Department of Media
Fachhochschule Kiel

Master Thesis

Foundation Models for Time Series Forecasting with Exogenous Variables

A Survey and Benchmark

submitted in fulfilment of the requirements for the degree of
Master of Science - Data Science

written by
Martin Wild
student number: 942659

first examiner:	Prof. Dr. Stephan Doerfel
second examiner:	Dr. Thomas Christ

June 9, 2025

Foundation Models for Time Series Forecasting with Exogenous Variables

A Survey and Benchmark

Abstract

Foundation Models (FMs) are emerging as a powerful new paradigm for Time Series Forecasting (TSF), offering the promise of strong generalization capabilities with minimal task-specific engineering. However, while the use of *exogenous variables* is critical for achieving high accuracy in many real-world forecasting scenarios, their integration into FMs remains a nascent and fragmented research area. This thesis addresses this significant research gap by providing the first systematic overview and comprehensive empirical benchmark of FMs that incorporate exogenous variables. The study follows a two-part methodology: first, a structured literature survey is conducted to identify and categorize the diverse architectural approaches into two dominant paradigms: *Modular / Staged Integration* and *Unified / Deep Integration*. Second, a novel empirical benchmark is executed, comparing selected FMs against established baseline models across a wide range of exogenous variable scenarios.

The key findings reveal that the impact of exogenous variables on FM performance is not universally positive but is highly dependent on both the *model's architecture* and the *specific characteristics of the dataset*. The modular approach by using an external regressor employed by the *Chronos FM* proved to be the most reliable strategy for leveraging external information. Conversely, the study demonstrates that for many FMs, an indiscriminate inclusion of exogenous variables can degrade performance, challenging the "more data is better" axiom and highlighting a deep sensitivity to configuration. The results also identify a critical trade-off: FMs offer significant advantages in computational speed and workflow simplicity but currently suffer from a profound lack of interpretability and model-specific issues.

Ultimately, this work concludes that for the practical application of FMs with *exogenous variables*, the field must likely move beyond a pure "zero-shot" approach toward a more nuanced "pre-train" and "fine-tune" paradigm. By providing a clear categorization of integration methods, a foundational empirical benchmark and actionable insights into model behavior, this thesis establishes a crucial baseline for future research in a dynamic field whose advancement is currently hampered by a scarcity of suitable public datasets.

Declaration

I, Martin Wild, certify that I have prepared the Master's thesis "Foundation Models for Time Series Forecasting with Exogenous Variables: A Survey and Benchmark" independently and without unauthorised external assistance and that I have specially marked all passages taken verbatim from other authors as well as the explanations of my work that are closely based on the ideas of other authors and have cited the corresponding sources.

For the creation of this thesis, AI-powered applications were used, including Gemini, ChatGPT, Copilot, Perplexity and DeepL Write. These applications were used as tools for research purposes, code completion, and as a writing assistant to improve the quality of the work and increase its scientific rigor. The conceptual content, scientific argumentation, selection and evaluation of sources, and final responsibility for all content remain solely with the author of this thesis. The use of AI-powered applications complied with the principles of good scientific practice. In particular, no automatically generated content was adopted directly or paraphrased without proper attribution.

This thesis has not yet been submitted to any examination authority.

Würzburg, 09.06.2025

City, Date

Martin Wild

Acknowledgements

I would first like to express my sincere gratitude to my first examiner, **Prof. Dr. Doerfel**. I am deeply thankful that he took on the role of supervising this thesis. I am especially grateful for his trust and confidence in my ability to work independently on this extensive topic and for his encouragement to pursue this important line of research beyond the scope of this Master's program.

My profound thanks also go to my second examiner and mentor, **Dr. Thomas Christ**. Over the past year, I embarked on this journey with very little knowledge of forecasting or Foundation Models and no prior research background. He managed to sparked a strong interest for this subject in me, consistently pushing me to grow beyond my own expectations, even when the work was most demanding. His invaluable mentorship, deep knowledge and unwavering availability to answer any question have been fundamental to my development and the success of this thesis.

This work would not have been possible without the extraordinary support of my colleagues at prognostica GmbH. I owe a special debt of gratitude to **Julian Boehm** and **Manuel Dörr**. They were my first point of contact for everything, providing weekly feedback, technical guidance, and the infrastructure necessary for the benchmark. Beyond their immense professional support, their understanding and empathy have meant a great deal to me. They have had the greatest influence on the conception and execution of this thesis.

I also wish to thank the entire team at **prognostica GmbH** for giving me the opportunity and providing the resources that formed the foundation of this project. The intellectually stimulating and collegial atmosphere made every conversation a learning experience. My thanks go to the entire team for their support and for fostering such a positive, collaborative, and inspiring work environment.

Finally, I must thank those whose support was of a more personal, yet no less critical, nature. To my work colleagues again, especially Thomas, Julian, and Manuel, thank you for being a constant source of support.

To my family: thank you to my mother, who likely worried about me and this thesis more than I did myself, and who was always there for me - often with a fresh meal when I needed it most. To my sister, Carolin, thank you for always being a listening ear and for being one of the few who truly understands. My thanks also go to my father, to Raphael and Paco.

To all my friends who have supported me, I am deeply grateful. To list everyone would be impossible, but I must give a special mention to Laura, Regina and Effy, who helped me through some truly difficult times. Lastly, to my flatmates, thank you for putting up with me and for all the supportive and insightful conversations along the way.

Contents

1. Introduction

1.1. Motivation and Problem Definition

Time Series Forecasting (TSF) represents a fundamental task with far-reaching significance in a variety of application domains [Chatfield 2005]. Forecasting future developments based on historical data is essential for operational planning, strategic decisions, and risk management in areas such as economics, finance, meteorology, energy consumption, and logistics [Danese and Kalchschmidt 2011].

The complexity of TSF arises not only from the inherent diversity of time series types, such as seasonal, cyclical and volatile series and their associated patterns but also from the potential influence of external factors. These factors can provide additional contextual information and are often referred to as **exogenous variables** (the primary term in this thesis for strictly external influences) or the broader term **covariates** (which can also include internal data). Examples include weather data, holidays, market indicators or information about underlying processes [Lima et al. 2025]. The inclusion of such exogenous variables can often significantly improve forecasting accuracy compared to a purely univariate perspective, which is based solely on the history of the target time series, and can lead to notable improvements in forecasting accuracy.

Historically, the field of TSF was long dominated by statistical models such as Autoregressive Integrated Moving Average (ARIMA) or exponential smoothing [De Gooijer and Hyndman 2006]. With the progress in the area of Machine Learning (ML) and Deep Learning (DL), the available methods have expanded to now include models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, or Transformer-based architectures [J. Kim et al. 2024]. Each of these models possesses specific properties, strengths and weaknesses and is more or less suitable for certain forecasting setups. The effective selection and application of these models often require both specific model knowledge and domain-specific expert knowledge about the time series to be forecasted, as well as the identification and preparation of relevant exogenous variables. This leads to the desire for more generalized models that can deliver robust results across different time series and use cases, ideally with reduced need for specialized knowledge.

More recently, so-called **Foundation Models (FMs)**, trained on large-scale datasets from various domains, have achieved impressive successes in areas such as Natural Language Processing (NLP) and Computer Vision (CV), particularly based on the Transformer architecture [Brown, Mann, Ryder, Subbiah, Kaplan, et al. 2020; Dosovitskiy et al. 2020; Vaswani et al. 2017]. The success of these models in capturing complex dependencies in sequential and structured data has sparked interest in transferring similar generalizing capabilities to the field of time series analysis and forecasting.

In recent years, various approaches to FMs in the context of TSF have been developed [Ansari et al. 2024; M. Chen et al. 2024; Das, Kong, Sen, et al. 2023;

Ekambaram, Jati, Dayama, et al. 2024; Woo et al. 2024]. Current benchmarks and survey studies investigate their performance, strengths and weaknesses compared to established State-of-the-Art (SOTA) methods [Aksu et al. 2024; Li et al. 2024; Qiu et al. 2024; Y. Wang, H. Wu, Dong, Yong Liu, et al. 2024]. These studies show that FMs have the potential to compete with SOTA methods and in some scenarios even surpass them.

A significant point observed in many of these early FM approaches for time series, is the initial focus on univariate forecasting scenarios. As mentioned earlier, the inclusion of additional context through exogenous variables is often of great importance for achieving improved forecasting accuracy. Although the integration of exogenous variables into FMs for time series is being discussed, this research area is still in its early stages of development [Arango et al. 2025; Lima et al. 2025]. Compared to univariate approaches, methods for integrating exogenous variables are not yet as firmly established. The comprehensive landscape is rapidly evolving due to the large number of new research publications emerging in a short period, reflecting the dynamic and rapidly evolving nature of the field (see Figure 1.1). The consideration of exogenous variables in the existing comprehensive benchmarks and overviews of FMs for time series has so far often played only a subordinate role. Therefore, there is a clear need for a structured overview of the current developments and the present state of research regarding the integration of exogenous variables into FMs for TSF.

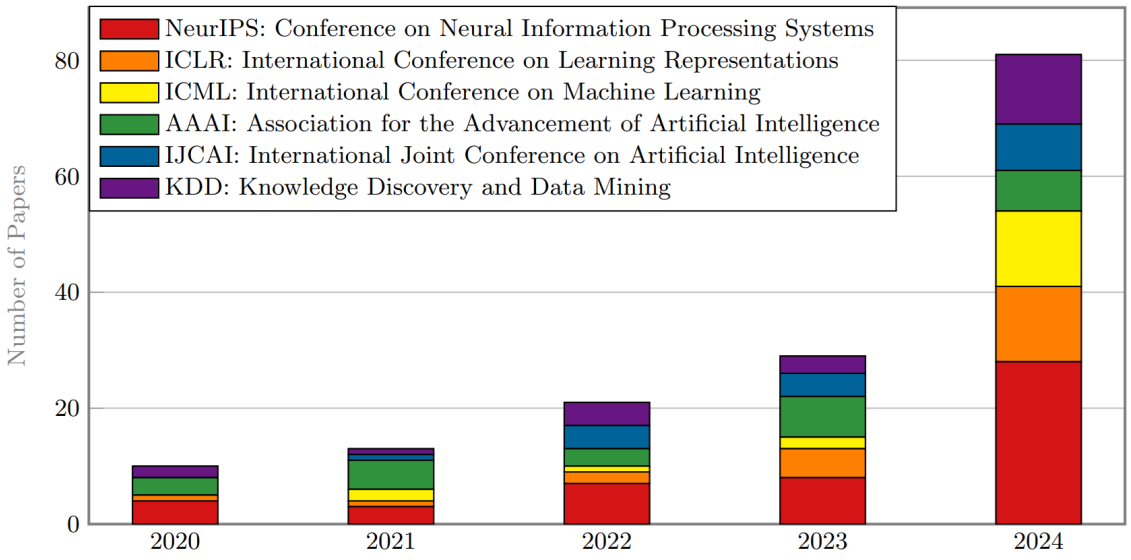


Figure 1.1.: Number of Top-tier AI and ML Conference Papers on TSF [J. Kim et al. 2024]

This work aims to address this specific research gap. It intends to provide a comprehensive overview of existing approaches for integrating exogenous variables into FMs for TSF. Furthermore, a comparative study and evaluation will be conducted to analyze how selected FMs that offer this functionality perform against each other and behave under various configurations of exogenous variables.

1.2. Research Questions

Based on the problem definition outlined in Section 1.1 and the identified research gap, the following central research questions arise, which are to be answered within the scope of this work:

F1: Which specific FMs or model approaches currently exist for TSF that enable the integration of exogenous variables, and how are they technically implemented?

F2: To what extent can the forecasting performance of FMs be improved by incorporating exogenous variables, and how does the forecasting performance of these models with exogenous variable integration compare to established, non-FMs that also utilize exogenous variables, on relevant datasets?

F3: How do different configurations of exogenous variables (e.g., regarding number, type, data form or preprocessing) influence the forecasting performance of the investigated FMs?

F4: What specific strengths and weaknesses do the identified approaches for exogenous variable integration in FMs exhibit, based on theoretical considerations and empirical results?

The following chapters are guided by these research questions, which frame both the investigation and the analysis.

1.3. Thesis Structure

The present Master's thesis is structured as follows to answer the formulated research questions and achieve the defined objectives:

Chapter 2: Provides an overview of the theoretical fundamentals of TSF, relevant classical and modern forecasting models (including common methods for handling exogenous variables), the concept of FMs, and the general importance of exogenous variables in TSF.

Chapter 3: Presents existing FMs developed for TSF that support exogenous variable integration. It discusses specific architectures and mechanisms proposed or used in these models for processing exogenous variables (partially addresses F1).

Chapter 4: Describes the experimental setup in detail, including the selection and characteristics of the used datasets, the selection and implementation of the models to be compared, the definition of the various configurations of exogenous variables (addresses F3), and the metrics used for evaluating forecasting accuracy.

Chapter 5: Presents and analyzes the results of the conducted comparative evaluation of the selected models under various experimental conditions and exogenous variable scenarios (addresses F2 and F3).

Chapter 6: Discusses the obtained results in detail within the context of the research questions, interpreting the findings and analyzing the strengths and weaknesses of the investigated approaches and models (F1, F2, F3, F4). It also addresses limitations and future directions.

Chapter 7: Summarizes the most important findings of the work and draws final conclusions.

2. Theoretical Background

2.1. Fundamentals of Time Series Forecasting

TSF is a scientific discipline dedicated to predicting future values by analyzing historically observed data [Cryer 1986]. In the context of rapidly increasing data availability, TSF has become a major analytical tool across diverse domains, empowering informed decision-making by revealing hidden patterns and anticipating future trends. This section establishes the foundational concepts of time series data and forecasting, delineating their diverse application landscapes, and examines the core characteristics that govern their behavior. A robust understanding of these fundamentals is paramount for appreciating both traditional modeling paradigms and the transformative potential of **FMs**, particularly when tackling the challenges posed by high-dimensional and heterogeneous exogenous variables.

2.1.1. Definition and Application Areas

A **time series** is fundamentally a sequence of data points collected at consistent, evenly spaced intervals over time and ordered chronologically. This inherent **temporal continuity** allows for the understanding and analysis of phenomena that evolve according to their time order. Each data point represents the state or value at a specific moment and through the observation of these data points, various patterns such as long-term trends, seasonality, cyclicality and irregularities can be recognized. More formally, a time series Y is a sequence of observations $\{y_t\}_{t=1}^{L}$, where y_t denotes the value(s) recorded at time point t. L is the total number of observations, representing the length of the series. This inherent sequential and ordered nature is what fundamentally distinguishes time series data from cross-sectional data, where observations are made at a single point in time across various subjects [Shumway and Stoffer 2017].

Time series forecasting (TSF) is the process of analyzing this historical, time-stamped data to make scientifically-grounded predictions about future values. This involves dissecting inherent temporal patterns and using past observations to estimate outcomes that are entirely unknown at the moment the forecast is generated. The forecast for a future time $T + h$, given all data available up to time T, is often denoted as $\hat{y}_{T+h|T}$ [Chatfield 2005]. While "forecasting" and "prediction" are often used interchangeably, "forecasting" typically refers to estimating values for specific future time points, usually within a defined future horizon. "Prediction" on the other hand, might concern future data more broadly, potentially without a strict temporal ordering or referring to interpolating missing values. Closely related is **time series analysis**, which aims to develop models to understand the underlying mechanisms and causes driving the observed data patterns, often serving as a prerequisite for effective forecasting. To illustrate the fundamental concept of TSF, Figure 2.1 displays the classic AirPassengers dataset [Box et al. 1976], which records

monthly international airline passengers from 1949 to 1961. The plot also includes an illustrative forecast extending beyond the observed data. This forecast predicts the passenger numbers for the subsequent two years and is distinctly marked in red.

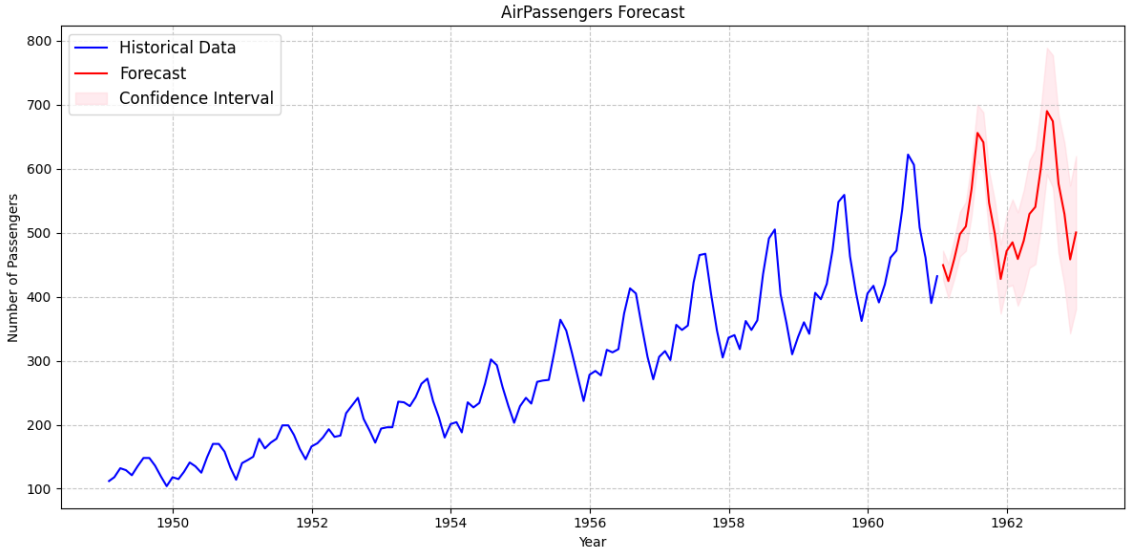


Figure 2.1.: The monthly international airline passengers dataset [Box et al. 1976] with an illustrative forecast. This visualization demonstrates the core aim of TSF: to predict future values based on observed historical patterns.

Time series can generally be classified into **univariate time series**, which involve observations of a single variable over time, and **multivariate time series**, which consist of observations of multiple, often interrelated, variables recorded simultaneously. The choice between these approaches depends on the complexity of the phenomenon being modeled and the availability of relevant influencing factors. The general forecasting task involves utilizing a **lookback window** of past data, spanning $p+1$ time steps from $t-p$ to t, to predict values over a future **forecast horizon** of H steps, from $t+1$ to $t+H$. This can be represented for a univariate series as $\hat{y}_{t+1:t+H} = f(y_{t-p:t})$ and for a multivariate series with D variables as $\hat{\mathbf{Y}}_{t+1:t+H} = f(\mathbf{Y}_{t-p:t})$, where $\mathbf{Y}$ denotes the vector of variables at each time step and D is the number of variables. Figure 2.2 visually illustrates this process for a multivariate time series with two variables, labeled as Channel A and Channel B. The forecasting model takes the data from the lookback window for both channels as input to predict their future values in the "Forecast Horizon", highlighted in yellow.

Another crucial characteristic of time series data is its **granularity** or **frequency**, which refers to the interval at which data points are collected. This can range from very high-frequency data, such as tick-by-tick stock market transactions or sensor readings in seconds, to lower frequencies like hourly, daily, weekly, monthly, quarterly or annually aggregated data, such as national economic indicators. The granularity directly influences the types of patterns that can be observed and modeled. For instance, daily data might reveal intra-week seasonality, while annual data would focus on multi-year trends or cycles. It also significantly impacts data volume and the choice of appropriate forecasting models.

The granularity of a time series often dictates the typical **forecast horizon** (H)

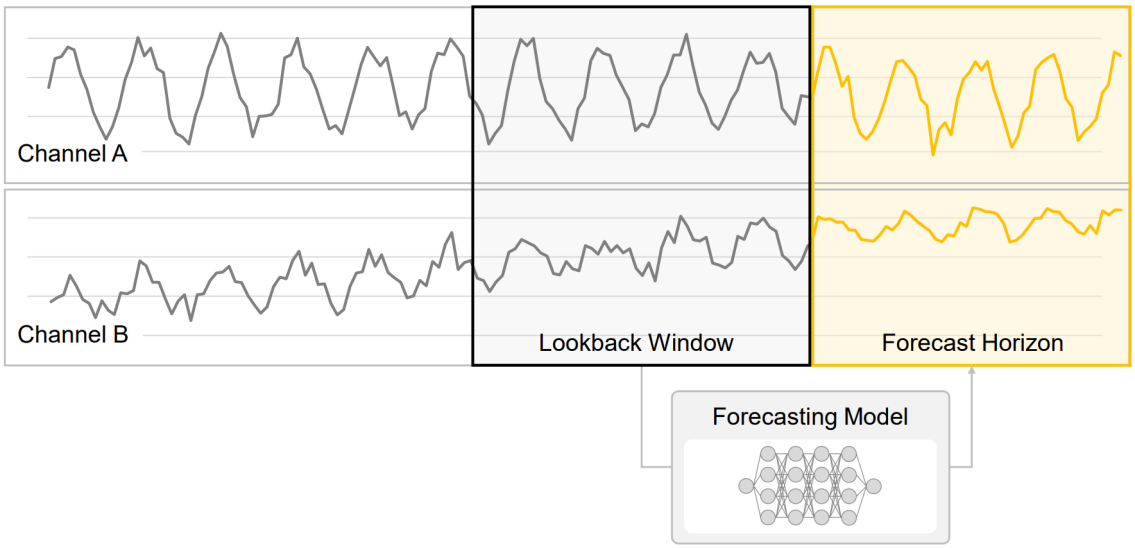


Figure 2.2.: Diagramm to illustrate multivariate forecasting [J. Kim et al. 2024].

and the distinction between short-term and long-term forecasting. **Short-term TSF** typically deals with predicting a few steps ahead relative to the data's frequency and is essential for immediate operational needs such as real-time anomaly detection or inventory adjustments. Conversely, **long-term TSF** aims to predict further into the future and supports strategic planning, policy-making, and capital investment decisions. The choice of an appropriate horizon H is thus closely tied to both the data's frequency and the specific goals of the forecasting task.

The scope of TSF has evolved considerably. Early approaches, such as Jan Tinbergen's econometric model in 1939, predominantly focused on the statistical extrapolation of past patterns within a single series [Tinbergen 1939]. However, the advent of **ML** and more recently, **DL** has significantly broadened this scope. Modern TSF increasingly incorporates these advanced methodologies, not only because they are capable of modeling complex non-linear patterns and effectively leveraging **exogenous variables** but also because they can handle the sheer volume and diverse modalities of contemporary time series data. This evolution signifies a fundamental shift from purely statistical techniques, which often rely heavily on a series' inherent autocorrelation structure, towards more data-driven, scalable, and pattern-recognition approaches that can capture intricate temporal dependencies and long-range interactions [De Gooijer and Hyndman 2006]. This progression naturally sets the stage for exploring the capabilities of FMs, which are designed to learn rich representations from vast datasets.

The importance of TSF is underscored by its broad applicability across a multitude of disciplines. It is a critical instrument for informed decision-making in economic planning, supply chain management, medical diagnosis, and weather prediction; however, these latter applications often involve specialized models and domain knowledge. TSF also plays a crucial role in energy consumption modeling, including renewable energy production, energy demand, and price forecasting, as well as in financial market analysis and various engineering domains like predictive maintenance [Danese and Kalchschmidt 2011]. For instance, in epidemiology, TSF models have been used for short-term forecasting of pandemic spread and assessing public health interventions [Allard 1998; Khan et al. 2021]. Similarly, cloud comput-

ing services utilize TSF for capacity planning to ensure service availability [Saxena and Singh 2021].

A key aspect of TSF, and a crucial consideration for any modeling approach, is distinguishing between phenomena that are genuinely predictable and those that are inherently stochastic or random. For example, the time of sunset next Monday is highly predictable due to well-understood physical laws, whereas next Saturday's lottery numbers are by design random and thus unpredictable. The fundamental assumption underpinning all forecasting endeavors is that the future exhibits some dependency on the past. Thus, historical data and experience are indispensable for reliable extrapolation. This inherent predictability, or lack thereof, directly influences the efficacy of any forecasting model. The true value of TSF, when applied to predictable phenomena, lies in its ability to facilitate a shift from reactive to proactive decision-making, empowering organizations to anticipate future conditions and implement preemptive strategies for resource allocation, risk mitigation, and enhanced operational efficiency [Bergmeir 2020].

2.1.2. Characteristics of Time Series

Time series data encapsulate various intrinsic characteristics that play a critical role in explaining the diverse patterns and fluctuations within observed phenomena. Understanding these characteristic elements is essential for effectively analyzing and predicting future values or detecting changes at critical moments [Box et al. 1976]. While numerous characteristics can define a time series, the following are generally considered the most fundamental for forecasting purposes.

Trend (T_t)

The **trend** (T_t) represents the long-term direction or systematic change in the mean level of a time series over an extended period. This movement can manifest as a sustained increase, decrease, or relative stability. A trend can be linear, often described by an equation such as $T_t = \beta_0 + \beta_1 t$, where t is the time index and β_0, β_1 are parameters, or it can be non-linear. The slope, or rate of change, of a trend may also vary. Exponential growth patterns may emerge in the initial development of new businesses, although such trends usually plateau as the system matures. The presence of a trend is a primary cause of non-stationarity (described below), as it implies that the expected value of the series, $E[Y_t]$, is not constant but changes over time. Handling trends, either by explicit modeling or by removal through detrending or differencing, is a fundamental step in much of time series analysis [Bergmeir 2020]. A separation of the trend component from the AirPassenger dataset is shown later in Figure 2.5 when discussing time series decomposition. It shows a nearly linear increase in passengers over the years.

Seasonality (S_t)

Seasonality (S_t) refers to predictable, periodic fluctuations in a time series that occur at regular intervals, known as the seasonal period, which is shorter than a year (e.g., daily, weekly, monthly). Unlike cyclical patterns (described below), the length of the seasonal period (m) is fixed and known *a priori* and is assumed not to change. These regular variations often arise due to factors like weather patterns

(e.g., higher electricity consumption in summer), holidays (e.g., increased retail sales in December), or other institutional or calendar-related influences. Mathematically, if m denotes the seasonal period (e.g., $m = 12$ for monthly data with an annual pattern), the seasonal component at time t is expected to be similar to the component at time $t - m$, i.e., $S_t \approx S_{t-m}$ [Hyndman and Athanasopoulos 2021]. Seasonal components can be:

- *Additive*: The magnitude of the seasonal swing is relatively constant regardless of the series' level.
- *Multiplicative*: The magnitude of the seasonal swing is proportional to the current level of the series.

As illustrated in the decomposition of the AirPassenger dataset (Figure 2.5), the data shows a stable multiplicative annual seasonality with peaks during the summer months and troughs in winter. The relative proportions of travel in each month of the year stay much the same, just scaled up by the increasing trend. In forecasting, it is often assumed that seasonality is known beforehand and can be extrapolated indefinitely.

Cyclicity (C_t)

Cyclicity (C_t) describes longer-term, recurrent up-and-down movements in the data that are not of a fixed or predictable period. Cyclical patterns are distinguished from seasonality primarily by their irregular periodicity and often more variable amplitude and duration. These cycles are typically associated with economic or business cycles, or other phenomena that occur over several years. The elusive nature of cyclical components makes them particularly challenging to model explicitly compared to trends or seasonality. Consequently, the cyclical component is frequently grouped with the trend component to form a "trend-cycle" component [Hyndman and Athanasopoulos 2021].

Stationarity

A time series is defined as **stationary** if its fundamental statistical properties, specifically its mean, variance, and autocorrelation structure, remain constant over time. This implies that the underlying data-generating process is not changing. Two forms of stationarity are commonly discussed:

- *Strict Stationarity*: A time series $\{Y_t\}$ is strictly stationary if, for any set of time points $t_1, \ldots, t_k$ and any lag h, the joint probability distribution of $(Y_{t_1}, \ldots, Y_{t_k})$ is identical to the joint probability distribution of $(Y_{t_1+h}, \ldots, Y_{t_k+h})$. This is a strong condition and often difficult to verify in practice.
- *Weak or Covariance Stationarity*: This is a less stringent and more commonly used definition. A time series is weakly stationary if it satisfies three conditions:
  1. The mean is constant: $E[Y_t] = \mu$ for all t.
  2. The variance is constant and finite: $\text{Var}(Y_t) = \sigma^2 < \infty$ for all t.

3. The autocovariance between Y_t and Y_{t+k} depends only on the lag k, not on time t: $\text{Cov}(Y_t, Y_{t+k}) = \gamma_k$.

The assumption of stationarity is critical for many traditional statistical forecasting methods, as these models often assume that patterns learned from the past will continue into the future in a statistically consistent manner. Non-stationary series, which may exhibit trends, changing variance (heteroskedasticity), or unit roots like random walks, often require transformation to achieve stationarity. Common transformations include differencing $\nabla y_t = y_t - y_{t-1}$ to remove a linear trend and logarithmic transformations $\log(y_t)$ to stabilize variance [Box et al. 1976; Hyndman and Athanasopoulos 2021].

Figure 2.3 illustrates this process using the monthly airline passengers dataset. The original series (top panel) shows both a strong upward trend and increasing seasonal variance, indicating clear non-stationarity. Applying first differencing (second panel) reduces the trend but does not fully stabilize the variance over time. The logarithmic transformation (third panel) stabilizes variance but leaves a slight positive trend. Only after applying both transformations (bottom panel) does the series exhibit weak stationarity, making it suitable for modeling with traditional approaches.

Differencing, however, primarily addresses specific forms of non-stationarity like unit roots and can lead to a loss of information about the series' original scale, which can sometimes be mitigated by incorporating the original scale or its logarithm as an additional input feature [Bergmeir 2020]. Statistical tests like the Augmented Dickey-Fuller test (ADF) test or the Kwiatkowski-Phillips-Schmidt-Shin test (KPSS) test are commonly used to assess stationarity [Dickey and Fuller 1979; Kwiatkowski et al. 1992].

Autocorrelation (ACF) and Partial Autocorrelation (PACF)

Autocorrelation refers to the correlation of a time series with its own past values. The **Autocorrelation Function (ACF)** measures the linear relationship between an observation at time t and observations at previous times $t - k$. The ACF at lag k, denoted ρ_k, is formally defined as:

$$\rho_k = \text{Corr}(Y_t, Y_{t-k}) = \frac{\text{Cov}(Y_t, Y_{t-k})}{\sqrt{\text{Var}(Y_t)\text{Var}(Y_{t-k})}} \tag{2.1}$$

Assuming weak stationarity, $\text{Var}(Y_t) = \text{Var}(Y_{t-k}) = \gamma_0$ (the variance of the series), and $\text{Cov}(Y_t, Y_{t-k}) = \gamma_k$ (the autocovariance at lag k). Thus, Equation (2.1) simplifies to $\rho_k = \gamma_k/\gamma_0$ [Shumway and Stoffer 2017].

The **Partial Autocorrelation Function (PACF)** at lag k measures the correlation between Y_t and Y_{t-k} after removing the linear influence of the intervening observations $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-(k-1)}$ [Shumway and Stoffer 2017]. Plots of the sample ACF and PACF against various lags k (**correlograms**) are essential tools in time series analysis, particularly for the identification stage of ARIMA modeling [Box et al. 1976].

Figure 2.4 displays the sample ACF and PACF plots for the AirPassengers dataset. In the ACF plot shows a characteristic slow decay of the autocorrelation values, which is typical for non-stationary series possessing a trend. Furthermore, there are distinct peaks at lags 12, 24 and generally at multiples of 12, indicating strong yearly

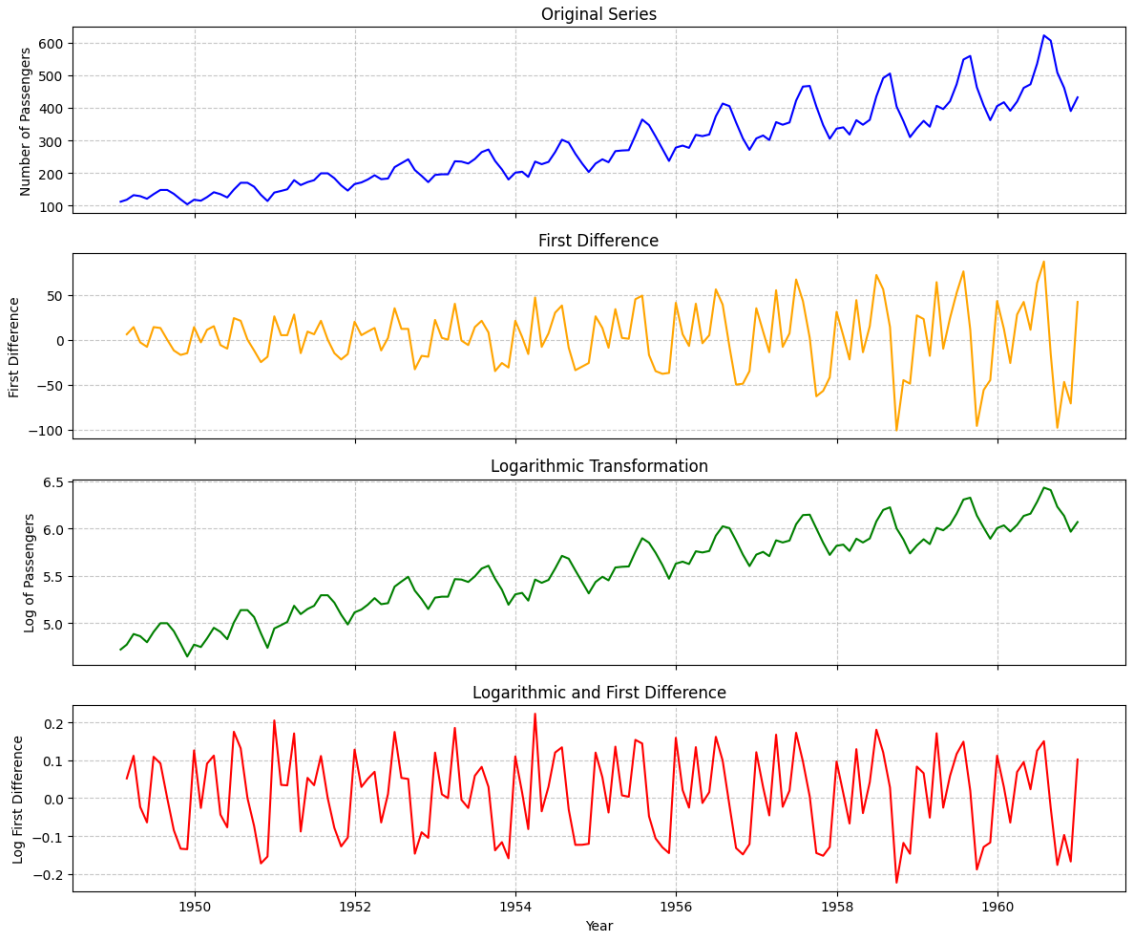


Figure 2.3.: Transformation of the monthly airline passengers dataset to achieve stationarity. Top to bottom: original series, first differencing, logarithmic transformation and a combination of both yielding weak stationarity.

seasonality. The PACF plot on the right also shows significant spikes extending beyond the confidence interval at lags 1, 2, 9 and 13. The significant spikes at the beginning are suggestive of an autoregressive (AR) component of order 2 which is further described in Section 2.3.1. The other spikes close to lag 12 are related to the seasonal dynamics of the series.

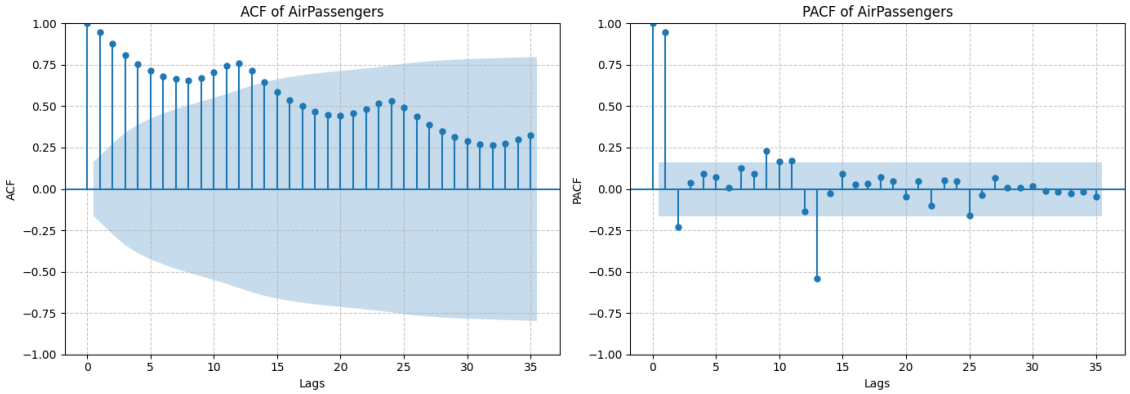


Figure 2.4.: ACF and PACF of the monthly airline passengers dataset.

Irregularity/Noise/Residuals (ϵ_t)

The **irregular component**, also referred to as **noise** or **residuals** (ϵ_t), represents the random, unpredictable variations in a time series that cannot be attributed to the systematic components of trend, seasonality, or cyclicity [Hyndman and Athanasopoulos 2021]. This component reflects the unexplained portion after the identifiable patterns have been accounted for by a model. In the context of a well-specified and adequately fitted forecasting model, the residuals should ideally resemble **white noise** [Hyndman and Athanasopoulos 2021]. A white noise process is a sequence of random variables that are independent and identically distributed with a mean of zero and constant variance, exhibiting no significant autocorrelation. If model residuals do not approximate white noise, it suggests that there is still systematic information in the data that the model has failed to capture, indicating model misspecification or underfitting [Chatfield 2005]. The remaining residuals after filtering trend and seasonality of the AirPassenger dataset is shown in Figure 2.5 in the bottom panel.

Outliers and Missing Values

Beyond the inherent structural components, time series data often present practical challenges such as **outliers** and **missing values**.

- *Outliers*: These are data points that deviate significantly from the overall pattern of the time series [Hyndman and Athanasopoulos 2021]. Outliers can arise from measurement errors, unusual events, or data entry mistakes. Their presence can influence model training, leading to biased parameter estimates and inaccurate forecasts. Identifying and appropriately handling outliers through removal, imputation, or robust modeling techniques is crucial for data quality and model performance.

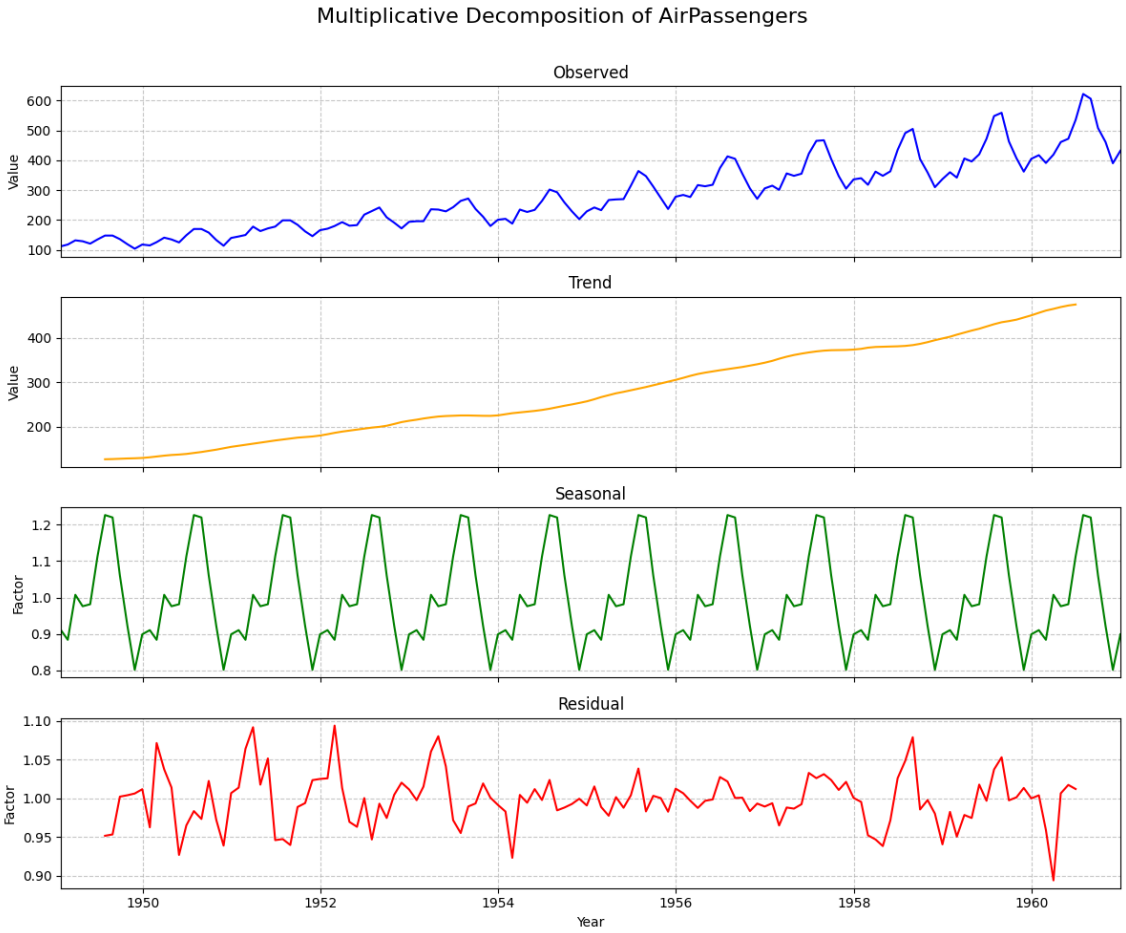


Figure 2.5.: The monthly international airline passengers dataset separated into trend, seasonality and residual parts.

- *Missing Values*: Gaps in time series data, where observations are absent, are common due to sensor failures, data collection interruptions, or survey non-responses [Chatfield 2005]. The presence of missing values can disrupt the temporal continuity and make it challenging to apply standard time series models. Common strategies for handling missing values include linear or seasonal interpolation, mean imputation or model-based imputation, depending on the nature of the missingness and the data characteristics [De Gooijer and Hyndman 2006].

While not always explicitly part of classical decomposition, addressing outliers and missing values is a vital preprocessing step for robust time series analysis and forecasting.

Sporadic and Erratic Time Series

A special class of time series, often encountered in demand forecasting for spare parts or infrequently consumed items are **sporadic** and **erratic** time series. These series are characterized by a large number of zero observations interspersed with non-zero demands.

- *Sporadic Series*: Also known as intermittent series have long periods of zero demand separated by infrequent, non-zero demands [Syntetos and Boylan 2005]. The demand arrivals are irregular, and the demand sizes, when they occur, can also vary.

- *Erratic Series*: These series have frequent non-zero demands, but the demand sizes are highly variable and unpredictable [Wallström and Jonsson 2009]. While demand is always present, its magnitude fluctuates wildly.

Traditional forecasting models often perform poorly on such series due to their inability to effectively model the high frequency of zeros or the extreme variability. Specialized methods, such as Croston's method [Croston 1972] or more advanced machine learning approaches that can handle sparse data, are often required for these types of series.

Time Series Decomposition Models

Time series decomposition is a technique used to separating a time series into its fundamental constituent components: typically trend (T_t), seasonality (S_t), and a residual or irregular component (R_t or ϵ_t) [Hyndman and Athanasopoulos 2021]. Some frameworks may also explicitly include a cyclical component (C_t). The primary purpose of decomposition is to better understand the underlying patterns, which can improve forecasting accuracy and aid in anomaly detection [Hyndman and Athanasopoulos 2021]. Two primary classical decomposition models are:

- *Additive Decomposition*: Assumes the observed value y_t is the sum of its components: $y_t = T_t + S_t + R_t$ (or $y_t = T_t + S_t + C_t + R_t$). This is appropriate when seasonal or irregular fluctuations are relatively constant in magnitude regardless of the series' level [Hyndman and Athanasopoulos 2021].

- *Multiplicative Decomposition*: Assumes y_t is the product of its components: $y_t = T_t \times S_t \times R_t$. This model is used when seasonal or irregular variations are proportional to the current level of the series [Hyndman and Athanasopoulos 2021].

A logarithmic transformation can convert a multiplicative relationship to an additive one: $\log(y_t) = \log(T_t) + \log(S_t) + \log(R_t)$ [Hyndman and Athanasopoulos 2021]. Methods like classical decomposition, Seasonal and Trend decomposition using Loess (STL) [Cleveland et al. 1990], Multiple Seasonalities STL (MSTL) for multiple seasonalities [Hyndman, E. Wang, et al. 2015] and Seasonal and Trend decomposition using Regression (STR) [Dokumentov 2020] are practical tools. Decomposition serves as an analytical method, a preprocessing step and a diagnostic tool.

2.2. Established Forecasting Models

Forecasting models have evolved from classical statistical methods, which assume an underlying stochastic process, to machine learning and deep learning approaches that learn complex patterns directly from data [Chatfield 2005]. Beyond this distinction, models can be fundamentally categorized by their training scope as either **local** or **global**. **Local models**, which include most traditional statistical methods, are trained on a single time series in isolation. A unique model is fitted for each individual series. In contrast, **global models**, a paradigm often powered by deep learning, involve training a single model across an entire collection of related time series. This approach allows the model to learn generalized patterns and share statistical strength across the dataset. The choice between these paradigms and specific model architectures depends on factors like data characteristics, the number of time series available, the forecast horizon and interpretability needs.

2.3. Established Forecasting Models

Forecasting models have progressed from classical statistical methods, which assume an underlying stochastic process, to machine learning and deep learning approaches that learn complex patterns directly from data [Hyndman and Athanasopoulos 2021]. The choice of model depends on data characteristics (e.g., trend, seasonality), forecast horizon, data volume, and interpretability needs.

2.3.1. Statistical Models

Statistical models for TSF are built on the assumption that the observed data are generated by an underlying stochastic process. These models aim to capture the statistical properties of this process, often relying on assumptions like linearity or stationarity [Box et al. 1976].

Naive and Mean Forecasts

Two of the simplest benchmarks are the naive forecast and the mean forecast.

- **Naive Forecast**: The standard naive forecast for a multi-step horizon H sets all future predicted values to be the last observed value at time T: $\hat{y}_{T+h|T} = y_T$ for all $h = 1, \ldots, H$.

  For seasonal data, the **Seasonal Naive Forecast** is commonly used. It predicts $\hat{y}_{T+h|T} = y_{T+h-m}$ for $h = 1, \ldots, H$, where m is the seasonal period. This means the forecast for $T+h$ is taken from the corresponding period in the last observed season $\hat{y}_{T+1|T} = y_{T+1-m}$, $\hat{y}_{T+2|T} = y_{T+2-m}$, ..., $\hat{y}_{T+m|T} = y_T$, and this specific sequence of m past values is repeated for subsequent steps if $H > m$.

  In the experimental evaluations presented in subsequent chapters of this thesis, a specific "rolling" or "persistence-based" benchmark variant of the naive forecast is employed for multi-step horizons. For evaluation purposes only, this approach sets the forecast for a given step h within the horizon ($1 \leq h \leq H$) to the actual observed value from the immediately preceding time step within the test set. That is, to evaluate the forecast for y_{T+h}, the value y_{T+h-1} is used where y_{T+0} is taken as y_T, the last point of the training data. It is important to note that for $h > 1$, this benchmark utilizes information from the test set specifically, the actual values $y_{T+1}, \ldots, y_{T+H-1}$ that would not be available at the original forecast time T. It serves as a strong baseline representing step-by-step persistence of actual values observed within the horizon.

- **Mean Forecast**: Predicts all future values over a horizon H to be the average of all past observations available at time T: $\hat{y}_{T+h|T} = \bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$ for $h = 1, \ldots, H$.

These methods serve as crucial benchmarks and any sophisticated model should aim to outperform them. The standard naive forecast considers only the very last observation, while the mean forecast weights all past observations equally. A central assumption in many advanced methods is that the more recent past is more important for forecasting future values [Box et al. 1976].

Exponential Smoothing (ES) Methods

Exponential Smoothing (ES) methods generate forecasts that are weighted averages of past observations, with weights decreasing exponentially for older observations [Holt 1957]. This gives more importance to recent data.

- **Simple Exponential Smoothing (SES)**: Suitable for data with no clear trend or seasonality. The level equation is:

$$\ell_t = \alpha y_t + (1 - \alpha)\ell_{t-1} \tag{2.2}$$

  The forecast equation for h steps ahead is:

$$\hat{y}_{T+h|T} = \ell_T \tag{2.3}$$

  Here, ℓ_t is the smoothed level at time t, y_t is the observation at time t, and $\alpha \in [0, 1]$ is the smoothing parameter for the level [Holt 1957].

- **Holt's Linear Trend Method**: Extends SES for data with a linear trend Holt [1957] It introduces a trend component b_t and a trend smoothing parameter $\beta^* \in [0, 1]$. The level equation is:

$$\ell_t = \alpha y_t + (1 - \alpha)(\ell_{t-1} + b_{t-1}) \tag{2.4}$$

The trend equation is:

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1 - \beta^*)b_{t-1} \tag{2.5}$$

The forecast equation for h steps ahead is:

$$\hat{y}_{T+h|T} = \ell_T + hb_T \tag{2.6}$$

A *damped trend* version introduces a damping parameter $\phi \in (0, 1)$ to moderate the trend for longer horizons:

$$\hat{y}_{T+h|T} = \ell_T + \left(\sum_{i=1}^{h} \phi^i\right) b_T \tag{2.7}$$

This is often used to be conservative about long-term trend forecasts [Bergmeir 2020].

- **Holt-Winters Seasonal Method**: Extends Holt's method to capture seasonality Winters [1960]. It adds a seasonal component s_t and a seasonal smoothing parameter $\gamma \in [0, 1]$. For *additive seasonality* where seasonal variations are roughly constant the forecast equation is:

$$\hat{y}_{T+h|T} = \ell_T + hb_T + s_{T+h-m(k+1)} \tag{2.8}$$

where $k = \lfloor (h-1)/m \rfloor$ and m is the seasonal period. For *multiplicative seasonality* where seasonal variations are proportional to the series level the forecast equation is:

$$\hat{y}_{T+h|T} = (\ell_T + hb_T)s_{T+h-m(k+1)} \tag{2.9}$$

The Theta Model

The Theta model, introduced by Assimakopoulos and Nikolopoulos [2000] is a univariate statistical forecasting method known for its simplicity and strong empirical performance, notably in the M3 forecasting competition [Makridakis and Hibon 2000]. The core idea involves decomposing the original time series or its deseasonalized version by modifying its local curvatures through a coefficient θ. The forecast is then derived from the combination of extrapolating these modified series, often referred to as Theta-lines.

The standard Theta method typically proceeds as follows:

1. **Seasonality Test and Adjustment**: First, the series is tested for seasonality. If seasonality is detected, the data y_t are deseasonalized, resulting in a series y'_t. If no seasonality is present, $y'_t = y_t$.

2. **Theta-line Construction and Forecasting**: Two Theta-lines are constructed and forecasted:
   - The first line ($Z'_t(0)$) corresponds to $\theta = 0$. This line is equivalent to fitting a simple linear regression trend to y'_t: $Z'_t(0) = a + bt$, where a is the intercept and b is the slope. The forecast from this line, $\hat{z}'_{T+h|T}(0)$, is a linear extrapolation of this trend.
   - The second line ($Z'_t(2)$) corresponds to $\theta = 2$. This line is constructed as $Z'_t(2) = 2y'_t - Z'_t(0)$. This effectively doubles the curvature of the original (deseasonalized) series relative to the linear trend. This $Z'_t(2)$ series is then forecasted using SES, yielding $\hat{z}'_{T+h|T}(2)$.
3. **Forecast Combination**: The forecasts from the two lines are combined, typically by simple averaging:

$$\hat{y}'_{T+h|T} = \frac{1}{2}\left(\hat{z}'_{T+h|T}(0) + \hat{z}'_{T+h|T}(2)\right) \tag{2.10}$$

4. **Reseasonalization**: If the data were initially deseasonalized, the combined forecast $\hat{y}'_{T+h|T}$ is then reseasonalized to produce the final forecast $\hat{y}_{T+h|T}$.

The coefficient θ in the general Theta model formulation, $Z_t(\theta)$, essentially dampens ($\theta < 1$), leaves unchanged ($\theta = 1$), or exaggerates ($\theta > 1$) the curvature of the original series. The standard choice of $\theta = 0$ and $\theta = 2$ for the two lines has shown robust performance. Hyndman and Billah [2003] later showed that the Theta method is equivalent to Simple Exponential Smoothing with drift under certain conditions, providing a statistical basis for its performance. Its strength lies in its ability to capture long-term trends effectively while also being responsive to recent changes via the SES component on the amplified line.

Autoregressive Integrated Moving Average (ARIMA) Models

ARIMA models provide a flexible framework for modeling time series with autocorrelation Box et al. [1976]. An ARIMA(p, d, q) model consists of:

- **AR(p) (Autoregressive)**: y_t is a linear combination of p past values: $y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t$. Here, c is a constant, ϕ_i are the autoregressive parameters, and ϵ_t is white noise.
- **I(d) (Integrated)**: d is the degree of differencing applied to achieve stationarity but loses information about the original scale.
- **MA(q) (Moving Average)**: y_t is a linear combination of q past white noise error terms: $y_t = c + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$. Here, θ_j are the moving average parameters.

The general model for the d-th differenced series $w_t = (1 - B)^d y_t$, where B is the backshift operator such that $By_t = y_{t-1}$, is:

$$(1 - \phi_1 B - \cdots - \phi_p B^p) w_t = c + (1 + \theta_1 B + \cdots + \theta_q B^q)\epsilon_t \tag{2.11}$$

Fitting Autoregressive Moving Average (ARMA) models involves estimating initial conditions and iterative optimization over the time series, requiring non-linear fitting procedures as there is no closed-form solution, which can be slow for long series. Any stationary Autoregressive (AR)(1) model has an equivalent Moving Average (MA)(∞) representation, and any invertible MA(1) model has an AR(∞) representation. This implies that the MA component of an ARMA model can often be mathematically represented by a more complex AR component. Therefore, choosing an ARMA model can be seen as a way to offer a more compact parametrization to describe the series, but this comes at the cost of a more complex model fitting process.

Seasonal Autoregressive Integrated Moving Average (SARIMA) models, denoted SARIMA$(p, d, q)(P, D, Q)_m$, extend ARIMA to handle seasonality by adding seasonal AR, I, and MA terms operating at lag m [Hyndman and Athanasopoulos 2021]:

$$\Phi_P(B^m)\phi_p(B)(1 - B^m)^D(1 - B)^d y_t = c + \Theta_Q(B^m)\theta_q(B)\epsilon_t \tag{2.12}$$

Here, $\Phi_P(B^m)$ and $\phi_p(B)$ are the seasonal and non-seasonal AR polynomials, and $\Theta_Q(B^m)$ and $\theta_q(B)$ are the seasonal and non-seasonal MA polynomials, respectively. The **Box-Jenkins methodology** [Box et al. 1976] is an iterative approach:

1. *Identification*: Assess stationarity (difference if needed). Use ACF and PACF plots of the stationary series to select orders (p, d, q) and $(P, D, Q)_m$.

2. *Estimation*: Estimate model parameters, for example, via Maximum Likelihood Estimation (MLE).

3. *Diagnostic Checking*: Evaluate if residuals are white noise (e.g., using ACF/PACF of residuals, Ljung-Box test [Ljung and Box 1978]). If inadequate, refine the model.

Standard implementations like 'auto.arima()' in R's 'forecast' package [Hyndman, Koehler, et al. 2008] automate model selection using criteria like the Corrected Akaike Information Criterion.

ETS / State Space Models (SSM)

The Error, Trend, Seasonality models (ETSs) offer a statistical framework that encompasses and extends ES methods, with each ETS having an underlying State Space Model (SSM) representation. The ETS taxonomy uses (E,T,S) for Error (Additive A / Multiplicative M), Trend (None N, Additive A, Additive Damped A_d, Multiplicative M, Multiplicative Damped M_d), and Seasonality (None N, Additive A, Multiplicative M) [Hyndman, Koehler, et al. 2008].

A general linear Gaussian SSM is defined by:

- **Observation Equation**:

$$y_t = Z_t\alpha_t + d_t + \epsilon_t, \quad \epsilon_t \sim \text{NID}(0, H_t) \tag{2.13}$$

- **State Equation**:

$$\alpha_{t+1} = T_t\alpha_t + c_t + R_t\eta_t, \quad \eta_t \sim \text{NID}(0, Q_t) \tag{2.14}$$

Here, y_t is the observed data, α_t is the unobserved state vector containing components like level, trend, and seasonality, Z_t, T_t, R_t are system matrices which can be time-varying, d_t and c_t are deterministic terms, and ϵ_t, η_t are white noise error terms, typically assumed to be Normally and Independently Distributed (NID). The **Kalman filter** [Kalman 1960] is used for optimal estimation of states, prediction, and smoothing. MLE is used for parameter estimation [Fisher 1922]. Advantages of ETS/SSM include likelihood calculation, model selection via criteria like the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC), and the generation of statistically valid prediction intervals [Akaike 1974; Schwarz 1978]. Diagnostic checking involves analyzing standardized one-step-ahead prediction errors, which should ideally be Gaussian white noise [Hyndman and Athanasopoulos 2021].

2.3.2. Classical Machine Learning Models

Classical ML models offer alternatives by learning input-output relationships directly from data, often non-parametrically and are adept at capturing non-linearities [Bergmeir 2020].

Adapting ML for Time Series (Feature Engineering)

Standard ML regression algorithms like Support Vector Regression (SVR), Random Forests (RFs), or Gradient Boosting Machiness (GBMs) are not inherently designed for sequential data [Hastie et al. 2003]. Time series data must be transformed into a supervised learning format (X, y) through feature engineering [L. Wang et al. 2012]. Key techniques include:

- **Lagged Variables**: Past values of the target series like $y_{t-1}, y_{t-2}, \ldots, y_{t-k}$ are used as input features to predict y_t or a future value y_{t+h}. This is conceptually similar to autoregression in statistical models and is often implemented with a sliding window approach.
- **Rolling Window Statistics**: Features derived from a rolling window over past data, such as the rolling mean, median, standard deviation, minimum, or maximum.
- **Time-Based Features**: Calendar-derived features like the hour of the day, day of the week, month of year... or indicators for holidays and special events. For cyclical features like month of the year sine and cosine transformations $\sin(2\pi \cdot \text{month}/12)$, $\cos(2\pi \cdot \text{month}/12)$ can be used to preserve their cyclical nature.
- **Exogenous Variables**: External factors that are believed to influence the time series used as additional input features.

The quality and relevance of engineered features are critical for the performance of ML models in TSF [L. Wang et al. 2012].

Support Vector Regression (SVR)

SVR adapts Support Vector Machine (SVM) for regression tasks. It aims to find a function $f(x) = w^T x + b$ that deviates from the target values y_i by at most a margin ϵ utilizing an ϵ-insensitive loss function while simultaneously minimizing the model complexity, often represented by $\|w\|^2$ [Vapnik et al. 1997]. The optimization problem is typically formulated as:

$$\begin{aligned} \text{Minimize: } & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{Subject to: } & y_i - (w^T x_i + b) \leq \epsilon + \xi_i, \\ & (w^T x_i + b) - y_i \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0 \quad \text{for } i = 1, \ldots, N. \end{aligned}$$

Here, $C > 0$ is a regularization parameter that balances model complexity and the amount of deviation larger than ϵ tolerated, and ξ_i, ξ_i^* are slack variables representing the magnitude of errors. Non-linearity is handled by mapping the input data into a higher-dimensional space using kernel functions, such as the linear, polynomial, or Radial Basis Function (RBF) kernel [Schölkopf and Smola 2002]. Advantages of SVR include good generalization performance, especially in high-dimensional spaces, and robustness to some types of outliers due to the ϵ-insensitive loss. Limitations include computational intensity, particularly for large dataset, and sensitivity to hyperparameter choices. Standard SVR's global ϵ-tube may not be suitable for volatile time series with changing noise levels. Localized Support Vector Regression (LSVR) attempts to address this by adapting ϵ locally [Schölkopf and Smola 2002].

Random Forest (RF)

RF is an ensemble learning method that constructs a multitude of decision trees during training [Breiman 2001]. For regression tasks, the predictions of individual trees are typically averaged to produce the final output. RF is adapted for TSF by using the engineered features as described in Section 2.3.2 as inputs to the trees. Its robustness and performance stem from dual randomization: using bootstrap aggregating (bagging) of training samples for each tree and selecting a random subset of features at each split point when growing the trees. Advantages include the ability to capture complex non-linear relationships, robustness to outliers, and an inherent mechanism for estimating feature importance [Bergmeir 2020]. Limitations include being computationally intensive for training and prediction with a large number of trees and its performance can be sensitive to the selection of lags if too many are used with short series, reducing the effective number of training instances. Moreover, RFs primarily interpolate based on the learned patterns and may not extrapolate well beyond the range of target values observed in the training data [Hyndman and Athanasopoulos 2021].

