# TempusBench: An Evaluation Framework for Time-Series Forecasting

## Anonymous Author(s)

#### **Abstract**

Foundation models have transformed natural language processing and computer vision, and a rapidly growing literature on time-series foundation models (TSFMs) seeks to replicate this success in forecasting. While recent open-source models demonstrate the promise of TSFMs, the field lacks a comprehensive and community-accepted model evaluation framework. We see at least four major issues impeding progress on the development of such a framework. First, current evaluation frameworks consist of benchmark forecasting tasks derived from often outdated datasets (e.g., M3), many of which lack clear metadata and overlap with the corpora used to pre-train TSFMs. Second, existing frameworks evaluate models along a narrowly defined set of benchmark forecasting tasks such as forecast horizon length or domain, but overlook core statistical properties such as non-stationarity and seasonality. Third, domain-specific models (e.g., XGBoost) are often compared unfairly, as existing frameworks neglect a systematic and consistent hyperparameter tuning convention for all models. Fourth, visualization tools for interpreting comparative performance are lacking. To address these issues, we introduce TempusBench, an open-source evaluation framework. TempusBench consists of 1) new datasets which are not included in existing TSFM pretraining corpora, 2) a set of novel benchmark tasks that go beyond existing ones, and 3) a model evaluation pipeline with a standardized hyperparameter tuning protocol, and 4) a tensorboardbased visualization interface. We provide access to our code on GitHub: https: //anonymous.4open.science/r/benchmark-0634/README.md.

## 1 Introduction

2

3

5

6

8

9

10

11

12

13

14

15

16

17

18

19

20

21

33

34

37

The success of foundation models (i.e., models trained on large and diverse datasets that can be 23 used to solve downstream tasks) in natural language processing (NLP) and computer vision has 24 inspired an emerging literature on time-series foundation models. Time-series foundation models (TSFMs) are models that take past time-series data (and possibly covariate time-series data) as input 26 and output future values (or distributions over them), typically formulated as neural networks trained 27 via supervised learning. While about a dozen open-source TSFMs are now available, comparing their 28 performance to one another and to traditional domain-specific models (e.g., ARIMA [1], SVR [2, 3]) 29 remains difficult. A handful of evaluation frameworks have been released, but the field still lacks 30 comprehensive, community-accepted standards for model evaluation [4], creating an impediment for 31 the replication of the success of foundation models in NLP and computer vision [5].

We see four major challenges facing existing evaluation frameworks. First, the evaluation ecosystem relies on outdated datasets such as M3 [6] and M4 [7], many lacking metadata (e.g., variable names). More importantly, the existing evaluation datasets overlap with the pretraining corpora of TSFMs, leading to inflated estimates of zero-shot generalization [8]. For example, except for Moirai2, all TSFMs assessed by GIFT-Eval include test data in their training corpus [9, 10]. Second, current frameworks define benchmark forecasting tasks only along narrow axes (i.e., forecast

horizon, variate type, frequency, and domain). While useful, these miss key statistical properties long studied in time-series analysis such as (non-)stationarity, and seasonality. Without evaluation 40 across such properties, it seems unlikely that frameworks can yield generalizable conclusions about 41 model capabilities. Third, existing frameworks have not yet developed standardized hyperparameter 42 tuning routines, leading to comparisons made between TSFMs and domain-specific models to be 43 unfair as the performance of domain-specific models depend heavily on hyperparameter choice. 44 Indeed, as noted by practitioners [11], simple statistical models with well-chosen hyperparameters 45 can outperform more complex ones, highlighting the need for consistent tuning routines. Fourth, currently, evaluation typically reduces to numerical metrics such as mean squared error, which 47 practitioners remark [12] provide limited interpretability. For instance, under GIFT-Eval, seasonal 48 naive outperforms five open-source TSFMs, but this offers no insight into the strength and weaknesses 49 of TSFMs, since seasonal naive fails when seasonality is weak. Beyond quantitative scores, qualitative 50 analyses—especially forecast visualizations—are essential. 51

To address these issues, we introduce TempusBench, an open-source evaluation framework. Tempus-Bench consists of 1) new datasets which are not included in existing TSFM pretraining corpora, 2) a set of novel benchmark tasks that goes beyond existing ones, , and 3) a model evaluation pipeline with a standardized hyperparameter tuning protocol, and 4) tensorboard-based visualization interface.

#### 56 1.1 Contributions

57

58

59

60

61

63

64

65

66

67

TempusBench, going beyond TSFMs, includes 20 forecasting models, a number of which such as XGBoost, have previously not been considered by evaluation frameworks, and overcomes the aforementioned four issues by improving along the following dimensions. First, we introduce new time-series datasets which do not come from existing time-series evaluation datasets, and which are not contained in the training corpus of open-source TSFMs released to date. Second, we propose new benchmark task types that extend beyond horizon length, variate type, frequency, and domain. These include categories based on stationarity, seasonality, variable type (continuous, count, binary, categorical), sparsity (sparse vs. dense), dataset size (small vs. large), and quality (noisy vs. measurement error). Third, we introduce a model evaluation pipeline which runs a standardized and automated hyperparameter selection procedure for all forecasting models with hyperparameters, allowing a fair comparison of all forecasting methods. Fourth, TempusBench comes packages with a tensorboard-based visualization application which easily allows researchers and practitioners to visualize and interpret the performance of various models on different task types.

Table 1: Property comparisons of various forecasting benchmarks.

| Property               | Monash [13 | TFB [14]  | LTSF [15] | BasicTS+ [16 | ] ProbTS [17] | GIFT-Eval [9] | TempusBench    |
|------------------------|------------|-----------|-----------|--------------|---------------|---------------|----------------|
| Frequency              | Second     | Minute    | Minute    | Minute       | Minute        | Second        | Second         |
| Range                  | to Year    | to Year   | to Week   | to Day       | to Week       | to Year       | to Year        |
| Num.                   | 7          | 6         | 5         | 3            | 5             | 7             | 10             |
| Domains                | ,          | Ü         | 3         | 3            | 3             | ,             | 10             |
| Train/Test             | 1          | 1         | 1         | 1            | 1             | /             | X              |
| data leak              |            | •         |           | •            | •             | •             |                |
| Variate                | Uni        | Uni/Multi | Multi     | Multi        | Multi         | Uni/Multi     | Uni/Multi      |
| Types                  | Oili       | Om/whiti  | Multi     | Multi        | Multi         | Om/Muiti      | Om/Muiti       |
| Prediction             | Short      | Short     | Long      | Short/Long   | Short/Long    | Short/Long    | Short/Long     |
| Length                 | Short      | Short     | Long      | Short Long   | Short/Long    | Short/Long    | Short Long     |
| Stat. Benchmarks       | <b>X</b>   | X         | X         | ×            | ×             | X             | ✓              |
| Forecaster types       | Stat./DL   | Stat./DL  | Stat./DL  | Stat./DL     | Stat./DL/FM   | Stat./DL/FM   | Stat./ML/DL/FM |
| Hyperparam. autotuning | Х          | Х         | X         | Х            | Х             | ×             | ✓              |

#### 2 Background

We refer the reader to Appendix A for the notational convention we adopt, as well as for additional mathematical preliminaries and evaluation metric definitions.

<sup>&</sup>lt;sup>1</sup>TSFMs require hyperparameter searches during pretraining, but not during evaluation.

Forecasters A (time-series) forecasting task  $\mathcal{T} \doteq (l,h,n,m,\mathcal{X},\mathcal{Y},\boldsymbol{X},\boldsymbol{Y})$  consists of a context length  $l \in \mathbb{N}$ , a forecast horizon  $h \in \mathbb{N}$ ,  $m \in \mathbb{N}$  target time-series  $\boldsymbol{Y} = (\boldsymbol{y}_1,\ldots,\boldsymbol{y}_m)^T$  where for each variate  $i \in [m]$ , entries of  $\boldsymbol{y}_i \in \mathcal{Y}_i^l$  take values from a set of target values  $\mathcal{Y} \subseteq \mathbb{R}$ , and  $n \in \mathbb{N}$  covariate time-series  $\boldsymbol{X} = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)^T$  where for each covariate  $j \in [n]$ ,  $\boldsymbol{x}_j \in \mathcal{X}^{l+h}$  takes values from a set of covariate values  $\mathcal{X}_j \subseteq \mathbb{R}$ . For convenience, we denote the joint set of target variate values by  $\mathcal{Y} \doteq \boldsymbol{\times}_{i \in [m]} \mathcal{Y}_i$  and the joint set of covariate values  $\mathcal{X} \doteq \boldsymbol{\times}_{j \in [n]} \mathcal{X}_j$ . A forecasting task  $\mathcal{T}$  is said to be univariate (resp. multivariate) iff m = 1 (m > 1). A forecasting task  $\mathcal{T}$  is said to be a continuous (resp. conditional) iff n = 1 (resp. n > 1). A forecasting task  $\mathcal{T}$  is said to be a continuous (resp. count | categorical | binary) forecasting task iff for all  $i \in [m]$   $\mathcal{Y}_i \subseteq \mathbb{R}$  is a continuous set (resp.  $\mathcal{Y}_i = \mathbb{N} \mid \mathcal{Y}_i \subseteq \mathbb{N} \mid \mathcal{Y}_i = \{0,1\}$ ).

A (point) forecast for a forecasting task  $\mathcal{T}$  is a matrix  $\hat{\boldsymbol{Y}} \doteq (\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_m)$  s.t. for all target variates  $i \in [m], \ \hat{\boldsymbol{y}}_i \in \mathcal{Y}_i^h$  corresponds to forecasted values of variate i for h steps. A (point) forecasting model (or, colloquially, a forecaster) is a mapping  $\boldsymbol{F}: \mathcal{X}^{l+h} \times \mathcal{Y}^l \to \mathcal{Y}^h$  s.t.  $\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{Y}) \doteq (\boldsymbol{f}_1(\boldsymbol{X}, \boldsymbol{Y}), \dots, \boldsymbol{f}_m(\boldsymbol{X}, \boldsymbol{Y}))^T$  is a forecast for  $\mathcal{T}^2$ . A probabilistic forecasting model is a mapping  $\boldsymbol{\pi}: \mathcal{X}^{l+h} \times \mathcal{Y}^l \to \Delta(\mathcal{Y}^h)$  s.t.  $\boldsymbol{\pi}(\boldsymbol{X}, \boldsymbol{Y})[\widehat{\boldsymbol{Y}}] > 0$  denotes the probability of  $\widehat{\boldsymbol{Y}} \in \mathcal{Y}^l$  being realized.

Forecasting Evaluation Frameworks In reality, many forecasters  $F^{\theta}: \mathcal{X}^{l+h} \times \mathcal{Y}^{l} \to \mathcal{Y}^{h}$  are dependent on some hyperparameters  $\theta \in \Theta$ , and it is more appropriate to talk about a *family* of forecasters  $\mathcal{F}^{\Theta} \doteq \{F^{\theta}\}_{\theta \in \Theta}$ , and choose the forecaster with parameters which is the most appropriate for a forecasting task.<sup>3</sup>

A forecaster evaluation framework  $\mathcal{B} \doteq (p,q,\mathcal{E},\{\Theta_i\}_{i=1}^p,\{\mathcal{F}^{\Theta_i}\}_{i=1}^p,\{\mathcal{T}_j\}_{j\in[q]})$  consists of  $p\in\mathbb{N}$  familes of forecasters, with for each  $i\in[p],\mathcal{F}^{\Theta_i}$  being defined by a set of hyperparameters  $\Theta_i;q\in\mathbb{N}$  forecasting tasks (or, colloquially, benchmarks, or benchmark tasks)  $\{\mathcal{T}_j\}_{j\in[q]}$ ; and a hyperparameter tuner  $\mathcal{E}:\mathcal{X}\times\mathcal{Y}\to\Theta$ , which takes as input a benchmark and outputs some hyperparameters.

## 6 3 TempusBench

110

In the workshop version of this paper, we release a restricted version of TempusBench, which is restricted to the evaluation of *(point) forecasting models* on unconditional time-series forecasting tasks. We describe in Section 4, additional extensions of TempusBench which will be released in the full-paper version.

