ ten evaluation metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Pearson Correlation Coefficient, Spearman Correlation Coefficient, Scaled Quantile Loss (SQL), Dynamic Time Warping(DTW), Time-Weighted Dynamic Time Warping (TDTW), Cross-Correlation Function (CCF), Hausdorff Distance and Fréchet Distance. Each metric provides unique insights into the time series' accuracy and correlation with the original data. Additionally, several postprocessing techniques were applied to the generated time series before the evaluation, including Moving Average (MA), Exponential Moving Average (EMA), Savitzky-Golay Filter, Wavelet Denoising and Low-pass Filtering. These preprocessing methods were essential in reducing noise, smoothing fluctuations, and preserving the underlying patterns in the data, which allowed for a more accurate evaluation of the synthetic time series' quality.

3.5.1 Evaluation Metrics

While our dataset does not contain chronologically indexed data, we treat these sequences as structured series where the order of elements reflects meaningful patterns. Therefore, we adopt time-dependent similarity metrics such as DTW, TDTW and Frechet Distance, not for capturing time-based causality but for their ability to quantify structural and sequential alignment. These metrics allow us to assess whether the generated pseudo-time series preserves the internal shape, continuity, and variability of the original sequences, even in the absence of actual timestamps.

This study employs several established evaluation metrics to quantitatively assess the performance of the synthetic time series generation methods. Each metric provides unique insights into different aspects of the generated data's quality.

Mean Absolute Error

As the name suggests, MAE is a measure of errors between paired observations. MAE measures how close the predictions are to the actual outcomes on average [33]. It is easy to understand and interpret, and errors are treated equally. However, since all errors are similarly treated, it does not differentiate between small and large errors. It is the average of the absolute errors and is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where y_i is the actual value (which is the original time series) and is $\hat{y}_i$ the predicted value (which is the generated time series). A lower MAE value indicates better model accuracy, as it signifies smaller errors on average.

Root Mean Squared Error

Another commonly used metric measures the differences between a model's predicted values and observed values. It is the square root of the average squared differences between prediction and actual observation [33]. Unlike MAE, where all the errors are equal, RMSE gives a relatively high weight to significant errors, making it useful when significant errors are undesirable, and it squares the differences before averaging. It is formulated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

As in MAE, in RMSE, y_i represents the actual value, while $\hat{y}_i$ is the predicted value. In comparison, RMSE is more sensitive to outliers than MAE, and MAE is useful when all errors are equally important. At the same time, RMSE is preferred when larger errors are more significant. Both metrics are implemented in this research to ensure robustness and understand the model's behavior across different error magnitudes. Hyndman and Koehler [34] discuss MAE and RMSE in their work, though it should be noted that they apply these metrics in a forecasting context, where RMSE $= 0$ indicates perfect prediction. In contrast, for synthetic time series evaluation as used in this study, RMSE $= 0$ would indicate an exact copy of the original data rather than a prediction. This distinction is important when interpreting the results, as our goal is to generate data with similar statistical properties rather than to predict future values.

Pearson Correlation

Another metric that has been used is the Pearson correlation coefficient. Werner et al. [35] bring an adequate discussion of whether Pearson correlation can be a measure for the linear relationships between time series. This method measures the linear relationship between two variables. This correlation produces a score within the interval of -1 and +1. These intervals indicate positive perfect and negative perfect linear relationships between two compared variables: original and generated time series. Two uncorrelated objects would have a Pearson score near zero [36].

The Pearson correlation is defined as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

r represents the Pearson correlation. x and y are two compared variables, while $\bar{x}$ and $\bar{y}$ are their respective means. n is the number of paired samples. Pearson correlation has been selected as one of the evaluation metrics because it serves as an effective indicator of model performance in capturing the linear patterns in the data. Also, the Pearson correlation is easy to implement and interpret as the correlation coefficient ranges from -1 to +1, making it easier to make inferences.

Spearman Correlation Coefficient

To further assess the relationship between the predicted and actual values, the Spearman Correlation Coefficient provides a robust measure, particularly for capturing monotonic trends. It describes the strength and direction of a monotonic relationship between two variables. It is nonparametric and does not assume a specific distribution between variables. In their paper, Ye et al. [37] evaluates the similarity between two time series generated by two sensors by using Spearman correlation.

Spearman Correlation can be described with the following formula :

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}$$

In this formula, the parameter d_i is the difference between the ranks of the corresponding values of x and y, two variables chosen for examining correlation. n represents the number of paired samples.

Like the Pearson correlation, the Spearman Correlation provides a score within the intervals of -1 and +1. ρ equaling -1 means these two variables have a negative monotonic relationship while equaling +1 shows the perfect negative monotonic relationship. If ρ is 0, then there is no monotonic relationship.

Spearman correlation is helpful because it can help us understand whether the relationship between the time series is monotonic (i.e., consistently increasing or decreasing), even if it is not linear. As mentioned in section 3.1, our time series dataset is based on grapevine leaves. Although there is a repeating pattern between time series samples within classes, they are not consistently increasing or decreasing. Therefore, the Spearman correlation is a vital evaluation metric for this research.

Scaled Quantile Loss

SQL helps measure the accuracy of predicted quantiles by comparing them directly to the actual values in the time series [38]. Unlike traditional loss functions, which typically focus on the mean or absolute differences between the predicted and actual values, SQL focuses on assessing how well the model predicts specific quantiles of the distribution, which is essential given the nature of our data. Additionally, to test how robust the model is constructed, some possibility of outliers needs to be taken into account as SQL handles both overestimations and underestimations in a way that traditional error metrics like MAE or RMSE fail to achieve.

The *SQL* for a given quantile is computed by comparing the predicted quantiles $\hat{y}_q$ to the true values y. The SQL formula is given by:

$$\text{SQL} = \frac{2\sum_{i=1}^{n}(y_i - \hat{y}_{q,i}) \cdot (q - \mathbb{I}(y_i < \hat{y}_{q,i}))}{\sum_{i=1}^{n}|y_i|}$$

Where:

- y_i is the true value at index i,
- $\hat{y}_{q,i}$ is the predicted quantile value at index i,
- q is the quantile of interest (e.g., 0.1 for the 10th percentile, 0.5 for the median, 0.9 for the 90th percentile),
- $\mathbb{I}(y_i < \hat{y}_{q,i})$ is an indicator function that equals 1 if y_i is less than the predicted quantile value $\hat{y}_{q,i}$, and 0 otherwise.

The numerator represents the weighted sum of the errors, where overestimations are penalized differently from underestimations based on the quantile. The denominator scales the loss by the sum of the absolute values of the true values, ensuring that the SQL is independent of the scale of the data.

After it is applied, SQL provides a non-negative score for evaluation. A lower SQL score signifies that the model's predictions are closer to the true quantiles, indicating better performance. On the other hand, a higher SQL score suggests a larger discrepancy between the predicted and actual values, pointing to poorer accuracy in capturing the desired quantiles.

Dynamic Time Warping

DTW is a powerful metric to assess the similarity between two-time series by measuring the minimum cumulative distance between them, considering temporal shifts [39]. Unlike traditional distance metrics like Euclidean distance, which only measure point-to-point differences, DTW accounts for variations in the timing of the data points, allowing for flexible alignment between sequences. This makes DTW particularly useful for comparing time series that may be similar but have temporal distortions or shifts in their patterns.

DTW works by aligning two-time series so that the distance between corresponding points is minimized. Although it is beneficial for comparing sequences with misalignments or phase shifts, such as in time series with varying sampling rates or noisy data, our dataset has been normalized and cleaned through postprocessing. As a result, the need to handle such misalignments carries less significance, but DTW still provides a valuable comparison metric for evaluating the similarity between the time series.

The DTW distance between two time series $\mathbf{x} = (x_1, x_2, ..., x_n)$ and $\mathbf{y} = (y_1, y_2, ..., y_m)$ is computed by minimizing the cumulative distance between points in the series. The DTW distance $D(i, j)$ between points x_i and y_j is defined recursively as:

$$D(i,j) = \text{dist}(x_i, y_j) + \min \begin{cases} D(i-1, j), \\ D(i, j-1), \\ D(i-1, j-1) \end{cases}$$

Where $\text{dist}(x_i, y_j)$ is the distance between the two points (typically Euclidean distance, but in this work, squared Euclidean distance is used), and the recurrence relation ensures that the optimal path is found by considering the cumulative cost of all possible alignments. The final DTW distance is the minimum cumulative distance between the two sequences found by backtracking through the warping path.

This study employs an approximation approach to DTW, which provides an approximate solution for computing the DTW distance in $O(n)$ time complexity (compared to the $O(n^2)$ complexity of the exact DTW). The distance between corresponding points in the two time series is quantified using the squared Euclidean distance metric. A lower DTW distance indicates that the two-time series are more similar, while a higher distance suggests greater separation.

Time-Weighted Dynamic Time Warping

Proposed by the work of Hailin Li [24], TDTW builds on DTW by introducing a time-dependent weighting function, which adjusts the influence of each data point during alignment. In this research, TDTW allows us to assign different importance to various segments of the time series based on their analytical significance. This capability is particularly valuable when certain patterns or time periods in the synthetic data require more precise reproduction than others.

The primary idea behind TDTW is to modify the original DTW distance calculation by introducing a time-dependent weight function. This enables more nuanced comparison between time series where the importance of matching may vary across the timeline.

The weighted DTW distance for each point t is given by:

$$D(i,j) = \text{dist}(x_i, y_j) \cdot w(t)$$

Where:

- $D(i,j)$ is the weighted distance between points i and j,
- $\text{dist}(x_i, y_j)$ is the original distance (e.g., Euclidean distance),
- $w(t)$ is the time-dependent weight, which can be adjusted depending on the importance of the time points.

While TDTW distance typically focuses on temporal alignment between time series, its application in our context requires clarification. Lower TDTW distance indicates that two sequences are well-aligned in terms of their overall pattern similarity, even if our dataset does not contain explicit temporal patterns or time-dependent relationships.

In our research, we use TDTW as a robust similarity metric that can handle potential variations in sequence alignment. This approach allows us to evaluate how well our synthetic data reproduces the underlying pattern distributions present in the original data, without making assumptions about time-dependency that isn't present in our dataset.

Hausdorff Distance

This metric allows the measurement of similarity between two sets of points by computing the greatest of all the distances from a point in one set to the closest point in the other set. In the context of time series, Hausdorff Distance measures the maximum deviation between two-time series by considering the worst-case alignment [40]. This makes it particularly useful in this project for evaluating how well the generated time series aligns with the original, especially when identifying significant deviations or outliers that might affect the overall quality of the synthetic data.

The Hausdorff Distance $H(A,B)$ between two sets A and B is defined as:

$$H(A,B) = \max\left(\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right)$$

Where:

- A and B are the two sets of points (in our case, two time series),
- $d(a, b)$ is the distance between the two points (typically Euclidean distance),
- sup represents the supremum (the least upper bound) of the set of distances, and
- inf represents the infimum (the greatest lower bound) of the set of distances.

This means that the distance to the closest point in the other set is computed for each point in one set, and the maximum of these values is taken. The result gives the maximum "error" regarding alignment between the two sets.

To evaluate our model using the Hausdorff distance metric, we compute the directed Hausdorff distance between two sets of points. This calculation identifies the maximum distance between points in one time series and their closest counterparts in the other, providing a robust measure of dissimilarity between the original and synthetic datasets.

The Hausdorff Distance measures the worst-case misalignment between the actual and predicted time series and produces a non-negative score. A lower Hausdorff distance indicates that the two-time series are well-aligned, while a higher value suggests that the time series have significant discrepancies at some points.

Fréchet Distance

Fréchet Distance is a metric used to measure the similarity between two curves or paths in a metric space, handy for time series data [22]. Unlike other distance metrics, It considers the spatial and temporal relationships between two-time series. The idea behind Fréchet Distance is to calculate the minimum "coupling" between the two curves, which represents the optimal way to match each point in one curve to a point in the other curve while minimizing the total distance. This method is often called the "dog and handler problem," where one curve represents the path of a dog and the other represents the path of its handler. The Fréchet Distance measures the minimum distance the dog must walk to stay with its handler, considering that both move along the curves.

In time series analysis, Fréchet Distance can be used to assess the similarity between two-time series by treating each series as a curve in 2D space, where each point is paired with its corresponding time index. Fréchet Distance identifies the maximum distance between corresponding points along an optimal alignment of the curves, considering both their location and sequential ordering.