Gradient Boosting Machines (GBM)

GBMs are a powerful ensemble method that constructs models—typically decision trees—in a sequential, stage-wise manner. Each new tree is trained to correct the

errors (residuals) of the ensemble of preceding trees [J. H. Friedman 2001]. Prominent implementations include XGBoost, LightGBM, and CatBoost [T. Chen and Guestrin 2016; Dorogush et al. 2018; Ke et al. 2017]. GBMs are adapted for TSF through the use of engineered features. Advantages often include high predictive accuracy, as evidenced by LightGBM's strong performance in competitions like the M5 [Makridakis, Spiliotis, et al. 2022], flexibility in incorporating diverse features, and native handling of categorical features in some implementations. CatBoost's ordered gradient boosting strategy is particularly noted for its suitability for time series data as it respects the temporal order of observations during training, reducing prediction shift [Bergmeir 2020]. Limitations include sensitivity to hyperparameter tuning, a potential for overfitting if not carefully regularized (e.g., through tree depth limits, learning rate, early stopping) and the requirement that future values of any exogenous variables used as features must be known or accurately forecasted themselves.

2.3.3. Deep Learning Models

DL models, characterized by multi-layered neural networks, can automatically learn hierarchical feature representations from data. They are particularly adept at capturing complex, non-linear patterns and long-range dependencies often present in sequential data.

Introduction to Deep Learning for Sequential Data

DL methods aim to autonomously extract intricate temporal features directly from the raw input data, often reducing the reliance on extensive manual feature engineering that is typical for classical ML approaches. This also contrasts with traditional statistical models, which usually require strong assumptions about the underlying stochastic process. Common DL architectures for TSF include DLs, LSTM networks, Gated Recurrent Units (GRUs), Transformer models, and Convolutional Neural Networks (CNNs) adapted for sequences. However, DL models are generally data-hungry, can be computationally expensive to train, and are often considered less interpretable "black boxes" compared to simpler models [Lim and Zohren 2021].

Multi-Layer Perceptrons (MLP) and MLP-based Architectures

A Multi-Layer Perceptron (MLP) is a fundamental class of feedforward artificial neural network. It consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node, or neuron, in one layer connects with a certain weight to every neuron in the subsequent layer. Non-linear activation functions (e.g., Rectified Linear Unit (ReLU), sigmoid, or tanh) are typically applied to the outputs of neurons in the hidden layers, enabling the MLP to learn complex, non-linear relationships in the data [Goodfellow et al. 2016].

For TSF, standard MLPs are applied similarly to classical machine learning models, requiring the transformation of time series data into a supervised learning format through feature engineering (as discussed in Section 2.3.2). This involves creating input vectors x_t composed of lagged values of the series, time-based features, rolling window statistics, and/or exogenous variables. The MLP then learns a mapping

$f(x_t)$ to predict a future value y_{t+h} or a sequence of future values. Each input vector x_t is processed independently through the network. The output of the l-th layer, $h^{(l)}$, can be described as:

$$h^{(l)} = \sigma^{(l)}(W^{(l)}h^{(l-1)} + b^{(l)}) \tag{2.15}$$

where $h^{(0)}$ is the input vector x_t, $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector for layer l respectively, and $\sigma^{(l)}$ is its activation function. The output layer's activation function is typically linear for regression tasks like forecasting.

Strengths of MLPs include their capability as universal function approximators and their ability to model non-linear patterns effectively [Goodfellow et al. 2016]. They are conceptually simpler than more specialized sequential architectures. **Limitations**, however, arise from their inherent feedforward nature. Standard MLPs do not explicitly model temporal order or dependencies within the raw sequence data; these must be encoded through the engineered features. They process fixed-size input vectors and may require careful tuning of hyperparameters to perform well and avoid overfitting. Despite these limitations, MLPs serve as important benchmarks and form the basis for more advanced architectures. Two such MLP-based deep learning models for TSF are N-BEATS and TiDE [Das, Kong, Leach, et al. 2023; Oreshkin et al. 2020].

N-BEATS (Neural Basis Expansion Analysis for Time Series) The N-BEATS model is a deep feed-forward network built from stacks of fully-connected MLP blocks with forward and backward residual links [Oreshkin et al. 2020]. Each block takes a look-back window of the time series and produces a "backcast" (the portion of the input series explained by the block) and a "forecast" output for a future horizon. The backcast is subtracted from the block's input, and the residual is passed to the next block. This iterative decomposition allows N-BEATS, particularly in its interpretable configuration, to model distinct time series components like trend and seasonality by having blocks project onto predefined basis functions (e.g., polynomials for trend, Fourier series for seasonality). In its generic configuration, each block projects onto an identity basis, learning arbitrary mappings. N-BEATS captures temporal dynamics implicitly through these deep MLP mappings within each block, without explicit recurrence or attention mechanisms; the depth and residual structure enable the learning of complex time patterns from the fixed-length input window.

TiDE (Time-series Dense Encoder) The Time-series Dense Encoder (TiDE) is another MLP-based model, specifically an encoder-decoder architecture designed for long-horizon forecasting [Das, Kong, Leach, et al. 2023]. It relies entirely on dense (fully-connected) layers and residual connections, deliberately avoiding recurrent or attention mechanisms. The TiDE architecture consists of a dense encoder and a dense decoder, plus a global linear skip connection. The encoder processes a flattened feature vector, which includes the target series history and potentially other engineered features, through several residual MLP layers to produce a fixed-size context encoding. The decoder then uses this encoding, along with information about the future horizon, to generate forecasts, again through dense layers. A key aspect of its temporal modeling is a global linear skip connection from past inputs directly to the forecast horizon, ensuring it can capture simple linear autoregressive

patterns. TiDE captures more complex time dependencies implicitly through its deep MLP structure operating on fixed-size windows of flattened time-series data and engineered features. The model has no built-in notion of time ordering beyond the fixed input positions of the flattened features.

Recurrent Neural Networks (RNN)

RNNs are a class of neural networks specifically designed for processing sequential data. They utilize recurrent connections, where the output from a previous time step is fed as input to the current time step, creating an internal 'memory' of past information [Goodfellow et al. 2016]. The hidden state h_t at time t is a function of the current input x_t and the previous hidden state h_{t-1}. The output y_t at time t is then typically generated from the current hidden state. These relationships can be expressed as:

$$h_t = \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.16}$$
$$y_t = \sigma_y(W_{hy}h_t + b_y) \tag{2.17}$$

where W_{xh}, W_{hh}, W_{hy} are weight matrices, b_h, b_y are bias vectors, and σ_h, σ_y are activation functions like sigmoid or tanh. Strengths of RNNs include their inherent ability to model sequences and temporal dependencies. Limitations primarily stem from the vanishing or exploding gradient problems, which hinder their capacity to learn long-range dependencies effectively in practice [Goodfellow et al. 2016].

Long Short-Term Memory (LSTM) Networks

LSTM networks were developed to address the long-range dependency limitations of simple RNNs [Hochreiter and Schmidhuber 1997]. LSTMs achieve this through a more complex recurrent unit that includes a memory cell (c_t) and three multiplicative gates: a forget gate (f_t), an input gate (i_t), and an output gate (o_t). These gates regulate the flow of information into and out of the cell state and to the hidden state. The key equations for an LSTM unit are:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.18}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.19}$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.20}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.21}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.22}$$
$$h_t = o_t \odot \tanh(c_t) \tag{2.23}$$

Simplified notation $[h_{t-1}, x_t]$ denotes concatenation of the previous hidden state and current input while $\odot$ denotes element-wise multiplication. Strengths of LSTMs lie in their proven ability to effectively capture long-range dependencies in various sequential data tasks. Bergmeir [2020] notes that LSTMs often still benefit from being structured with explicit input/output windows akin to feature engineering for classical ML, which can enhance their autoregressive nature and make the internal state less critical for capturing the entirety of relevant history.

Gated Recurrent Unit (GRU) Networks

GRUs, introduced by Cho et al. [2014], provide a simplification of the LSTM architecture while retaining comparable performance on many tasks. GRUs combine the forget and input gates into a single "update gate" (z_t) and merge the cell state and hidden state. They employ two gates: an update gate (z_t) and a reset gate (r_t). The key equations for a GRU unit are simplified:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{2.24}$$
$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{2.25}$$
$$\tilde{h}_t = \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h) \tag{2.26}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.27}$$

Strengths of GRUs include a good balance between performance and computational efficiency. They often train faster and require fewer parameters than LSTMs which can make them less prone to overfitting on smaller datasets [Cho et al. 2014].

Transformer Models

Transformer models, originally introduced for machine translation by Vaswani et al. [2017] have become a dominant architecture in many sequence processing tasks, including TSF [Lim and Zohren 2021]. Unlike RNNs, LSTMs, and GRUs, Transformers are not based on recurrence but rely almost entirely on an attention mechanism, particularly **Self-Attention**, to draw global dependencies between inputs and outputs. They typically employ an encoder-decoder structure, though variations exist.

The **Self-Attention Mechanism** allows the model to weigh the importance of different elements in a sequence when processing a particular element, effectively modeling relationships between all pairs of elements regardless of their distance. A common variant is Scaled Dot-Product Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.28}$$

where Q (Query), K (Key), and V (Value) are matrices derived from the input sequence, and d_k is the dimension of the key vectors [Vaswani et al. 2017]. **Multi-Head Attention** enhances this by performing the attention mechanism multiple times in parallel with different, learned linear projections of Q, K, V, allowing the model to jointly attend to information from different representation subspaces at different positions. Since Transformers do not use recurrence, **Positional Encoding** is added to the input embeddings to provide the model with information about the relative or absolute position of the elements in the sequence [B. Kim et al. 2024]. Strengths of Transformers include their exceptional ability to capture long-range dependencies due to the direct pairwise interactions enabled by self-attention and their high parallelizability during training [B. Kim et al. 2024]. Limitations include their significant data requirements, the quadratic computational complexity ($\mathcal{O}(L^2)$) of the vanilla self-attention mechanism with respect to sequence length L , and the critical role of appropriate positional encoding.

To overcome the specific limitations of applying Transformers to time series, two key strategies have emerged: patching and modeling cross-dimensional relationships.

Patching for Time Series A fundamental challenge is that individual time series data points, unlike words in a sentence, carry little semantic meaning in isolation. The point-wise self-attention mechanism, therefore, struggles to capture local patterns and can be computationally inefficient over long sequences. Inspired by the success of Vision Transformers, patching has been adapted for time series [Nie et al. 2023]. This technique involves dividing the input lookback window into a sequence of smaller, contiguous patches, as shown in Figure 2.6. Each patch is then projected into an embedding vector. The Transformer's self-attention mechanism operates on this sequence of patches rather than on the raw data points. This approach has two main advantages:

1. **Preservation of Locality:** By embedding an entire patch, the model captures local contextual information (e.g., trends or seasonal shapes) within each token.

2. **Computational Efficiency:** It significantly reduces the length of the input sequence for the attention mechanism from sequence length L to number of patches N, where $N << L$, mitigating the quadratic complexity and allowing the model to efficiently process much longer lookback windows.

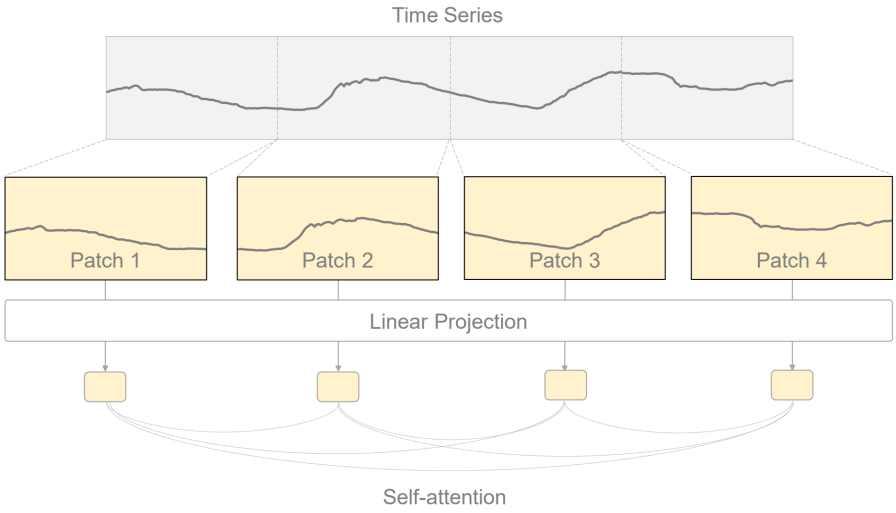


Figure 2.6.: The patching technique. The input time series is segmented into patches, which become the input tokens for the Transformer, preserving local information and reducing sequence length [J. Kim et al. 2024].

Modeling Cross-Dimensional Dependencies In multivariate time series, understanding the relationships between different variables (or channels) is often crucial. Standard Transformers typically process each channel independently (a "channel-independent" approach) before a final prediction layer. While surprisingly effective in some models like PatchTST, this neglects to explicitly learn from inter-variable correlations. To address this, cross-dimensional attention mechanisms have been developed (Figure 2.7). Instead of applying attention only along the time axis, these models also compute attention across the variable dimension. This allows the model to learn which variables influence others and with what potential time lags (e.g., how a change in a "temperature" variable affects a "power consumption" variable).

Architectures may handle this by adding a separate attention stage for dimensions [Zhang et al. 2023] or, in a paradigm shift, by inverting the model to treat variables as the primary tokens and learn their relationships first before making temporal predictions [Yongliang Liu et al. 2024].

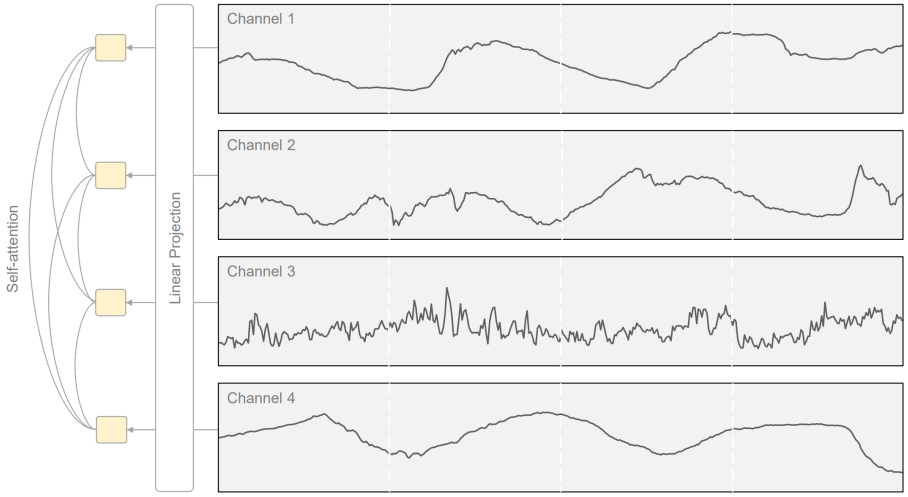


Figure 2.7.: Cross-dimensional self-attention. The attention mechanism is applied across the different variables (channels) to explicitly model their interdependencies [J. Kim et al. 2024].

Convolutional Neural Networks (CNNs) for Time Series Forecasting

CNNs, while predominantly known for their success in computer vision, can also be highly effective for TSF. Recent studies and empirical results suggest that CNN-based architectures can perform on par with or even outperform RNNs on various TSF tasks, often with significantly faster training times [Borovykh et al. 2017; J. Kim et al. 2024]. Architectures like WaveNet, originally developed for audio generation [Oord et al. 2016], have popularized the use of *causal convolutions* for time series. Causal convolutions ensure that the prediction for time step t only depends on inputs from t and earlier, preventing any leakage of future information. Furthermore, *dilated convolutions* are often employed to exponentially increase the receptive field of the network with fewer layers, allowing the model to capture longer-range dependencies without a dramatic increase in computational cost. Since CNNs generally lack an explicit internal state like RNNs, the input window provided to the network must be sufficiently long to cover all relevant historical information, which can necessitate very long input windows, especially when trying to capture long seasonal patterns, even with the use of dilations [J. Kim et al. 2024].

Hybrid and Combined Deep Learning Architectures

Some advanced deep learning models combine elements from different architectural paradigms to leverage their respective strengths. A prominent example is the Temporal Fusion Transformer.

Temporal Fusion Transformers (TFT) The TFT, introduced by Lim, Arık, et al. [2021], is a hybrid deep learning architecture designed for multi-horizon forecasting

that explicitly handles various types of input features (static exogenous variables, past time-varying inputs, and known future time-varying inputs, with covariate handling detailed in Section 2.5.3). TFT integrates several specialized components:

- **Gated Residual Networks (GRNs)**: These are used throughout the model as a core building block, employing Gated Linear Units (GLUs) to allow the network to adaptively skip over or modify layers, aiding in dynamic depth control and gradient flow.
- **Variable Selection Networks**: Separate variable selection networks are applied to different categories of input features, using GRNs and softmax layers to learn and assign importance weights to each input feature, enabling instance-wise feature selection.
- **Static Covariate Encoders**: Static features are processed by GRNs to produce context vectors that condition subsequent parts of the model.
- **Sequence-to-Sequence LSTM Encoder-Decoder**: An LSTM-based encoder-decoder is used to capture local temporal patterns and process sequences of time-varying inputs.
- **Interpretable Multi-Head Self-Attention**: After LSTM encoding, a masked multi-head self-attention layer is applied. This allows each future forecast step to attend to relevant features across the entire historical context, capturing long-range dependencies.

Through its gating mechanisms and attention weights, TFT aims to provide a degree of interpretability regarding feature importance and temporal patterns. Local short-term patterns are primarily handled by the LSTM component, while long-range dependencies are captured by the self-attention layer.

2.4. Evaluation of Time Series Forecasts

Effective evaluation is crucial for assessing the performance and reliability of TSF models. This section outlines the different types of forecasts, common strategies for splitting and validating time series data and the key metrics used to quantify forecast accuracy.

2.4.1. Forecast Types

Time series forecasts can broadly be categorized into point forecasts and probabilistic forecasts. **Point forecasts** provide a single, "best guess" future value for each time step in the forecast horizon. They are typically the mean or median of the underlying predictive distribution. While straightforward to interpret and common in practice, point forecasts do not convey the inherent uncertainty associated with the prediction. **Probabilistic forecasts**, in contrast, quantify the uncertainty of future outcomes by providing a full predictive distribution, typically expressed as a set of quantiles or prediction intervals. This allows for a more comprehensive understanding of potential future scenarios and associated risks. Probabilistic forecasts

are particularly valuable in decision-making contexts where risk assessment is critical, such as inventory management or energy load forecasting [Gneiting and Raftery 2007].

The forecast illustrated in Figure 2.1 provides an excellent example of both types. The red line represents the **point forecast**, offering the most likely future passenger numbers. The surrounding shaded area is a 95% **prediction interval**, which is a form of probabilistic forecasting that communicates the range of likely outcomes.

2.4.2. Data Splitting and Validation Strategies

Proper data splitting is essential to ensure that a forecasting model's performance is evaluated on unseen data, thereby simulating its behavior in a real-world application. Unlike cross-sectional data, time series data possess an inherent temporal order, which must be preserved during the splitting and validation process to avoid look-ahead bias. The two primary approaches, **fixed origin** and **rolling origin**, are visually contrasted in Figure 2.8. Common strategies include:

- **Fixed Origin Evaluation or Hold-out:** The simplest method, where the available time series is split chronologically into a single training set (historical data) and a single test set (future data). The model is trained once on the training data and evaluated on the test data. This approach is straightforward but may not capture performance variability over different time periods and can be sensitive to the specific point chosen for the split.
- **Rolling Forecast Origin or Walk-Forward Validation:** This strategy simulates a real-time forecasting scenario more closely by repeatedly splitting the data. The forecast origin (the point separating training from test data) moves forward in time, and the model is typically retrained and re-evaluated at each step. This provides a more robust estimate of a model's performance across different periods and can help assess its adaptability to potential concept drift. This rolling approach has two main variants for handling the training data window, as illustrated in Figure 2.9:
  - **Fixed or Sliding Window:** The training window size remains constant, sliding forward in time. As new data becomes available for training, the oldest data is dropped from the training set. This keeps the computational cost per iteration relatively stable and ensures the model adapts to the most recent data patterns.
  - **Expanding Window:** The training window grows with each iteration, incorporating all historical data up to the current forecast origin. This ensures the model always uses the maximum available historical information but can become computationally expensive for very long time series and may be less adaptive if old data patterns are no longer relevant.

The choice of validation strategy depends on the stationarity of the series, the length of the available data, computational constraints, and the evaluated model.

2.4.3. Forecast Accuracy Metrics

Evaluating forecast accuracy requires appropriate metrics that quantify the discrepancy between predicted values and actual outcomes. The choice of metrics depends

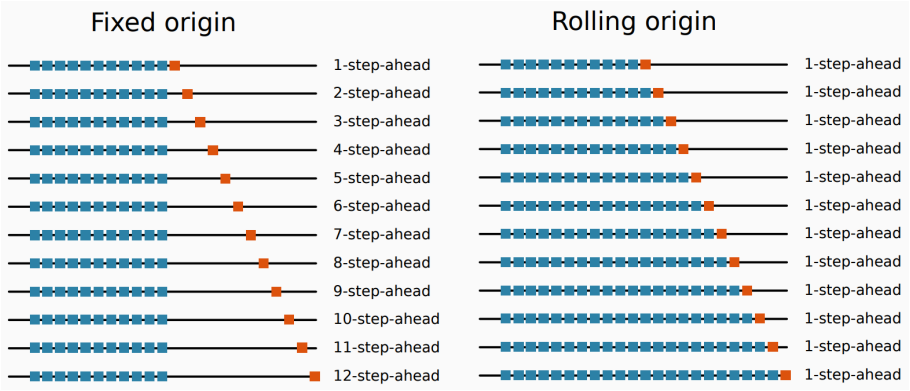


Figure 2.8.: Comparison of evaluation strategies. "Fixed origin" uses a single training set to make multi-step forecasts. "Rolling origin" repeatedly moves the forecast origin forward, typically for single-step forecasts, to simulate real-world model deployment [Bergmeir 2020].

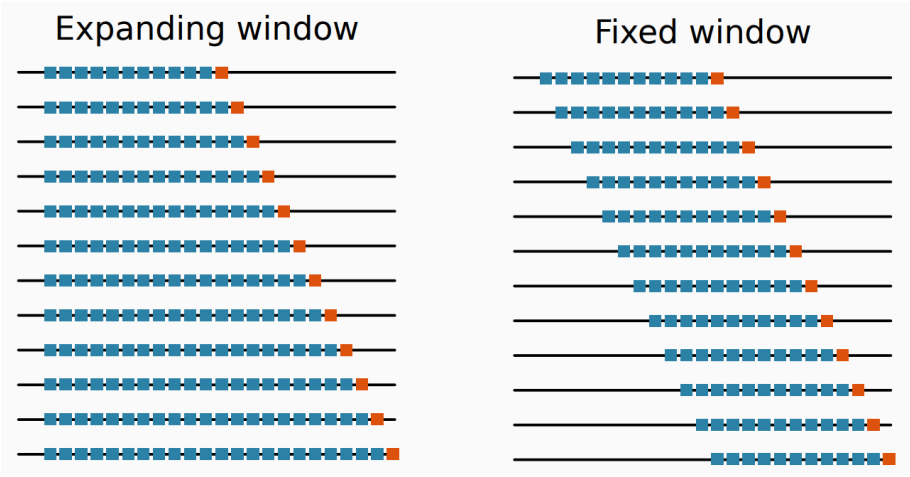


Figure 2.9.: Windowing strategies for rolling origin validation. The "expanding window" (left) accumulates all past data for training at each step. The "fixed window" (right) maintains a constant training set size by sliding through time [Bergmeir 2020].

on the forecast type (point or probabilistic) and the characteristics of the time series data like the scale, presence of zeros or outliers and specific application costs of errors.

Point Forecast Metrics

For point forecasts ($\hat{y}_{T+t}$ for actual y_{T+t} over a forecast horizon H), common error measures include:

- **Mean Absolute Error (MAE):** Measures the average magnitude of the errors without considering their direction. It is less sensitive to outliers than squared-error measures.

$$\text{MAE} = \frac{1}{H}\sum_{t=1}^{H}|y_{T+t} - \hat{y}_{T+t}| \tag{2.29}$$

[Hyndman and Athanasopoulos 2021].

- **Mean Squared Error (MSE):** Penalizes larger errors more heavily due to the squaring of errors.

$$\text{MSE} = \frac{1}{H}\sum_{t=1}^{H}(y_{T+t} - \hat{y}_{T+t})^2 \tag{2.30}$$

[Hyndman and Athanasopoulos 2021].

- **Root Mean Squared Error (RMSE):** Provides the error in the same units as the target variable, making it more interpretable than MSE. Like MSE, it is sensitive to large errors.

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{H}\sum_{t=1}^{H}(y_{T+t} - \hat{y}_{T+t})^2} \tag{2.31}$$

[Hyndman and Athanasopoulos 2021].

- **Mean Absolute Percentage Error (MAPE):** Expresses error as a percentage of the actual values, which can be useful for comparing accuracy across time series with different scales.

$$\text{MAPE} = \frac{100}{H}\sum_{t=1}^{H}\left|\frac{y_{T+t} - \hat{y}_{T+t}}{y_{T+t}}\right| \% \tag{2.32}$$

[Hyndman and Athanasopoulos 2021]. MAPE is undefined if any $y_{T+t} = 0$ and can be problematic if actual values are close to zero.

- **Mean Absolute Scaled Error (MASE):** A scale-independent error measure that compares the forecast error against the average error of a naive one-step forecast on the training data [Hyndman and Athanasopoulos 2021]. It is robust to zero actual values and often preferred for general accuracy comparisons.

$$\text{MASE} = \frac{\frac{1}{H}\sum_{t=1}^{H}|y_{T+t} - \hat{y}_{T+t}|}{\frac{1}{L_{\text{train}}-m}\sum_{i=m+1}^{L_{\text{train}}}|y_i - y_{i-m}|} \tag{2.33}$$

Here, L_{train} is the length of the training series, and m is the seasonal period (or $m = 1$ for non-seasonal data).

- **Root Mean Squared Scaled Error (RMSSE):** Similar to MASE but based on squared errors, providing a scale-independent error metric sensitive to larger errors [Hyndman and Athanasopoulos 2021].

$$\text{RMSSE} = \sqrt{\frac{\frac{1}{H}\sum_{t=1}^{H}(y_{T+t} - \hat{y}_{T+t})^2}{\frac{1}{L_{\text{train}}-m}\sum_{i=m+1}^{L_{\text{train}}}(y_i - y_{i-m})^2}} \tag{2.34}$$

- **Cumulative Forecast Error (sum of errors, measures bias) (CFE) / Sum of Errors**: Measures the total bias of the forecast over the horizon H. It is the sum of all forecast errors.

$$\text{CFE} = \sum_{t=1}^{H} (y_{T+t} - \hat{y}_{T+t}) \tag{2.35}$$

  A positive CFE indicates that, on aggregate, the forecast has underestimated the actual values (demand), while a negative CFE suggests systematic overestimation. Values closer to zero indicate less overall bias [Hyndman and Athanasopoulos 2021].

- **Periods In Stock (PIS):** Measures the cumulative sum of forecast errors over time, indicating how long errors persist and their accumulated effect. It is particularly relevant for intermittent or sporadic time series and inventory management contexts. Let $e_j = y_{T+j} - \hat{y}_{T+j}$ be the forecast error at step j of the horizon. PIS is defined as the negative sum of the cumulative errors at each step k within the horizon H:

$$\text{PIS} = -\sum_{k=1}^{H} \left(\sum_{j=1}^{k} e_j \right) = -\sum_{k=1}^{H} \sum_{j=1}^{k} (y_{T+j} - \hat{y}_{T+j}) \tag{2.36}$$

  A positive PIS value suggests an accumulated overstock resulting from forecasts that were, on average, too high cumulatively over time, while a negative PIS value indicates an accumulated understock which means forecasts are cumulatively too low. Values closer to zero are preferable, indicating that cumulative errors tend to cancel out over the horizon [Syntetos, Boylan, and Croston 2010].

Probabilistic Forecast Metrics

For probabilistic forecasts, metrics assess the quality of the entire predictive distribution, considering both calibration (reliability of probability statements) and sharpness (concentration of the distribution).

- **Weighted Quantile Loss (WQL)** or **Pinball Loss (PL)**: This metric evaluates the accuracy of predicted quantiles. The quantile loss or pinball loss for a specific quantile level $\alpha \in (0, 1)$, given a quantile forecast q and an actual value x, is defined as:

$$PL_\alpha(x, q) = \begin{cases} \alpha(x - q), & \text{if } x > q \\ (1 - \alpha)(q - x), & \text{if } x \leq q \end{cases} \tag{2.37}$$

  WQL is typically the average of these losses over multiple quantiles and forecast horizons, often with appropriate weighting for different quantiles or horizons [Gasthaus et al. 2019; Gneiting and Raftery 2007].

- **Continuous Ranked Probability Score (CRPS):** A strictly proper scoring rule that measures the difference between the predicted Cumulative Distribution Function (CDF) F and the empirical CDF of the observed value y which

is a step function at y. It generalizes the MAE to probabilistic forecasts.

$$\text{CRPS}(F, y) = \int_{-\infty}^{\infty} (F(x) - \mathbf{1}\{x \geq y\})^2 dx \tag{2.38}$$

[Gneiting and Raftery 2007]. Here, $\mathbb{1}\{x \geq y\}$ is an indicator function.

- **Mean Scaled Interval Score (MSIS):** Evaluates the quality of prediction intervals. It considers both the coverage (whether the interval contains the true value) and the width of the interval, penalizing intervals that are too wide or those that miss the actual value. It is scaled similarly to MASE for comparability across series [Hyndman and Athanasopoulos 2021].

The choice of metric should align with the specific goals of the forecasting task and the properties of the data. For instance, MASE and RMSSE are preferred for their scale independence and robustness, while WQL and CRPS are essential for evaluating the quality of probabilistic forecasts.

2.5. Exogenous Variables in Time Series Forecasting

The inclusion of external information, in the form of exogenous variables, can significantly enhance the accuracy and interpretability of time series forecasts. This section defines exogenous variables, discusses their importance, categorizes them, outlines common integration methods in established models and addresses the inherent challenges in their use.

2.5.1. Definition and Significance of Exogenous Variables

In TSF, *exogenous variables*, also termed external factors, exogenous variables, predictors, or explanatory variables, are external factors that influence the behavior of the primary variable of interest (named endogenous or target variable) but are, within the model's defined scope, not affected by it [De Gooijer and Hyndman 2006; Y. Wang, H. Wu, Dong, Qin, et al. 2024]. For consistency within this thesis, 'exogenous variables' will be the primary term used, with 'exogenous variables' sometimes used as a direct synonym where contextually appropriate. The distinction is paramount for correct model specification: endogenous variables are determined by other variables within the system including their own past, while exogenous variables act as independent inputs. The nature of exogeneity is model-dependent and context-specific. For example, interest rates might be exogenous when forecasting a single company's sales but endogenous in a macroeconomic model.

Mathematically, if y_t is the endogenous variable and $x_t = (x_{1t}, x_{2t}, \ldots, x_{Mt})^T$ is a vector of M exogenous variables, a general model might be $y_t = f(y_{t-1}, y_{t-2}, \ldots, x_t, x_{t-1}, \ldots, \epsilon_t)$. A critical assumption for strict exogeneity is $E[x_{jt}\epsilon_s] = 0$ for all predictors j and all relevant time periods t, s.

The significance of incorporating relevant exogenous variables includes:

- **Improved Forecasting Accuracy**: Exogenous variables can provide additional context and explanatory power not captured by the endogenous series'

own dynamics alone. For example, electricity prices are often linked to weather conditions, fuel prices and broader market conditions [Nowotarski and Weron 2018].

- **Enhanced Understanding**: They allow for the exploration of correlations and potential causal relationships, leading to better model performance and richer interpretability of the system being modeled.

- **Handling Non-stationarity and Complexity**: Trends and seasonal patterns in an endogenous variable can sometimes be explained or captured by corresponding patterns in external factors e.g. a sales trend influenced by population growth.

- **Policy Evaluation and Scenario Analysis**: Models incorporating exogenous variables are indispensable for assessing the potential impacts of policy changes or for forecasting under hypothetical future conditions for these external factors.

The true benefit of exogenous variables arises when they contribute unique, predictive information not already encapsulated in the past behavior of the endogenous variable itself [Hamilton 1994].

2.5.2. Types of Exogenous Variables

Exogenous variables, or exogenous variables, provide auxiliary information that can enhance time series forecasts. They can be categorized based on their time-dependence, their data type, and the availability of their future values. A primary distinction is made between static, dynamic, and categorical exogenous variables.

- **Static Exogenous Variables**: A static exogenous variable is a feature or attribute that remains constant over the entire observational window and the forecast horizon. Formally, if s_i denotes the i-th static exogenous variable, then its value $s_i(t)$ is the same for all time points t. These exogenous variables capture time-invariant characteristics, often distinguishing between different entities or groups in panel data settings. *Examples include:*
  - Product attributes like category (e.g., "electronics," "clothing"), brand ID, or date of launch (which is fixed once launched).
  - Store-specific features like location (e.g., ZIP code, region) or store size, when forecasting for multiple stores.
  - Subject characteristics in longitudinal studies (e.g., demographic group).

  Static exogenous variables are crucial for modeling systematic differences in level, trend, or seasonality across different time series in a dataset.

- **Dynamic Exogenous Variables**: A dynamic (or time-varying) exogenous variable is any variable whose value may change at each time point t. For forecasting, a critical aspect of dynamic exogenous variables is the availability of their future values:

  - *Future-Known Dynamic Exogenous Variables (Deterministic):* These are dynamic exogenous variables whose future values are known or can be perfectly determined in advance. This category includes:
    * Calendar-based variables like time trends (e.g., a simple index t), specific dates, day-of-the-week, month-of-year, or holiday indicators (e.g., a binary flag for Christmas). Bergmeir [2020] refers to these as seasonal indicators or holiday effects, often one-hot encoded or represented via distance maps.
    * Pre-announced events or policy changes (e.g., a planned promotional period, a known tax rate change).
    * Operational variables like the number of trading days in a month.
  - *Future-Unknown Dynamic Exogenous Variables (Stochastic):* These are dynamic exogenous variables whose future values are uncertain and must themselves be forecasted if they are to be used as predictors for future values of the target variable. This introduces additional modeling complexity and uncertainty. Examples include:
    * Weather forecasts (e.g., temperature, precipitation).
    * Economic indicators (e.g., GDP growth, inflation rates, stock prices).
    * Competitor actions (e.g., pricing changes not known in advance).

  Dynamic exogenous variables can influence the target variable contemporaneously or with a lag.

- **Categorical Exogenous Variables**: A categorical exogenous variable takes values from a finite, discrete set of labels or categories, $\mathcal{C} = \{C_1, \ldots, C_K\}$. These are non-numeric by nature and require encoding (e.g., one-hot encoding, dummy variables, or learned embeddings) before being used in most forecasting models. Categorical exogenous variables can be either static or dynamic:
  - *Static Categorical Exogenous Variables:* The category label is constant over time for a given series. Examples: product segment (e.g., "premium," "economy"), geographic region (e.g., "North," "West").
  - *Dynamic Categorical Exogenous Variables:* The category label can change over time. Examples: day-of-the-week (e.g., "Monday," "Tuesday"), season (e.g., "Spring," "Summer"), or binned hour of the day (e.g., "morning," "afternoon"). Event flags (e.g., "promotion active") can also be considered dynamic categorical (often binary).

Furthermore, dynamic exogenous variables (both numeric and categorical-encoded) can be characterized by their temporal relationship to the target variable y_t:

- **Contemporaneous Exogenous Variables**: Values of x_t that affect y_t at the same time period t.
- **Lagged Exogenous Variables**: Past values of exogenous variables (x_{t-l} for $l > 0$) that affect the current y_t.

- **Lead Exogenous Variables (Future-Known component of dynamic exogenous variables)**: Future values of x_{t+k} (for $k > 0$) that are known at time t and can be used to predict y_t or y_{t+h}. This primarily applies to the deterministic dynamic exogenous variables mentioned above.

Understanding these different types of exogenous variables is essential for appropriate feature engineering, model selection, and handling the nuances of their future availability for forecasting.

2.5.3. Common Methods for Integrating Exogenous Variables in Established Models

Statistical Models

- **Dynamic Regression Models (Regression with ARIMA Errors)**: The model takes the form $y_t = x_t^T \beta + \eta_t$, where x_t is the vector of exogenous variables which can be contemporaneous or lagged and η_t is an ARIMA process that captures the serial correlation in the error term [Hyndman and Athanasopoulos 2021]. Typically, all variables y_t and individual components of x_t must be stationary or be differenced to achieve stationarity before model fitting.

- **Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX)/Seasonal Autoregressive Integrated Moving Average with Exogenous Variables (SARIMAX) Models**: Exogenous variables x_t are directly included in the ARIMA equation for y_t or its differenced form. For example, an ARIMAX model can be written as:

$$\phi(B)(1 - B)^d y_t = \delta(B)x_t + \theta(B)\epsilon_t \tag{2.39}$$

  where $\phi(B), \theta(B)$ are the AR and MA polynomials and $\delta(B)$ is a polynomial for the exogenous variable(s) x_t [Chatfield 2005].

- **Transfer Function (TF) Models**: These models describe the dynamic relationship where an output series Y_t responds to an input series X_t via a rational distributed lag structure $\nu(B) = \frac{\omega(B)}{\delta(B)} B^b$. The model is $Y_t = C + \nu(B)X_t + N_t$, where N_t is typically an ARIMA noise process, C is a constant, and b is a delay parameter [Chatfield 2005].

- **Vector Autoregressive Model with Exogenous Variables (VARX)**: For a vector of k multivariate endogenous variables y_t, the model is:

$$y_t = \nu + \sum_{i=1}^{p} A_i y_{t-i} + \sum_{j=0}^{s} B_j x_{t-j} + u_t \tag{2.40}$$

  where ν is a $k \times 1$ intercept vector, A_i are $k \times k$ parameter matrices for lagged y_t, B_j are $k \times M$ parameter matrices for contemporaneous and lagged exogenous variables x_t (an $M \times 1$ vector) and u_t is a $k \times 1$ white noise error vector. Exogenous variables x_t are assumed to affect y_t but not vice-versa within the model specification [Lutkepohl 2005].

- **SSM / Structural Time Series Model (STSM)**: Exogenous variables x_t can be flexibly included in the observation equation (e.g., $y_t = Z_t\alpha_t + D_t x_t + \epsilon_t$) for a direct, contemporaneous effect on y_t, or in the state equation (e.g., $\alpha_{t+1} = T_t\alpha_t + C_t x_t + R_t\eta_t$) to influence latent components like the trend or seasonality [Durbin and Koopman 2012].

Classical Machine Learning Models

As discussed in Section 2.3.2, classical ML models such as SVRs, RFs, and GBMs incorporate exogenous variables by including them as additional features in the input matrix X. This augmented feature set, which also typically includes lagged endogenous variables and other engineered time-based features, is then used to predict the target variable y_t. For instance, GBMs are noted to be particularly well-suited for handling diverse exogenous variables that may be on different scales (e.g., sales data, weather information, pricing details) [Bergmeir 2020].

Deep Learning Models

Deep learning models offer flexible ways to incorporate various types of exogenous variables, often learning complex interactions automatically.

- **RNNs, LSTMs, GRUs**: Exogenous variables are typically concatenated with the endogenous variable(s) at each time step to form the input vector $z_t = [y_t, x_{1t}, \ldots, x_{Mt}]$ (or their lagged versions) that is fed into the recurrent layers [Lim and Zohren 2021]. Alternatively, separate embedding layers can be used for endogenous and exogenous features before they are combined and processed by the recurrent architecture [Lim, Arık, et al. 2021]. Static exogenous variables (e.g., store ID, product category) might be incorporated by feeding them into the initial hidden state of the RNN or by concatenating them with the dynamic inputs at each time step, often after passing them through an embedding layer if they are categorical. Time-varying exogenous variables known in advance (e.g., scheduled promotions) and those observed only in the past (e.g., historical weather data) are usually included in the input vector z_t for the corresponding time steps they affect or are relevant to.

- **NBEATSx (Neural Basis Expansion Architecture with Exogenous Variables)**: NBEATSx handles exogenous variables by distinguishing between static and time-varying types [Olivares, Challu, et al. 2023]. Static exogenous variables (e.g., region or category IDs) are incorporated as fixed features or embeddings; they do not typically pass through time-convolutional layers but can influence outputs via shared parameters or by providing contextual biases to the network blocks. Time-varying exogenous variables (e.g., calendar features, external forecasts) are first fed into a dedicated convolutional encoder (such as a Temporal Convolutional Network (TCN)), which processes their historical patterns and temporal ordering. The resulting encoded context vector from this encoder is then combined with the standard N-BEATS block outputs, for instance, by being added before the final output layer of a block or used to modulate the block's forecast. This means the influence of dynamic exogenous variables on the forecast is learned via these specialized convolutional sub-networks. In the interpretable configuration of

NBEATSx, exogenous effects might also be modeled via additional, learned basis functions. The target series (endogenous history) is processed by the original NBEATS-style blocks, while exogenous inputs have a distinct pathway through their dedicated encoders before their information is fused. Static exogenous variables are effectively integrated through parameter sharing or embeddings that condition the model's behavior globally or at the block level.

- **TiDE (Time-series Dense Encoder)**: TiDE processes exogenous variables by first projecting dynamic exogenous variables at each time point to a lower-dimensional embedding using a small residual MLP (feature projection) [Das, Kong, Leach, et al. 2023]. Static exogenous variables (time-invariant features) are concatenated directly into a large, flattened input vector for the main encoder. This flattened vector also includes the past target history (endogenous values) and the embeddings of past dynamic exogenous variables. If future values of dynamic exogenous variables are known, their embeddings are also included in this flattened encoder input. Crucially, TiDE does not employ separate architectural pathways for endogenous versus exogenous data after the initial projection stage; all (projected) exogenous variables and the target history are fed into the same sequence of dense MLP layers in the encoder. In the temporal decoder, the embeddings of known future exogenous variables for a specific future time step are concatenated with the decoded feature vector for that step, allowing a direct "highway" for future exogenous variables to influence the corresponding prediction. Thus, exogenous variable effects are learned jointly with the target dynamics through the shared MLP representations rather than through specialized modules.

- **Transformer Models (General Integration and Specific Architectures)**: Beyond generic concatenation, more sophisticated integration is common in Transformer-based models. This can involve separate embedding layers for endogenous and exogenous series, and specialized attention mechanisms like cross-attention to model interactions explicitly. For instance, TimeXer uses patch-level embeddings for endogenous series and variate-level embeddings for (potentially irregular) exogenous series, with an "endogenous global token" designed to bridge information flow [Y. Wang, H. Wu, Dong, Qin, et al. 2024]. FutureTST employs variate-wise cross-attention where endogenous encodings act as queries to exogenous encodings [Zheng et al. 2024].

- **TFT**: TFT explicitly separates exogenous variables into static and dynamic types, with further distinction for dynamic inputs into past-observed and future-known [Lim, Arık, et al. 2021]. Static features are passed through their own GRN encoder(s) to produce context vectors. These context vectors then condition various parts of the network, for example, by initializing the hidden states of LSTMs or by providing adaptive biases to other GRNs within the architecture. Dynamic inputs (both past-observed, including the target series history, and future-known exogenous variables like planned promotions) undergo a variable selection process. Each variable is first embedded, then passed through a GRN and a softmax layer to compute instance-specific importance weights. These weighted features are then summed to form a processed input vector at each time step, which is fed into the main temporal processing

pipeline (LSTM encoder-decoder followed by multi-head self-attention). Static contexts do not directly traverse the recurrent or attention layers but influence processing through conditioning. The target series' own history is treated as one of the past-observed inputs and goes through the same variable selection and encoding pathway as any exogenous time-varying series.

- **Hybrid Models**: These models combine statistical approaches with ML/DL techniques. Exogenous variables can be introduced into either or both components (e.g., an ARIMAX model for the linear part, with a Neural Network (NN) modeling the residuals using exogenous variables).

Specialized deep learning architectures like DeepAR [Gasthaus et al. 2019] and others are explicitly designed to handle various types of exogenous variables (static, known future, observed past) and may include internal mechanisms for variable selection or learning attentive feature representations [J. Kim et al. 2024].

2.5.4. Challenges in Using and Selecting Exogenous Variables

Despite their potential benefits, incorporating exogenous variables into forecasting models presents significant challenges:

- **Forecasting Future Exogenous Variables**: A primary challenge is that future values of stochastic exogenous variables are typically required to generate forecasts for the target variable itself. These future exogenous variable values must themselves be forecasted, which introduces additional modeling effort, uncertainty, and potential error propagation into the main forecast [Hyndman and Athanasopoulos 2021].

- **Data Irregularity and Heterogeneity**: Real-world external data often suffer from issues such as missing values, temporal misalignment (e.g., daily target series vs. monthly exogenous variable), differing frequencies of observation, and discrepancies in series length or start/end dates [Hyndman and Athanasopoulos 2021]. Specialized preprocessing techniques or model architectures capable of handling such irregularities are often necessary (e.g., TimeXer's variate embedding for irregular inputs [Y. Wang, H. Wu, Dong, Qin, et al. 2024]).

- **Causality versus Correlation**: Statistical models primarily capture correlations between variables. A strong correlation observed in historical data does not necessarily imply a true causal effect. It could be spurious (e.g., due to common underlying trends or unobserved confounding variables) [Bergmeir 2020]. Relying on such spurious correlations can lead to poor out-of-sample forecasting performance. Granger causality tests assess predictive precedence rather than philosophical causality and have limitations, including sensitivity to lag selection and omitted variable bias.

- **Endogeneity**: An "exogenous" variable might, in reality, be endogenous, i.e., correlated with the model's error term. This can occur due to simultaneity (where the target variable also influences the exogenous variable), omitted relevant variables that affect both, or measurement error in the exogenous variable

[Hamilton 1994]. Endogeneity leads to biased and inconsistent parameter estimates in many statistical models. Techniques like Instrumental Variables (IV) estimation can address this but require finding valid instruments, which can be difficult [Lima et al. 2025].

- **Multicollinearity**: High correlation among exogenous variables makes it difficult to disentangle their individual effects on the target variable and can lead to unstable or unreliable parameter estimates [Y. Wang, H. Wu, Dong, Qin, et al. 2024]. Detection methods include examining correlation matrices or using the Variance Inflation Factor (VIF). Mitigation strategies involve feature selection, Principal Component Analysis (PCA), or using penalized regression methods like Lasso or Ridge regression [Lima et al. 2025].

- **Curse of Dimensionality and Overfitting**: Including a large number of exogenous variables, especially if many are weakly relevant or redundant, increases model complexity and the risk of overfitting. Overfitting occurs when the model captures noise in the training data rather than the true underlying signal, leading to poor generalization to unseen data [Lim and Zohren 2021].

- **Variable Selection and Feature Engineering**: Identifying the truly relevant subset of exogenous variables from a potentially vast pool, and engineering them into an appropriate form, is a critical and non-trivial task [Lima et al. 2025]. Methods range from statistical tests and information criteria (e.g., AIC, BIC) to regularization techniques (e.g., Lasso, which can shrink coefficients of irrelevant features to zero), filter methods (based on correlation or mutual information), wrapper methods (e.g., Recursive Feature Elimination (RFE)), and embedded methods (e.g., tree-based feature importance, Lasso) [Lima et al. 2025]. Domain expertise is often invaluable in guiding this process.

- **Model Interpretability and Explainable AI (XAI)**: As models, especially ML and DL approaches, incorporate many exogenous variables, they can become "black boxes," making it difficult to understand how specific external factors influence the forecasts. While classical statistical models often offer coefficient analysis or impulse response functions for interpretation, ML/DL models typically require post-hoc XAI techniques such as SHapley Additive exPlanations (SHAP), Local Interpretable Model-agnostic Explanations (LIME), or permutation feature importance to gain insights [Lim, Arık, et al. 2021]. It is also noted that attention weights in Transformer models are not always reliable indicators of feature importance for interpretation [Lima et al. 2025].

- **Uncertainty Quantification**: Accurately estimating Prediction Intervalss (PIs) must account for all sources of uncertainty, including the uncertainty arising from forecasted exogenous variables [Gasthaus et al. 2019]. Overlooking this often leads to PIs that are too narrow and provide a false sense of confidence. Analytical PIs for models with exogenous variables frequently assume that future exogenous variable values are known. Simulation-based methods, bootstrapping, or Bayesian approaches (e.g., as used in Prophet) can offer more robust ways to propagate this uncertainty. Deep probabilistic models like DeepAR aim to directly predict the entire probability distribution of the

target variable, implicitly handling some of these uncertainties [Gasthaus et al. 2019].

Addressing these challenges effectively is crucial for the robust and reliable application of exogenous variables in TSF. The multifaceted challenges associated with exogenous variable usage underscore the need for more advanced and flexible modeling paradigms. FMs, which are introduced in the next section and form the core of this thesis, offer potential avenues to mitigate some of these difficulties. For instance, their ability to learn rich representations from vast and diverse datasets might improve generalization and reduce the reliance on manual feature engineering for exogenous variables, while their large capacity could enable more sophisticated modeling of complex interactions between endogenous series and various types of external factors. Chapter 3 will delve into how specific FM architectures aim to address these opportunities in the context of TSF with exogenous variables.

2.6. Foundation Models

2.6.1. Concept of Foundation Models

FMs represent a paradigm shift in artificial intelligence, characterized by their immense scale and broad applicability across diverse tasks. They are typically **large-scale models**, often with billions of parameters, pre-trained on vast and heterogeneous datasets, which allows them to learn **general-purpose representations** and capabilities [Kottapalli et al. 2025]. This pre-training phase enables the model to acquire a deep understanding of patterns, structures, and semantic relationships within the data, going beyond specific task-driven objectives. The core idea is that such a broadly trained model can then be **fine-tuned** or adapted with minimal effort (e.g., through prompt engineering or few-shot learning) to perform a wide range of downstream tasks, rather than requiring a separate model to be trained from scratch for each specific application [Géron 2019]. Indeed, a significant manifestation of this adaptability is their capacity for **zero-shot learning**, where the model can perform new tasks based on instructions or by recognizing patterns analogous to those seen during pre-training, without requiring any task-specific examples or fine-tuning for that new task [Kottapalli et al. 2025]. This zero-shot capability further underscores the model's generalization prowess. The power of FMs also lies in their **emergent capabilities** behaviors or functionalities that are not explicitly programmed but arise from the scale of the model and its training data, allowing them to perform tasks for which they were not explicitly trained . This paradigm has reshaped fields like NLP and CV, where models such as Generative Pre-trained Transformer 3 (GPT-3) and Bidirectional Encoder Representations from Transformers (BERT) for text, or Contrastive Language-Image Pre-training (CLIP) and DALL-E for images, have demonstrated remarkable versatility and performance across various applications [Brown, Mann, Ryder, Subbiah, Jared Kaplan, et al. 2020; Devlin et al. 2019; Radford et al. 2021; Ramesh et al. 2021]. The potential for such generalized intelligence has naturally extended the inquiry into other complex data domains, including time series.

2.6.2. Relevant Architectural Principles

The impressive capabilities of FMs are largely attributable to advancements in neural network architectures. The **Transformer architecture** [Vaswani et al. 2017] has been particularly influential. Introduced primarily for sequence-to-sequence tasks in NLP, Transformers revolutionized the field by leveraging **self-attention mechanisms**. Unlike RNNs or CNNs, which process sequences sequentially or locally, Transformers can process all elements of a sequence in parallel. This allows them to capture **long-range dependencies** efficiently, regardless of their position in the sequence. This parallel processing capability also facilitates **massive scaling** during training, making it feasible to train models with billions of parameters on vast datasets. **Positional encodings** are crucial for Transformers to incorporate information about the order or position of data points, which is inherently important for sequential data [Kottapalli et al. 2025].

Beyond the Transformer, other architectural principles are gaining prominence. **MLP-based architectures** represent a simpler, yet potentially effective, class of models that have seen a resurgence. While often considered less complex than Transformers, advancements in designs like MLP-Mixers have shown that pure MLPs can achieve strong performance on sequential data by processing data in patches or segments and using channel-mixing or token-mixing layers [Ekambaram, Jati, Nguyen, et al. 2023]. These architectures can offer computational efficiency and straightforward parallelization, making them attractive for certain FM applications, especially in domains with specific data structures like time series.

More recently, **SSMs** and their specialized variants like **Mamba** have emerged as powerful alternatives for sequence modeling [Gu and Dao 2023]. SSMs process sequences by maintaining a hidden state that is updated at each step, offering efficiency in handling very long sequences due to their linear complexity and ability to capture long-range dependencies effectively without relying on attention mechanisms. Mamba, in particular, incorporates selective mechanisms that allow it to dynamically filter information, making it highly efficient and expressive for diverse sequential data tasks, including potential applications in time series analysis where long temporal contexts are crucial.

Diffusion models define a generative backbone via a pair of Markov processes: a *forward noising* chain that gradually corrupts data into Gaussian noise and a learned *reverse denoising* chain that restores clean signals [Ho et al. 2020]. Architecturally, the reverse network—often a U-Net or Transformer—estimates either the original data or its score (gradient of the log–density) at each timestep, enabling iterative generation without autoregression. Unlike attention-based models, diffusion backbones process the entire sequence at each denoising step in parallel, naturally capturing long-range interactions and multimodal uncertainties. Conditioning on past observations or exogenous inputs is straightforward: one simply concatenates or cross-attends these contexts at every denoising layer [Salimans et al. 2020].

Additionally, techniques like **Mixture of Experts (MoE)** layers contribute to the performance and scalability of FMs [Shazeer et al. 2017]. MoE allows a model to selectively activate different "expert" sub-networks (often MLPs or Transformer blocks) for different inputs or tokens, enhancing model capacity without proportionally increasing the computational cost for every input. The combination and refinement of these architectural innovations provide the structural backbone for the learning and generalization abilities observed in state-of-the-art FMs.

This chapter has traced the evolution of forecasting methodologies, from classical statistical models designed for individual series to sophisticated deep learning architectures. The introduction of the FM paradigm, powered by scalable and expressive architectures like the Transformer and modern State Space Models, represents the latest frontier. This shift from specialized, task-specific models to large-scale, pre-trained systems presents both immense opportunities and unique challenges for the field of TSF. The following chapter will build directly upon this background, exploring how these powerful, general-purpose models are being specifically adapted to handle the complexities of temporal data and reviewing the current landscape of FMs for Forecasting.

3. Foundation Models for Forecasting with Exogenous Variable Integration

This chapter delves into the landscape of FMs specifically applied to TSF , with a particular focus on the diverse mechanisms employed for integrating exogenous variable information. It addresses Research Question F1 by exploring existing FMs and their technical realizations of exogenous variable handling, building upon the theoretical concepts established in Chapter 2. To achieve this, the chapter will first outline the potential and challenges of applying the FM paradigm to time series, then detail the crucial step of input tokenization, and provide a historical overview of the field's evolution, before proceeding to an analysis of key benchmarks and models.

3.1. The Foundation Model Paradigm for Time Series: Potential and Pitfalls

The success of FMs in domains like NLP and CV has spurred significant interest in applying this paradigm to TSF. Time series data, like language, possesses complex sequential dependencies and varying patterns that require an understanding of context. This section will first explore the potential benefits of applying the FM paradigm to TSF, focusing on generalization and the integration of diverse information sources. It will then address the fundamental challenges that differentiate the temporal domain from others, necessitating specialized approaches.

3.1.1. The Promise of Generalization and Reasoning

The primary appeal of FMs for TSF lies in their potential for **transferability**. A model pre-trained on a vast and diverse collection of time series could learn universal patterns, such as trend, seasonality, and cyclical behaviors, along with their underlying dynamics [Kottapalli et al. 2025]. Such a model could then be adapted to new, unseen time series with minimal fine-tuning, offering a compelling alternative to training specialized models for each individual dataset [J. Kim et al. 2024]. This paradigm shift reduces the need for extensive, task-specific feature engineering and promises improved performance in scenarios with sparse data or for cold-start forecasting problems where historical data is limited.

Beyond improving predictive accuracy through generalization, these large-scale models hold the promise of exhibiting more advanced cognitive capabilities. Drawing parallels from their success in NLP, where models can interpret complex instructions, time series FMs could potentially analyze anomalous patterns or infer relationships

between variables. This opens the door to more sophisticated forecasting tasks that require a deeper contextual understanding.

Furthermore, this paradigm naturally accommodates **multi-modal or multivariate understanding**, allowing exogenous variables to be seamlessly integrated into the model's representation [J. Kim et al. 2024; W. Wu et al. 2025]. Instead of being treated as simple numeric inputs, external factors—such as economic indicators, weather data, or even unstructured text from news reports—can be processed as another modality, providing richer context for the forecast. While advanced techniques like Chain of Thought (CoT) or Retrieval-Augmented Generation (RAG) represent emerging frontiers in this area, the core potential lies in creating a single, unified model that can reason across these different data types to produce more robust and context-aware predictions [Ning et al. 2025; Potosnak et al. 2025; X. Wang et al. 2024].

3.2. A Taxonomy of Foundation Models for Time Series Forecasting

The application of the FM paradigm to TSF has evolved rapidly. To understand the current landscape, it is useful to classify the approaches into three primary strategies. These strategies trace a path from adapting existing, powerful models to creating highly specialized, native time series systems, and finally to fusing different data modalities for a more holistic understanding.

3.2.1. Strategy 1: Adapting Pre-trained General-Purpose Models

The initial wave of research focused on repurposing models that were pre-trained on other domains, most notably Large Language Models (LLMs). The core idea was to leverage the immense scale and powerful in-context learning capabilities of models like Generative Pre-trained Transformer (GPT) by treating time series as a "language of numbers" [Ansari et al. 2024; J. Kim et al. 2024]. This typically involves flattening the historical time series window into a string of numerical text and feeding it to the LLM which is a form of **instance-level embedding**. The model is then prompted to extrapolate the sequence, relying on its pre-trained ability to parse and continue numerical patterns.

While promising, this strategy has been extended in more radical directions, questioning whether time series pre-training is necessary at all. A prominent example is **VisionTS**, which reframes forecasting as an image reconstruction problem. It employs a visual Masked Autoencoder pre-trained on ImageNet to perform zero-shot time series predictions, suggesting that the patterns learned from natural images have powerful temporal analogues [M. Chen et al. 2024]. Similarly, **TabPFN for Time Series** adapts a model pre-trained exclusively on synthetic *tabular* data. By converting a time series into a simple tabular format, it can outperform larger, specialized time series FMs, highlighting the power of general-purpose pattern recognition while avoiding the benchmark contamination risks associated with pre-training on common forecasting datasets [Hoo et al. 2025]. These approaches showcase the continued viability of leveraging powerful, existing FMs from other domains.

3.2.2. Strategy 2: Building Native Time Series Foundation Models

Recognizing that time series data possesses unique statistical properties, a significant shift occurred towards pre-training models from scratch exclusively on time series data. This second, now-dominant wave aims to create "native" TSFMs that learn the underlying patterns and temporal dynamics inherent to time series without the inductive biases of language or vision.

Much of the improvement in this area stems from building upon **advancements in deep learning architectures for TSF**, particularly the Transformer. These native FMs directly inherit and scale up architectural innovations like patching-based tokenization, which was shown in Chapter 2 to be effective for capturing local information while maintaining computational efficiency. The primary tokenization methods for this strategy are:

- **Patching:** The de facto standard, where a lookback window is divided into smaller sub-sequences (patches) that are independently embedded. This allows the model to process long histories efficiently.
- **Value Quantization:** An alternative where the continuous values of the series are discretized into a finite set of bins, which are then treated as tokens in a vocabulary, similar to words in NLP.

A critical bottleneck for this strategy, however, is **data scarcity**. Unlike NLP or CV, the time series domain lacks a universal, public corpus equivalent to the Common Crawl or ImageNet. Consequently, models like **Moirai** and **Sundial** are often pre-trained on a heterogeneous mix of proprietary, domain-specific, and synthetic datasets, making direct, reproducible comparisons challenging [Yong Liu, Qin, Shi, et al. 2025; Woo et al. 2024].

Despite this, innovation within this strategy is rapid. **Lag-Llama** and **TimeGPT-1** are foundational examples of applying scaled-up Transformer architectures to diverse time series corpora [Garza et al. 2023; Rasul et al. 2023]. Others push the boundaries further: **Sundial** introduces a "TimeFlow Loss" to enable true generative modeling of complex forecast distributions on continuous data without discretization [Yong Liu, Qin, Shi, et al. 2025]. **Yinglong** explores the reasoning process itself, using a "Delayed Chain of Thought" where later predictions help refine earlier ones [X. Wang et al. 2024]. Finally, **Toto**, developed for observability metrics, demonstrates the power of domain specialization, achieving state-of-the-art performance by combining a native architecture with a massive, albeit proprietary, in-domain dataset of over two trillion data points [Cohen et al. 2025].

