Efficient Model Selection for Time Series Forecasting via LLMs

Anonymous ACL submission

Abstract

Model selection is a critical step in time series forecasting, traditionally requiring extensive performance evaluations across various datasets. Meta-learning approaches aim to automate this process, but they typically depend on a pre-constructed performance matrix, which is expensive and time-consuming to build and maintain. In this work, we propose to leverage Large Language Models (LLMs) as a lightweight alternative for model selection. Our method eliminates the need for an explicit performance matrix by utilizing the inherent knowledge and reasoning capabilities of LLMs. Through extensive experiments on over 320 diverse datasets with Llama, GPT, and Gemini, we demonstrate that our approach outperforms traditional meta-learning techniques and heuristic baselines, while significantly reducing computational overhead. These findings underscore the potential of LLMs in efficient model selection for time series forecasting.

1 Introduction

Time series forecasting plays a crucial role in a wide range of real-world applications, enabling informed decision-making and strategic planning across various domains, including finance (Sezer et al., 2020), healthcare (Bui et al., 2018), software monitoring (Sun et al., 2023), energy (Chou and Tran, 2018), retail (Fildes et al., 2022), and weather prediction (Han et al., 2024). Selecting an appropriate forecasting model is often a labor-intensive process requiring domain expertise and extensive computational resources. Traditional time series forecasting methods typically require substantial domain expertise and manual efforts in model design, feature engineering, and hyperparameter tuning. Recent advances in transformer-based forecasting models, such as FEDformer (Zhou et al., 2022), iTransformer (Liu et al., 2024), Chronos (Ansari et al., 2024), ChronosX (Arango et al., 2025), and Time-MoE (Shi et al., 2025), have shown strong performance across benchmarks by modeling temporal dependencies more effectively. However, even with powerful backbone models, no single learning strategy could consistently outperform the others across all forecasting tasks, due to the inherent diversity of time series data (Abdallah et al., 2022). Consequently, these methods often fail to deliver high-quality predictions across diverse application domains. A straightforward but naïve solution would be evaluating the performances for thousands of models on a given dataset to identify the most suitable one. However, such an approach is impractical due to the excessive computational cost and training time required for model evaluation on each new dataset.

To address the impracticality of exhaustively evaluating all models for each new dataset, meta-learning has recently gained great popularity in applications demanding model selection such as anomaly detection and classification (Zhao et al., 2021), graph learning (Park et al., 2023), and recommendation (Wang et al., 2022), especially for forecasting (Abdallah et al., 2022), which could quickly infer the best forecasting model after training on the models' performances on historical datasets and the time-series meta-features of these datasets. Even though Abdallah et al. (2022) selects the best performing forecasting algorithm and its associated hyper-parameters with a 42× median inference time reduction averaged across all datasets compared to the naïve approach, nearly all state-of-the-art meta-learning approaches still require the construction of a large performance matrix, consisting of evaluations of hundreds or even thousands of models across a vast collection of forecasting datasets. This performance matrix, while crucial for traditional meta-learning-based model selection, is extremely costly and time-consuming to obtain in practice. Each dataset-model pair must be exhaustively evaluated, which demands signifi-

cant computational resources and time. In addition, it requires continuous updates to accommodate emerging state-of-the-art models and new datasets. Furthermore, this matrix is typically used in conjunction with a carefully engineered meta-feature vector extracted from each time-series dataset to train a meta-learning model that can generalize and infer the best model for new forecasting tasks.

In this work, we propose an alternative paradigm: leveraging LLMs to directly select forecasting models without constructing an explicit performance matrix which requires substantial computational resources and time to build. We assess the effectiveness of our method through extensive experiments on over 320 diverse datasets and demonstrate that our method outperforms strategies such as directly selecting popular methods and even traditional meta-learning approaches(Kadioglu et al., 2010) that rely on a costly performance matrix during training and require retraining when new models or datasets are introduced.

Summary of Main Contributions. The key contributions of this work are as follows:

- **LLM-Driven Model Selection for Time-Series Forecasting.** To the best of our knowledge, this work is the first to investigate the use of LLMs for selecting the most suitable time series forecasting model. By evaluating multiple LLMs with various prompt designs, we demonstrate that our LLM-based selection method consistently outperforms both popular forecasting models and traditional meta-learning approaches.
- **Computational Efficiency in Training and Inference.** Unlike conventional model selection techniques that require training and evaluation of multiple forecasting models and the costly pre-computed performance matrix required in traditional meta-learning, our approach leverages LLMs to infer the optimal model and hyperparameters instantly. This results in a significant reduction in computational overhead, making the method highly scalable and efficient for real-world forecasting applications.
- **Extensive Experiments.** We conduct extensive experiments including an ablation study to analyze the impact of incorporating meta-features and CoT reasoning in prompts across different LLMs. The findings offer insights into effective prompt design strategies, guiding future improvements in LLM-driven model selection for time-series forecasting.