TempusBench, denoted  $\mathcal{B}^{TB}$ , is a forecasting evaluation framework where the hyperparameter tuner  $\mathcal{E}^{TB}$  is given by three-step procedure: given a benchmark, a (sub)set of hyperparameters, and a family of forecasters, 1) a validation dataset of subsets of the target and covariate time-series are created, 2) for each hyperparameter in the (sub)set of hyperparameters, the average MSE is computed across all samples in the validation dataset, 3) the hyperparameter with the lowest MSE is output. We summarize the set of families of forecasters, and the set of benchmarks included in TempusBench in Table 4 (Appendix B.1) and Table 6 (Appendix B.2) respectively.

We include a summary of the results of TempusBench, in Table 3 and a visualization of the tensorboard interface.

#### 4 Next Directions and Conclusion

We omit for the workshop version of TempusBench two directions in which we have been developing
TempusBench, namely the inclusion of conditional forecasting problems, and the evaluation of
probabilistic forecasting models. We plan to release this more general version of TempusBench in
the coming months as part of the full-version of our paper.

<sup>&</sup>lt;sup>2</sup>While our definition is in line with the literatur [9], more generally, a forecaster can be defined as a mapping from forecasting tasks to forecasts, i.e.,  $\mathcal{T} \mapsto F(\mathcal{T}) = \widehat{Y}$ .

<sup>&</sup>lt;sup>3</sup>For instance, the forecast of an ARIMA model is dependent on choices of hyperparameters given by the order of number of time lags, the degree of differencing, and the order of the moving-average model, and it is more appropriate to talk of the family of ARIMA models.

Table 2: Taxonomy of all univariate and multivariate benchmark tasks included in TempusBench.

| Category         | Benchmark Tasks                                                                            |
|------------------|--------------------------------------------------------------------------------------------|
| Movement         | Stationary, Non-Stationary                                                                 |
| Data Quality     | Noisy data, Data with measurement error                                                    |
| Frequency        | Seconds, Minutes, Hours, Days, Weeks, Months, Quarterly, Years                             |
| Context Length   | 30, 100, 500, 1000                                                                         |
| Forecast Horizon | 1, 20, 100, 500, 1000                                                                      |
| Seasonality      | Cyclical, Non-Stationary cyclical, Regressive, Irregular, Additive, Multiplicative         |
| Domain           | Energy, Transport, Climate, Software, Web, Sales, Nature, Econ., Healthcare, Manufacturing |
| Dataset Coverage | sparse, dense                                                                              |
| Target Type      | continuous, count, binary, categorical                                                     |

| Model                 | RMSE    |
|-----------------------|---------|
| Moirai MoE            | 15.4136 |
| Timesfm               | 15.7026 |
| LSTM                  | 17.9253 |
| Random Forest         | 18.5096 |
| Moment                | 19.7116 |
| Moirai                | 20.1223 |
| Tabpfn                | 21.1180 |
| SVR                   | 22.4834 |
| Tiny Time Mixer       | 23.6602 |
| Seasonal Naive        | 25.6318 |
| Arima                 | 25.7544 |
| Exponential Smoothing | 26.3256 |
| XGBoost               | 27.1328 |
| Chronos               | 27.5313 |
| Theta                 | 32.7155 |
| Lagllama              | 45.3234 |
| Toto                  | 49.1227 |
| Prophet               | 63.9555 |

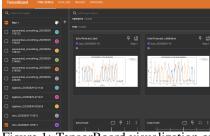


Figure 1: TensorBoard visualization of time-series forecasting experiments.

Table 3: Overall average performance of all models across all tasks. The best result for each metric is highlighted in bold.

We expect that the datasets used to define our benchmarks will eventually get included in the pretaining corpus of TSFMs, as has been the case often with NLP benchmarks. To this end, we are developing dynamic benchmarks where test data is continuously refreshed. While dynamic benchmarks can easily be defined benchmarks making use of synthetic data (e.g., our seasonality benchmarks) by continuously generating new datasets, for other benchmarks (e.g., our domain benchmarks) we are building a rotating set of datasets which are pulled from live data APIs.

building a rotating set of datasets which are pulled from live data APIs.

Finally, for the workshop version of TempusBench, in line with existing forecasting evaluation frameworks, we consider benchmark categories such as target variate type, context length, forecast length as defining individual forecasting tasks. However, a more comprehensive way to see these benchmark categories would be as hyperparameters for other benchmark categories such as domains. That is, for instance, a more comprehensive list of benchmarks would test the performance of forecasting models for each domain (e.g., economics) for different choices of target variate types, context lengths, and forecast lengths) We are planning to release these more comprehensive benchmark types in the coming months as part of the full-version of our paper.

#### References

129

130

131

[1] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco, 1970. 1

- [2] Vladimir N. Vapnik. The Nature of Statistical Learning Theory. Springer, New York, 1995. 1
- [3] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alexander Smola, and Vladimir
   Vapnik. Support vector regression machines. In *Advances in Neural Information Processing* Systems 9, pages 155–161. MIT Press, 1997. 1
- [4] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In
   Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining, pages 6555–6565, 2024.
- [5] Prajakta S Kalekar et al. Time series forecasting using holt-winters exponential smoothing.
   Kanwal Rekhi school of information Technology, 4329008(13):1–13, 2004. 1, 18, 27, 28, 29
- [6] Spyros Makridakis and Michele Hibon. The M3-competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000. 1, 26
- [7] Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. The M4 competition:
   Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):
   802–808, 2018. 1, 26
- [8] Pooja Anand, Mayank Sharma, and Anil Saroliya. A comparative analysis of artificial neural networks in time series forecasting using arima vs prophet. In 2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE), pages 527–533. IEEE, 2024.
   1, 19, 26, 27, 28, 29
- [9] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong,
   and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation.
   arXiv preprint arXiv:2410.10393, 2024. 1, 2, 3
- 154 [10] Salesforce. Gift-eval. Hugging Face Space, 2024. URL https://huggingface.co/ 155 spaces/Salesforce/GIFT-Eval. Accessed: 2025-08-29. 1
- 156 [11] u/nkafr. The rise of foundation time-series forecasting models. https: //www.reddit.com/r/datascience/comments/1e865bt/the\_rise\_of\_ 158 foundation\_timeseries\_forecasting/, 2024. URL https://www.reddit. 159 com/r/datascience/comments/1e865bt/the\_rise\_of\_foundation\_ 160 timeseries\_forecasting/. Reddit post on r/datascience. 2
- [12] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006. 2
- 163 [13] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, et al. Monash time series forecasting archive. In *NeurIPS Datasets and Benchmarks Track*, 2021. 2, 26
- Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo,
   Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. Tfb: Towards comprehensive and fair
   benchmarking of time series forecasting methods. arXiv preprint arXiv:2403.20150, 2024. 2,
   26
- [15] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023. 2, 26
- [16] Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Tao
   Sun, Guangyin Jin, Xin Cao, et al. Exploring progress in multivariate time series forecasting:
   Comprehensive benchmarking and heterogeneity analysis. *IEEE Transactions on Knowledge* and Data Engineering, 2024. 2
- [17] Jiawen Zhang, Xumeng Wen, Zhenwei Zhang, Shun Zheng, Jia Li, and Jiang Bian. Probts:
   Benchmarking point and distributional forecasting across diverse prediction horizons. *Advances in Neural Information Processing Systems*, 37:48045–48082, 2024.

- 179 [18] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. *arXiv preprint* arXiv:2402.02592, 2024. 13
- [19] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao
   Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series
   foundation models with sparse mixture of experts. arXiv preprint arXiv:2410.10469, 2024. 13
- [20] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model
   for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.
   13, 14
- Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin
   Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham
   Kapoor, et al. Chronos: Learning the language of time series. arXiv preprint arXiv:2403.07815,
   2024. 14, 26
- [22] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: How tabpfn v2 outperforms specialized time series forecasting models. arXiv preprint arXiv:2501.02945,
   2025. 15
- Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam Nguyen, Wesley M
   Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained
   models for enhanced zero/few-shot forecasting of multivariate time series. Advances in Neural
   Information Processing Systems, 37:74147–74181, 2024.
- [24] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian
   Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir
   Hassen, et al. Lag-llama: Towards foundation models for probabilistic time series forecasting.
   arXiv preprint arXiv:2310.08278, 2023. 16
- 203 [25] Ben Cohen, Emaad Khwaja, Youssef Doubli, Salahidine Lemaachi, Chris Lettieri, Charles
  204 Masson, Hugo Miccinilli, Elise Ramé, Qiqi Ren, Afshin Rostamizadeh, et al. This time is
  205 different: An observability perspective on time series foundation models. *arXiv preprint*206 *arXiv:2505.14766*, 2025. 16, 29
- [26] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski.
   Moment: A family of open time-series foundation models. arXiv preprint arXiv:2402.03885,
   2024. 17, 26, 29
- 210 [27] Sima Siami-Namini and Akbar Siami Namin. Forecasting economics and financial time series:
  211 Arima vs. lstm. *arXiv preprint arXiv:1803.06386*, 2018. 17, 19, 26, 27, 28, 29
- 212 [28] Thomas R Willemain, Charles N Smart, Joseph H Shockor, and Philip A DeSautels. Fore-213 casting intermittent demand in manufacturing: a comparative evaluation of croston's method. 214 *International Journal of forecasting*, 10(4):529–538, 1994. 18
- [29] Lin Lin, Fang Wang, Xiaolong Xie, and Shisheng Zhong. Random forests-based extreme
   learning machine ensemble for multi-regime time series prediction. *Expert Systems with Applications*, 83:164–176, 2017.
- 218 [30] Senyao Wang and Jin Ma. A novel ensemble model for load forecasting: Integrating random forest, xgboost, and seasonal naive methods. In 2023 2nd Asian Conference on Frontiers of Power and Energy (ACFPE), pages 114–118. IEEE, 2023. 20, 22, 27, 28, 29
- [31] Fan Zhang and Lauren J O'Donnell. Support vector regression. In *Machine learning*, pages
   123–140. Elsevier, 2020. 21, 27
- 223 [32] Dimitrios D Thomakos and Konstantinos Nikolopoulos. Forecasting multivariate time series with the theta method. *Journal of Forecasting*, 34(3):220–229, 2015. 21, 26, 27, 28
- 225 [33] Indeed. Software Development Job Postings on Indeed in the United States
  226 [IHLIDXUSTPSOFTDEVE]. https://fred.stlouisfed.org/series/
  227 IHLIDXUSTPSOFTDEVE, 2025. Retrieved August 29, 2025. 23

