Giza Systems

Giza Systems ML: A Collaborative Meta-learning Based Framework Using LLM For Automated Time-Series Forecasting

GizaML: A Collaborative Meta-learning Based Framework Using LLM For Automated Time-Series Forecasting

Esraa Sayed
Giza Systems - Software Development Center
esraa.sayed@gizasystems.com

Mohamed Maher
University of Tartu
Tartu, Tartu
mohamed.abdelrahman@ut.ee

Omar Sedeek
Giza Systems - Software Development Center
omar.sedeek@gizasystems.com

Ahmed Eldamaty
Giza Systems - Software Development Center
ahmed.aldamati@gizasystems.com

Amr Kamel
Giza Systems - Software Development Center
amr.kamel@gizasystems.com

Radwa El Shawi
University of Tartu
Tartu, Tartu
radwa.elshawi@ut.ee

ABSTRACT

Machine learning algorithms demonstrate notable success in applications relying on time-series data, such as energy forecasting, environmental monitoring, and telecommunications. With the increasing prevalence of time-series data, there is a growing demand for accurate and generalized models for forecasting tasks. Training such models is a highly iterative process, requiring a profound understanding of both time-series data and machine learning algorithms. We demonstrate GizaML, a meta-learning-based framework designed specifically for automated algorithm selection and hyperparameter tuning for time-series forecasting. GizaML primarily comprises two key phases: the data and feature engineering phase, and the recommendation and optimization phase. In the data and feature engineering phase, GizaML resamples the dataset for uniform time intervals, handles outliers, and extracts various time-series-related features automatically. In the recommendation and optimization phase, GizaML employs a meta-model that proposes instantiations of machine learning pipeline configurations that are anticipated to exhibit strong performance on a novel dataset. These configurations warm start the optimization phase that employs an efficient Bayesian optimization method. The meta-model employs a Large Language Model (LLM) that is used to generate an embedding representation vector for the datasets' representations. GizaML utilizes 9 different regression machine learning algorithms and different hyper-parameters configurations for each one. Moreover, GizaML leverages new runs to continuously enhance the performance and robustness of the meta-model recommendations for future time-series forecasting tasks. Our demonstration scenario shows that GizaML outperforms the current state-of-the-art open-source automated machine learning frameworks.

1 INTRODUCTION

Time-series data and machine learning present unique challenges and opportunities in the ever-evolving domain of data science. Time-series data analysis in the field of machine learning is focused on developing algorithms capable of autonomously learning and improving their predictive capabilities for forecasting without direct human guidance. The efficacy of these machine learning methodologies is heavily reliant on the abundance of extensive time-series datasets. It is indisputable that the greater the volume of accessible **clean** data, the more comprehensive and resilient the insights and outcomes these algorithms can predict [14]. In the current landscape, there is a persistent expansion in both the scale and accessibility of time-series data across various facets of daily life [1]. Consequently, there has been a notable surge in achievements by machine learning within the domain of real-time time-series forecasting [8]. This trend has spurred a heightened demand for proficient data scientists equipped with profound knowledge and practical experience in employing diverse machine learning algorithms. Such expertise is essential for constructing models capable of attaining desired forecasting performance and coping with the escalating influx of the daily produced time-series data. The machine learning modeling process is iterative and lacks a universal solution for all scenarios. Experimenting with diverse algorithms and configurations is inefficient. Consequently, there is a growing emphasis on automating the machine learning modeling for time series data due to the scalability challenges faced by data scientists. [9].

Several frameworks have been developed to facilitate the automation of the machine learning modeling process on tabular datasets. For instance, Auto-Sklearn [2], a framework built on the popular Python scikit-learn machine learning package [1]. Auto-Sklearn considers prior performance on comparable datasets and forms ensembles from the models evaluated during the optimization process. Another example is TPOT[6] that employs genetic programming and SmartML [7] which follows a meta-learning based approach for algorithm selection and Bayesian optimization for hyper-parameter tuning of the selected algorithms. Additional tools in the time-series domain specifically encompass Auto-Gluon [5] employs Hyperband and Bayesian methods for hyper-parameter optimization. In addition, Auto-Gluon integrates traditional time-series statistical models, machine learning-based forecasting methods, and ensemble techniques. Yet, the tool relies on the manual extraction of the exogenous features that may influence the target time-series predictions. Furthermore, AutoTS[2] enables users to construct and choose from a variety of time series models, including techniques such as ARIMA, SARIMAX, VAR, decomposable models considering trend, seasonality, and holidays, along with ensemble these statistical machine learning models. However, the parameter estimation of these statistical models is computationally expensive and not suitable for real-time applications with high-velocity data.