2 Related Work

Model Selection in Time Series Forecasting. Model selection in time series forecasting has evolved through various methodologies, encompassing traditional statistical approaches, meta-learning techniques, and emerging LLMs.

Traditional methods often rely on statistical criteria to choose the most suitable forecasting model. For instance, the average rank method evaluates multiple models across different datasets, selecting the one with the lowest average rank based on performance metrics (Cerqueira et al., 2022). While straightforward, these methods can be computationally intensive and may not generalize well across diverse time series data. To overcome these limitations, Lemke and Gabrys (2010) explored meta-learning strategies that utilize characteristics of time series data to predict the performance of various forecasting models, facilitating more efficient and accurate model selection. Similarly, Prudêncio and Ludermir (2004) investigated meta-learning techniques to rank and select time series models based on extracted meta-features, demonstrating improved forecasting accuracy. Recently, Abdallah et al. (2022) have also demonstrated that meta-learning can be used to infer the best model given dataset characteristics and model space without needing an exhaustive evaluation of all existing models on a new dataset. However, these approaches still require constructing performance matrix that capture the evaluation results of all models across all datasets, which is computationally expensive and time-consuming.

LLMs for Time Series Forecasting. The integration of LLMs into time series forecasting has garnered significant attention, with recent studies exploring their potential for model selection and prediction tasks. Jin et al. (2024) introduced Time-LLM, a framework that reprograms LLMs for time series forecasting by aligning time series data with natural language inputs. Gruver et al. (2024) demonstrated that LLMs, such as GPT-3 and Llama-2, can perform zero-shot time series forecasting by encoding time series as sequences of numerical digits, framing forecasting as a next-token prediction task. Cao et al. (2024) introduced an interpretable prompt-tuning-based generative transformer for time series representation learning. Zhang et al. (2024) provided a comprehensive survey on the application of LLMs in time series analysis, highlighting their potential to enhance

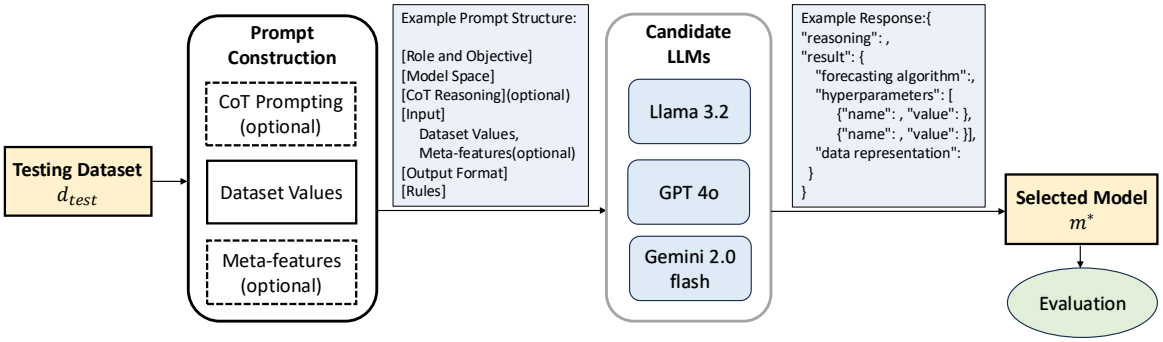

Figure 1: An overview of model selection via LLMs.

forecasting performance across various domains. However, these studies differ from our approach as they employ LLMs directly as forecasting models for new datasets, whereas our work focuses on leveraging LLMs for model selection. Specifically, we demonstrate how LLMs can effectively identify the most suitable forecasting model for time series data to achieve optimal predictive performance.

Prompting. Prompting has emerged as the primary approach for tailoring language models to various downstream applications. CoT prompting improves multi-step reasoning by incorporating intermediate reasoning steps into the prompt, leading to better performance on complex tasks (Wei et al., 2023). Surveys on prompt design strategies provide comprehensive overviews of techniques such as manual design and optimization algorithms, emphasizing their impact on LLM performance across diverse tasks (Li, 2023). These developments underscore the critical role of prompt engineering in leveraging LLMs for complex reasoning and decision-making tasks.

3 Methodology

3.1 Overview

The model selection task for time series forecasting is formulated as a mapping from dataset-based prompts to candidate forecasting models. Let $S : \mathcal{P} \to \mathcal{M}$, where $\mathcal{P}$ denotes the set of possible prompts and $\mathcal{M}$ represents the space of candidate forecasting models. For each dataset $d_i \in \mathcal{D}$, the process comprises three components:

- **Prompt Construction**: Construct a prompt $p_i \in \mathcal{P}$ from d_i using one of the predefined prompt templates.
- **LLM-Based Model Selection**: The prompt p_i is submitted to an LLM to obtain the recommended model $m_i = S(p_i)$, where $m_i \in \mathcal{M}$.
- **Forecasting and Evaluation**: Apply m_i to d_i to produce forecasts and evaluate performance using appropriate metrics.

Problem Statement. Given any dataset d_{i} (i.e., unseen time series forecasting data), select a model $m \in \mathcal{M}$ to employ on that dataset.