- [34] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 95–104, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.
   233
- 234 [35] Energy and Geoscience Institute at the University of Utah. Utah FORGE: Well 16A(78)-32 Drilling Data. Accessed via Data.gov, 2025. Accessed on August 29, 2025. 23
- [36] DeltaTrup. LT 1-Minute Historical Stock Data (2003 -236 2024). https://www.kaggle.com/datasets/deltatrup/ 237 lt-1-minute-historical-stock-data-2003-2024, may 2024. Accessed 238 on August 29, 2025. 23 239
- 2001-2022 240 [37] IgnacioQG. Hourly Dataset of Pollution in Madrid. https://www.kaggle.com/datasets/ignaciogg/ 241 20012022-hourly-dataset-of-pollution-in-madrid, 2022. Accessed 242 on August 29, 2025. 23 243
- 244 [38] Coinbase. Coinbase Litecoin [CBLTCUSD]. https://fred.stlouisfed.org/ 245 series/CBLTCUSD, 2025. Retrieved August 29, 2025. 23
- [39] Nisarg Chodavadiya. Daily Gold Price (2015-2021)Time Series. 246 https://www.kaggle.com/datasets/nisargchodavadiya/ 247 daily-gold-price-20152021-time-series, 2025. Accessed on August 29, 248 2025. 23 249
- 250 [40] Board of Governors of the Federal Reserve System (US). Federal Funds Effective Rate [FF].
  251 https://fred.stlouisfed.org/series/FF, 2025. Retrieved August 29, 2025. 23
- U.S. Census Bureau. Manufacturers: Inventories to Sales Ratio [MNFCTRIRSA]. https://fred.stlouisfed.org/series/MNFCTRIRSA, 2025. Retrieved August 29, 2025.
   23
- 255 [42] GabrielSantello. Airline Baggage Complaints Time Series Dataset.
  256 https://www.kaggle.com/datasets/gabrielsantello/
  257 airline-baggage-complaints-time-series-dataset, 2023. Accessed
  258 on August 29, 2025. 23
- 259 [43] Bank for International Settlements. Real Residential Property Prices for Germany [QDER628BIS]. https://fred.stlouisfed.org/series/QDER628BIS, 2025. Retrieved August 29, 2025. 23
- 262 [44] U.S. Bureau of Economic Analysis. Personal Consumption Expenditures: Chain-type
  263 Price Index [DPCERG3A086NBEA]. https://fred.stlouisfed.org/series/
  264 DPCERG3A086NBEA, 2025. Retrieved August 29, 2025. 23
- [45] BITS Pilani Goa. SplitSmart: An Open Dataset for Enabling Research in Energy-Efficient
   Ductless-Split Air Conditioner, 2024. Accessed on August 29, 2025. 23
- [46] Jorge Bañuelos-Gimeno, Natalia Sobrino, and Rosa Arce-Ruiz. Initial Insights into Teleworking's Effect on Air Quality in Madrid City. *Environments*, 11(9):204, 2024. doi: 10.3390/environments11090204. URL https://www.mdpi.com/2076-3298/11/9/204. 23
- 270 [47] SumanthVrao. Daily Climate Time Series Data. https://www.kaggle.com/ 271 datasets/sumanthvrao/daily-climate-time-series-data, 2021. Accessed 272 on August 29, 2025. 23
- 273 [48] Riccardo Taormina et al. The Battle of the Attack Detection Algorithms: Disclosing Cyber
  274 Attacks on Water Distribution Networks. *Journal of Water Resources Planning and Management*,
  275 144(8):04018048, aug 2018. doi: 10.1061/(ASCE)WR.1943-5452.0000969. URL https:
  276 //www.batadal.net/data.html. 23

- 277 [49] RaminHuseyn. Web Traffic Time Series Dataset. https://www.kaggle.com/ 278 datasets/raminhuseyn/web-traffic-time-series-dataset, 2024. Ac-279 cessed on August 29, 2025. 23
- 280 [50] NoeyIsLearning. Soil and Environmental Monitoring. https://www.kaggle.com/ 281 datasets/noeyislearning/soil-and-environmental-monitoring, 2024. Accessed on August 29, 2025. 23
- 283 [51] U.S. Bureau of Labor Statistics. All Employees, Health Care [CES6562000101]. https:
  284 //fred.stlouisfed.org/series/CES6562000101, 2025. Retrieved August 29,
  2025. 23
- [52] City of New York. COVID-19 Daily Counts of Cases, Hospitalizations, and Deaths, 2025.
   Accessed on August 29, 2025. Daily count of NYC residents who tested positive for SARS-CoV-2, hospitalized with COVID-19, and deaths among COVID-19 patients. 23
- 289 [53] Alistair Johnson et al. MIMIC-III Clinical Database Demo (version 1.4). https://doi. 290 org/10.13026/C2HM2Q, 2019. RRID:SCR\_007345. 23
- [54] UCI Machine Learning Repository. Hungarian Chickenpox Cases. https://doi.org/10.
   24432/C5103B, 2021. 23
- 293 [55] Paulo Cortez and Aníbal Morais. Forest Fires. https://doi.org/10.24432/C5D88D, 294 2007. 23
- 295 [56] Adarsh Pal Singh and Sachin Chaudhari. Room Occupancy Estimation. https://doi. 296 org/10.24432/C5P605, 2018. 23
- 297 [57] Andrea Martiniano and Ricardo Ferreira. Absenteeism at work. https://doi.org/10. 24432/C5X882, 2012. 23
- 299 [58] Daqing Chen. Online Retail. https://doi.org/10.24432/C5BW33, 2015. 23
- [59] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley, 5 edition, 2015.
- [60] Rob J. Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. Forecasting with
   Exponential Smoothing: The State Space Approach. Springer, 2008.
- Vassilis Assimakopoulos and Konstantinos Nikolopoulos. The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, 16(4):521–530, 2000. 26
- 306 [62] Helmut Lütkepohl. New Introduction to Multiple Time Series Analysis. Springer, 2005. 26
- <sup>307</sup> [63] Valentin Flunkert, David Salinas, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks, 2017. 26
- Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural
   basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2020. 26
- 312 [65] Abhimanyu Das et al. Long-term forecasting with tide: Time-series dense encoder. In *Neural Information Processing Systems (NeurIPS)*, 2023. 26
- [66] Bryan Lim, Sercan Ö. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for
   interpretable multi-horizon time series forecasting, 2019.
- [67] Yifan Nie, Zhihan Huang, Li Wang, Yuheng Sun, Yating He, and Zhifeng Zhang. A time series
   is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023. 26
- 1319 [68] Han Liu et al. itransformer: Inverted transformers are effective for time series forecasting, 2023.
  26

- [69] Kashif Rasul, V Ashkinazi, I Schuster, A Schneider, and A Mishkin. Autoregressive denoising
   diffusion models for multivariate probabilistic time series forecasting. In *Neural Information Processing Systems (NeurIPS)*, 2021.
- Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *Neural Information Processing Systems (NeurIPS)*, 2021. 26
- [71] Kashif Rasul et al. Multivariate probabilistic time series forecasting via conditioned normalizing
   flows, 2020. 26
- [72] Sangwoo Woo et al. Moirai: Foundation models for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024. 26
- [73] Abhimanyu Das et al. Timesfm: Time series foundation models at scale, 2023. 26
- 332 [74] Kashif Rasul et al. Lag-llama: Towards foundation models for time series forecasting, 2023. 26
- <sup>333</sup> [75] Yan Liu et al. Timer: Efficient time-series foundation model, 2024. 26
- 1334 [76] Tian Gao et al. Units: Universal time series foundation models, 2024. 26
- 77] Vinay Ekambaram et al. Multi-level tiny time mixers for efficient time-series foundation models,
   2024. 26
- 337 [78] Xing Chen et al. Visionts: Multimodal time-series foundation models, 2024. 26
- Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. The M5 accuracy
   competition: Results, findings and conclusions. *International Journal of Forecasting*, 38(4):
   1346–1364, 2022. 26
- [80] Haixu Zhang et al. Probts: Benchmarking probabilistic forecasting. In Neural Information
   Processing Systems (NeurIPS), 2023. 26
- Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1): 37–45, 2018. 26
- Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, et al. sktime: A unified interface for machine learning with time series. In *Proceedings of the 2nd Workshop on Systems for ML* (NeurIPS), 2019. 26
- <sup>348</sup> [83] Alexander Alexandrov et al. Gluonts: Probabilistic time series models in python, 2020. 26
- Kashif Rasul. Pytorchts: A probabilistic deep learning library for time series, 2021. GitHub repository: https://github.com/zalandoresearch/pytorch-ts. 26
- Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. 26
- Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association, 102(477):359–378, 2007. 26
- Raj Kumar Tamatta. *Time series forecasting of hospital Inpatients and Day case waiting list using ARIMA, TBATS and Neural Network Models*. PhD thesis, Dublin, National College of Ireland, 2018. 26
- 258 [88] Caitlin Haskins. 5 Financial Forecasting Methods to Help Your Business. https://online. 259 hbs.edu/blog/post/financial-forecasting-methods, apr 2021. Accessed on 260 August 14, 2025. 27

## 361 A Additional Mathematical Background

#### 362 A.1 Mathematical notation

- We adopt the following calligraphic conventions to insist on the nature of the mathematical object at hand: We use calligraphic uppercase letters to denote sets (e.g.,  $\mathcal{X}$ ), bold uppercase letters to denote matrices (e.g.,  $\mathcal{X}$ ), bold lowercase letters to denote vectors (e.g.,  $\mathcal{P}$ ), lowercase letters to denote scalar quantities (e.g.,  $\mathcal{X}$ ), and uppercase letters to denote random variables (e.g.,  $\mathcal{X}$ ). We denote the ith row vector of a matrix (e.g.,  $\mathcal{X}$ ) by the corresponding bold lowercase letter with subscript i (e.g.,  $x_i$ ). Similarly, we denote the jth entry of a vector (e.g., p or  $x_i$ ) by the corresponding lowercase letter with subscript j (e.g.,  $p_j$  or  $x_{ij}$ ). We denote functions by a letter determined by the value of
- letter with subscript j (e.g.,  $p_j$  or  $x_{ij}$ ). We denote functions by a letter determined by the value of the function, e.g., f if the mapping is scalar valued, f if the mapping is vector valued, and  $\mathcal{F}$  if the mapping is set valued.
- We denote the set  $\{1,\ldots,n\}$  by [n], the set  $\{n,n+1,\ldots,m\}$  by [n:m], the set of natural numbers by  $\mathbb N$ , and the set of real numbers by  $\mathbb R$ . We denote the positive and strictly positive elements of a set using a + or ++ subscript, respectively, e.g.,  $\mathbb R_+$  and  $\mathbb R_{++}$ . For any  $n\in\mathbb N$ , we denote the n-dimensional vector of zeros and ones by  $\mathbf 0_n$  and  $\mathbf 1_n$ , respectively.

#### 376 A.2 Mathematical Definitions

We let  $\Delta_n = \{ \boldsymbol{x} \in \mathbb{R}^n_+ \mid \sum_{i=1}^n x_i = 1 \}$  denote the unit simplex in  $\mathbb{R}^n$ , and  $\Delta(A)$  denote the set of all probability measures over a given set A. We also define the support of a probability density function  $f \in \Delta(\mathcal{X})$  as  $\operatorname{supp}(f) \doteq \{ \boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) > 0 \}$ . Finally, we denote the orthogonal projection operator onto a set C by  $\Pi_C$ , i.e.,  $\Pi_C(\boldsymbol{x}) \doteq \arg\min_{\boldsymbol{y} \in C} \|\boldsymbol{x} - \boldsymbol{y}\|^2$ .

#### A.3 Evaluation Metrics

381

An evaluation metric  $\ell: \mathcal{Y}^h \times \mathcal{Y}^h \to \mathbb{R}_+$  is a positive-, scalar-valued function s.t. for any forecast  $\widehat{\boldsymbol{Y}} \in \mathcal{Y}^h$  and realized future target values  $\boldsymbol{Y}^* \in \mathcal{Y}^h$ ,  $\ell(\widehat{\boldsymbol{Y}}, \boldsymbol{Y}^*) \geq 0$  denotes the distance between the forecast and the realized values. We consider the following evaluation metrics at present. The *mean absolute error (MAE)* is defined as  $\ell^{\text{MAE}}(\widehat{\boldsymbol{Y}}, \boldsymbol{Y}^*) \doteq \frac{1}{mh} \sum_{i \in [m]} \sum_{t=1}^h |\widehat{y}_{it} - y_{it}^*|$ . The *mean squared error (MSE)* is defined as  $\ell^{\text{MASE}}(\widehat{\boldsymbol{Y}}, \boldsymbol{Y}^*) \doteq \frac{1}{mh} \sum_{i \in [m]} \sum_{t=1}^h (\widehat{y}_{it} - y_{it}^*)^2$ . The *mean absolute scale error (MASE)* is defined as  $\ell^{\text{MASE}}(\widehat{\boldsymbol{Y}}, \boldsymbol{Y}^*) \doteq \frac{1}{mh} \sum_{i \in [m]} \sum_{t=1}^h \frac{|\widehat{y}_{it} - y_{it}^*|}{\frac{1}{h-1} \sum_{t=1}^{l-1} |y_{it+1} - y_{it}|}$ .

<sup>&</sup>lt;sup>4</sup>We note MAE is scale-dependent but less sensitive to outliers, MSE disproportionately penalizes large forecast errors and is therefore more outlier-sensitive, while MASE normalizes errors w.r.t. the forecasts of naive forecast method (i.e., setting the next time-step's forecast to be the current time-step realized value), making it scale-free and comparable across datasets or domains.