3.2.3. Strategy 3: Fusing Modalities for Context-Aware Forecasting

The third and most sophisticated strategy seeks to combine the best of both worlds by creating multi-modal systems. These models explicitly address the need to incorporate complex exogenous information by fusing a native time series backbone with other data modalities, most commonly natural language. For example, a model like **ChatTime** can process numerical stock price data alongside textual financial news

reports or analyst ratings to create a more context-aware forecast [Ren et al. 2023]. By training a model to understand both the temporal dynamics of the series and the semantic content of related text, these systems can capture relationships that would be invisible to a uni-modal model. This approach represents a sophisticated synthesis of the previous strategies and provides a concrete framework for advanced techniques like RAG, where relevant external documents could be retrieved and fed to the model to dynamically inform the forecast.

3.2.4. Benchmarks for Time Series Foundation Models

The rigorous evaluation of FMs in TSF is crucial for understanding their capabilities and limitations. Several key benchmarks have recently emerged to facilitate this, each with specific focuses and contributions.

FoundTS

FoundTS, proposed by Li et al. [2024], is a benchmark designed to enable thorough and fair evaluation and comparison of TSF FMs. It addresses limitations of earlier benchmarks by being more comprehensive and user-friendly, featuring a wider range of TSF models and evaluation strategies. FoundTS covers a variety of TSF FMs, including those based on LLMs and those pre-trained on time series data. It supports different forecasting strategies, including zero-shot, few-shot, and full-shot, which facilitates more thorough evaluations. Furthermore, FoundTS offers a pipeline that standardizes evaluation processes such as dataset splitting, loading, normalization, and few-shot sampling, thereby promoting fair evaluations. The benchmark incorporates datasets spanning different domains and with diverse characteristics to assess model performance thoroughly. FoundTS aims to provide reliable insights into the characteristics, advantages, and disadvantages of time series FMs.

GIFT-Eval

GIFT-Eval (*General Time Series Forecasting Model Evaluation*), introduced by Aksu et al. [2024], is a pioneering benchmark aimed at promoting evaluation across diverse datasets for general TSF models. It is a widely utilized benchmark for FMs. GIFT-Eval encompasses 23 datasets, over 144,000 time series, and 177 million data points, spanning seven domains, ten frequencies, multivariate inputs, and prediction lengths ranging from short to long-term forecasts. To facilitate effective pre-training and evaluation of FMs, it also provides a non-leaking pre-training dataset containing approximately 230 billion data points. GIFT-Eval includes a comprehensive analysis of 17 baselines, which encompass statistical models, deep learning models, and FMs. It offers a taxonomy covering both time series characteristics and properties, providing evaluations for FMs with zero-shot support. The benchmark is designed to address gaps identified in existing TSF benchmarks by providing a wider and more extensive dataset collection.

BOOM

BOOM (*Benchmark of Observability Metrics*), introduced by Cohen et al. [2025], is a large-scale, open-source evaluation framework specifically designed to capture the

unique forecasting challenges posed by modern observability workloads. Released recently (as of 2025), BOOM aims to provide a uniquely realistic and comprehensive evaluation environment for forecasting models in this domain. The dataset comprises approximately 350 million observations across 2,807 distinct multivariate time series. Unlike existing multi-domain benchmarks, BOOM is composed entirely of real-world observability data, sourced exclusively from Datadog's telemetry and internal observability metrics. It captures diverse time series dynamics and challenging behaviors across several subdomains. BOOM adopts evaluation protocols similar to those in GIFT-Eval, though it is distinct in its scale (approximately twice as many time series points as GIFT-Eval) and its specific domain focus. BOOM's data and evaluation code are openly available, aiming to accelerate research in this important real-world application.

The proliferation of these specialized benchmarks highlights the growing recognition of FMs' potential in TSF, as well as the increasing need for standardized and, where appropriate, domain-specific evaluation environments to accurately assess their capabilities and limitations.

3.3. Presentation of Selected Foundation Models with Exogenous Variable Integration

Building upon the overview of FMs and their classification, this section delves into specific examples of FMs applied to TSF. The models presented have been selected based on their representation of diverse techniques for incorporating exogenous information, their prominence in recent literature, and the availability of architectural details.

To the best of my knowledge, the landscape of pre-trained FMs explicitly designed with dedicated mechanisms for exogenous variables is still nascent. Many FMs are capable of *multivariate* forecasting, a functionality that can be repurposed to include covariates by treating them as additional input channels. In this common workaround, the model also generates forecasts for these exogenous channels, which are then simply discarded. This thesis, however, consciously excludes such models from this analysis to focus specifically on architectures that are genuinely **covariate-aware** or present novel ideas for achieving this awareness. Consequently, the models discussed here are all examples of the *Strategy 2* (Native Time Series FMs) approach from the taxonomy, as this is where the most direct research into covariate integration is currently taking place. Each model is reviewed to understand its core architecture and, crucially, the specific mechanisms it employs to leverage exogenous variables, thereby providing concrete illustrations for the concepts outlined in Section 2.5.3 and directly addressing Research Question F1.

3.3.1. Tiny Time Mixers

Recent advancements in TSF have seen the emergence of large pre-trained models, promising strong zero-shot and few-shot learning capabilities. However, these models often come with significant computational costs and may overlook crucial aspects like cross-channel correlations and the influence of external factors [J. Kim et al. 2024]. Addressing these limitations, **Tiny Time Mixers (TTM)** have been introduced

as a family of compact, efficient and effective pre-trained models for multivariate TSF , with variants starting from as few as 1 million parameters [Ekambaram, Jati, Dayama, et al. 2024]. TTM encompasses a range of pre-trained models, including different releases (R1, R2 and research models), each offering various pre-defined configurations of context length and forecast horizon to suit diverse forecasting needs. This analysis delves into the architectural intricacies of TTM and examines its mechanisms for integrating exogenous variables. The overall architecture of TTM is depicted in Figure 3.1.

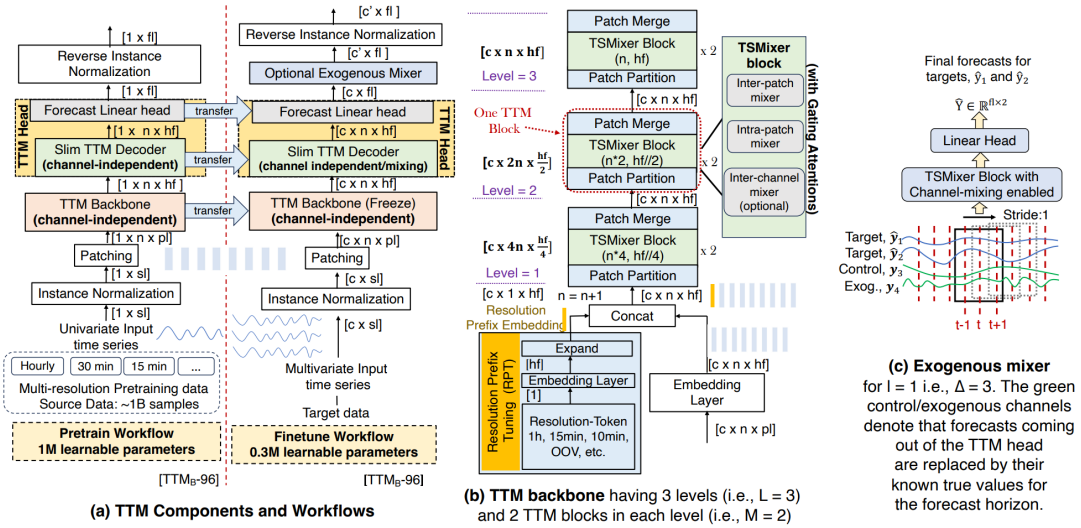


Figure 3.1.: Architectural overview of the TTM FM [Ekambaram, Jati, Dayama, et al. 2024].

Architectural Framework of TTM

TTM builds upon the lightweight TSMixer architecture, which itself is based on MLP blocks, offering an efficient alternative to computationally intensive Transformer models [Ekambaram, Jati, Nguyen, et al. 2023]. The TSMixer architecture provides a foundation of efficiency for sequence modeling. The core TTM architecture comprises several key innovations designed to handle diverse time series characteristics with minimal model capacity.

Core Components: The TTM model follows a multi-level architecture, consisting of four main components:

- **TTM Backbone:** This forms the primary learning module and is constructed using building blocks derived from the TSMixer architecture. A notable feature is **Adaptive Patching (AP)**, where different layers of the backbone operate at varying patch lengths and numbers of patches. This allows the model to generalize better when pre-training on datasets with different resolutions. The patched input data $X_p \in \mathbb{R}^{c \times n \times pl}$ (where c is channels, n is number of patches, and pl is patch length) is first passed through an embedding layer to project it to a hidden dimension $X_h \in \mathbb{R}^{c \times n \times hf}$ (where hf is the hidden feature dimension). The backbone consists of L_{levels} levels, each with M_{blocks}

TTM blocks. At each level i, a patch partition block adjusts the number of patches and patch dimension before the data is processed by a vanilla TSMixer block, followed by a patch merging block to revert to the original shape for that level. AP is confined to the backbone to keep the decoder lightweight.

- **TTM Decoder:** This component mirrors the backbone's architecture but is significantly smaller, typically 10-20% of the backbone's size. During fine-tuning, the decoder can be adjusted to incorporate channel mixing for multivariate target data or maintain channel independence for univariate data. If multivariate modeling is required, the channel-mixer block within the TSMixer components of the decoder is enabled to capture cross-channel correlations.

- **Forecast Head:** A linear head is used to produce the final point forecast output $\hat{Y} \in \mathbb{R}^{c' \times fl}$ (where c' is the number of forecast channels and fl is the forecast length). TTM is designed to provide point forecasts and does not natively generate probabilistic forecasts.

- **Optional Exogenous Mixer:** This module is specifically designed to fuse exogenous data into the forecasting process, particularly during fine-tuning.

The TTM decoder and forecast head collectively form the **TTM Head**, whose weights are updated during the fine-tuning process.

Preprocessing and Patching: Input time series $X \in \mathbb{R}^{c \times sl}$ where sl is sequence length, also referred to as context length undergoes instance normalization per channel, to zero mean and unit standard deviation, to address distributional shifts. This normalization is reversed before loss computation. The normalized data X is then divided into n non-overlapping patches (see Section 2.3.3) of length pl, resulting in $X_p \in \mathbb{R}^{c \times n \times pl}$.

Innovations for Resource-Constrained Pre-training: To effectively pre-train small models on heterogeneous, multi-resolution datasets, TTM introduces several architectural and training enhancements:

- **AP:** As described for the backbone, AP allows different layers to operate with varying patch lengths, aiding generalization across datasets with diverse resolutions and improving performance when pre-training data is limited.

- **Diverse Resolution Sampling (DRS):** This data augmentation technique addresses the scarcity of public datasets with diverse resolutions and potential biases towards finer resolutions. It involves creating lower-resolution datasets from high-resolution ones by averaging sequential, non-overlapping windows or by conventional decimation (retaining every k-th sample). This ensures more uniform coverage across resolutions and improves model performance.

- **Resolution Prefix Tuning (RPT):** This technique explicitly embeds resolution information into the model by prepending a learnable prefix (a new patch embedding based on the input resolution) to the input data. The resolution is mapped to an integer, passed through an embedding layer to get a hidden representation, and then expanded across channels to form a prefix of shape

$c \times 1 \times hf$. RPT helps the model condition its predictions on the resolution, especially beneficial for small models handling diverse datasets or when context length (sl) is short, making automatic resolution detection challenging.

Exogenous Variable Integration in TTM

TTM explicitly incorporates mechanisms to handle exogenous variables, a critical feature often lacking in other pre-trained time series models. The integration of exogenous variables primarily occurs during the fine-tuning stage via the **Exogenous Mixer Block** [Ekambaram, Jati, Dayama, et al. 2024].

The Exogenous Mixer Block: This module is activated when the target fine-tuning dataset contains exogenous variables whose future values are known throughout the forecast horizon. The process is as follows:

- Let the initial forecast from the forecast head be $Y' \in \mathbb{R}^{c \times fl}$. The channels $x_0, \ldots, x_{c'}$ are target variables, and $x_{c'+1}, \ldots, x_c$ are exogenous variables with known future values.
- The forecasted values for the exogenous channels in Y' are replaced with their true known future values (denoted as Y_{exog}). This results in a modified forecast tensor $\hat{Y}_e = [\hat{y}_0, \ldots, \hat{y}_{c'}, y_{c'+1}, \ldots, y_c] \in \mathbb{R}^{fl \times c}$ (after transposition).
- To capture inter-channel lagged correlations, $\hat{Y}_e$ is patched into overlapping windows using a stride of 1. This creates a new tensor $\hat{Y}_{e,p} \in \mathbb{R}^{fl \times \Delta \times c}$, where $\Delta = 2 \cdot l_{ctx} + 1$, with l_{ctx} being the context length on either side of a time point. Zero-padding of length l_{ctx} is applied to $\hat{Y}_e$ on both sides for this step.
- This patched tensor $\hat{Y}_{e,p}$ is then passed through a vanilla TSMixer block with channel-mixing explicitly enabled. This step allows the model to learn the lagged dependencies of the target channel forecasts on the exogenous channels.
- Finally, a linear head is attached to produce the refined forecasts for only the target channels, reshaped to $Y_{final} \in \mathbb{R}^{c' \times fl}$.

This dedicated mechanism allows TTM to seamlessly infuse information from exogenous variables, a practical necessity for many industrial forecasting applications. The multi-level modeling strategy, where the backbone is pre-trained channel-independently and the decoder/exogenous mixer handle channel correlations and exogenous signals during fine-tuning, is a key design choice enabling this functionality.

Pre-training and Fine-tuning Workflows

Pre-training: TTM is pre-trained on a large collection of diverse public datasets. For instance, TTM-R1 variants were pre-trained on approximately 250 million public training samples, while the more recent TTM-R2 models were trained on a significantly larger dataset of around 700 million samples. This pre-training uses a direct forecasting objective with MSE loss. Initially, pre-training is performed in a univariate fashion, with independent channels, due to the challenge of varied channel

counts in pre-training datasets. Multivariate datasets are transformed into independent univariate time series for this stage. The goal is to learn common temporal dynamics and seasonal patterns. Different TTM releases offer a variety of models pre-trained with specific context length and prediction length combinations [IBM Granite Team 2025].

Fine-tuning: For a target domain, TTM offers several options:

- **Zero-shot forecasting:** The pre-trained model is used directly on the target test data.
- **Few-shot forecasting:** A small portion (e.g., 5-10%) of the target training data is used to update the weights of the TTM Head (decoder and forecast head), keeping the backbone frozen.
- **Full-shot forecasting:** The entire training part of the target data is used to fine-tune the TTM Head, with the backbone remaining frozen.

During fine-tuning, if the target data is multivariate, the channel-mixer block in the TTM decoder is enabled to explicitly capture cross-channel correlations. If exogenous variables with known future values are present, the Exogenous Mixer block is also applied. The fine-tuning process also optimizes the MSE forecasting objective. It is generally recommended to select a TTM variant whose pre-training data characteristics and context/prediction length configuration best align with the target data, and experimentation with different variants may be necessary for optimal performance.

Performance and Efficiency

TTM has demonstrated strong performance, outperforming existing benchmarks in zero-shot and few-shot forecasting by significant margins (4-40%) while drastically reducing computational requirements [Ekambaram, Jati, Dayama, et al. 2024]. For instance, TTM can be as small as 1 million parameters, compared to hundreds of millions in larger models [Kottapalli et al. 2025]. This compactness allows TTM to be executed even on Central Processing Unit (CPU)-only machines, enhancing its usability in resource-constrained environments. The pre-training time is also notably fast, 24-30 hours with 6 Graphics Processing Unit (GPU) A100s for TTM-R1's 1 billion samples experimental training, compared to days or weeks for larger counterparts [Das, Kong, Sen, et al. 2023; Garza et al. 2023].

Capabilities and Limitations

TTM presents a compelling balance of efficiency and effectiveness, offering several notable capabilities alongside some inherent limitations.

Capabilities:

- **Compactness and Efficiency:** With models starting from as few as 1 million parameters, TTMs are significantly smaller and computationally less demanding than many other FMs, allowing for faster pre-training and inference.

- **Strong Zero-shot and Few-shot Performance:** TTMs have demonstrated robust generalization to new datasets with minimal or no fine-tuning.
- **Exogenous Variable Integration:** The dedicated Exogenous Mixer Block provides an explicit mechanism to incorporate known future exogenous variables during fine-tuning.
- **Handling of Diverse Resolutions:** Innovations like AP, DRS, and RPT enable effective pre-training and generalization across time series with varying granularities.
- **Range of Models:** TTM is available in different releases and configurations, offering various context lengths and prediction lengths to suit different tasks.

Limitations:

- **Point Forecasts Only:** TTM is designed to produce point forecasts and does not natively support probabilistic forecasting to quantify uncertainty.
- **Granularity Constraints:** The performance of TTM can be sensitive to the granularity of the time series. Early versions (R1) were primarily pre-trained on minutely and hourly data. While later versions (R2.1) expanded this to include daily and weekly granularities, forecasting on other resolutions (e.g., monthly, yearly) may yield suboptimal results, particularly if the available context length of a chosen TTM variant is insufficient to capture relevant long-term patterns for these lower frequencies.
- **Context Length Sensitivity:** While TTM models are designed for specific context lengths, they can technically process inputs shorter than their intended context window. However, doing so is not recommended as it typically leads to a degradation in forecasting performance. Users should select a model variant whose context length aligns with their available historical data.
- **Model Selection:** With multiple releases and various configurations, selecting the optimal TTM variant for a specific target dataset may require some experimentation.

Discussion on Design Choices for Compactness and Effectiveness

Several key design choices contribute to TTM's ability to achieve high accuracy with a small model capacity:

- **Focus on Resolution Diversity over Sheer Data Volume:** TTM's development highlighted that "limited" pre-training data with "high resolution diversity" (achieved via DRS) is more crucial for generalization in time series FMs than simply increasing pre-training data size. This allowed TTM R1 to be pre-trained effectively on approximately 250 million samples (the 1 billion sample figure mentioned in an earlier iteration was for a larger experimental setup perhaps, original TTM-R1 paper points to 250M, R2 to 700M).

- **TSMixer as a Lightweight Base:** Opting for the TSMixer architecture, which uses MLPs and simple mixing components instead of quadratic self-attention blocks, drastically reduced model size and computational complexity. TSMixer has previously shown that interleaving mixing components across patches, channels, and features can achieve strong forecasting results with limited capacity [Ekambaram, Jati, Nguyen, et al. 2023].

- **Innovative Enhancements for Small Models:** Architectural innovations like AP, DRS, and RPT were crucial for enabling small TSMixer-based models to effectively learn from large, heterogeneous, multi-resolution pre-training datasets without significantly increasing model size.

- **Direct Forecasting Objective:** Pre-training TTM with a direct forecasting objective, rather than traditional masking-based approaches, was found to improve zero-shot performance. This likely enables the model to learn robust non-linear mappings between context and forecast windows that generalize well.

- **Multi-Level Modeling for Specialized Tasks:** The separation of concerns, with the backbone focusing on general temporal patterns and the head handling target-specific aspects like channel correlation and exogenous data infusion during fine-tuning, allows for efficient adaptation and specialization of the compact model.

In summary, TTM represent a significant step towards democratizing pre-trained models for TSF. By leveraging a lightweight base architecture and incorporating novel techniques for handling data diversity and exogenous information, TTMs achieve state-of-the-art performance with substantially reduced computational overhead and model size. Their design philosophy emphasizes the quality and diversity of pre-training data and specialized architectural components over massive parameter counts, paving the way for wider adoption in practical, resource-constrained settings.

3.3.2. Chronos

Chronos has emerged as a significant development in the field of TSF , proposing a novel framework for pre-trained probabilistic models [Ansari et al. 2024]. This analysis delves into the core architecture of Chronos, including its recent efficient variant Chronos-Bolt [Amazon Science n.d.(a)], and subsequently explores the methodologies employed for integrating external exogenous variables with the framework. An overview of how the Chronos architecture is processing time seires data is presented in Figure 3.2.

Core Architectural Framework of Chronos

The original Chronos distinguishes itself by adapting principles from LLMs to the domain of TSF . Its architecture is characterized by simplicity and effectiveness, built upon two primary innovations: time series tokenization and the leveraging of existing transformer-based language model structures.

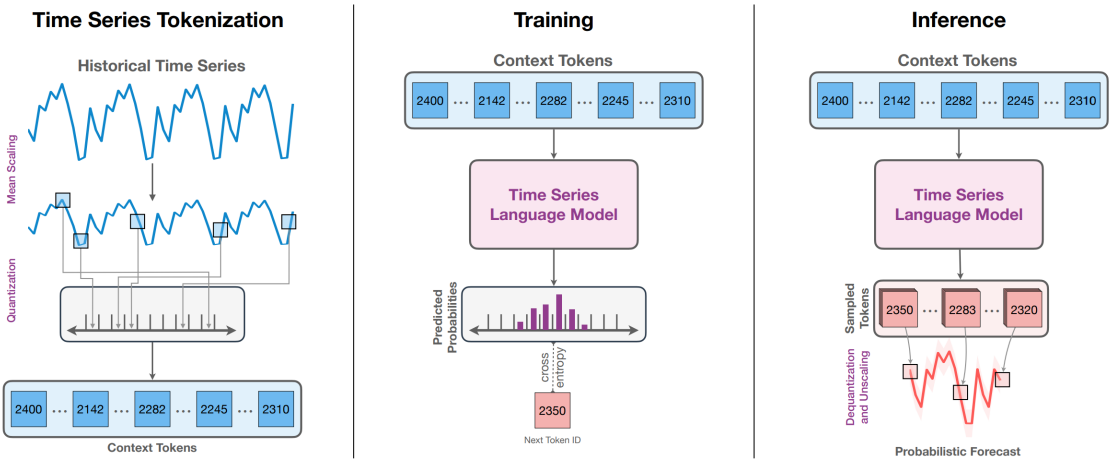


Figure 3.2.: Architectural overview of the Chronos FM [Ansari et al. 2024].

Data Preprocessing: Scaling and Quantization The foundational step in Chronos is the transformation of continuous real-valued time series data into a sequence of discrete tokens, akin to words in a sentence. This process involves two main sub-steps:

- *Scaling:* Each input time series $z_i = (z_{i,1}, z_{i,2}, \ldots, z_{i,L_i})$ within a batch, where L_i is the length of the series, undergoes a scaling procedure. Chronos employs mean scaling. The scaled value $x_{i,t}$ for an observation $z_{i,t}$ is computed as:

$$x_{i,t} = \frac{z_{i,t}}{\mu_i} \quad \text{where} \quad \mu_i = \frac{1}{L_i} \sum_{j=1}^{L_i} |z_{i,j}| + \epsilon \tag{3.1}$$

  The small constant ϵ (e.g., 10^{-6}) is added for numerical stability, particularly for series with many zero values. This scaling method preserves the sign and relative magnitudes within the series while normalizing its overall scale.

- *Quantization:* The scaled time series values $x_{i,t}$ are then quantized into a fixed number of discrete bins, N_B. These bins constitute the vocabulary of the model. The original Chronos model typically employs a vocabulary size (N_B) of 4096 bins. The k-th bin is defined by its edges $[b_k, b_{k+1})$, where $k \in \{0, \ldots, N_B - 1\}$. The bin edges are typically determined based on the percentiles of the aggregated scaled values from the entire training dataset, ensuring a balanced distribution of tokens and effective representation of the data range [Ansari et al. 2024].

Modeling: Time Series as a Language Once the time series is tokenized, Chronos reframes the forecasting task as a language modeling problem. Given a sequence of context tokens $y_{1:C} = (y_1, y_2, \ldots, y_C)$ derived from the historical time series, the objective is to predict the sequence of future tokens $y_{C+1:C+H} = (y_{C+1}, \ldots, y_{C+H})$ for a desired prediction horizon H.

- *Transformer Architecture:* Chronos utilizes existing transformer-based language model architectures, such as those from the Text-to-Text Transfer Transformer (T5) family [Raffel et al. 2019]. These models, whether encoder-decoder

or decoder-only, are well-suited for sequence-to-sequence tasks due to their self-attention mechanisms that can capture long-range dependencies.

- *Probabilistic Forecasting and Loss Function:* The model is trained to output a probability distribution over the vocabulary for each future token, conditioned on the preceding tokens. That is, it models $p_\phi(y_t \mid y_{1:t-1})$, where ϕ represents the model parameters. The training objective is to minimize the negative log-likelihood, which corresponds to the standard cross-entropy loss function used in language modeling:

$$\mathcal{L}(\phi) = -\sum_{i \in \text{Dataset}} \sum_{t=C+1}^{C+H} \log p_\phi(y_{i,t} \mid y_{i,1:t-1}) \tag{3.2}$$

- *Prediction:* During inference, future tokens are typically sampled autoregressively from the predicted probability distributions. To obtain real-valued forecasts, these sampled tokens are de-quantized (e.g., by taking the center of the corresponding bin) and then de-scaled by multiplying with the original scaling factor μ_i. For probabilistic forecasts, the full predicted distribution over the tokens for each future time step can be transformed back into a distribution over the real values.

This approach allows Chronos to be pre-trained on vast and diverse collections of time series data, learning general patterns and temporal dynamics, which can then be applied to specific downstream forecasting tasks, often in a zero-shot or few-shot manner.

Chronos-Bolt: An Efficient Architecture Variant

The Chronos-Bolt model represents a significant advancement over the original Chronos framework, introducing key innovations to enhance performance and efficiency while retaining the T5 encoder-decoder architecture as its base [Amazon Science n.d.(a)].

Architectural Enhancements Chronos-Bolt incorporates several architectural modifications [Amazon Science n.d.(b)]:

- *Patch-Level Embedding:* Instead of processing individual time steps, Chronos-Bolt divides historical time series data into non-overlapping patches. These patches are then embedded into fixed-dimensional representations before being fed into the encoder. This patch-based approach reduces the effective sequence length the model processes, leading to considerable improvements in computational speed and memory usage.

- *Direct Multi-Step Quantile Forecasting:* A major departure from the original Chronos's autoregressive, one-step-ahead token prediction is Chronos-Bolt's capability for direct multi-step forecasting. It simultaneously predicts multiple future time steps using quantile regression. This method not only accelerates inference but also directly provides probabilistic forecasts by outputting various quantiles, offering richer insights into prediction uncertainty.

Performance Improvements These architectural changes translate into substantial performance gains:

- *Speed and Efficiency:* Chronos-Bolt models are reported to be up to 250 times faster and 20 times more memory-efficient than original Chronos models of comparable size. Notably, the Chronos-Bolt Base model is stated to surpass the original Chronos Large model in forecasting accuracy while being over 600 times faster [Amazon Science n.d.(b)].
- *Accuracy:* Despite being evaluated in a zero-shot setting, Chronos-Bolt models have demonstrated superior performance compared to commonly used statistical and deep learning models on benchmarks measuring WQL and MASE across numerous datasets [Amazon Science n.d.(b); Ansari et al. 2024].

Model Availability Chronos-Bolt is available in several sizes, catering to different computational budgets and performance needs:

- Tiny: 9 million parameters
- Mini: 21 million parameters
- Small: 48 million parameters
- Base: 205 million parameters

These models are designed for use on CPUs and have been integrated into platforms such as AutoGluon and Amazon SageMaker JumpStart, facilitating easier deployment and fine-tuning for practical applications [Amazon Science n.d.(a)].

Exogenous Variable Integration in Chronos

The Chronos family of models including both the original framework and Chronos-Bolt, as presented by Ansari et al. [2024] and subsequent developments, primarily focuses on the univariate forecasting paradigm through its tokenization approach. Direct, built-in mechanisms for deeply integrating exogenous variables within the core token-based architecture are not explicitly detailed. However, given the importance of such external information, strategies such as external or residual modeling are commonly employed to incorporate exogenous variables when using Chronos models.

External/Residual Modeling for Exogenous Variable Integration A common strategy to incorporate exogenous variables with inherently univariate or tokenization-focused models like Chronos is through an external or residual modeling approach, as utilized by frameworks like AutoGluon-TimeSeries [Amazon Science n.d.(b)]. This approach is applicable to both the original Chronos and Chronos-Bolt variants. The core idea is to use a separate regression model to capture the effect of exogenous variables. The process typically involves:

- *Exogenous Variable Regressor Training:* A separate tabular regression model (e.g., LightGBM, CatBoost, or a simple linear model) is trained to predict the target time series y_t using available known exogenous variables x_t. Let these predictions be $f_R(x_t)$.

- *Residual Calculation:* The predictions from this regressor are subtracted from the actual target values to compute residuals $r_t = y_t - f_R(x_t)$.
- *Univariate Forecasting on Residuals:* The univariate FM (Chronos or Chronos-Bolt) is then trained to forecast these residuals r_t.
- *Final Forecast Combination:* The final forecast $\hat{y}_{t+h}$ is obtained by adding the residual forecast $\hat{r}_{t+h}$ to the regressor's prediction $f_R(x_{t+h})$ using future known exogenous variable values: $\hat{y}_{t+h} = \hat{r}_{t+h} + f_R(x_{t+h})$.

This modular approach allows the Chronos models to focus on learning temporal patterns from tokenized or patched sequences, while the effects of exogenous variables are handled by a dedicated regressor.

Capabilities and Limitations of Chronos

Capabilities:

- **Probabilistic Forecasting:** The Chronos framework is designed for probabilistic forecasting. The original version predicts full probability distributions over tokens, while Chronos-Bolt directly outputs quantiles for multi-step probabilistic forecasts.
- **Adaptation of LLM Principles:** It successfully adapts powerful concepts from LLMs (tokenization, Transformer architectures like T5) to the time series domain.
- **Simplicity of Core Idea:** The tokenization approach with scaling and quantization offers a conceptually simple way to handle diverse time series.
- **Zero-Shot Performance:** Chronos-Bolt, in particular, has demonstrated strong zero-shot forecasting capabilities on various benchmarks [Aksu et al. 2024; Li et al. 2024].
- **Efficiency and Speed:** The Chronos-Bolt variant introduces significant improvements in computational speed and memory efficiency through patch-level embedding and direct multi-step forecasting.
- **Model Availability and Accessibility:** Chronos-Bolt offers a range of model sizes and is integrated into accessible platforms like AutoGluon and Amazon SageMaker JumpStart, facilitating broader adoption.

Limitations:

- **Computational Cost (Original Chronos):** The original Chronos models, especially larger versions, can be computationally intensive and memory-demanding due to their autoregressive, token-by-token processing of long sequences.
- **Indirect Exogenous Variable Integration:** The primary mechanism for incorporating exogenous variables relies on external/residual modeling rather

than a deeply integrated approach within the core architecture. This modularity might not capture complex, non-additive interactions between exogenous variables and the target series' own dynamics as effectively as fully integrated systems.

- **Fixed Vocabulary and Quantization Effects:** The tokenization process relies on a fixed vocabulary size (N_B) determined by quantization bins. This might lead to loss of information for series with very wide dynamic ranges or where extremely fine-grained value distinctions are crucial. The effectiveness of percentile-based binning can also depend on the diversity and representativeness of the pre-training data.

- **Scaling Method Sensitivity:** While mean scaling is simple, its universal effectiveness might vary for time series with strong trends, non-stationarities, or vastly different scales not fully normalized by the mean of absolute values.

- **Patching in Chronos-Bolt:** While patching improves efficiency, the optimal patch size and its interaction with different time series characteristics might require careful consideration or tuning.

Conclusion

Chronos presents an innovative approach to TSF by leveraging the power of pre-trained language models through a unique tokenization strategy. Its core architecture transforms time series into a "language" that can be learned by Transformers, enabling probabilistic predictions. The introduction of the Chronos-Bolt variant further enhances this framework by incorporating architectural improvements like patch-level embedding and direct multi-step quantile forecasting, leading to significant gains in efficiency and speed while maintaining strong predictive accuracy. While the Chronos family of models is primarily univariate in its core design, its utility in scenarios with exogenous variables can be effectively achieved through external residual modeling techniques, as exemplified in frameworks like AutoGluon-TimeSeries. These methods allow Chronos and its variants to benefit from rich contextual information, making them versatile tools for complex real-world forecasting challenges.

3.3.3. TimeGPT

TimeGPT has emerged as a significant development in TSF, positioning itself as one of the first FMs specifically architected for this domain [Garza et al. 2023]. It aims to provide accurate predictions across a diverse range of datasets, even those not encountered during its training phase, by leveraging a Transformer-based structure. It is important to note that, unlike several other FMs in the TSF domain, TimeGPT is provided as a proprietary cloud-based service accessible via an Application Programming Interface (API) managed by Nixtla, generally involving service subscriptions, rather than being an open-source model.

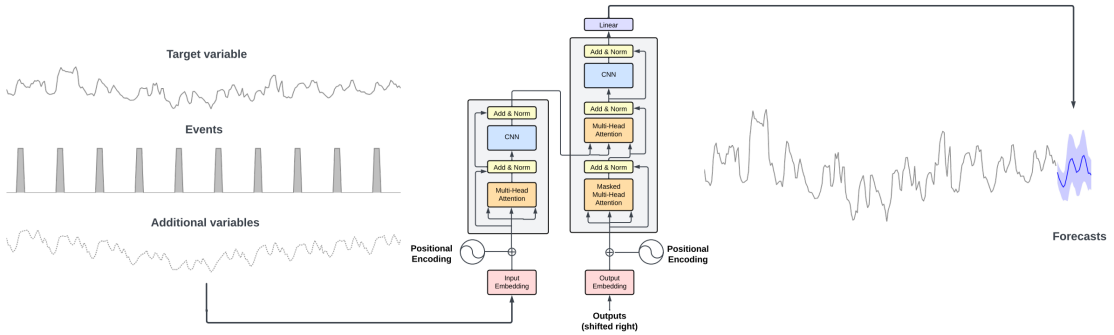


Figure 3.3.: Conceptual architectural overview of the TimeGPT FM , illustrating key components and data flow [Garza et al. 2023].

Core Architectural Framework

TimeGPT is fundamentally built upon a Transformer-based architecture, renowned for its efficacy in sequence modeling tasks primarily due to its self-attention mechanisms. Unlike some contemporary time series models that adapt existing LLMs, TimeGPT boasts a specialized architecture, meticulously designed and trained from the ground up to optimize forecasting accuracy [Garza et al. 2023].

The model adheres to an encoder-decoder structure. This architectural paradigm involves:

- An **encoder component** that processes the input time series, including historical values of the target variable and any associated exogenous variables. This part of the network is responsible for creating a rich, contextual representation of the input sequence.
- A **decoder component** that takes the encoder's representation and generates the forecast for the desired future horizon.

Internally, both the encoder and decoder are composed of multiple layers. Each of these layers typically incorporates standard Transformer building blocks:

- **Multi-Head Self-Attention:** This allows the model to weigh the importance of different parts of the input sequence when making predictions for a particular time step.
- **Feed-Forward Networks:** Position-wise feed-forward networks are applied independently to each position.
- **Residual Connections:** These are employed around each of the two sub-layers (attention and feed-forward network) to aid in training deeper models by mitigating the vanishing gradient problem.
- **Layer Normalization:** Applied after the residual connections, layer normalization helps stabilize the learning process and reduces training time.

A final linear layer is appended to the output of the decoder. The role of this layer is to map the high-dimensional representation learned by the decoder to the specific dimensionality of the forecasting window, thereby producing the actual predicted

values for future time steps. A key characteristic of TimeGPT is its design to handle time series data exhibiting varied frequencies and characteristics. It is engineered to be flexible concerning input sizes and the length of the forecast horizon, a crucial feature for a general-purpose FM.

Exogenous Variable Integration in TimeGPT

TimeGPT is explicitly designed to incorporate external information in the form of exogenous variables to enhance its predictive power. The model accepts historical values of the target variables alongside these additional exogenous variables as inputs to generate forecasts.

The model's documentation and conceptual diagrams indicate support for:

- **General Exogenous Variables:** These can be any external time series that are believed to have an impact on the target variables.
- **Events:** The model can process event-based exogenous variables, which are often dynamic and categorical in nature, such as public holidays, promotions, or other significant occurrences. Indeed, conceptual diagrams of TimeGPT, such as the one presented in Figure 3.3, often explicitly depict "Additional variables" and "Events" as distinct inputs to the model.

Given its Transformer-based architecture, one common method for integrating exogenous variables is to process them as additional input sequences or features. In such an approach, these sequences would typically be concatenated or otherwise combined with the target variable's historical data before being fed into the encoder. The self-attention mechanisms inherent in Transformers are designed to learn complex dependencies and interactions, which would include those between the target series and any incorporated exogenous variables.

However, a limitation exists in the publicly available information from Nixtla regarding the specific architectural mechanisms for exogenous variable integration. Details pertaining to:

- *Dedicated Embedding Layers for Exogenous Variables:* Whether exogenous variables (categorical or numerical) undergo a separate embedding process before being fused with the target series embeddings.
- *Specialized Attention Mechanisms:* If there are tailored attention mechanisms or fusion strategies (e.g., cross-attention between target and exogenous variable representations at specific layers) designed explicitly for integrating exogenous variable information within the encoder-decoder structure.
- *Distinguished Processing for Exogenous Variable Types:* The precise methods by which different types of exogenous variables (e.g., static vs. dynamic, past-known vs. future-known) are differentiated and processed within the model's pipeline.

While TimeGPT demonstrably accepts and utilizes exogenous variables, the granular details of their internal embedding, processing, and the specific ways in which the attention mechanism learns their influence on the target series are not thoroughly elucidated in currently available public descriptions from Nixtla. This lack

of explicit detail makes it challenging to fully ascertain the nuances of how different exogenous variable characteristics are leveraged or how their potential interactions are modeled at a deep architectural level.

Capabilities and Limitations of TimeGPT

Capabilities:

- **Transformer-Based Architecture:** Leverages a Transformer encoder-decoder structure, which is well-suited for capturing temporal dependencies in sequence data.
- **Specialized for TSF :** Designed and pre-trained specifically for TSF tasks, rather than being a direct adaptation of an LLM pre-trained on text.
- **Versatility:** Aims to handle a diverse range of time series datasets, varying frequencies, input sizes, and forecast horizons.
- **Exogenous Variable Support:** Explicitly designed to incorporate general exogenous variables and specific event-based data to potentially improve forecast accuracy.
- **Pioneering FM for TSF:** Positioned as one of the early FMs dedicated to TSF .
- **Managed API Service:** Offered as a cloud-based API, which can simplify deployment and reduce the need for users to manage infrastructure, potentially speeding up time-to-forecast for some applications.

Limitations:

- **Proprietary Nature:** Being a commercial, closed-source model accessible only via a paid API (Nixtla) limits its accessibility for academic research, independent scrutiny, replication, and customization. This also introduces potential vendor lock-in.
- **Lack of Architectural Transparency:** Specific details regarding the internal mechanisms for exogenous variable embedding, fusion techniques, and specialized attention (if any) are not publicly disclosed by Nixtla, making a full understanding and comparative analysis of its integration methods challenging.
- **Performance Verification:** While positioned as a high-performance model, comprehensive, independent, third-party benchmarking against a wide array of open models on diverse public datasets is less prevalent than for open-source alternatives. Evaluation often relies on vendor-provided information or studies conducted via the API.
- **Architectural Currency:** As one of the earlier FMs for TSF, its specific architectural components, while Transformer-based, might not incorporate the very latest innovations or specialized adaptations for time series that have emerged in more recent research within the rapidly evolving FM landscape.

Conclusion

In summary, TimeGPT presents a robust Transformer-based encoder-decoder architecture tailored for TSF . It explicitly supports the integration of exogenous variables, which are understood to be processed as additional input features, allowing the self-attention mechanism to capture their influence. Nevertheless, a deeper understanding of the specific internal mechanisms for embedding and fusing these exogenous variables awaits more detailed architectural disclosures from its developers, Nixtla. Its nature as a proprietary API-based service also distinguishes its accessibility and transparency within the landscape of FMs for TSF, presenting both advantages for ease of use in some contexts and limitations for open research and full methodological comparison.

3.3.4. TimesFM

TimesFM, a FM for TSF developed by Google Research [Das, Kong, Sen, et al. 2023], represents a significant stride in leveraging large-scale pre-training for enhanced predictive performance. It is an open-source model available for both Just After eXecution (JAX) and PyTorch frameworks. TimesFM has been released in different versions, notably TimesFM-1.0-200M (with context lengths up to 512 time points) and the more recent TimesFM-2.0-500M (with context lengths up to 2048 time points), the latter reportedly offering up to 25% improvement in accuracy. It was pre-trained on an extensive corpus, reportedly comprising 100 billion real-world time series data points for the initial version, enabling it to learn a wide array of temporal patterns and dependencies. This analysis delves into the core architectural framework of TimesFM and critically examines its methodology for integrating exogenous variables, a crucial aspect for improving forecast accuracy and contextual understanding in diverse real-world scenarios.

Core Architectural Framework

TimesFM is architecturally designed as a decoder-only Transformer model. This choice aligns with the successes observed in NLP where decoder-only architectures, such as GPT models, have demonstrated remarkable capabilities in sequence generation tasks. Different versions of TimesFM support varying maximum context lengths, with TimesFM-1.0 handling up to 512 time points and TimesFM-2.0 extending this to 2048 time points. Figure 3.4 visually illustrates the architecture of TimesFM.

A key characteristic of TimesFM's architecture is its adoption of **patching**. Instead of processing individual time points, the input time series is segmented into patches, which are sequences of contiguous time points. This patching mechanism allows the model to:

- Capture local semantic information from groups of data points more effectively.
- Process longer time series contexts more efficiently than point-wise processing, by reducing the sequence length fed into the Transformer.

Let an input time series be denoted by $Y = \{y_1, y_2, \ldots, y_{L_{series}}\}$, where L_{series} is the length of the series. With a patch length P_{len} and stride S_{stride}, the series is

transformed into a sequence of patches $X = \{x_1, x_2, \ldots, x_N\}$, where each patch x_i (in this context, x_i represents a patch of the target series Y) represents a segment of Y. These patches are then typically embedded into a higher-dimensional space.

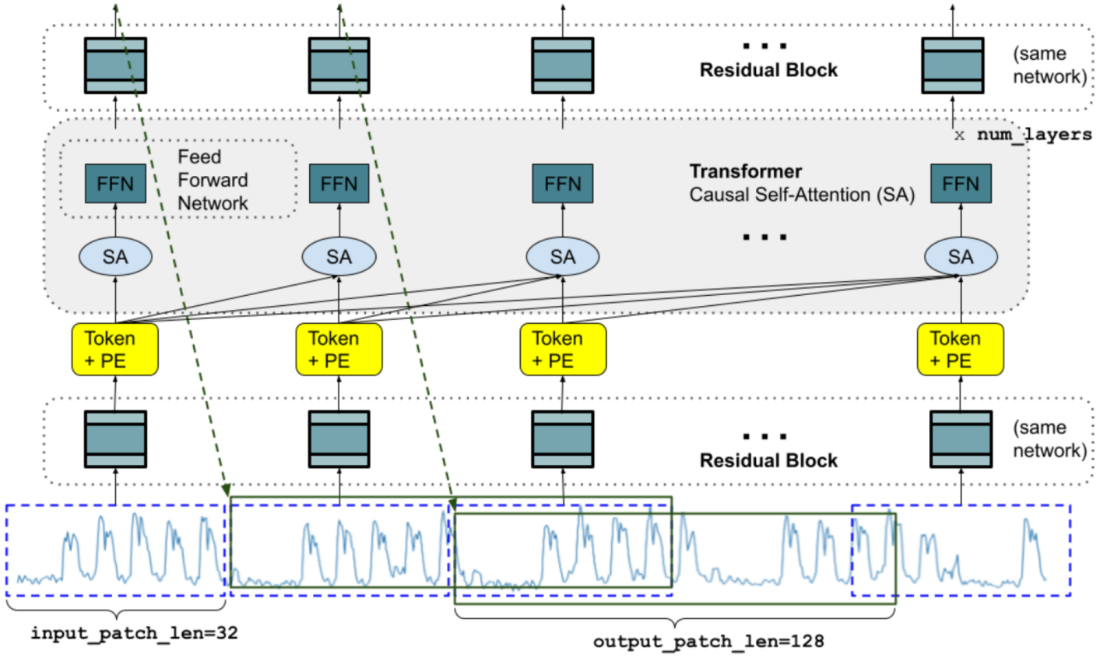


Figure 3.4.: Conceptual architectural overview of the TimesFM model [Das, Kong, Sen, et al. 2023].

The architecture incorporates **residual block structures**. These are fundamental to deep Transformer models, facilitating the training of deeper networks by allowing gradients to propagate more easily through the addition of the input of a block to its output. This can be represented as:

$$\text{Output} = \text{Layer}(\text{Input}) + \text{Input} \tag{3.3}$$

This structure is utilized for embedding the input series and for mapping the output of the decoder (which consists of multiple Transformer layers with self-attention mechanisms) to the final predictions. The self-attention mechanism within the decoder allows the model to weigh the importance of different patches in the historical context when predicting future patches. Both released versions of TimesFM are primarily trained for point forecasts, although TimesFM-2.0 experimentally offers 10 quantile heads for probabilistic forecasting. However, these quantile heads have not been calibrated after pre-training, making their probabilistic output currently experimental [Google Research n.d.].

Mechanisms for Embedding and Processing Exogenous Variables

TimesFM employs a distinct and somewhat externalized strategy for incorporating exogenous variables, rather than deeply integrating them into the core neural architecture during the primary forecasting step. Exogenous variables are treated as batched in-context exogenous regressors (XReg). The influence of these exogenous variables is modeled primarily through linear models that operate outside the main TimesFM neural network.

The final forecast generated by the TimesFM system is typically a combination, often a sum, of the forecast produced by the core TimesFM model (based on the historical target series) and the forecast or adjustment derived from the linear model of exogenous variables. Two primary options for this integration are outlined:

- **"timesfm + xreg":** First, a forecast, denoted $\hat{Y}_{\text{TimesFM}}$, is obtained from the core TimesFM model using only the historical target series. The residuals are then calculated: $R = Y_{\text{actual}} - \hat{Y}_{\text{TimesFM}}$. A linear model is subsequently fitted to regress these residuals on the available exogenous variables x: $R \sim \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \epsilon$. The prediction from this linear model, $\hat{R}_{\text{xreg}}$, which represents the exogenous variables' effect on the residuals, is added to TimesFM's initial forecast to produce the final forecast: $\hat{Y}_{\text{final}} = \hat{Y}_{\text{TimesFM}} + \hat{R}_{\text{xreg}}$.

- **"xreg + timesfm":** First, a linear model is fitted to regress the target time series Y_{actual} directly on the exogenous variables x: $Y_{\text{actual}} \sim \gamma_0 + \sum_{j=1}^{k} \gamma_j x_j + \epsilon'$. Let the forecast from this linear model be $\hat{Y}_{\text{xreg}}$. The residuals from this initial linear model are calculated: $R' = Y_{\text{actual}} - \hat{Y}_{\text{xreg}}$. TimesFM is then used to forecast these residuals: $\hat{R}'_{\text{TimesFM}}$. The TimesFM forecast of residuals is added to the linear model's forecast of the target to yield the final prediction: $\hat{Y}_{\text{final}} = \hat{Y}_{\text{xreg}} + \hat{R}'_{\text{TimesFM}}$.

This approach relies on an external 'xreg lib' module, which necessitates the JAX library for its operations, though the model itself is also available in PyTorch. This modular design separates the learning of complex temporal dynamics by the FM from the modeling of exogenous variable effects, which are assumed to be largely linear.

Handling Diverse Exogenous Variable Types

The TimesFM framework, through its external regressor library, is designed to support a variety of exogenous variable types:

- **Static Exogenous Variables:** These are time-independent attributes. Examples include product category (e.g., "food," "skin product") or store ID. These are typically handled via one-hot encoding or embedding layers within the linear model context if the library supports it, or by creating separate models per category.

- **Dynamic Exogenous Variables:** These variables change over time. Examples include day of the week or a "Has_promotion" flag (Yes/No), or numerical values like daily temperature or an estimated electricity generation forecast ('gen forecast').

A mandatory requirement for dynamic exogenous variables within the TimesFM framework is that their values must be available for both the historical context period used for forecasting and the entire future forecast horizon H.

For dynamic exogenous variables whose future values are unknown at prediction time, the TimesFM documentation suggests two "hacky options" [Google Research n.d.], underscoring the challenges this scenario presents to its primary exogenous variable integration strategy:

- **Shift and Repeat (Lagging):** Past dynamic exogenous variables can be shifted (lagged) and repeated to serve as proxies for their future values. For example, the value of an exogenous variable from $t - H$ might be used as an input for predicting at time t, where H is the forecast horizon. This assumes some level of persistence or cyclicality in the exogenous variable.
- **Bootstrapping:** This involves a multi-step forecasting approach: First, use TimesFM (or another forecasting model) to forecast the future values of these "future-unknown" dynamic exogenous variables. Then, use these generated forecasts of the exogenous variables as inputs when making the final forecast for the primary target variable with the TimesFM + XReg framework. The accuracy of this approach is contingent on the accuracy of the exogenous variable forecasts themselves.

The reliance on external linear modeling and the described nature of handling future-unknown dynamic exogenous variables suggest potential limitations in capturing complex, non-linear interactions between the target series and exogenous variables, or when rich future exogenous variable information is unavailable or uncertain.

Capabilities and Limitations of TimesFM

Capabilities:

- **Open-Source and Framework Availability:** TimesFM is an open-source model available in both JAX and PyTorch, facilitating broader research and adoption.
- **Large-Scale Pre-training:** Pre-trained on a very large corpus of time series data (e.g., 100 billion points for v1.0), enabling it to learn diverse temporal patterns.
- **Multiple Versions:** Offers different versions (e.g., TimesFM-1.0-200M with up to 512 context, TimesFM-2.0-500M with up to 2048 context and improved accuracy), allowing users to choose based on their needs and data characteristics.
- **Decoder-Only Transformer Architecture:** Utilizes an efficient and effective Transformer variant with patching for processing long sequences.
- **Modular Exogenous Variable Integration:** Provides explicit support for static and dynamic (future-known) exogenous variables through an external linear regressor module (XReg lib), with flexible "timesfm + xreg" and "xreg + timesfm" strategies.
- **Demonstrated Performance Gains:** Shows improved forecast accuracy when exogenous variables are incorporated using its methodology.

Limitations:

- **Primarily Point Forecasts:** Both main versions are trained for point forecasts. While TimesFM-2.0 offers experimental, uncalibrated quantile heads, reliable probabilistic forecasting is not its primary design output.

- **Externalized and Linear Exogenous Variable Integration:** The reliance on external linear models for exogenous variables may not capture complex, non-linear interactions between the target series and these variables as effectively as deeply integrated neural approaches might.

- **Requires Future Values for Dynamic Exogenous Variables:** The primary integration strategies necessitate that future values of dynamic exogenous variables are known for the entire forecast horizon.

- **Ad-hoc Handling of Future-Unknown Exogenous Variables:** Suggested methods for future-unknown dynamic exogenous variables ("hacky options" like lagging or bootstrapping via separate forecasts) can introduce additional complexity, assumptions, and potential error propagation.

- **Dependency on External Library:** The ‘xreg lib‘ for exogenous variable processing introduces an additional dependency (e.g., JAX, although the core model is also in PyTorch).

Conclusion

TimesFM presents a powerful, open-source decoder-only Transformer architecture for TSF , benefiting from large-scale pre-training and efficient patch-based input processing. Its strategy for exogenous variable integration is characterized by modularity, relying on external linear models to incorporate their effects, which has demonstrated tangible accuracy improvements. This approach simplifies the core FM's task and offers flexibility. However, it primarily assumes linear relationships for exogenous variable effects and mandates the availability of future values for dynamic exogenous variables for its most direct integration method, with more ad-hoc solutions for scenarios where future exogenous variables are unknown. While effective in many contexts, this externalized, linear-focused integration may not fully capture deeply intertwined, non-linear dynamics between exogenous variables and the target series that more integrated architectural approaches might address. Its availability as an open-source model in multiple frameworks is a significant advantage for the research community.

3.3.5. MOIRAI

The **MOIRAI** model, an acronym for Masked Encoder-based Universal Time Series Forecasting Transformer, represents a significant stride in the domain of deep learning for TSF . Developed by Salesforce AI Research, **MOIRAI** is engineered as a FM aiming to provide universal forecasting capabilities. It is available in several sizes with up to 311M parameters. Pre-trained on a vast and diverse corpus of time series data, the Large-scale Open Time Series Archive (LOTSA), it endeavors to deliver robust zero-shot forecasting performance across a multitude of downstream tasks [Woo et al. 2024]. This analysis will elucidate **MOIRAI**'s core architectural framework, including its innovative "Any-variate Attention" mechanism, and its methodology for integrating exogenous variables, thereby enabling it to handle complex, multivariate forecasting scenarios. It also discusses the MOIRAI-MoE variant.

Introduction (Original MOIRAI Concepts)

Traditional deep learning approaches to TSF have predominantly followed a "one-model-per-dataset" paradigm. This limits the potential for leveraging knowledge transfer from large, diverse datasets. **MOIRAI** seeks to overcome these limitations by establishing a single, pre-trained model capable of addressing heterogeneous forecasting tasks without task-specific retraining.

The core challenges in constructing such a universal time series model adresed by MOIRAI include:

- Cross-granularity learning: Effectively handling time series data sampled at different granularities.
- Arbitrary variate accommodation: Seamlessly processing multivariate time series with a varying number of target variables and exogenous variables.
- Distributional shifts: Addressing the diverse statistical properties inherent in large-scale, multi-domain time series data.

MOIRAI confronts these challenges through novel enhancements to the standard Transformer architecture, particularly its encoder-only design, dynamic patch sizing, and the "Any-variate Attention" mechanism.

Core Architectural Framework (Original MOIRAI)

MOIRAI is fundamentally a masked encoder-only Transformer model. This architectural choice emphasizes learning rich representations from the input time series data. The model is available in Small (14M parameters), Base (91M parameters), and Large (311M parameters) versions.

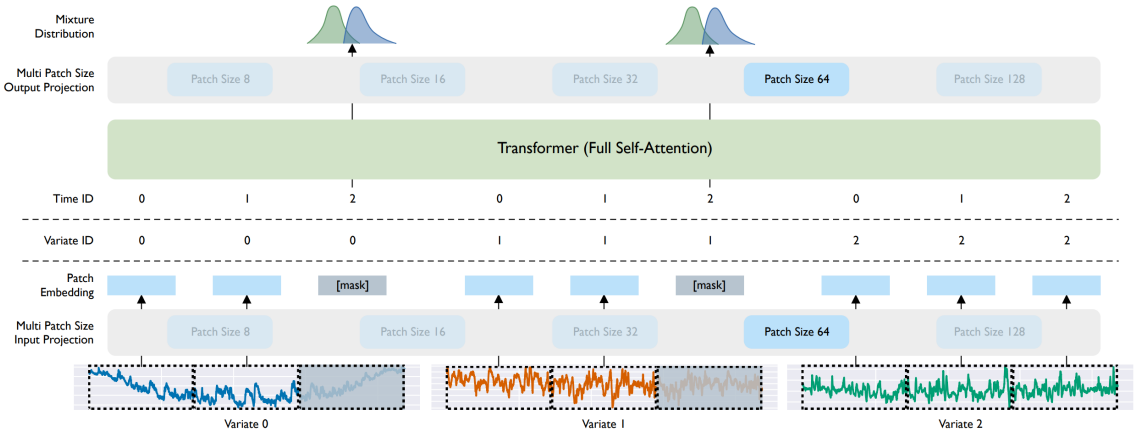


Figure 3.5.: Conceptual architectural overview of MOIRAI data processing.

Input Processing and Patching The model ingests multivariate time series. A conceptual overview of how the MOIRAI architecture processes multivariate time series is illustrated in Figure 3.5. Let an input multivariate time series be denoted as $X \in \mathbb{R}^{N \times L_{in}}$, where N is the total number of variates (including both target series and any accompanying exogenous variables) and L_{in} is the length of the input time series context.

- **Patchification:** Each of the N variates is independently segmented into patches. If a variate $x_i \in \mathbb{R}^{L_{in}}$ (where $i = 1, \ldots, N$) is divided into patches of length P_{len} with a stride S_{stride}, it results in a sequence of $M_p = \lfloor (L_{in} - P_{len})/S_{stride} + 1 \rfloor$ patches for that variate. Each patch is then typically flattened and projected into an embedding space of dimension D_{model}. **MOIRAI** incorporates dynamic patch sizing, allowing P_{len} to vary, which aids in handling time series of different granularities.
- **Embeddings:**
  - **Patch Embeddings:** The projected patches form the primary token representations.
  - **Positional Embeddings:** Standard positional encodings are added to these patch embeddings to provide information about their sequence order.
  - **Variate ID Embeddings:** Crucially, each patch is also associated with a learnable embedding that identifies its source variate (e.g., "variate 1," "variate 2," ..., "variate N"). This allows the model to distinguish between different time series, including targets and exogenous variables, even when they are processed in a flattened sequence.

Masked Encoder The core of **MOIRAI** consists of a stack of Transformer encoder layers. During pre-training, a masking strategy akin to that used in BERT is employed [Devlin et al. 2019]. A certain percentage of input patches are masked (e.g., replaced with a special `[MASK]` token or corrupted), and the model is trained to reconstruct these original masked patches. This self-supervised task enables the model to learn meaningful temporal patterns and inter-variate dependencies from the vast LOTSA dataset [Woo et al. 2024]. The loss function for pre-training is typically a MSE between the predicted patches and the original masked patches. The output of the encoder for a given patch incorporates contextual information from other unmasked patches across time and variates, mediated by the attention mechanism.

Exogenous Variable Integration in MOIRAI (Original Framework)

MOIRAI adopts a "homogenized input channels/variates" strategy for exogenous variables. This means exogenous variables are not treated through separate dedicated modules but are integrated seamlessly as additional variates within the input tensor X.

Unified Input Representation If a forecasting task involves N_{target} target series and N_{exo} exogenous variable series, the total number of input variates to **MOIRAI** becomes $N = N_{target} + N_{exo}$. Each exogenous variable series, whether static or dynamic, is treated as another channel in the input.

- **Dynamic Exogenous Variables:** These are time-varying series (e.g., temperature, promotional flags) and are naturally represented as individual variates. Their values, known for the historical context and potentially for the future forecast horizon, are patchified just like the target series [Woo et al. 2024].

- **Static Exogenous Variables:** These are time-independent attributes (e.g., store ID, product category). They can be incorporated by representing them as dynamic series with constant values repeated across the entire time dimension (both context and horizon). Categorical static exogenous variables would first be embedded into a numerical representation before being treated as a constant channel.

The variate ID embeddings become essential here, allowing the model to learn the specific role and influence of each variate, distinguishing, for example, a temperature-related exogenous variable from a sales target series.

Patchification and Flattening All N variates (targets and exogenous variables) undergo the same patchification process. The collection of all patches from all variates is then effectively flattened into a single sequence of $N \times M_p$ patch tokens. For example, if there are 2 target series and 3 exogenous variable series, each yielding M_p patches, the input to the encoder would be a sequence of $5 \times M_p$ tokens.

Any-variate Attention (AVA) in Original MOIRAI

The **Any-variate Attention (AVA)** mechanism is a key innovation in **MOIRAI**, enabling the effective processing of the homogenized input that includes an arbitrary number of target and exogenous variable series [Woo et al. 2024]. AVA allows **MOIRAI** to:

- **Handle Arbitrary Dimensions:** Process time series with any number of variates in a principled manner.
- **Learn Cross-Variate Dependencies:** The attention mechanism can learn interactions between patches from different variates (e.g., how a patch from a holiday indicator exogenous variable influences a patch from a demand target series) and temporal dependencies within the same variate.
- **Support Flexible Modeling Modes:**
  - **Channel-Independent (CI) Pre-training:** During pre-training on diverse datasets (like LOTSA, which contains many univariate series), AVA can be configured to operate in a channel-independent mode. In this mode, attention is primarily focused within each variate, allowing the model to learn general temporal patterns without being confounded by inconsistent multivariate structures across different datasets in the archive.
  - **Channel-Mixed (CM) Fine-tuning/Inference:** When fine-tuning on a specific multivariate dataset or during inference, AVA operates in a channel-mixed mode. Here, attention is computed across all patches from all variates, enabling the model to capture intricate cross-variate correlations, including the influence of exogenous variables on target variables.

While the precise mathematical formulation of AVA, including how variate ID embeddings explicitly bias or gate the attention scores, the conceptual outcome is an attention mechanism that is aware of which variate each patch belongs to and can modulate its behavior accordingly. The self-attention calculation for a query Q_i

from patch i, and keys K_j and values V_j from patches j in the flattened sequence can be generally written as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B\right)V \tag{3.4}$$

where B could be an attention bias matrix potentially incorporating variate ID information to facilitate CI or CM behavior.

Handling Diverse Exogenous Variable Types (Original MOIRAI)

MOIRAI's architecture, through its homogenized input and AVA, allows it to ingest various types of exogenous variables (as defined in Section 2.5.2).