3.2 Prompt Construction

We designed four distinct prompt structures, each varying in the inclusion of meta-features and CoT reasoning. The detailed structure of our prompts is illustrated in Appendix.

- **Dataset Values Only**: Providing raw time series data.
- **Dataset Values and Meta-Features**: Combining raw data with pre-computed meta-features. Details of meta-features are available in Appendix.
- **Dataset Values with CoT**: Including raw data along with a step-by-step reasoning instruction in the prompt to guide the LLM.
- **Dataset Values and Meta-Features with CoT**: Integrating raw data, meta-features, and CoT reasoning.

3.3 Model Selection

The model space is denoted as: $\mathcal{M} = \{m_1, m_2, \dots\}$. Each model $m_i \in \mathcal{M}$ is given by the tuple $m_i = (a_i, h_i, g_i(\cdot))$, where a_i is the forecasting algorithm, h_i is the hyper-parameter vector associated with a_i, and $g_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is the time-series data representation (e.g., raw, exponential smoothing).

Unlike traditional meta-learning approaches that operate on a predefined, discrete model space, our method allows for an infinite and continuous model

Algorithm 1 Model Selection via LLMs

Input: Time series dataset d_{i}, model space $\mathcal{M}$, prompt template $\mathcal{P}$
Output: Selected forecasting model m^*

Step 1: Prompt Construction
Get dataset values X_{i} from d_{i}
if Meta-features are included **then**
  Include meta-features F_{i} of d_{i}
end if
if CoT is included **then**
  Incorporate reasoning steps into prompt
end if
Generate prompt p using:

$$p = \text{Format}(X_{\text{i}}, [F_{\text{i}}], [\text{CoT}])$$

Step 2: Query LLM for Model Selection
Obtain selected model $m^* = (a^*, h^*, g^*)$ by querying the LLM:

$$m^* = S(p)$$

where a^* is the forecasting algorithm, h^* is the hyperparameter set, and g^* is the data representation.

Step 3: Forecasting and Evaluation
Apply m^* to generate forecasts for d_{i}
Compute performance metrics
Return: Selected model m^*

space, where hyperparameters h_i can take any real-valued configuration.

3.4 Comparison with Meta-Learning

Meta-learning methods typically rely on an extensive performance matrix: $\mathbf{P} \in \mathbb{R}^{n \times m}$ where $\mathbf{P}_{i,j}$ represents the performance of model M_j on dataset D_i. This matrix is computationally expensive to construct and is essential for training meta-learners. In contrast, our approach eliminates the need for:

- **Explicit performance matrix.** Our method does not require historical model-dataset performance mappings.
- **Feature engineering.** While meta-learners depend on carefully designed meta-features, our LLM-based selection can operate without them.
- **Fixed model spaces.** Our method does not restrict selection to a predefined set of models and hyperparameters.

4 Experiments

We evaluate our LLM-based model selection approach through a series of experiments designed to address the following research questions:

1. Does employing LLMs for time-series forecasting model selection improve performance compared to not using model selection or other techniques like meta-learners?
2. How much reduction in training and inference time do LLM-based methods achieve over the naïve approach, and what is the associated token cost for model selection?
3. To what extent do meta-features and CoT prompting contribute to model selection performance, efficiency, and token usage?

4.1 Experiment Settings

Dataset Source. We use the same dataset as (Abdallah et al., 2022), which consists of 321 forecasting datasets spanning various application domains, including finance, IoT, energy, and storage, such as GDP, population and global Carbon dioxide levels. For each dataset, we randomly sample time windows of fixed length ($= 16$) as our evaluation samples.

Evaluation Metrics. Our evaluation focuses on two primary metrics: hit@k accuracy and average Mean Squared Error (MSE). Hit@k accuracy quantifies whether the selected model ranks among the top k models based on ground truth performance, while MSE measures the forecast error magnitude. Formally, hit@k accuracy is defined as:

$$\text{hit@}k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(M_i^* \in \mathcal{M}_{\text{ranked}}^k(D_i)\right) \quad (1)$$

where $\mathcal{M}_{\text{ranked}}(D_i)$ denotes the set of models ranked by their performance for a given dataset D_i, $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if M_i^* is within the top k models and 0 otherwise, and N is the total number of test datasets. In addition, we record training and inference time, as well as token usage, to assess the computational efficiency and resource overhead of the approaches. To make it fair to compare, we adopt the same model space as our baselines $\mathcal{M}$ from Abdallah et al. (2022), which comprises 322 unique models (see Table 6 in Appendix for the complete list). This model space pairs seven state-of-the-art time-series forecasting algorithms with their corresponding hyperparameters and various data representation methods. In