Fréchet Distance between two curves P and Q, where $P = (p_0, p_1, ..., p_n)$ and $Q = (q_0, q_1, ..., q_m)$, is defined as:

$$d_F(P, Q) = \inf_{\alpha} \sup_{t \in [0,1]} \|p_{\alpha(t)} - q_t\|$$

Where:

- $p_{\alpha(t)}$ and q_t represent points on the curves P and Q, respectively,
- $\|p_{\alpha(t)} - q_t\|$ is the distance between corresponding points on the curves,
- $\alpha(t)$ is a continuous mapping of time t along the curves, representing the optimal alignment between the two time series,
- The supremum (sup) is taken over all time steps, and the infimum (inf) is over all possible couplings of the points.

Just as in the Hausdorff Distance, a non-negative score is produced for evaluation, and a smaller Fréchet distance indicates that the two-time series are closely aligned. At the same time, a larger value suggests that the generated time series is farther from the true time series in terms of both timing and value.

Cross-Correlation Function

Although the pseudo-time series in this study are not indexed by real chronological time, the Cross-Correlation Function (CCF) can still provide valuable insights into the structural alignment between the original and generated sequences [41]. In this context, the CCF measures how closely the generated sequence aligns with the original sequence across different shifts along their index, treating the sequence position as a pseudo-temporal order. This analysis helps determine whether the generated series captures the sequential dependencies and ordered patterns observed in the original data.

While traditional CCF is often used to detect lead-lag relationships in real time series, here it serves as a proxy for structural consistency, rather than actual temporal causality. By examining correlations across different index shifts, CCF enables the detection of consistent patterns, possible distortions, or systematic misalignments introduced during the data generation process.

The cross-correlation between two time series $y(t)$ and $x(t)$ at a given lag k is defined as:

$$CCF(k) = \frac{\sum_{t=1}^{N-k}(y_t - \overline{y})(x_{t+k} - \overline{x})}{\sqrt{\sum_{t=1}^{N}(y_t - \overline{y})^2 \sum_{t=1}^{N}(x_t - \overline{x})^2}}$$

Where:

- $y(t)$ and $x(t)$ are the two time series,
- $\overline{y}$ and $\overline{x}$ are the means of the time series $y(t)$ and $x(t)$, respectively,
- k is the time lag, and
- N is the total number of time points in the series.

For the analysis, the sequences are flattened into one dimensional arrays, allowing standardized correlation calculations across different lags. However, because the CCF provides a curve instead of a single score, it can be challenging to summarize the model's performance.

- Positive CCF values: A positive correlation at a shift k indicates that the two sequences move similarly after adjusting the index by k steps. For instance, if the CCF is positive at lag 1, higher values in the original sequence correspond to higher values in the generated sequence at the next index.
- Negative CCF values: A negative correlation at a shift k suggests that higher values in one sequence align with lower values in the other after the shift, indicating an inverse structural relationship.
- Lag 0: The CCF value at lag 0 measures the direct structural correlation between the original and generated series without any shift, representing the most straightforward alignment.
- Peak in CCF: The highest positive or negative peak in the CCF plot indicates the shift at which the two sequences are most strongly aligned, providing insight into potential systematic misalignments introduced during generation.

Chapter 4

Results

Suppose we recall the original time series used from Cabernet Franc class and put them side by side with a synthesized time series. In that case, our synthesized time series is considerably noisier than the actual data. Still, it captures the same fluctuations as the original time series samples, shown in Figure 4.1. With various postprocessing techniques to reduce noise and smooth the series, the synthesized time series samples demonstrate significant potential. Considering that Bayesian models do not use deep learning techniques to make the data generation and it is considerably faster to make initial predictions compared to machine learning methods [42], Bayesian models did a satisfactory job at generating time series samples from the original time series samples.

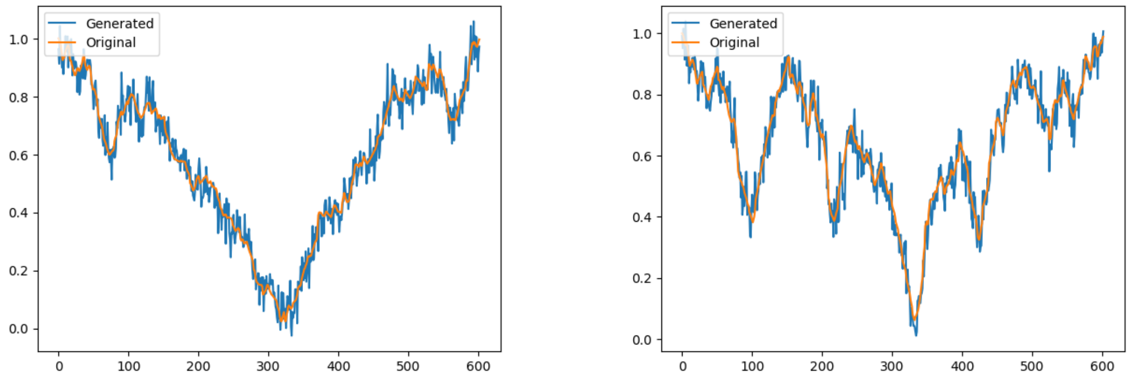


Figure 4.1: Synthesized (blue) and original (orange) time series from Cabernet Franc class

4.1 Evaluation Metrics

To display the quality of pseudo-time series in detail and make the results more assessable, ten evaluation metrics that were explained in Subsection 3.5.1 are used. To evaluate the accuracy of the synthesized time series, various comparisons were conducted between the original data and the generated versions of the time series. For these comparisons, a sample from the Cabernet Franc class was selected and processed using the methods discussed earlier. A synthetic time series sample was generated and further refined using the postprocessing techniques outlined in Subsection 3.4. The results from the original synthesis process, along with those from the postprocessing methods, were recorded and compared against the original time series sample.

Our goal is to achieve consistent performance across all evaluation metrics. If the model produces lower scores on metrics where lower values indicate better performance but worse scores on others, it may suggest overfitting or that the model is only optimizing specific aspects of the time series. Where possible, the results will be compared with those from similar approaches in relevant papers that utilize the same evaluation metrics. Given the limited research conducted using non-GAN strategies, these metrics may also be compared to results from GAN-based approaches.

Table 4.1 presents MAE and RMSE evaluation metrics comparing various approaches to the original time series. Lower values indicate closer similarity to the original data. Our synthetic time series demonstrates good performance with relatively low error metrics (MAE of 0.0360, RMSE of 0.0447), effectively capturing many characteristics of the original data.

Among the postprocessing methods, Low-pass filtering and Savitzky-Golay filtering yield the best results (MAE of 0.0187/0.0194 and RMSE of 0.0236/0.0248, respectively), significantly outperforming the base synthetic approach. These methods appear to more accurately preserve the essential patterns of the original data while potentially reducing noise. In contrast, the moving average methods (MA and EMA) show substantially higher error values (MAE of 0.1335/0.1164 and RMSE of 0.1537/0.1442), suggesting these approaches may distort important signal characteristics.

The Wavelet method demonstrates intermediate performance (MAE of 0.0680, RMSE of 0.0714), better than moving average techniques but not as effective as the other approaches. When compared to the GANs strategy reported in [9], our base synthetic method appears somewhat inferior; however, when enhanced with appropriate postprocessing (particularly Low-pass or Savitzky-Golay filtering), our approach achieves superior error metrics.

Comparison	**MAE**	**RMSE**
Original to Synthetic	0.0360	0.0447
Original to MA	0.1335	0.1537
Original to EMA	0.1164	0.1442
Original to Savitzky-Golay	0.0194	0.0248
Original to Wavelet	0.0680	0.0714
Original to Low-pass	0.0187	0.0236

Table 4.1: MAE and RMSE values for the comparison of the original to synthetic and postprocessed time series

Table 4.2 presents the Pearson and Spearman correlation scores for various time series approaches compared to the original data. Higher correlation values (closer to +1) indicate better alignment and consistency between the compared time series. Pearson correlation captures linear relationships, while Spearman correlation evaluates monotonic relationships, making it robust to nonlinear trends.

Comparison	**Pearson**	**Spearman**
Original to Synthetic	0.9882	0.9855
Original to MA	0.8455	0.8249
Original to EMA	0.8836	0.8975
Original to Savitzky-Golay	0.9968	0.9948
Original to Wavelet	0.9975	0.9957
Original to Low-pass	0.9969	0.9950

Table 4.2: Spearman and Pearson correlation values for the comparison of the original to synthetic and postprocessed time series

Our synthetic time series demonstrates strong correlation with the original data (Pearson: 0.9882, Spearman: 0.9855), indicating that our Bayesian model effectively captures the underlying structure of the original series. However, certain postprocessing techniques yield even stronger results.

Among the postprocessing methods, Wavelet denoising achieves the highest correlation scores (Pearson: 0.9975, Spearman: 0.9957), closely followed by Low-pass filtering (Pearson: 0.9969, Spearman: 0.9950) and Savitzky-Golay filtering (Pearson: 0.9968, Spearman: 0.9948). These three methods demonstrate exceptional ability to preserve the patterns and relationships in the time series while potentially reducing noise.

In contrast, the moving average methods show more moderate performance. The EMA achieves reasonable correlation values (Pearson: 0.8836, Spearman: 0.8975), retaining more dynamic patterns than the standard MA, which exhibits the lowest correlation values (Pearson: 0.8455, Spearman: 0.8249) due to its more aggressive smoothing effect that may eliminate important signal features.

These correlation results, combined with the MAE and RMSE metrics, provide comprehensive evidence that our approach, particularly when enhanced with appropriate postprocessing techniques like Savitzky-Golay filtering or Low-pass filtering, successfully generates high-quality synthetic time series that closely resemble the original data.

Table 4.3 presents the SQL scores for comparisons between the original time series and synthetic or postprocessed series across three quantiles: $q = 0.1, 0.5$, and 0.9. Lower SQL values indicate better alignment between the compared series at specific quantiles.

Comparison	**Quantile 0.1**	**Quantile 0.5**	**Quantile 0.9**
Original to Synthetic	0.0647	0.0581	0.0516
Original to MA	0.2708	0.2294	0.1881
Original to EMA	0.2612	0.1986	0.1361
Original to Savitzky-Golay	0.0372	0.0307	0.0241
Original to Wavelet	0.0227	0.1137	0.2046
Original to Low-pass	0.0367	0.0302	0.0237

Table 4.3: SQL values for different quantiles comparing the original to synthetic and postprocessed time series

Our synthetic time series demonstrates good performance with consistent SQL scores across all quantiles (0.0647, 0.0581, and 0.0516 for the 0.1, 0.5, and 0.9 quantiles respectively), showing that our Bayesian model effectively captures the distributional characteristics of the original time series.

Among the postprocessing techniques, Savitzky-Golay and Low-pass filtering stand out with exceptionally low SQL scores across all quantiles. Low-pass filtering achieves the best overall performance (0.0367, 0.0302, and 0.0237), closely followed by Savitzky-Golay (0.0372, 0.0307, and 0.0241). These methods successfully balance noise reduction while preserving quantile-level patterns, ensuring strong alignment with the original series.

Wavelet denoising shows interesting behavior with excellent performance at lower quantiles (0.0227 for q=0.1) but increasing SQL scores for middle and upper quantiles (0.1137 and 0.2046), suggesting it may alter the distribution of higher values in the time series.

The moving average methods exhibit notably higher SQL scores across all quantiles, with MA being the least effective (0.2708, 0.2294, and 0.1881). EMA performs somewhat better than MA (0.2612, 0.1986, and 0.1361), as it retains more of the original series' dynamic features, but both methods appear to simplify the data, losing important distributional details.

These results further confirm the effectiveness of our Bayesian model approach and highlight the particular strength of Savitzky-Golay and Low-pass filtering techniques in enhancing the alignment between the generated and original time series across different quantile levels.

Table 4.4 presents the DTW and TDTW distances between the original pseudo-time series and synthetic or postprocessed series. For these metrics, lower values generally indicate better pattern similarity between compared series. DTW measures sequence similarity by finding the optimal alignment between two series, while TDTW adds position-based weighting to this comparison. Since we're working with pseudo-time series, these distance metrics primarily reflect pattern similarity rather than temporal alignment.

Comparison	**DTW Distance**	**TDTW Distance**
Original to Synthetic	0.3646	17.7593
Original to MA	0.8267	12.0719
Original to EMA	0.6667	13.4493
Original to Savitzky-Golay	0.0864	5.7058
Original to Wavelet Denoised	0.3103	10.4362
Original to Low-pass Filter	0.0786	5.2335

Table 4.4: DTW and TDTW distances for comparing the original to synthetic and postprocessed time series

Our synthetic series demonstrates a relatively low DTW distance of 0.3646, indicating good pattern similarity with the original pseudo-time series. Its higher TDTW distance of 17.7593 reflects differences when position-weighted comparison is applied, suggesting some variations in how patterns are distributed across the series.

Among the postprocessing techniques, Low-pass Filtering and Savitzky-Golay achieve exceptionally low DTW distances (0.0786 and 0.0864 respectively) and

TDTW distances (5.2335 and 5.7058). These results indicate these techniques produce series that closely match the pattern characteristics of the original data across both metrics.