- **Dynamic Past-Observed Exogenous Variables:** These are included as variates, with their historical values patchified. The model learns their influence on the target through channel-mixed attention.
- **Dynamic Future-Known Exogenous Variables:** If future values (e.g., scheduled promotions, weather forecasts for the prediction horizon) are available, they are included as part of their respective variate series spanning both the context and prediction lengths. The masked encoding pre-training helps the model learn to use this future contextual information.
- **Static Exogenous Variables (Categorical and Numerical):** These are typically transformed into constant-valued time series channels. Categorical static exogenous variables would first be embedded into a numerical representation. The model learns their persistent influence via the attention mechanism, distinguished by their variate IDs.

MOIRAI-MoE: Mixture-of-Experts Variant

To address limitations of frequency-level specialization and enhance adaptability to diverse temporal patterns, a MoE variant, MOIRAI-MoE, was developed [X. Liu et al. 2024].

Motivation and Design Philosophy Traditional universal time series FMs like the original MOIRAI often use frequency-level specialization (e.g., separate input/output projections per sampling granularity) to handle data heterogeneity. However, this approach has limitations: time series from different granularities can share patterns, while series of the same granularity can differ significantly. Moreover, non-stationarity within a single series makes coarse granularity buckets suboptimal. MOIRAI-MoE replaces manual granularity-based projections with a single shared input projection, delegating pattern specialization to a sparse MoE Transformer. Each patch token is routed to "expert" sub-networks best suited for its local dynamics, enabling data-driven specialization.

Architectural Changes and Token Construction MOIRAI-MoE employs a decoder-only Transformer architecture.

- **Patching and Normalization:** Input series are divided into non-overlapping patches of size P. Each patch is normalized, with a portion of the window (e.g., 30%) used for robust normalizer estimation without contributing to prediction loss, enhancing parallelism.
- **Shared Input Projection:** After normalization, each patch passes through a single shared residual MLP to produce a D_{model}-dimensional token: $\mathbf{x} \in \mathbb{R}^{N\times P} \longrightarrow \text{ResMLP}(\text{normalize}(\mathbf{x})) \in \mathbb{R}^{N\times D_{model}}$. This contrasts with original MOIRAI's separate projections per granularity.

Mixture-of-Experts Layers Each Transformer block's feed-forward sublayer is replaced by a sparse MoE layer.

- **Composition:** Each MoE layer has M expert networks (typically $M = 32$), each an Feed-Forward Network (FFN), and a gating function $G(\cdot)$ that activates K experts per token (typically $K = 2$). The MoE output for a token $\mathbf{z}$ is $\text{MoE}(\mathbf{z}) = \sum_{i=1}^{M} G(\mathbf{z})_i \cdot E_i(\mathbf{z})$.
- **Model Sizes:** Two sizes, MOIRAI-MoE$_{\text{SM}}$ (Small, 11M activated params) and MOIRAI-MoE$_{\text{B}}$ (Base, 86M activated params), are designed to match the activated parameter counts of MOIRAI's Small and Base dense versions, respectively, despite having larger total parameter counts (e.g., MOIRAI-MoE$_{\text{SM}}$ has 117M total params).
- **Gating Mechanisms:**
  - *Linear Projection Gating:* $G_{\text{linear}}(\mathbf{z}) = \text{Softmax}\Big(\text{TopK}\big(\mathbf{z}W_g\big)\Big)$, with $W_g \in \mathbb{R}^{D_{model}\times M}$. An auxiliary load-balancing loss $\mathcal{L}_{\text{load}} = M\sum_{i=1}^{M} D_i P_i$ is often used.
  - *Token Clustering (Centroid-Guided) Gating:* Uses M centroids C_j precomputed from a dense Moirai model's embeddings. Gating weights are based on negative Euclidean distances: $G_{\text{cluster}}(\mathbf{z}) = \text{Softmax}\big(\text{TopK}(-\|\mathbf{z}-C_j\|_2)\big)$. This method empirically performs best.

Training Objective and Efficiency MOIRAI-MoE is trained with a decoder-only autoregressive loss, predicting the next patch token x_{t+1} given the context $x_{t-\ell+1:t}$. The negative log-likelihood $\mathcal{L}_{\text{pred}} = -\log p\big(x_{t+1} \,\big|\, \hat{\phi}\big)$ is minimized, where $\hat{\phi} = f_\theta(x_{t-\ell+1:t})$. This supports both point and probabilistic forecasts. This decoder-only approach is more computationally efficient for training compared to MOIRAI's original masked-encoder pre-training.

Exogenous Variable Handling in MOIRAI-MoE The preprint describing MOIRAI-MoE does not explicitly detail a mechanism for incorporating auxiliary exogenous variables (such as calendar features or other external time series) directly into the MoE architecture. The focus is on leveraging the MoE layers for pattern specialization based on the input time series patches themselves (which could be multivariate, including original targets and any pre-combined exogenous series as variates fed into the shared projection). Thus, if exogenous variables are to be used, they would likely need to be treated as additional input variates processed through the same shared

input projection and MoE layers, similar to the original MOIRAI's homogenized input strategy. However, dedicated mechanisms or evaluations for exogenous variable integration specific to the MoE variant are not a primary highlight of its introduction.

Empirical Performance Summary (MOIRAI-MoE) MOIRAI-MoE models have shown significant MAE improvements over their dense MOIRAI counterparts (e.g., MOIRAI-MoE$_{\text{SM}}$ by 17% over MOIRAI-S) and other foundation baselines, while activating a similar number of parameters. They also achieve strong zero-shot performance on unseen datasets. Inference latency is comparable to dense counterparts due to similar activated parameter counts, and patch-based prediction (e.g., $P = 16$) is faster than token-level autoregression.

Capabilities and Limitations (Overall MOIRAI Family)

Capabilities:

- **Universality and Zero-Shot Capability:** Designed to handle a wide array of forecasting tasks across diverse datasets without task-specific training, leveraging knowledge from the LOTSA pre-training corpus.
- **Arbitrary Variate Handling (Original MOIRAI):** The AVA mechanism in the original MOIRAI is crucial for flexibly managing any number of target and exogenous variable series.
- **Dynamic Patch Sizing (Original MOIRAI):** Adapts to different time series granularities.
- **Unified Exogenous Variable Integration (Original MOIRAI):** Exogenous variables are treated as additional variates, allowing the model to learn complex, non-linear interactions through the shared Transformer encoder and AVA.
- **Data-Driven Specialization (MOIRAI-MoE):** The MoE variant replaces manual granularity-based specialization with token-level routing to experts, adapting to diverse patterns.
- **Efficiency and Performance (MOIRAI-MoE):** The MoE variant offers improved accuracy and training efficiency over the original MOIRAI, with comparable inference latency for similar activated parameters.
- **Probabilistic Forecasting Support (MOIRAI-MoE):** The decoder-only autoregressive training objective naturally supports probabilistic outputs.

Limitations:

- **Exogenous Variable Handling in MOIRAI-MoE:** The MOIRAI-MoE variant, as described in its introductory preprint, does not feature an explicit, distinct mechanism for integrating auxiliary exogenous variables beyond treating them as input variates to the shared projection. The effectiveness of this for diverse exogenous variable types compared to more specialized integration techniques may require further investigation.

- **Potential for Noise (Original MOIRAI & MoE with many variates):** If many irrelevant or noisy exogenous variables are included as input variates, they might dilute the signal from more informative ones. Effective attention or expert routing is crucial to mitigate this.
- **Complexity of Mechanisms:** Advanced components like AVA and the MoE routing with token clustering, while powerful, add to the model's complexity and may require careful tuning or understanding for optimal application.
- **Computational Cost of Pre-training:** Pre-training large Transformer models like MOIRAI on massive datasets (LOTSA) is inherently computationally intensive, although this is a one-time cost amortized over many downstream zero-shot uses.

Conclusion

MOIRAI and its Mixture-of-Experts variant, **MOIRAI-MoE**, stand as compelling examples of FMs tailored for TSF . The original **MOIRAI**'s masked encoder-only Transformer architecture, with dynamic patch sizing and "Any-variate Attention," provides a robust framework for universal forecasting, integrating exogenous variables as homogenized input variates. **MOIRAI-MoE** further refines this by employing a decoder-only Transformer with sparse MoE layers, aiming for more data-driven pattern specialization and enhanced efficiency. While the original MOIRAI offers a clear strategy for unified exogenous variable integration through AVA, the MoE variant's documentation primarily focuses on its expert-based processing of input series patches, with less explicit emphasis on specialized mechanisms for auxiliary exogenous variables beyond their inclusion as input variates. Both approaches mark significant departures from external or purely linear exogenous variable modeling, pushing towards end-to-end learning of intricate temporal dynamics in multivariate contexts. These design principles offer a promising direction for building versatile and powerful TSF systems.

3.3.6. TimerXL

TimerXL has emerged as a significant contribution to the field of TSF , proposing a unified framework based on a generative, decoder-only Transformer architecture [Yong Liu, Qin, Huang, et al. 2024]. It aims to redefine forecasting by generalizing the "next token prediction" paradigm to a "multivariate next token prediction" task, framing diverse scenarios as long-context generation problems. While the primary focus of the cited work is this unifying framework and its architectural innovations, which are highly relevant for the development and application of advanced time series models including FMs , TimerXL is presented here for its sophisticated mechanisms for integrating exogenous variables. This analysis delves into the core architectural components of TimerXL, with a particular focus on these mechanisms [Yong Liu, Qin, Huang, et al. 2024].

Core Architectural Framework

At its heart, TimerXL redefines TSF by generalizing the standard "next token prediction" paradigm, prevalent in NLP, to a "multivariate next token prediction" task.

This conceptual shift allows TimerXL to frame diverse forecasting scenarios as a long-context generation problem. The model leverages a decoder-only Transformer architecture, which is inherently suited for capturing causal dependencies from input contexts of varying lengths. A fundamental preprocessing step in TimerXL is **patch tokenization**. Input time series, whether univariate or multivariate (including target series and any associated exogenous variables), are segmented into non-overlapping patches. These patches serve as the fundamental tokens that the Transformer operates upon. This patching mechanism allows the model to capture local temporal patterns within segments of the time series.

Mechanisms for Embedding and Processing Exogenous Variables

TimerXL's approach to exogenous variable integration is characterized by treating exogenous variables as additional time series variables. These are seamlessly incorporated into a unified, flattened input sequence alongside the primary target variable(s). The combined multidimensional series then undergoes patchification [Yong Liu, Qin, Huang, et al. 2024].

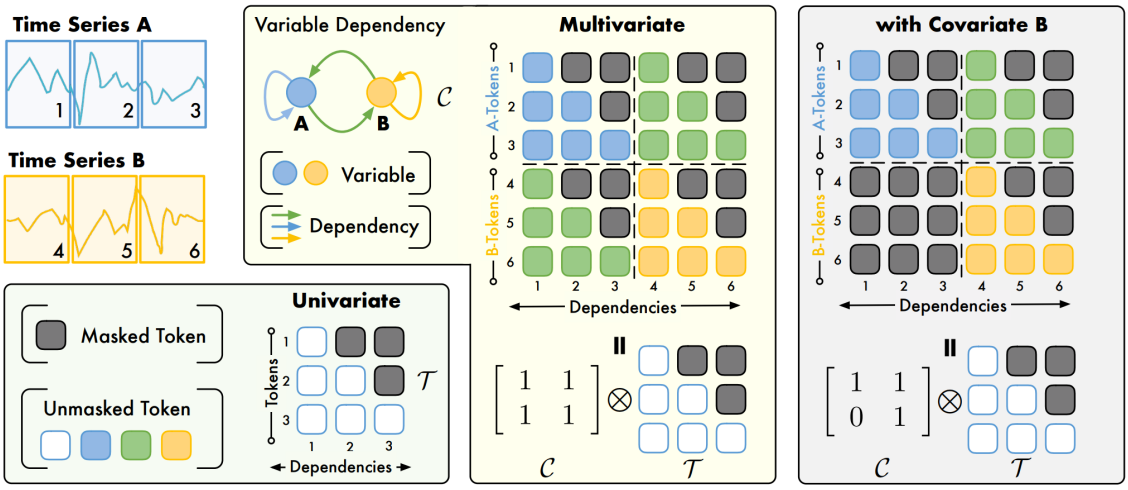


Figure 3.6.: Illustration of TimeAttention. For univariate series, temporal mask T keeps the causality. Given multivariate patch tokens sorted in a temporal-first order, we adopt the variable dependencies C, an all-one matrix, as the left-operand of Kronecker product, expanding temporal mask to a block matrix, which exactly reflects dependencies of multivariate next token prediction. The formulation is also generalizable to univariate and covariate-informed contexts with pre-defined variable dependency.

The cornerstone of TimerXL's ability to process these complex inputs, including exogenous variables, is its **Universal TimeAttention** mechanism illustrated in Figure 3.6. This specialized attention module is designed to operate effectively on the flattened, multidimensional sequence of patch tokens. Key features of **Universal TimeAttention** include:

- **Flattened Token Processing:** All patch tokens, originating from both target series and exogenous variables, are processed in a unified manner. Their original two-dimensional indices (variable index, time index) are flattened into

a one-dimensional sequence. This is typically done in a temporal-first order, meaning tokens are ordered primarily by their time step and secondarily by their variable index.

- **Position Embedding:** To preserve information about the original position and identity of each token in the flattened sequence, TimerXL employs sophisticated position embedding strategies:
  - **Temporal Dimension:** Rotary Position Embedding (RoPE) is applied to the query and key vectors in the attention mechanism. RoPE is known for its ability to encode relative positional information effectively, which is crucial for capturing temporal dependencies.
  - **Variable Dimension:** Learnable scalar embeddings are associated with each variable (channel). These scalar embeddings provide the model with information to distinguish between different time series, including differentiating endogenous (target) variables from exogenous variables.

- **Kronecker-based Masking:** This is a critical component that governs the flow of information and dependencies within the attention mechanism [Yong Liu, Qin, Huang, et al. 2024]. It combines two types of masks:
  - **Variable Dependency Graph (C):** This is an $N \times N$ adjacency matrix, where N represents the total number of variates (target series + exogenous variables). An entry $C_{m,n} = 1$ signifies that variable m is dependent on variable n, and thus, variable m is allowed to attend to variable n. Conversely, if $C_{m,n} = 0$, attention from m to n is masked. This matrix C is not learned by the model; instead, it is pre-specified based on domain knowledge or assumptions about the relationships between the variables. For instance, a target variable might depend on its own past values and the past (or even future-known) values of several exogenous variables. Exogenous variables might depend on their own past or other exogenous variables but not necessarily on the target variable if they are purely exogenous.
  - **Causal Temporal Mask (T):** This is a standard lower-triangular mask that ensures autoregressive behavior in the temporal dimension. It restricts attention for predicting a token at time t to only those tokens at or before time t, thus maintaining causality.

  The final attention mask applied within the self-attention layers is typically a combination of these two masks, often formulated using the Kronecker product: $\text{Mask}(C \otimes T)$. This composite mask allows for fine-grained control over which tokens can attend to which other tokens, respecting both pre-defined inter-variable dependencies and temporal causality.

During optimization, TimerXL typically computes the loss function only on the predictions for the target variable(s), even though exogenous variables form an integral part of the input and their interactions are modeled through the **Universal TimeAttention** mechanism.

Handling Diverse Exogenous Variable Types

The pre-specified Variable Dependency Graph (C) provides TimerXL with considerable flexibility in defining how different types of exogenous variables interact with target variables and with each other:

- **Dynamic Exogenous Variables (Past and Future-Known):** These time-varying exogenous variables are included as additional channels in the multivariate input. The structure of the C matrix explicitly dictates their influence. For past dynamic exogenous variables, C would allow target variables to attend to the historical values of these exogenous variables. For future-known dynamic exogenous variables (e.g., upcoming holidays, scheduled promotions, weather forecasts), the C matrix can be configured to allow target variable tokens to attend to future exogenous variable tokens. For example, to predict target y_t, the model could be allowed to see exogenous variable x_{t+H} if x_{t+H} is known at the time of prediction, where H is the relevant future horizon step. It's important to note that the Causal Temporal Mask T still ensures that the prediction of the target variable itself remains autoregressive with respect to its own past values.

- **Static Exogenous Variables:** Time-independent attributes (e.g., store ID, product category) can be incorporated by representing them as dynamic channels with constant values repeated over the entire relevant time horizon. Their interaction with other variables would again be governed by the definitions within the C matrix. The learnable scalar embeddings for the variable dimension can also help the model distinguish these static features.

Ablation studies highlighted in the TimerXL research, particularly on the Electricity Price Forecasting (EPF) task which inherently includes exogenous variables, demonstrated TimerXL's strong performance. These studies also indicated that enforcing causal dependency learning within the exogenous variables themselves (i.e., predicting future values of exogenous variables based only on their own past, rather than allowing them to see their own future non-causally if they were also being predicted) led to improved overall model performance. This underscores the robustness of the causal "next token prediction" paradigm extended to a multivariate setting.

Strengths and Limitations

Strengths:

- **Flexibility in Defining Dependencies:** The Variable Dependency Graph (C) allows users to encode domain-specific knowledge about inter-variable relationships, including complex interactions between targets and various types of exogenous variables.

- **Unified Handling of Long Contexts:** The Transformer architecture is well-suited for capturing dependencies over long historical contexts, which can be crucial for time series with long-range dependencies influenced by exogenous variables.

- **Sophisticated Attention Mechanism:** Universal TimeAttention with RoPE and learnable variable embeddings provides a nuanced way to process integrated multivariate sequences.

- **Strong Empirical Performance:** The model has demonstrated strong performance on challenging benchmarks that include exogenous variables.

Limitations:

- **Pre-specified Dependency Graph:** The C matrix must be defined beforehand by the user and is not learned by the model. The quality of forecasting can therefore depend significantly on the accuracy and completeness of this pre-specified graph. Designing an optimal C matrix can be challenging and may require substantial domain expertise.

- **Complexity of Interaction Details:** While flexible, the exact nature of how the model leverages the dependencies defined in C through the attention mechanism can be intricate and may not always be transparent for interpretation.

- **Flattened Sequence Processing:** While Universal TimeAttention aims to mitigate issues, processing a fully flattened sequence of (variable $\times$ time) patches might still pose challenges for very high numbers of variables or extremely long sequences in terms of distinguishing individual variable dynamics perfectly.

Conclusion

TimerXL presents a sophisticated and unified methodology for TSF , with a well-defined architecture for incorporating exogenous variables. By treating exogenous variables as additional variables within a flattened, patched sequence and employing the **Universal TimeAttention** mechanism with its **Kronecker-based Masking**, TimerXL can model intricate dependencies between target series and a diverse array of exogenous factors. The explicit control offered by the Variable Dependency Graph allows for the integration of domain knowledge, making TimerXL a powerful tool for exogenous variable-informed forecasting, though the reliance on a pre-specified dependency structure remains a key consideration in its application.

3.4. Exogenous Variable Integration Approaches

Having presented specific FMs and their methods for handling exogenous variables in Section 3.3, this section now categorizes and summarizes these diverse integration strategies. The effective integration of exogenous variables is paramount for achieving high accuracy and contextual relevance in TSF , as discussed in Section 2.5. FMs, designed for broad applicability, employ various sophisticated mechanisms for incorporating this external information, directly addressing a key aspect of Research Question F1.

3.4.1. Unified Input / Early and Deep Integration Strategies

This category encompasses approaches where exogenous variables are incorporated directly with the target series at the input stage of the FM. The combined representation is then processed by the main model architecture, allowing for deep, learned interactions between the target series and the exogenous variables from the earliest layers.

Feature Augmentation and Concatenation at Input

A fundamental and widely adopted strategy involves treating exogenous variables as additional features that augment the representation of the target time series at each input step or patch.

Description: Numerical exogenous variables are typically normalized and then concatenated with the (similarly normalized or embedded) target time series values or patches. Categorical exogenous variables are usually converted into numerical representations via embedding layers or one-hot encoding before this concatenation. This augmented feature vector or sequence of vectors then serves as the input to the FM's initial embedding layers or directly to its core processing blocks (e.g., Transformer attention layers, MLP layers). The FM is subsequently tasked with implicitly learning the relationships and predictive power of these integrated exogenous features alongside the target series' own dynamics.

Examples: Many standard deep learning models for TSF, including RNNs, LSTMs, and basic Transformer setups, commonly employ this method. For instance, the input vector z_t fed into an RNN at each time step might be formed as $[y_t, x_{1t}, \ldots, x_{Mt}]$, combining the target y_t with M exogenous variables (as noted for general RNNs in Section 2.5.3). While specific architectural details of **TimeGPT** (Section 3.3.3) are proprietary, its acceptance of exogenous variables as inputs alongside the target series suggests an approach aligned with this category, where these are processed together by its Transformer architecture. Models like N-HITS and NBEATSx also utilize concatenation of the primary time series with past and future exogenous variables to form a fixed-size input [J. Kim et al. 2024].

Homogenized Multi-Variate Processing with Specialized Mechanisms

This more sophisticated sub-category under unified input treats all time series—both targets and exogenous variables—as distinct 'variates' or 'channels' within a single, comprehensive multivariate input tensor. The FM architecture then employs specialized mechanisms to differentiate, manage, and learn interactions among these homogenized variates.

Description: Key components often include:

- **Variate Embeddings:** Learnable embeddings are assigned to each variate (target or exogenous variable) to provide the model with an identity for each channel in the multivariate stream.

- **Tailored Attention Mechanisms:** Specialized attention modules are designed to explicitly model cross-variate dependencies. These mechanisms can be guided by the variate embeddings or employ specific masking strategies to control information flow between different variates.

This approach allows the model to learn not just how exogenous variables affect a target, but also how exogenous variables might interact with each other in influencing the forecast.

Examples: **MOIRAI** (Section 3.3.5) exemplifies this with its AVA mechanism and variate ID embeddings. All input series (targets and exogenous variables) are patchified and processed as a unified sequence of tokens, with AVA dynamically learning inter-dependencies in its channel-mixed mode. **TimerXL** (Section 3.3.6) also fits this category. It processes a flattened, multidimensional sequence of patch tokens from both target series and exogenous variables. Its "Universal TimeAttention" uses RoPE for temporal information and learnable scalar embeddings for variate distinction. Crucially, its Kronecker-based masking, combining a user-defined Variable Dependency Graph (C) with a Causal Temporal Mask (T), allows for explicit, fine-grained control over how different variates (including exogenous variables) attend to each other.

3.4.2. Modular / Staged Integration Strategies

In these strategies, the influence of exogenous variables is handled by a distinct module or in a separate processing stage. The output or effect of this exogenous variable processing is then combined with the output or an intermediate representation from the core FM, which might primarily focus on the target series' intrinsic dynamics.

External Regression / Residual Modeling

This approach leverages a separate model, often a classical statistical or machine learning regressor, specifically to capture the impact of exogenous variables. The FM's role is then to model the aspects of the time series not explained by this external regressor.

Description: Two common variants exist:

1. The external regressor models the target series using exogenous variables ($Y_{actual} \sim f(x_t)$), and the FM is then trained to forecast the residuals ($R' = Y_{actual} - \hat{Y}_{xreg}$). The final forecast is $\hat{Y}_{xreg} + \hat{R}'_{FM}$.

2. The FM first forecasts the target series based on its history ($\hat{Y}_{FM}$). An external regressor is then trained to model the residuals ($R = Y_{actual} - \hat{Y}_{FM}$) using exogenous variables. The final forecast is $\hat{Y}_{FM} + \hat{R}_{xreg}$.

This decouples the learning of complex temporal patterns (by the FM) from the (often assumed simpler or linear) impact of exogenous variables.

Examples: **Chronos** and its variant **Chronos-Bolt** (Section 3.3.2), when used with exogenous variables (e.g., within frameworks like AutoGluon-TimeSeries), typically employ this strategy. A separate tabular model acts as the external regressor. **TimesFM** (Section 3.3.4) explicitly uses this approach with its 'xreg lib', offering both "timesfm + xreg" (variant 2 above) and "xreg + timesfm" (variant 1 above) integration options.

Dedicated Internal Refinement Modules / Adapter-like Blocks

This strategy involves distinct neural components, which can be seen as adapters or specialized blocks within or alongside the main FM architecture. These modules are specifically designed to process exogenous variable information and use it to modulate or refine the FM's internal representations or outputs.

Description: These blocks might take initial representations from the FM and combine them with processed exogenous variable features, or directly influence aspects like token embeddings or output logits. A key characteristic is often the modularity, allowing these blocks to be added to a pre-trained FM with only the adapter's parameters being fine-tuned.

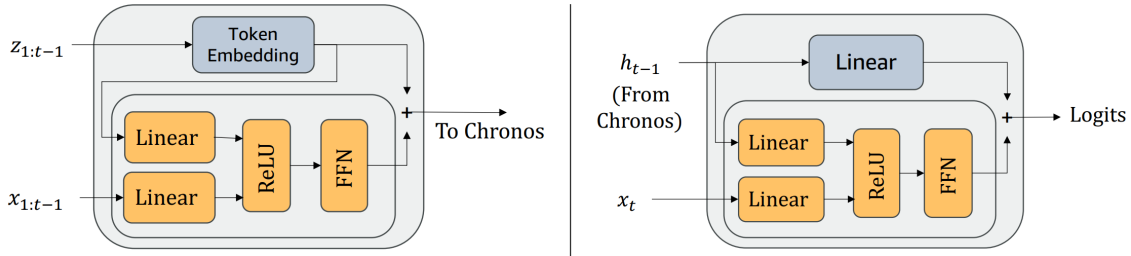
Examples: The **Exogenous Mixer Block** in **TTM** (Section 3.3.1) fits here. It takes an initial forecast from TTM's main head and known future exogenous variables, processes them through a TSMixer block with channel-mixing enabled, and produces a refined forecast for the target channels. The **ChronosX** framework, proposed by Arango et al. [2025] as an extension for pre-trained models like Chronos, exemplifies this with its modular adapter design. It utilizes:

- An **Input Injection Block (IIB):** This block updates the pre-trained token embeddings using information from past exogenous variables. It typically involves passing the original token embeddings and the exogenous variable features through independent linear layers, concatenating their outputs, and then processing the result with a FFN.
- An **Output Injection Block (OIB):** This block uses future exogenous variables to directly modulate the output logits (or pre-final activations) of the pre-trained FM before the final prediction is generated.

Figure 3.7 visually illustrates the architecture of these ChronosX blocks, detailing how both past and future covariates are integrated with the pre-trained model's components. These ChronosX blocks are typically lightweight and are the primary components fine-tuned, keeping the main FM backbone frozen. This adapter-based methodology is designed to be extensible to other FMs, with modifications for specific input types like patches for models termed TIMESFMX or MOMENTX [Arango et al. 2025].

3.4.3. Complex Hybrid Architectures with Explicit Exogenous Variable Pathways

This category is reserved for models featuring highly specialized, often multi-component, architectures with distinct and explicit pathways designed to ingest, process, and in-



(a) Input Injection Block (IIB) (b) Output Injection Block (OIB)

Figure 3.7.: Figure (a) shows that the input injection block takes a pretrained tokenized embedding together with covariates of the past, whereas Figure (b) takes the pretrained logits (expressed as the final hidden state multiplied by a pretrained matrix) together with covariates of the future. The blue color indicates that the module is taken from the pretrained model [Arango et al. 2025]

tegrate different types of exogenous variables (static, past-observed dynamic, future-known dynamic) at various stages of the network.

Description: These architectures go beyond simple concatenation or single dedicated blocks, often employing techniques like separate encoders for different input streams (e.g., one for target history, another for dynamic exogenous variables), variable selection networks to weigh the importance of different inputs, and sophisticated gating mechanisms or fusion layers to combine information from these diverse pathways. The aim is to allow the model to learn very nuanced interactions and leverage the specific nature of each type of exogenous variable optimally.

Examples: The **TFT**, discussed in Section 2.3.3 of Chapter 2, is a prime example. It uses GRNs, dedicated Variable Selection Networks for static, past dynamic, and future-known dynamic inputs (including exogenous variables and the target itself), and specific encoders to create rich representations that are then processed by an LSTM-based sequence-to-sequence component and a self-attention layer. Models like **NBEATSx** and **N-HITS**, which extend their base architectures to handle exogenous variables explicitly [Olivares, Challu, et al. 2023], and specialized models like **DeepAR** [Gasthaus et al. 2019], also often incorporate distinct mechanisms for processing and integrating exogenous variable features throughout their architectures.

3.4.4. Discussion and Comparison of Approaches

The choice of an exogenous variable integration strategy in FMs involves trade-offs and depends on various factors, including the nature of the exogenous variables, the complexity of their relationship with the target series, the architecture of the core FM, and computational considerations.

Unified Input / Early and Deep Integration Strategies (Category 1) offer the potential for the FM to learn complex, non-linear interactions between exogenous variables and the target series end-to-end. Specialized attention mechanisms within this category (e.g., MOIRAI, TimerXL) provide sophisticated ways to manage and

leverage multivariate dependencies. However, these approaches might require more data to learn these interactions effectively and can sometimes make it harder to isolate the specific contribution of individual exogenous variables. The homogenization of all inputs also means the model must internally differentiate the roles of various inputs.

Modular / Staged Integration Strategies (Category 2) provide flexibility and can be easier to implement, especially when augmenting existing pre-trained FMs (as seen with ChronosX adapters or the external regressor approach for Chronos and TimesFM). External regression allows leveraging well-understood models for exogenous variable effects, which might be primarily linear or simpler to model separately. Dedicated refinement modules like TTM's Exogenous Mixer can offer a more integrated neural approach than fully external models while still maintaining some modularity. A potential drawback is that by decoupling the exogenous variable modeling from the core FM's deep temporal learning, some intricate, deeply intertwined non-linear dynamics might be missed. The effectiveness of residual modeling also depends on the assumption that the effects are largely additive.

Complex Hybrid Architectures with Explicit Pathways (Category 3), such as TFT, represent the most tailored approach, designed to meticulously handle different types of exogenous variables through specialized components. This can lead to high performance and better interpretability regarding variable importance. However, these architectures are often more complex to design, train, and potentially adapt to new types of exogenous variables not initially considered.

Ultimately, the most suitable integration approach depends on the specific FM architecture, the characteristics of the available exogenous data (e.g., availability of future values, linearity of impact), and the desired balance between predictive performance, model complexity, interpretability, and ease of implementation. The ongoing research in FMs for TSF continues to explore more effective and robust ways to leverage the rich contextual information provided by exogenous variables.

4. Experimental Setup

This chapter details the comprehensive methodology employed to conduct the empirical evaluation of FMs for TSF with exogenous variable integration. It covers the selection and preparation of datasets, the choice and configuration of all models, the definition of various exogenous variable scenarios, the execution of forecasting experiments and the methods used for evaluation.

4.1. Dataset Selection and Description

This section outlines the criteria guiding the selection of time series datasets, the preprocessing steps applied to unify their format, and the selection of time series from those datasets based on their characteristics.

4.1.1. Dataset Selection Criteria

Given this work's focus on exogenous variable integration, datasets were exclusively selected if they contained such external variables. A notable challenge immediately identified was the scarcity of publicly available time series datasets rich in exogenous variables. Consequently, traditional forecasting challenge datasets were largely unsuitable, as they predominantly focus on univariate and, at times, multivariate forecasting scenarios [Aksu et al. 2024; Hahn et al. 2023; Li et al. 2024]. While multivariate series can be adapted by designating some variables as exogenous variables rather than targets, the preference was for datasets explicitly providing them.

As discussed in Section 2.5.2), various types of exogenous variables exist. The selected datasets were chosen to represent a broad spectrum of dynamic types. Specifically, static exogenous variables, those constant over the entire period, were excluded. The decision to exclude them stems from two primary considerations: firstly, their inclusion significantly increases complexity in standardizing data loading and preprocessing pipelines to ensure a unified and consistently processable format across diverse datasets. Secondly, while multiple static exogenous variables can represent hierarchical relationships and potentially improve overall model performance, managing the distinct ways different FMs might internally support or require specialized handling for them would broaden the scope of integration challenges beyond the primary focus on dynamic exogenous variables in this study. Therefore, they were deliberately excluded to streamline these practical and analytical complexities.

Furthermore, attention was paid to selecting datasets exhibiting diverse characteristics, including varying numbers of variables, time series lengths and granularities. All chosen datasets are publicly accessible to ensure reproducibility. Detailed descriptions of each dataset, including their original sources and specific characteristics, are provided in Appendix A.

4.1.2. Dataset Preprocessing

To achieve a unified format across all diverse datasets and facilitate their seamless loading and preparation for various forecasting models, a series of preprocessing steps were meticulously performed. Missing values, a common challenge in time series data (as discussed in Section 2.1.2), were addressed systematically. The following steps were applied:

- All raw data files were loaded and combined into a single pandas DataFrame through concatenation and merging operations.
- Timestamps were standardized to the 'YYYY-MM-DD HH:MM:SS' format. For granularities of daily and above (e.g., weekly, monthly), the hourly to second-level components of the timestamp were truncated.
- DataFrames were restructured by pivoting improperly formatted datasets into a consistent layout: a "date" column for timestamps, a "ts_name" column as a unique identifier for each time series within a dataset and all other columns representing target or exogenous variable series with their variable names as column headers.
- Missing values were addressed systematically. An initial step involved generating heatmaps to visualize missing observations across all series within a dataset, providing an overview of the extent and patterns of missingness. For instance, Figure 4.1 illustrates such a heatmap for the Acea Water dataset, where a significant block of missing values across most variables from the beginning of the record up to the end of 2010 is evident. Consequently, this entire period was trimmed from the dataset. Following trimming, if internal missing values persisted, a more detailed analysis was conducted for each affected series. The series was plotted with various imputation methods overlaid to qualitatively assess the best fit. Figure 4.2 demonstrates this process for the 'Depth_toGroundwater_LT2' series from the Acea Water dataset (post-trimming), comparing zero imputation, mean imputation, forward fill and linear interpolation. This qualitative selection of an appropriate imputation technique, guided by the aim of preserving local data characteristics such as trends or seasonality evident in the non-missing segments, was applied to remaining series with internal missing values.

Following these preprocessing steps, a set of key characteristics were determined for each dataset to create a comprehensive metadata overview. These features include the dataset's domain, granularity, the length of its time series and the number of target variables. The count of exogenous variables was also cataloged, distinguishing between those known only in the past, those known in the future and categorical variables, which could belong to either the past or future category.

It is important to note that while missing values were handled through trimming and imputation due to most models' inability to process NaNs internally, outliers were deliberately not subjected to specific preprocessing steps. Figure 4.2, for example, shows a potential outlier at the end of the 'Depth_toGroundwater_LT2' series, where the value jumps to 0. The expectation is that the forecasting models themselves should be robust to or capable of handling such outliers.

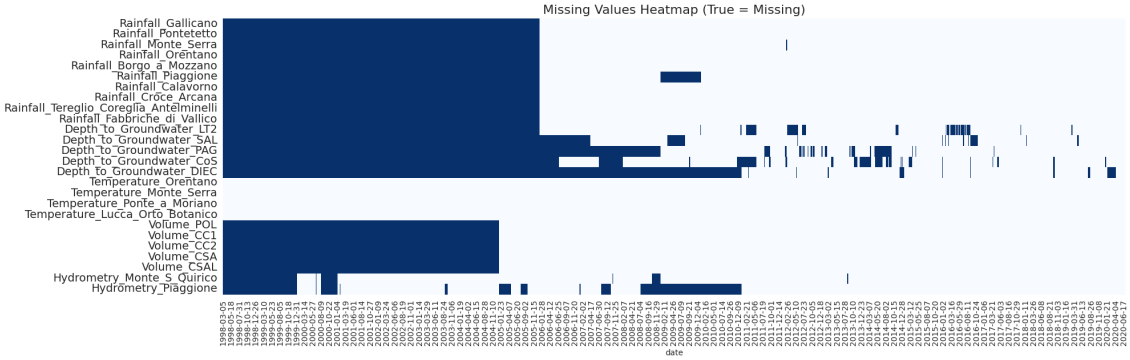


Figure 4.1.: Heatmap of missing values of the Acea Water dataset, illustrating a large segment of missing data prior to 2011 that was subsequently trimmed.

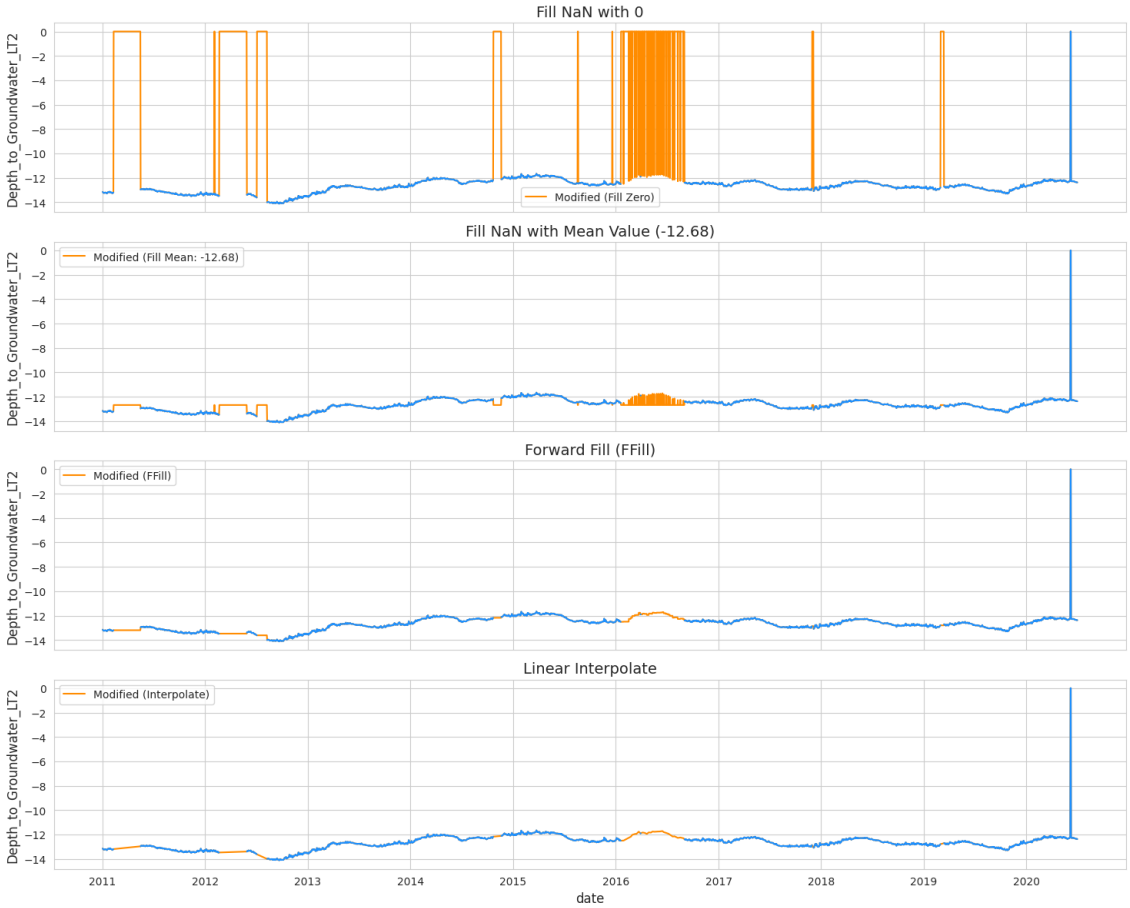


Figure 4.2.: Time series plot with different imputation methods (zero fill, mean fill, forward fill, linear interpolation) for the 'Depth_toGroundwater_LT2' target value from the Acea Water dataset (after initial trimming). Linear interpolation was qualitatively selected as the most suitable.

4.1.3. Presentation of Used Datasets

A curated collection of 29 publicly accessible datasets was initially compiled, chosen for their inclusion of external exogenous variable information and diverse characteristics. Detailed descriptions and sources for all 29 datasets are available in Appendix A. However, due to limitations in obtaining sufficient forecast results for a subset of these (as will be further detailed in Section 4.1.4), the descriptive statistics and visualizations presented in this subsection focus on the 20 datasets that were carried through the complete experimental evaluation.

These 20 datasets span various domains, ensuring a diverse representation for the forecasting tasks. As shown in Figure 4.3, the distribution includes 7 datasets from the nature domain, 6 from energy, 3 related to human activity, 2 from economics and 2 concerning demand forecasting. This selection covers fields where exogenous variables are commonly utilized and impactful. It represents the main application areas of TSF while also ensuring a wide variety of data characteristics. Each domain contributes not only distinct time series properties but also a unique set of relevant exogenous variables. For example, energy forecasting often relies on weather data, economic predictions may use interest rates and inflation and demand series are influenced by promotions and pricing. This diversity is further highlighted by the inclusion of sporadic time series, which are prevalent in demand datasets but less common elsewhere. Consequently, this collection provides a robust basis for evaluating a model's ability to handle different data structures and effectively leverage external information.

The granularities of these datasets also vary, as depicted in Figure 4.4. The collection comprises 11 hourly, 7 daily, 1 weekly, and 1 quarterly dataset. It was observed that datasets with coarser granularities (weekly, monthly, yearly) rich in exogenous variables were relatively scarce. Furthermore, such datasets often featured very short time series, posing challenges for robust evaluation with multiple forecast windows and for effective model training.

The characteristics of these 20 datasets exhibit considerable diversity. The time series lengths range significantly, from a minimum of 74 to a maximum of 100,057 data points, with a median length of 4,343.5 points. Figure 4.5 provides a visual summary of other key characteristics using boxplots: the number of unique time series per dataset (series distinguished by a unique identifier), the number of target variables, the number of available past exogenous variables (historically observed), and the number of available future-known exogenous variables. Illustratively, the number of unique series per dataset ranges up to 185, the number of target variables to a maximum of 123, the count of past exogenous variables from 0 to 21 and future exogenous variables from 0 to 5 with a median of 0, indicating many datasets lack future-known exogenous variables beyond generated temporal features. The majority of exogenous variables is only historically observed over all datasets.

This structured presentation of dataset characteristics highlights the heterogeneity of the selected data, which is crucial for a comprehensive evaluation of forecasting models.

4.1.4. Time Series Characterization and Selection

The initial 29 selected datasets, when considering each distinct series identifier and target variable combination, collectively encompassed 1492 unique target time se-

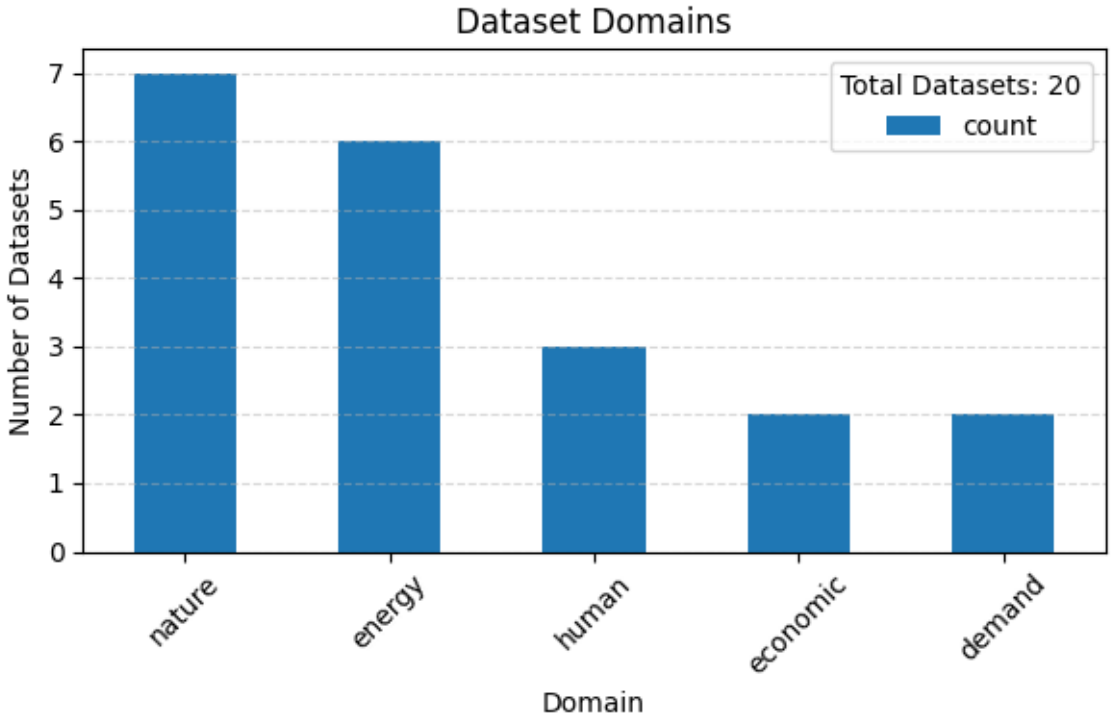


Figure 4.3.: Distribution of the 20 analyzed datasets across different domains.

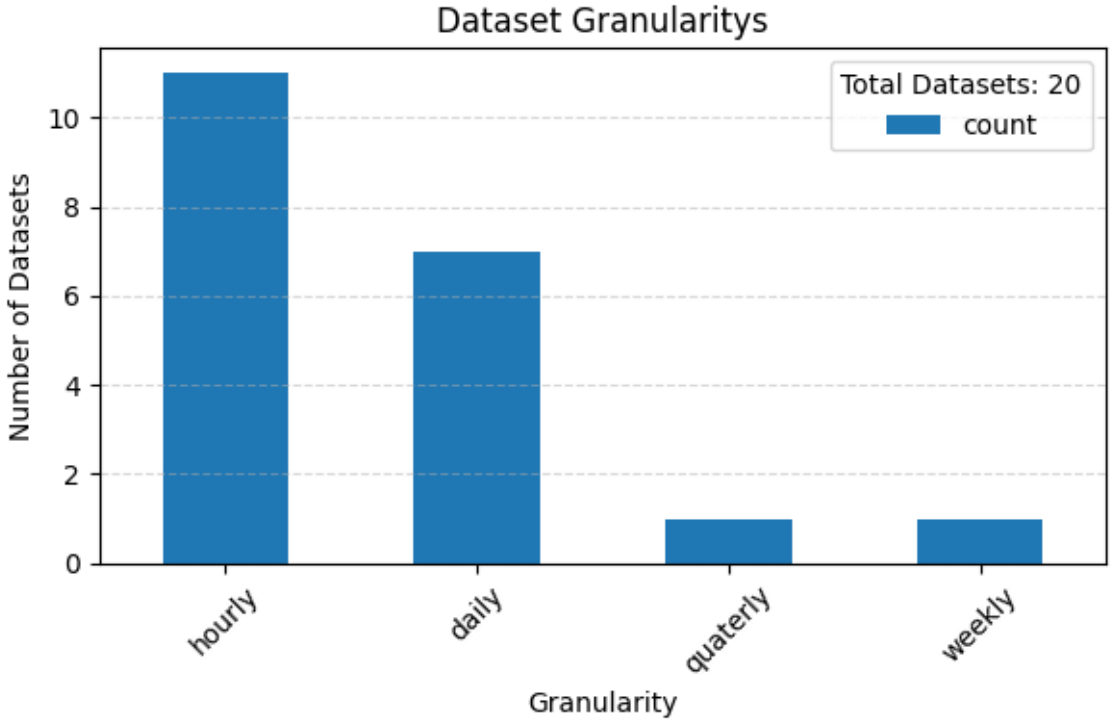


Figure 4.4.: Distribution of granularities for the 20 analyzed datasets.

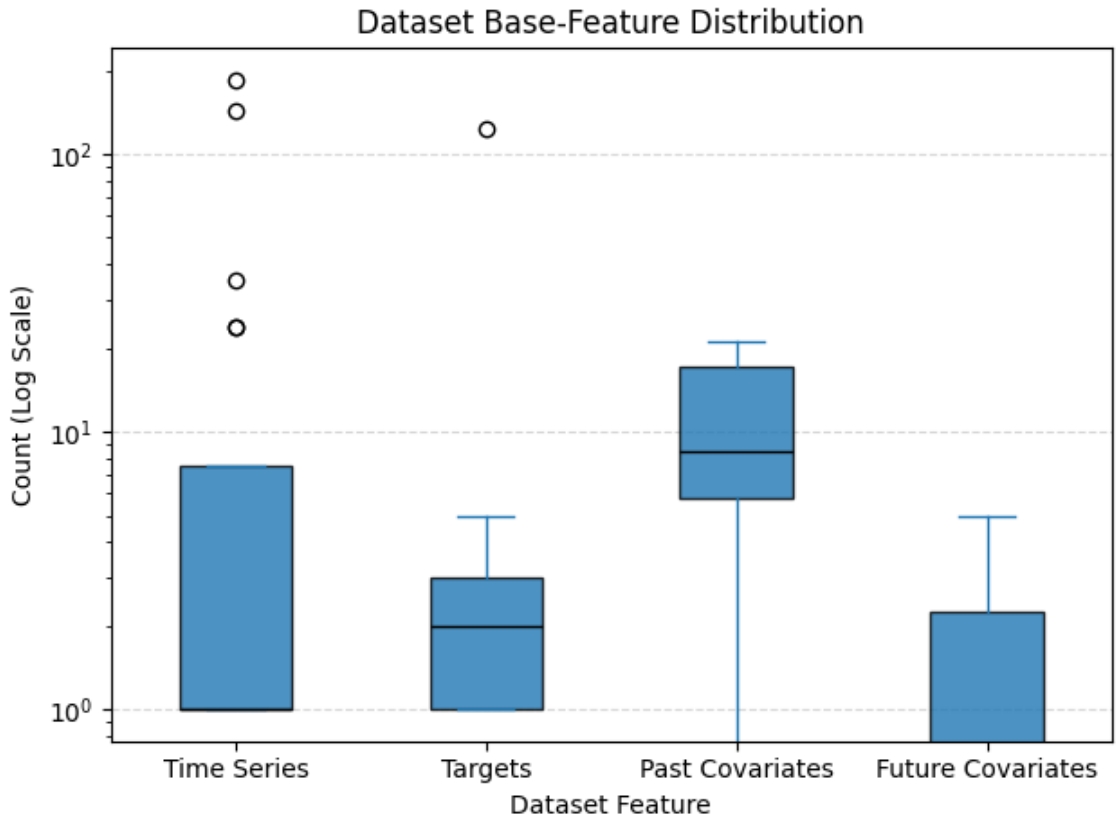


Figure 4.5.: Boxplot distributions of key characteristics for the 20 analyzed datasets: number of unique series, targets, past exogenous variables and future exogenous variables.

ries. To characterize these series, a set of features capturing diverse aspects such as trend strength, seasonality, autocorrelation, stability and complexity was calculated for each one using the Python package "tsfeatures". A detailed description of the calculated features is provided in Appendix B. This library is based on the original tsfeatures package for R, developed by Rob J. Hyndman [Hyndman, E. Wang, et al. 2015].

Based on these calculated features, the dissimilarity between every pair of target time series y_i and y_j was computed using a normalized Euclidean distance metric. The selection of specific features for this distance calculation follows the methodology of the Libra benchmark [Bauer et al. 2021], where significant effort was dedicated to analyzing and ensuring time series diversity based on a comprehensive feature set. The distance, $D(y_i, y_j)$, defined as:

$$D(y_i, y_j) = \sum_{m=1}^{M} \frac{|c_{y_i,m} - c_{y_j,m}|}{\max_k(c_{y_k,m}) - \min_k(c_{y_k,m}) + \epsilon}$$

where M is the number of selected features, $c_{y_i,m}$ is the m-th feature of time series y_i, and $\max_k(c_{y_k,m})$ and $\min_k(c_{y_k,m})$ are the maximum and minimum values of the m-th feature across all k time series, respectively. A small constant ϵ (e.g., 10^{-6}) is added to the denominator to prevent division by zero in cases where a feature has zero variance across all series.

Given the large number of initial time series and the constraints on computational resources and time for model benchmarking, a smaller, more heterogeneous subset was selected using a greedy Farthest Point Sampling (FPS) algorithm [Eldar et al. 1997]. This algorithm initiates by selecting the time series that has the largest sum of distances to all other series. Subsequently, it iteratively selects the next series that is farthest from the set of already selected series. A crucial aspect of this implementation was the prioritization of datasets not yet adequately represented, ensuring that at least five time series from each of the 29 original datasets was included in the selection or all available series if a dataset had fewer than five. This process resulted in a candidate set of 150 target time series.

The effectiveness of the FPS algorithm in selecting a diverse set of time series is illustrated in Figure 4.6. The figure presents boxplots of the pairwise distances between time series before and after the FPS selection. Before filtering, the distribution of distances shows many series with low dissimilarity with a median distance of approximately 2000. After applying FPS, the selected time series exhibit a notably higher overall distance distribution (median distance approximately 7000), indicating increased heterogeneity.

Following this selection process and after conducting initial experimental runs, it was determined that for 9 of the 29 datasets, sufficient and consistent forecast results could not be obtained across all models and scenarios. Consequently, these 9 datasets, along with the time series selected from them, were excluded from the final analysis. This resulted in a final experimental set of 117 time series, drawn from the remaining 20 datasets, upon which the evaluations in this thesis are based. It is important to note that the "filtered" distribution shown in the Figure 4.6 already reflects the final selection of 117 time series from the 20 datasets used in the subsequent analysis.

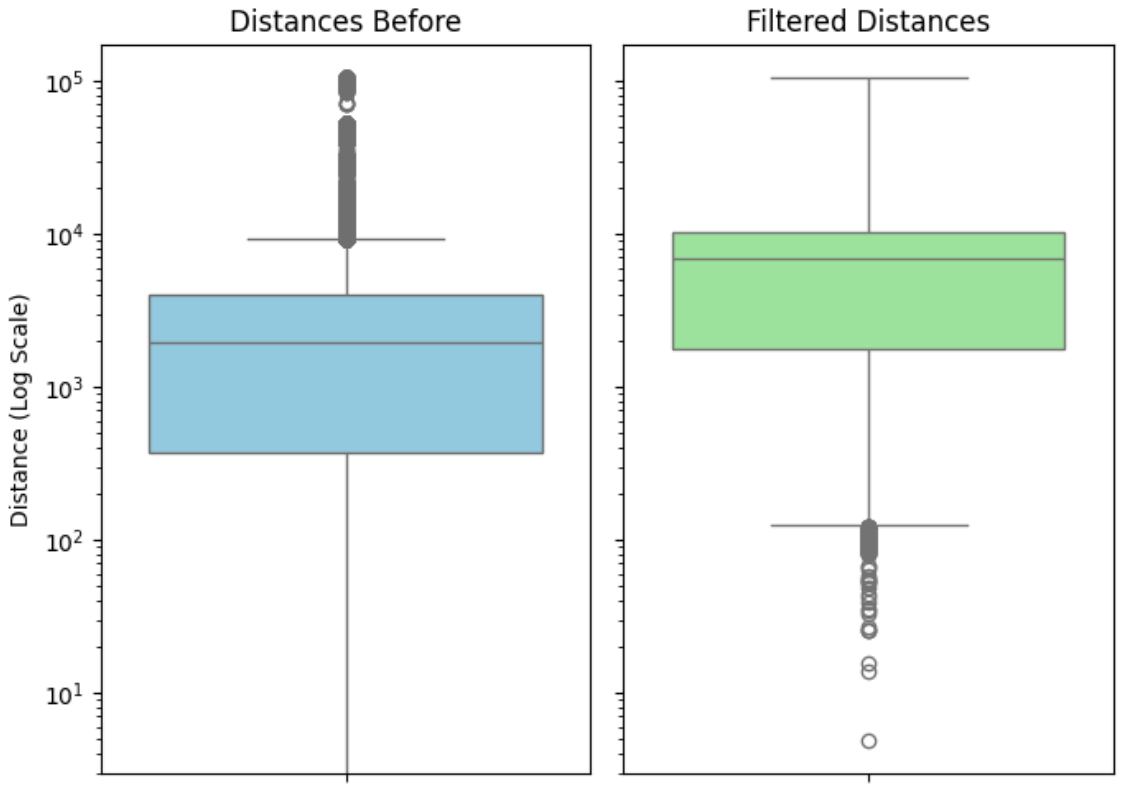


Figure 4.6.: Distribution of pairwise normalized Euclidean distances between target time series before and after selection.

4.2. Model Selection and Configuration

This chapter details the rationale behind the selection of both FMs and comparison models, along with their implementation specifics and hyperparameter configurations.

4.2.1. Foundation Model Selection

The selection of FMs for this study focused on models offering robust support for exogenous variables, as this is a key feature not yet universally available. The models chosen for evaluation, based on their architectural integration of external data or well-supported regressor frameworks, include:

- **Tiny Time Mixers:** Selected due to its lightweight nature and direct exogenous variable integration capabilities via its Exogenous Mixer, see Section 3.3.1). For this work, the pretrained weights from the model's `R2` release were utilized, chosen for their pretraining on a more extensive dataset with a wider range of granularities. The model was evaluated in two configurations: a **zero-shot** variant using the weights directly, and a **fine-tuned** variant. To reduce the overall training time, the fine-tuning process was constrained by a maximum number of training steps.
- **MOIRAI:** Chosen for its direct integration of exogenous variables within its flexible architecture via Any-variate Attention, as described in Section 3.3.5.

For the evaluation, the `base` size models were chosen as this size offers an optimal balance between accuracy and computational efficiency compared to the other available versions. Specifically, the `1.1-R-base` and its **Mixture of Experts** variant `moe-1.0-R-base` were evaluated.

- **TimeGPT:** Included as a prominent commercial FM that supports exogenous variables via its forecasting client provided by Nixtla (see Section 3.3.3), although the underlying model itself is not open-sourced. Both **zero-shot** use and a **fine-tuned** variant were evaluated. For long-horizon scenarios, a dedicated long-term forecasting model version offered by the service was utilized.

- **TimesFM:** A univariate model by design (see Section 3.3.4), which integrates exogenous variables through an external regressor framework provided in its associated packages. This framework offers flexibility and the `timesfm + xreg` configuration was specifically tested, aligning with the External Regression / Residual Modeling approach detailed in Section 3.4.2. Specifically, the `version 2.0-500m`, was employed for this study, as it features a larger parameter size and offers enhanced predictive accuracy.

- **Chronos:** Another primarily univariate model (see Section 3.3.2) that supports external regressors, notably via integration with the AutoGluon package. It was evaluated with both **Linear Regression** and **LightGBM** as the regressor. The `Bolt` variant was specifically chosen for this work due to its excellent computational efficiency, even on CPUs, and its improved accuracy over the original version. Furthermore, the `base` model size was selected to ensure the highest predictive performance.

Other FMs discussed in Chapter 3 were considered but ultimately excluded. TimerXL was excluded because no pre-trained model was publicly available at the commencement of the experiments. Similarly, the promising `injection block` approach from ChronosX was not included, as its official implementation had not been released at the time of this study.

4.2.2. Selection of Comparison Models

To contextualize the performance of the FMs, a range of comparison models was chosen, encompassing statistical, ML, and DLparadigms.

For a baseline representing purely univariate forecasting, the following classical statistical models were included (see Section 2.3.1):

- **ETS:** A family of exponential smoothing models valued for their robustness.

- **Theta Method:** A simple yet effective method known for its strong benchmark performance.

For comparison against models capable of using exogenous variables, the following were selected:

- **Statistical - (S)ARIMAx:** An automated version of the popular ARIMA model (see Section 2.3.1), capable of handling seasonality (S) and incorporating exogenous variables (x).

- **ML - LightGBM:** A high-performing gradient boosting machine (see Section 2.3.2), recognized for its efficiency and effectiveness with tabular data structures.
- **DL - NBEATSx and Time-series Dense Encoder (TiDE):** The ChronosX benchmark [Arango et al. 2025] identified Neural Basis Expansion Architecture with Exogenous Variables (NBEATSx), TiDE, and TFT as particularly promising DLmodels. Given computational resource limitations, this study focused on the MLP-based NBEATSx and TiDE (see Section 2.3.3), as they offer a less computationally intensive solution than the Transformer-based TFT.

The number of comparison models was deliberately kept manageable to maintain the primary focus on the in-depth analysis of FMs.

4.2.3. Implementation and Hyperparameter Configuration

Models were categorized based on their optimal execution environment, with DLand FMs primarily designed to run on GPU infrastructure, while statistical and traditional ML models typically execute efficiently on CPU resources. Dedicated Docker containers were set up for both environments, operating on the company's respective CPU or GPU clusters.

GPU Environment and ProgAI API

The GPU environment is managed by a FastAPI application, exposing three primary endpoints:

- `GET /model`: Provides detailed information about the currently loaded model and its hyperparameters.
- `POST /model`: Allows switching between different models or updating their configurations and hyperparameters based on a provided model path.
- `POST /forecast`: Facilitates forecasting on a provided dataset. It performs input validation and preprocessing specific to the chosen model, executes the model inference, processes the output into a unified format, and returns the results.

The API is designed to process all requests sequentially and locks the model during operations to ensure safe forecasting and manage limited GPU resources effectively.

Models on GPU:

- **DL Models (TiDE and NBEATSx):** Integrated using the `neuralforecast` package by Nixtla. Given that no hyperparameter tuning was performed, both models utilized the package's default settings, which are designed to be robust, general-purpose configurations. The `batch size` was reduced to 16 for efficiency across various dataset sizes and an `early stopping patience` of 10 was used to strike a balance between preventing significant overfitting and avoiding premature termination of the training process due to minor fluctuations in validation loss. The model `input length` for both models, a crucial

parameter for handling different granularities, is dynamically determined based on the forecast horizon length (further details in Appendix A).