## **B** Additional Information on TempusBench

## **B.1** Forecasting Models Included in TempusBench

Table 4: Summary of forecasters included in TempusBench.

| Category                   | Included Models                                                                                        | Core Characteristics                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------------|--------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Foundation Models          | Moirai, Moirai-MoE,<br>TimesFM, TimesFM-2.0,<br>Chronos, Lag-Llama, Toto,<br>MOMENT, TTM,<br>TabPFN-TS | Paradigm: Universal, zeroshot/few-shot forecasting. A single large model is pre-trained on massive, diverse datasets and generalizes to new tasks without retraining.  Architecture: Primarily based on Transformers or other deep learning structures like MLP-Mixers. They process raw time series via patching or novel tokenization schemes.  I/O: Often produce probabilistic forecasts and can natively handle univariate, multivariate, and covariate data.     |
| Classic Machine Learning   | LSTM, Random Forest,<br>XGBoost, SVR                                                                   | Paradigm: Supervised learning models trained per-dataset. They excel at capturing complex, nonlinear relationships but require specific training for each task.  Architecture: Diverse, including Recurrent Neural Networks (for sequence memory), Tree Ensembles (for interaction effects), and Kernel Methods.  I/O: Typically require explicit feature engineering (e.g., lags, calendar variables) to create a tabular format. Most often produce point forecasts. |
| Statistical & Decomposable | ARIMA, Holt-Winters,<br>Prophet, Theta Method,<br>Croston's Method, Seasonal<br>Naive                  | Paradigm: Assume the time series is generated by an underlying statistical process or can be decomposed into simpler, interpretable components like trend and seasonality.  Architecture: An explicit mathematical formula is fitted directly to an individual time series.  I/O: Highly interpretable point forecasts. Often specialized for particular data patterns (e.g., intermittency with <i>Croston's</i> ).                                                   |

In this section, we summarize the forecasting models which have been included in TempusBench.

We summarize all models in Table 4, and provide and comparison of TSFMs, machine learning

forecasting models, and statistical forecasting models in Table 5.

Table 5: Comparative Overview of Forecasting Models

| Feature         | Group 1: Foundation Models                                                                                                                                                                  | Group 2: Machine Learning Models                                                                                                                                                           | Group 3: Classical & Statistical Models                                                                                                                                                                  |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Models Included | <ul> <li>Moirai / Moirai-MoE</li> <li>TimesFM / TimesFM-2.0</li> <li>Chronos</li> <li>Lag-Llama</li> <li>Toto</li> <li>MOMENT</li> <li>TabPFN-TS</li> <li>Tiny Time Mixers (TTM)</li> </ul> | • LSTM • Random Forest • XGBoost • SVR                                                                                                                                                     | <ul> <li>ARIMA</li> <li>Holt-Winters</li> <li>Prophet</li> <li>Theta Method</li> <li>Croston's Method</li> <li>Seasonal Naive</li> </ul>                                                                 |
| Core Paradigm   | Large, pre-trained models designed for universal, zero-shot forecasting. They learn general time-series patterns from massive, diverse datasets.                                            | Models trained for a specific forecasting task, often relying on feature engineering. They leverage distinct architectures (e.g., recurrence, ensembles) rather than massive pre-training. | Model-based approaches assuming an underlying stochastic process or decomposable structure. Parameters are estimated directly from the target time series.                                               |
| Architecture    | Primarily Transformer-based (Encoder, Decoder, or both). Innovations include MoE layers, residual forecasting, and specialized attention mechanisms.                                        | Diverse architectures: MLP-Mixer (TTM), RNN (LSTM), Tabular-Transformer (TabPFN-TS), Tree Ensembles (RF, XG-Boost), and Kernel-based (SVR).                                                | Mathematical formulations: State-space models (ARIMA, Holt-Winters), decomposable additive models (Prophet, Theta), and simple heuristics (Croston's, S. Naive).                                         |
| Input Handling  | Process raw time series, typically via patching (Moirai, TimesFM), lag-based tokenization (Lag-Llama), or value quantization (Chronos). Can natively handle uni/multivariate series.        | Generally require explicit feature engineering (e.g., lags, calendar variables) to create a tabular dataset (RF, XGBoost, SVR). LSTM and TTM process raw sequences.                        | Operate directly on the univariate time series. May require stationarity ( <i>ARIMA</i> ) or be specialized for patterns like seasonality ( <i>Holt-Winters</i> ) or intermittency ( <i>Croston's</i> ). |
| Output Type     | Mostly probabilistic, predicting the parameters of a flexible distribution. Point forecasts are derived from the distribution (e.g., median).                                               | Primarily point forecasts. <i>TabPFN-TS</i> is a notable exception, providing a probabilistic output by approximating the posterior.                                                       | Primarily point forecasts. <i>Prophet</i> is an exception, generating uncertainty intervals. Probabilistic versions exist but are not standard.                                                          |
| Key Trait       | Powerful zero-shot/few-shot performance. High model capacity and ability to generalize across domains without dataset-specific training.                                                    | Model-specific strengths: computational efficiency (TTM), modeling long-term dependencies (LSTM), and capturing complex non-linear interactions (RF, XG-Boost).                            | High interpretability and strong statistical foundations. Often specialized and highly efficient for specific data patterns (e.g., trend, seasonality, intermittency).                                   |

#### B.1.1 Moirai

Moirai is a universal time series forecasting model developed by Salesforce AI Research, built upon a masked encoder-only Transformer architecture. It is designed as a single, large pre-trained model capable of handling diverse forecasting tasks without dataset-specific retraining. The model is pre-trained on LOTSA, a large-scale archive of over 27 billion observations, enabling it to perform powerful zero-shot forecasting. [18]

- **Input:** Accepts univariate or multivariate time series with an arbitrary number of variates and covariates.
- Output: Produces a probabilistic forecast by predicting the parameters of a flexible mixture distribution (composed of Student's t, Negative Binomial, Log-Normal, and low-variance Normal distributions).
- Architecture: Employs a masked encoder-only Transformer. Its key innovations include:
  - Multi Patch Size Projection: Uses different patch sizes to effectively process time series of varying frequencies.
  - Any-variate Attention: Flattens multivariate series into a single sequence and uses binary attention biases to manage an arbitrary number of variates while maintaining permutation equivariance.
  - **Forecasting Type:** A universal, zero-shot, probabilistic forecaster. It can generate point forecasts by taking the median of the predicted distribution.

#### 412 B.1.2 Moirai-MoE

405

406

407

408

409

410

411

426

427 428

429

430

Moirai-MoE is an advanced version of the Moirai foundation model that integrates a Sparse Mixture of Experts (MoE) architecture. Instead of relying on heuristic-based, frequency-specific projection layers, Moirai-MoE delegates the task of modeling diverse time series patterns to specialized "expert" networks within its Transformer layers. This allows for automatic, token-level specialization in a data-driven manner, leading to improved accuracy and greater efficiency in terms of activated parameters. [19]

- **Input:** Accepts univariate or multivariate time series with an arbitrary number of variates and covariates.
- Output: Produces a probabilistic forecast by predicting the parameters of a flexible mixture distribution for the next token in an autoregressive manner.
- **Architecture:** Employs a decoder-only Transformer that replaces the standard Feed-Forward Network (FFN) layers with MoE layers. Key architectural changes from the original Moirai include:
  - Mixture of Experts (MoE): A gating function routes each time series token to a small subset
    of specialized expert networks, allowing the model to handle diverse patterns at a granular
    level.
  - Single Projection Layer: It uses a single input/output projection layer for all time series, removing the dependency on frequency-based heuristics.
- Forecasting Type: A universal, zero-shot, probabilistic forecaster that is more accurate and efficient (in terms of activated parameters) than the original Moirai model. It can generate point forecasts by taking the median of the predicted distribution.

#### 434 **B.1.3 TimesFM**

TimesFM is a time-series foundation model developed by Google Research, designed for zero-shot forecasting. It is based on a decoder-only Transformer architecture and is pretrained on a very large corpus of time series data, combining both real-world and synthetic sources. The model's key objective is to provide accurate out-of-the-box point forecasts on unseen datasets without requiring any dataset-specific training. [20]

- **Input:** Accepts a univariate time series context window.
  - Output: Produces a point forecast for a given prediction horizon.

- **Architecture:** Employs a decoder-only Transformer architecture that processes the time series in patches. Key architectural features include:
  - Decoder-Only Transformer: Utilizes a standard decoder-style attention mechanism to autoregressively predict future values patch by patch.
  - Input Patching: The input time series is segmented into non-overlapping patches, which are then embedded using a residual block of MLPs before being fed to the Transformer.
  - Forecasting Type: A universal, zero-shot, point forecaster designed primarily for long-horizon forecasting tasks.

#### 450 B.1.4 TimesFM-2.0

445

446

447

448

449

460

461

462

463

464

478

479

480

481

482 483

- TimesFM-2.0 is an improved version of the original foundation model from Google Research. While retaining the same decoder-only Transformer architecture, its key innovation lies in forecasting the residual component of a time series after performing a seasonal-trend decomposition. This approach makes the model significantly more accurate, particularly for time series that exhibit clear trends.
- **Input:** Accepts a univariate time series context window.
- Output: Produces a point forecast for a given prediction horizon.
- **Architecture:** Based on the original decoder-only Transformer with input patching. The primary architectural update is its **residual forecasting** methodology:
  - Seasonal-Trend Decomposition: The model first decomposes the input series to separate its trend and seasonal components.
  - Residual Forecasting: The core Transformer then forecasts the residual (the signal remaining after decomposition). This forecast is added back to the projected trend to produce the final prediction.
- Forecasting Type: A universal, zero-shot, point forecaster with enhanced performance on trended time series compared to its predecessor.

#### 467 B.1.5 Chronos

- Chronos is a family of pretrained time series models developed by Amazon Science that frames forecasting as a language modeling task. The core idea is to "tokenize" time series values by scaling and quantizing them into a fixed vocabulary. By doing so, standard Transformer-based language model architectures can be trained on sequences of these tokens using a cross-entropy loss, effectively learning the "language" of time series. [21]
- **Input:** Accepts a univariate time series context window.
- Output: Produces a probabilistic forecast by generating multiple sample future trajectories. A point forecast can be derived from the median of these samples.
- **Architecture:** Based on standard language model architectures (specifically the T5 encoder-decoder family). Its defining characteristic is its unique data preprocessing pipeline:
  - Tokenization via Quantization: The model first applies mean scaling to the input time series. It then quantizes these scaled values into a finite set of discrete tokens, converting the continuous series into a sequence of categorical variables.
  - Language Model Training: The model is trained to predict the next token in a sequence using a standard cross-entropy loss, analogous to how a language model predicts the next word.
- Forecasting Type: A universal, zero-shot, probabilistic forecaster.

#### 485 **B.1.6 TabPFN**

TabFPN is a forecasting framework that adapts feature pyramid networks (FPN), originally developed for computer vision tasks, to tabular time-series data. The approach builds hierarchical feature representations across multiple temporal resolutions, enabling the model to capture both short- and long-range dependencies. Unlike traditional time-series architectures, TabFPN treats forecasting as a

structured feature-learning problem on tabularized sequences, combining multiscale decomposition
 with probabilistic prediction.

- **Input:** A univariate or multivariate time series, converted into tabular form with hierarchical features at multiple temporal resolutions.
- Output: Produces probabilistic forecasts by estimating distributions over future values at each horizon; point forecasts can be obtained from the distribution mean or median.

#### • Architecture:

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

515

516

517

518

519

520

521

522

523

529

530

531

534

535

536

537

538

- Feature Pyramids: The series is decomposed into multiple temporal scales (e.g., short-term, medium-term, seasonal) using windowed transformations. Each scale yields a feature representation.
- FPN Backbone: These features are passed into a feature pyramid network adapted for tabular regression, allowing cross-scale information flow and refinement.
- Prediction Head: Aggregates multiscale features to generate forecasts, with uncertainty quantification via distributional outputs.
- Forecasting Type: A universal, zero-shot, probabilistic forecaster with explicit multiscale feature integration.