Wavelet Denoising shows moderate similarity with a DTW distance of 0.3103 and a TDTW distance of 10.4362, comparable to our synthetic approach in overall pattern matching but with some differences in position-weighted comparison.

The moving average methods show higher distances, with MA having a DTW value of 0.8267 and TDTW of 12.0719, while EMA performs somewhat better (DTW: 0.6667, TDTW: 13.4493). These higher values reflect how moving average techniques (MA and EMA) tend to modify the pattern characteristics of the original series.

The results suggest that Low-pass Filtering and Savitzky-Golay are particularly effective at preserving the pattern characteristics of the original pseudo-time series.

Table 4.5 presents the Hausdorff and Fréchet distances between the original pseudo-time series and various postprocessed and synthetic series. The Hausdorff distance measures the maximum deviation between two series, while the Fréchet distance evaluates the minimum path that aligns two curves, accounting for both shape and sequential alignment. For both metrics, lower values indicate better similarity.

Comparison	**Hausdorff Distance**	**Fréchet Distance**
Original to Synthetic	0.0122	0.1331
Original to MA	0.0161	0.4143
Original to EMA	0.0157	0.3346
Original to Savitzky-Golay	0.0309	0.0670
Original to Wavelet Denoised	0.0453	0.1283
Original to Low-pass Filter	0.0318	0.0710

Table 4.5: Hausdorff and Fréchet distances for comparing the original to synthetic and postprocessed time series

Our synthetic pseudo-time series demonstrates good alignment with the original, showing a relatively low Hausdorff distance of 0.0122 and a Fréchet distance of 0.1331. These values suggest that the synthetic series effectively reproduces the pattern characteristics of the original series with limited deviations.

Among the postprocessing techniques, Savitzky-Golay and Low-pass filtering stand out with the lowest Fréchet distances (0.0670 and 0.0710 respectively), indicating excellent preservation of curve shapes and sequential patterns. However, their Hausdorff distances (0.0309 and 0.0318) are slightly higher than the synthetic series, suggesting occasional larger point-wise deviations despite better overall curve similarity.

Wavelet Denoising shows moderate performance with a Hausdorff distance of 0.0453 and a Fréchet distance of 0.1283, comparable to our synthetic approach in Fréchet distance but with higher maximum deviations as indicated by the Hausdorff metric.

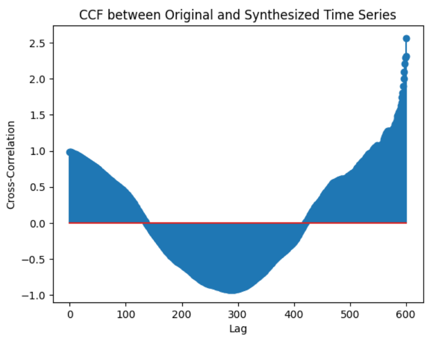
The moving average methods exhibit a different pattern: relatively low Hausdorff distances (MA: 0.0161, EMA: 0.0157), but substantially higher Fréchet distances (MA: 0.4143, EMA: 0.3346). This unusual combination suggests that while these methods may limit the maximum point-wise deviation, they significantly alter the overall curve shape and sequential pattern characteristics of the original pseudo-time series.

These results highlight the complementary nature of these distance metrics when evaluating pattern similarity. While our synthetic approach maintains good balance across both metrics, Savitzky-Golay and Low-pass filtering excel at preserving overall pattern shapes (low Fréchet), and moving average techniques limit maximum deviations (low Hausdorff) but at the cost of pattern distortion.

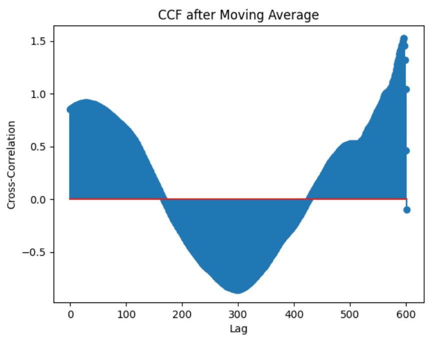
Last evaluation metric that is used is CCF. Higher CCF values indicate a better preservation of the correlation structure between the two time series. As seen in Figure 4.2, the synthetic time series maintains a relatively strong correlation with the original series, although there are some deviations at higher lags, indicating that the synthetic model could be further refined to capture long-range dependencies.

Among the postprocessing methods, Savitzky-Golay and Wavelet Denoising filters show the best performance, as they preserve the original correlation structure with minimal deviations. These methods retain the overall characteristics of the time series, making them suitable for postprocessing without losing significant temporal information. On the other hand, the MA and EMA smoothing methods show more significant deviations from the original series, especially at higher lags. This suggests that these methods introduce smoothing effects that reduce the correlation, indicating a loss of finer temporal details in the time series. Low-pass filtering, similar to the Savitzky-Golay and Wavelet Denoising methods, performs relatively well by preserving the overall structure of the time series, but it still causes a slight reduction in correlation, especially at higher lags.

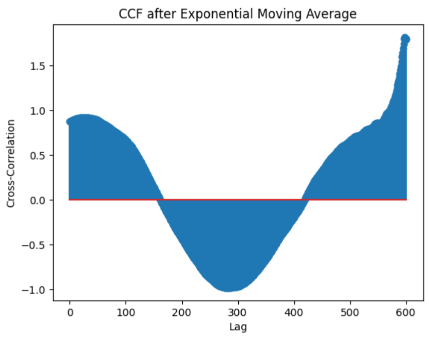
In summary, while the synthetic time series captures the overall trend of the original series, methods such as Savitzky-Golay and Low-pass filtering demonstrate superior performance across multiple evaluation metrics.



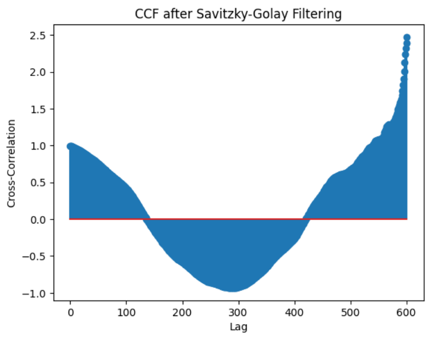
Original vs Synthetic



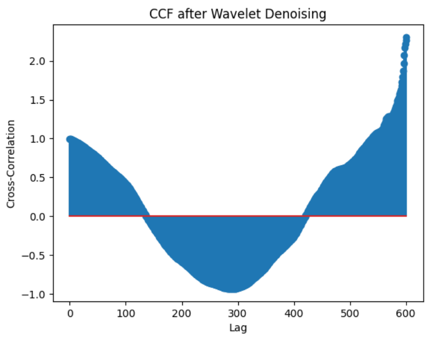
Original vs after MA



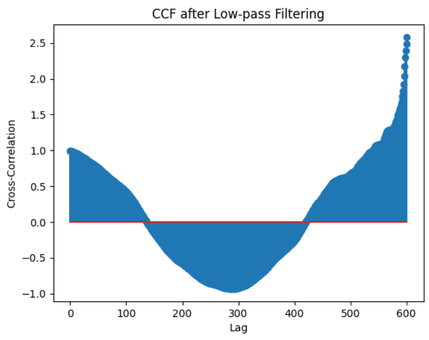
Original vs after EMA



Original vs after Savitzky-Golay



Original vs after Wavelet

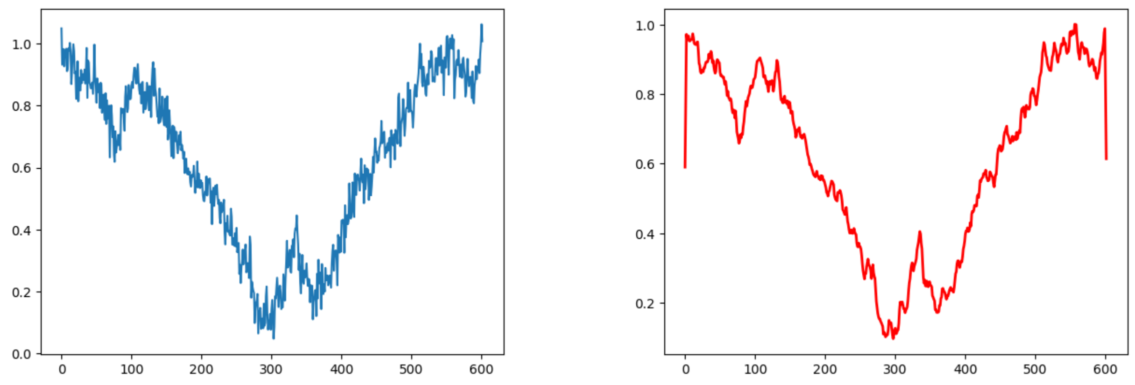


Original vs Low-pass

Figure 4.2: Comparison of CCF for original to synthetic and postprocessed time series

4.2 Qualitative Assessment of Postprocessing Techniques

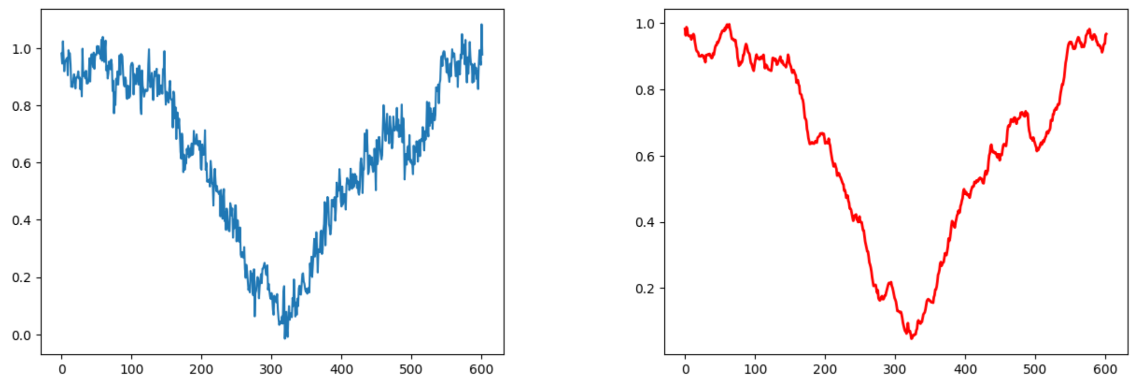
While numerical metrics provide a quantitative assessment of postprocessing effectiveness, visual comparisons help clarify how each method alters the structure of the generated time series. This qualitative perspective complements the statistical evaluation in subsection 4.1 by illustrating the trade-off between noise reduction and pattern preservation.



Synthesized time series sample After MA applied

Figure 4.3: Synthetic and smoothed synthetic time series using MA

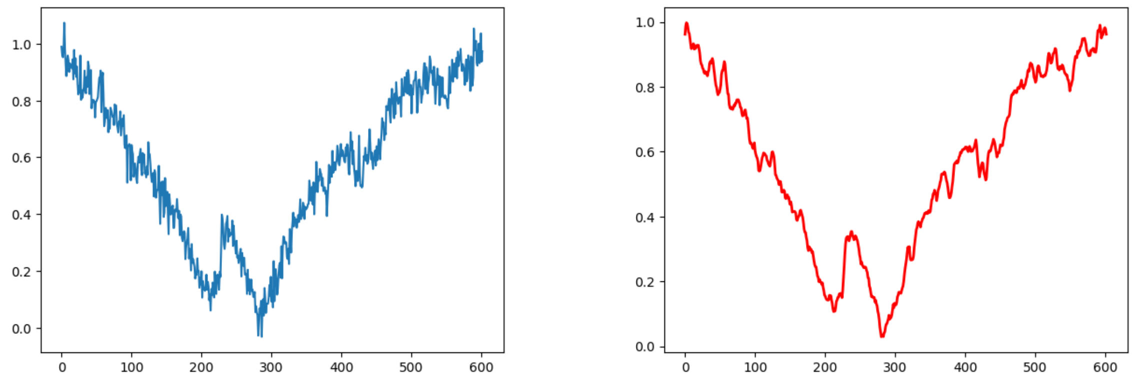
The MA smoothing technique (Figure 4.3) moderately reduces fluctuations, resulting in a smoother curve. However, MA draws incomprehensible lines at the start and the end of the time series while smoothing and causing a detail loss, which aligns with the high MAE and RMSE values compared to other smoothing techniques, reported in Table 4.1. This technique effectively removes noise but may introduce some elements not present in the original data.