- **TTM:** Implementation was based on the collection of pretrained weights from the Hugging Face path `ibm-granite/granite-timeseries-ttm-r2`. The specific model is chosen programmatically using a utility function from the TTM package. The default behavior of this function is to select the model with the longest possible context length that fits the input series. While suitable for zero-shot forecasting, this is suboptimal for fine-tuning, where more training steps are beneficial. Therefore, for the fine-tuning scenario, the selection logic was modified to deliberately choose a model with a shorter context length, thereby increasing the number of training samples available from each series. The model was evaluated in both its zero-shot and fine-tuned forms, with fine-tuning performed for a maximum of 2000 steps per series.

- **TimesFM:** The `google/timesfm-2.0-500m-pytorch` version is utilized, specifically testing the "timesfm + xreg" configuration which employs the external linear regressor for exogenous variable integration. A crucial preprocessing step is required for past-known covariates: the main TimesFM model first forecasts these variables into the required horizon. These forecasted values are then used as inputs by the linear regressor for the final target prediction, a two-stage approach that carries a potential risk of error propagation. To ensure the model could handle long-term forecasting scenarios, the `horizon len` parameter was set to 256.

- **MOIRAI:** The evaluation included two pre-trained versions of the model: the standard `Salesforce/moirai-1.1-R-base` and the `Salesforce/moirai-moe-1.0-R-base` MoE variant. The `patch size` hyperparameter was set to 'auto', a recommended configuration that allows the model to dynamically select an appropriate patch size based on the granularity of the input time series.

The implementation of all models closely followed the code samples, tutorials and best practices provided by their respective authors and packages. While key hyperparameters have been noted in this section, for any parameter not otherwise specified, the default value from the corresponding library or package was used.

Initially, Chronos was planned for the GPU setup. However, minor dependency issues during setup prompted a re-evaluation of its placement. Given that a key feature of the newer `Bolt` variant is its excellent performance on CPUs, the decision was made to shift it to the CPU container. This transition not only resolved the initial setup challenge but also provided a valuable opportunity to test its advertised efficiency on more accessible hardware.

CPU Environment and Core Software Components

The CPU environment hosts the main project, which orchestrates the entire experimental workflow. This includes managing data, defining experimental runs, making API calls to the GPU service for DL and applicable FM forecasts and running CPU-native models.

Models on CPU:

- **Statistical Baseline Models:** Automated versions of ETS, Theta Model (Theta) and SARIMA are integrated from the `statsforecast` library by Nixtla, utilizing their "Auto" models for internal model selection and hyperparameter tuning.
- **Statistical SARIMAX Model:** The AutoARIMA model from `statsforecast` is used. When no exogenous variables are provided, it serves as a univariate SARIMA baseline. When exogenous variables are provided, it functions as an exogenous variable-aware SARIMAX model.
- **ML Model (LightGBM):** Integrated via the `autogluon` library, which provides automated hyperparameter tuning and model selection capabilities.
- **Chronos-Bolt:** The `bolt base` variants, integrated via the `autogluon` package, were used. It supports external regressors for exogenous variable integration and was evaluated with two different regressor configurations: one using Linear Regression Linear Regression (LR) and another using LightGBM GBM. This includes a preprocessing step to forecast past exogenous variables into the forecast horizon for the regressor, similar to TimesFM.

Internal Software Components (Classes): The internal architecture of the main project is structured around several key Python classes designed to manage the data flow and experimental execution:

- **Dataset:** Responsible for loading and providing preprocessed datasets as a pandas DataFrame, including exogenous variable information and granularity.
- **CovariateSelection:** Implements the procedure for selecting exogenous variables based on the LightGBM methodology for various lags, as detailed in Section **??**.
- **Samples:** Represents a unique combination of a dataset, a time series identifier `ts_name`, a context window, and a forecast horizon. It also holds additional metadata such as target names and a unique `ts_id`.
- **Dataloader:** An iterable class that loads and prepares time series data. For each sample defined in a provided `sample_info_csv`, it yields a `ModelInput` object. The preparation of this object is configured by the chosen exogenous variable mode (described in Section 4.3.1).
- **Models Directory:** A collection of classes to handle forecasting with the different models:
  - **ModelInput:** A data schema class that defines a standardized structure for model input. It encapsulates the time series DataFrame and its associated parameters, and provides methods to convert the data into the specific formats required by different forecasting libraries like Nixtla, AutoGluon, and the internal ProgAI API.
  - **Baseline Models:** Implementations for interacting with simple statistical models like AutoETS, AutoTheta, and AutoARIMA.

  - **Other Models:** Includes implementations for the MLmodel LightGBM and the FM Chronos-Bolt via AutoGluon.
  - **ProgAI:** A class that provides an API client for interacting with the GPU-based DLand FMs served by the FastAPI application.
  - **TimeGPT:** A dedicated API wrapper class for the TimeGPT forecasting service.

- **ForecastExecution:** Orchestrates the end-to-end forecasting pipeline. It manages model configurations and takes a `Dataloader` instance to iterate through all data sample, model and covariate mode combinations. A key feature is its intelligent skipping logic, which allows it to efficiently resume interrupted runs and automatically bypass configurations that consistently fail. It handles model execution and batch-wise storage of results seperated in meta- and forecast data and employs a structured logging system. To facilitate easier error investigation, traceback logs are linked via unique identifiers, a design that also supports potential future automation of error analysis.

- **Evaluation:** Handles the end-to-end evaluation pipeline. It loads generated forecasts, associated metadata, and ground truth actuals to compare predictions against real values. The class then calculates a range of performance metrics (detailed in Section 4.5), aggregates these results by different dimensions such as model, mode or dataset, processes statistical tests and provides various plotting functionalities for visualization.

- **Main:** Coordinates the overall forecasting and evaluation tasks, using multiprocessing to separate API call processing from CPU-intensive model training and evaluation.

The complete source code developed for these experiments is provided as a supplement to this thesis. It is designed to be largely self-explanatory, with detailed docstrings and comments for all major components. It should be noted that the description here focuses on the main orchestration project. The accompanying project for the GPU API is simpler in structure, consisting mainly of the FastAPI application, wrappers for the individual models, and schemas for standardized input and output. While this section outlines the core architectural classes of the main project, it omits, for brevity, numerous helper functions. The overall project is structured following modern software engineering best practices to ensure readability, maintainability and reproducibility.

4.3. Exogenous Variable Scenario Definition

This section details the various exogenous variable scenarios defined for the experimental evaluation, highlighting the preparation steps for exogenous variables within these scenarios and the methodology used for specific exogenous variable selection techniques.

4.3.1. Overview of Exogenous Variable Scenarios

To thoroughly assess the impact of exogenous variables and the capabilities of different models to integrate them, the following distinct scenarios were defined for each time series in the experimental dataset:

- **No Exogenous Variables:** Only the historical values of the target time series are used for forecasting. This serves as a baseline for each model.
- **All Exogenous Variables:** All exogenous variables available in the specific dataset for a given time series are included in the forecast.
- **Permuted Exogenous Variables (Noise):** All available exogenous variables are included, but their values are randomly permuted (shuffled) independently for each exogenous variable series across its entire length. This scenario aims to test model robustness against irrelevant or noisy exogenous variable inputs and to understand if models spuriously assign importance to such inputs.
- **Past-Only Exogenous Variables:** Only exogenous variables whose values are known up to the current forecast origin are included.
- **Future-Only Exogenous Variables:** Only exogenous variables whose future values are known or assumed to be perfectly forecasted for the entire forecast horizon are included.
- **Temporal Exogenous Variables:** Only time-based exogenous variables like day of the week, month of the year, and special holidays, which are typically future-known, are included (see Section 2.5.2). The number and type of these temporal features are directly dependent on the granularity of the time series. For instance, while a monthly series might only have a "month of year" feature, a daily series can additionally include "day of week," and an hourly series can further incorporate an "hour of day" feature.
- **Lagged Target Variables as Exogenous Variables:** Only lagged versions of the target variable itself are used as exogenous features (as discussed in Section 2.3.2), with specific lag selections based on time series granularity.
- **Selected Exogenous Variables:** A subset of available exogenous variables (excluding temporal and lagged target features) is chosen based on the selection methodology described in Section 4.3.3.
- **Selected Exogenous Variables with Lags:** A subset of available exogenous variables is chosen along with their optimal historical lags, based on the selection methodology in Section 4.3.3.

4.3.2. Exogenous Variable Preparation and Preprocessing for Scenarios

Specific preprocessing and generation steps were performed to prepare exogenous variables for the defined scenarios:

- **Permutation of Exogenous Variables:** For the "Permuted Exogenous Variables" scenario, the values of all existing non-temporal exogenous variables for a given time series were independently and randomly permuted across the complete time series length. This ensures that any inherent temporal relationship between an exogenous variable and the target series is broken, effectively introducing noise.

- **Past-Only and Future-Only Exogenous Variables Identification:** These scenarios rely on the inherent nature of the exogenous variables within each dataset. Datasets were chosen to ideally include a mix, but it's acknowledged that not all datasets contain clearly distinguishable past-known versus future-known dynamic exogenous variables.

- **Temporal Exogenous Variables Generation:** Future-known temporal exogenous variables (e.g., day of week, day of month, month of year, week of year) were generated using the `gluonts.time_feature.time_features_from_frequency_str` function, which derives these features based on the dataset's granularity.

- **Lagged Target Variable Generation:** Lagged versions of the target variable were generated using the `gluonts.time_feature.get_lags_for_frequency` function. This involved selecting a set of appropriate lags based on the series granularity, typically including recent lags and key seasonal lags (+/- delta) with a maximum of 7 different lags per target. For example, for weekly data, this includes lags $[1, 2, 3, 8, 12, 51, 52]$ representing the three most recent weeks, approximate bi-monthly/quarterly patterns and the primary annual seasonality.

4.3.3. Exogenous Variable Selection Methodology

The selection of exogenous variables for the `Selected Exogenous Variables` and `Selected Exogenous Variables with Lags` scenarios was implemented using a unified, model-agnostic approach: A LightGBM model (described in Section 2.3.2) served as the core mechanism for feature ranking. For each individual dynamic exogenous variable and for a predefined range of historical lags (specifically, lags 0 to 6), forecasts were generated. The decision to test this narrow range of lags was a pragmatic choice to limit the computational expense of the selection process, while still being a suitable range for capturing short-term effects across many common granularities (e.g., daily lags within a week or quarterly lags within a year). This was done by training the LightGBM model on an 80/20 train-test split of the target series, using only the single exogenous variable-lag combination in question as an exogenous feature at a time.

The performance of each exogenous variable-lag combination was then evaluated using the MASE metric (defined in Section 2.4.3) on the test split. The MASE values were subsequently ranked to identify the most impactful exogenous variables and their optimal lags from the tested range.

- For the `Selected Exogenous Variables` scenario, the top three unique exogenous variables, when used with lag 0, that resulted in the best MASE scores were chosen.

- For the `Selected Exogenous Variables with Lags` scenario, the top three unique ranked exogenous variables were selected, each paired with its respective optimal lag from the tested range that yielded the best MASE score for that particular exogenous variable.

Selecting a fixed number of three covariates was a methodological choice aimed at creating a straightforward and standardized process, with the acknowledgement that a more dynamic approach could be a potential area for future improvement. It is acknowledged that various sophisticated approaches exist for exogenous variable selection, including wrapper methods specific to each forecasting model, filter methods based on statistical properties and embedded methods where selection is part of the model training (further described in Section 2.5.4. Ultimately, no single approach is universally optimal across all forecasting models. However, given the significant computational effort required to perform model-specific exogenous variable selection for each of the numerous forecasting models evaluated in this thesis and with the aim of maintaining a comparable basis for selection across models, this LightGBM-based ranking approach was chosen. It provides a straightforward, relatively fast and unified method applicable across all models in the study for identifying potentially useful exogenous variables.

4.4. Forecasting Experiment Execution

This section describes the overall strategy for executing the forecasting experiments, including data splitting, forecast horizon definitions and the general procedure for generating forecasts across models and exogenous variable scenarios.

4.4.1. Data Splitting and Validation Strategy

For each of the 117 selected time series (as determined in Section 4.1.4), a systematic approach was used to define training data and forecast evaluation periods, employing a rolling origin validation strategy introduced in Section 2.4.2:

- **Forecast Horizons:** The length of the forecast horizon (H) was defined based on the granularity of the time series to represent both short-term and potentially long-term forecasting objectives:
  - Hourly: 24 steps (short-term) and 168 steps (long-term)
  - Daily: 7 steps (short-term) and 90 steps (long-term)
  - Weekly: 26 steps
  - Monthly: 12 steps
  - Quarterly: 8 steps

  For hourly and daily series, experiments were run for both their short-term and long-term horizons. For weekly, monthly, and quarterly series, a single forecast horizon was applied, generally reflecting common short-to-medium term outlooks for these granularities and considering typical time series lengths.

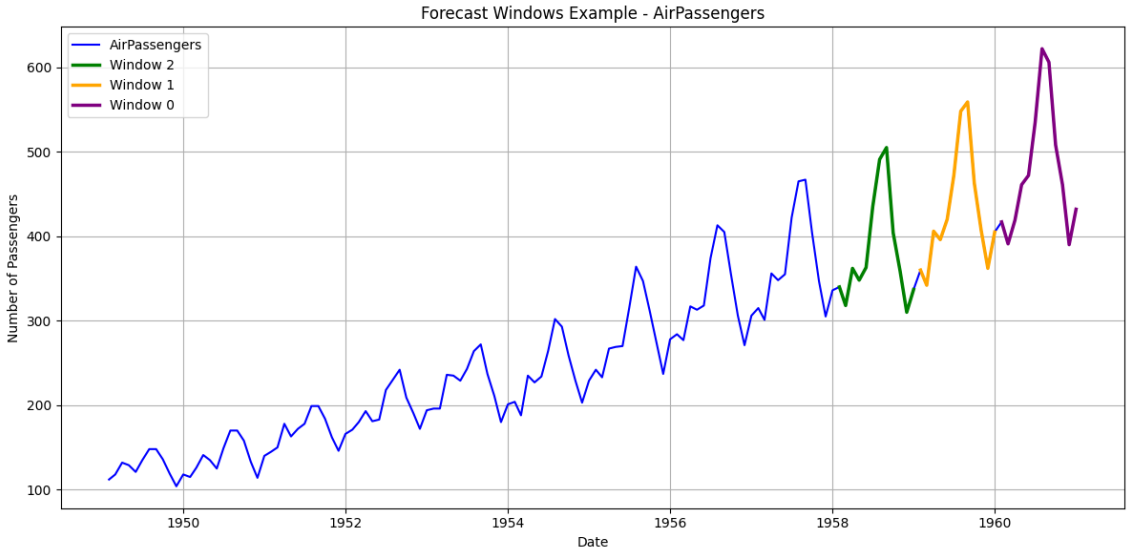


Figure 4.7.: Enter Caption

- **Multiple Forecast Windows (Rolling Origin):** To assess performance robustness across different periods of each time series, three distinct forecast windows (origins) were defined using a rolling origin mechanism. Let L be the length of a given time series and H be the forecast horizon.
  - **Window 0 (Latest):** The historical context consists of the first $L - H$ data points, and the model forecasts the final H data points.
  - **Window 1 (Shifted):** The time series is first truncated to length $L-H$. The historical context then consists of the first $(L - H) - H$ data points of this truncated series, and the model forecasts the subsequent H data points.
  - **Window 2 (Further Shifted):** The time series is first truncated to length $L - 2H$. The historical context then consists of the first $(L - 2H) - H$ data points of this further truncated series, and the model forecasts the subsequent H data points.

  This approach ensures that models are tested on different segments of the time series, provided the series is long enough to accommodate all three windows. Figure 4.7 provides a visual illustration of this data splitting strategy using the AirPassengers dataset, where three distinct, non-overlapping forecast periods of 12 month are defined by rolling the forecast origin backward in time.

4.4.2. Forecast Procedure Definition

Across all selected time series, their defined forecast windows, and applicable forecast horizons, a total of approximately 398 unique samples (time series segment and horizon combinations) were generated for the experiments. Forecasting was performed systematically for each selected model across various exogenous variable scenarios for these samples.

- **FMs:** Each of these models was evaluated across all applicable exogenous variable scenarios detailed in Section 4.3.1.
- **Traditional Exogenous Variable-Aware Comparison Models ((S)ARIMAX, LightGBM, TiDE, NBEATSx):** These models were subjected to forecasting under a comprehensive set of key exogenous variable scenarios to allow for robust comparison. This included: no exogenous variables, all available exogenous variables, and the "selected exogenous variables with lags" scenario.
- **Baseline Univariate Models (ETS, Theta, and SARIMAX):** These models, which do not integrate external exogenous variables, were forecasted once per sample, serving as a fundamental performance benchmark.

For each forecast generated, four key outputs were collected: the point forecast, the 10% and 90% prediction quantiles and the forecast execution time in seconds. To ensure full traceability, these were stored alongside comprehensive metadata detailing configuration specifics like the dataset, model, and exogenous variable mode, among others. The decision to focus on the 10% and 90% quantiles was both practical and strategic. With the exception of MOIRAI, which can return a full probability distribution, most models capable of uncertainty estimation output a fixed set of quantiles (0.1, 0.2, ..., 0.9). From this available set, the 10% and 90% quantiles were selected as they are the most critical for risk assessment and decision-making. The availability of these probabilistic results was, however, limited to this specific subset of models.

The systematic application of these scenarios allows for a detailed analysis of how different models leverage, or are affected by, various types and configurations of exogenous variable information.

4.5. Evaluation Methods

This section details the metrics used to quantify forecast accuracy, the methodology for comparing computational practicability, and the statistical tests applied to assess the significance of the experimental results.

4.5.1. Forecast Accuracy Metric Selection

To comprehensively evaluate the forecasting performance of the models across various scenarios, a selection of widely recognized accuracy metrics was employed. As defined in Section 2.4.3, these include point metrics such as:

- **MASE:** A scale-independent metric, robust to zeros and useful for comparing forecast accuracy across different series (scaled by the in-sample, one-step naive forecast error from the training data).
- **RMSSE:** Another scale-independent metric, similar to MASE but based on squared errors, providing insights into larger forecast deviations by penalizing them more heavily.

For evaluating probabilistic forecasts and prediction intervals, the following metrics are used:

- **WQL:** Crucial for evaluating the quality of predicted quantile distributions, assessing both calibration and sharpness.
- **MSIS:** Measures the quality of prediction intervals, penalizing intervals that are too wide or too narrow, and those that fail to contain the true value, while also being scale-independent.

Additionally, the following metrics will be utilized:

- **PIS:** A metric particularly relevant for evaluating forecasts of intermittent or sporadic time series, assessing inventory implications by quantifying stockout periods or overstock situations based on forecast errors.
- **CFE**: Provides insight into the bias of the forecasts by summing up errors over time, indicating systematic over or under-forecasting.

4.5.2. Methodology for Practicability Comparison

Beyond predictive accuracy, a model's practicability was evaluated along two key dimensions: computational speed and operational robustness.

- **Total Forecast Time:** The primary metric for speed was a single wall-clock time measurement for each forecast. This time captures the entire end-to-end process, from data loading and any necessary model training or fine-tuning to the final inference step. This unified metric allows for a direct comparison of the total computational cost across all model types.
- **Operational Robustness:** This was assessed by systematically analyzing the structured error logs generated by the experimental framework. Specific failure modes, such as out-of-memory errors, processing timeouts and other critical exceptions, were logged and traced to provide insights into each model's stability and resource requirements.

Together, these metrics offer a comprehensive view of each model's real-world practicability.

4.5.3. Statistical Tests for Significance Testing of Results

In any empirical evaluation of forecasting methodologies, it is essential not only to report raw performance metrics but also to rigorously assess whether observed differences among competing models are unlikely to have arisen by chance. In this context, the choice of statistical tests and complementary graphical tools serves two interconnected purposes:

- to formally test hypotheses about relative performance across multiple datasets and
- to provide transparent visualizations that elucidate the magnitude and uncertainty of model-specific effects.

The following methods and plot types will be employed to yield a comprehensive understanding of the results.

Friedman Test with Nemenyi Post-Hoc Test When comparing more than two forecasting algorithms across a collection of heterogeneous time series, the assumptions underlying classical parametric tests are often violated. Particularly since error distributions can vary widely from one series to the next. The Friedman test is a nonparametric alternative to repeated-measures ANOVA that operates on the ranks of model errors rather than the raw error values themselves [M. Friedman 1937]. Concretely, for each individual time series, each model's performance metric is converted into a rank. With rank 1 assigned to the lowest error, rank 2 to the second lowest, and so on. The Friedman statistic then aggregates these ranks across all series to test the null hypothesis that all models have equivalent median ranks. If the Friedman test rejects this null hypothesis at a chosen significance level of $\alpha = 0.05$, one concludes that at least one pair of models differs in performance in a statistically meaningful way. However, the Friedman test itself does not indicate which specific pairs of models are significantly different. To localize these differences, the Nemenyi post-hoc test is typically applied, which compares the average ranks of each pair of models and calculates a `critical difference` (CD) threshold [Nemenyi 1963]. Excellent idea. Using a heatmap is a very clear and effective way to visualize the Nemenyi test results. Here is the updated final paragraph. If the absolute difference in average ranks exceeds the CD, then the corresponding pair is judged to differ significantly. To visualize these pairwise comparisons, this study utilizes a heatmap. This heatmap displays the absolute difference in average ranks for every pair of models. The cells are color-coded to indicate statistical significance: a cell is colored pure blue if the rank difference between the two models exceeds the critical difference (CD) threshold, indicating a significant performance difference. Otherwise, the cell remains white. This method provides a clear and immediate visual matrix of all pairwise comparisons, allowing for quick identification of which models significantly outperform others.

Bar Plots with Confidence Intervals Even when statistical tests indicate significant differences, it is pedagogically useful to show the actual magnitudes of performance metrics and their uncertainty in a straightforward bar plot. In practice, the mean or median of a chosen error metric will be computed for each model across multiple replications or multiple data subsets. To quantify the precision of these aggregate estimates, nonparametric bootstrapping will be employed: for example, resampling the set of time series (with replacement) many times (e.g., 1,000 bootstrap samples) to generate an empirical distribution of the mean MASE for each model. The endpoints of the 95% bootstrap percentile interval then define the lower and upper bounds of the error bar. By plotting each model's point estimate together with its corresponding 95% confidence interval, one can visually assess overlap among intervals. Non-overlapping intervals suggest but do not formally prove—statistical distinction, while wide intervals flag greater uncertainty. This combination of point estimates and interval estimates provides a clearer picture than raw numbers alone, by making explicit the variability induced both by the finite sample of series and by stochastic elements.

Scatter Plots of Timing Versus Series Characteristics To analyze the practical performance of the models, scatter plots are used to visualize the relationship between forecast time and key data characteristics. In these plots, each forecast

run is represented as a point, with the y-axis showing the total forecast time and the x-axis representing a data attribute of interest, such as time series length or dimension (the number of endogenous and exogenous variables). Additionally the timeseries length is encoded using color of the points. By examining the resulting distribution of points, one can identify systematic trends in a model's computational performance.

Box Plots of Percentage Improvement A particularly intuitive way to quantify the added value of a modeling choice, for example inclusion of exogenous variables, is to compute, for each series, the relative percent change in error when moving from a baseline to an enhanced model. Formally, if $e_{\text{baseline},i}$ denotes the error metric on series i without exogenous variables and $e_{\text{exo},i}$ is the error with exogenous variables, then the percent improvement is:

$$\Delta_i = \frac{e_{\text{baseline},i} - e_{\text{exo},i}}{e_{\text{baseline},i}} \times 100\% \tag{4.1}$$

Aggregating $\{\Delta_i\}$ across all series yields a distribution of percentage improvements. A box plot succinctly displays the median , interquartile range (25th to 75th percentile), and potential outliers. If the box plot is centered above zero and the lower whisker remains above zero, one can confidently assert that the exogenous variable-augmented model offers a consistently positive gain across nearly all series. Conversely, a wide spread or presence of negative outliers suggests that the improvement is inconsistent or that exogenous variable inclusion may degrade performance for certain data regimes. By comparing multiple box plots side-by-side, readers can readily gauge not only which modeling choice yields higher average gains, but also which exhibits less variability or risk of underperformance.

Concluding Remarks on Statistical Evaluation Taken together, the Friedman/Nemenyi testing framework and the categories of visualization constitute a comprehensive toolbox for both inferential and descriptive analysis of the experimental results. The nonparametric rank-based testing safeguards against unjustified parametric assumptions, while the graphical summaries promote transparency regarding effect sizes, uncertainty, and potential trade-offs. Incorporating these methods ensures that practitioners and researchers alike can make well-grounded judgments about whether observed performance differentials reflect genuine modeling advances or merely sampling variability. This level of statistical rigor is particularly noteworthy, as it is often underrepresented in forecasting benchmarks, especially within the DL literature [Brigato et al. 2025]. These statistical evaluations and visualizations will be crucial for drawing valid conclusions and addressing the research questions posed in this thesis.

5. Results

This chapter presents the empirical findings from the comprehensive experimental evaluation detailed in Chapter 4. The primary objective is to assess the forecasting performance of selected FMs and compare them against established statistical, ML and DL methods, particularly focusing on the impact of exogenous variable integration. The chapter will first delve into an in-depth analysis for each of the investigated FMs (Section 5.1), examining their individual behavior across different exogenous variable configurations, forecast horizons, and practicability aspects. Subsequent sections will then provide an overall comparison of all models under various exogenous variable scenarios (Section 5.2), analyses of short-term versus long-term forecasting accuracy (Section 5.2.1), cross-dataset performance, the overall impact of exogenous variables (Section 5.2.2), and the forecasting speed of the different models (Section 5.2.4), utilizing MASE as the primary performance metric.

5.1. In-depth Analysis of Foundation Models

This section begins the empirical results by providing a more detailed analysis for each of the investigated FMs. The aim is to understand their individual performance characteristics, their response to different exogenous variable configurations, variations across forecast horizons, improvements on a per-dataset basis, and specific timing and practicability aspects.

5.1.1. Chronos

This section delves into a detailed analysis of the Chronos FM (as described in Section 3.3.2), examining its performance with different exogenous variable modes, across varying forecast horizons, its improvement by dataset, and its timing characteristics.

Example Forecast Plots for Chronos

To offer a clear introduction to our results, this section presents illustrative time series plots with Chronos's forecasts. These visualizations provide an intuitive understanding of the model's performance, serving as an accessible entry point before delving into the comprehensive quantitative analysis of all models.

The following four time series plots (Figures 5.1 to 5.4) are all from the "Bike Sharing" dataset (described in Section 4.1.3 of Chapter 4), showing the "Rented Bike Count" on the Y-axis and time steps on the X-axis. The data is sampled hourly and exhibits complex multiple seasonalities over a 24-hour period. Typically, few bikes are rented at night. A strong peak occurs in the morning, followed by a period of relatively stable rentals (around 700 bikes) and a second peak later in the day, likely corresponding to the end of work or school hours. The dataset also

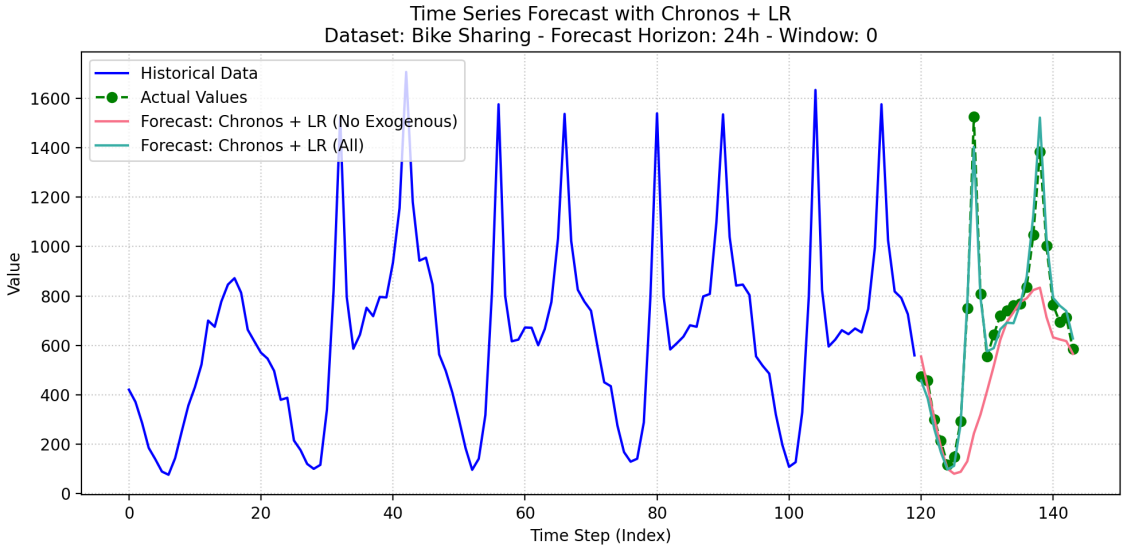


Figure 5.1.: Chronos forecast for Bike Sharing (Window 0, Horizon 24h) with and without exogenous variables.

displays weekly seasonality, with a decrease in rented bikes on some weekends. All plots show the actual values for the forecast horizon alongside Chronos forecasts, both with and without exogenous variables. The specific "windows" for context and the forecast horizon, align with the experimental design in Section 4.4.1 and Section 4.4.2. For visualization, the plotted history is consistently five times the forecast horizon. It's important to note that the model was provided with a significantly larger context length with over 8,000 data points for its predictions.

The first time series plot, Figure 5.1, has a forecast horizon of one day (24 hours). It is clear that the forecast without exogenous variables underperforms. It fails to capture the two daily peaks and seems to reproduce a pattern from approximately five days prior, which lacked these prominent peaks. With the inclusion of exogenous variables, the model captures the peaks almost perfectly, with the forecast closely following the actual values. In this instance, exogenous variable inclusion brings a significant improvement.

Figure 5.2 shows a forecast for the same horizon (24 hours) but for a context window shifted two days earlier. Window 2, compared to Window 0 in the previous plot. Thus, the context window is truncated by 48 hours, and the forecast horizon is also shifted, as per the rolling origin validation in Chapter 4. In this case, the inclusion of exogenous variables offers only minimal improvement. The model follows the seasonal peaks reasonably well even without exogenous variables.

Again, using the same time series and dataset, Figure 5.3 displays a forecast for Window 2 but now with a forecast horizon of one week (168 hours). This time, the forecast without exogenous variables performs significantly better, capturing the jagged structure and individual peaks accurately. In contrast, the forecast with exogenous variables is close to zero and only minimally follows the seasonal peaks. It can be observed that the historical time series contains some periods where the number of rented bikes is constantly zero. These periods might represent outliers that do not follow normal behavior, possibly due to errors or failures in the tracking system, which would not reflect actual demand and could introduce misinformation.

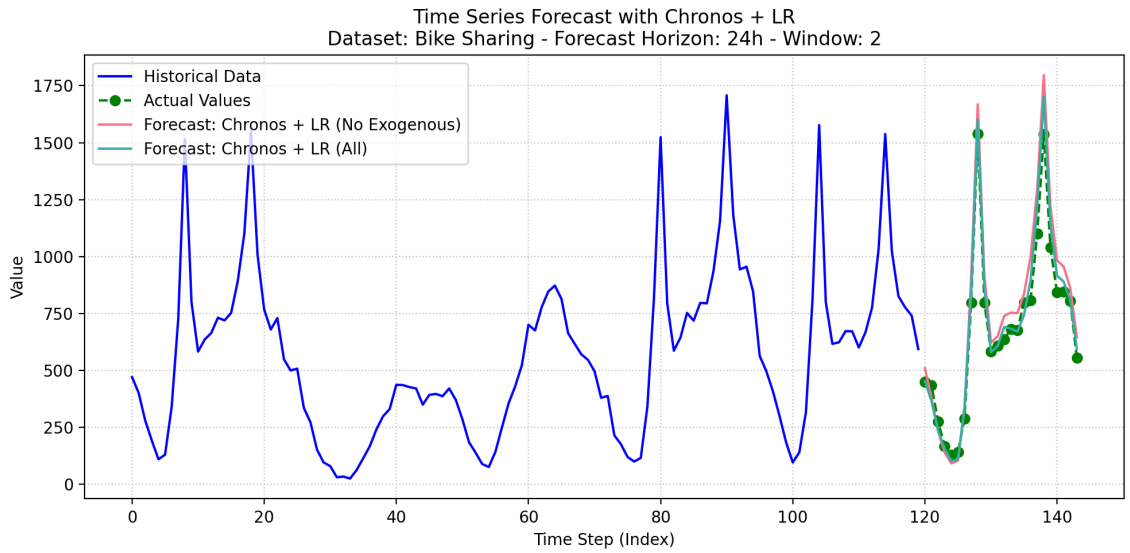


Figure 5.2.: Chronos forecast for Bike Sharing (Window 2, Horizon 24h) with and without exogenous variables.

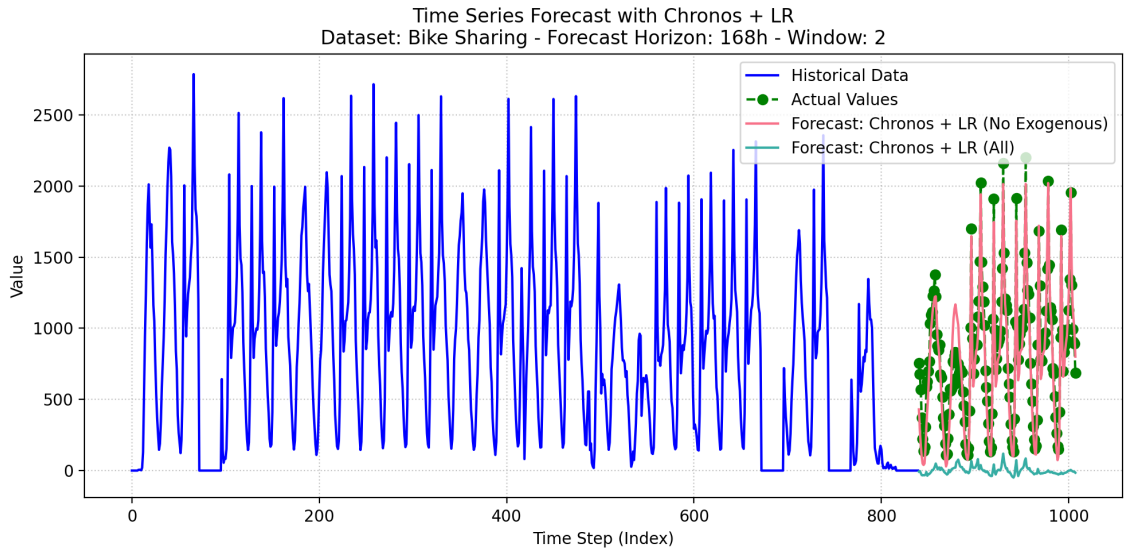


Figure 5.3.: Chronos forecast for Bike Sharing (Window 2, Horizon 168h) with and without exogenous variables.

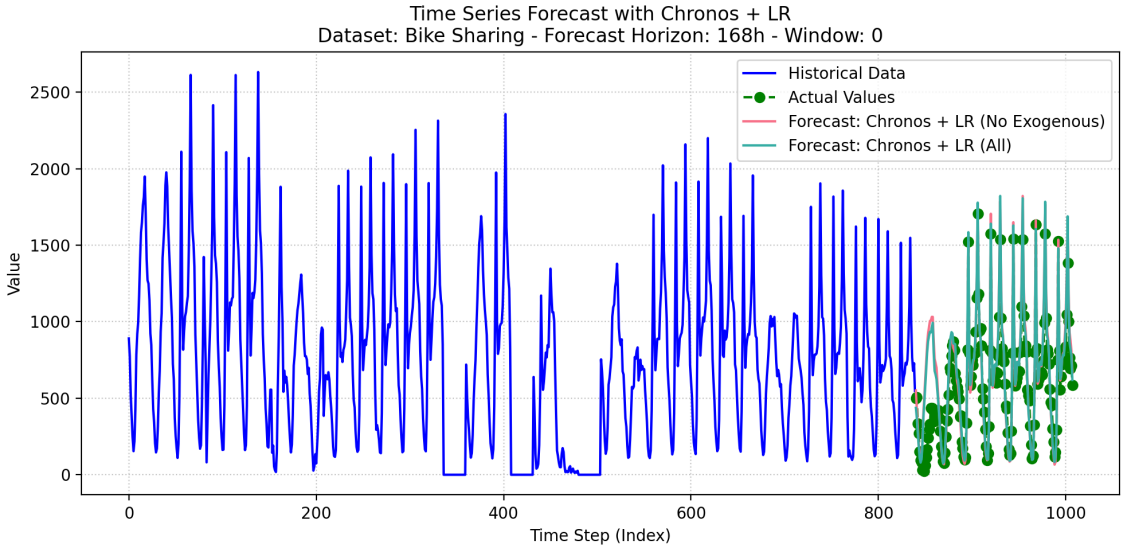


Figure 5.4.: Chronos forecast for Bike Sharing (Window 0, Horizon 168h) with and without exogenous variables.

Alternatively, they could represent maintenance periods where no bikes are actually rented. These zero-value phases appear to correlate with certain exogenous variables that have similar values in the forecast horizon, causing the external regressor used by Chronos to drive the forecast value towards zero.

Figure 5.4 shows the same time series again with a weekly forecast horizon but this time for a different window (Window 0). In this case, the forecasts with and without exogenous variables are once again very similar and represent the actual progression of the series effectively.

These time series plots illustrate a crucial point: forecasting results are not solely dependent on the intrinsic characteristics of the time series and exogenous variables. They are profoundly influenced by the chosen historical context window and the selected forecast horizon. This observation holds true across the diverse range of models evaluated in this study, underscoring the critical role of experimental setup inTSF.

Comparison of Regressor Models for Chronos

As mentioned in Section 4.2 and as will be detailed in the overall model comparisons later in this chapter (Section 5.2), a Linear Regressor and a LightGBM model were tested for the external regressor component of Chronos. A comprehensive comparison of these two regressor options across all datasets and relevant scenarios revealed no statistically significant difference in overall forecast accuracy. The time taken for forecasting with the different regressors also differed only marginally, with the LightGBM regressor being slightly slower on average than Linear Regression, as will be further discussed in Section 5.2.4. Given the comparable accuracy of the linear regressor, the detailed plots in this specific Chronos analysis section focus on results obtained with the linear regressor.

Comparison of Different Exogenous Variable Modes for Chronos

This section examines the performance of Chronos when paired with its linear external regressor (LR) under various exogenous variable configurations. The definitions

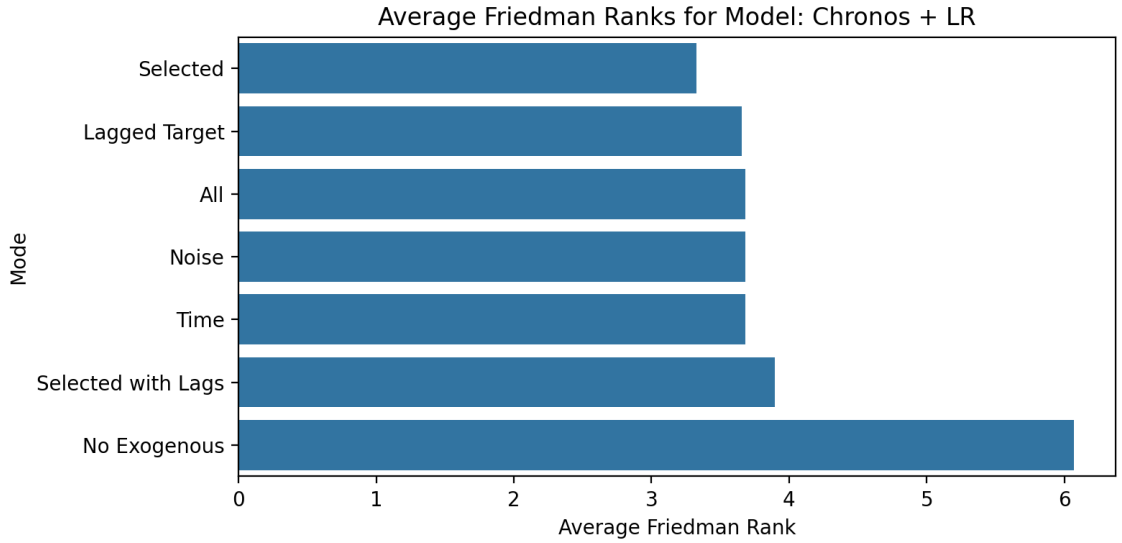


Figure 5.5.: Mean Friedman ranks for Chronos + LR using linear regressor with different exogenous variable modes.

for these exogenous variable modes are detailed in Section 4.3.1.

The Friedman rank plot in Figure 5.5 shows the performance of Chronos across these different exogenous variable modes. It is immediately apparent that including exogenous variables, regardless of the specific mode, generally leads to an improvement in rank compared to the "No Exogenous Variables" mode.

The Nemenyi post-hoc test heatmap (Figure 5.6) confirms this observation. Across all tested exogenous variable modes, the p-value for the comparison against the "No Exogenous Variables" mode is well below the significance threshold of 0.05, indicating a statistically significant improvement when using exogenous variables. A slight improvement can also be observed with certain exogenous variable selections over others. For instance, the "Selected Exogenous Variables with Lags" mode (referring to the methodology in Section 4.3.3) performs worse in ranks compared to other selection strategies like "Selected Exogenous Variables" (unlagged). However, these differences between specific exogenous variable modes excluding "No Exogenous Variables" do not fall below the significance threshold in pairwise comparisons.

Short-term vs. Long-term Forecasting Performance for Chronos

This section examines how Chronos's forecasting accuracy varies between short-term and long-term horizons across different exogenous variable modes. It's important to note that while the overall comparison in Section 5.1.1 encompassed all datasets and granularities, the short-term versus long-term analysis presented here focuses solely on daily and hourly datasets, which still constitute the majority (16 out of 20) of the evaluated datasets and (107 out of 117) target time series.

The bar plot in Figure 5.7 clearly shows that short-term accuracy is better than long-term accuracy for Chronos across all displayed exogenous variable modes. This behavior is typical for many forecasting models. It is particularly emphasized here because the MASE metric, used for this evaluation (defined in Section 4.5.1 of Chapter 4 and detailed in Section 2.4.3), scales the forecast error by the error of a naive forecast. For this study, an autoregressive naive forecast was chosen. Such

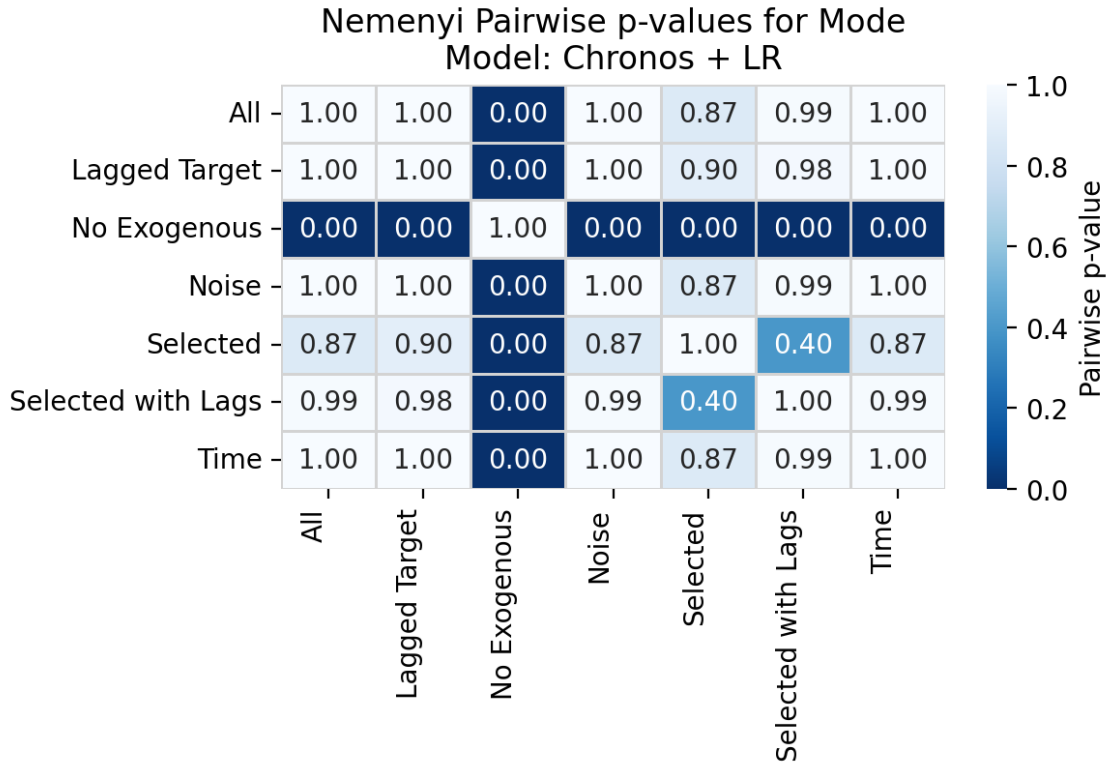


Figure 5.6.: Nemenyi post-hoc test p-value heatmap for Chronos + LR with different exogenous variable modes.

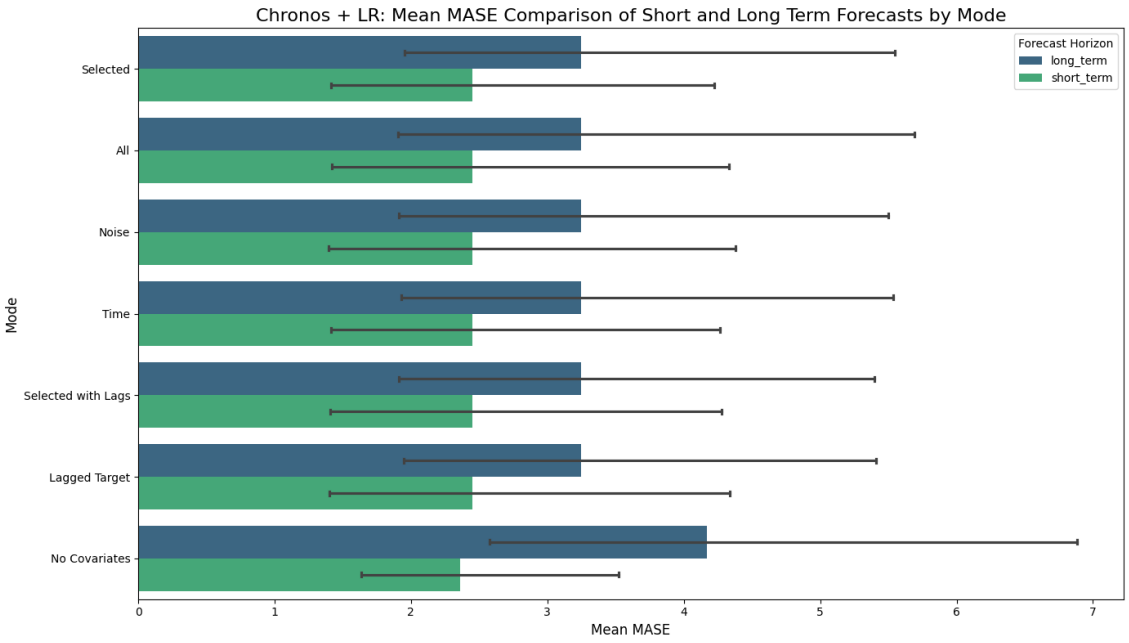


Figure 5.7.: Comparison of mean MASE for Chronos + LR: short-term vs. long-term forecasts across various exogenous variable modes.

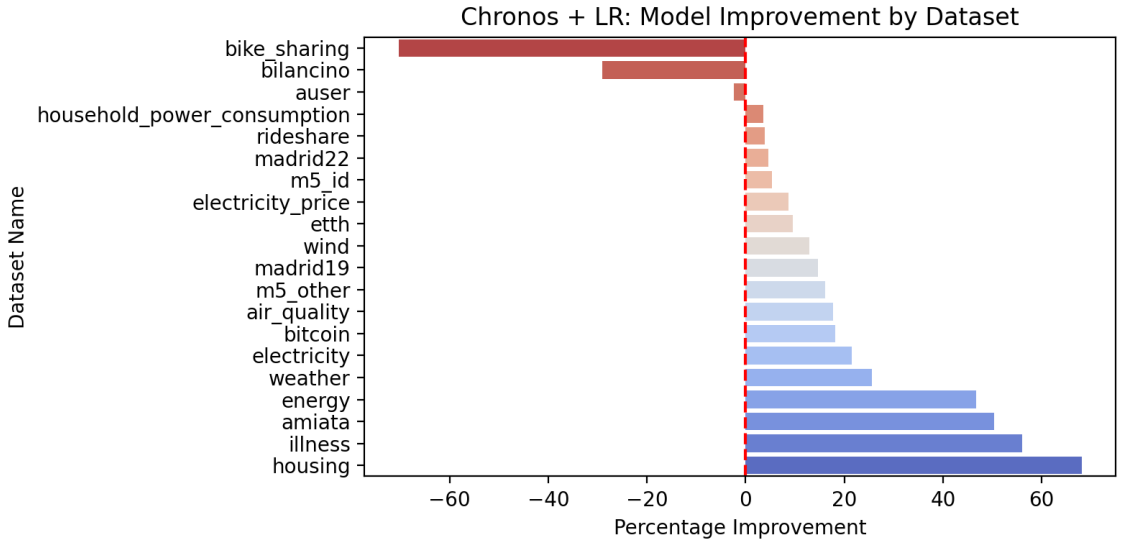


Figure 5.8.: Percentage MASE improvement for Chronos + LR when using the "selected exogenous variables" versus the "no exogenous variables" by dataset.

a naive forecast tends to perform relatively better for long horizons and degrades significantly over shorter horizons. This relative strength of the naive baseline at longer horizons consequently inflates the MASE values for the assessed models, making their long-term accuracy appear less favorable in comparison.

The plot also shows that the confidence interval for short-term forecasts is considerably smaller than for long-term forecasts. This indicates less variance and fewer extreme outliers in shorter horizon predictions with Chronos. This is explicable, as incorrect model assumptions or accumulating errors tend to have a more pronounced negative impact as the forecast length increases.

Building on this, the Figure 5.7 also reveals a nuanced insight regarding the impact of exogenous variables. Whereas in the previous Section 5.1.1 showed that incorporating exogenous variables consistently improves forecasting accuracy, this does not universally hold true for short-term predictions. Specifically, the mean MASE values for short-term forecasts across all exogenous variable modes are notably close, with the forecast without exogenous variables even exhibiting a marginally lower MASE. This suggests that the significant benefits of including exogenous variables are predominantly realized in long-term forecasting, where their additional information content genuinely contributes to superior predictive performance.

Improvement with Exogenous Variables by Dataset for Chronos

This section examines the percentage improvement in MASE for Chronos + LR when including exogenous variables, compared to its performance without exogenous variables, across all datasets. The specific exogenous variable configuration used here is the "Selected Exogenous Variables" scenario defined in Section 4.3.3.

Figure 5.8 displays the percentage MASE improvement for Chronos when incorporating the "selected exogenous variables" across various datasets. While most datasets show an improvement, it's crucial to note that three datasets exhibit a decrease in accuracy. Specifically, the "Bike Sharing" dataset sees a substantial drop

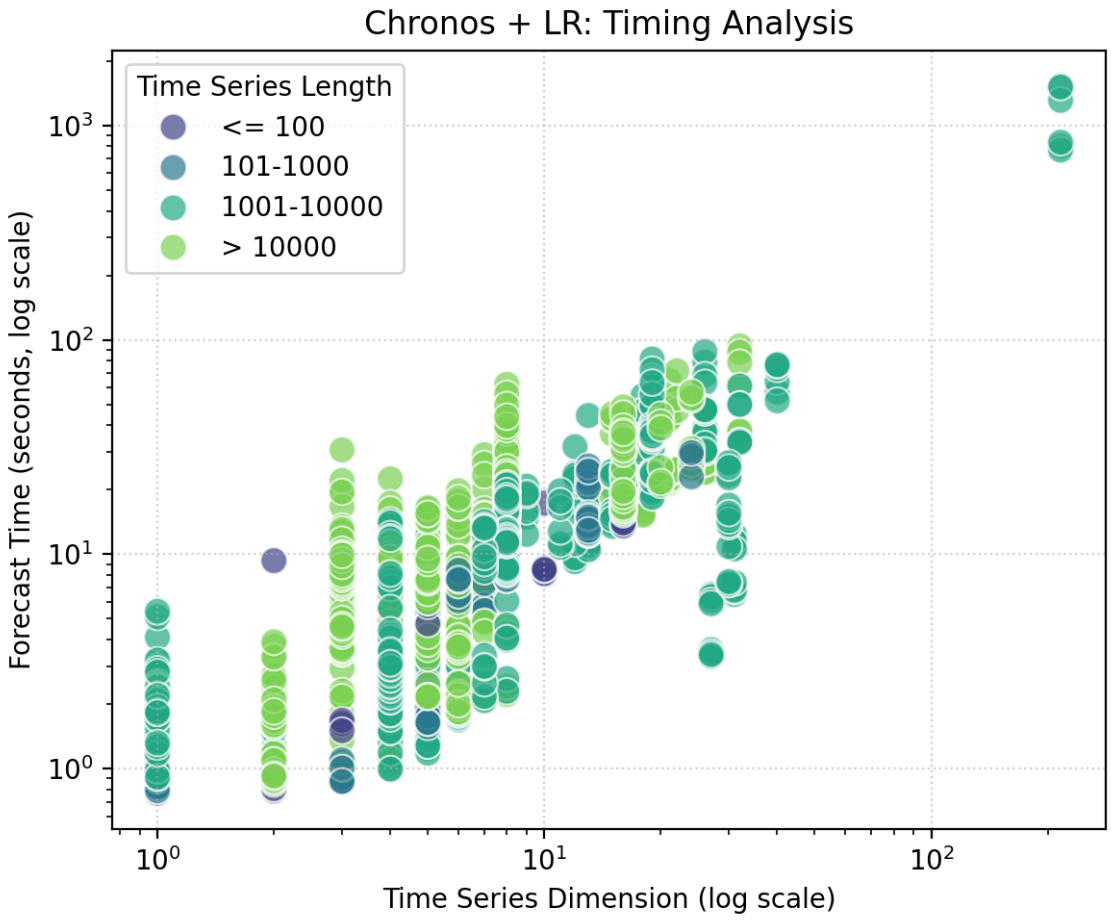


Figure 5.9.: Chronos (using its linear external regressor) forecasting time versus time series length and dimensionality (number of exogenous variables/features).

of over 60%, and "Bilancio" experiences a significant nearly 30% decrease. The "AUSER" dataset also shows a slight decline.

Conversely, the integration of exogenous variables leads to remarkable gains in other instances. For example, datasets such as "Energy," "Amiata," "Illness," and "Housing" all demonstrate improvements exceeding 40%.

Therefore, while the benefits are not universal, for Chronos, the integration of an external regressor with this selected exogenous variable configuration generally yields notable performance improvements across a wide range of datasets. These specific instances of degradation highlight the importance of careful exogenous variable selection and validation, though overall, their integration remains a generally recommendable approach for this model under such a setup.

Timing and Practicability Analysis for Chronos

The timing plot in Figure 5.9 demonstrates typical behavior for Chronos, which involves training an external regressor for exogenous variable integration at each forecasting instance.

Due to the training process of its external regressor, there is an almost linear increase in Chronos's forecasting time with the dimensionality (number of endogenous

and exogenous variables/features) of the dataset. This relationship appears largely independent of the overall historical time series length itself. This latter point is likely because the core Chronos FM operates on a fixed context length, a common characteristic of most FMs as discussed in Section 2.6. Thus, the computational load on the main FM component remains relatively constant regardless of the total length of the input series, while the regressor's training time scales with the number of features and the length of the specific context window fed to it.

Further aspects of practicability are also noteworthy. While the core Chronos model (specifically Chronos-Bolt variants used here) leverages pre-trained weights for zero-shot inference, it's important to recognize that the external regressor component still requires training for each specific forecasting task. This regressor training constitutes the primary fitting overhead per forecast instance, making it not truly zero-shot in this setup. Beyond this, our experiments on the 20 selected datasets revealed no systematic failures or any missing forecasts. As Chronos-Bolt variants are optimized for CPU environments no forecast failures due to RAM limitations were observed. A direct comparison of CPU versus GPU execution was not part of this study. Its performance on very short series would depend on the feasibility of training a meaningful external regressor on limited historical data.

5.1.2. TimesFM

This section provides an in-depth analysis of the TimesFM FM (described in Section 3.3.4), focusing on its performance with various exogenous variable integration strategies, its behavior across different forecast horizons, observed improvements by dataset, and its timing and practicability aspects.

Comparison of Different Exogenous Variable Modes for TimesFM

The performance of TimesFM under different exogenous variable configurations is examined in this section. The definitions for these exogenous variable modes are provided in Section 4.3.1.

The Friedman rank plot for TimesFM (Figure 5.10) reveals interesting behavior across different exogenous variable modes. The best results, on average, are achieved either without exogenous variables or with a selection of unlagged exogenous variables. Both temporally generated exogenous variables and lagged target values used as exogenous variables tend to perform worse. This could be because both types of features are already effectively captured or incorporated internally by TimesFM's architecture (as discussed in Section 3.3.4), potentially leading to redundancy or overfitting when these are supplied externally again. The "Noise" mode refers to the "Permuted Exogenous Variables (Noise)" scenario from Chapter 4.

The Friedman rank plot for TimesFM (Figure 5.10) reveals intriguing behavior across different exogenous variable modes. On average, the most favorable results are achieved either when no exogenous variables are included, or when a selection of unlagged exogenous variables is utilized. Conversely, both temporally generated exogenous variables and lagged target values used as exogenous variables generally tend to yield worse performance. This degradation could potentially be attributed to these feature types already being effectively captured or intrinsically incorporated within TimesFM's architecture (as discussed in Section 3.3.4), thereby leading to redundancy or overfitting when supplied externally.

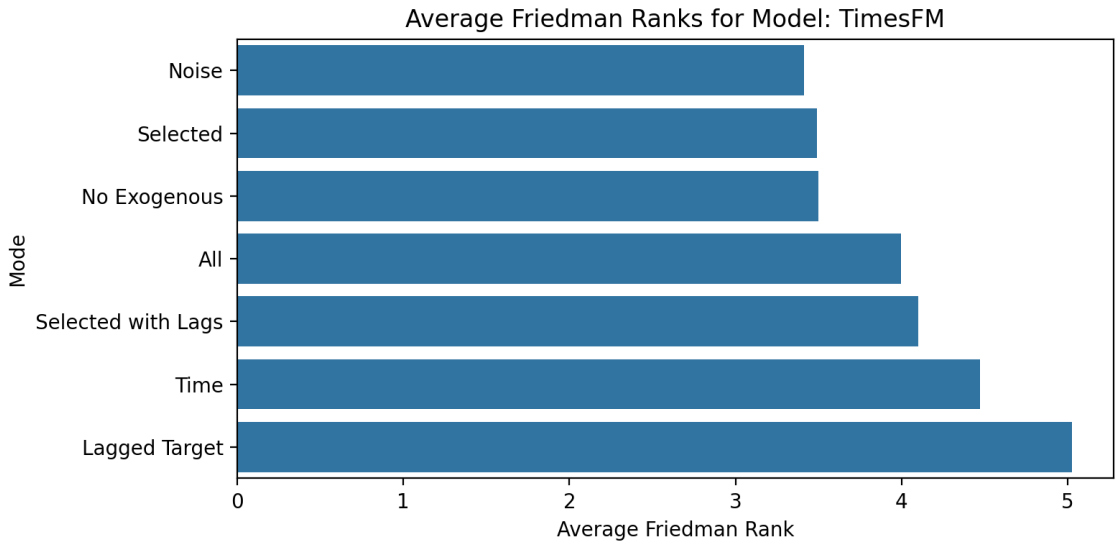


Figure 5.10.: Mean Friedman ranks for TimesFM with different exogenous variable modes.

A particularly noteworthy and unexpected observation is that the "Noise" mode, referring to the "Permuted Exogenous Variables (Noise)" scenario performs comparably well to the best-performing modes. This finding is surprising and warrants further investigation to understand the underlying mechanisms that enable TimesFM with an external linear regressor to maintain performance even when presented with noise.

The Nemenyi post-hoc test heatmap (Figure 5.11) confirms these findings. "No Exogenous Variables," "Permuted Exogenous Variables (Noise)," and "Selected Exogenous Variables" perform significantly better on average in terms of rank and are statistically comparable to each other. The "Lagged Target" as an exogenous variable mode is shown to be significantly worse than all other modes except for "Temporal Exogenous Variables," which also ranks poorly.

Short-term vs. Long-term Forecasting Performance for TimesFM

TimesFM exhibits similar behavior to Chronos regarding the impact of forecast horizon: short-term predictions generally achieve lower errors than long-term predictions across almost all exogenous variable modes presented in Figure 5.12. The "Lagged Target" exogenous variable mode appears as a slight exception, where the mean MASE for long-term performance is lower than for short-term. However, the confidence interval for the long-term "Lagged Target" scenario is wide, suggesting the presence of influential outliers or high variability, making this observation less conclusive.