Methods	hit@1 Accuracy ↑	hit@5 Accuracy ↑	hit@10 Accuracy ↑	hit@50 Accuracy ↑
Random selection	0.31	1.25	3.63	14.75
Popular Selection	0	1.33	3.77	19.94
$ISAC^*$	0.82	2.67	4.10	11.45
MLP^*	0.62	1.13	4.51	22.25
Ours-Llama3.2				
w. data	0.83	3.84	6.65	26.27
w. data+CoT	0.62	3.12	5.82	26.69
w. data+meta_features	**1.14**	**4.47**	**7.27**	**29.60**
w. data+meta_features+CoT	1.14	3.43	6.44	25.96
Ours-GPT4o				
w. data	0.21	2.39	4.47	21.39
w. data+CoT	0.10	1.25	4.36	21.39
w. data+meta_features	0.62	2.39	4.88	20.56
w. data+meta_features+CoT	0.52	2.60	4.88	21.91
Ours-Gemini2.0 flash				
w. data	0.21	0.62	3.53	20.77
w. data+CoT	0.31	0.93	2.91	19.94
w. data+meta_features	0	1.04	2.80	20.87
w. data+meta_features+CoT	0.21	1.35	3.01	17.13

Table 1: hit@k Accuracy(%, the higher (↑), the better) comparison of LLMs against different baselines. $*$ denotes meta-learning methods which utilized performance matrix during training. All others don't require performance matrix.

addition, we utilize the precomputed performance matrix from Abdallah et al. (2022) to evaluate our proposed methods.

4.2 LLMs and Hardware

To evaluate the effectiveness of different LLMs in the forecasting model selection task, we conducted experiments using three competitive models: Llama 3.2-3B-Instruct (MetaAI, 2024), GPT-4o (OpenAI, 2024), and Gemini 2.0 Flash (Google, 2024). In experiments with Llama 3.2-3B-Instruct, we utilized a single NVIDIA A100 GPU with 80GB of memory. GPT-4o and Gemini 2.0 Flash are accessed via API.

4.3 Baselines

We compare our proposed approach against various baseline methods. They fall into two categories: methods that do not perform explicit model selection and meta-learning-based approaches.

No Dynamic Model Selection. In this category, the same fixed model configuration or an ensemble of all models is applied, hence no model is dynamically selected. We consider the following strategies:

1. **Random Model.** A model configuration is randomly selected from the model space for each time-series dataset.
2. **Popular Model.** The most widely used forecasting model, Prophet (Taylor and Letham, 2017), is selected given its strong community support (e.g., over 19k stars on GitHub).
3. **SOTA Model.** We consider seven state-of-the-art forecasting models. For each model, we create multiple configurations by adjusting hyperparameters and data representations, resulting in 10 to 72 variants per model, as detailed in Table 6 in Appendix. The variant that achieves the best average performance across all training datasets is selected.

Meta-learners. These approaches leverage performance matrix to guide model selection:

4. **ISAC** (Kadioglu et al., 2010): This clustering-based method groups training datasets based on their extracted meta-features. For a new dataset, ISAC identifies the nearest cluster and selects the best-performing model within that cluster.
5. **MLP.** Given the training datasets and selected time window, the MLP regressor directly maps the meta-features onto model performances by regression (Abdallah et al., 2022).

4.4 Overall Results

Superiority of LLM-based Methods in hit@k and MSE. The results presented in Tables 1 and 2

	Methods	MSE ↓
No Dynamic Model Selection	Seasonal Naïve	0.0345 ± 0.0382
	DeepAR	0.0164 ± 0.0506
	Deep Factors	0.0217 ± 0.0415
	Random Forest	0.0199 ± 0.0398
	Prophet	0.0155 ± 0.0295
	Gaussian Process	0.1661 ± 0.2104
	VAR	0.0602 ± 0.1260
Meta learner	$ISAC^*$	$\mathbf{0.0071 \pm 0.0145}$
	MLP^*	0.0351 ± 0.1186
LLM based	Ours-Llama3.2	$\mathit{0.0081 \pm 0.0297}$
	Ours-GPT4o	0.0234 ± 0.0596
	Ours-Gemini2.0 flash	0.0169 ± 0.0407

Table 2: Results for one-step ahead forecasting (MSE; the lower (↓) the better). The selected model by LLMs yields second best performance compared to baseline meta-learners and SOTA methods. $*$ denotes meta-learning methods which utilized performance matrix during training. All others don't require performance matrix.

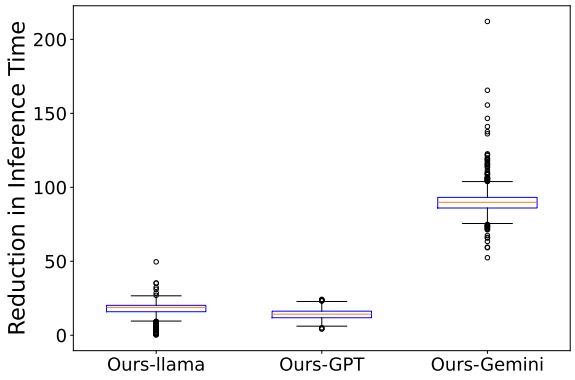

Figure 2: The inference time reduction of LLM-based methods over the naïve approach. Our Llama, GPT, and Gemini-based methods give a median reduction of 14X,18X, and 89X over naïve approach on all the datasets.