Synthesized time series sample After EMA applied

Figure 4.4: Synthetic and smoothed synthetic time series using EMA

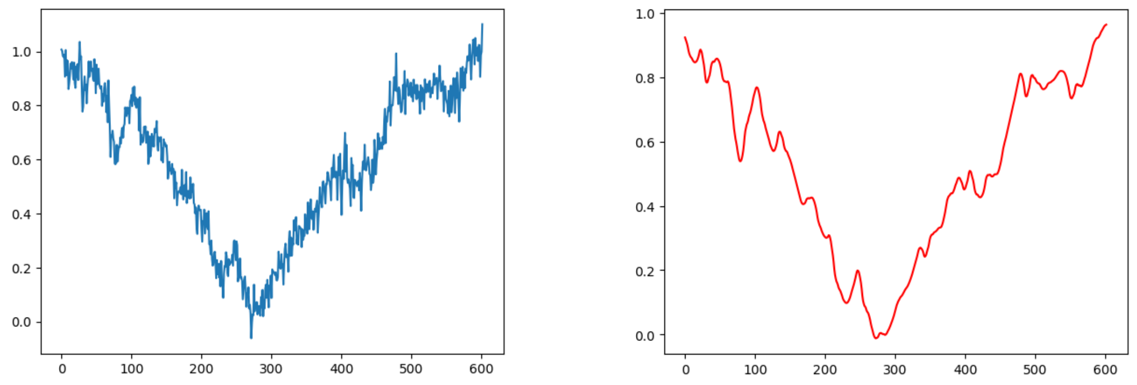
EMA (Figure 4.4) responds more dynamically to recent changes compared to the standard MA. This is visually evident in the smoother transitions that better retain the shape of the original series. However, the quantitative results still indicate underperformance relative to other methods, likely due to residual oversmoothing.



Synthesized time series sample　　After Savitzky-Golay filter applied

Figure 4.5: Synthetic and smoothed synthetic time series using Savitzky-Golay filter

Unlike averaging methods, the Savitzky-Golay filter (Figure 4.5) preserves higher-order features such as peaks and valleys. This preservation is visible in the filtered sequence and aligns with the strong performance in RMSE, correlation, and SQL metrics. It maintains visual clarity while minimizing distortion.



Synthesized time series sample　　After Wavelet denoising applied

Figure 4.6: Synthetic and smoothed synthetic time series using Wavelet denoising

Wavelet Denoising (Figure 4.6) retains the general shape of the sequence while selectively reducing high-frequency noise. However, the resulting curve appears excessively flattened. This technique is effective for removing noise but may oversimplify important features.

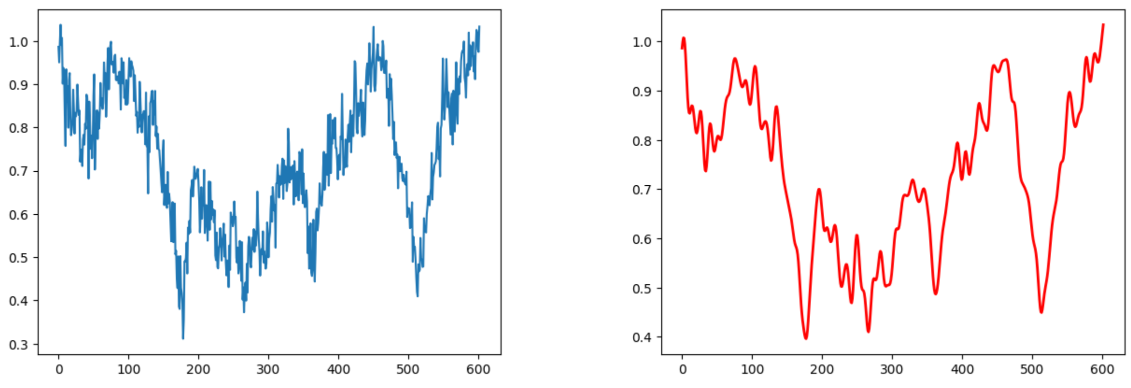


Figure 4.7: Synthetic and smoothed synthetic time series using Low-pass filtering

Low-pass filtering (Figure 4.6) provides an effective approach for suppressing noise while preserving essential structure. Its strong performance across almost all evaluation metrics, especially MAE, RMSE, and DTW, supports this observation.

These visualizations support the metric-based analysis and confirm that Low-pass filtering and Savitzky-Golay offer the best balance between clarity and structural fidelity, while methods like Wavelet denoising tend to over-smooth and obscure important features.

4.3 Classification-Based Transferability Analysis

Although time series synthesis through Bayesian modeling has been proven possible, assessing its limitations and shortcomings requires more than conventional evaluation and similarity metrics. Traditional statistical and distance-based measures often fail to capture deeper structural patterns [43]. In this section, we introduce a classification-based transferability framework, not to confirm the success of our Bayesian modeling approach, but to critically examine its shortcomings [44]. While the synthesized sequences mimic the structure of real samples, it's important to note that the original dataset represents a pseudo-time series, a structure where sequential ordering is inferred or imposed, rather than naturally occurring over time. Thus, what we examine here are limitations in preserving structural or shape based sequence features, rather than true temporal dynamics. Our framework analyzes whether classifiers can transfer learned patterns between real and synthetic domains, revealing whether critical structural signals are retained by the generative process.

The core idea behind this framework is that if synthetic data captures the key sequential patterns and structural features found in the original pseudo-time series, machine learning models should generalize well and keep consistent performance across domains, regardless of whether they were trained on real or synthetic data. The research carried out by Le et al. [45] even displayed improved robustness when synthetic data were used for training. The divergence from this expectation indicates where the synthetic data fails to replicate important aspects of the original distribution. In particular, when models trained on synthetic data perform poorly on real data (or vice versa), it highlights shortcomings in the generative process and guides further diagnosis of these limitations.

To gain deeper insight into these limitations, a varied set of classification models is employed, including Support Vector Machines (SVM), Random Forest (RF), XGBoost, k-Nearest Neighbors (k-NN) and Gaussian Process Classification (GPC). These algorithms, chosen for their diverse inductive biases, allow us to examine different aspects of model-data interaction.

We implement this framework using four evaluation scenarios:

- Real $\rightarrow$ Real: Training and testing on real data
- Synthetic $\rightarrow$ Synthetic: Training and testing on synthetic data
- Real $\rightarrow$ Synthetic: Training on real, testing on synthetic
- Synthetic $\rightarrow$ Real: Training on synthetic, testing on real

In particular, we investigate asymmetries across the cross-domain scenarios by comparing model performance across these four scenarios. The representational weaknesses of the synthesized data are shown by these discrepancies. Since these failures expose fundamental flaws in the generating process, we place more attention on where and how generalization fails.

4.3.1 Dataset Preparation

For these classification tasks, all five existing classes were used to provide a better foundation for understanding shortcomings and achieving consistent results. The dataset that contains real-time series has 377 samples, as mentioned in Section 3.1, and the same number of time series have been synthesized with respect to the class distribution of the original samples. The real and synthetic datasets were split into training (80%) and testing (20%) sets using stratified sampling, leaving 301 samples for training and 76 samples for testing. However, for GPC, we followed the same cross-domain evaluation approach but utilized the entire datasets without splitting, enabling this classifier to leverage all available samples during both training and evaluation phases.

4.3.2 Results and Analysis

Support Vector Machines

SVMs, with their capacity to define optimal separating hyperplanes, are especially sensitive to differences in class boundaries and structural consistency [46]. When applied across the four evaluation scenarios, the SVM's behavior reveals systematic differences in how the synthesized pseudo-time series represent class-specific patterns and feature distributions compared to the original data.

Train → Test	Accuracy	Precision	Recall	F1-Score
Real → Real	0.447	0.341	0.447	0.383
Real → Synthetic	0.434	0.331	0.434	0.372
Synthetic → Real	0.434	0.329	0.434	0.371
Synthetic → Synthetic	0.434	0.329	0.434	0.371

Table 4.6: SVM classification performance across training and testing domain combinations. Metrics are macro-averaged.

The results in all scenarios reveal a consistent pattern. The real trained and tested scenario sets a baseline accuracy of approximately 43.4%, with the synthetic-involved scenarios showing remarkably similar performance metrics. This consistency across all domains suggests that both the original and synthetic pseudo-time series share fundamental characteristics regarding class separability. The SVM arrives at similar decision boundaries regardless of which domain (real or synthetic) is used for training or testing, demonstrating that our generative approach reproduces the challenging classification landscape of the original data.

Moreover, classes with a lower number of true samples, specifically class 1 (Kékfrankos) and class 4 (Tramini), are consistently misclassified., even in the real trained and tested baseline. This is due to SVM being a margin-based learner that is biased towards majority classes when minority classes lack representation [47].

Another notable observation is that classifiers trained on synthetic data generalize poorly to real data. Even though the generated sequences appear to be similar on the surface, their internal structure lacks the complexity and discriminative characteristics required to support accurate classification in practical settings. The classifier trained on synthetic data performs identically when tested on real or synthetic samples, reinforcing the idea that synthetic sequences, while statistically feasible, do not encode meaningful structural patterns across the sequence or class-specific information. A deeper limitation of the Bayesian generative process is revealed by this consistency across evaluation setups: it captures global distributional properties but does not embed class-conditional structure.

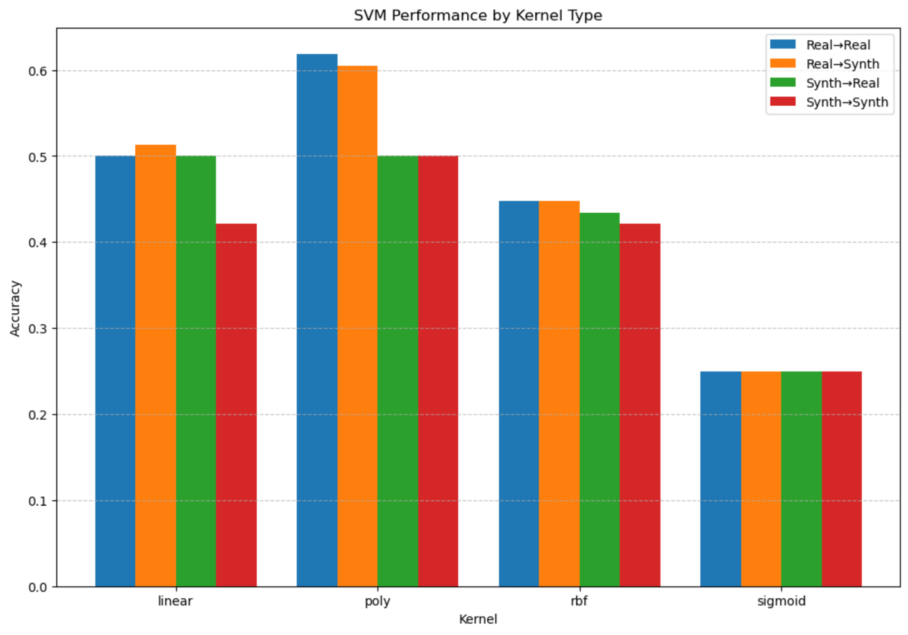


Figure 4.8: SVM Performance by kernel type

To further investigate these limitations, we conducted a kernel comparison experiment using linear, polynomial, RBF and sigmoid kernels [48]. The results show that the polynomial kernel, which helps capture higher-order relationships, performs best with real data, achieving an accuracy of around 62%. However, when trained on synthetic data, it only reaches about 50% accuracy on real data, suggesting that the synthetic data is missing important higher-order patterns present in the real sequences. Additionally, all kernels show lower performance when synthetic data is involved, indicating that our Bayesian model tends to produce simplified versions of the time series that lack certain discriminative features. More importantly, these simplified versions of the synthesized time series exhibit internal consistency but do not align with real-world patterns. Consistent performance drops across kernels also support the idea that this is a fundamental limitation, rather than an outcome of any specific analysis method.

In conclusion, the classifier results highlight significant limitations in the synthesized time series. Specifically, the sequences lack class-specific structural features and higher-order positional relationships that appear to play a role in the real dataset's classification landscape. Although the Bayesian model is capable of generating sequences that are statistically plausible and internally consistent, this does not ensure that they capture the structured variations necessary for real-world applicability.

Random Forest

The Random Forest classifier offers a different perspective on the structural integrity of the synthesized time series. In contrast to the SVM's uniformly poor performance across synthetic scenarios, RF demonstrates variation in classification effectiveness depending on the training-testing configuration, which offers a richer window into the underlying dynamics of the data. Unlike margin-based classifiers such as SVM, RF builds an ensemble of decision trees that recursively split the input space based on feature thresholds [49], making it particularly effective at capturing non-linear relationships and localized patterns, even in high variance or noisy data, which is mostly the case for our synthesized time series.