#### **B.1.7 TabPFN-TS**

TabPFN-TS is a novel approach that adapts TabPFN-v2, a general-purpose tabular foundation model, for time series forecasting. The core methodology involves recasting the forecasting problem as a tabular regression task. This is achieved through lightweight feature engineering on the time index, without relying on lagged values. Notably, the underlying TabPFN-v2 model was pretrained exclusively on synthetic tabular data and has not seen any time series data. [22]

- **Input:** A univariate time series, which is converted into a feature matrix based on timestamps.
- Output: Produces a probabilistic forecast by approximating the posterior predictive distribution for each future time step. Point forecasts can be derived from the mean or median of this distribution.
  - Architecture: It does not use a time-series-specific architecture. Instead, it relies on:
    - Feature Engineering: The time series is transformed into a tabular dataset by creating
      features from timestamps. These include standard calendar features (e.g., hour of day, day of
      week), automatically detected seasonal features via a Fourier transform, and a simple running
      index.
    - TabPFN-v2 Model: The generated tabular data is fed into the pretrained TabPFN-v2 model, which performs the regression task to predict future values.
  - Forecasting Type: A universal, zero-shot, probabilistic forecaster.

#### **B.1.8** Tiny Time Mixers (TTM)

Tiny Time Mixers (TTM) is a family of lightweight pre-trained models from IBM Research, based on the efficient TSMixer architecture. In contrast to large, LLM-based approaches, TTMs are designed to be extremely small (<1M parameters) and fast, while still providing strong zero-shot and few-shot forecasting performance. The models are pre-trained exclusively on a large corpus of public time series datasets, making them a highly efficient alternative for universal forecasting. [23]

- **Input:** Accepts univariate or multivariate time series, with optional support for exogenous variables during the fine-tuning stage.
- Output: Produces a point forecast for a given prediction horizon.
- **Architecture:** Based on the MLP-Mixer architecture. The model is pre-trained in a channel-independent manner and uses a multi-level structure to handle diverse data and tasks.
  - TSMixer Backbone: The core of the model uses simple MLP blocks for temporal and feature mixing, avoiding the computational overhead of Transformer-based attention.
  - Multi-Resolution Pre-training: Employs several novel techniques to handle heterogeneous datasets, including adaptive patching (using different patch configurations at different layers) and data augmentation via downsampling.

- Multi-level Modeling: Uses a frozen pre-trained backbone and a smaller, fine-tunable decoder, which can incorporate channel-mixing and an exogenous mixer to fuse external signals for target-specific tasks.
- Forecasting Type: A universal, zero-shot/few-shot, point forecaster, notable for its small size and computational efficiency.

#### 4 B.1.9 Lag-Llama

539

540

541

551

552

555

556

557

558

559

573

574

575

577

578

579

580

581

582

Lag-Llama is a foundation model for univariate probabilistic time series forecasting. It is built upon a decoder-only Transformer architecture, similar to LLaMA, and is pretrained on a large, diverse corpus of open-source time series data. The model's key innovation is its tokenization strategy, which uses lagged values of the time series as input features, allowing it to generalize across different frequencies and domains. [24]

- **Input:** Accepts a univariate time series context window.
  - **Output:** Produces a probabilistic forecast by outputting the parameters of a Student's t-distribution for the next time step. Future trajectories are generated autoregressively.
- **Architecture:** Based on a decoder-only Transformer (LLaMA). Its defining characteristic is its input representation:
  - Tokenization via Lag Features: Instead of patching, the input token for each time step is a vector composed of lagged values from the time series history (e.g., values from 1, 7, and 14 days prior). This is augmented with standard date-time features.
  - Value Scaling: Applies robust scaling (using median and IQR) to normalize the input values and includes the scaling parameters as additional features.
  - Forecasting Type: A universal, zero-shot/few-shot, probabilistic forecaster.

#### 561 **B.1.10 Toto**

Toto (Time Series Optimized Transformer for Observability) is a foundation model from Datadog, specifically designed for multivariate time series forecasting with a focus on observability metrics. It is built on a decoder-only Transformer architecture and incorporates several novel components to handle the unique challenges of observability data, such as high non-stationarity and heavy-tailed distributions. The model is pretrained on a large and diverse corpus that includes real-world observability data, public datasets, and synthetic data. [25]

- Input: Accepts multivariate time series.
- Output: Produces a probabilistic forecast by predicting the parameters of a Student-T mixture model.
- **Architecture:** A decoder-only Transformer with several key innovations tailored for observability data:
  - Patch-based Causal Normalization: A novel per-patch scaling method that computes normalization statistics from current and past data to handle highly nonstationary series.
  - Proportional Factorized Attention: An efficient attention mechanism that uses a mix of timewise and variate-wise attention blocks to judiciously model interactions in high-dimensional multivariate data.
  - Student-T Mixture Model Head: An output layer that models the predictive distribution
    as a mixture of Student-T distributions to better capture the complex, heavy-tailed nature of
    observability metrics.
  - Composite Robust Loss: A hybrid loss function combining negative log-likelihood with a robust point-wise loss to stabilize training in the presence of outliers.
  - Forecasting Type: A universal, zero-shot, probabilistic forecaster for multivariate time series.

#### 584 **B.1.11 MOMENT**

MOMENT (Multi-task, Open-source, Foundation Model for Time-series) is a family of open-source foundation models from Carnegie Mellon University designed for general-purpose time series analysis.

The models are built on a Transformer encoder architecture and are pretrained on a large, diverse collection of public time series called the "Time Series Pile." A key characteristic of MOMENT is its versatility; it is designed to serve as a building block for a wide range of downstream tasks, including forecasting, classification, anomaly detection, and imputation, often with minimal task-specific fine-tuning. [26]

- **Input:** Accepts a univariate time series of a fixed length. Multivariate time series are handled by treating each channel independently.
  - Output: Produces a reconstructed version of the input time series. This output can be adapted
    for various downstream tasks, such as generating forecasts by masking future values or extracting
    embeddings for classification.
  - **Architecture:** A standard Transformer encoder that processes time series data in patches.
    - Masked Pre-training: The model is pretrained using a masked time series prediction task. It learns to reconstruct randomly masked patches of the input time series, enabling it to learn robust representations.
    - Patching: The input time series is segmented into non-overlapping patches, which are then linearly projected into embeddings for the Transformer.
    - Lightweight Prediction Head: A simple linear layer is used to reconstruct the time series from the Transformer's output embeddings. This head can be easily replaced or adapted for different downstream tasks.
  - Forecasting Type: A universal foundation model for general time series analysis. It can be used for zero-shot or few-shot forecasting (point-based), classification, anomaly detection, and imputation.

#### B.1.12 ARIMA

594

595

596

597

598

599

600

601 602

603

604 605

606

607

608

615

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

The Autoregressive Integrated Moving Average (ARIMA) model is a class of statistical models for analyzing and forecasting time series data. It is a generalization of the simpler Autoregressive Moving Average (ARMA) model that can be applied to non-stationary time series. The model's name reflects its three core components: Autoregression (AR), Integrated (I), and Moving Average (MA). These components capture the key temporal structures within the data, such as dependencies on past observations and past forecast errors. [27]

- **Input:** A univariate time series.
- **Output:** A point forecast for future time steps. While classical ARIMA produces point forecasts, probabilistic forecasts can be generated by assuming a distribution for the error term.
- Mathematical Formulation: An ARIMA(p, d, q) model is defined by three parameters: the order of the autoregressive component (p), the degree of differencing (d), and the order of the moving average component (q). The model assumes that the differenced time series,  $\tilde{y}_t = (1-B)^d y_t$ , is stationary, where B is the backshift operator. The formulation for the stationary series  $\tilde{y}_t$  is:

$$\widetilde{\mathbf{y}}_t = c + \sum_{i=1}^p \phi_i \widetilde{\mathbf{y}}_{t-i} + \sum_{j=1}^q \theta_j \epsilon_{t-j} + \epsilon_t \tag{1}$$

where:

- p is the autoregressive order, representing the number of lagged observations included in the model.
- d is the degree of differencing, representing the number of times the raw observations are differenced to achieve stationarity.
- q is the moving average order, representing the size of the moving average window applied to past forecast errors.
- $\phi$  is the vector of autoregressive coefficients.
- $\theta$  is the vector of moving average coefficients.
- c is a constant term.
  - $-\epsilon_t$  is the white noise error term at time t, typically assumed to be drawn from a Gaussian distribution with zero mean.
- Forecasting Type: A statistical model that provides point forecasts. It is often used as a baseline in forecasting tasks. Seasonal variations can be included by using a Seasonal ARIMA (SARIMA) model.

#### **B.1.13** Croston's Method

637

646

647

648

649

650

655

656

658

659

660

661

662

668

669

Croston's method is a forecasting technique specifically designed for intermittent demand time series, which are characterized by sporadic, non-zero values interspersed with periods of zero demand. The method decomposes the original time series into two separate components: the magnitude of the non-zero demand and the time interval between consecutive demands. By forecasting these two components separately using Simple Exponential Smoothing and then combining them, the model provides a more accurate estimate of the mean demand per period compared to standard smoothing methods, which can be biased when applied to intermittent data. [28]

- Input: A univariate time series with intermittent demand.
  - Output: A point forecast for the average demand per period.
  - Mathematical Formulation: The method maintains and updates two estimates: one for the non-zero demand size  $(\widehat{z})$  and one for the interval between demands  $(\widehat{p})$ . Let  $y_t$  be the demand at time t, and let q be the time elapsed since the last demand. The updates occur only when a non-zero demand is observed  $(y_t > 0)$ :

$$\widehat{\boldsymbol{z}}_t = \widehat{\boldsymbol{z}}_{t-1} + \alpha(\boldsymbol{y}_t - \widehat{\boldsymbol{z}}_{t-1}) \tag{2}$$

$$\widehat{\boldsymbol{p}}_t = \widehat{\boldsymbol{p}}_{t-1} + \alpha(q - \widehat{\boldsymbol{p}}_{t-1}) \tag{3}$$

If demand at time t is zero, the estimates are not updated  $(\hat{z}_t = \hat{z}_{t-1}, \hat{p}_t = \hat{p}_{t-1})$  and the interval counter q is incremented. After a demand occurs, q is reset to 1. The final forecast for the mean demand per period,  $\hat{y}_t$ , is the ratio of the two smoothed components:

$$\widehat{\boldsymbol{y}}_t = \frac{\widehat{\boldsymbol{z}}_t}{\widehat{\boldsymbol{p}}_t} \tag{4}$$

where  $\alpha$  is the smoothing parameter.

• Forecasting Type: A statistical model for point forecasting, specialized for intermittent or "lumpy" demand patterns.

#### 657 B.1.14 Holt-Winters Exponential Smoothing

Holt-Winters is an extension of exponential smoothing that explicitly models trend and seasonality. It is a widely used statistical method for forecasting time series data that exhibit these components. The method operates by applying exponential smoothing to three components: the level, the trend, and the seasonality. There are two primary variations of the model, additive and multiplicative, which differ in how they incorporate the seasonal component. [5]

- Input: A univariate time series with trend and seasonality.
- Output: A point forecast for future time steps.
- Mathematical Formulation: The model provides separate updating equations for the level  $(\widehat{l}_t)$ , trend  $(\widehat{b}_t)$ , and seasonal  $(\widehat{s}_t)$  components, using smoothing parameters  $\alpha$ ,  $\beta$ , and  $\gamma$ , respectively. Let L be the length of the seasonal period.

**Additive Method:** Used when the seasonal variation is roughly constant throughout the series.

Level: 
$$\hat{l}_t = \alpha(\boldsymbol{y}_t - \hat{s}_{t-L}) + (1 - \alpha)(\hat{l}_{t-1} + \hat{b}_{t-1})$$
 (5)

Trend: 
$$\hat{b}_t = \beta(\hat{l}_t - \hat{l}_{t-1}) + (1 - \beta)\hat{b}_{t-1}$$
 (6)

Seasonality: 
$$\hat{s}_t = \gamma(\boldsymbol{y}_t - \hat{l}_t) + (1 - \gamma)\hat{s}_{t-L}$$
 (7)

The forecast for h steps ahead is given by:

$$\widehat{\boldsymbol{y}}_{t+h|t} = \widehat{l}_t + h\widehat{b}_t + \widehat{s}_{t-L+h_L^+} \quad \text{where } h_L^+ = \lfloor (h-1) \pmod{L} \rfloor + 1 \tag{8}$$

Multiplicative Method: Used when the seasonal variation changes in proportion to the level of the series.