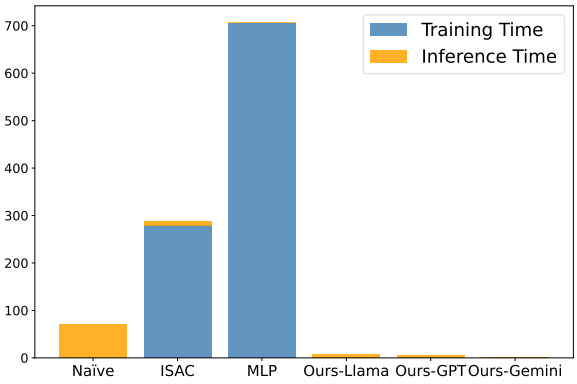

Figure 3: Average training and inference time in seconds (lower is better). Detailed mean and standard deviation values are provided in Table 5.

highlight the effectiveness of our LLM-based approach compared to all baseline methods. **Ours-LLaMA3.2** consistently outperforms other selection strategies across both hit@k accuracy and mean squared error (MSE). For instance, Ours-LLaMA3.2 achieves 100.27%, 92.83%, 77.32%, and 61.20% higher hit@10 accuracy compared to Random Selection, Popular Selection, ISAC, and MLP, respectively. In terms of forecasting performance, the model selected by Ours-LLaMA3.2 achieves the second lowest MSE among all tested methods, outperforming traditional SOTA forecasting models while achieving performance comparable to the best meta-learning method. Notably, unlike meta-learning approaches that require an extensive precomputed performance matrix for training, our LLM-based method selects models instantly without training.

Runtime Analysis. The inference runtime statistics of our methods are presented in Table 5, where our best Llama-based method achieves an inference time of 6.7 seconds for most time-series datasets. Additionally, as illustrated in Figure 2, LLM-based methods demonstrate a substantial reduction in inference time compared to the naïve approach, which involves doing inference using all possible models and selecting the best-performing one. Specifically, our Llama, GPT, and Gemini-based methods achieve median inference time reductions of 14X, 18X, and 89X, respectively, over the naïve approach on all the datasets.

4.5 Ablation Studies and Additional Analyses

Meta-Features. As shown in Table 1, incorporating meta-features in the prompt improves the performance of Llama and GPT-based methods, while the Gemini-based method appears to be less impacted. This improvement likely stems from the additional information provided by meta-features, which aids in selecting more suitable models. This performance gain comes at the cost of increased computational overhead, as shown in Table 5 and Table 4, the inference time rises by at least 25%, and prompt token usage expands by at least 7X.

Chain-of-Thought Prompting. Based on the results in Table 1, we observe that explicitly incorporating CoT reasoning in the prompt does not necessarily enhance model selection performance and sometimes even degrades it while significantly

hit@k accuracy	Methods	Data Representation		
		LLM Selection	Exponential Smoothing	Raw
hit@1	Ours-Llama3.2	**1.14**	1.04	0.31
	Ours-GPT4o	0.52	0.62	0.31
	Ours-Gemini2.0 flash	0.21	0	0.42
hit@5	Ours-Llama3.2	4.47	**4.98**	3.32
	Ours-GPT4o	2.60	2.60	1.66
	Ours-Gemini2.0 flash	0.62	0.31	2.18
hit@10	Ours-Llama3.2	7.27	**8.41**	4.47
	Ours-GPT4o	4.88	6.33	3.01
	Ours-Gemini2.0 flash	3.53	1.87	2.70
hit@50	Ours-Llama3.2	29.60	**31.15**	14.23
	Ours-GPT4o	21.91	23.36	11.84
	Ours-Gemini2.0 flash	20.77	19.94	8.10

Table 3: hit@k accuracy(in %) of LLM-based model selection, where the data representation is either chosen by the LLM or defaults to Exponential Smoothing or Raw.

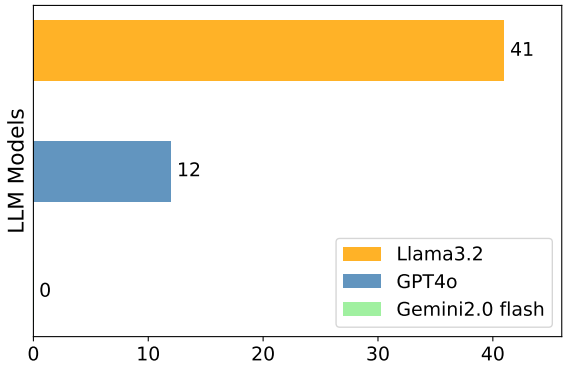

Figure 4: Average Number of Invalid Outputs for LLMs.