Train → Test	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Real → Real	0.566	0.458	0.480	0.460
Real → Synthetic	0.539	0.438	0.460	0.443
Synthetic → Real	0.500	0.400	0.428	0.411
Synthetic → Synthetic	0.487	0.397	0.412	0.399

Table 4.7: Random Forest classification performance across training and testing domain combinations. Metrics are macro-averaged

The highest performance, as expected, has been achieved in the baseline scenario where RF reaches the accuracy of 56.6%. Just like other classifiers, this serves as an upper bound in the current setup. This result was closely followed by real trained on synthetic test (53.9%) then synthetic trained on real test (50.0%) and finally synthetic trained on synthetic test (48.7%). This ranking highlights that RF, even when faced with the limitations of synthetic data, is still able to extract and exploit some useful structural features. Notably, classes 2 (Sárgamuskotály) and 3 (Szürkebarát) are relatively well classified, while class 4 (Tramini) is consistently misclassified even in real trained and tested scenarios. This outcome aligns well with RF's ensemble approach, which tends to favor well-represented classes unless explicitly balanced.

Although the synthetic sequences appear statistically reasonable, even visually similar to real ones when noise and fluctuations are disregarded, the classification results reveal that they fail to encode meaningful class distinctions. This conclusion is reinforced by the consistently low and flat precision, recall, and F1-scores across all scenarios involving synthetic data. More fundamentally, these results suggest that the generated samples lack coherent sequence-level structure: the Bayesian model, as implemented, primarily captures local correlations between adjacent time points, but fails to model higher-order relationships that span larger portions of the sequence. Since class distinctions in this dataset appear to emerge from broader shape patterns and positional progression across the sequence, this structural gap critically undermines the practical utility of the synthetic data for downstream classification tasks.

Another observable pattern, as also seen in SVM results, is that while the synthesized time series demonstrate internal consistency, they remain externally incoherent. Models trained and tested solely on synthetic data perform moderately well, but this performance does not translate when applied to real data. This inconsistency highlights a significant risk when models trained on synthetic samples may fail to generalize or perform reliably in real-world applications.

To enable classification, each pseudo-time series was represented as a fixed-length vector by resampling all sequences to a uniform length. This transformation yielded a two-dimensional dataset, where each row corresponds to a sample and each column represents a specific position in the sequence. These positional indices were treated as input features, allowing standard classifiers such as RF to be applied directly.

It is important to note that, unlike conventional tabular features, these columns do not represent independent variables. Instead, they correspond to fixed positions within an ordered, pseudo-temporal structure. While this approach does not preserve or model dependencies between adjacent positions, it facilitates interpretable feature importance analysis, where each feature's contribution reflects its relative influence on the classifier's decisions.

Real Data			**Synthetic Data**		
Rank	**Sequence Position**	**Importance**	**Rank**	**Sequence Position**	**Importance**
1	8	0.0114	1	8	0.0066
2	6	0.0113	2	33	0.0048
3	7	0.0083	3	20	0.0046
4	5	0.0074	4	44	0.0045
5	3	0.0069	5	593	0.0044
6	26	0.0065	6	17	0.0042
7	598	0.0065	7	346	0.0042
8	4	0.0064	8	335	0.0042
9	9	0.0062	9	329	0.0042
10	597	0.0062	10	561	0.0041

Table 4.8: Top 10 most influential sequence positions for Random Forest classification, based on feature importance scores. Each "sequence position" refers to an index in the standardized pseudo-time series input. These do not represent actual timestamps but positional features after resampling. The table compares the most informative positions in real vs. synthetic data and shows that the classifier relies on different regions of the sequence in each domain.

As shown in Table 4.8, we analyzed the top 10 most important sequence positions for classification using RF models trained on real and synthetic data. Each “sequence position” refers to an index in the standardized, fixed-length pseudo-time series, not to a timestamp or temporally meaningful point. Because our model treats each position as a separate input feature, the importance scores reflect how much each part of the sequence contributes to the classification decision.

This analysis provides insight into how class-discriminative information is distributed. In the real data, 7 out of the 10 most important features are located within the first 10 sequence positions, suggesting that class-relevant patterns are concentrated toward the start of the sequence. In contrast, synthetic data shows a much more dispersed pattern of importance across the sequence (e.g., positions 8, 17, 20, 33, 593), with minimal overlap between the domains. This structural mismatch further confirms that while the synthetic sequences may appear plausible, they do not preserve the localized, class-relevant structure found in the real data.

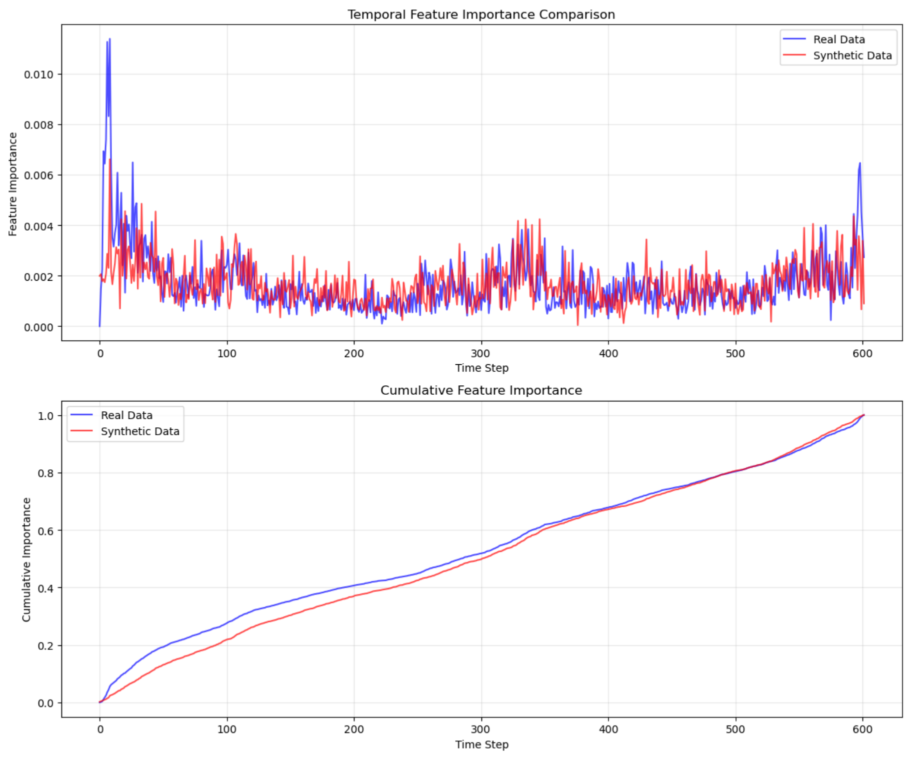


Figure 4.9: Temporal and cumulative feature importance

The figure 4.9 compares how important different sequence positions are for classification when using RF, trained separately on real and synthetic data. Importantly, each position in the sequence represents a fixed index in the resampled pseudo-time series. These positions function more like ordered feature slots in a vector, rather than elements of a time-aware structure.

In the top plot, we see the feature importance assigned to each sequence position. For the real data (blue line), a significant amount of importance is concentrated in the early part of the sequence—especially within the first 50 to 100 positions. This suggests that the classifier rely heavily on specific parts of the sequence when distinguishing between classes. On the other hand, the synthetic data (red line) shows a much more uniform and dispersed importance pattern, without strong peaks. This indicates that synthetic sequences do not exhibit the same kind of structured, class-relevant variation across positions.

The bottom plot shows cumulative feature importance, which tells us how quickly the model accumulates predictive signal as we move through the sequence from start to end. For real data, importance accumulates faster at the beginning, confirming that informative features are concentrated in early positions. In contrast, the synthetic data builds up more slowly, suggesting that useful information is spread out more evenly or simply less concentrated overall.

In summary, the RF results show that although the synthesized sequences appear statistically plausible, they lack the positional structure necessary for effective classification. RF was able to extract some useful patterns, but the poor generalization between real and synthetic domains suggests that the generated sequences do not preserve the positional patterns that distinguish between classes. This highlights a key limitation in the current generative approach: it captures local characteristics of the data but fails to reproduce the broader distributional structure that informs class separability.

XGBoost

XGBoost is a gradient boosting algorithm that builds an ensemble of decision trees in a sequential way. It is particularly effective at capturing complex feature interactions and non-linear relationships, which makes it useful for datasets with subtle patterns. However, its tendency to overfit on noise or overly specific patterns can lead to inconsistent results, especially when there are differences between training and testing distributions [50].

Train → Test	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Real → Real	0.566	0.460	0.489	0.465
Real → Synthetic	0.408	0.437	0.377	0.386
Synthetic → Real	0.553	0.636	0.495	0.509
Synthetic → Synthetic	0.513	0.407	0.431	0.417

Table 4.9: XGBoost classification performance across training and testing domain combinations. Metrics are macro-averaged

As expected, the real trained and tested scenario serves as a baseline, achieving 56.6% accuracy and a relatively balanced F1 score among well-represented classes. However, even in this scenario, class 4 (Tramini) suffers from zero recall, consistent with what was observed with previous classifiers. However, when XGBoost is trained on synthetic data and tested on real data, it performs better (55.3%) than the real-trained and synthetic-tested scenario (40.8%) when the opposite is commonly expected, catching up to the baseline performance. This asymmetry can be attributed to differences in the structural patterns and sequential consistency between the synthetic and real datasets.

Class	SHAP Correlation
0	0.1739
1	0.1402
2	0.1525
3	0.1616
4	0.1537

Table 4.10: SHAP Value Correlation Analysis (Real vs Synthetic)

To further investigate this claim, SHAP analysis has been employed, inspired by the work of Wang et al. [51]. Despite a very high overall correlation (0.995) in the feature importance across sequence positions between models trained on real and synthetic data, the class-specific SHAP correlations were substantially lower (ranging from 0.14 to 0.17). This indicates that while the synthetic data broadly mimics the overall distribution of positional relevance, it fails to preserve the nuanced, class-specific patterns needed for reliable classification. In other words, the generative model captures general trends across the sequence but distorts or overlooks the specific patterns that differentiate individual classes.

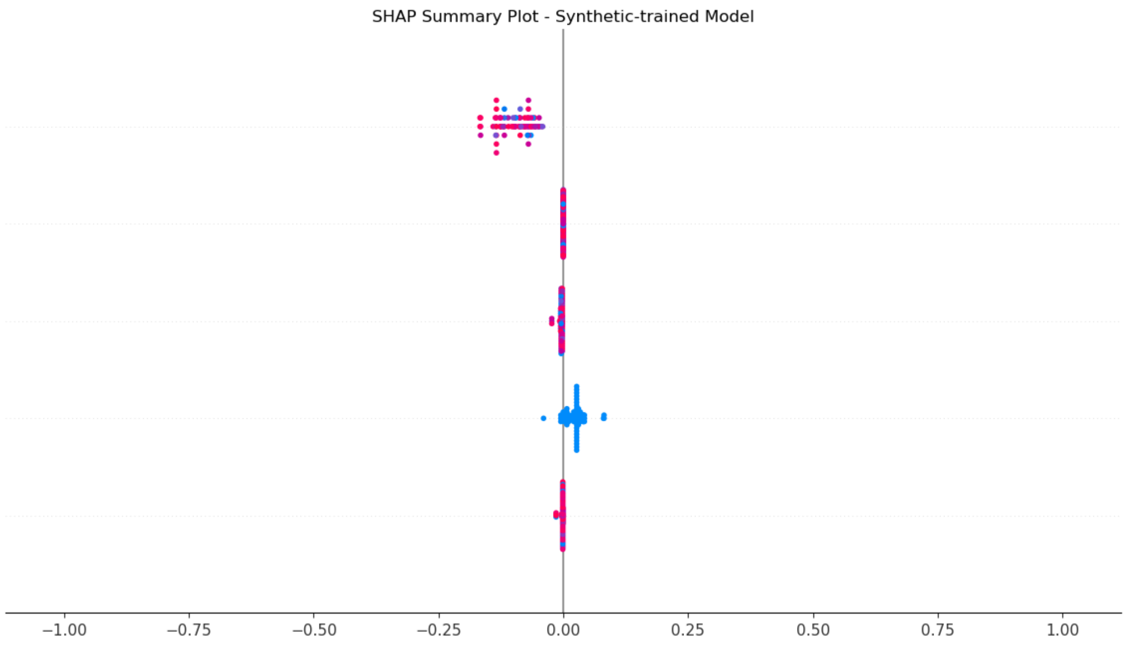


Figure 4.10: SHAP Summary Plot of synthetic trained model

The SHAP summary plot (Figure 4.10) further exposes a negative attribution bias in the synthetic model, suggesting it identifies patterns more by their absence than presence, a distribution shift difficult to detect with standard metrics.

While the Bayesian generative process captures broad structural similarities and class-defining features, it introduces additional variability not present in the real samples. These fluctuations weaken the integrity of temporal patterns that real-trained models have learned to expect from cleaner, more stable real sequences. This explanation is consistent with the known limitations of Bayesian approaches in preserving the exact autocorrelation structures of time series data.