Generally, for TimesFM, the confidence intervals for long-term forecasts are noticeably smaller than those observed for Chronos (compare with Figure 5.7), indicating potentially less variability or fewer extreme errors in TimesFM's performance over longer horizons compared to Chronos, which is a distinguishing characteristic.

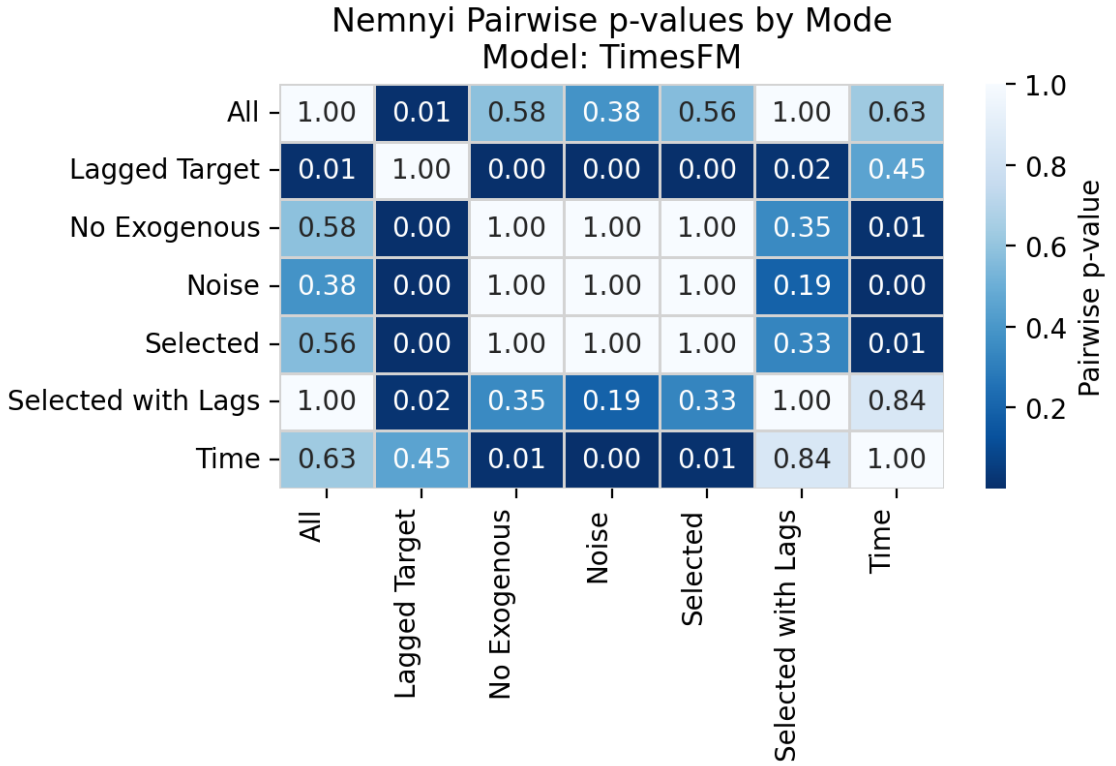


Figure 5.11.: Nemenyi post-hoc test p-value heatmap for TimesFM with different exogenous variable modes.

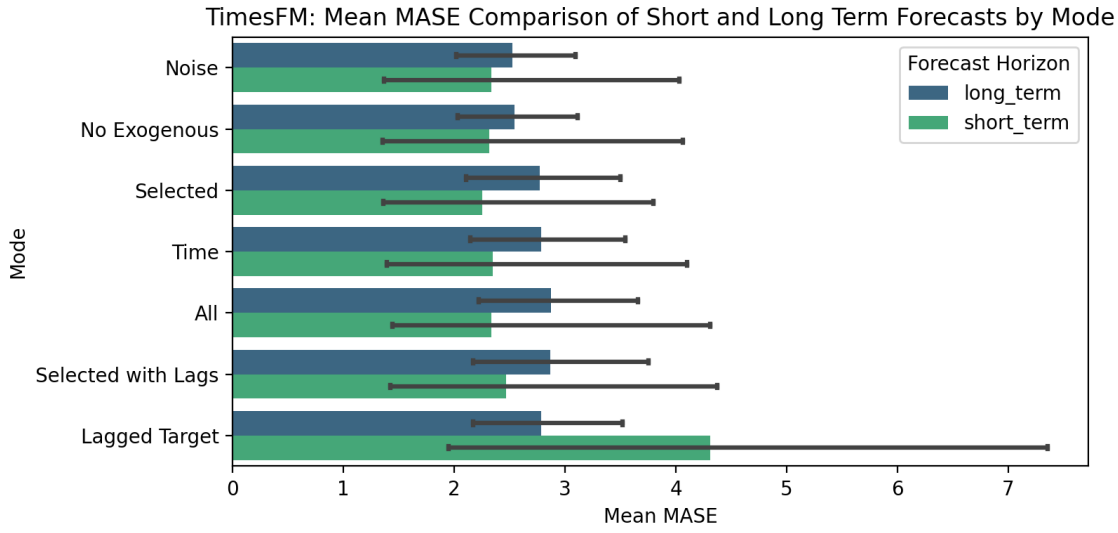


Figure 5.12.: Comparison of mean MASE for TimesFM: short-term vs. long-term forecasts across various exogenous variable modes.

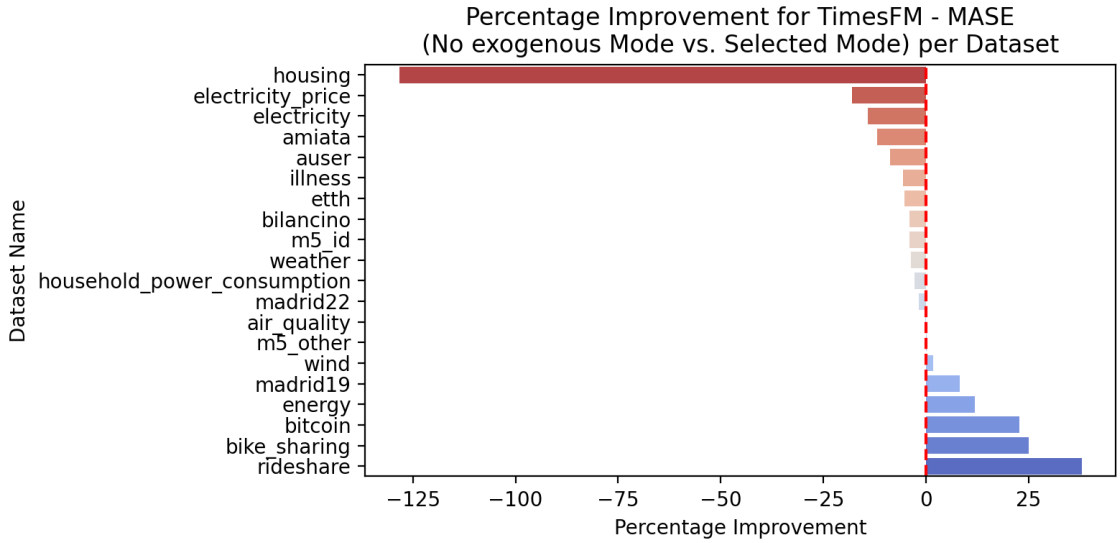


Figure 5.13.: Percentage MASE improvement for TimesFM when using the "selected exogenous variables" scenario versus the "no exogenous variables" scenario, by dataset.

Improvement with Exogenous Variables by Dataset for TimesFM

Figure 5.13 shows the percentage improvement in MASE when including the "selected exogenous variables" with TimesFM, compared to no exogenous variables, across different datasets. It is noticeable that in approximately two-thirds of the datasets displayed, the integration of these exogenous variables led to a negative improvement in performance. This deterioration is particularly pronounced for the "housing" dataset, which comprises only about 80 quarterly data points. For this dataset, a negative improvement of approximately -125% was observed, meaning the MASE increased by 125% when exogenous variables were added.

On the other hand, this does not imply that exogenous variables universally worsen TimesFM's results. For the "Rideshare" dataset, an improvement of over 40% was observed with the inclusion of these selected exogenous variables. This highlights the dataset-specific nature of exogenous variable impact for TimesFM, even when a consistent selection strategy is applied.

These results present a striking contrast to the findings for Chronos. For Chronos, the "housing" dataset exhibited one of the largest improvements with covariates, while the "bike_sharing" dataset saw a strong performance degrade. Intriguingly, with TimesFM, "Bike Sharing" shows a nearly 25% improvement, making it one of the datasets with the second-best gain. This stark difference in the impact of exogenous variables between Chronos and TimesFM demonstrates that the efficacy of incorporating an external regressor is not uniformly transferable across distinct FMs.

Timing and Practicability Analysis for TimesFM

Since TimesFM utilizes an external regressor for incorporating exogenous variables its forecasting time behavior, shown in Figure 5.14, shares similarities with Chronos.

There is an almost linear increase in forecasting time with the dimensionality of

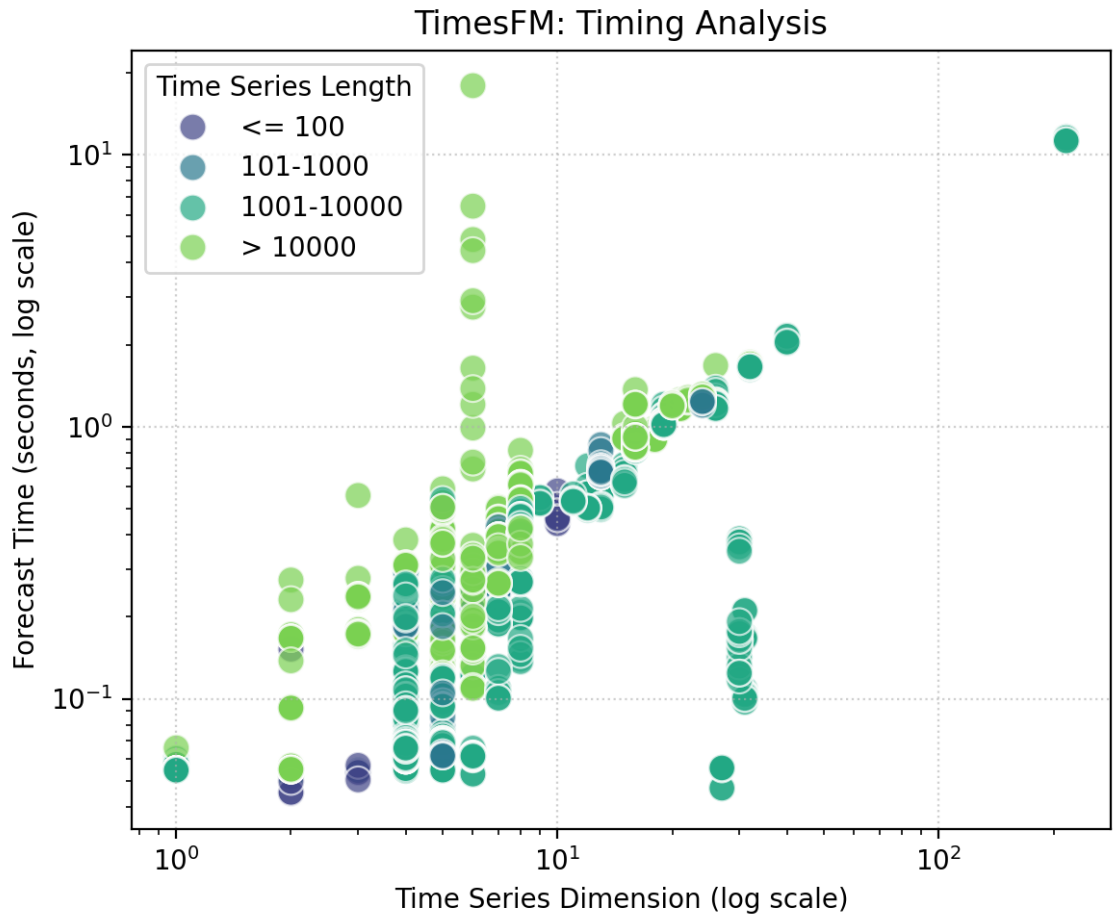


Figure 5.14.: TimesFM forecasting time versus time series length and dimensionality (number of exogenous variables/features).

the time series. The overall length of the historical time series plays a less significant role in the total forecasting time. This is mainly reasoned by the fixed input context length of the model (up to 2048 for v2.0, as detailed in Section 3.3.4). Consequently, the computational demand of the main FM component remains relatively stable, while the primary driver of increased forecasting time with more exogenous variables is the training of the external regressor on these exogenous variables.

Beyond this, our experiments did not reveal any systematic failures or missing forecasts specifically attributable to TimesFM during the main experimental runs. Only a single forecast failed because of not enough data for training the regressor. timesfm was run on GPU and the total Video Random Access Memory (VRAM) use was steady around 5gb.

Beyond this, our experiments did not reveal any systematic failures or missing forecasts specifically attributable to TimesFM during the main experimental runs. Only a single forecast attempt failed due to insufficient data for training the external regressor. The total VRAM usage remained steady at approximately 5 GB throughout the experiments.

5.1.3. TimeGPT

This section examines the TimeGPT FM (introduced in Section 3.3.3), focusing on its performance characteristics with different exogenous variable integration approaches, its behavior across varying forecast horizons, observed improvements by dataset, and its distinct timing and practicability aspects related to its API-based deployment.

TimeGPT was also evaluated with fine-tuning. However, its behavior across the various covariate setups was identical to the non-fine-tuned version. Consequently, most of the following subsections primarily discuss TimeGPT without fine-tuning and present plots only for this configuration, as the results are virtually indistinguishable.

Comparison of Different Exogenous Variable Modes for TimeGPT

For TimeGPT, forecasts without the integration of exogenous variables clearly perform best on average, as shown by the Friedman ranks in Figure 5.15. Similar to the observations for TimesFM, using "Temporal Exogenous Variables" or "Lagged Target" values as external exogenous variables results in poorer average ranks. The reason for this might again be the internal handling of such temporal information and autoregressive features within TimeGPT's architecture (see Section 3.3.3), potentially leading to redundancy or counterproductive interactions when these are supplied externally. Other exogenous variable modes show relatively similar performance to each other, though still generally worse in rank than using no exogenous variables at all.

The Nemenyi post-hoc test results (Figure 5.16) confirm the significantly better performance of TimeGPT without exogenous variables compared to almost all other modes. The exception is the "Selected Exogenous Variables" mode, where the difference from "No Exogenous Variables" is not statistically significant. The poorer performance of "Temporal Exogenous Variables" and "Lagged Target" exogenous variable modes is also statistically confirmed against the other modes.

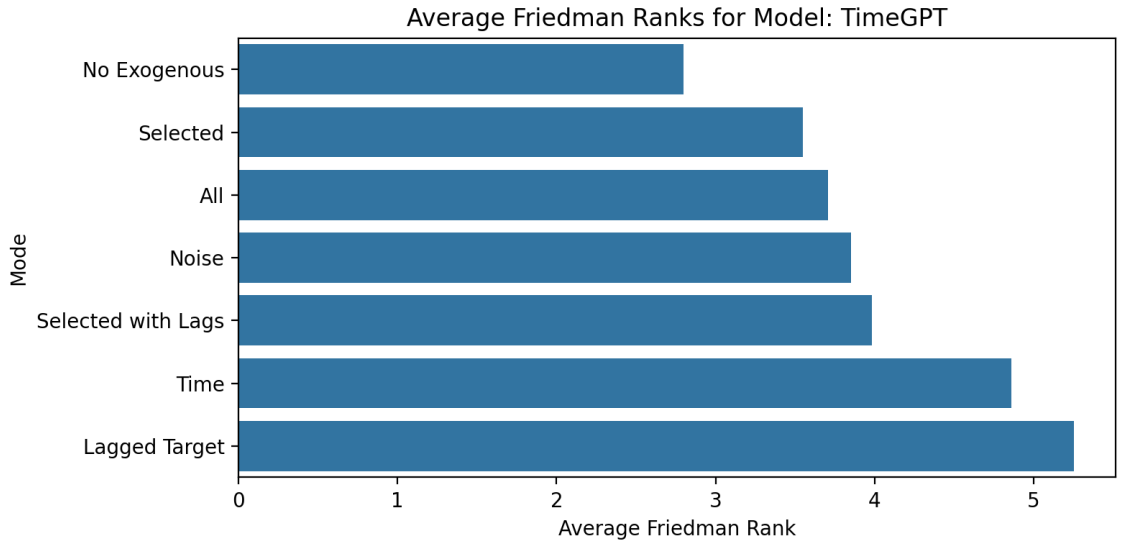


Figure 5.15.: Mean Friedman ranks for TimeGPT with different exogenous variable modes.

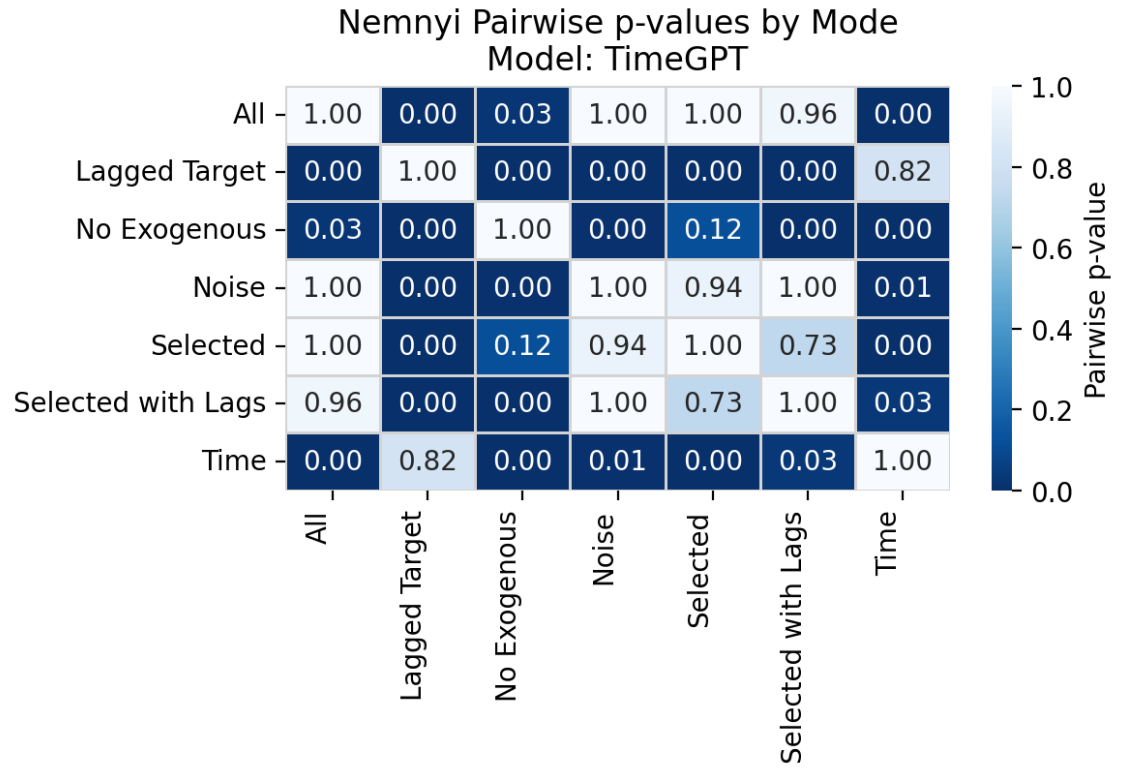


Figure 5.16.: Nemenyi post-hoc test p-value heatmap for TimeGPT with different exogenous variable modes.

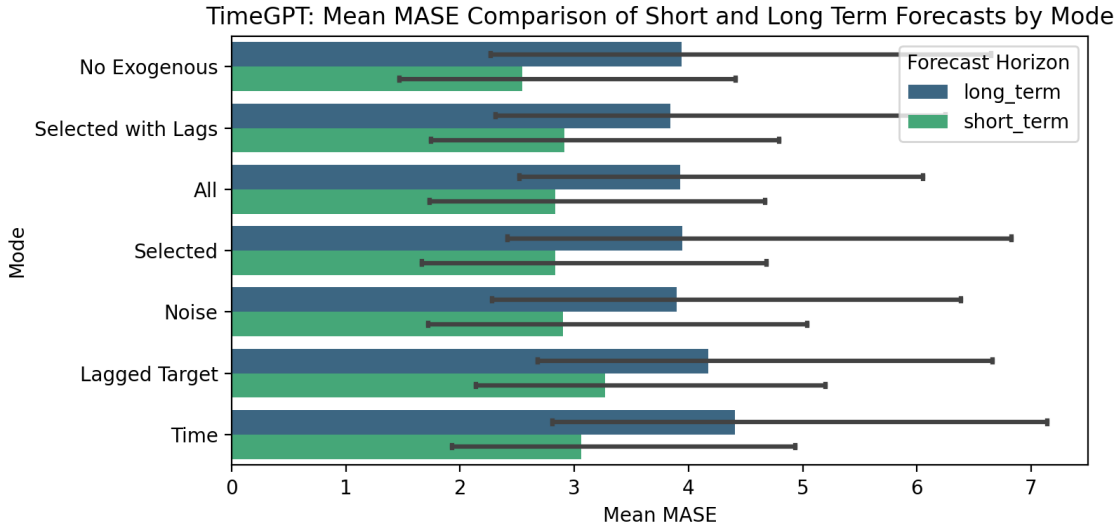


Figure 5.17.: Comparison of mean MASE for TimeGPT: short-term vs. long-term forecasts across various exogenous variable modes.

Short-term vs. Long-term Forecasting Performance for TimeGPT

TimeGPT also achieves markedly better results on shorter forecast horizons compared to longer ones, as depicted in Figure 5.17. The confidence interval for the short-term setting is also generally smaller across most exogenous variable modes, indicating more consistent performance. The performance difference between short-term and long-term forecasting is particularly pronounced when using TimeGPT without exogenous variables, where the mean MASE for long-term forecasts is substantially higher. It is important to note that for long-term forecasting, different pre-trained model weights were utilized for TimeGPT, as suggested by the Nixtla API documentation.

Improvement with Exogenous Variables by Dataset for TimeGPT

Similar to TimesFM, TimeGPT shows an improvement with the inclusion of these "selected exogenous variables" in only about one-quarter of the datasets displayed in Figure 5.18. Furthermore, the ratio of potential improvement to potential deterioration is strongly imbalanced. This suggests that the risk of performance degradation often outweighs the potential benefits of adding exogenous variables.

Figure 5.19 illustrates the percentage MASE improvement for the fine-tuned TimeGPT with selected exogenous variables, revealing notable shifts compared to the non-fine-tuned version. While datasets exhibiting the most significant improvement ("Rideshare") and largest deterioration ("Housing") remain consistent across both fine-tuned and non-fine-tuned models, a substantial number of datasets show a complete shift in their response to covariates. For instance, the "Illness" dataset, which saw an improvement without fine-tuning, now experiences an over 30% worsening in results when covariates are included with fine-tuning. Similarly, the "Ami-ata" dataset's performance drastically changes from a slight deterioration of just over 20% without fine-tuning to a nearly 100% deterioration with fine-tuning. Overall, the magnitudes of both improvements and deteriorations tend to be even more pronounced for most datasets in the fine-tuned setting, with the aforementioned

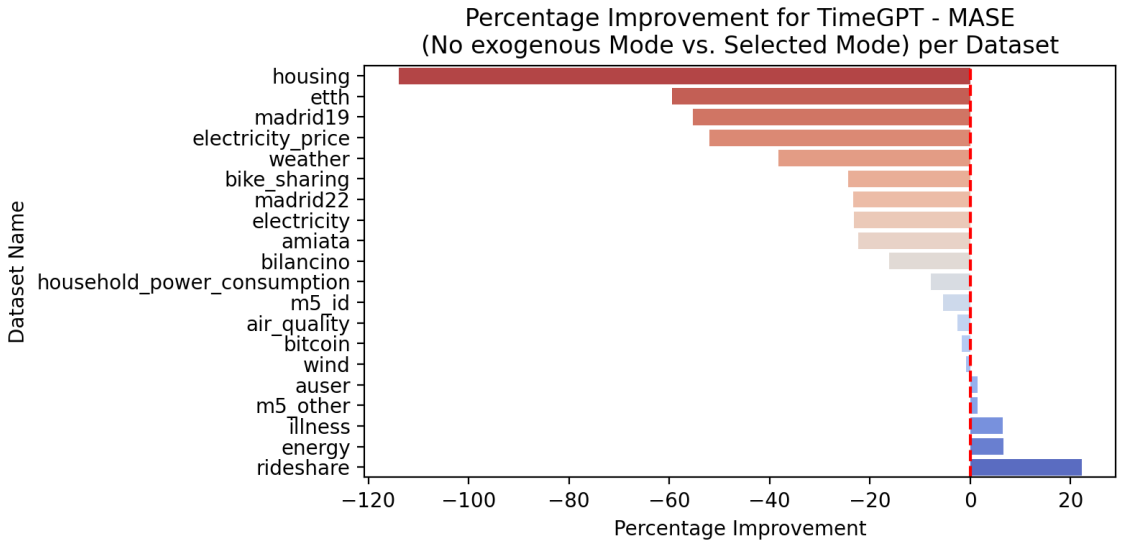


Figure 5.18.: Percentage MASE improvement for TimeGPT when using the "selected exogenous variables" scenario versus the "no exogenous variables" scenario, by dataset.

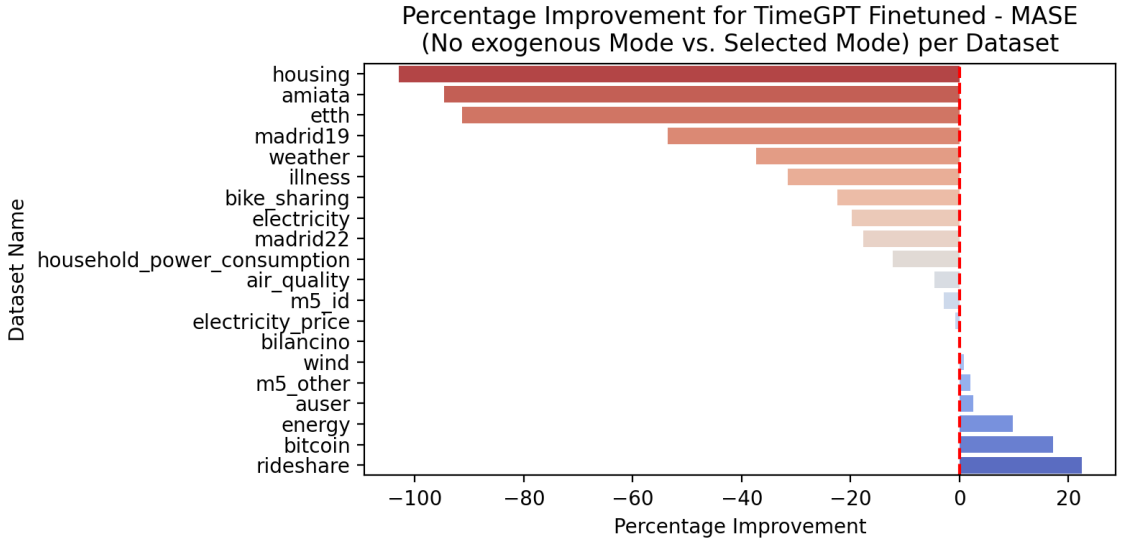


Figure 5.19.: Percentage MASE improvement for TimeGPT Finetuned when using the "selected exogenous variables" scenario versus the "no exogenous variables" scenario, by dataset.

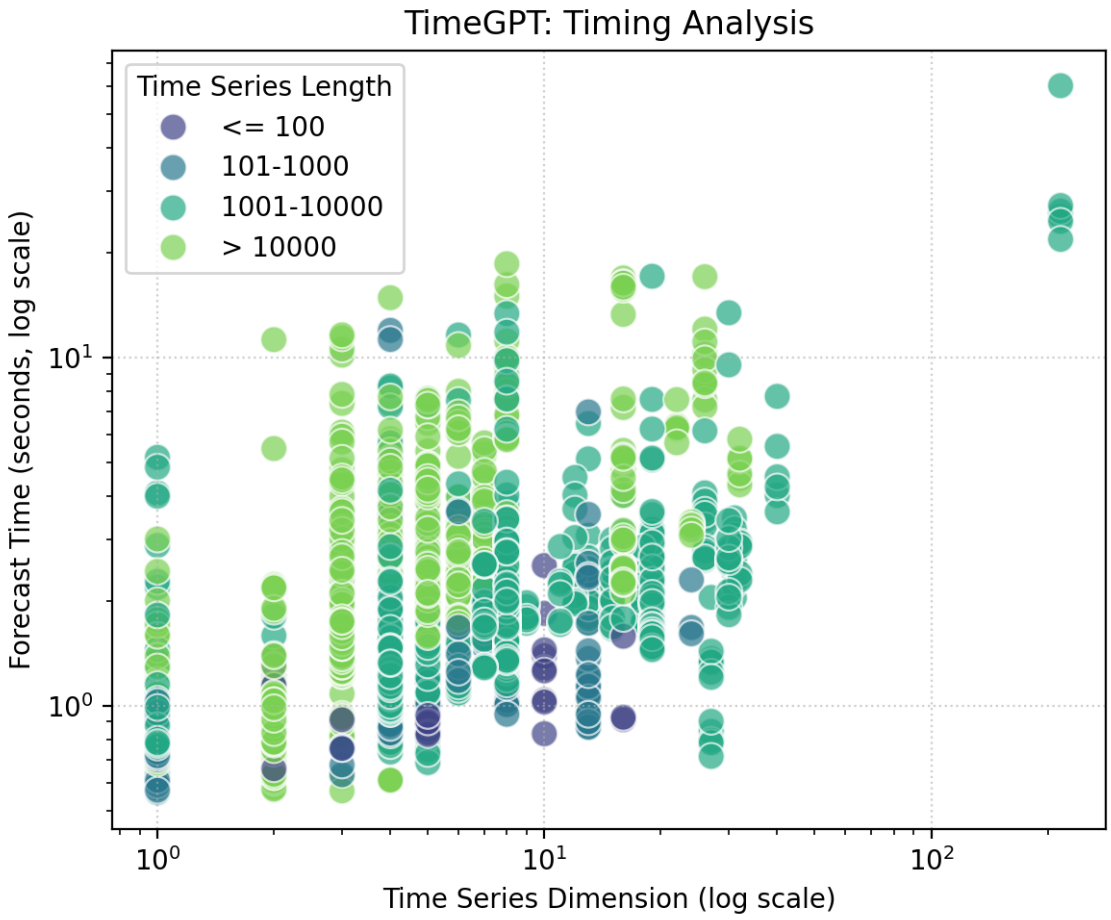


Figure 5.20.: TimeGPT forecasting time versus time series length and dimensionality.

exceptions.

This intensified variability and the stronger positive and negative impacts can be attributed to the fine-tuning process attempting to adapt the pre-trained model to better leverage or integrate the exogenous variables. When successful, this specialization leads to notable improvements. However, when the fine-tuning data or process leads the model to learn spurious correlations or overfit to the noise introduced by the covariates, it results in a significant performance degradation. Conversely, for cases where fine-tuning aids in a more effective integration of the covariates, it can lead to superior results compared to the non-fine-tuned model that might not have been sufficiently sensitive to these external features from its pre-training alone.

Timing and Practicability Analysis for TimeGPT

Due to its API-based nature, TimeGPT exhibits a nearly constant forecasting time, as illustrated in Figure 5.20. Even when fine-tuning is selected, this time increases only marginally. Based on observations during the experiments, the mean forecast/response time was approximately 1.93 seconds for non-fine-tuned runs, increasing to 3.49 seconds for fine-tuned runs, with both configurations exhibiting a maximum response time of around one minute. No significant differences in prediction time related to the dimensionality or length of the input time series were

observed through the API. This is expected, as the computational load is handled by Nixtla's infrastructure, and the time measured is predominantly the round-trip API call latency plus the fixed processing time on the remote server.

Further aspects of practicability specific to its API-based deployment include:

- **API-Specific Considerations:** The measured 'forecasting time' for TimeGPT primarily reflects API round-trip latency, data transfer overheads to and from Nixtla's servers, and server-side processing. Financial costs associated with the API usage model, while not quantified as a performance metric in this study, represent a significant practical consideration for real-world deployment and scalability.

- **Failures, Errors, and API Limitations:** No widespread API errors or timeouts distinct from general experimental setup issues were encountered for TimeGPT during the evaluation period. However, two specific time series failed during long-horizon forecasting due to insufficient historical data for the particular long-horizon model weights chosen, resulting in a `ValueError: Some series are too short. Please make sure that each serie contains at least 300 observations`.

- **Fine-tuning Process via API:** TimeGPT offers a fine-tuning capability via its API, a process which primarily involves submitting training data and awaiting job completion. The user-configurable aspects of fine-tuning are limited to a maximum number of steps and a 'fine-tuning depth' parameter, ranging from 1 to 5. While this simplified approach eliminates the need for extensive parameter search, it simultaneously presents a challenge in determining the optimal depth due to a lack of further information or direct insights into the fine-tuning progress and internal mechanisms.

5.1.4. MOIRAI

This section presents an in-depth analysis of the MOIRAI FM (introduced in Section 3.3.5). It examines its performance with various exogenous variable integration approaches, improvements by dataset, and specific timing and practicability considerations, including the practability of its MoE variant. It is important to note that, due to a significant number of missing forecasts, the entire evaluation of MOIRAI in this section is based on only 13 out of the 20 total datasets. The reasons for these missing forecasts will be discussed in detail in the practicability part.

Comparison of Different Exogenous Variable Modes for Moirai

Figure 5.21 presents the Friedman rank plot comparing different exogenous variable modes for MOIRAI. It should be noted that "Lagged Target" and "Temporal Exogenous Variables" modes were excluded from this specific Moirai analysis due to a high number of failed forecasts with these configurations, which prevented a clear assessment (further details in Section 5.1.4).

Similar to TimesFM and TimeGPT, the best average performance for Moirai is achieved without the inclusion of exogenous variables. Interestingly, the "Permuted Exogenous Variables (Noise)" mode performs better in rank than including

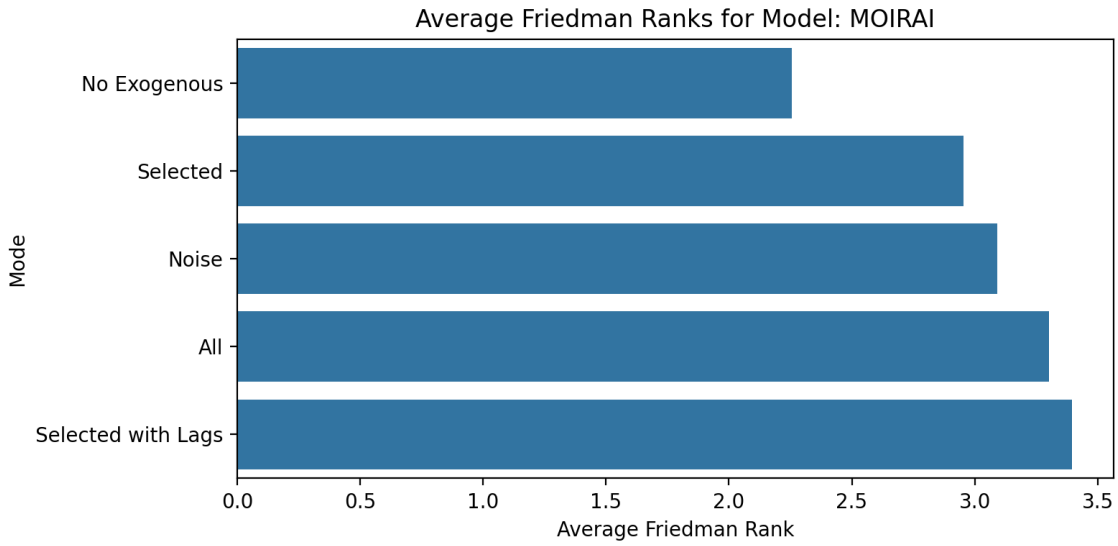


Figure 5.21.: Mean Friedman ranks for MOIRAI with different exogenous variable modes.

all available exogenous variables available or, even more clearly, than a selection of exogenous variables with lags.

The Nemenyi post-hoc test (Figure 5.22) largely confirms these observations. Statistically significantly better results are achieved with "No Exogenous Variables" compared to including "All Exogenous Variables" or "Selected Exogenous Variables with Lags." No statistically significant differences are found among the other modes shown.

Improvement with Exogenous Variables by Dataset for Moirai

When comparing the improvement with these selected exogenous variables across different datasets for Moirai (Figure 5.23), it is first noticeable that results are available for a smaller number of datasets compared to similar analyses for other models (reasons discussed in Section 5.1.4). An improvement in results was achieved in only 2 out of the 13 datasets shown. In contrast, for 10 of these datasets, forecast accuracy deteriorated, sometimes significantly.

Crucially, even when deterioration occurs, the maximal negative impact is comparably low across the datasets, remaining below -20%. This stands out, especially considering that datasets previously identified as problematic for other models, such as "Housing" (which heavily deteriorated for TimeGPT and TimesFM) and "Bike Sharing" (which worsened for Chronos), are present in MOIRAI's evaluation set. However, it's also important to note that, on the other hand, the maximum observed improvement is comparably lower than that seen with other models.

Timing and Practicability Analysis for Moirai

This section discusses the timing characteristics and overall practicability of the MOIRAI models, including both the standard version and the MOIRAI-MoE variant, based on observations during the experimental evaluation.

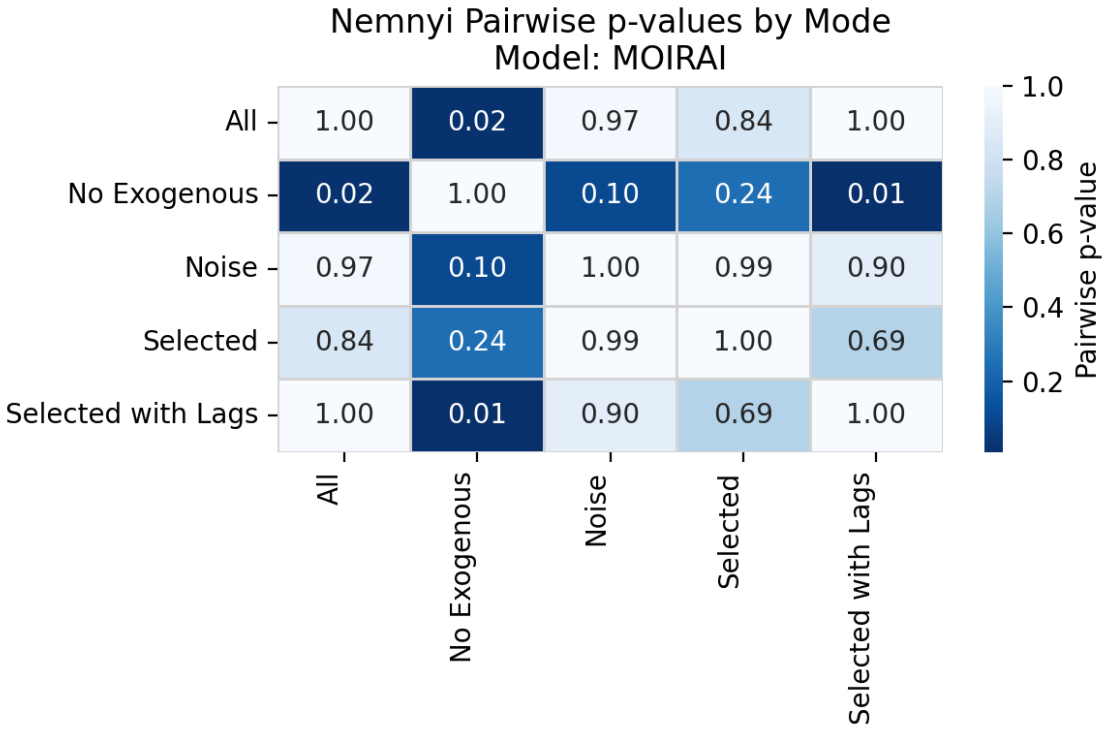


Figure 5.22.: Nemenyi post-hoc test p-value heatmap for Moirai with different exogenous variable modes.

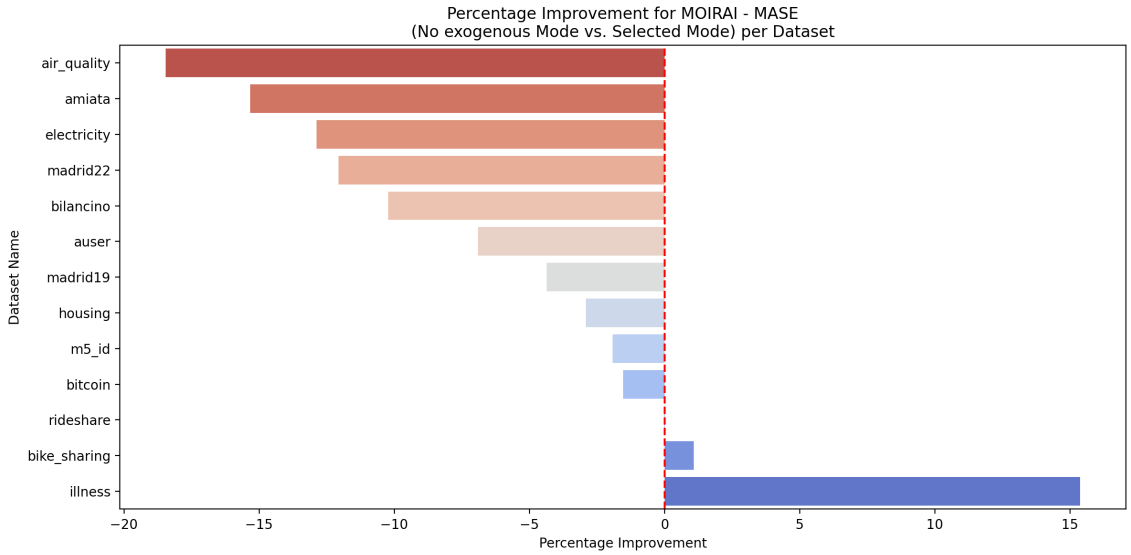


Figure 5.23.: Percentage MASE improvement for Moirai when using the "selected exogenous variables" scenario versus the "no exogenous variables" scenario, by dataset.

Timing Characteristics As will be detailed in the overall timing comparisons later in this chapter (Section 5.2.4), both the standard MOIRAI model and its MoE variant generally exhibited very fast inference times, with a mean of just a few milliseconds. These forecast times showed nearly no variation, indicating a highly consistent and predictable computational load for all predictions.

MOIRAI Practicability MOIRAI frequently encountered "CUDA out of memory" errors, particularly with longer context lengths, higher data dimensionality and longer forecast horizons. This indicates that MOIRAI, despite its overall fast inference time, has significant GPU VRAM requirements that can limit its applicability to complex or high-volume time series data on standard GPU hardware.

These memory issues directly led to a high number of forecast failures in specific scenarios:

- *Specific Exogenous Variable Modes:* The "Lagged Target" and "Temporal Exogenous Variables" modes were excluded from MOIRAI's analysis. The "Lagged Target" mode significantly increased dimensionality (number of targets multiplied by 7 lags), while the "Temporal Exogenous Variables" mode added a substantial number of time-based features. These factors increase VRAM demands, an issue further compounded by the model's patch size mechanism. MOIRAI automatically selects a smaller patch size for finer granularities (e.g., hourly), which increases the total number of patches the model must process, thereby escalating memory consumption. Longer forecast horizons also contributed to higher VRAM usage, exacerbating these issues when combined with multiple targets.

- *Limited Datasets for Improvement Analysis:* The "Improvement with Exogenous Variables by Dataset" analysis (Figure 5.23, Section 5.1.4) presented results for a reduced number of datasets. This reduction was directly due to forecast failures stemming from VRAM limitations when using the "selected exogenous variables" scenario on the excluded datasets.

- *Omission of Short-term vs. Long-term Analysis:* A dedicated "Short-term vs. Long-term Forecasting Performance" analysis was not included for MOIRAI. This was primarily due to the data inconsistencies and the high number of forecast failures encountered with several exogenous variable modes, which prevented a robust and comprehensive comparison across a sufficient range of scenarios and datasets for both short and long horizons. Those failures were mainly caused by the finer granularities combined with longer forecast horizons.

- *Omission of Short-term vs. Long-term Analysis:* A dedicated "Short-term vs. Long-term Forecasting Performance" analysis was not included for MOIRAI. This was primarily due to a high number of forecast failures encountered with several exogenous variable modes, particularly when combining finer granularities with longer forecast horizons.

While reducing the context length helped to mitigate these errors even with higher dimensionality and forecast horizons, attempts to further address this issue by reducing batch size and running inference with `no_grad` were unsuccessful, suggesting

that the problem warrants further investigation beyond these standard optimization techniques. Systematic memory profiling was not performed in this study.

- **General Limitations:** Beyond VRAM issues, forecasts with context windows containing long sequences of zero values also led to failures.

MOIRAI-MoE Practicability The MOIRAI-MoE variant was largely excluded from the final analysis due to a significantly high rate of failed forecasts and exceptionally poor accuracy in initial tests. This underperformance rendered it impractical for a robust comparison against other models in this study. The primary reasons for these failures appear to be architectural and relate to how the model processes time series with characteristics that differ from its pre-training data.

- **Failure on Zero-Value Sequences:** The core issue stems from the model's inability to handle long sequences of zero values, a pattern present in the evaluation datasets but rare in the LOTSA dataset used for pre-training. This manifests in two key ways:
  - **Unstable Normalization:** The model processes data in patches. A patch containing only zeros has a standard deviation of zero, making the normalization step numerically unstable. This instability creates distorted or meaningless token embeddings from these periods of inactivity.
  - **Expert Routing Failure:** The MoE architecture relies on a gating network to route each token to a specialized expert model. When faced with the distorted tokens from zero-patches, the gating network becomes confused. Furthermore, since the experts were not trained on such sparse patterns, they lack a specialized "expert" for handling zero-value sequences, leading to erroneous or failed forecasts.
- **Exclusion from Scenarios:** Due to this fundamental mismatch between the model's pre-trained capabilities and the data's characteristics, the MOIRAI-MoE variant was dropped from further analysis, including scenarios with exogenous variables where high dimensionality would likely have compounded these issues with VRAM limitations.

MOIRAI-MoE Practicability The MOIRAI-MoE variant was largely excluded from the final analysis due to a significantly high rate of failed forecasts and exceptionally poor accuracy in initial tests. This underperformance rendered it impractical for a robust comparison against other models in this study. The primary reasons for these failures appear to be architectural and relate to how the model processes time series with characteristics that differ from its pre-training data.

- **Failure on Zero-Value Sequences:** The core issue stems from the model's inability to handle long sequences of zero values, a pattern present in the evaluation datasets but rare in the LOTSA dataset used for pre-training. This manifests in two key ways:
  - **Unstable Normalization:** The model processes data in patches. A patch containing only zeros has a standard deviation of zero, making the normalization step numerically unstable. This instability creates distorted or meaningless token embeddings from these periods of inactivity.

  - **Expert Routing Failure:** The MoE architecture relies on a gating network to route each token to a specialized expert model. When faced with the distorted tokens from zero-patches, the gating network becomes confused. Furthermore, since the experts were not trained on such sparse patterns, they lack a specialized "expert" for handling zero-value sequences, leading to erroneous or failed forecasts.

- **Confirmed Failure in High-Dimensional Scenarios:** These architectural weaknesses were confirmed during tests involving exogenous variables. Attempts to generate forecasts with the **MOIRAI-MoE** variant in these high-dimensional settings consistently failed. The failures were a combination of internal processing issues and significant GPU VRAM errors. As with the standard MOIRAI model, these memory errors were driven not only by the high dimensionality from the covariates but were also compounded by long context lengths, extended forecast horizons, and finer data granularities that require smaller patch sizes. This combination of factors highlights the model's practical limitations for complex, multivariate forecasting tasks and cements its exclusion from the main comparison.

Conclusion

In summary, the practicability of the MOIRAI architecture is significantly constrained by its substantial GPU VRAM requirements, which lead to frequent forecast failures in complex scenarios involving high data dimensionality, long forecast horizons or fine granularities. The model demonstrates its most reliable performance in simpler contexts without exogenous variables. The inclusion of covariates did not yield consistent accuracy improvements and often resulted in forecast failures, suggesting their benefit is unpredictable at best.

The MOIRAI-MoE variant proved to be even less practical for the tasks in this study. It suffers from fundamental architectural weaknesses, particularly in handling sequences with zero values, and faces exacerbated computational and memory issues. These severe limitations rendered it unsuitable for a robust comparison. Consequently, while the standard MOIRAI model may be useful for other forecasting tasks, its overall utility is hampered by resource-intensive demands and the MOIRAI-MoE variant is not recommended for datasets with characteristics similar to those tested here.

5.1.5. TinyTimeMixer

This section provides an in-depth analysis of the TinyTimeMixer FM, as introduced in Section 3.3.1. It covers its performance with various exogenous variable integration strategies, improvements observed by dataset, and specific timing and practicability considerations, including issues related to its exogenous mixer and handling of certain data types.

Comparison of Different Exogenous Variable Modes for TTM

The average Friedman ranks for TTM (Figure 5.24) clearly indicate that including certain types of exogenous variables can, on average, reduce forecast error across

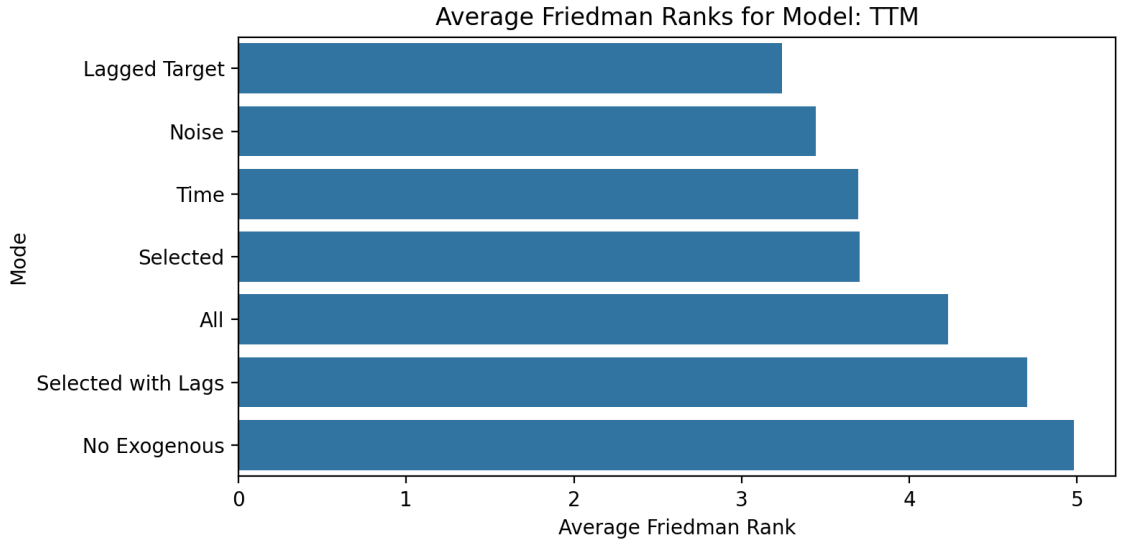


Figure 5.24.: Mean Friedman ranks for TTM with different exogenous variable modes.

various datasets. The specific type of exogenous variable seems to matter: "Lagged Target" as an exogenous variable leads the ranks. Additionally selected exogenous variables mode temporal exogenous variables are on average more beneficial than including all available exogenous variables. Conversely, a selection of exogenous variables with pre-selected lags performs worse in rank than using all exogenous variables. This might suggest that TTM internally handles the temporal relationships between exogenous variables and targets effectively via its exogenous mixer and external pre-shifting of exogenous variables could lead to information loss or misalignment rather than improvement.

Intriguingly, the "Permuted Exogenous Variables (Noise)" mode again performs very well, ranking second. This recurring observation with "Permuted Exogenous Variables (Noise)" for some models might suggest that either the model is robust to such unstructured additional inputs, or that other structured exogenous variable modes are actively detrimental for TTM in many cases, making the neutral "Noise" appear relatively better. It could also be that the model is extracting predictive signal from genuinely random noise but less likely.

Intriguingly, the "Permuted Exogenous Variables (Noise)" mode again performs exceptionally well, ranking second in the Friedman comparison. This recurring observation with "Noise" across certain models, including TTM, suggests several possibilities. Firstly, it might indicate that the model possesses a remarkable robustness to unstructured or irrelevant additional inputs, effectively ignoring the noise rather than being misled by it. Secondly, and perhaps more plausibly, it could imply that the other conventionally structured exogenous variable modes are, in fact, actively detrimental for TTM's performance in many instances, thereby making the truly neutral "Noise" scenario appear comparatively superior. A less likely, albeit theoretically possible, explanation is that the model is somehow extracting a subtle predictive signal even from what is intended to be genuinely random noise.

The Nemenyi post-hoc test (Figure 5.25) statistically substantiates many of these prior statements. Statistically significant improvements in rank against "No Exoge-

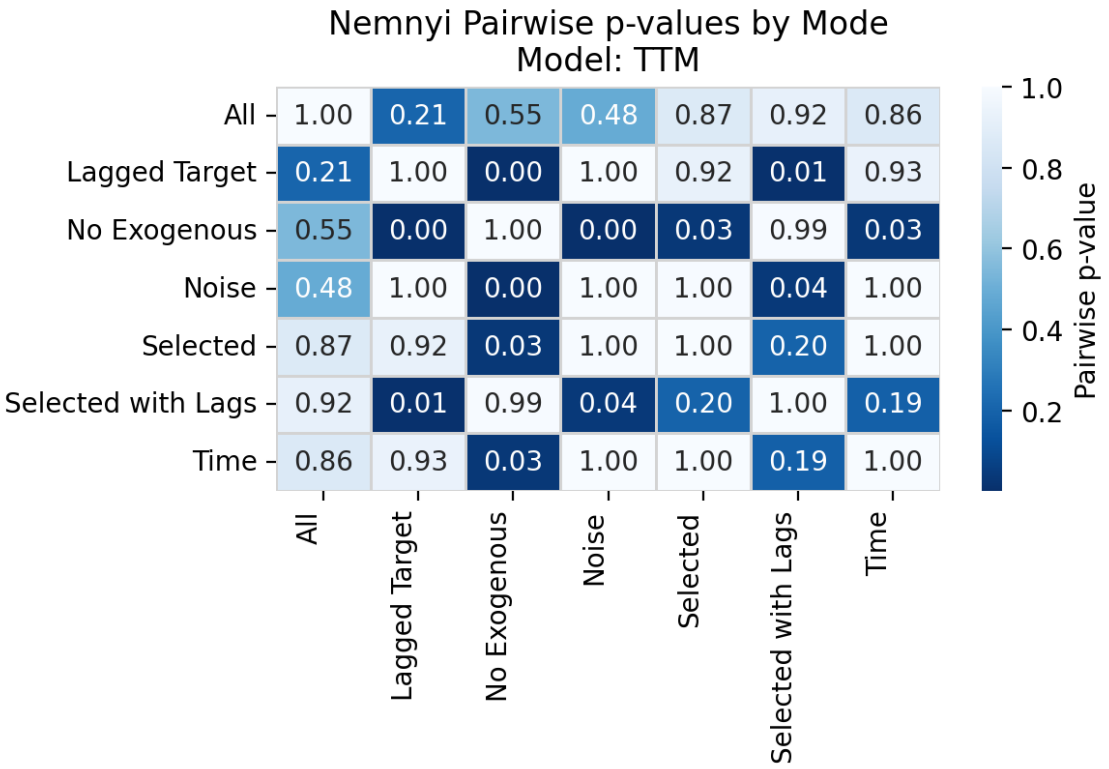


Figure 5.25.: Nemenyi post-hoc test p-value heatmap for TTM (standard model) with different exogenous variable modes.

nous Variables" are confirmed for "Lagged Target," "Permuted Exogenous Variables (Noise)," "Selected Exogenous Variables," and "Temporal Exogenous Variables". Other modes, like "All Exogenous Variables," show no statistically significant difference in rank from "No Exogenous Variables." The "Selected Exogenous Variables with Lags" mode is confirmed to be significantly worse in rank than "Lagged Target" and "Permuted Exogenous Variables (Noise)."

Improvement with Exogenous Variables by Dataset for TTM

Figure 5.26 shows the percentage improvements for TTM across various datasets using the "lagged target" exogenous variable configuration. This specific mode was chosen for this comparison as it was identified as the optimal approach in the preceding Friedman rank analysis. An MASE reduction was observed across 8 of the datasets displayed. However, for the "Auser" dataset, performance deteriorated significantly by nearly 40% with the inclusion of these covariates, while the "Housing" and "M5_ID" datasets showed no improvement or deterioration at all. In stark contrast, the remaining five datasets all experienced strong improvements, with the "Bilancio" dataset benefiting remarkably by nearly 100%. This demonstrates that while the "lagged target" approach can be highly beneficial for TTM on certain datasets, its impact is not universally positive.

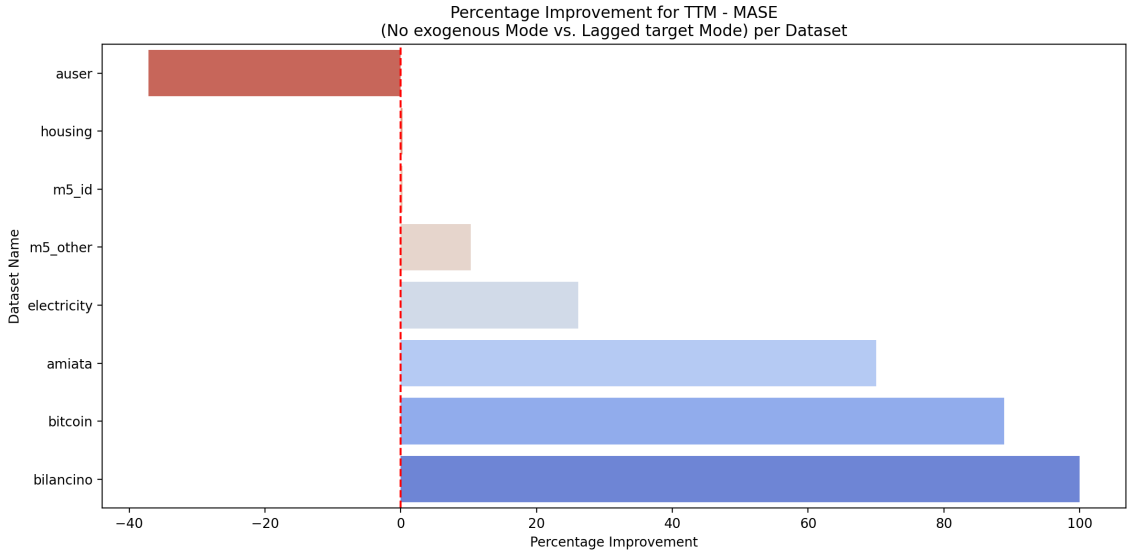


Figure 5.26.: Percentage MASE improvement for TTM when using the "lagged target" versus the "no exogenous variables" scenario, by dataset.

Timing and Practicability Analysis for TTM

This section discusses the timing characteristics and overall practicability of the TTM models, covering both the standard (zero-shot) TTM and the TTM-finetuned variant. The model's architectural design and methodological framework are central to understanding its performance and limitations.

TTM (Standard/Zero-shot Model)

- **Timing and Impact of the Exogenous Mixer:** The standard TTM model had a mean forecast time of approximately 35 seconds. However, it exhibited a significant increase in forecasting time when exogenous variables were introduced (as shown in Figure 5.27), frequently reaching the 5-minute timeout limit set for the experiments. This slowdown is a direct consequence of its architecture: for each forecast involving covariates, TTM trains a small "exogenous mixer" block on the fly. The computational cost of this mixer scales with both context length and the number of channels, making it a major bottleneck.

- **Pre-trained Model Selection Challenges:** A core challenge in using TTM is the selection of appropriate pre-trained model weights. The framework provides multiple weights, each trained with a specific context length and forecast horizon combination often geared towards finer granularities. An automated function selects the pretrained weights for a given forecast, but this choice is not always optimal.

- **Data Sensitivity and Forecast Failures:** TTM's performance was highly sensitive to data characteristics, leading to a high rate of forecast failures.
  - *Series with Zero Values:* The model struggled with time series containing long sequences of zero values, likely due to the instability of its instance normalization method on channels with near-zero variance.