Methods	Input Tokens	Output Tokens
Ours-Llama3.2		
w. data	661.35 ± 1.16	157.14 ± 508.51
w. data+CoT	739.35 ± 1.16	519.00 ± 1445.28
w. data+MF	20547.43 ± 125.70	116.75 ± 274.24
w. data+MF+CoT	20632.43 ± 125.70	350.21 ± 931.21
Ours-GPT4o		
w. data	2418.98 ± 1472.00	68.89 ± 7.32
w. data+CoT	2711.98 ± 1472.00	297.18 ± 46.68
w. data+MF	22475.06 ± 1490.00	67.54 ± 8.37
w. data+MF+CoT	22750.06 ± 1490.00	300.68 ± 46.50
Ours-Gemini		
w. data	3075.64 ± 2192.18	84.33 ± 7.28
w. data+CoT	3075.64 ± 2192.18	340.53 ± 60.29
w. data+MF	26761.32 ± 2267.30	80.99 ± 4.06
w. data+MF+CoT	27080.32 ± 2267.30	352.31 ± 80.54

Table 4: Input and output token count for each time series dataset. Ours-Gemini refers to Gemini 2.0 flash. MF refers to meta-features.

increasing computational costs, leading to at least a 2X increase in inference time and a 4X rise in output token usage as shown in Table 5 and Table 4. We suspect that CoT prompting introduces unnecessary complexity, causing the LLM to overanalyze irrelevant aspects of the selection process. Unlike tasks where reasoning clarifies logic, model selection may benefit more from direct pattern recognition. The added reasoning steps could increase the risk of hallucination, leading to suboptimal choices.

Data Representation. Instead of allowing the LLM to select the data representation, we fixed it to either exponential smoothing or raw data. The results in Table 3 indicate that exponential smoothing enhances model selection performance for Llama and GPT-based methods, whereas it negatively impacts Gemini's selection performance.

Limitations of Different LLMs. Llama3.2 achieves the best overall model selection accuracy among the three LLMs evaluated, however, it also generates the highest proportion (4.26%) of invalid or incomplete outputs, as shown in Figure 4. We define invalid outputs as those where the predicted configuration falls outside our predefined model space of 322 candidates; such cases are filtered via rule-based matching and conservatively treated as the worst possible outcome, without attempting to post-process them into valid ones. In contrast, Gemini2.0 flash consistently generates complete

and valid outputs, with only 1.25% invalid outputs, while also having the lowest inference time and token usage. However, its performance is the weakest, under certain prompt settings, it even underperforms random selection. GPT4o serves as a balanced choice, delivering strong performance that surpasses almost all baselines, with only a few invalid outputs.

5 Conclusion, Limitations and Future Work

In this work, we applied LLMs to the time-series forecasting model selection problem for the first time. Through extensive experiments over 320 datasets across diverse domains, including finance, energy, retail, and IoT., we demonstrated that LLMs can effectively address this task without relying on a precomputed performance matrix of historical model-dataset pair evaluations, which enables seamless extension to unseen models and datasets without any retraining. Additionally, this method significantly reduces computational overhead, achieving up to 89× faster inference compared to exhaustive model evaluation. Despite the strong empirical performance, we observe that different LLMs exhibit distinct preferences for forecasting models, which in turn influence their selection behavior and overall performance. However, the underlying mechanisms driving these preferences remain unclear. Furthermore, our current approach has been evaluated solely on univariate datasets. In the future, we aim to expand our testbed for a more diverse set of datasets and models, further exploring the interpretability and generalizability of LLM-based model selection.

References

Mustafa Abdallah, Ryan Rossi, Kanak Mahadik, Sungchul Kim, Handong Zhao, and Saurabh Bagchi. 2022. Autoforecast: Automatic time-series forecasting model selection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 5–14, New York, NY, USA. Association for Computing Machinery.

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner TÃ¼rkmen, and Yuyang Wang. 2020. Gluonts: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. 2024. Chronos: Learning the language of time series. *Preprint*, arXiv:2403.07815.

Sebastian Pineda Arango, Pedro Mercado, Shubham Kapoor, Abdul Fatir Ansari, Lorenzo Stella, Huibin Shen, Hugo Senetaire, Caner Turkmen, Oleksandr Shchur, Danielle C. Maddix, Michael Bohlke-Schneider, Yuyang Wang, and Syama Sundar Rangapuram. 2025. Chronosx: Adapting pretrained time series models with exogenous variables. *Preprint*, arXiv:2503.12107.

C. Bui, N. Pham, Anh Vo, A. Tran, A. Nguyen, and Trung Le. 2018. Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases. In *6th International Conference on the Development of Biomedical Engineering in Vietnam (BME6)*, pages 809–818.

Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. 2024. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. *Preprint*, arXiv:2310.04948.

Vitor Cerqueira, Luis Torgo, and Carlos Soares. 2022. Model selection for time series forecasting: Empirical analysis of different estimators. *Preprint*, arXiv:2104.00584.

Jui-Sheng Chou and Duc-Son Tran. 2018. Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders. *Energy*, 165:709–726.