Level: 
$$\hat{l}_t = \alpha \left( \frac{\mathbf{y}_t}{\hat{s}_{t-L}} \right) + (1 - \alpha)(\hat{l}_{t-1} + \hat{b}_{t-1})$$
 (9)

Trend: 
$$\hat{b}_t = \beta(\hat{l}_t - \hat{l}_{t-1}) + (1 - \beta)\hat{b}_{t-1}$$
 (10)

Seasonality: 
$$\hat{s}_t = \gamma \left( \frac{\mathbf{y}_t}{\hat{l}_t} \right) + (1 - \gamma)\hat{s}_{t-L}$$
 (11)

The forecast for h steps ahead is given by:

$$\widehat{\boldsymbol{y}}_{t+h|t} = (\widehat{l}_t + h\widehat{b}_t)\widehat{s}_{t-L+h_L^+} \quad \text{where } h_L^+ = \lfloor (h-1) \pmod{L} \rfloor + 1 \tag{12}$$

• Forecasting Type: A statistical model for point forecasting that can handle various combinations of trend and seasonality.

### 675 B.1.15 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture specifically designed to address the vanishing gradient problem, allowing it to learn and remember long-term dependencies in sequential data. Unlike traditional neural networks, LSTMs have internal mechanisms called "gates" that regulate the flow of information. These gates enable the network to selectively remember or forget information over long periods, making it particularly well-suited for time series forecasting. [27]

- Input: A sequence of historical time series observations.
  - Output: A point forecast for one or more future time steps.
  - Mathematical Formulation: The core of an LSTM unit is its cell state,  $\hat{c}_t$ , which acts as a memory. The flow of information into and out of the cell is controlled by three gates: the forget gate  $(f_t)$ , the input gate  $(i_t)$ , and the output gate  $(o_t)$ . At each time step t, these gates update the cell state and produce a hidden state,  $\hat{h}_t$ .

Forget Gate: 
$$\mathbf{f}_t = \boldsymbol{\sigma}(\mathbf{w}_f \cdot [\widehat{\mathbf{h}}_{t-1}, \mathbf{y}_t] + b_f)$$
 (13)

Input Gate: 
$$i_t = \sigma(\boldsymbol{w}_i \cdot [\hat{\boldsymbol{h}}_{t-1}, \boldsymbol{y}_t] + b_i)$$
 (14)

Candidate State: 
$$\widetilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{w}_c \cdot [\widehat{\boldsymbol{h}}_{t-1}, \boldsymbol{y}_t] + b_c)$$
 (15)

Cell State Update: 
$$\hat{c}_t = f_t \odot \hat{c}_{t-1} + i_t \odot \tilde{c}_t$$
 (16)

Output Gate: 
$$o_t = \sigma(\boldsymbol{w}_o \cdot [\hat{\boldsymbol{h}}_{t-1}, \boldsymbol{y}_t] + b_o)$$
 (17)

Hidden State Update: 
$$\hat{\boldsymbol{h}}_t = \boldsymbol{o}_t \odot \tanh(\hat{\boldsymbol{c}}_t)$$
 (18)

where W and b are the weight matrices and bias vectors for each gate,  $\sigma$  is the sigmoid function, and  $\odot$  denotes element-wise multiplication. The final prediction is typically generated by passing the hidden state  $\hat{h}_t$  through a dense output layer.

• Forecasting Type: A neural network model for point forecasting that can capture complex non-linear patterns in time series data.

#### 693 **B.1.16 Prophet**

683

684

685

686

687

688

689

690

691

692

698

699

702

703

706

Prophet is a forecasting procedure developed by Meta, based on a decomposable time series model. It is designed to be robust to missing data and shifts in the trend, and it typically handles holidays and seasonal effects well. The model fits an additive model with components for trend, seasonality, and holidays. [8]

- **Input:** A univariate time series with timestamps.
- Output: A point forecast, along with uncertainty intervals.
- Mathematical Formulation: The Prophet model is specified as a sum of three components:

$$\mathbf{y}_t = g(t) + s(t) + h(t) + \epsilon_t \tag{19}$$

701 where:

- g(t) is the trend component, which is modeled as either a piecewise linear or logistic growth function. This allows the model to capture non-periodic changes in the time series.
- -s(t) is the seasonality component, which models periodic changes (e.g., yearly, weekly, daily). It is approximated by a Fourier series:

$$s(t) = \sum_{n=1}^{N} \left( a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$
 (20)

where P is the period of the seasonality (e.g., 365.25 for yearly).

- h(t) is the holiday component, which represents the effects of holidays and special events. It is modeled as a sum of indicator functions for each holiday.
  - $\epsilon_t$  is the error term, assumed to be normally distributed white noise.
- Forecasting Type: A decomposable statistical model for point and probabilistic forecasting, particularly effective for business time series with strong seasonal patterns and holiday effects.

#### 712 B.1.17 Random Forest

709

724

725

726

727

- Random Forest is an ensemble machine learning model that operates by constructing a multitude of decision trees at training time. For time series forecasting, it is applied as a regression model to a featurized dataset. By fitting numerous trees on various sub-samples of the data and employing randomness in feature selection, it improves predictive accuracy and controls over-fitting. The final prediction is an average of the outputs from all individual trees, making the model robust and capable of capturing complex, non-linear relationships. [29]
- Input: A feature matrix X where rows are observations and columns are engineered features (e.g., lags, calendar variables), and a corresponding target vector y.
- Output: A point forecast for each input feature vector.
- Architecture and Formulation: A Random Forest is an ensemble of B decision trees. Its predictive power comes from two sources of randomness introduced during training:
  - Bagging (Bootstrap Aggregating): Each individual tree,  $f_b$ , is trained on a bootstrap sample (a random sample drawn with replacement) from the original training dataset.
  - Feature Randomness: When splitting a node in a tree, the algorithm considers only a random subset of the total features, which decorrelates the trees in the forest.
- For a new input feature vector x, the forecast is the average of the predictions from all B trees in the ensemble:

$$\widehat{\boldsymbol{y}}(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} f_b(\boldsymbol{x})$$
 (21)

• Forecasting Type: An ensemble machine learning model for point forecasting. It is non-parametric and highly effective at modeling non-linear relationships between features and the target variable.

#### 2 B.1.18 Seasonal Naive

- The Seasonal Naive model is a simple yet effective baseline method for forecasting time series with a strong seasonal component. Its core principle is that the forecast for a future period is equal to the last observed value from the same season. For example, the forecast for this Monday would be the value from last Monday. Despite its simplicity, it serves as a crucial benchmark for more complex models. [30]
- **Input:** A univariate time series with a known seasonal period.
- Output: A point forecast for future time steps.
- Mathematical Formulation: The forecast for h steps ahead from time t, denoted  $\widehat{y}_{t+h|t}$ , is the last observed value from the corresponding season. Let L be the seasonal period (e.g., L=7 for daily data with weekly seasonality). The forecast is given by:

$$\widehat{\boldsymbol{y}}_{t+h|t} = \boldsymbol{y}_{t+h-L\cdot k} \tag{22}$$

- where  $k = \lceil h/L \rceil$  is an integer that ensures the lagged time index refers to the most recent observation from the target season. For a one-season-ahead forecast (h = L), this simplifies to  $\widehat{y}_{t+L|t} = y_t$ .
  - Forecasting Type: A simple statistical baseline for seasonal point forecasting.

#### **B.1.19** Support Vector Regression (SVR)

747

756

757

758

759

764

765

767

769

770

771

772

773

777

778

779

780

781 782

783

784

Support Vector Regression (SVR) is a supervised learning algorithm that extends the principles of Support Vector Machines (SVMs) to regression problems. Instead of finding a hyperplane that separates classes, SVR aims to find a function that deviates from the target values by a value no greater than a specified margin,  $\epsilon$ , for as many of the training points as possible. It is particularly effective in high-dimensional spaces and is robust to some outliers due to its use of an  $\epsilon$ -insensitive loss function, which ignores errors within this margin. [31]

- Input: A feature matrix X and a corresponding target vector y.
- Output: A point forecast for each input feature vector.
  - Mathematical Formulation: The goal of SVR is to find a function  $f(x) = w^T x + b$  that is as "flat" as possible. This is achieved by minimizing the norm of the weight vector,  $||w||^2$ . The optimization problem is formulated to tolerate errors up to a margin  $\epsilon$  while penalizing points that fall outside this margin using slack variables  $\xi_i$  and  $\xi_i^*$ . The primal optimization problem is:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2} ||\boldsymbol{w}||^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*)$$
 (23)

subject to the constraints:

$$\mathbf{y}_i - (\mathbf{w}^T \mathbf{x}_i + b) \le \epsilon + \xi_i \tag{24}$$

$$(\boldsymbol{w}^T \boldsymbol{x}_i + b) - \boldsymbol{y}_i \le \epsilon + \xi_i^* \tag{25}$$

$$\xi_i, \xi_i^* \ge 0 \tag{26}$$

where C is a regularization parameter that controls the trade-off between the flatness of the model and the amount up to which deviations larger than  $\epsilon$  are tolerated. Non-linear relationships are handled by mapping the data to a higher-dimensional space using a kernel function.

• Forecasting Type: A machine learning model for point forecasting that is robust to some outliers and effective in high-dimensional feature spaces.

#### 766 B.1.20 Theta Method

The Theta method is a statistical forecasting technique based on the concept of decomposition. It models a time series by breaking it down into two components, or "theta lines." The first line represents the long-term trend of the data, while the second line is constructed to capture the short-term dynamics by modifying the curvature of the original series. These two lines are forecasted independently and then combined to produce the final forecast. The standard Theta model has been shown to be equivalent to Simple Exponential Smoothing with a drift term. [32]

- Input: A univariate time series.
- Output: A point forecast for future time steps.
- Mathematical Formulation: The method decomposes the original time series,  $y_t$ , into two theta lines.
  - Line 1 (Trend Component): This line is the simple linear trend fitted to the data, which is
    found by ordinary least squares regression:

$$\widetilde{\boldsymbol{y}}_t^{(1)} = \widehat{\boldsymbol{a}} + \widehat{\boldsymbol{b}}t\tag{27}$$

This line is extrapolated linearly to produce its forecast.

- Line 2 (Short-term Component): This line is constructed by modifying the original series with a coefficient  $\theta$ . A common and effective choice is  $\theta=2$ , which doubles the local curvatures of the series. This modified series,  $\widetilde{y}_t^{(2)}$ , is then forecasted using Simple Exponential Smoothing (SES).

The final forecast,  $\hat{y}_{t+h}$ , is a simple average of the forecasts from the two lines:

$$\widehat{y}_{t+h} = \frac{1}{2} \left( \widehat{y}_{t+h}^{(1)} + \widehat{y}_{t+h}^{(2)} \right)$$
 (28)

• **Forecasting Type:** A statistical decomposition model for point forecasting, often used as a strong baseline for its simplicity and performance.

#### 7 **B.1.21 XGBoost**

XGBoost (Extreme Gradient Boosting) is a powerful and efficient implementation of the gradient boosting framework. It is an ensemble model that builds decision trees sequentially, where each new tree is trained to correct the errors made by the previous ones. For time series forecasting, XGBoost is used as a regression model on a featurized dataset, making it highly effective at capturing complex, non-linear relationships between the engineered features (e.g., lags, calendar variables) and the target. [30]

- Input: A feature matrix X and a corresponding target vector y.
- Output: A point forecast for each input feature vector.
- Architecture and Formulation: XGBoost builds an additive model where the final prediction is the sum of the predictions from K decision trees:

$$\widehat{\boldsymbol{y}}_i = \sum_{k=1}^K f_k(\boldsymbol{x}_i) \tag{29}$$

The trees are added one at a time in a greedy fashion. The k-th tree,  $f_k$ , is chosen to minimize a regularized objective function:

$$\mathcal{L}^{(k)} = \sum_{i=1}^{n} l(\boldsymbol{y}_i, \widehat{\boldsymbol{y}}_i^{(k-1)} + f_k(\boldsymbol{x}_i)) + \Omega(f_k)$$
(30)

where l is a differentiable loss function,  $\widehat{y}_i^{(k-1)}$  is the prediction from the first k-1 trees, and  $\Omega$  is a regularization term that penalizes the complexity of the tree:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{31}$$

Here, T is the number of leaves in the tree,  ${\pmb w}$  is the vector of scores on the leaves, and  $\gamma$  and  $\lambda$  are regularization parameters.