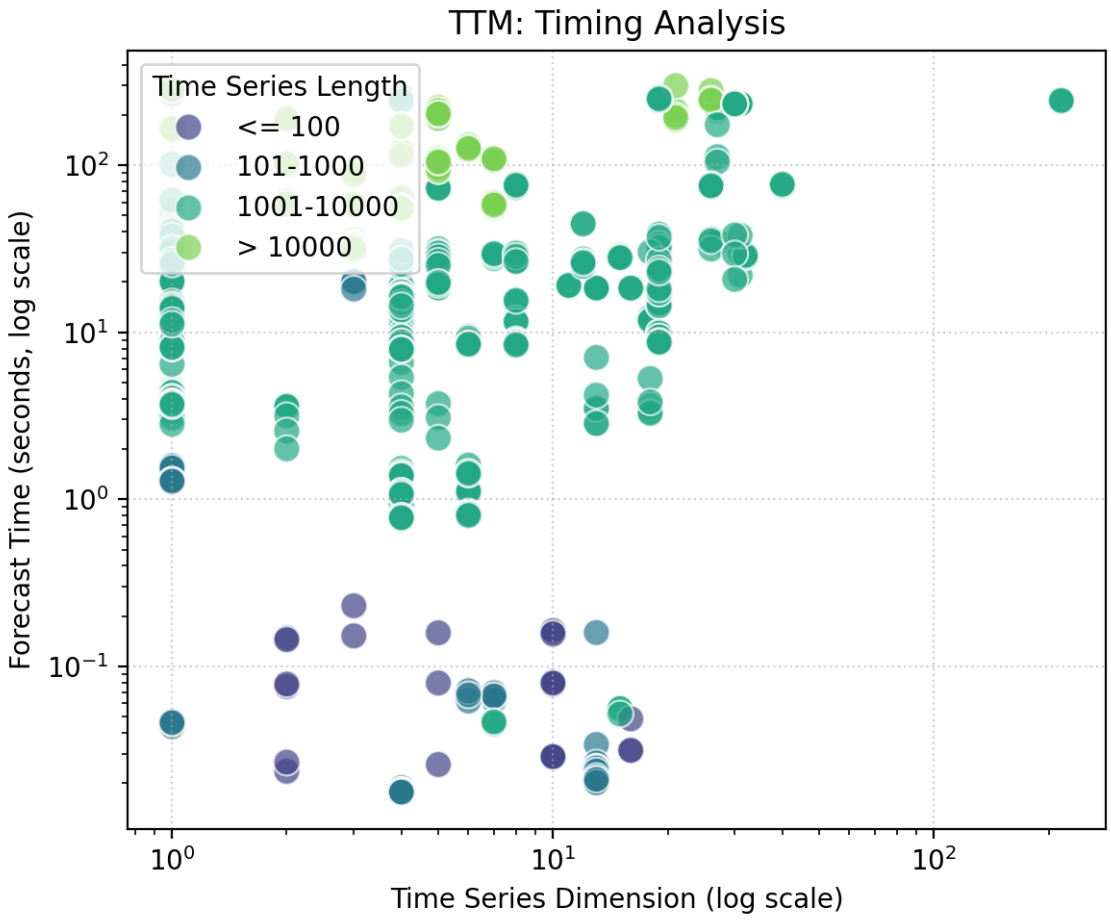


Figure 5.27.: TTM (standard model) forecasting time versus time series length and dimensionality (number of exogenous variables/features).

  - *Short Time Series:* Failures were common for short time series. This is due to two factors: First, the pre-trained models have minimum context length requirements that may not be met. Second, the on-the-fly mixer requires a sufficient amount of data to train effectively.

- **Resource Usage:** Unlike the MOIRAI model, GPU VRAM was not a significant bottleneck for TTM. The primary resource limitation was computational time. The intensive on-the-fly training of the exogenous mixer for high dimensional data led to long processing durations and frequent timeouts, rather than memory exhaustion.

- **Limited Scope of Performance Analysis:** The high failure rate significantly limited the scope of the analysis. A dedicated short-term versus long-term performance comparison was omitted. Furthermore, the evaluation of exogenous variable modes was restricted to only 8 datasets, as forecasts could not be successfully generated for many dataset-mode combinations.

TTM-finetuned Variant

- **Practicability and Timeout Issues:** The TTM-finetuned variant was largely excluded from the comparative analysis due to severe practicability issues.

Mostly it hit the 5-minute timeout limit, especially in scenarios with many exogenous variables.

- **Architectural and Methodological Reasons for Failure:** The frequent failures during fine-tuning can be attributed to several intertwined factors:
  - *Decoder and Mixer Capacity:* The model's channel-independent backbone is frozen, and fine-tuning activates a small decoder and mixer with very low capacity, which may be insufficient to learn highly complex cross-channel correlations.
  - *Compounded Computational Cost:* The fine-tuning process repeatedly runs the computationally expensive exogenous mixer, amplifying the timing issues seen in the standard model and leading directly to timeouts.
  - *Sub-optimal Model Selection:* The automated selection of pre-trained weights is particularly problematic for fine-tuning. For instance, the process can fail if the chosen model's context length is not a good fit for the training data's length (e.g., if the series length is not a multiple of the context length), or if the series is too short to begin with.
- **Performance Implications:** Due to the consistent timeouts and failures, a comprehensive performance profile for the TTM-finetuned model could not be established. The variant's significant reliability issues made it impractical for the forecasting tasks in this study.

5.2. Overall Model Comparison

In this section, we compare all models, including FMs and established methods, providing an overall overview of the results. The focus here is on general performance, with detailed analysis of different exogenous variable setups presented in subsequent subsections.

Comparison of All Models without Exogenous Variables

Figure 5.28 presents a bar plot of the mean MASE for all models, with color highlighting for different model categories: statistical, ML, DL and FMs. The results are averaged over all datasets, time series, forecast horizons and windows, as detailed in Chapter 4. The findings indicate that TimesFM performs best in this scenario, albeit by a small margin over other FMs such as Moirai and TTM. The ML model LightGBM is positioned in the middle range, also with a small distance to the top-performing models. A more significant performance gap exists to the baseline models (ARIMA, ETS, Theta) and the DL models (NBEATSx and TiDE). Moirai MoE exhibits a mean MASE nearly three times that of the other leading models, positioning it as the poorest performer in this setup.

The plot also includes confidence intervals which illustrate the variability in model performance across different datasets and time series. These confidence intervals are notably large, suggesting that there is no single universally superior model across all experimental setups. Consequently, further analysis using statistical tests, as outlined in Section 4.5.3, is required to determine if statistically significant differences exist in model performance and accuracy.

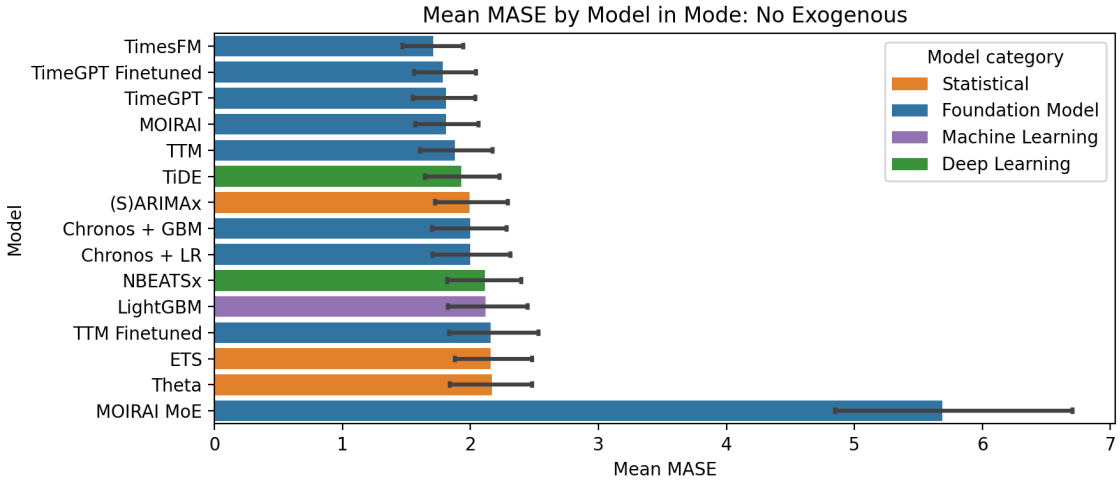


Figure 5.28.: Mean MASE barplot for all models without exogenous variables, aggregated across all forecast setups.

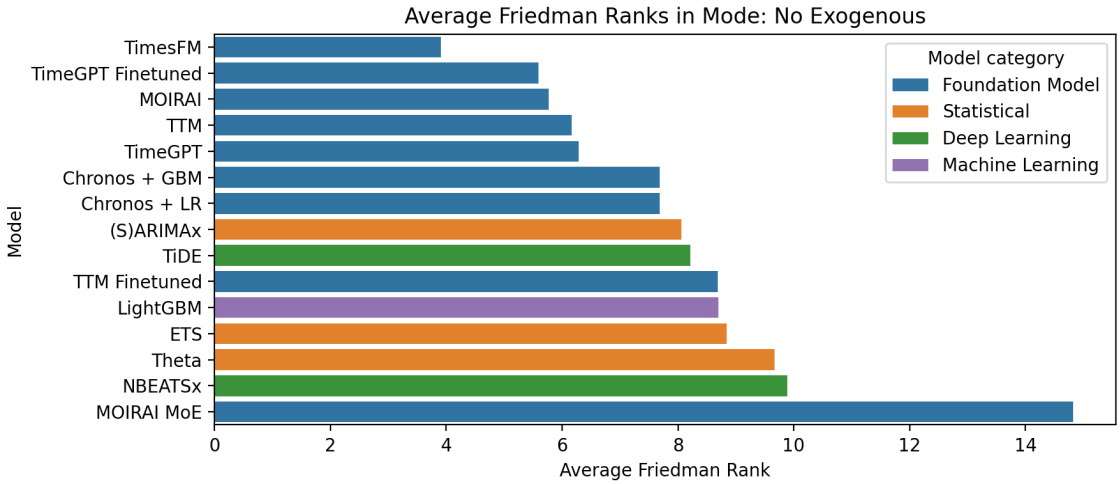


Figure 5.29.: Mean Friedman ranks for all models without exogenous variables.

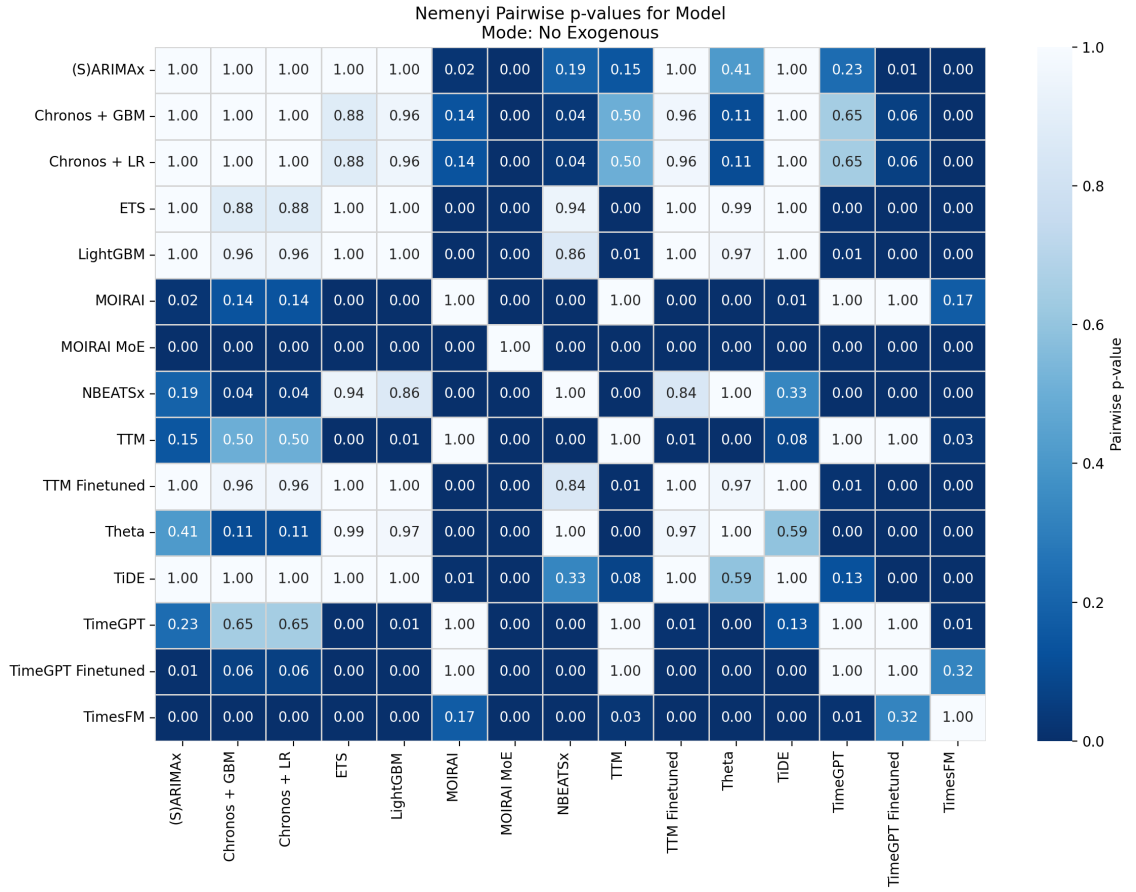


Figure 5.30.: Heatmap of p-values from the Nemenyi post-hoc test for model performance without exogenous variables.

The subsequent plot (Figure 5.29), displaying the mean Friedman ranks across all setups, largely corroborates the results from the MASE bar plot. TimesFM again ranks highest, with an even more pronounced lead than indicated by the mean MASE. Conversely, Moirai MoE is clearly ranked lower than the other models. However, this rank plot still lacks detailed pairwise comparison information. Therefore, we next examine a heatmap of p-values derived from the Nemenyi post-hoc test, which facilitates direct comparisons between pairs of models.

The heatmap in Figure 5.30 presents the p-values from the Nemenyi post-hoc test, clarifying statistically significant differences in accuracy. When viewed in conjunction with the Friedman rank plot, it becomes evident that TimesFM significantly outperforms most other models. This is indicated by the blue squares in the heatmap along the TimesFM row/column representing p-values below the significance threshold of 0.05. Statistically significant differences are not observed when comparing TimesFM to Moirai and TimeGPT-finetuned. Models such as ARIMA, Chronos, ETS, LightGBM, TTM-finetuned and TiDE demonstrate comparable performance levels among themselves, despite minor variations in their mean ranks, as they do not show statistically significant differences from each other in many pairwise comparisons.

In conclusion, for the scenario without exogenous variables, no single model consistently outperforms all others across every dataset and time series. However, TimesFM provides the best overall results on average and is therefore a strong

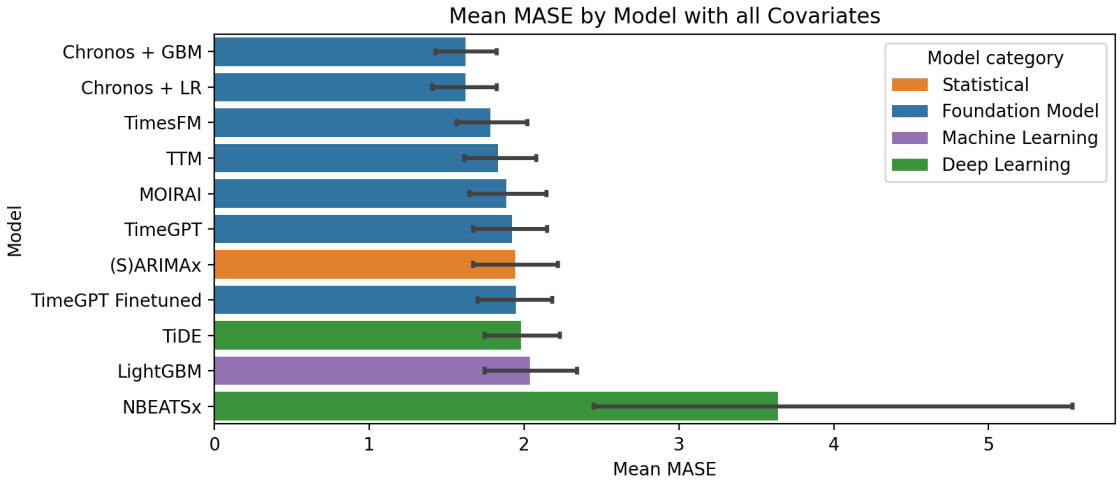


Figure 5.31.: Mean MASE barplot for models with all available exogenous variables, aggregated across all forecast setups.

candidate when no exogenous variables are considered. Moirai MoE, on the other hand, performs notably poorly in this setup. This contrasts with claims from its creators about improved accuracy over the standard Moirai model, as was further discussed in the MOIRAI analysis (Section 5.1.4).

Comparison of All Models with All Exogenous Variables

Figure 5.31 once again displays a bar plot of mean MASE with confidence intervals, this time for models incorporating all available exogenous variables. For this analysis, Moirai MoE and TTM-finetuned were excluded due to a high number of missing forecasts. These omissions were caused by exceeding computational resources (Moirai MoE) and encountering timeouts during fine-tuning (TTM-finetuned), as detailed in their respective practicability subsections in Section 5.1 (specifically Section 5.1.4 and Section 5.1.5). Additionally, baseline models that cannot natively integrate exogenous variables were also excluded from this specific comparison.

In this exogenous variable-inclusive setting, the Chronos model performs best, with nearly identical results for both regressor types tested. NBEATSx shows a considerable performance gap to the other models and is clearly positioned at the bottom. This is contrary to some benchmarks and papers where DL models like NBEATSx and TiDE typically perform better than many established methods [Arango et al. 2025; Li et al. 2024]. The subpar performance of NBEATSx and TiDE in this study is likely attributable to the absence of extensive hyperparameter tuning and the imposition of a maximum limit on training steps, as detailed in Section 4.2.3.

The Friedman rank plot (Figure 5.32), based on MASE values, presents these results with greater clarity. Chronos is distinctly at the top, followed by TimesFM. The remaining models perform comparably to each other. The DL models, NBEATSx and TiDE, are now both worst-performing, which, as previously mentioned, is due to the training and hyperparameter limitations.

Combined with the Nemenyi post-hoc heatmap (Figure 5.33), we can conclude that Chronos is statistically superior to the other models, with the exception of

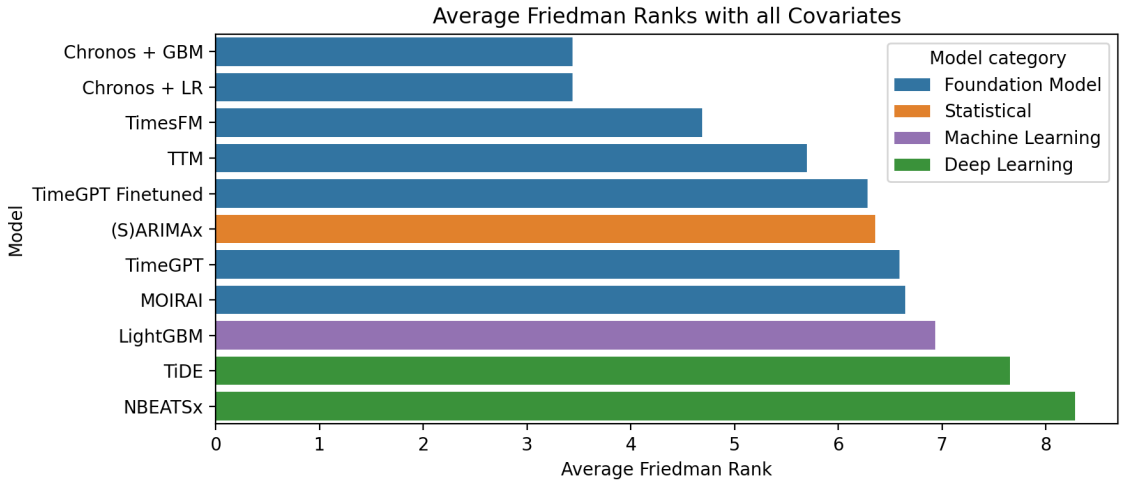


Figure 5.32.: Mean Friedman ranks for models with all available exogenous variables.

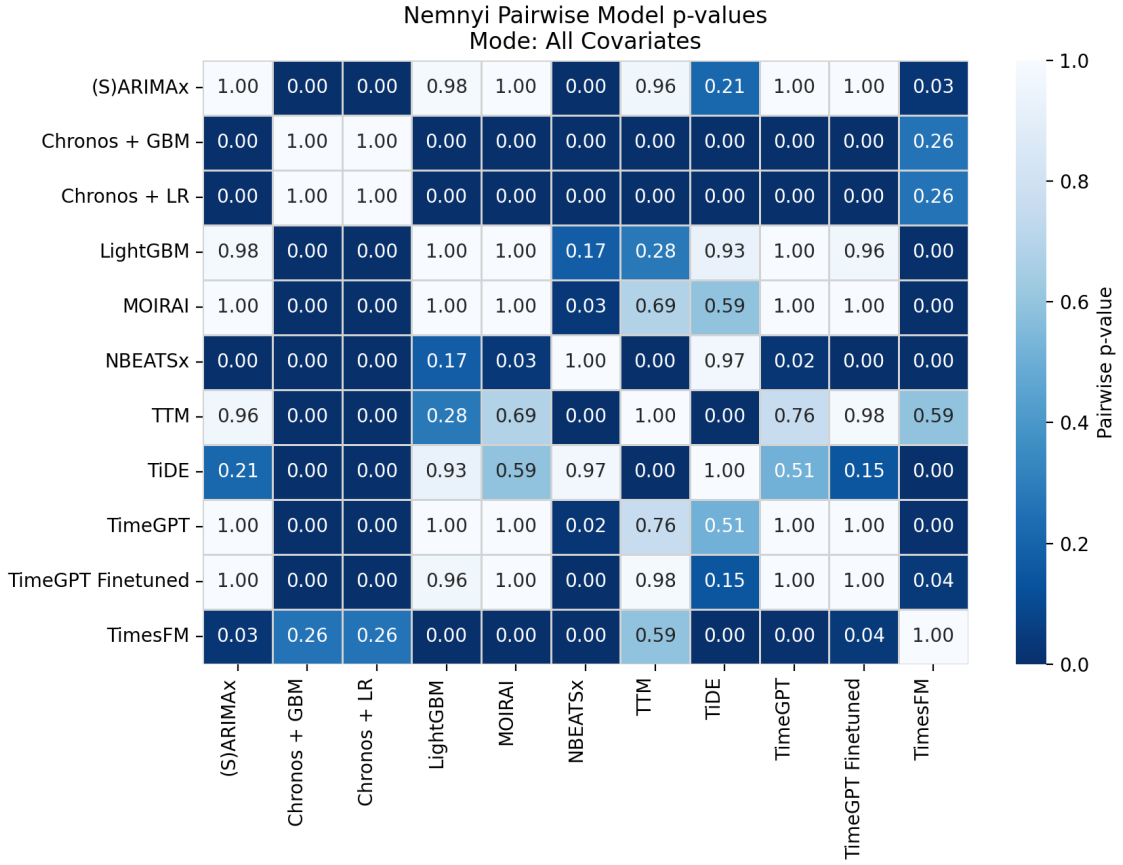


Figure 5.33.: Heatmap of p-values from the Nemenyi post-hoc test for model performance with all available exogenous variables.

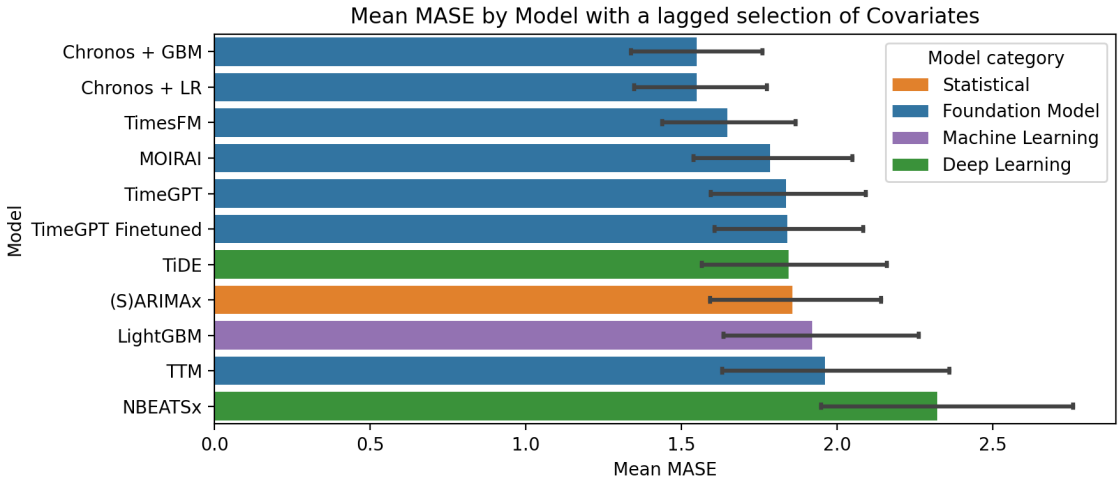


Figure 5.34.: Mean MASE barplot for models with selected exogenous variables and individually chosen lags, aggregated across all forecast setups.

TimesFM. TimesFM, in turn, significantly outperforms other models except for Chronos and TTM. While some statistically provable differences exist between other pairs of models, their overall performance levels are relatively close. Therefore, among these other models, while specific model-to-model differences are apparent, distinct overall performance tiers are less evident.

The inclusion of all available exogenous variables demonstrably alters the performance landscape of the models. TimesFM and Chronos, which utilize external regressors, performed well even without exogenous variables (as seen in Section 5.2) and appear to benefit further from their integration, with Chronos showing a more pronounced improvement in this "all exogenous variables" scenario. However, this aggregate improvement requires further investigation at the individual model level, as some models exhibit a decrease in accuracy with exogenous variables, a point explored for each FM in Section 5.1.

Comparison of All Models with Selected Exogenous Variables

This section evaluates model performance using a curated set of exogenous variables, where both the exogenous variables themselves and their respective lags were selected based on a specific methodology (detailed in Section 4.3.3 of Chapter 4). This scenario is termed "Lagged Selection".

Figure 5.34 presents the comparison bar plot of mean MASE with confidence intervals for models using the "selected exogenous variables with individually chosen lags" scenario. Compared to the "all available exogenous variables" scenario, both LightGBM and (S)ARIMAx show improved performance. This improvement is expected, particularly for LightGBM, as the variable selection methodology itself is based on this model's feature importance.

Conversely, TTM's performance has degraded significantly, moving it from the upper midrange to the second-to-last position. N-BEATSx, while still the lowest-performing model, also narrowed the performance gap to the other models compared to its performance with all covariates. So N-BEATSx sees the biggest improvement by an covariate selection. Another notable finding is that TimeGPT benefits from

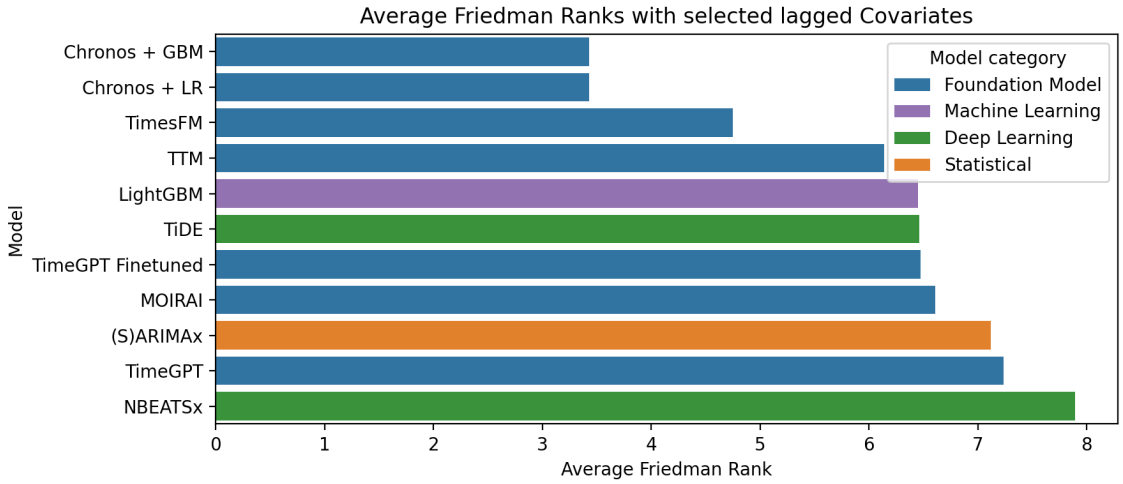


Figure 5.35.: Mean Friedman ranks for models with selected exogenous variables and individually chosen lags.

fine-tuning in this scenario, improving its standing, which contrasts with the "all covariates" scenario where no fine-tuning was preferable. Chronos remains the top-performing model in this configuration.

The Friedman rank plot (Figure 5.35), which evaluates the frequency of outperformance rather than the magnitude of error, reveals a different performance hierarchy compared to the MASE bar plot (Figure 5.34). Notably, TTM, LightGBM, and TiDE achieve surprisingly better ranks, positioning them in the upper-middle tier. This is especially significant for TTM, which improves from being the second-worst model in the MASE comparison to fourth place in the rankings. For LightGBM, the strong rank further underscores the advantage gained from being the model used for the exogenous variable and lag selection process.

The overall ranking structure shows Chronos as the clear leader, followed by TimesFM. A noticeable performance gap then separates these top two models from the remaining models, which are clustered closely together.

Together, the Friedman rank plot and the Nemenyi post-hoc heatmap (Figure 5.36) reveal similar conclusions to those from the "all available exogenous variables" analysis (Section 5.2). Chronos is demonstrably better than the other models, except for TimesFM. In this scenario, the performance differences among the other models (excluding NBEATSx, which remains a lower performer) have narrowed compared to the "all available exogenous variables" scenario. This suggests that a curated set of relevant exogenous variables and lags tends to bring their performance closer together.

5.2.1. Short-term vs. Long-term Forecasting Performance

This section investigates how model performance, measured by mean MASE, varies between short-term and long-term forecasting horizons. We examine this aspect first for the scenario without exogenous variables and then for the scenario with selected lagged exogenous variables.

Figure 5.37 replicates the mean MASE bar plot with confidence intervals, now disaggregated to compare short-term and long-term forecasting performance for the

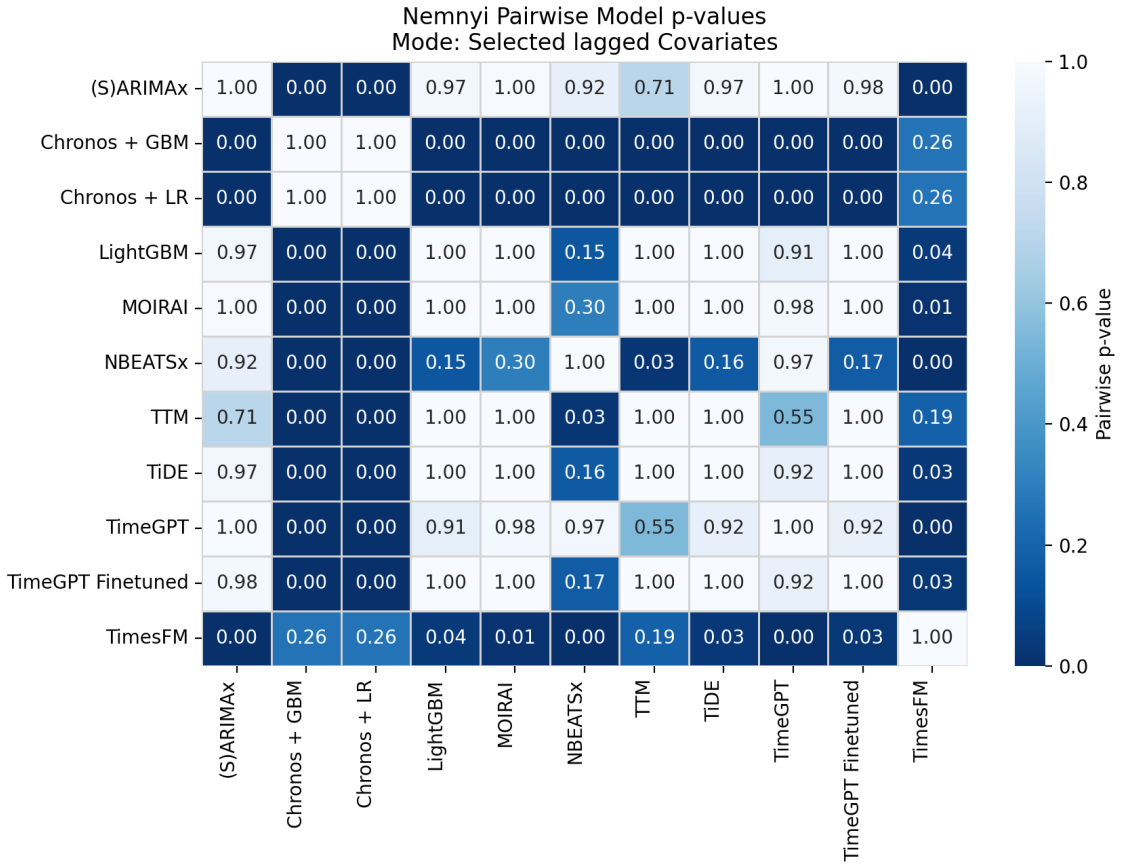


Figure 5.36.: Heatmap of p-values from the Nemenyi post-hoc test for model performance with selected exogenous variables and individually chosen lags.

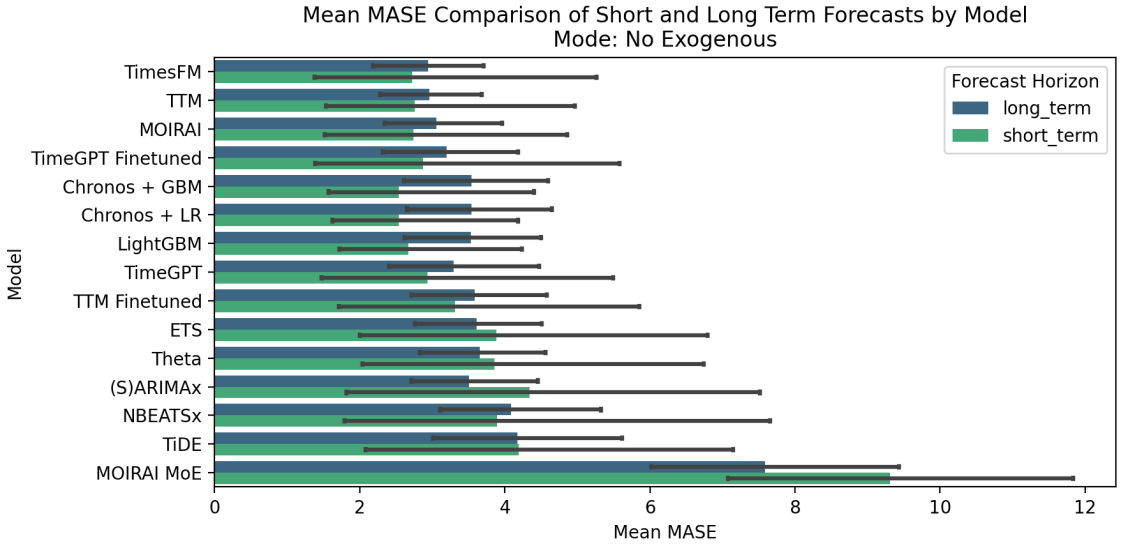


Figure 5.37.: Comparison of mean MASE for short-term versus long-term forecasts without exogenous variables, aggregated across all forecast setups.

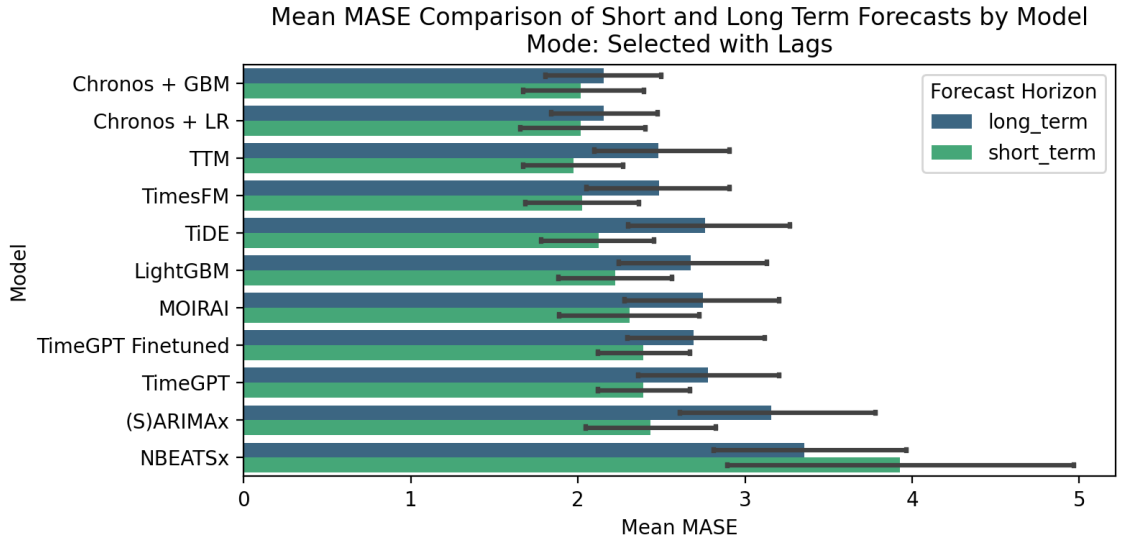


Figure 5.38.: Comparison of mean MASE for short-term versus long-term forecasts with selected lagged exogenous variables, aggregated across all forecast setups.

setup without exogenous variables. Most models perform better in the short-term setting, generally without statistically significant differences in the aggregated view, though individual model behaviors vary. However, MOIRAI MoE and (S)ARIMAx notably show better mean MASE in the long-term setting under these conditions while Chronos and LightGBM show the opposite.

Figure 5.38 presents the same comparison, but this time for the scenario including "selected exogenous variables with individually chosen lags". Again, most models demonstrate superior performance in the short-term setting. For some models, such as TiDE, TTM, TimesFM, LightGBM and MOIRAI, the differences in mean performance between short-term and long-term horizons appear more pronounced in this setup with selected exogenous variables compared to the no-exogenous-variables scenario. (S)ARIMAx switched from better performance on long-term forecasts without exogenous variables to better performance on short-term forecasts with the selected exogenous variables.

5.2.2. Cross-Dataset Performance Analysis

To further explore model performance beyond aggregated metrics, this section examines how different models perform across a selection of individual datasets. The datasets included in this analysis are described in detail in Section 4.1.3 of Chapter 4.

Figure 5.39 displays a heatmap with models on the y-axis and a selection of datasets on the x-axis. The exogenous variable mode used here is again the "selected exogenous variables with individually chosen lags" setup. This plot largely reinforces the findings from the aggregated plots. Chronos appears to perform best on average across these datasets, indicated by the highest number of dark blue squares representing the lowest mean MASE for a given dataset.

However, there is no universally consistent pattern in this plot. For instance, NBEATSx, which was identified as an overall weaker performer in the aggregated

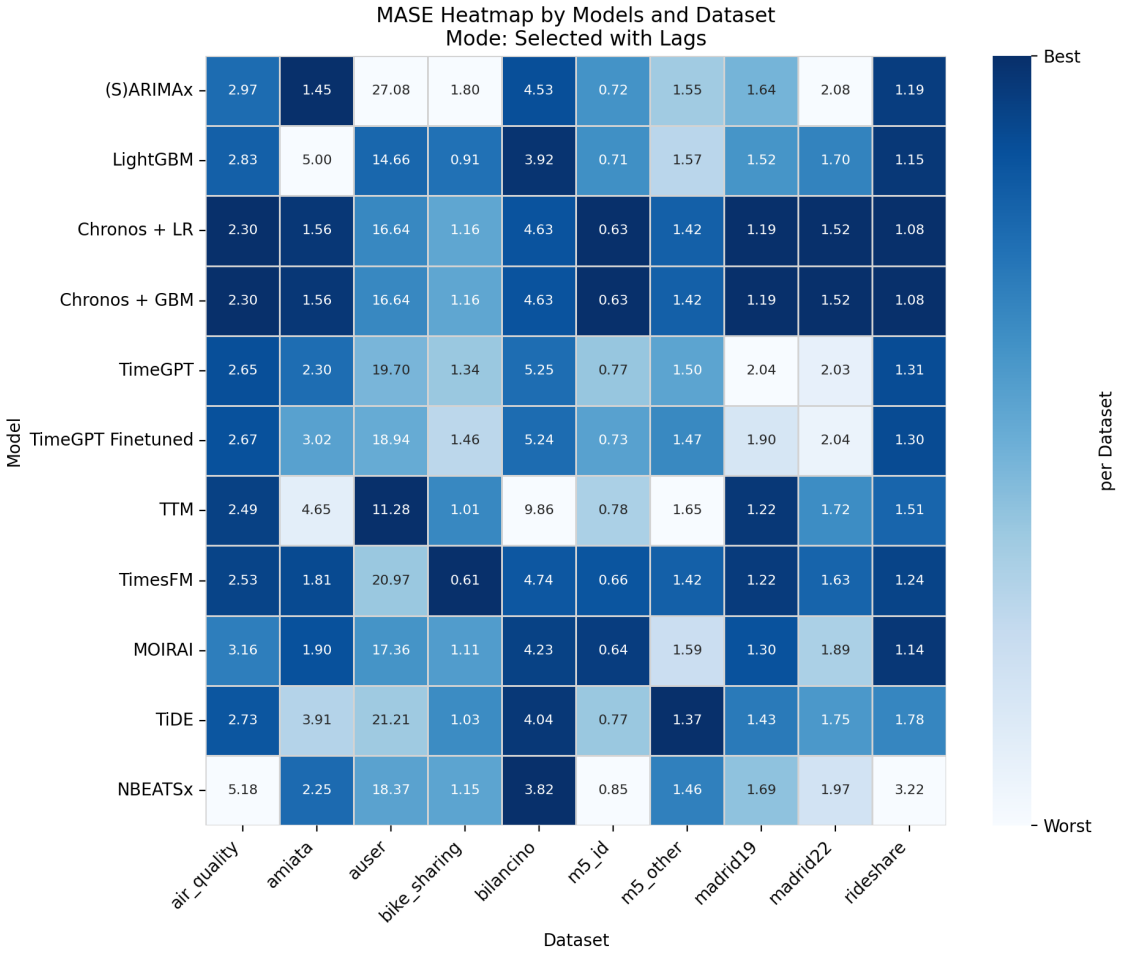


Figure 5.39.: Heatmap of mean MASE per model across a selection of datasets, using the "selected exogenous variables with individually chosen lags" setup.

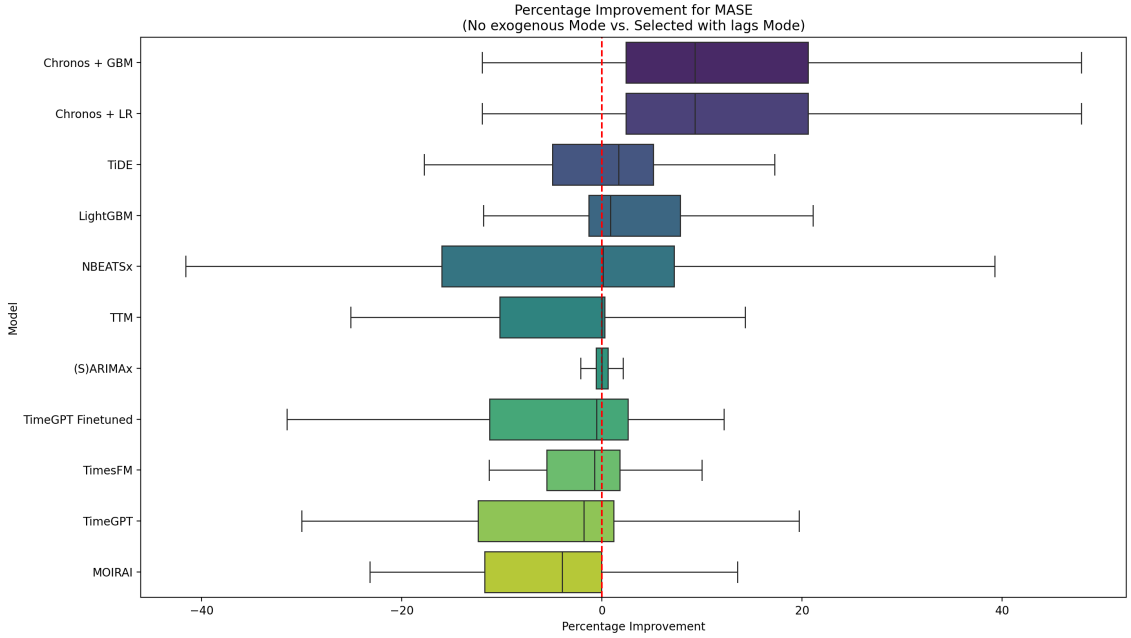


Figure 5.40.: Boxplots of percentage improvement in MASE when using the "selected exogenous variables with individually chosen lags" scenario compared to the "no exogenous variables" scenario, by model. Positive values indicate improved MASE (lower error) with exogenous variables.

analyses (Sections 5.2 and 5.2), is the best model on the "bilancino" dataset according to this heatmap. This observation underscores a critical point in TSF: no single model, not even large-scale FMs, is universally optimal. Model selection should ideally be guided by the specific characteristics of the dataset at hand and the particular forecasting task.

5.2.3. Impact of Exogenous Variables on Model Improvement

This section specifically investigates the extent to which the inclusion of exogenous variables improves or degrades model forecasting accuracy. The analysis compares model performance in the "no exogenous variables" scenario against the "selected exogenous variables with individually chosen lags" scenario.

Figure 5.40 further illustrates that while some models generally perform better than others (as seen in Sections 5.2 to 5.2), there is no single best model overall, and the benefit of exogenous variables is highly model-dependent. The plot shows boxplots for each model, depicting the percentage improvement in MASE when including "selected exogenous variables with individually chosen lags" compared to using no exogenous variables. A positive percentage indicates that MASE was reduced with exogenous variables, while a negative percentage indicates MASE increased.

The plot also reveals no substantial difference for the chosen regressor model type used with Chronos in terms of overall improvement distribution. Chronos consistently shows improvement from exogenous variable integration in most cases, with a positive median improvement. TiDE and LightGBM also exhibit a mean and median improvement. However, their lower 25% quantiles are negative, indicating

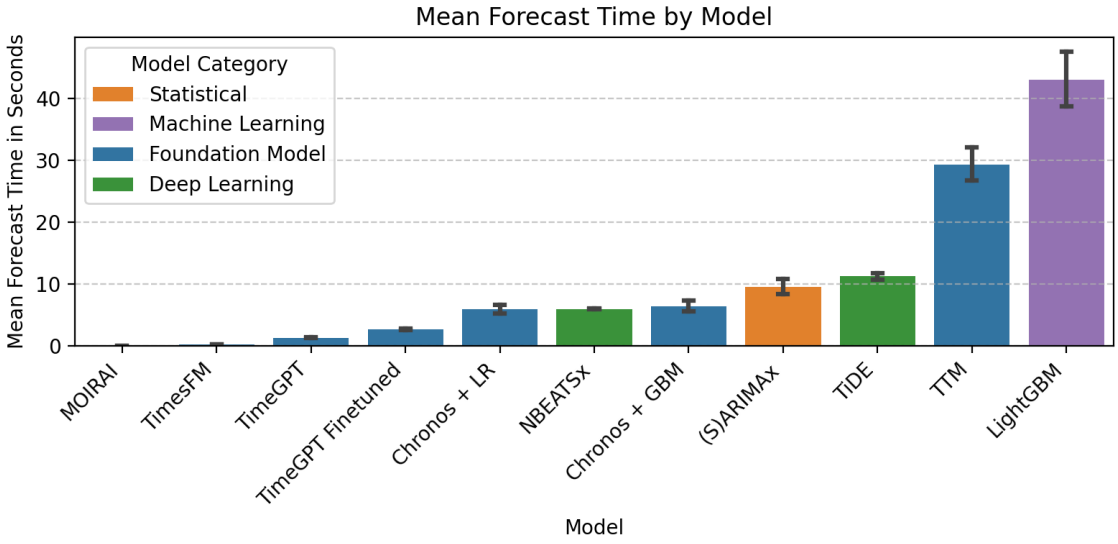


Figure 5.41.: Mean forecasting time per series for different models, averaged across various experimental setups.

that for a notable portion of time series, exogenous variables do not guarantee improvement and can occasionally degrade performance for these models.

N-BEATSx and TTM show a mean improvement close to zero, but with very wide interquartile ranges. N-BEATSx in particular exhibits large performance variations in the range of +/-40%, which highlights the necessity of pre-evaluation and parameter tuning on specific datasets before deploying it with exogenous variables. In contrast, ARIMAx also has a mean improvement around zero but shows a comparably small variation, suggesting that the inclusion of exogenous variables has only a minor influence on its overall performance.

TimeGPT, TimesFM, and especially Moirai demonstrate a negative mean and median improvement. This suggests that these specialized FMs, under this specific "selected exogenous variables with individually chosen lags" configuration, might indeed be optimized for univariate forecasting, or that this particular method of exogenous variable integration is not beneficial for them on average. While exogenous variables can sometimes improve their forecasts, this is not the case for the majority of datasets evaluated. For Moirai accuracy worsened in approximately 75% of the time series when these selected exogenous variables were included.

5.2.4. Forecasting Speed of Different Models

Beyond forecast accuracy, the practical usability of models also depends on their computational efficiency, particularly their forecasting speed. This section provides an overview of the mean forecasting time per series for the evaluated models.

Figure 5.41 presents a bar plot of the mean forecasting time for different models, averaged over various forecast setups. As anticipated, FMs generally exhibit fast prediction times. Moirai is notably quick, with predictions typically completed in under milliseconds. It is closely followed by TimesFM. This is interesting because its forecasting time includes the training of its external regressor, which appears to be highly optimized. TimeGPT ranks third, followed by its fine-tuned version. It is important to note that TimeGPT forecasts were obtained via the Nixtla API,

which operated on different computational resources than the other GPU-accelerated models. Surprisingly, the fine-tuning process for TimeGPT, when invoked, is also completed in seconds and does not significantly extend the zero-shot inference time based on these observations.

Chronos follows, with a mean of approximately 6 seconds per forecast for both external regressor models. These are commendable results for a FM running on a CPU. NBEATSx's timing falls between the two Chronos setups. This highlights its optimized MLP architecture allowing for relatively fast training/inference, but, as noted in Section 5.2, its accuracy performance in this study was subject to training limitations outlined in Chapter 4. A timeout of 5 minutes was imposed for GPU API requests during the experiments, which particularly affected DL models like NBEATSx and TiDE if they required extensive training for a given series. This also applies to the TiDE model. Forecasts that resulted in such timeouts are not included in this timing analysis, meaning these reported times are for successful, non-timeout runs.

TTM is the slowest among the FMs evaluated here, with a mean forecasting time of 28 seconds. This is primarily due to the training of its exogenous mixer component, which occurs at each forecast instance when exogenous variables are used. LightGBM takes the longest on average, averaging over 40 seconds per forecast. It should be noted that LightGBM, ARIMA, and Chronos were run on CPUs, while other models utilized GPUs, making direct speed comparisons imperfect across these hardware categories but still indicative within them. The 'statsforecast' implementation of AutoARIMA is quite fast, especially considering it performs underlying hyperparameter optimization and training multiple model versions during its process.

6. Discussion

This chapter discusses the key findings of the thesis, which was designed to investigate the integration of exogenous variables into FMs for TSF. The primary objective was to determine whether, and to what extent, these models can effectively incorporate external information to improve their predictive accuracy compared to both their univariate counterparts and established forecasting methods. The investigation surveyed existing FMs capable of this integration, analyzed their architectural approaches, evaluated their performance under various exogenous variable scenarios, and benchmarked them against other models.

The following sections will interpret the empirical results presented in Chapter 5 by systematically addressing the four research questions established in Section 1.2. For clarity, these questions are restated here:

- **F1:** Which specific FMs or model approaches currently exist for TSF that enable the integration of exogenous variables, and how are they technically implemented?
- **F2:** To what extent can the forecasting performance of FMs be improved by incorporating exogenous variables, and how does the forecasting performance of these models with exogenous variable integration compare to established, non-FMs that also utilize exogenous variables, on relevant datasets?
- **F3:** How do different configurations of exogenous variables (e.g., regarding the number, type, data format, or preprocessing) influence the forecasting performance of the investigated FMs?
- **F4:** What specific strengths and weaknesses do the identified approaches to exogenous variable integration in FMs exhibit, based on theoretical considerations and empirical results?

This structured discussion aims to synthesize the theoretical and empirical insights gained throughout this work, leading to a comprehensive conclusion on the current state and future potential of FMs in this domain.

6.1. Interpretation of Key Findings

6.1.1. Synthesis of Foundation Models and Covariate Integration Techniques

The survey of FMs in Chapter 3 reveals a diverse and rapidly evolving landscape of techniques for integrating exogenous variables. As categorized in Section 3.4, these methods do not follow a single standard but can be broadly grouped into two dominant paradigms: Modular / Staged Integration Strategies and Unified Input /

Early and Deep Integration Strategies. The technical maturity of these approaches varies significantly, with many appearing as experimental additions rather than core, fully developed features.

A prominent pattern is the modular or staged integration, where the influence of exogenous variables is handled by a distinct module or in a separate processing stage. The methods in this category are not truly zero-shot, as they require a training or fitting step on the specific time series being forecast. One sub-category is External Regression / Residual Modeling, a pragmatic approach adopted by **Chronos** [Ansari et al. 2024] and **TimesFM** [Das, Kong, Sen, et al. 2023]. This method, reminiscent of classical models like **ARIMAX**, leverages a separate tabular model (e.g., **LightGBM**) to capture the impact of covariates. The FM's role is then to model either the original series or the residuals from the regressor. The `xreg lib` of **TimesFM**, for instance, explicitly implements both variants. While the pre-trained FM core is used in a zero-shot capacity, the need to train the external regressor makes the end-to-end process dependent on the new data. The primary strengths of this method are its implementation simplicity and speed. However, it carries notable weaknesses: it fundamentally relies on the availability of future-known covariates, it risks error propagation if those covariates must themselves be forecasted, and simple regressors may fail to capture complex, time-dependent interactions. A second, more integrated sub-category involves Dedicated Internal Refinement Modules. **TTM** 's [Ekambaram, Jati, Dayama, et al. 2024] *Exogenous Mixer Block* is a key example, functioning as an adapter-like component that is fine-tuned on-the-fly to refine forecasts using known future covariates. The **ChronosX** framework [Arango et al. 2025] advances this concept with model-agnostic adapters, namely an *IIB* for past covariates and an *OIB* for future ones, which can be added to a frozen pre-trained FM backbone.

In contrast, the unified input strategy treats exogenous variables as additional variates to be processed deeply alongside the target series within a single, homogenized architecture. The simplest form is Feature Augmentation and Concatenation, likely used by **TimeGPT**, but more sophisticated FMs employ Homogenized Multi-Variate Processing. **MOIRAI** [Garza et al. 2023] pioneered this with its innovative *AVA* mechanism and variate ID embeddings, allowing the model to process an arbitrary number of variables. However, its practical zero-shot effectiveness is constrained by a pre-training regimen that did not explicitly teach meaningful cross-variate relationships. Consequently, its performance with covariates can be unpredictable without further training. The architectural potential is nonetheless significant, as demonstrated by its winning performance in a forecasting competition where a dedicated fine-tuning effort successfully adapted the model to the specific multivariate characteristics of the dataset. Arguably the most advanced concept in this category is found in **TimerXL** [Yong Liu, Qin, Huang, et al. 2024]. Its key innovation is a Kronecker-based masking mechanism that incorporates a user-defined *Variable Dependency Graph (C)*. This allows for the explicit injection of expert domain knowledge about inter-variable relationships directly into the model's attention patterns, offering a powerful pathway to context-aware forecasting that may mitigate the need for extensive fine-tuning.

In summary, the technical landscape for covariate integration in TSF FMs is still nascent. The trade-offs are clear: modular approaches offer flexibility and preserve the core univariate FM but may miss deeply integrated non-linearities, while unified

strategies promise to learn these complex interactions but are more demanding on the quality and structure of pre-training data and often require fine-tuning. The challenges observed confirm that most integration methods are not yet fully mature. The development of sophisticated, model-agnostic adapters like those in **ChronosX** signals a particularly promising path forward, potentially offering the best of both worlds. This approach maintains the integrity of a powerful, pre-trained univariate FM while using dedicated, non-linear injection blocks to model complex covariate relationships, thereby emulating the deep-learning strength of unified strategies. One could even envision future research further improving this paradigm by combining the adapter-based methodology with the expert-guided mechanisms of **TimerXL**, allowing a *Variable Dependency Graph* to inform the behavior of the injection blocks for an even more tailored and effective integration.

6.1.2. Analysis of Forecasting Performance with Exogenous Variables

The empirical results from Chapter 5 lead to a nuanced primary conclusion: the impact of incorporating exogenous variables on the performance of FMs is not universally positive but is instead highly contingent on the model's specific integration architecture, as discussed in Section 6.1.1, and on the underlying characteristics of the dataset. The example forecasts for **Chronos** presented at the beginning of the analysis (Section 5.1.1) effectively illustrate this duality, showing that covariates can both clarify and obscure underlying patterns, leading to either improved or degraded predictive accuracy. This confirms that despite the scale of FMs, the "no free lunch" theorem persists. There is no single configuration that guarantees improvement across all contexts.

The investigation reveals starkly divergent behaviors across the different FMs, largely aligning with their architectural strategies. Models with modular integration strategies showed the most consistent benefits. **Chronos**, with its external regressor approach, proved to be the most reliable, achieving a performance improvement on approximately 85% of the datasets. This gain was largely independent of the specific regressor type used (e.g., linear model vs. LightGBM), pointing to the robustness of the residual modeling strategy itself. Similarly, **TTM** showed statistically significant improvements, particularly when supplied with autogenerated features like lagged target values or temporal features, suggesting its mixer block is effective at leveraging such information.

In stark contrast, models with different integration schemes struggled to reliably extract value from covariates. The results for **TimesFM** were particularly revealing, despite also using an external regressor, it saw no statistically significant improvement. In fact, its performance with selected covariates was often statistically indistinguishable from using no covariates or even random noise. For nearly two-thirds of the datasets, adding covariates actively degraded its forecasts, suggesting its core univariate mechanism is so powerful, or its integration method so basic, that it cannot reliably distinguish signal from noise in external variables. This negative effect was even more pronounced for **TimeGPT** and **MOIRAI**, where adding covariates was far more likely to harm than help performance. For these models, it appears the addition of high-dimensional, potentially irrelevant information can pollute the embedding space or distract the attention mechanism, leading to worse

performance than a focused univariate forecast.

When comparing the models in a multivariate setting, these architectural differences proved decisive. According to the Friedman tests and performance rankings (e.g., Figure 5.35 and Figure 5.34), **Chronos** emerges as the clear top-performing model when exogenous variables are utilized. It consistently outperformed not only other FMs but also established methods, demonstrating that its hybrid approach—a powerful pre-trained univariate core paired with a simple, adaptable machine learning model—was superior in this benchmark. Traditional models like **LightGBM** and **SARIMAX** remained competitive, performing only slightly worse than the leading FMs. It is also important to note that the DL models **NBEATSx** and **TiDE** underperformed in this study due to restricted training configurations, as they are known to be top performers in other benchmarks [Arango et al. 2025].

Ultimately, the analysis underscores that no single model is a panacea for all forecasting problems, a fact visually reinforced by the wide performance distribution in the dataset heatmap (Figure 5.39). Even the overall weakest performer, **NBEATSx** , was the best model for one specific dataset. The per-model improvement boxplots (Figure 5.40) further confirm this high variance, showing that the benefit of covariates is inconsistent. For a practitioner, this implies that while an FM like **Chronos** presents a strong a priori choice for problems with rich exogenous data, it does not eliminate the need for rigorous, dataset-specific benchmarking against both other FMs and well-established classical methods to ensure optimal model selection.

6.1.3. Impact of Covariate Configurations

The experimental results offer a clear verdict on the conventional "more data is always better" assumption: in the context of feeding exogenous variables to FMs, this assumption is largely invalidated. The findings consistently demonstrate that an indiscriminate inclusion of all available covariates can be detrimental. This was most evident with **TimesFM**, **TimeGPT**, and **MOIRAI**, all of which struggled with high-dimensional inputs and frequently exhibited performance degradation when supplied with an uncurated set of all available variables. This suggests that without a sufficiently sophisticated integration mechanism, an influx of high-dimensional data can introduce more noise than signal, overwhelming or distracting the models' learned patterns.

This leads to the clear conclusion that some form of feature selection is beneficial, even for FMs that are theoretically supposed to weigh input importance internally. However, the effectiveness of a specific configuration is highly model-dependent. For **Chronos**, while a pre-selected set of variables appeared helpful, the improvement over using all available covariates was not statistically significant. Its robust external regressor approach seems capable of extracting value from almost any provided set of external variables. In sharp contrast, **TimesFM**'s performance was statistically indistinguishable across ‘no exogenous‘, ‘noise‘, and ‘selected‘ scenarios, indicating that its integration method was unable to differentiate between meaningful signals and random data. This highlights a critical insight: for some FM architectures, a simple feature selection performed externally is insufficient to guarantee performance gains.