Another important observation comes from the synthetic-trained, synthetic-tested configuration, where XGBoost achieves 51.3% accuracy. While this is lower than the baseline (real-trained, real-tested), it supports findings from the RF experiments: the synthetic data exhibits internal consistency and forms its own separable distribution. XGBoost is able to exploit recurring statistical patterns within this synthetic domain, even if those patterns diverge from those found in real-world data. This reinforces the idea that synthetic data enables learning within its own distribution but does not support generalization across domains.

In conclusion, the XGBoost results, together with SHAP analysis, highlight a deeper issue with the current generative approach. While the model produces sequences that are internally coherent and statistically structured, it fails to replicate the class-specific feature dependencies necessary for robust classification on real data. This results in synthetic sequences that appear plausible and consistent but ultimately lack meaningful alignment with real-world label structures, explaining their limited transferability.

K-Nearest Neighbors

The performance of the k-NN classifier across different training and testing domain combinations reveals a distinct and insightful behavior compared to more complex models like SVM, RF, and gradient boosting-based algorithms such as XGBoost. While those models aim to learn complex patterns, uncover non-linear decision boundaries, and sometimes exploit temporal dependencies within the features, k-NN takes a much more straightforward, statistical approach to classification.

Unlike parametric models, k-NN is a non-parametric, instance-based learner. It makes predictions based solely on the geometric proximity of a test sample to its neighbors in the training data, without learning any internal model or structure. As such, it does not learn in the traditional sense, nor does it model long-term dependencies, which are the elements that might be crucial in time series classification. The performance of k-NN across the different domain pairings is presented in Table 4.11.

Train → Test	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Real → Real	0.474	0.580	0.437	0.449
Real → Synthetic	0.474	0.579	0.437	0.450
Synthetic → Real	0.461	0.565	0.426	0.441
Synthetic → Synthetic	0.461	0.569	0.426	0.438

Table 4.11: k-NN classification performance across training and testing domain combinations. All metrics are macro-averaged.

The k-NN classifier achieves relatively low accuracy across all training and testing domain combinations, with performance ranging between 46–47%. This modest accuracy suggests that the classification task is challenging regardless of whether the data is real or synthetic, indicating that strong class-discriminative patterns are not prominent in the feature space.

Furthermore, there is a striking consistency between in-domain and cross-domain performances. Training on real data and testing on synthetic data yields almost identical accuracy (47.4%) and macro-averaged metrics (precision, recall, F1-score) compared to baseline scenario. Similarly, training on synthetic data results in very similar performance whether tested on real data (46.1%) or synthetic data (46.1%).

This symmetry between domain combinations suggests two key insights: the synthetic pseudo-time series successfully replicate the feature distributions of the real data, and the differences between real and synthetic domains are negligible from the perspective of a distance-based classifier like k-NN. In other words, synthetic data is statistically very close to real data at the feature space level for this classification task. However, it is important to note that k-NN is a non-parametric, instance-based learner that relies solely on geometric proximity in the feature space and does not model any internal structure. Therefore, while the synthetic data appears similar to the real data in terms of feature distributions, these results cannot reveal deeper structural shortcomings. Given that both real and synthetic pseudo-time series likely lack meaningful temporal dynamics, the synthetic data effectively matches the real data's nature but within a limited scope.

Paper published by Mahato et al. [52] explored the approach of using k-NN with DTW in addition to other techniques. Therefore, to investigate and explain the source of our findings, DTW distance matrices have been utilized across domains.

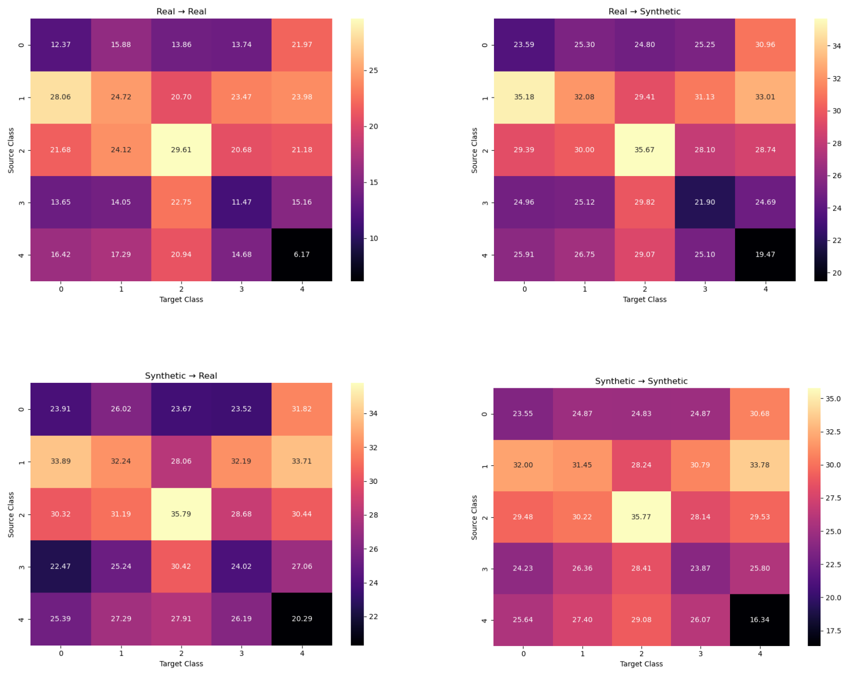


Figure 4.11: DTW distance matrices analysis

Analysis of these DTW matrices reveals that real time series exhibit substantially lower intra-class and inter-class distances compared to synthetic data, indicating that real samples are more tightly clustered around their respective class centers. In contrast, synthetic sequences display higher and more uniform distance values across classes, both when compared within the synthetic domain and between synthetic and real domains. This more homogeneous distance distribution for synthetic data introduces a regularization effect that benefits k-NN classification. Specifically, the synthetic samples create a smoother decision landscape, where class boundaries are less affected by local noise or outlier behavior, which often occurs in real data. This phenomenon is particularly visible for challenging classes like class 2 (Sárgamuskotály), which continues to demonstrate higher and more diffuse DTW distances across all settings.

However, this performance increase can't be interpreted as Bayesian modeling is able to generate sequences that is very close or even superior than the real data. Instead, the variability and noise introduced by the generative process act as an implicit form of data augmentation, simplifying the classification problem for distance-based models like k-NN. While this effect may benefit simple classifiers, it does not preserve the more complex positional or class-specific structures that more sophisticated models rely upon. Thus, synthetic data provides an advantage for specific classifiers but reflects a trade-off between structural fidelity and robustness to noise.

Gaussian Process Classification

The Gaussian Process Classifier (GPC) provides a distinctive perspective on the synthetic time series through its probabilistic, kernel-based approach to classification. Unlike the deterministic boundaries of SVM or the recursive partitioning of tree-based methods, GPC models the probability distribution of each class across the feature space, offering insights into how global statistical structures and class-specific patterns are preserved in the synthetic sequences.

Train → Test	Accuracy	Precision	Recall	F1-Score
Real → Real	0.660	0.562	0.546	0.522
Real → Synthetic	0.653	0.559	0.539	0.514
Synthetic → Real	0.623	0.532	0.515	0.492
Synthetic → Synthetic	0.684	0.583	0.565	0.539

Table 4.12: Gaussian Process Classifier performance across training and testing domain combinations. Metrics are macro-averaged.

The results align with expectations given the nature of the data and the classifier. Among the domain combinations, GPC achieves its highest accuracy (68.4%) when both training and testing on synthetic data, slightly exceeding its performance on real trained and tested scenario (66.0%). This is consistent with the fact that both the synthetic data and GPC share a probabilistic foundation: the synthetic sequences, generated through a stochastic process, naturally create data distributions that are well-suited to GPC's probabilistic decision boundaries. Furthermore, as discussed in Section 4, the synthetic samples displays greater noise and fluctuation compared to the real sequences, further aligning with GPC's reliance on modeling covariance structures and statistical relationships across the input space.

Examining class-specific metrics reveals similar patterns observed with other classifiers. Classes 0 (Cabernet Franc), 2 (Sárgamuskotály), and 3 (Szürkebarát) are classified with reasonable effectiveness, while class 4 (Tramini) consistently shows zero precision and recall across all scenarios.

In addition, the performance gap observed in cross-domain testing supports similar trends seen with other classifiers. The drop in accuracy when training on synthetic data and testing on real data, followed by a higher accuracy when both training and testing on synthetic data, suggests that the synthetic sequences exhibit internal consistency that does not fully align with the distributional properties of the real dataset. Specifically, the real trained synthetic tested scenario achieves accuracy (65.3%), exceeding accuracy of synthetic trained real tested scenario (62.3%), implies that models trained on real data learn more generalizable patterns, while models trained on synthetic data may introduce simplified structures during generation. This indicates a potential domain shift between real and synthetic data that limits transferability.

To deepen our understanding of GPC's behavior with synthetic time series, we conducted an uncertainty analysis that examines the probabilistic aspects of classification decisions. Gaussian processes inherently model uncertainty, making them particularly valuable for evaluating how well our Bayesian generative approach preserves the probabilistic structure of the original data.

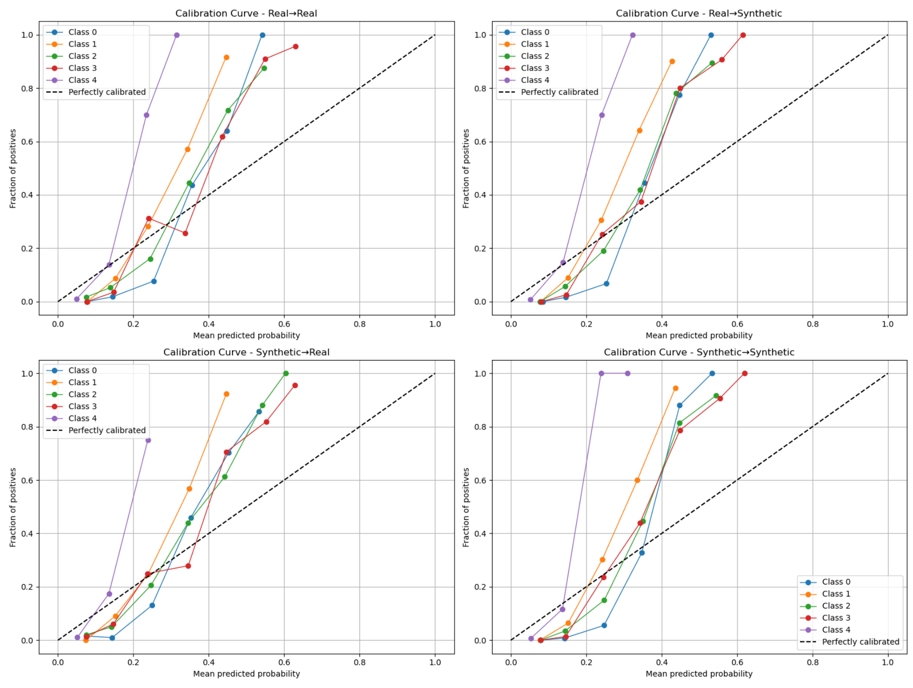


Figure 4.12: GPC calibration curve for all domains

Figure 4.12 presents calibration curves for all combinations of domains, revealing significant miscalibration patterns between scenarios. The divergence from the perfectly calibrated line indicates that GPC consistently overestimates its confidence. This is especially evident in the synthetic trained and tested scenario where Expected Calibration Error (ECE) reaches 0.288, the highest among all combinations despite achieving the best accuracy, visible in Table 4.13. This is a particularly interesting result, as typically better classification performance correlates with better calibration.

Scenario	Mean Confidence	Mean Uncertainty	Entropy	ECE
Real→Real	0.437	0.563	1.368	0.229
Real→Synthetic	0.414	0.586	1.403	0.253
Synthetic→Real	0.433	0.567	1.372	0.204
Synthetic→Synthetic	0.430	0.570	1.382	0.288

Table 4.13: Uncertainty metrics across training and testing domain combinations.

This contradictory observation suggests that synthetic data creates patterns that make the classification decisions easier; however, these patterns do not preserve the proper uncertainty of the real data. This leads to a classifier that is confidently making predictions, achieving high accuracy, but the confidence levels themselves are poorly calibrated, supported by a high ECE score.