• **Forecasting Type:** An ensemble machine learning model for point forecasting, known for its high performance, speed, and regularization capabilities.

#### **B.2** Benchmark Tasks Included in TempusBench

In this section, we describe the datasets that have been used for each benchmark task. We summarize 807 the dataset used for each benchmark task in Table 2. 808

Table 6: Summary of datasets used for benchmark tasks, separated into univariate and multivariate tasks.

| Benchmark         | Univariate task                                       | Multivariate task                                 |  |  |  |
|-------------------|-------------------------------------------------------|---------------------------------------------------|--|--|--|
| Trend             |                                                       |                                                   |  |  |  |
| Non-stationary    | Software Development Job Postings [33]                | Electricity Consumption [34]                      |  |  |  |
| Decomposition     |                                                       |                                                   |  |  |  |
| Additive          | Synthetically Generated Additive (Appendix B.3)       |                                                   |  |  |  |
| Multiplicative    | Synthetically Generated Multiplicative (Appendix B.4) | _                                                 |  |  |  |
| Frequency         |                                                       |                                                   |  |  |  |
| Seconds           | -                                                     | Utah Drilling [35]                                |  |  |  |
| Minutes           | _                                                     | Historical Stock Data (2003-2024) [36]            |  |  |  |
| Hours             | _                                                     | Madrid Transport Pollution [37]                   |  |  |  |
| Days              | Coinbase Litecoin [38]                                | Gold Price in India [39]                          |  |  |  |
| Weeks             | Federal Funds Effective Rate [40]                     | _                                                 |  |  |  |
| Months            | Inventories to Sales Ratio [41]                       | Airlines Baggage Complains [42]                   |  |  |  |
| Quarters          | German House Prices [43]                              | _                                                 |  |  |  |
| Years             | Personal Consumption Expenditures [44]                | _                                                 |  |  |  |
|                   | Seasonality                                           |                                                   |  |  |  |
| Periodic          | ·                                                     |                                                   |  |  |  |
| Quasiperiodic     |                                                       |                                                   |  |  |  |
|                   | Domain                                                |                                                   |  |  |  |
| Energy            | Room SplitSmart [45]                                  | Room SplitSmart [45]                              |  |  |  |
| Transport         | Madrid BEN pollution [46]                             | Madrid BEN pollution [46]                         |  |  |  |
| Climate           | Delhi Climate [47]                                    | Delhi Climate [47]                                |  |  |  |
| Software          | Software Development Job Postings [33]                | Cyber Attacks on Water Distribution Networks [48] |  |  |  |
| Web               | Web Traffic [49]                                      | _                                                 |  |  |  |
| Sales             | German House Prices [43]                              | Airlines Baggage Complains [42]                   |  |  |  |
| Nature            | Soil Monitoring [50]                                  | Soil Monitoring [50]                              |  |  |  |
| Economics/Finance | Coinbase Litecoin [38]                                | Gold Price in India [39]                          |  |  |  |
| Healthcare        | Employees Health Care [51]                            | NYC Covid Cases [52]                              |  |  |  |
| Manufacturing     | Inventories to Sales Ratio [41]                       | Utah Drilling [35]                                |  |  |  |
| Data sparsity     |                                                       |                                                   |  |  |  |
| Sparse            | Patient Chart [53]                                    | _                                                 |  |  |  |
| Dense             | Chicken Pox [54]                                      | Gold Price in India [39]                          |  |  |  |
| Value type        |                                                       |                                                   |  |  |  |
| Continuous        | Forest Fires [55]                                     | Gold Price in India [39]                          |  |  |  |
| Count             | Occupancy [56]                                        | Madrid BEN pollution [46]                         |  |  |  |
| Binary            | Absenteeism at Work [57]                              |                                                   |  |  |  |
| Categorical       | Online Retail [58]                                    |                                                   |  |  |  |

## **B.3** Synthetic Data: Cyclic Seasonality with Additive Trends

#### **B.3.1** Description 810

This category of synthetic data models a time series that exhibits both a complex seasonal pattern 811 and a persistent, long-term trend. The data is generated using two related methods. Both methods 812

start with a foundational signal that combines multi-frequency sinusoids with a linear trend. The

813

second, more complex method builds upon this foundation by introducing an additional, randomized 814

sinusoidal component to the signal.

In both cases, non-negative noise from an exponential distribution is added to the deterministic signal.

These datasets are ideal for testing a model's ability to identify and separate periodicities from an

underlying linear trend, with the second method providing a more complex seasonal structure.

#### 819 B.3.2 Mathematical Formulation

The generation process for both methods is based on a primary signal,  $y_{\text{base}}(t)$ , which includes seasonal, trend, and offset components:

$$y_{\text{base}}(t) = \underbrace{2\sin(t) + 2\cos\left(\frac{t}{2}\right)}_{\text{Seasonality}} + \underbrace{\frac{1}{4}t}_{\text{Trend}} + \underbrace{\frac{4}{\text{Offset}}}_{\text{Offset}}$$
(32)

Method 1: Fixed Additive Trend In the first method, the true signal,  $y_1(t)$ , is simply the base function.

The final observed value,  $Y_t$ , is this signal plus an additive noise term,  $\epsilon_t$ .

$$Y_t = y_1(t) + \epsilon_t = y_{\text{base}}(t) + \epsilon_t \tag{33}$$

Method 2: Randomized Additive Trend The second method introduces additional complexity. For

each generated time series, a random frequency parameter,  $\alpha$ , is sampled once from a continuous

826 uniform distribution:

$$\alpha \sim U(a,b) \tag{34}$$

In the provided code, this range is fixed from a=0 to b=5. This parameter is used to create an additional sinusoidal component that is added to the base signal. The true signal,  $y_2(t)$ , is therefore:

$$y_2(t) = y_{\text{base}}(t) + \sin(\alpha t) \tag{35}$$

The final observed value,  $Y_t$ , is this enhanced signal plus the noise term:

$$Y_t = y_2(t) + \epsilon_t \tag{36}$$

Noise Model For both methods, the noise term,  $\epsilon_t$ , is drawn from an exponential distribution with a scale parameter  $\beta$ :

$$\epsilon_t \sim \text{Exponential}(\beta)$$
 (37)

#### 832 B.3.3 Adjustable Parameters

The data generation process is controlled by the following parameters.

- Number of Points (num\_points, N): This integer parameter sets the total number of data points, defining the length of the time series.
- Start Time (start\_time, t<sub>0</sub>): This parameter defines the initial time value for the series.
- Noise Scale (noise\_std,  $\beta$ ): This parameter represents the scale (and mean) of the exponential noise distribution. A larger value for  $\beta$  increases the average magnitude of the positive noise added to the base signal.
- Random Frequency (alpha,  $\alpha$ ): (Method 2 only) This parameter is not set by the user but is sampled internally from a uniform distribution U(0,5) for each generated series. It introduces variability in the seasonal component across different datasets created by the second method.

#### 844 B.4 Synthetic Data: Cyclic Seasonality with Multiplicative and Additive Trends

#### 845 B.4.1 Description

834

835

836

837

838

839

840

841

842

843

This category of synthetic data models a time series characterized by a complex interaction of seasonal

components and trends. A key feature is a multiplicative trend, where the amplitude of one of the

8 seasonal components grows exponentially over time. This is combined with another stable seasonal

component and a linear additive trend.

The data is generated using two related methods. The first method uses a fixed, deterministic signal.

The second method introduces additional complexity by adding another sinusoidal component with

a randomized frequency to the base signal. In both cases, non-negative noise from an exponential

distribution is added. These datasets are particularly useful for testing a model's ability to handle

heteroscedasticity, where the variance of the series changes over time, in the presence of other

855 seasonalities and trends.

#### 856 B.4.2 Mathematical Formulation

Both methods are built upon a primary signal,  $y_{\text{base}}(t)$ , which is a composite of several functions:

$$y_{\text{base}}(t) = \underbrace{e^{t/100} \sin(t)}_{\text{Multiplicative Seasonality}} + \underbrace{3\cos\left(\frac{t}{2}\right)}_{\text{Additive Seasonality}} + \underbrace{\frac{1}{2}t}_{\text{Linear Trend}}$$
(38)

Method 1: Fixed Multiplicative Trend In the first method, the true signal,  $y_1(t)$ , is simply the base function. The final observed value,  $Y_t$ , is this signal plus an additive noise term,  $\epsilon_t$ .

$$Y_t = y_1(t) + \epsilon_t = y_{\text{base}}(t) + \epsilon_t \tag{39}$$

860 Method 2: Randomized Additive Component The second method adds another layer of seasonality.

For each generated time series, a random frequency parameter,  $\alpha$ , is sampled once from a continuous

862 uniform distribution:

$$\alpha \sim U(a,b) \tag{40}$$

In the provided code, this range is fixed from a=5 to b=10. The true signal,  $y_2(t)$ , is the base signal plus this new randomized sinusoidal component:

$$y_2(t) = y_{\text{base}}(t) + \sin(\alpha t) \tag{41}$$

The final observed value,  $Y_t$ , is this enhanced signal plus the noise term:

$$Y_t = y_2(t) + \epsilon_t \tag{42}$$

Noise Model For both methods, the noise term,  $\epsilon_t$ , is drawn from an exponential distribution with a scale parameter  $\beta$ :

$$\epsilon_t \sim \text{Exponential}(\beta)$$
 (43)

#### 868 B.4.3 Adjustable Parameters

870

871

872

873

874

875

876

877

878

879

The data generation process is controlled by the following parameters.

- Number of Points (num\_points, N): This integer parameter sets the total number of data points, defining the length of the time series.
- Start Time (start time,  $t_0$ ): This parameter defines the initial time value for the series.
- Noise Scale (noise\_std,  $\beta$ ): This parameter represents the scale (and mean) of the exponential noise distribution. A larger value for  $\beta$  increases the average magnitude of the positive noise added to the base signal.
- Random Frequency (alpha,  $\alpha$ ): (Method 2 only) This parameter is not set by the user but is sampled internally from a uniform distribution U(5,10) for each generated series. It introduces variability in the seasonal component across different datasets created by the second method.

#### C Additional Related Works

Classical time-series forecasting began with statistical models that exploit stochastic structure and domain priors, including ARIMA and its Box–Jenkins methodology [59], exponential-smoothing state-space ETS [60], the Theta method [61], and multivariate VAR models [62]. Deep learning methods later advanced accuracy and scale by learning nonlinear temporal dependencies from large corpora: DeepAR [63], N-BEATS [64], DLinear [15], TiDE [65], TFT [66], PatchTST [67], and iTransformer [68]. Probabilistic forecasters further model predictive distributions, e.g., diffusion-based TimeGrad [69], score-based CSDI for imputation and forecasting [70], and conditional-flow GRU-NVP [71].

**TSFMs.** Inspired by NLP/vision pretraining, TSFMs train on heterogeneous corpora and evaluate in zero/few-shot settings across domains and horizons. Representative models include Moirai [72], Chronos [21], TimesFM [73], Lag-Llama [74], Timer [75], UniTS [76], TTM (Tiny Time Mixers) [77], Moment [26], and multimodal VisionTS [78]. Collectively, they demonstrate strong zero-shot point and probabilistic accuracy on diverse benchmarks while revealing open challenges at long horizons (error accumulation) and at very high frequencies.