While this thesis focused on zero-shot performance, the limited fine-tuning experiments with **TimeGPT** revealed another layer of complexity. Fine-tuning had

a dual, amplifying effect: in many cases, it exaggerated the existing trend, making good results even better and bad results worse likely through overfitting. However, for a handful of datasets where the zero-shot model's performance was degraded by covariates, fine-tuning was able to reverse this outcome and produce an improvement. This indicates that while fine-tuning can make a model more sensitive and prone to overfitting, it may be a necessary step for certain architectures to properly learn the complex relationships between a target series and its covariates.

Furthermore, the analysis revealed critical, model-specific reactions to feature engineering, particularly concerning lagged variables. Manually lagging *exogenous covariates*, a standard necessity for classical ML regressors, was generally not an effective strategy for the FMs. This suggests these advanced models, designed to process raw temporal sequences, perform better when they learn time-shifted relationships internally rather than being fed pre-lagged features. An even more nuanced picture emerged from the *'lagged target'* scenario. This single feature produced sharply contrasting outcomes: it significantly improved performance for **TTM** , indicating its *Exogenous Mixer Block* effectively leverages this explicit auto-regressive signal, yet it substantially degraded the performance of **TimesFM** and **TimeGPT**. This negative impact is likely due to architectural redundancy, as these latter models presumably generate such auto-regressive features internally, making their external provision duplicated and therefore confusing.

In conclusion, the choice of covariate configuration is not a trivial preprocessing step but a critical factor that can determine the success or failure of leveraging external data. The results show that some models, like **Chronos**, are robust and benefit from nearly any external information, while others, like **TimesFM** and **TTM** , exhibit extreme sensitivity where the same feature type can be either highly beneficial or highly detrimental. This implies that effective use of covariates requires an understanding of the model's internal mechanisms and intended workflow (zero-shot vs. fine-tuning). The performance gap between a beneficial and a harmful configuration can be significant, making the art of covariate curation as important as the choice of the model itself.

6.1.4. Strengths, Weaknesses, and Architectural Implications

The empirical investigation reveals a complex profile of strengths and weaknesses for the current generation of FMs when applied to forecasting with exogenous variables. A final analysis of these aspects, linked back to the architectural choices discussed in Section 6.1.1, is essential for addressing F4.

Strengths: Simplicity, Speed, and Generalization The primary strength of these FMs is inherited from their univariate application: they largely circumvent the traditional, labor-intensive forecasting pipeline. For a practitioner, this means less time spent on model selection, hyperparameter tuning, and training. The models leverage their vast pre-training to offer strong generalization capabilities, which is particularly advantageous for time series with limited historical data. Two practical advantages became particularly evident in the context of this study. First, the integration of covariates, especially in models like **Chronos** or **TTM** , simplifies the workflow by not strictly requiring an exhaustive feature pre-selection process. Second, and most impressively, is their computational efficiency. Even when in-

corporating a "training" step (e.g., for an external regressor or mixer block), the FMs were significantly faster than traditional methods. For instance, **TimesFM** and **MOIRAI** generated forecasts in under a second on GPU hardware. Even **Chronos**, running on a CPU , produced an average forecast in approximately five seconds—twice as fast as a highly optimized **AutoARIMA** model. This speed, combined with the ease of parallelization on modern GPU infrastructure, marks a significant practical advantage over both classical and many bespoke DL models.

Weaknesses: Practicability, Interpretability, and Dependencies Despite their strengths, the FMs exhibit considerable weaknesses. On a practical level, while data preprocessing requirements like imputation are standard for all models, FMs introduce new challenges. Most models are optimized for and often require GPU hardware, though their parameter counts are typically small enough for consumer-grade cards. More critically, specific models showed significant practicability issues: **MOIRAI** suffered from high VRAM consumption with high-dimensional inputs, and **TTM** 's fine-tuning process led to prohibitively long training times and forecast failures (timeouts). A general weakness observed across several DL-based models was a vulnerability to sequences of zero values: If an entire input patch consisted of zeros, it could cause normalization failures, a key reason for the reduction of datasets used in this study. A major conceptual limitation is the lack of interpretability. The models largely function as "black boxes," providing no insight into which features are driving the forecast. While it should be theoretically possible to derive feature importance from the external regressor models, this functionality is not yet implemented in standard libraries. This stands in contrast to established DL models like **TFT**, which demonstrate that architectural components for interpretability are indeed possible. Finally, the models' performance is subject to strong dependencies. As previously discussed, the benefit of covariates is highly dataset-dependent, which runs contrary to the "foundational" promise of universal applicability. Furthermore, the most successful modular approaches rely on future-known covariates, a significant constraint for many real-world problems. The workaround of pre-forecasting these covariates is inherently risky, as it can propagate errors into the final target forecast.

Architectural Implications Synthesizing the results, a clear pattern emerges regarding architectural choices. Within the zero-shot or near-zero-shot focus of this thesis, the modular integration strategy proved to be the more reliable and effective paradigm. The success of **Chronos** demonstrates that combining a powerful, pre-trained univariate core with a simple, separate regressor was more robust than the more complex, unified input methods that were not subjected to extensive fine-tuning. This does not imply that unified input architectures like **MOIRAI**'s are inherently inferior, but rather that their true potential is likely conditional on a fine-tuning step, which was outside the primary scope of this work. The most telling architectural insight comes from comparing **Chronos** and **TimesFM**. Both use an identical external regressor strategy (modeling residuals), yet they produce opposite results regarding the utility of covariates. This strongly suggests that the difference in outcome is not due to the integration method itself, but rather to the intrinsic properties of the core FM architectures and how their internal mechanisms respond to the statistical characteristics of the time series data — whether original or residual

— that they are tasked with forecasting.

6.2. Cross-Cutting Themes and Synthesis

The individual findings of this thesis converge on several cross-cutting themes regarding the current state of FMs in TSF. This section synthesizes these themes to provide a holistic perspective on their generalizability, practical readiness, and emerging design patterns when handling exogenous variables.

A primary conclusion is that the celebrated generalizability of FMs does not yet seamlessly extend to the integration of covariates in a zero-shot context. As the results in Chapter 5 demonstrate, the inclusion of external variables often produces inconsistent, sometimes random, changes in performance. The finding that permuted noise variables could produce results statistically equivalent to a curated feature set for models like **TimesFM** and **TTM** is a stark indicator of this limitation. The root cause appears to be a critical gap in the pre-training ecosystem: a lack of large-scale, publicly available datasets with rich and diverse covariate information. If the models' univariate generalization is already incomplete — as evidenced by the fact that no single model consistently outperforms the others across all dataset characteristics - then it is logical that they struggle to learn the additional, vast complexity introduced by exogenous variables without specific training. The **ChronosX** study reinforces this, suggesting that this knowledge gap can be effectively bridged with targeted, lightweight fine-tuning.

The findings point to a nuanced answer concerning the models' practical readiness. For widespread deployment involving covariates, FMs are best viewed not as zero-shot universal tools, but as powerful, adaptable engines within a "pre-train and fine-tune" paradigm. In this role, they are exceptionally well-suited for industrial applications involving thousands or millions of time series, such as inventory management, where their speed and scalability are paramount. However, for applications demanding the highest possible accuracy on a single critical forecast, specialized and deeply-tuned models will likely remain superior. A hybrid strategy, such as hierarchical forecasting, could effectively combine both worlds: using FMs for rapid forecasts at low-level, high-volume granularities (e.g., individual products with general economic covariates), which are then aggregated and reconciled with more specialized forecasts at higher levels of the hierarchy. Without fine-tuning or a specific evaluation, relying on an FM with covariates is ill-advised, as one cannot be certain if the external data will improve or degrade the otherwise strong univariate forecast.

Synthesizing the architectural and empirical findings, an emerging "best practice" becomes apparent: a hybrid, modular approach. The most successful and robust strategy observed in this thesis involves selecting a FM with proven, powerful univariate capabilities and augmenting it with a dedicated, external module for covariate integration. The adapter-based design of the **ChronosX** framework stands out as the most promising blueprint for this pattern. While its code was not yet public at the time of this study, its methodology represents a pragmatic and powerful path forward. The **TimerXL** architecture, with its unique capacity for injecting domain knowledge, is another highly promising avenue that warrants future investigation. This modular paradigm appears to be the most effective current solution, successfully bridging the gap between the generalized knowledge of

a pre-trained core and the specific, contextual information provided by exogenous variables.

6.3. Limitations

A critical reflection on the scope and methodology of this thesis is necessary to properly contextualize its findings. The research was subject to several limitations across its experimental design, model selection, and the broader context of the current research field.

6.3.1. Limitations of the Experimental Design

The most significant constraint on this study was the availability and nature of suitable datasets. The investigation was limited by the availability and suitability of large-scale, public time series datasets that include a rich set of well-documented exogenous variables. This made it impossible to conduct a detailed comparison of how models handle *past-known* versus *future-known* covariates, a crucial distinction for practical applications. Many available datasets also lacked diversity in granularity, with a heavy focus on hourly and daily data. Furthermore, the time series were often too short to reliably support the training of DL models or extensive fine-tuning experiments. This scarcity prevented a systematic, characteristic-based selection of diverse *datasets*, similar to the one performed for the individual time series within them 4.3.3.

The scope of the evaluation was further constrained by limited computational resources and the project timeline. This necessitated a focused evaluation on a specific set of forecast horizons and a "global" forecasting approach, where all target series from a dataset were processed simultaneously. This global method might encourage models to learn cross-series patterns but could mask poor performance on individual series whose relationship with covariates is unique. A "local" approach, forecasting each series individually, was not investigated and could yield different results. Additionally, the evaluation focused primarily on point forecast accuracy metrics.

The evaluation centered on point forecast accuracy to establish a robust performance baseline. While this is a critical first step, this focus means that several important analyses were designated as future work. These include a detailed analysis of probabilistic forecasts to assess how well models quantify uncertainty, a specialized assessment for sporadic time series using metrics like PIS or CFE , and a formal investigation into the models' robustness to outliers. The data required for these deeper evaluations, including the probabilistic forecasts and their corresponding error metrics, has been collected, positioning these topics as the primary objective for forthcoming research.

6.3.2. Limitations of the Model and Configuration Scope

The selection of models and configurations also imposes limitations. The baseline DL models (**NBEATSx** , **TiDE**) were intentionally trained with limited resources and without extensive hyperparameter optimization. While this arguably created a fairer comparison against the zero-shot FMs, it does not reflect the full potential

of these models, which are designed to achieve state-of-the-art performance with proper, dataset-specific training. The practicability analysis could also have been more extensive, for example, by including detailed VRAM tracking or a deeper root-cause analysis of model failures.

Furthermore, several promising models and frameworks were not included in the benchmark. Most notably, **TimerXL**, whose code became available late in the research process, and the **ChronosX** framework, which was not yet publicly released, were omitted. The exclusion of specialized DL architectures like **TFT**, a key benchmark for interpretable forecasting with covariates, also limits the comparative breadth of the findings.

Covariate Selection Methodology The process for selecting covariates represents a significant methodological limitation. While systematic, the approach relied on a single model, **LightGBM**, to rank and select the most important features. This method is inherently biased. The features deemed important by a tree-based model like **LightGBM** are not necessarily the same ones that a Transformer-based FM would find most useful. An FM might be better suited to identifying subtle, long-range, non-linear dependencies that a tree-based model might overlook. A more robust, albeit computationally prohibitive, approach would have been a model-specific selection process, such as a wrapper method that uses each FM to evaluate the utility of each potential covariate. The combinatorial explosion from such a method rendered it infeasible. Alternative statistical or ML -based techniques, such as those discussed in Section 2.5.4 like Recursive Feature Elimination (RFE) or regularization-based selection, were also not explored.

6.3.3. General Limitations in the Research Field

Beyond the specifics of this project, this work was constrained by a systemic issue in the time series research community: the lack of standardized, high-quality, and large-scale public benchmark datasets for forecasting with exogenous variables. This gap hinders the ability of researchers to rigorously pre-train, fine-tune, and fairly evaluate the capabilities of FMs in leveraging external information. The development of such resources is a critical next step for advancing the field and advancing robust, generalizable forecasting solutions.

6.4. Future Work and Research Directions

The findings and limitations of this thesis illuminate several promising avenues for future research. Advancing the state of FMs for forecasting with exogenous variables will require concerted efforts in developing more sophisticated architectures, creating richer benchmarks, and pursuing unanswered methodological questions.

6.4.1. Future Directions in Model Architecture and Integration

The field of FMs for time series is characterized by rapid and continuous innovation. New architectures are constantly being proposed, and performance on univariate

benchmarks is consistently improving as different approaches leapfrog one another. This dynamic landscape presents a clear research opportunity: to decouple the most effective *integration strategies* from any single base model and evaluate them as modular, plug-and-play components.

Future architectural research should therefore focus on the most promising paradigms identified in this work. A key direction is the systematic evaluation of modular approaches on a portfolio of new and emerging FMs. The adapter-based design of the **ChronosX** framework, for example, should be tested not only with **Chronos** but also with other state-of-the-art models to rigorously assess its claims of model-agnosticism and effectiveness. A parallel track could evaluate the simpler external regressor approach across these same models to serve as a robust baseline. Furthermore, a comprehensive benchmark including **TimerXL** is essential to evaluate the practical benefits of its sophisticated, expert-guided attention mechanism. Another critical direction is the incorporation of interpretability-by-design, drawing inspiration from architectures like **TFT** to move FMs beyond their current "black box" status. Finally, research should address the integration of static covariates, often linked to hierarchical forecasting problems. Adapting advanced frameworks like Nixtla's Hierarchical Mixture Networks (HINT) for use with FMs could be a fruitful area of investigation [Olivares, Luo, et al. 2023].

6.4.2. Development of Next-Generation Benchmarks and Datasets

The single greatest catalyst for progress would be the development of new, high-quality benchmark datasets. Future work should focus on creating larger and more diverse datasets with a wider range of granularities and sufficient length for robust DL model training and fine-tuning. This can be achieved through several avenues. First, by expanding upon existing work on synthetic data generation, such as the simplified approach in the **ChronosX** paper. While a good starting point, this method should be developed further to create more realistic data that captures a wider spectrum of complex, non-linear relationships between endogenous and exogenous variables. Second, novel cross-modal approaches, inspired by models like **VisionTS** which found links between image and time series data, could be explored for generating rich, correlated multivariate datasets, potentially helping to close the current data gap. Third, a pragmatic approach involves the manual enrichment of existing public datasets with domain-expert-sourced covariates, such as adding detailed weather and holiday information to energy consumption datasets. Any new benchmark should also include a deep characterization of not only the time series but also the covariates and their interactions to enable more insightful analysis of model performance.

6.4.3. Unanswered Questions and Methodological Investigations

This thesis revealed several specific gaps in the current understanding that warrant dedicated future studies. A central unanswered question is the precise role of **fine-tuning**. A systematic investigation is needed to determine whether, as this work

suggests, some form of few-shot learning is a prerequisite for FMs to effectively learn from covariates.

The scope of evaluation should also be expanded. A dedicated study on **probabilistic forecasting** is required. This is a critical area, particularly as this work noted that models like **TimesFM** and **TTM** do not currently provide probabilistic outputs when covariates are integrated, limiting the analysis to **Chronos**, **MOIRAI**, and **TimeGPT**. Likewise, a targeted evaluation on **sporadic and intermittent time series** using appropriate metrics like PIS and CFE is needed. While MASE was used here as a robust general-purpose metric, different forecasting tasks have different objectives, and analyzing model bias (over- or under-forecasting via CFE) is vital. Although the data for such an analysis was generated in this thesis, it was not evaluated in depth.

Further methodological deep-dives could include: a controlled comparison of the impact of **past-known vs. future-known covariates**, which requires the development of suitable datasets. A formal study on the **robustness of FMs to outliers** in both target and covariate series, testing the claim that pre-training confers such capabilities and a systematic analysis of **global vs. local forecasting** strategies to understand the trade-offs for multivariate-capable models like **MOIRAI** and **TTM** . The exploration of more advanced **covariate selection** techniques is a massive field in itself and a clear avenue for future work. Finally, the error logging system developed for this thesis, which links errors to forecast metadata via a unique ID, enables a future study on **automated error log analysis** to connect for example specific failure modes to underlying time series characteristics.

6.5. Summary of the Discussion

This discussion has synthesized the primary findings of the thesis into several key conclusions that define the current landscape of FMs for TSF with exogenous variables:

The "No Free Lunch" Rule for Covariates: The most critical finding is that adding exogenous variables to FMs is not a guaranteed improvement but a high-risk, architecture-dependent endeavor. The results show that while some models benefit reliably, others can be significantly degraded, challenging the "more data is better" axiom and proving that the method of integration is paramount.

Primacy of Modular Architectures (in the Current Paradigm): In the near-zero-shot context of this thesis, modular integration strategies proved more robust and practical than deeply unified ones. The success of the external regressor approach and the promise of adapter-based frameworks suggest that augmenting a powerful univariate FM core is a reliable path to success. This conclusion is strongly linked to the following point, as this preference for modularity exists in a research landscape where the full potential of deeply unified architectures with extensive fine-tuning has not yet been fully explored.

A Necessary Shift from Zero-Shot to Fine-Tuning: The effective use of covariates with FMs likely requires a paradigm shift from a pure zero-shot application to a "pre-train and fine-tune" workflow. The research indicates that the generalized knowledge from pre-training is often insufficient to capture specific, complex covariate relationships, making some form of targeted, few-shot fine-tuning a necessary step for reliable practical deployment.

The Data Bottleneck: The primary bottleneck for advancing FMs in this domain is the systemic lack of large-scale, high-quality public benchmark datasets with rich and well-documented covariates. Without such data for pre-training and evaluation, the full potential of any architecture, whether modular or unified, will remain unrealized, and generalizability will remain limited.

7. Conclusion

7.1. Summary of Problem Definition and Study Objectives

TSF is a fundamental task for strategic planning and decision-making across countless domains. The accuracy of these forecasts can often be significantly enhanced by including external factors, so-called exogenous variables, that provide important context. The methodologies for TSF have undergone a significant evolution, progressing from classical statistical models like ARIMA to more flexible DL architectures and, most recently, to the paradigm of large-scale, pre-trained **FMs**. These FMs , having demonstrated remarkable success in other sequence-based domains like Natural Language Processing, offer the promise of powerful generalization and reduced need for specialized, task-specific model design.

However, at the outset of this research, it was observed that the initial application of FMs to time series was predominantly focused on univariate scenarios. The critical area of exogenous variable integration was an emerging yet fragmented field, characterized by a multitude of new publications but a lack of structured overviews or comparative analyses. This created a significant research gap: practitioners and researchers lacked a clear understanding of which FMs could handle covariates, what architectural strategies they employed, and how they performed in practice. This thesis was therefore conceived to address this specific gap.

The primary objective of this study was twofold: first, to systematically map and categorize the landscape of existing FMs that support exogenous variable integration and second, to conduct a novel, comprehensive empirical benchmark to evaluate their practical performance. To this end, the work was guided by four central research questions:

- To identify which technical approaches exist for integrating exogenous variables into FMs (F1).
- To evaluate the extent of performance improvements from exogenous variables and benchmark the FMs against established models (F2).
- To analyze the models' sensitivity to different exogenous variables configurations (F3).
- To provide a comprehensive assessment of the strengths and weaknesses of the identified approaches based on theoretical and empirical evidence (F4).

7.2. Key Findings and Answering the Research Questions

This thesis set out to answer four central research questions. The empirical and theoretical investigations have yielded clear findings for each, which are summarized in detail below.

In response to F1, the study identified and categorized the diverse technical approaches FMs use to integrate exogenous variables, finding they fall into two dominant paradigms. The first is **Modular / Staged Integration**, which handles covariates in a separate component. This includes the simple but effective *external regressor* approach used by **Chronos** and **TimesFM**, as well as more sophisticated *adapter-based* modules like the *Exogenous Mixer Block* in **TTM** or the framework proposed by **ChronosX**. The second paradigm is **Unified / Deep Integration**, where covariates are processed homogenously with the target series, exemplified by the *Any-variate Attention* in **MOIRAI**. Within the near-zero-shot context of this study, the modular approaches proved to be the more practically reliable and effective strategy.

For F2, the key finding is that performance gains from exogenous variables are not guaranteed. The benefit is highly dependent not only on the **FM's architecture** but also on the overall **forecasting setup**, including the specific dataset, context window, and forecast horizon. While the modular approach of **Chronos** made it the clear favorite and most reliable model for leveraging covariates, it was not a universal victor, as no single model dominated all datasets. In the comparison against established methods, a well-suited FM like **Chronos** demonstrated superior performance. However, it is critical to note that this was in a benchmark setting that did not include extensive hyperparameter tuning for the traditional or DL baselines, which could have improved their standing.

The investigation into F3 demonstrated that covariate configuration is a complex, model-specific challenge. The study invalidated the "more data is not always better" assumption, as the indiscriminate inclusion of all features often degraded performance. For some models, adding permuted noise was statistically indistinguishable from adding curated features. Furthermore, models exhibited extreme sensitivity to the *type* of configuration. The 'lagged target' feature, for example, proved highly beneficial for **TTM** but significantly impaired the performance of **TimesFM**, likely due to architectural redundancies in the latter. This underscores that effective configuration requires a deep understanding of the specific FM's internal workings.

Finally, regarding F4, the study identified a clear trade-off profile for using FMs with covariates. The primary strengths are operational: significant gains in **computational speed** over traditional methods and a **simplified workflow** that reduces the burden of model selection and tuning. The main weaknesses, however, are substantial. These limitations include a profound **lack of interpretability**, significant model-specific **practicability issues** like prohibitively long fine-tuning times (**TTM**) or failures on certain data patterns like zero-value patches and a strong dependency on **future-known covariates** for the most successful modular integration methods, which limits their applicability in many real-world scenarios.

7.3. Contribution of the Study to Research

This thesis contributes to the field of TSF through a systematic, multi-faceted investigation into the integration of exogenous variables with FMs . The primary contributions can be summarized in four key areas: providing a structured theoretical overview, delivering a novel empirical benchmark and software framework, and identifying emergent insights into the behavior of FMs .

First, this work provides a **structured, systematic overview** of a nascent and previously fragmented research area. By surveying the landscape of existing models, this study introduces a clear taxonomy of covariate integration strategies, categorizing them into two dominant paradigms: **Modular / Staged Integration** and **Unified / Deep Integration**. This framework provides a common vocabulary and an analytical lens through which new and existing models can be more effectively understood, compared, and situated within the field, thereby establishing a foundational basis for future research.

Second, the thesis presents a **novel and extensive empirical benchmark** with a specific focus that was absent in prior studies. While other benchmarks for FMs have concentrated primarily on univariate performance, this work's central contribution is its dedicated and rigorous evaluation of the impact of exogenous variables. The study compares multiple FMs against a wide range of traditional and DL baselines across numerous, well-defined covariate scenarios. This provides the first concrete, comparative performance data on this specific problem, extending the empirical foundations of the field.

Furthermore, to facilitate this empirical work, this thesis produced a tangible and reusable **software framework as a key contribution**. This includes the complete benchmark codebase and a modular **Foundation Model Wrapper API**, designed to streamline the process of evaluating different FMs . This API, in particular, provides a valuable foundation for future research that extends beyond the specific focus of this thesis, offering a standardized way to use various FMs on diverse forecasting tasks. The framework will be made publicly available to accelerate further investigation and promote reproducible research in the field.

Finally, this research yields several **actionable insights and emergent findings** that can guide both practitioners and future researchers. It establishes that the most reliable current "best practice" in a near-zero-shot context is a modular approach. Beyond this, by uncovering surprising results—such as the "Chronos/TimesFM paradox," where an identical integration strategy produced opposite outcomes, the study contributes a deeper understanding of FM behavior. This reveals that the core pre-trained architecture is a more decisive factor in performance than the integration method itself, a critical insight that challenges common assumptions about these models.

7.4. Concluding Remarks

The integration of exogenous variables with FMs marks a key step forward of TSF. Real-world phenomena are rarely univariate. They are complex, interconnected systems driven by a multitude of external factors. By enabling models to process this rich contextual information, we move beyond simple extrapolation and take a fundamental step towards creating forecasting systems that possess a deeper, more

causal understanding of the processes they are modeling. This thesis has charted the initial landscape of this frontier, revealing its complexities, its challenges and its immense potential. The journey has shown that while no single, perfect solution yet exists, the field is in a vibrant and dynamic state of exploration. This abundance of different ideas and the rapid pace of innovation are not signs of a field in disarray, but rather indicators of one filled with opportunity. For researchers inspired to venture into this domain, the message is clear: the ground is fertile for new discoveries. Whether in designing novel architectures, creating richer datasets, or exploring new evaluation paradigms. Every contribution has the potential to make a significant impact. The path is open to build upon this work and help shape the next generation of truly context-aware and intelligent forecasting models.

List of Abbreviations

ACF Autocorrelation Function. 10, 12, 19, F

ADF Augmented Dickey-Fuller test. 10

AI Artificial Intelligence. 3, F

AIC Akaike Information Criterion. 20, 40

AP Adaptive Patching. 49, 50, 53, 54

API Application Programming Interface. 59, 62, 63, 93–96, 160

AR Autoregressive. 19, 36

ARIMA Autoregressive Integrated Moving Average. 2, 10, 18, 19, 36, 158

ARIMAX Autoregressive Integrated Moving Average with Exogenous Variables. 36, 39, 147

ARMA Autoregressive Moving Average. 19

AVA Any-variate Attention. 70, 71, 73, 74, 80, 147

BERT Bidirectional Encoder Representations from Transformers. 41, 69

BIC Bayesian Information Criterion. 20, 40

CFE Cumulative Forecast Error (sum of errors, measures bias). 32, 102, 153, 156

CI Channel-Independent. 70, 71

CLIP Contrastive Language-Image Pre-training. 41

CM Channel-Mixed. 70, 71

CNN Convolutional Neural Network. 22, 27, 42

CPU Central Processing Unit. 52, 57, 92–96, 113, 145, 151

CRPS Continuous Ranked Probability Score. 32, 33

CV Computer Vision. 2, 41, 44, 46

DL Deep Learning. 2, 7, 22, 39, 40, 92–94, 96, 104, 105, 133, 136, 145, 149, 151, 153–155, 158, 160

DRS Diverse Resolution Sampling. 50, 53, 54

EPF Electricity Price Forecasting. 77

ES Exponential Smoothing. 16, 19

ETS Error, Trend, Seasonality model. 19, 20, 95, 101

FFN Feed-Forward Network. 72, 81

FM Foundation Model. 2–5, 7, 41–45, 48, 49, 55, 60, 62, 63, 67, 74, 78, 84, 92–94, 101, 105, 113, 116, 118, 123, 133, 143–160, F

GBM Gradient Boosting Machines. 20–22, 37, 95

GLU Gated Linear Unit. 28

GPT Generative Pre-trained Transformer. 45, 63

GPT-3 Generative Pre-trained Transformer 3. 41

GPU Graphics Processing Unit. 52, 93, 94, 96, 118, 126, 128, 132, 145, 151

GRN Gated Residual Network. 28, 38, 82

GRU Gated Recurrent Unit. 22, 25, 37

IIB Input Injection Block. 81, 147

IV Instrumental Variables. 40

JAX Just After eXecution. 63, 65–67

KPSS Kwiatkowski-Phillips-Schmidt-Shin test. 10

LIME Local Interpretable Model-agnostic Explanations. 40

LLM Large Language Model. 45, 47, 54, 58, 60, 62

LOTSA Large-scale Open Time Series Archive. 67, 69, 70, 73, 74, 127

LR Linear Regression. 95, 109–111, G

LSTM Long Short-Term Memory. 2, 22, 24, 25, 28, 37–39, 79, 82

LSVR Localized Support Vector Regression. 21

MA Moving Average. 19, 36

MAE Mean Absolute Error. 30, 33, 73

MAPE Mean Absolute Percentage Error. 31

MASE Mean Absolute Scaled Error. 31, 33, 57, 98, 99, 101, 103, 105, 120, 121, 125, 130, 131, 133–136, 138–143, 156, H, I

ML Machine Learning. 2, 3, 7, 20, 22, 24, 37, 39, 40, 93, 95, 96, 105, 133, 150, 154, F

MLE Maximum Likelihood Estimation. 19, 20

MLP Multi-Layer Perceptron. 22–24, 38, 42, 145

MoE Mixture of Experts. 42, 67, 71–74, 94, 123, 126–128

MSE Mean Squared Error. 31, 51, 52, 69

MSIS Mean Scaled Interval Score. 33, 102

MSTL Multiple Seasonalities STL. 15

NBEATSx Neural Basis Expansion Architecture with Exogenous Variables. 93, 101, 149, 153

NID Normally and Independently Distributed. 20

NLP Natural Language Processing. 2, 41, 42, 44, 46, 63, 74

NN Neural Network. 39

OIB Output Injection Block. 81, 147

PACF Partial Autocorrelation Function. 10, 12, 19, F

PCA Principal Component Analysis. 40

PI Prediction Intervals. 40

PIS Periods In Stock. 32, 102, 153, 156

PL Pinball Loss. 32

RBF Radial Basis Function. 21

ReLU Rectified Linear Unit. 22

RF Random Forest. 20, 21, 37

RFE Recursive Feature Elimination. 40

RMSE Root Mean Squared Error. 31

RMSSE Root Mean Squared Scaled Error. 31, 33, 101

RNN Recurrent Neural Network. 2, 24, 25, 27, 37, 42, 79

RoPE Rotary Position Embedding. 76, 78, 80

RPT Resolution Prefix Tuning. 50, 51, 53, 54

SARIMA Seasonal Autoregressive Integrated Moving Average. 19, 95

SARIMAX Seasonal Autoregressive Integrated Moving Average with Exogenous Variables. 36, 95, 101, 149

SES Simple Exponential Smoothing. 16–18

SHAP SHapley Additive exPlanations. 40

SOTA State-of-the-Art. 3

SSM State Space Model. 19, 20, 37, 42

STL Seasonal and Trend decomposition using Loess. 15

STR Seasonal and Trend decomposition using Regression. 15

STSM Structural Time Series Model. 37

SVM Support Vector Machine. 21

SVR Support Vector Regression. 20, 21, 37

T5 Text-to-Text Transfer Transformer. 55, 56, 58

TCN Temporal Convolutional Network. 37

TF Transfer Function. 36

TFT Temporal Fusion Transformers. 27, 28, 38, 82, 83, 93, 151, 154, 155

Theta Theta Model. 95, 101

TiDE Time-series Dense Encoder. 93, 101, 149, 153

TSF Time Series Forecasting. 2–8, 15, 20–23, 25, 27, 28, 33, 41, 43–45, 48, 49, 54, 59, 62, 63, 67, 68, 74, 78, 84, 87, 108, 146, 147, 152, 156, 158, 160, F

TTM Tiny Time Mixers. 48–54, 81, 83, 94, 128–133, 135, 136, 138, 139, 141, 144, 145, 147, 148, 150–152, 156, 159, F, H

VARX Vector Autoregressive Model with Exogenous Variables. 36

VIF Variance Inflation Factor. 40

VRAM Video Random Access Memory. 118, 126–128, 132, 151, 154

WQL Weighted Quantile Loss. 32, 33, 57, 102

XAI Explainable AI. 40

List of Tables

List of Figures

List of Equations

References

Hirotugu Akaike. 1974. "A new look at the statistical model identification". *IEEE Transactions on Automatic Control*, 19, 6, 716–723.

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Oct. 2024. "GIFT-eval: A benchmark for General Time Series Forecasting Model Evaluation". *arXiv [cs.LG]*, (Oct. 2024).

R Allard. 1998. "Use of time-series analysis in infectious disease surveillance". *Bull. World Health Organ.*, 76, 4, 327–333.

Amazon Science. N.d.(a). *Chronos Forecasting: Official Code Repository*. `https://github.com/amazon-science/chronos-forecasting`. Accessed: 2025-6-7. ().

Amazon Science. N.d.(b). *Chronos-Bolt-Base Model (AutoGluon integration)*. `https://huggingface.co/autogluon/chronos-bolt-base`. Accessed: 2025-6-7. ().

Abdul Fatir Ansari et al.. Mar. 2024. "Chronos: Learning the language of time series". *arXiv [cs.LG]*, (Mar. 2024).

Sebastian Pineda Arango et al.. Mar. 2025. "ChronosX: Adapting pretrained time series models with exogenous variables". *arXiv [cs.LG]*, (Mar. 2025).

Vassilios Assimakopoulos and Konstantinos Nikolopoulos. 2000. "The Theta model: an insight into the statistical model and its performance". *Int. J. Forecast.*, 16, 4, 521–527.

André Bauer, Marwin Züfle, Simon Eismann, Johannes Grohmann, Nikolas Herbst, and Samuel Kounev. Apr. 2021. "Libra: A benchmark for time series forecasting methods". In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. ACM, New York, NY, USA, (Apr. 2021).

Christoph Bergmeir. 2020. *Forecasting for Data Scientists*. (2020).

Alexander Borovykh, Yura Boers, Cornelis Salazar, Maarten de Rijke, and Aaron van den Oord. 2017. "Dilated Recurrent Neural Networks for Time Series Forecasting". *arXiv preprint arXiv:1710.09913*.

George E P Box, Gwilym M Jenkins, and Gregory C Reinsel. 1976. *Time Series Analysis, Forecasting and and Control*. (Third ed.). Series G. Holden-Day, Oakland, CA.

Leo Breiman. 2001. "Random Forests". *Machine Learning*, 45, 1, 5–32.

Lorenzo Brigato, Rafael Morand, Knut Strømmen, Maria Panagiotou, Markus Schmidt, and Stavroula Mougiakakou. Feb. 2025. "Position: There are no Champions in Long-Term Time Series Forecasting". *arXiv [cs.LG]*, (Feb. 2025).

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J Kaplan, et al.. May 2020. "Language Models are Few-Shot Learners". *Neural Inf Process Syst*, abs/2005.14165, (May 2020), 1877–1901.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al.. 2020. "Language Models are Few-Shot Learners". *Advances in Neural Information Processing Systems*, 33, 1877–1901.

Chris Chatfield. Sept. 2005. "Time-series forecasting". en. *Signif. (Oxf.)*, 2, 3, (Sept. 2005), 131–133.

Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. Aug. 2024. "VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters". *arXiv [cs.CV]*, (Aug. 2024).

Tianqi Chen and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System". *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". *arXiv preprint arXiv:1406.1078*.

Robert B Cleveland, William S Cleveland, Jean E McRae, and Ina J Terpenning. 1990. "STL: A Seasonal-Trend Decomposition Procedure Based on Loess". *Journal of Official Statistics*, 6, 1, 3–73.

Ben Cohen et al.. May 2025. "This time is different: An observability perspective on time series foundation models". *arXiv [cs.LG]*, (May 2025).

John D Croston. 1972. "Forecasting and stock control for intermittent demands". *Operational Research Quarterly*, 23, 3, 289–303.

Jonathan D Cryer. Jan. 1986. *Time Series Analysis*. PWS, London, OH, (Jan. 1986), 286.

Pamela Danese and Matteo Kalchschmidt. May 2011. "The role of the forecasting process in improving forecast accuracy and operational performance". en. *Int. J. Prod. Econ.*, 131, 1, (May 2011), 204–214.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Apr. 2023. "Long-term forecasting with TiDE: Time-series Dense Encoder". *arXiv [stat.ML]*, (Apr. 2023).

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. Oct. 2023. "A decoder-only foundation model for time-series forecasting". *arXiv [cs.CL]*, (Oct. 2023).

Jan G De Gooijer and Rob J Hyndman. Jan. 2006. "25 years of time series forecasting". en. *Int. J. Forecast.*, 22, 3, (Jan. 2006), 443–473.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

D Dickey and W Fuller. June 1979. "Distribution of the estimators for autoregressive time series with a unit root". *Journal of the American Statistical Association*, 74, 366a, (June 1979), 427–431.

Alexander Dokumentov. 2020. "Seasonal and Trend decomposition using Regression (STR)". *SSRN Electronic Journal*.

Anna G Dorogush, Vasily Ershov, and Andrey Gulin. 2018. "CatBoost: gradient boosting with categorical features support". In: *NIPS 2018 Conference on Systems and Machine Learning*.

Alexey Dosovitskiy et al.. Oct. 2020. "An image is worth 16x16 words: Transformers for image recognition at scale". *Int Conf Learn Represent*, abs/2010.11929, (Oct. 2020).

James Durbin and Siem Jan Koopman. May 2012. *Time series analysis by state space methods: Second edition*. en. (2nd ed.). Oxford Statistical Science Series. Oxford University Press, London, England, (May 2012).

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H Nguyen, Wesley M Gifford, Chandra Reddy, and Jayant Kalagnanam. Jan. 2024. "Tiny Time Mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series". *arXiv [cs.LG]*, (Jan. 2024), 74147–74181. Ed. by A Globerson, L Mackey, D Belgrave, A Fan, U Paquet, J Tomczak, and C Zhang.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. June 2023. "TSMixer: Lightweight MLP-Mixer model for multivariate time series forecasting". *arXiv [cs.LG]*, (June 2023).

Y Eldar, M Lindenbaum, M Porat, and Y Y Zeevi. 1997. "The farthest point strategy for progressive image sampling". en. *IEEE Trans. Image Process.*, 6, 9, 1305–1315.

Ronald A Fisher. 1922. "On the mathematical foundations of theoretical statistics". *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222, 309–368.

Jerome H Friedman. 2001. "Greedy Function Approximation: A Gradient Boosting Machine". *The Annals of Statistics*, 29, 5, 1189–1232.

Milton Friedman. 1937. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". *Journal of the American Statistical Association*, 32, 200, 675–701.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Oct. 2023. "TimeGPT-1". *arXiv [cs.LG]*, (Oct. 2023).

Jan Gasthaus et al.. 2019. "Probabilistic forecasting with DeepAR: Theory and application". In: *International Conference on Machine Learning (ICML) Workshops*.

Aurélien Géron. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Tilmann Gneiting and Adrian E Raftery. 2007. "Strictly Proper Scoring Rules, Prediction, and Estimation". *Journal of the American Statistical Association*, 102, 477, 359–378.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
Google Research. N.d. *TimesFM: Official Code Repository*. `https://github.com/google-research/timesfm`. Accessed: 2025-6-7. ().
Albert Gu and Tri Dao. 2023. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". *arXiv preprint arXiv:2312. 00752*.
Yannik Hahn, Tristan Langer, Richard Meyes, and Tobias Meisen. Mar. 2023. "Time series dataset survey for forecasting with deep learning". en. *Forecasting*, 5, 1, (Mar. 2023), 315–335.
James D Hamilton. 1994. *Time series analysis*. Princeton university press.
Trevor Hastie, Robert Tibshirani, and Jerome Friedman. July 2003. *The elements of statistical learning: Data mining, inference, and prediction*. en. (1st ed.). Springer series in statistics. Vol. 2. Springer, New York, NY, (July 2003).
Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33, 6840–6851.
Sepp Hochreiter and Jürgen Schmidhuber. 1997. "Long Short-Term Memory". *Neural Computation*, 9, 8, 1735–1780.
Charles C Holt. 1957. "Forecasting trends and seasonals by exponentially weighted moving averages". *Office of Naval Research Memorandum No. 52*.
Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. Jan. 2025. "From tables to time: How TabPFN-v2 outperforms specialized time series forecasting models". *arXiv [cs.LG]*, (Jan. 2025).
Rob J Hyndman and George Athanasopoulos. 2021. *Forecasting: Principles and Practice*. (3rd ed.). OTexts.
Rob J Hyndman and B A Billah. 2003. "The theta model for forecasting". *J. R. Stat. Soc. Ser. C. Appl. Stat.*, 52, 1, 89–99.
Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. 2008. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Science & Business Media.
Rob J Hyndman, Earo Wang, and Nikolay Laptev. Nov. 2015. "Large-scale unusual time series detection". In: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, (Nov. 2015).
IBM Granite Team. 2025. *TTM Model Path yaml File*. `https://github.com/ibm-granite/granite-tsfm/blob/077331f5c6e297e13db097af7249dd7887c4bfca/tsfm_public/resources/model_paths_config/ttm.yaml`. Accessed: 2025-6-7. (2025).
R E Kalman. 1960. "A new approach to linear filtering and prediction problems". *Journal of Basic Engineering*, 82, 1, 35–45.
Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Vol. 30, 3146–3154.
N Khan, M T Islam, M S Talukdar, and S Sarowar. 2021. "Predicting COVID-19 spread using simple time-series statistical models". *KAUST Repository*.
Bonggun Kim, Jong-Hyun Lim, Sunghun Kang, Dong-Jun Choi, Seungki Shin, Jae-Hwan Lee, and Hwalsuk Kim. 2024. "Are Transformers Effective for Time Series Forecasting?" *arXiv preprint arXiv:2402. 13324*.
Jongseon Kim, Hyungjoon Kim, Hyungi Kim, Dongjun Lee, and Sungroh Yoon. Oct. 2024. "A comprehensive survey of deep learning for time series forecasting: Architectural diversity and open challenges". *arXiv [cs.LG]*, (Oct. 2024).
Siva Rama Krishna Kottapalli, Karthik Hubli, Sandeep Chandrashekhara, Garima Jain, Sunayana Hubli, Gayathri Botla, and Ramesh Doddaiah. Apr. 2025. "Foundation models for time series: A survey". *arXiv [cs.LG]*, (Apr. 2025).
D Kwiatkowski, P Phillips, P Schmidt, and Y Shin. Oct. 1992. "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?" *Journal of Econometrics*, 54, 1-3, (Oct. 1992), 159–178.
Zhe Li et al.. Oct. 2024. "FoundTS: Comprehensive and unified benchmarking of foundation models for time Series Forecasting". *arXiv [cs.LG]*, (Oct. 2024).
Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Oct. 2021. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting". en. *Int. J. Forecast.*, 37, 4, (Oct. 2021), 1748–1764.
Bryan Lim and Stefan Zohren. Apr. 2021. "Time-series forecasting with deep learning: a survey". en. *Philos. Trans. A Math. Phys. Eng. Sci.*, 379, 2194, (Apr. 2021), 20200209.

Rafael Diniz Toscano de Lima, Sergio Murilo Maciel Fernandes, and Sidney Marlon Lopes de Lima. Jan. 2025. "Time series forecasting with exogenous variables: a literature review to identify promising gaps in computational research". *Rev. Principia - Divulg. Cient. Tecnol. IFPB*, 62, (Jan. 2025).

Xu Liu et al.. Oct. 2024. "Moirai-MoE: Empowering time series foundation models with sparse mixture of experts". *arXiv [cs.LG]*, (Oct. 2024).

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Oct. 2024. "Timer-XL: Long-context Transformers for unified time series forecasting". *arXiv [cs.LG]*, (Oct. 2024).

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Feb. 2025. "Sundial: A family of highly capable time series foundation models". *arXiv [cs.LG]*, (Feb. 2025).

Yongliang Liu, Guokun Zhao, Song Jin, Yuyang Wang, Yuhong Liu, and Zichao Liu. 2024. "iTransformer: Inverted Transformers Are Everything You Need for Time Series Forecasting". *International Conference on Learning Representations (ICLR)*.

Greta M Ljung and George E P Box. 1978. "On a modification of the Box–Pierce test statistic". *Biometrika*, 65, 2, 297–303.

Helmut Lutkepohl. Dec. 2005. *New introduction to multiple time series analysis*. en. Springer, Berlin, Germany, (Dec. 2005).

Spyros Makridakis and Michèle Hibon. 2000. "The M3-Competition: Results, conclusions and implications". *International Journal of Forecasting*, 16, 4, 451–476.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2022. "The M5 accuracy competition: Results, findings and conclusions". *International Journal of Forecasting*, 38, 4, 1323–1345.

Peter Nemenyi. 1963. "Distribution-free multiple comparisons". PhD thesis. Princeton University.

Yuqi Nie et al.. 2023. "PatchTST: A Time Series Is Worth 16 Words: A Revolutionizing Approach to Time Series Forecasting with Transformers". *arXiv preprint arXiv:2303. 04639*.

Kanghui Ning et al.. Mar. 2025. "TS-RAG: Retrieval-Augmented Generation based Time Series Foundation Models are Stronger Zero-Shot Forecaster". *arXiv [cs.LG]*, (Mar. 2025).

Jakub Nowotarski and Rafał Weron. Jan. 2018. "Recent advances in electricity price forecasting: A review of probabilistic forecasting". en. *Renew. Sustain. Energy Rev.*, 81, (Jan. 2018), 1548–1568.

Kin G Olivares, Cristian Challu, Grzegorz Marcjasz, Rafał Weron, and Artur Dubrawski. Apr. 2023. "Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx". en. *Int. J. Forecast.*, 39, 2, (Apr. 2023), 884–900.

Kin G Olivares, David Luo, Cristian Challu, Stefania La Vattiata, Max Mergenthaler, and Artur Dubrawski. 2023. "HINT: Hierarchical mixture networks for coherent probabilistic forecasting". *arXiv preprint arXiv:2303. 00392*.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals Oriol, Alex Graves, Nal Kalchbrenner, Max Welling, and Koray Kavukcuoglu. 2016. "WaveNet: A Generative Model for Raw Audio". *arXiv preprint arXiv:1609. 03499*.

Boris N Oreshkin, Omar Aldabass, Leonardo Parmezani, Cláudio P de Andrade, Jorge de Oliveira, and Mikolaj Binkowski. 2020. "N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting". *International Conference on Learning Representations (ICLR)*.

Willa Potosnak, Cristian Challu, Mononito Goswami, Kin G Olivares, Michał Wiliński, Nina Żukowska, and Artur Dubrawski. Feb. 2025. "Investigating compositional reasoning in time series foundation models". *arXiv [cs.LG]*, (Feb. 2025).

Xiangfei Qiu et al.. Mar. 2024. "TFB: Towards comprehensive and fair benchmarking of Time Series Forecasting methods". *arXiv [cs.LG]*, (Mar. 2024).

Alec Radford, Jong Wook Kim, Chris Chen, Ilya Sutskever, Kim Alec, Chen Jong Wook, Sutskever Chris, and Ilya. 2021. "Learning Transferable Visual Models From Natural Language Supervision". *Proceedings of the 38th International Conference on Machine Learning*.

Colin Raffel, Noam M Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Oct. 2019. "Exploring the limits of transfer learning with a unified text-to-text transformer". *J. Mach. Learn. Res.*, 21, 140, (Oct. 2019), 140:1–140:67.

Aditya Ramesh et al.. 2021. "Zero-Shot Text-to-Image Generation". *International Conference on Machine Learning*.

Kashif Rasul et al.. Oct. 2023. "Lag-Llama: Towards foundation models for probabilistic time series forecasting". *arXiv [cs.LG]*, (Oct. 2023).

Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. Dec. 2023. "TimeChat: A time-sensitive multimodal large language model for long video understanding". *arXiv [cs.CV]*, (Dec. 2023).

Tim Salimans, Xihui Chen, Diederik P Kingma, Prafulla Dhariwal, David Sontag, and Jonathan Ho. 2020. "TimeGrad: A Unified Framework for Probabilistic Time Series Forecasting". *arXiv preprint arXiv:2002. 04611*.

Deepika Saxena and Ashutosh Kumar Singh. June 2021. "Workload forecasting and resource management models based on machine learning for cloud computing environments". *arXiv [cs.DC]*, (June 2021).

Bernhard Schölkopf and Alexander J Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Gideon Schwarz. 1978. "Estimating the dimension of a model". *The Annals of Statistics*, 6, 2, 461–464.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V Le, Geoffrey Hinton, and Jeff Dean. 2017. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". *arXiv preprint arXiv:1701. 06538*.

Robert H Shumway and David S Stoffer. 2017. *Time Series Analysis and Its Applications: With R Examples*. Springer.

Aris A Syntetos and John E Boylan. 2005. "The accuracy of intermittent demand estimates". *International Journal of Forecasting*, 21, 2, 303–314.

Aris A Syntetos, John E Boylan, and John D Croston. 2010. "An assessment of the impact of promotions on the demand for slow-moving items: A case study from a manufacturing company". *J. Oper. Res. Soc.*, 61, 4, 616–624.

Jan Tinbergen. 1939. *Statistical Testing of Business-Cycle Theories: Volume I. A Method and its Application to Investment Activity*. League of Nations, Geneva.

Vladimir N Vapnik, Steven E Golowich, and Alex J Smola. 1997. "Support vector method for function approximation, regression estimation, and signal processing". *Advances in neural information processing systems*, 9, 281–287.

Ashish Vaswani, Noam M Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. June 2017. "Attention is All you Need". *Neural Inf Process Syst*, 30, (June 2017), 5998–6008.

Anders Wallström and Patrik Jonsson. 2009. "The forecasting performance of intermittent demand estimators: An empirical study". *International Journal of Production Economics*, 118, 2, 487–494.

Lin Wang, Yi Zhang, and Jinxing Wang. 2012. "Feature engineering for time series forecasting". *J. Commun. Inf. Netw.*, 1, 1, XX–YY.

Xue Wang, Tian Zhou, and Jinyang Gao Bolin Ding Jingren Zhou. 2024. "Output Scaling: YING-LONG Delayed Chain of Thought in a Large Pretrained Time Series Forecasting Model".

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. July 2024. "Deep time series models: A comprehensive survey and benchmark". *arXiv [cs.LG]*, (July 2024).

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Feb. 2024. "TimeXer: Empowering transformers for time series forecasting with exogenous variables". *arXiv [cs.LG]*, (Feb. 2024).

Peter R Winters. 1960. "Forecasting sales by exponentially weighted moving averages". *Management Science*, 6, 3, 324–342.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. "Unified training of universal time series forecasting transformers". Research Collection School Of Computing and Information Systems 235, 53140.

Wenfa Wu, Guanyu Zhang, Zheng Tan, Yi Wang, and Hongsheng Qi. May 2025. "Dual-Forecaster: A multimodal time series model integrating descriptive and predictive texts". *arXiv [cs.LG]*, (May 2025).

Yuqi Zhang et al.. 2023. "Crossformer: Scalable Transformer for Learning Long-Range Correlations in Time Series". *International Conference on Machine Learning (ICML)*.

Juncheng Zheng, Yizhuo Chen, Haoyi Zhou, Huaimin Chen, Chunming Gu, Fan Yan, and Jielin Li. 2024. "FutureTST: Towards robust long-term time series forecasting with future information". *arXiv preprint arXiv:2403. 04500*.

A. Dataset Presentation

The following table, Table A.1, provides a comprehensive summary of the time series datasets utilized in this study. For each dataset, the table outlines its fundamental properties, including its domain, temporal granularity, length, and the total number of individual time series.

Furthermore, it details attributes relevant for forecasting, such as the number of target variables and the breakdown of available covariates into past, future and categorical types. Finally, the last two columns specify which datasets were included in the evaluation process and the precise number of series selected from each for analysis.

Dataset Descriptions

Housing

This dataset explores supply and demand factors influencing US home prices, including metrics like housing permits, mortgage rates, construction spending, and consumer sentiment.
Source: `https://www.kaggle.com/datasets/utkarshx27/factors-influence-house-price-in-us/data`

Electricity

This dataset examines hourly electricity demand and pricing in Victoria, Australia, from 2015 to 2020. It includes related features such as weather conditions, rainfall, and indicators for school days and holidays.
Source: `https://www.kaggle.com/datasets/aramacus/electricity-demand-in-victoria-australia`

Bike Sharing

This dataset contains hourly bike rental data from Seoul, South Korea. It includes environmental factors like temperature, humidity, wind speed, and solar radiation, as well as seasonal and operational indicators, to analyze rental demand patterns.
Source: `https://www.kaggle.com/datasets/saurabhshahane/seoul-bike-sharing-demand-prediction`

Energy

This dataset provides four years of hourly data on Spain's electricity system, covering consumption, pricing, and detailed generation data across various sources, including fossil fuels, nuclear, and multiple types of renewables.

Source: https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather

Bitcoin

This dataset aggregates daily time series that are potential influencers of the Bitcoin price, including on-chain metrics like hash rate and block size, alongside public sentiment data from Google Trends and tweets.
Source: https://zenodo.org/records/5122101

KDD Cup Air Quality

This dataset records hourly air quality measurements (PM2.5, PM10, NO2, CO, O3, SO2) from 35 stations in Beijing. It is designed for air quality analysis and forecasting tasks.
Source: https://zenodo.org/records/4656756

ETT Electricity Transformer Temperature

This dataset tracks electricity usage and the oil temperature of an electrical transformer over two years. It is intended for predictive maintenance and improving the safety and efficiency of power transformers.
Source: https://github.com/zhouhaoyi/ETDataset

Household Power Consumption

This dataset contains high-frequency (minute-by-minute) electricity consumption data from a single household over 47 months. It includes total power usage as well as sub-metered data for different areas of the home.
Source: https://github.com/jbrownlee/Datasets/tree/master

Stack Overflow

This dataset provides the monthly count of questions on Stack Overflow for a wide range of programming libraries, frameworks, and data science topics, allowing for trend analysis of developer interest over time.
Source: https://www.kaggle.com/datasets/aishu200023/stackindex

COVID-19

This dataset contains daily, country-level data on the COVID-19 pandemic, tracking key metrics such as the cumulative number of confirmed cases, deaths, recoveries, and active cases.
Source: https://www.kaggle.com/datasets/imdevskp/corona-virus-report/data

Weather

This dataset contains hourly climate data from Monash University, Australia, spanning over 11 years. It includes key meteorological variables such as temperature, humidity, pressure, radiation, and wind speed.
Source: `https://zenodo.org/records/5184708`

COVID-19 Mobility

This dataset tracks daily percentage changes in visitor numbers to various location categories (e.g., retail, workplaces, parks) across 131 regions, reflecting the impact of the COVID-19 pandemic on human mobility.
Source: `https://zenodo.org/records/4663809`

Illness

This dataset provides weekly statistics on influenza-like illness (ILI) in the United States. It includes weighted and unweighted percentages of ILI-related patient visits, along with case counts for different age groups.
Source: `https://github.com/thuml/Time-Series-Library`

Madrid Air Quality and Traffic

This dataset combines air quality, meteorological, and traffic data for the city of Madrid. It contains features such as pollutant concentrations (NO2), weather conditions, and traffic flow metrics like speed and intensity.
Source: `https://zenodo.org/records/7308425`

Rideshare

This dataset contains hourly time series for Uber and Lyft rideshare services in New York City between November and December 2018. It includes various attributes such as price, distance, and surge multipliers, along with weather conditions.
Source: `https://zenodo.org/records/5122232`

ACEA Water Datasets

The following nine datasets are from the ACEA Smart Water Prediction competition and represent different Italian waterbodies. The goal is to forecast water availability to help preserve these resources.
Source: `https://www.kaggle.com/competitions/acea-water-prediction`

Auser

Represents an aquifer with two subsystems (a water table aquifer and a confined artesian aquifer), with levels monitored through various wells.

Petrignano

Represents an aquifer fed by three underground systems and the Chiascio river, influenced by rainfall, temperature, and drainage.

Doganella

Represents an aquifer system fed by rainfall, consisting of an upper water table and a lower artesian layer, monitored via nine different wells.

Luco

Represents an aquifer fed by meteoric infiltration and is accessed by three wells.

Amiata

Represents a volcanic aquifer fed by meteoric infiltration, with water levels accessed through four different water springs.

Lupa

Represents a water spring in the Rosciano Valley that provides drinking water to the city of Terni.

Arno

Represents the Arno river, the main waterway in Tuscany and a primary water source for the Florence metropolitan area.

Bilancino

Represents the Bilancino artificial lake, which is used as a reservoir to refill the Arno river during dry summer months.

Electricity Price

This dataset focuses on the German EPEX Day Ahead electricity market, providing hourly electricity prices. It is enriched with covariates designed for forecasting, including holiday indicators, cyclically encoded time features, and lagged electricity generation data.
Source: Sourced from public ENTSOE and DWD data as part of a master's thesis.

Wind

This dataset provides data for forecasting onshore wind power generation in Germany. Features include lagged and forecasted wind generation, installed energy capacities, and clustered weather data.
Source: Sourced from public ENTSOE and DWD data as part of a master's thesis.

M5 Demand

This dataset, from the M5 Forecasting Competition, contains hierarchical daily sales data for thousands of products sold at Walmart stores in the US. It includes prices, promotions, and event indicators to support detailed sales forecasting.
Source: `https://www.kaggle.com/competitions/m5-forecasting-accuracy/data`

Demo Demand

This is an internal dataset from prognostica used for demonstrating demand forecasting capabilities.
Source: Internal dataset.

Table A.1.: A Summary and Evaluation Inclusion Status of Time Series Datasets

Dataset	**Domain**	**Granularity**	**Length**	**# TS**	**# Targets**	**# Covariates**			**In Evaluation**	**# Eval. Series**
						Past	**Future**	**Cat.**		
housing	economic	QS-OCT	74	1	2	8	0	0	Yes	2
electricity	energy	D	2106	1	2	9	2	2	Yes	2
bike_sharing	human	h	8760	1	1	8	3	3	Yes	1
energy	energy	h	30233	1	2	20	4	0	Yes	2
bitcoin	economic	D	2659	1	1	17	0	0	Yes	1
air_quality	nature	h	10898	35	4	2	0	0	Yes	32
etth	energy	h	17420	2	1	6	0	0	Yes	2
household_power_consumption	energy	h	16758	1	2	5	0	0	Yes	2
stackoverflow	human	MS	132	1	30	50	0	0	No	0
covid	human	D	188	187	2	2	0	0	No	0
weather	nature	h	100057	1	2	6	0	0	Yes	2
covid_mobility	human	D	413	42	1	5	0	0	No	0
illness	human	W-TUE	966	1	3	4	0	0	Yes	3
madrid19	nature	h	4344	24	1	18	0	8	Yes	5
madrid22	nature	h	4343	24	1	18	0	8	Yes	5
rideshare	human	h	541	144	3	12	0	0	Yes	5
electricity_price	energy	h	50472	1	1	14	3	3	Yes	1
wind	energy	h	50424	1	1	19	1	0	Yes	1
auser	nature	D	3336	1	5	21	0	0	Yes	5
doganella	nature	D	540	1	9	12	0	0	No	0
luco	nature	D	404	1	4	17	0	0	No	0
petrignano	nature	D	4199	1	2	5	0	0	No	0
bilancino	nature	D	6025	1	1	7	0	0	Yes	1
arno	nature	D	1283	1	1	15	0	0	No	0
amiata	nature	D	1618	1	4	11	0	0	Yes	4
lupa	nature	D	4150	1	1	1	0	0	No	0
m5_other	demand	D	1941	1	123	0	4	4	Yes	27
m5_id	demand	D	1941	185	1	0	5	4	Yes	14
demo	demand	MS	120	1	72	7	0	0	No	0

B. Time Series Features

This appendix provides a brief description of the features calculated to characterize each time series. The features are derived from the `tsfeatures` package and its underlying methodologies.

series_length The total number of observations in the time series.

entropy The spectral entropy of the time series. This measures the forecastability of the series, where a value close to zero indicates a highly regular, predictable series, and a value close to one indicates a more random and less predictable series.

lumpiness Measures the variance of variances calculated from tiled (non-overlapping) windows of the series. It is a measure of changing volatility.

stability Measures the variance of means calculated from tiled (non-overlapping) windows of the series. It is a measure of changing level.

hurst The Hurst coefficient, which indicates the level of fractional differencing and measures the long-term memory of a time series. A value of 0.5 suggests a random walk.

trend The strength of the trend in the time series, calculated from an STL decomposition. A value close to 1 indicates a strong trend.

spike Measures the prevalence and magnitude of outliers or "spikes" in the series. This feature is not explicitly defined in the reference documentation but is included for its utility in identifying anomalous data points.

linearity A measure of the linearity of the time series, based on the nonlinearity test by Lee, White, and Granger. Values close to 0 indicate a linear series.

curvature A measure of the nonlinearity of the time series, derived from the same test as the `linearity` feature.

e_acf1 The first autocorrelation coefficient of the remainder (error) series after an STL decomposition. This indicates the amount of remaining short-term dependence once the trend and seasonality are removed.

seas_acf1 The first autocorrelation coefficient at the primary seasonal lag.

diff1_acf1 The first autocorrelation coefficient of the first-differenced series.

diff2_acf1 The first autocorrelation coefficient of the second-differenced series.

diff1_acf10 The sum of the first ten squared autocorrelation coefficients of the first-differenced series.

diff2_acf10 The sum of the first ten squared autocorrelation coefficients of the second-differenced series.

seas_pacf The partial autocorrelation coefficient at the first seasonal lag.

diff1x_pacf5 The sum of the first five squared partial autocorrelation coefficients of the first-differenced series.

diff2x_pacf5 The sum of the first five squared partial autocorrelation coefficients of the second-differenced series.

nonlinearity A statistic measuring the degree of nonlinearity in a time series, based on the test by Lee, White, and Granger.

unitroot_kpss The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test statistic for stationarity.

unitroot_pp The Phillips-Perron (PP) unit root test statistic.

alpha, beta The smoothing parameters for the level and trend, respectively, from Holt's linear trend method. These values are estimated from the data.