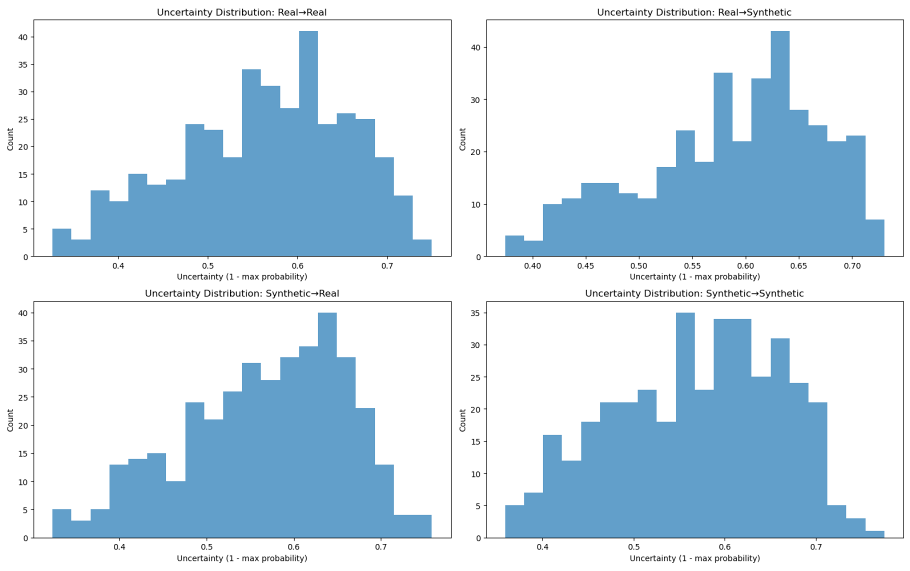


Figure 4.13: GPC uncertainty distribution for all domains

The phenomenon is further explained by the uncertainty distributions (Figure 4.13). A real-trained and synthetic-tested scenario demonstrates higher uncertainty (0.586) and entropy (1.403), revealing that the classifier faces greater trouble estimating probability when it is exposed to synthetic data after training on real samples. This suggests our generative approach preserves enough class patterns for basic classification but distorts the finer probabilistic relationships that GPC relies on for accurate uncertainty estimates.

This probabilistic analysis extends our understanding of the previously observed performance inconsistencies. The higher accuracy in synthetic trained and tested scenarios, in addition to poorer calibration, points to time series with internal consistency accompanied by systematic biases. The consistent calibration differences across all scenarios further emphasize that statistical fidelity alone is insufficient for generating functionally equal synthetic time series.

Summary

Overall, the results across all five classifiers provide a different perspective of the strengths and weaknesses in the synthesized pseudo-time series compared to evaluation metrics. The SVM results show consistently low and flat performance across all scenarios, indicating that neither the real nor synthetic datasets offer strong class separability for this classifier. However, the addition of the kernel comparison experiment reveals that the Bayesian model generates simplified versions of the real data that lack high-order patterns.

Random Forest offers slightly more flexible decision-making and shows modest differences across domain combinations. However, its performance drops drastically when tested on synthetic data, particularly in the synthetic-trained setting. Feature importance analysis further reveals that the model relies on different sequence regions depending on the data domain, indicating that the synthetic sequences fail to replicate the positional structure that distinguishes classes in real data.

XGBoost yields an unexpected result by achieving higher accuracy in the synthetic trained real tested scenario than in the real trained synthetic tested one. This behavior suggests that the synthetic data may introduce simplified structures or regularized patterns that align better with real data than vice versa. However, SHAP analysis reveals that this performance does not arise from accurate class-specific alignment but from the model leveraging generalized patterns that do not reflect the class semantics of the real sequences.

The k-NN classifier showed a stable level of performance across all training and testing combinations, consistently between the range of 46–47% accuracy. This uniformity suggests that the synthetic data shares a similar statistical profile to the real dataset. However, analysis using DTW distance matrices revealed that the synthetic sequences tend to have greater variability and exhibit more uniform distances within and between classes. Instead of reducing noise, the generative model introduces additional fluctuations, which appear to smooth the decision boundaries in a way that benefits distance-based methods like k-NN. At the same time, this increased noise disrupts the more structured, class-specific patterns that complex classifiers rely on to make accurate distinctions.

The Gaussian Process Classifier (GPC) achieves the best accuracy when trained and tested on synthetic data, outperforming even the baseline scenario. This suggests that the synthetic data displays strong internal consistency that aligns well with the probabilistic nature of GPC. However, calibration and uncertainty analysis shows that the classifier becomes overconfident and poorly calibrated on synthetic data, indicating that while classification is easier, the model's uncertainty estimates are untrustworthy.

In summary, while the Bayesian synthesis process effectively captures some global statistical characteristics of the real pseudo-time series and produces sequences with internal consistency, it fails to replicate the structured, class-specific signals necessary for robust classification. The synthetic data appears plausible but lacks alignment regarding positional feature importance and probabilistic behavior. As a result, models trained on synthetic data perform well only within that domain and generalize poorly to real-world samples. These findings highlight a key limitation of the current generative approach: statistical fidelity alone is insufficient for generating synthetic data that is functionally comparable to the real data in classification tasks.

Chapter 5

Discussion

This chapter discusses the results of the study in a broader context. It examines how well the proposed Bayesian approach performed in generating pseudo-time series data, highlights the strengths and limitations observed during the experiments, and compared to related work. The goal is not only to interpret the numerical results but also to understand their implications for the practical use of Bayesian models in time series synthesis. In addition, this chapter reflects on potential challenges and areas for improvement.

5.1 Summary of key findings

This study introduced a Bayesian approach for synthesizing pseudo-time series as an interpretable and computationally efficient alternative to GAN-based methods. The generated sequences successfully replicated important statistical characteristics of the original data while introducing additional variability and noise. While the raw synthetic data displayed noise and heavy fluctuations, applying postprocessing techniques such as Low-pass filtering and the Savitzky-Golay filter significantly improved the quality on evaluation metrics and achieved a visual appearance that closely resembled the real samples.

However, classification based evaluation revealed notable limitations in the synthesized data. Models such as k-NN showed consistent but modest performance across different domain combinations, suggesting that while the synthetic data captures the overall statistical distribution of the real data, it does not fully preserve class-distinctive patterns. GPC performed well within the synthetic domain but exhibited overconfidence and miscalibration, indicating that the generated data aligns well with the classifier's internal assumptions rather than with real-world variability. More advanced models such as Random Forest and XGBoost, supported by SHAP and feature importance analyses, further demonstrated that while the Bayesian model effectively learns global structure and variability, it does not explicitly capture class-discriminative features, reflecting of the model's generative nature and its focus on statistical reconstruction rather than supervised differentiation.

5.2 Interpretation and Implications

The promising results of the Bayesian approach can be explained with few factors. Bayesian models offer a framework where uncertainty is explicitly modeled with prior distributions. This allows better control over the generation process compared to GANs, whose training dynamics often lack interpretability and require large-scale data and extensive tuning. Additionally, the use of hierarchical priors and GRW enabled the model to capture both shared global patterns and distinctive trends across time series. This allows synthesized time series to mirror real data characteristics and contributes to the model's high fidelity in synthesis tasks. Further implementation of postprocessing techniques led to the generation of cleaner, better-performing, and realistic time series. This strategy was particularly effective in addressing the noise and fluctuation introduced by the Bayesian generation process.

5.3 Comparison with Related Work

Compared to GAN-based approaches such as those presented by Salmi and Seixas [9], Yadav et al. [20] and Karahoda [53] , synthesized time series that have been generated with Bayesian models produced inferior performance on similar key evaluation metrics. GANs, especially those optimized for multivariate or dynamic contexts, have demonstrated superior results in capturing complex patterns and generating highly realistic synthetic data.

Despite the performance difference, the Bayesian framework presents strong advantages. It offers a more interpretable and structured generative process, which can be especially valuable in low-data settings or domains where model transparency is critical, such as healthcare or scientific research. Furthermore, the model's simplicity and lower computational cost make it a practical alternative in scenarios where the resource demands of GANs are restrictive.

5.4 Limitations

Although the proposed Bayesian framework demonstrates promising results, especially in replicating the statistical properties of real pseudo-time series, several methodological limitations should be acknowledged.

The fixed-length resampling applied to sequences before modeling introduces potential distortion. While this step ensures compatibility with the Bayesian model, it may alter certain shape-related features or sequence-level characteristics, potentially suppressing subtle signals relevant for classification.

Also, smoothing techniques, though effective in improving evaluation metrics and visual similarity, they risk over-regularizing the data. This can lead to sequences that appear cleaner but lose some of the natural variability present in real samples by over-smoothing.

Although the classification results generally confirm the structural credibility of the synthetic pseudo-time series, they also reveal a key limitation: while models trained on synthetic data perform well within that domain, they tend to generalize less effectively to real data. This suggests that the generative process captures overall statistical patterns but does not fully preserve the class-specific relationships needed for robust cross-domain performance. Classifiers often adapt well to the synthetic distribution, which appears internally consistent yet diverges mildly from the real data's underlying class structure, highlighting a gap between statistical resemblance and functional equivalence.

5.5 Broader Applications and Future Outlook

The framework introduced in this research presents a compelling, interpretable and computationally efficient alternative to GANs for time series synthesis, especially in the case with limited data availability. These advantages are especially beneficial in industries such as healthcare or finance, where transparency and reliability are of utmost importance.

Moreover, the use of Bayesian models could facilitate integration with privacy-preserving methods, such as differential privacy, due to their inherent probabilistic nature. Future work may explore extending the framework to multivariate time series, irregular sampling, or incorporation of domain-specific priors to further improve synthesis quality. Evaluating the synthetic data's utility in specific real-world tasks, such as anomaly detection or forecasting, could also provide deeper insights into its practical value. We acknowledge that results focused primarily on well represented classes, and further work is needed to evaluate model stability and representativeness on sparse classes such as Tramini.

Chapter 6

Conclusion

In this study, Bayesian models were used to generate synthetic pseudo-time series data as an interpretable and efficient alternative to GAN-based approaches. The process included normalizing the dataset, building a generative model using hierarchical priors and GRW, and generating synthetic sequences from the posterior predictive distribution. After the generation step, several postprocessing techniques were applied to improve the quality and structure of the time series. The results were then evaluated using various statistical and shape-based metrics, along with a classification-based analysis.

The results showed that while the Bayesian model did not perform better than GAN-based methods in all areas, especially in metrics like MAE and RMSE, it still managed to capture key statistical patterns. In particular, it performed well on correlation-based metrics such as Spearman and SQL. Postprocessing methods like Savitzky-Golay filtering and low-pass filtering helped improve the results further by reducing noise and making the series more similar to the original ones.

The classification analysis helped uncover some of the limitations. While the statistical metrics showed high similarity, classifiers sometimes struggled to generalize between real and synthetic data, suggesting that some deeper patterns in the original data were not fully captured. These findings show that common metrics alone might not always be enough to evaluate synthetic time series, and that a broader evaluation strategy is needed.

Despite its limitations, Bayesian modeling offers several important advantages. It provides interpretability, operates efficiently with lower computational requirements, and performs well in low-data settings. These qualities make it particularly suitable for domains such as healthcare or privacy-sensitive applications, where transparency and data availability are critical concerns. With further refinement and the integration of class-specific or domain-aware priors, this approach has the potential to serve as a compelling alternative to more complex generative models.

Bibliography

[1] William W.S. Wei. "458Time Series Analysis". In: *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2: Statistical Analysis*. Oxford University Press, Mar. 2013. ISBN: 9780199934898. DOI: 10.1093/oxfordhb/9780199934898.013.0022. eprint: https://academic.oup.com/book/0/chapter/352586926/chapter-ag-pdf/44419347/book_41363_section_352586926.ag.pdf. URL: https://doi.org/10.1093/oxfordhb/9780199934898.013.0022.

[2] Emma Peeling and Allan Tucker. "Making Time: Pseudo Time-Series for the Temporal Analysis of Cross Section Data". In: *Advances in Intelligent Data Analysis VII*. Ed. by Michael R. Berthold, John Shawe-Taylor, and Nada Lavrač. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 184–194. ISBN: 978-3-540-74825-0.

[3] James Jordon et al. *Synthetic Data - what, why and how?* Tech. rep. The Royal Society, 2024. URL: file:///mnt/data/Synthetic_Data_Survey-24.pdf.

[4] Da Zhang, Ming Ma, and Likun Xia. "A comprehensive review on GANs for time-series signals". In: *Neural Computing and Applications* 34.5 (2022), pp. 3551–3571.

[5] Zinan Lin et al. "Using gans for sharing networked time series data: Challenges, initial promise, and open questions". In: *Proceedings of the ACM internet measurement conference*. 2020, pp. 464–483.

[6] Eoin Brophy et al. "Generative adversarial networks in time series: A systematic literature review". In: *ACM Computing Surveys* 55.10 (2023), pp. 1–31.