Robert Fildes, Shaohui Ma, and Stephan Kolassa. 2022. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318. Special Issue: M5 competition.

Google. 2024. Gemini 2.0 flash.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2024. Large language models are zero-shot time series forecasters. *Preprint*, arXiv:2310.07820.

Tao Han, Song Guo, Zhenghao Chen, Wanghan Xu, and Lei Bai. 2024. How far are today's time-series models from real-world weather forecasting applications? *Preprint*, arXiv:2406.14399.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. Time-llm: Time series forecasting by reprogramming large language models. *Preprint*, arXiv:2310.01728.

Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. 2010. Isac –instance-specific algorithm configuration. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, page 751–756, NLD. IOS Press.

Christiane Lemke and Bogdan Gabrys. 2010. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10):2006–2016. Subspace Learning / Selected papers from the European Symposium on Time Series Prediction.

Richard Lewis and Gregory C Reinsel. 1985. Prediction of multivariate time series by autoregressive model fitting. *Journal of Multivariate Analysis*, 16(3):393–411.

Yinheng Li. 2023. A practical survey on zero-shot prompt design for in-context learning. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 641–647, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Andy Liaw and Matthew Wiener. 2001. Classification and regression by randomforest. *Forest*, 23.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. itransformer: Inverted transformers are effective for time series forecasting. *Preprint*, arXiv:2310.06625.

MetaAI. 2024. Llama 3.2 model card.

Pablo Montero-Manso, George Athanasopoulos, Rob J. Hyndman, and Thiyanga S. Talagala. 2020. Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1):86–92. M4 Competition.

OpenAI. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Namyong Park, Ryan Rossi, Nesreen Ahmed, and Christos Faloutsos. 2023. Metagl: Evaluation-free selection of graph learning models via meta-learning. *Preprint*, arXiv:2206.09280.

Ricardo B.C. Prudêncio and Teresa B. Ludermir. 2004. Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137. Hybrid Neurocomputing: Selected Papers from the 2nd International Conference on Hybrid Intelligent Systems.

David Salinas, Valentin Flunkert, and Jan Gasthaus. 2019. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *Preprint*, arXiv:1704.04110.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. 2020. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. 2025. Time-moe: Billion-scale time series foundation models with mixture of experts. *Preprint*, arXiv:2409.16040.

Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and Statistical Modeling with Python. In *Proceedings of the 9th Python in Science Conference*, pages 92 – 96.

Yongqian Sun, Daguo Cheng, Tiankai Yang, Yuhe Ji, Shenglin Zhang, Man Zhu, Xiao Xiong, Qiliang Fan, Minghan Liang, Dan Pei, and 1 others. 2023. Efficient and robust kpi outlier detection for large-scale datacenters. *IEEE Transactions on Computers*, 72(10):2858–2871.

Sean Taylor and Benjamin Letham. 2017. Forecasting at scale.

Joaquin Vanschoren. 2018. Meta-learning: A survey. *Preprint*, arXiv:1810.03548.

Chunyang Wang, Yanmin Zhu, Haobing Liu, Tianzi Zang, Jiadi Yu, and Feilong Tang. 2022. Deep meta-learning in recommendation systems: A survey. *Preprint*, arXiv:2206.04415.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Weizhong Yan, Hai Qiu, and Ya Xue. 2009. Gaussian process for long-term time-series forecasting. In *2009 International Joint Conference on Neural Networks*, pages 3420–3427.

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K. Gupta, and Jingbo Shang. 2024. Large language models for time series: A survey. *Preprint*, arXiv:2402.01801.

Yue Zhao, Ryan A. Rossi, and Leman Akoglu. 2021. Automating outlier detection via meta-learning. *Preprint*, arXiv:2009.10606.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27268–27286. PMLR.

A Appendix

A.1 Prompt Structure

The prompt structure we used is illustrated as follows.

Prompt and Response Structure

Prompt: "[Role and Objective]...
[Model Space]...
[CoT Reasoning](optional)...
[Input]...Dataset Values...Meta Features(optional)...
[Output Format]...
[Rules]..."
Response:
{"reasoning": "...",
"result": { "forecasting algorithm": "...",
"hyperparameters": [{"name": "...", "value": "..."}, {"name": "...", "value": "..."}],
"data representation": "..." } }

We formulate the model selection problem following the framework of Abdallah et al. (2022).

A.2 Datasets and Meta-Features

Our approach relies on a collection of historical time-series forecasting datasets, denoted as $\mathcal{D} = \{D_1, D_2, \ldots, D_N\}$, where N is the total number of datasets.

Each dataset D_i comprises a sequence of observations in $\mathbb{R}^{n_i}$, with n_i representing the number of observations in D_i.

For each dataset $D_i \in \mathcal{D}$, we randomly sample T windows. Each time window w_t is a contiguous segment of observations from D_i with length $|w_t|$ that is smaller than the total length of D_i. For example, $|w_{10}| = 16$ indicates that the 10th window contains 16 consecutive observations.