**Public datasets and repositories.** Public corpora have underpinned progress from statistical to foundation-model eras. The M-competitions (M3 and M4) provided broad univariate benchmarks across domains and frequencies [6, 7], followed by the retail-demand M5 competition [79]. The Monash Time-Series Forecasting Archive curates a large, standardized repository spanning many domains and sampling granularities [13]. Large-scale pretraining/evaluation collections include LOTSA (released with Moirai) [72], the Chronos corpus with in-domain/zero-shot splits [21], and the diverse univariate corpus aggregated in Lag-Llama [74]. Task-focused collections such as the LTSF suite [15] (e.g., ETT datasets) and broader benchmarks like TFB [14] and ProbTS [80] assemble datasets emphasizing horizon length, covariates, and probabilistic outputs.

**Evaluation frameworks and benchmarks.** Tooling and standardized evaluation have evolved in parallel. Practitioner libraries such as Prophet [81] and sktime [82] offer classical and ML baselines with unified interfaces, while GluonTS [83] and PyTorchTS [84] provide probabilistic deep-learning pipelines. Benchmarking efforts including LTSF [15], BasicTS+ [?], TFB [14], and ProbTS [80] compare statistical, deep, and (in some cases) foundation models, but differ in task taxonomies, splits, and leakage controls. Standardized metrics such as MASE [85] and CRPS [86] enable cross-dataset aggregation of point and probabilistic performance, yet consistent pretraining/evaluation protocols and leakage-free large-scale corpora remain key needs for fair TSFM assessment.

The collective consequence of these issues is a research environment where it is difficult to distinguish genuine methodological advances from circumstantial performance on a narrow, and potentially contaminated, set of tasks. This is particularly damaging for the development of TSFMs. The significant computational and financial resources required to pre-train these models demand a rigorous, fair, and comprehensive evaluation framework to justify their development and guide future research [32]. The current state of affairs falls short of this standard. Indeed, studies have shown that existing TSFMs, often pre-trained on general-purpose academic datasets, can struggle to generalize to the unique and challenging characteristics of specialized domains like observability data [87].

The field has thus reached an inflection point. Progress is no longer primarily limited by our ability to design novel model architectures, but by our inability to reliably and fairly measure their performance. Recognizing this crisis, recent efforts have focused on creating the next generation of evaluation infrastructure. The development of large-scale, standardized benchmarks such as GIFT-Eval and the domain-specific Benchmark of Observability Metrics (BOOM) represent a direct and necessary response [27]. These initiatives introduce carefully curated and decontaminated pre-training and evaluation sets, standardized protocols, and data that reflects the complexity of real-world applications. They treat the benchmark not as a mere dataset, but as a carefully designed scientific instrument [8]. This establishes a clear and urgent research gap: the critical need for a new, large-scale, and meticulously curated public benchmark that can serve as a gold standard for evaluating the next generation of time-series models. Such a contribution is not merely a prerequisite for future research; it is a foundational scientific contribution in its own right, providing the essential infrastructure required to move the field from an era of fragmented claims to one of robust, reproducible, and generalizable progress [26].

Contemporary time-series data seldom conform to the idealized assumptions of stationarity and linearity that underpin classical models. Instead, real-world data streams are characterized by a confluence of complex, interacting properties that present formidable modeling challenges [30].

- Non-Linearity: Perhaps the most fundamental challenge is the prevalence of non-linear relationships. Economic systems, biological processes, and energy grids are governed by complex feedback loops and interactions that cannot be adequately captured by linear models [27]. Traditional methods like Autoregressive Integrated Moving Average (ARIMA) are, by their construction, limited in their ability to model such non-linear dynamics, which is a primary reason for their performance ceiling on complex, real-world problems [30].
- Multi-Regime Behavior: Many time series exhibit structural breaks or distinct operational regimes, where the underlying data-generating process changes over time [5]. Examples include the shift between bull and bear markets in financial data or the different performance characteristics of an industrial machine under varying loads and environmental conditions. A single, global model often fails to capture this complex inner structure, leading to significant predictive errors when the system transitions between regimes [88].
- **Intermittency:** As noted previously, intermittent demand patterns are characterized by a high proportion of zero-valued observations, with non-zero demands occurring sporadically. This dual source of randomness—in both the timing and the magnitude of events—violates the assumptions of continuity and regular sampling inherent in many classical smoothing and regression-based techniques [5].
- Heightened Volatility and Novel Data Sources: The modern data ecosystem is characterized by the emergence of new data sources that introduce unprecedented levels of volatility and complexity [5]. The integration of renewable energy sources into power grids is a prime example, creating load patterns with high-frequency noise and non-stationary behavior that challenge traditional forecasting approaches. A parallel development is the explosion of "observability data" generated by large-scale distributed software and cloud computing systems. This data, which includes metrics on CPU load, network latency, and application error rates, is often characterized by extreme non-stationarity, high dimensionality (thousands of correlated variables), heavy-tailed distributions, and sparsity, posing a unique and difficult set of modeling challenges [5].

#### C.0.1 An Arms Race of Methodological Innovation

The progression of forecasting methodologies can be understood as a direct response to this escalating data complexity. Each new paradigm has sought to overcome the limitations of its predecessors, leading to the current diverse and powerful toolkit available to researchers and practitioners [5].

- The Classical Foundation: The field was built upon a foundation of statistical methods developed primarily in the mid-20th century. Models such as ARIMA and its variants [8], Holt-Winters Exponential Smoothing [5], and the Theta method [32] became the workhorses of the discipline. These models excel at capturing and extrapolating clear patterns of trend and seasonality from univariate time series. Their enduring appeal lies in their statistical rigor, interpretability, and computational efficiency. However, their reliance on strong assumptions about the underlying data-generating process, particularly linearity and stationarity, fundamentally limits their applicability to the more complex data common today [8].
- The Machine Learning Advance: The rise of machine learning in the late 20th and early 21st centuries provided a new set of tools capable of addressing the challenge of non-linearity. Non-parametric models like Support Vector Regression (SVR) [31] offered a principled approach, grounded in statistical learning theory, to model non-linear relationships in high-dimensional spaces via the "kernel trick" [27]. Concurrently, ensemble methods, particularly those based on decision trees like Random Forest and Gradient Boosting (e.g., XGBoost), proved to be exceptionally powerful and robust [30]. By combining the predictions of many weak learners, these models can capture complex, non-linear interactions and have consistently demonstrated state-of-the-art performance in a wide array of forecasting competitions and applications.
- The Deep Learning Revolution: While machine learning ensembles excelled at capturing complex feature interactions, they were not explicitly designed to model the long-range

temporal dependencies inherent in sequential data. This limitation was addressed by the deep learning revolution. Recurrent Neural Networks (RNNs), and more specifically architectures like Long Short-Term Memory (LSTM) networks, were developed with internal memory mechanisms (gates) that allow them to capture and retain information over long sequences [27]. Empirical studies have shown that on complex financial and economic data, LSTMs can significantly outperform classical models like ARIMA by better modeling non-linear temporal dynamics [5]. Following the success of LSTMs, Transformer-based architectures, with their self-attention mechanism, have emerged as the next frontier, offering a powerful alternative for capturing dependencies across time without the sequential processing limitations of RNNs [27].

This co-evolution of data challenges and modeling paradigms, summarized in Table 7, illustrates a clear trajectory towards models of increasing complexity and representational power.

Table 7: The Co-evolution of Time-Series Challenges and Modeling Paradigms [5] [32] [8] [27] [30].

|                      | <u> </u>                                     |                                           | 8 8 131 311 11                                |                                                               |  |
|----------------------|----------------------------------------------|-------------------------------------------|-----------------------------------------------|---------------------------------------------------------------|--|
| Era                  | Primary Data Challenge(s)                    | Dominant Model<br>Paradigm                | Key Models                                    | Inherent Limitations                                          |  |
| Statistical          | Trends, Seasonality,<br>Stationarity         | Time-Domain Statistical Models            | ARIMA, Holt-Winters, Theta                    | Struggle with non-<br>linearity and com-<br>plex dependencies |  |
| Machine<br>Learning  | Non-Linearity, Complex Interactions          | Non-parametric                            | Ensemble Models (SVR, Random Forest, XGBoost) | Limited handling of long-range temporal dependencies          |  |
| Deep<br>Learning     | Long-Range Dependencies, Sequential Patterns | Recurrent,<br>Attention-based<br>Networks | LSTMs, Transformers                           | Data-hungry, computationally intensive, task-specific         |  |
| Foundation<br>Models | Heterogeneity, Scale,<br>Task Generalization | Large Pre-trained<br>Models               | MOMENT,<br>TOTO, Chronos                      | Reliance on massive, curated datasets; evaluation bottleneck  |  |

However, this progression is not a simple linear march where newer, more complex models invariably render older ones obsolete. Empirical evidence reveals a more nuanced reality, one that aligns with the well-known "No Free Lunch" theorem in machine learning. While deep learning models like LSTMs have been shown to decisively outperform ARIMA on certain complex datasets, recent large-scale studies have also found that in zero-shot or limited-supervision settings, simpler statistical methods often outperform sophisticated deep learning models [27]. Furthermore, in production environments like large-scale observability systems, classical models remain prevalent due to the operational infeasibility of training and maintaining millions of distinct, complex neural network models [5]. This apparent contradiction is not a flaw in the research, but rather a reflection of a fundamental truth: the performance of any given forecasting model is highly contingent on the specific characteristics of the data, the length of the forecast horizon, the availability of computational resources, and the degree of supervision. This recognition implies that the central problem in the field is not merely the invention of more powerful algorithms, but the development of a deeper, more systematic understanding of the complex performance landscape that governs the interaction between data characteristics and model architectures [27].

#### C.1 The New Frontier: Pre-trained Foundation Models for Time Series

In response to the challenges of data heterogeneity and the high cost of developing task-specific models, the field is currently undergoing another paradigm shift, mirroring recent transformations in natural language processing and computer vision: the move towards large, pre-trained Time-Series Foundation Models (TSFMs). This new frontier aims to leverage the power of large-scale, self-supervised learning to create general-purpose models that can be adapted to a wide range of downstream tasks with minimal fine-tuning [27].

The core premise of the foundation model paradigm is to pre-train a single, high-capacity model (typically a Transformer) on a massive and diverse corpus of unlabeled data. This process allows the model to learn a rich, generalizable representation of temporal patterns. Subsequently, this pre-trained model can serve as a powerful building block for various downstream applications, including long- and short-horizon forecasting, time-series classification, anomaly detection, and missing value imputation. Models such as MOMENT, Chronos, and TOTO are at the vanguard of this movement. They are designed to be effective "out-of-the-box," providing strong zero-shot or few-shot performance without the need for extensive task-specific training [26] [25]. This approach holds particular promise for domains like observability, where the sheer scale and diversity of time series—often numbering in the millions or billions—make the traditional approach of training one model per series operationally intractable [30].

The primary enabler of this paradigm, and simultaneously its greatest bottleneck, is the availability of data. The success of foundation models in other domains was built on the existence of vast, cohesive, and publicly accessible datasets like The Pile for text and ImageNet for vision [26]. The time-series domain, by contrast, has historically been characterized by a fragmented landscape of smaller, scattered, and task-specific public datasets [5]. This data scarcity has been a major impediment to large-scale pre-training. To overcome this, pioneering research efforts have begun the monumental task of data curation. The creators of MOMENT compiled *The Time Series Pile*, a large collection of public repositories, while the TOTO model was pre-trained on a corpus containing a mixture of public, synthetic, and large-scale proprietary observability data, resulting in a dataset 4 to 10 times larger than those used for other leading TSFMs [25].

This focus on data curation signals a significant maturation of the field. In earlier eras, the primary axis of innovation was model architecture—for example, the design of the gating mechanisms in an LSTM or a novel attention variant in a Transformer [27]. The advent of the TSFM paradigm, however, shifts the research bottleneck. While architectural innovation remains important, the most critical and scientifically challenging work is now increasingly centered on the curation of massive, diverse, and clean datasets, and on the development of robust frameworks for evaluating the models trained on them. The value proposition of a new TSFM is now as much about the data it was trained on and the benchmark it was tested against as it is about its internal architecture. This implies that the most impactful contributions in this new era may not be designing a marginally better model, but rather creating the foundational data and evaluation infrastructure that enables the entire field to advance [8].