[7] Matthew J. Schneider and John M. Abowd. "A New Method for Protecting Interrelated Time Series with Bayesian Prior Distributions and Synthetic Data". In: *Journal of the Royal Statistical Society Series A: Statistics in Society*

178.4 (Jan. 2015), pp. 963–975. ISSN: 0964-1998. DOI: 10.1111/rssa.12100. eprint: https://academic.oup.com/jrsssa/article-pdf/178/4/963/49348278/jrsssa_178_4_963.pdf. URL: https://doi.org/10.1111/rssa.12100.

[8] Shen Liu et al. “2 - Classification methods”. In: *Computational and Statistical Methods for Analysing Big Data with Applications*. Ed. by Shen Liu et al. San Diego: Academic Press, 2016, pp. 7–28. ISBN: 978-0-12-803732-4. DOI: https://doi.org/10.1016/B978-0-12-803732-4.00002-7. URL: %5Curl%7Bhttps://www.sciencedirect.com/science/article/%20pii/B9780128037324000027%7D.

[9] Zakaria Salmi and José Luis Seixas Junior. “Data Augmentation for Pseudo-Time Series Using Generative Adversarial Networks”. In: *ITAT Conference 2023 - Information Technologies - Applications and Theory*. CEUR Workshop Proceedings. 2023, pp. 62–74. URL: https://ceur-ws.org/Vol-3498/paper5.pdf.

[10] Yuelei Zhang, Xiao Chang, and Xiaoping Liu. “Inference of gene regulatory networks using pseudo-time series data”. In: *Bioinformatics* 37.16 (Feb. 2021), pp. 2423–2431. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab099. eprint: https://academic.oup.com/bioinformatics/article-pdf/37/16/2423/50339492/btab099.pdf. URL: https://doi.org/10.1093/bioinformatics/btab099.

[11] M Sanchez-Castillo et al. “A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data”. In: *Bioinformatics* 34.6 (Sept. 2017), pp. 964–970. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx605. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/6/964/48914307/bioinformatics_34_6_964.pdf. URL: https://doi.org/10.1093/bioinformatics/btx605.

[12] Zehua Liu et al. “Reconstructing cell cycle pseudo time-series via single-cell transcriptome data”. en. In: *Nat. Commun.* 8.1 (June 2017).

[13] James Joyce. “Bayes’ Theorem”. In: *The Stanford Encyclopedia of Philosophy*. Fall 2021. Metaphysics Research Lab, Stanford University, 2021.

[14] Greg M Allenby and Peter E Rossi. "Hierarchical bayes models". In: *The handbook of marketing research: Uses, misuses, and future advances* (2006), pp. 418–440.

[15] Germain Forestier et al. "Generating Synthetic Time Series to Augment Sparse Datasets". In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, pp. 865–870. DOI: `10.1109/ICDM.2017.106`.

[16] Hoang Anh Dau et al. "The UCR Time Series Archive". In: *arXiv preprint arXiv:1810.07758* (2019). URL: `https://arxiv.org/pdf/1810.07758v2`.

[17] Brian Kenji Iwana and Seiichi Uchida. "An empirical survey of data augmentation for time series classification with neural networks". en. In: *PLoS One* 16.7 (July 2021), e0254841.

[18] Sylvia Frühwirth-Schnatter and Hedibert F. Lopes. "Bayesian Time Series Models and Computational Methods for Capturing Temporal Dependencies". In: *Statistical Science* 32.2 (2017), pp. 249–257.

[19] Athanasios Kottas and Peter Müller. "Bayesian Nonparametric Modeling of Multivariate Time Series with a Transformation Approach". In: *Statistica Sinica* 23.1 (2013), pp. 365–389.

[20] Parul Yadav et al. "Qualitative and Quantitative Evaluation of Multivariate Time-Series Synthetic Data Generated Using MTS-TGAN: A Novel Approach". In: *Applied Sciences* 13.7 (2023), p. 4136.

[21] Nazanin Fouladgar, Marjan Alirezaie, and Kary Främling. "Metrics and Evaluations of Time Series Explanations: An Application in Affect Computing". In: *IEEE Access* 10 (2022), pp. 23995–24009. DOI: `10.1109/ACCESS.2022.3155115`.

[22] Anne Driemel, Amer Krivosija, and Christian Sohler. "Clustering time series under the Fréchet distance". In: *CoRR* abs/1512.04349 (2015). arXiv: `1512.04349`. URL: `http://arxiv.org/abs/1512.04349`.

[23] Lei Wang and Piotr Koniusz. *Uncertainty-DTW for Time Series and Sequences*. 2022. arXiv: `2211.00005 [cs.CV]`. URL: `https://arxiv.org/abs/2211.00005`.

[24] Jianjun Li et al. "Human action recognition based on tensor shape descriptor". In: *IET Computer Vision* 10.8 (2016), pp. 905–911. DOI: https://doi.org/10.1049/iet-cvi.2016.0048. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-cvi.2016.0048. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-cvi.2016.0048.

[25] José Luis Seixas Junior and Tomáš Horváth. "KNN Algorithm with DTW Distance for Signature Classification of Vine Leaves". In: *ELTE 2020 - Department of Data Science and Engineering, Eötvös Loránd University, Faculty of Informatics*. 2020. URL: https://ceur-ws.org/Vol-2718/paper33.pdf.

[26] Neil Gershenfeld and Andreas Weigend. "The Future of Time Series: Learning and Understanding". In: Feb. 2018, pp. 349–429. ISBN: 9780429493362. DOI: 10.1201/9780429493362-15.

[27] Aistis Raudys, Vaidotas Lenčiauskas, and Edmundas Malčius. "Moving averages for financial data smoothing". In: *Information and Software Technologies: 19th International Conference, ICIST 2013, Kaunas, Lithuania, October 2013. Proceedings 19*. Springer. 2013, pp. 34–45.

[28] Volodymyr Lotysh, Larysa Gumeniuk, and Pavlo Humeniuk. "COMPARISON OF THE EFFECTIVENESS OF TIME SERIES ANALYSIS METHODS: SMA, WMA, EMA, EWMA, AND KALMAN FILTER FOR DATA ANALYSIS". In: *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska* 13.3 (Sept. 2023), pp. 71–74. DOI: 10.35784/iapgos.3652. URL: https://ph.pollub.pl/index.php/iapgos/article/view/3652.

[29] Ruyin Cao et al. "A simple method to improve the quality of NDVI time-series data by integrating spatiotemporal information with the Savitzky-Golay filter". In: *Remote Sensing of Environment* 217 (2018), pp. 244–257. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2018.08.022. URL: https://www.sciencedirect.com/science/article/pii/S0034425718303985.

[30] Jianbo Gao et al. "Denoising Nonlinear Time Series by Adaptive Filtering and Wavelet Shrinkage: A Comparison". In: *IEEE Signal Processing Letters* 17.3 (2010), pp. 237–240. DOI: 10.1109/LSP.2009.2037773.

[31] Ivan W. Selesnick et al. “Simultaneous Low-Pass Filtering and Total Variation Denoising”. In: *IEEE Transactions on Signal Processing* 62.5 (2014), pp. 1109–1124. DOI: 10.1109/TSP.2014.2298836.

[32] Michael Stenger et al. “Thinking in Categories: A Survey on Assessing the Quality for Time Series Synthesis”. In: *ACM Journal of Data and Information Quality* 16.2 (2024), pp. 1–32.

[33] Cort J. Willmott and Kenji Matsuura. “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance”. In: *Climate Research* 30.1 (2005), pp. 79–82. DOI: 10.3354/cr030079. URL: https://doi.org/10.3354/cr030079.

[34] Rob J. Hyndman and Anne B. Koehler. “Another look at measures of forecast accuracy”. In: *International Journal of Forecasting* 22.4 (2006), pp. 679–688. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001. URL: https://www.sciencedirect.com/science/article/pii/S0169207006000239.

[35] Rolf Werner, Dimitar Valev, and Dimitar Danov. “The Pearson’s correlation -a measure for the linear relationships between time series?” In: Sept. 2009.

[36] Jules J. Berman. “Chapter 4 - Understanding Your Data”. In: *Data Simplification*. Ed. by Jules J. Berman. Boston: Morgan Kaufmann, 2016, pp. 135–187. ISBN: 978-0-12-803781-2. DOI: https://doi.org/10.1016/B978-0-12-803781-2.00004-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128037812000047.

[37] Jiaqi Ye et al. “Time Series Similarity Evaluation Based on Spearman’s Correlation Coefficients and Distance Measures”. In: *Cloud Computing and Big Data*. Ed. by Weizhong Qiang, Xianghan Zheng, and Ching-Hsien Hsu. Cham: Springer International Publishing, 2015, pp. 319–331. ISBN: 978-3-319-28430-9.

[38] Julien Bodelet et al. *Statistical Quantile Learning for Large, Nonlinear, and Additive Latent Variable Models*. 2023. arXiv: 2003.13119 [stat.ME]. URL: https://arxiv.org/abs/2003.13119.

[39] Kishansingh Rajput, Duong Binh Nguyen, and Guoning Chen. *Evaluating DTW Measures via a Synthesis Framework for Time-Series Data*. 2024. arXiv: 2402.08943 [cs.LG]. URL: https://arxiv.org/abs/2402.08943.

[40] Richard Moeckel and Brad Murray. "Measuring the distance between time series". In: *Physica D: Nonlinear Phenomena* 102.3 (1997), pp. 187–194. ISSN: 0167-2789. DOI: https://doi.org/10.1016/S0167-2789(96)00154-6. URL: https://www.sciencedirect.com/science/article/pii/S0167278996001546.

[41] Minoru Tanaka. "Notes on Estimation of Cross Correlation Function between two Time Series". PhD thesis. Senshu University, 2019.

[42] Ozancan Özdemir. "Performance Comparison of Machine Learning Methods and Traditional Time Series Methods for Forecasting". A thesis submitted to the Graduate School of Natural and Applied Sciences in partial fulfillment of the requirements for the degree of Master of Science in Statistics. Master's thesis. Ankara, Turkey: Middle East Technical University, Aug. 2020.

[43] Shuai Xiao et al. "Learning Time Series Associated Event Sequences With Recurrent Point Process Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.10 (2019), pp. 3124–3136. DOI: 10.1109/TNNLS.2018.2889776.

[44] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. *Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs*. 2017. arXiv: 1706.02633 [stat.ML]. URL: https://arxiv.org/abs/1706.02633.

[45] Tuan Anh Le et al. "Using synthetic data to train neural networks is model-based reasoning". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 3514–3521. DOI: 10.1109/IJCNN.2017.7966298.

[46] Richard Brereton and Gavin Lloyd. "Support Vector Machines for classification and regression". In: *The Analyst* 135 (Feb. 2010), pp. 230–67. DOI: 10.1039/b918972f.

[47] Rukshan Batuwita and Vasile Palade. "Class Imbalance Learning Methods for Support Vector Machines". In: vol. 83. June 2013, pp. 83–99. ISBN: 9781118074626. DOI: 10.1002/9781118646106.ch5.

[48] Muhammad Hussain et al. “A Comparison of SVM Kernel Functions for Breast Cancer Detection”. In: *2011 Eighth International Conference Computer Graphics, Imaging and Visualization*. 2011, pp. 145–150. DOI: 10.1109/CGIV.2011.31.

[49] Pengyu Hao et al. “Feature Selection of Time Series MODIS Data for Early Crop Classification Using Random Forest: A Case Study in Kansas, USA”. In: *Remote Sensing* 7.5 (2015), pp. 5347–5369. ISSN: 2072-4292. DOI: 10.3390/rs70505347. URL: https://www.mdpi.com/2072-4292/7/5/5347.

[50] Zheng-gang Fang et al. “Application of a data-driven XGBoost model for the prediction of COVID-19 in the USA: a time-series study”. In: *BMJ Open* 12.7 (2022). ISSN: 2044-6055. DOI: 10.1136/bmjopen-2021-056685. eprint: https://bmjopen.bmj.com/content/12/7/e056685.full.pdf. URL: https://bmjopen.bmj.com/content/12/7/e056685.

[51] Mo Wang et al. “An XGBoost-SHAP approach to quantifying morphological impact on urban flooding susceptibility”. In: *Ecological Indicators* 156 (2023), p. 111137. ISSN: 1470-160X. DOI: https://doi.org/10.1016/j.ecolind.2023.111137. URL: https://www.sciencedirect.com/science/article/pii/S1470160X23012797.

[52] Vivek Mahato, Martin O’Reilly, and Padraig Cunningham. “A Comparison of k-NN Methods for Time Series Classification and Regression”. In: Dec. 2018.

[53] Bertan Karahoda. “Generating Time Series Data With Real-Valued DC-GAN From Complex Time-Frequency Domain: Application to ECG Synthesis”. In: *IEEE Access* 12 (2024), pp. 143215–143225. DOI: 10.1109/ACCESS.2024.3469541.