Meta-Features Tensor. To analyze the impact of meta-features on model selection in our approach, we utilize extracted meta-features for each time-series dataset from Abdallah et al. (2022)'s work.

Definition 1. Given a time-series dataset D_i, we define the meta-features tensor $\mathcal{F}_i = \{F_1^i, \ldots, F_T^i\} \in \mathbb{R}^{T \times d}$, where the meta-features matrix $F_k^i \in \mathbb{R}^d$ captures the set of meta-features corresponding to the time window w_k of the dataset D_i, given by

$$F_k^i \triangleq \{\psi(w_k(D_i)) \mid \psi : \mathbb{R}^{|w_k|} \to \mathbb{R}^d\}, \quad (2)$$

Methods	Training Time	Inference Time
Naïve	0	70.95 ± 1.78
$ISAC^*$	278.81 ± 57.99	10.25 ± 2.72
MLP^*	705.29 ± 123.37	1.27 ± 0.52
Ours-Llama3.2		
w. data	0	4.72 ± 16.95
w. data+CoT	0	11.62 ± 32.77
w. data+MF	0	6.71 ± 17.87
w. data+MF+CoT	0	17.13 ± 50.10
Ours-GPT4o		
w. data	0	1.49 ± 0.72
w. data+CoT	0	4.78 ± 2.02
w. data+MF	0	2.46 ± 0.75
w. data+MF+CoT	0	5.24 ± 1.41
Ours-Gemini		
w. data	0	0.78 ± 0.05
w. data+CoT	0	2.16 ± 0.34
w. data+MF	0	0.98 ± 0.06
w. data+MF+CoT	0	2.50 ± 0.44

Table 5: Average and standard deviation inference and training time performance in seconds over all datasets. Ours-Gemini refers to Gemini 2.0 flash. MF refers to meta-features.

where $\psi(\cdot) : \mathbb{R}^{|w_k|} \to \mathbb{R}^d$ represents the feature extraction module and d denotes the number of the meta-features.

The extracted meta-features capture the key characteristics of each dataset and are grouped into five categories, as proposed by(Vanschoren, 2018):

- **Simple:** General task properties.
- **Statistical:** Properties of the underlying dataset distributions.
- **Information-theoretic:** Entropy measures.
- **Spectral:** Frequency domain properties.
- **Landmarker:** Forecasting models' attributes on the task.

A.3 Performance Matrix

Now we introduce the performance matrix:

Definition 2. Given a training database $\mathcal{D}$ and a model space $\mathcal{M}$, we define the performance matrix $\mathbf{P} \in \mathbb{R}^{T \times n \times m}$ as

$$\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_T\},$$

where $\mathbf{P}_k = (p_k^{i,j}) \in \mathbb{R}^{n \times m}$ and the element $p_k^{i,j} = M_j(w_k(D_i))$ denotes the jth model M_j's

Forecasting Algorithm	HyperParameter(s)	Data Representation	Total
DeepAR (Salinas et al., 2019)	num_cells = [10,20,30,40,50] num_rnn_layers = [1,2,3,4,5]	{Exp_smoothing, Raw}	50
DeepFactor (Salinas et al., 2019)	num_hidden_global = [10,20,30,40,50] num_global_factors = [1,5,10,15,20]	{Exp_smoothing, Raw}	50
Prophet (Taylor and Letham, 2017)	changepoint_prior_scale = [0.001, 0.01, 0.1, 0.2, 0.5] seasonality_prior_scale = [0.01, 0.1, 1.0, 5.0, 10.0]	{Exp_smoothing, Raw}	50
Seasonal Naive (Montero-Manso et al., 2020)	season_length = [1,5,7,10,30]	{Exp_smoothing, Raw}	10
Gaussian Process (Yan et al., 2009)	cardinality = [2,4,6,8,10] max_iter_jitter = [5,10,15,20,25]	{Exp_smoothing, Raw}	50
Vector Auto Regression (Lewis and Reinsel, 1985)	cov_type={"HC0","HC1","HC2","HC3","nonrobust"} trend = {'n', 'c', 't', 'ct'}	{Exp_smoothing, Raw}	40
Random Forest Regressor (Liaw and Wiener, 2001)	n_estimators = [10,50,100,250,500,1000] max_depth = [2,5,10,25,50,'None']	{Exp_smoothing, Raw}	72
			322

Table 6: Time-Series Forecasting Model Space. See hyperparameter definitions for various algorithms from GluonTS(Alexandrov et al., 2020) and statsmodels(Skipper Seabold and Josef Perktold, 2010). The number of models (last column) is all possible combinations of hyperparameters and data representations.

performance on the time window w_k of the ith training dataset D_i. We denote

$$p_k^i = \begin{bmatrix} p_k^{i,1} & \dots & p_k^{i,m} \end{bmatrix}$$

as the performance vector of all models in $\mathcal{M}$ on time window w_k of D_i.

We denote the performance of a model on a time window using forecasting error metrics such as Mean Squared Error (e.g., MSE) of that model on that window.