In this demonstration, we present GizaML, a powerful Python package tailored for seamless integration into Python environments and web applications. Unlike conventional approaches for

[1]https://scikit-learn.org/stable/
[2]https://github.com/AutoViML/Auto_TS

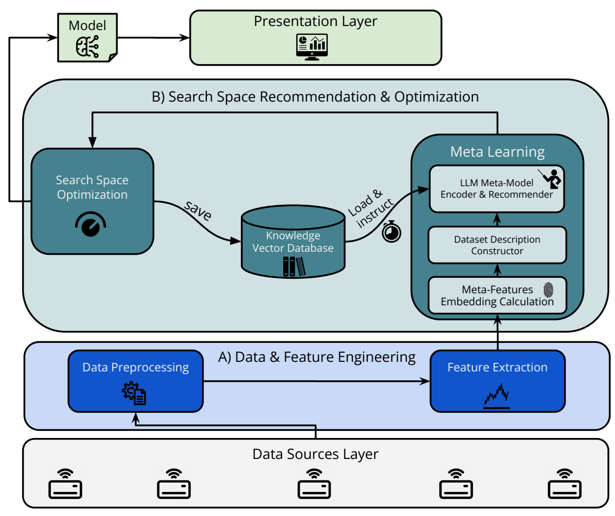

Model
Presentation Layer
B) Search Space Recommendation & Optimization
Meta Learning
LLM Meta-Model Encoder & Recommender
Search Space Optimization
save
Knowledge Vector Database
Load & instruct
Dataset Description Constructor
Meta-Features Embedding Calculation
Data Preprocessing
A) Data & Feature Engineering
Feature Extraction
Data Sources Layer

Figure 1: GizaML: Framework Architecture.

Type	Description
Time	Sampling frequency in minutes
Time	# Stationary features
Time	# Non Stationary features
Time	# Stationary Features after 1st order differencing
Time	# Stationary features after 2nd order differencing
Time	# Significant lags by pACF
Time	# Insignificant lags between the last and first significant ones
Time	# Seasonality components
Time	# Fractal Dimension
Time	Series Type (additive/multiplicative)
Time	Trend type (linear/logistic)
Time	pACF value for the first 10 lags
Statistical	Target Feature Kurtosis
Statistical	Target Feature Skewness
Statistical	Target Feature Missing Values %
Statistical	# Instances
Statistical	Min, Max, Avg, Stdev of the 10 percentiles of Target Feature

Table 1: List of the extracted meta-features from the time-series datasets.

tabular data, GizaML is specifically engineered for time-series data, offering end-to-end automation for training forecasting machine learning algorithms in addition to integration with interpretability techniques. One of the key features of GizaML is the recommendation component that employs a meta-model that suggests machine learning models with hyper-parameters configurations expected to demonstrate high performance on a new given dataset. More specifically, the meta-model provides tailored recommendations for machine learning regressors for each unique time-series dataset. Notably, these recommendations maintain confidentiality, as GizaML refrains from revealing exact instances from the dataset. Following SmartML [7], the meta-model draws insights from previous runs and continuously updates its knowledge base. This knowledge-driven approach reduces the hyper-parameter tuning search space and expedites the optimization process. In this demonstration, we provide a performance comparison with other state-of-the-art frameworks, highlighting GizaML's capacity to excel, particularly within constrained running time budgets.

2 GIZAML ARCHITECTURE

Figure 1 illustrates the architecture of GizaML. The framework mainly consists of two phases: a) the data and feature engineering

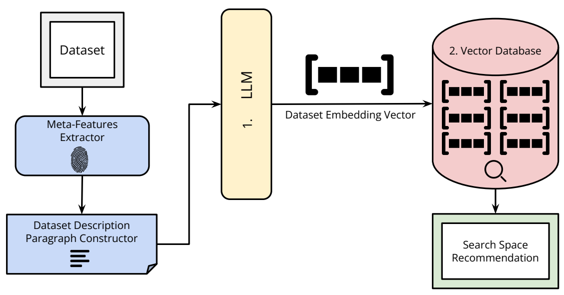
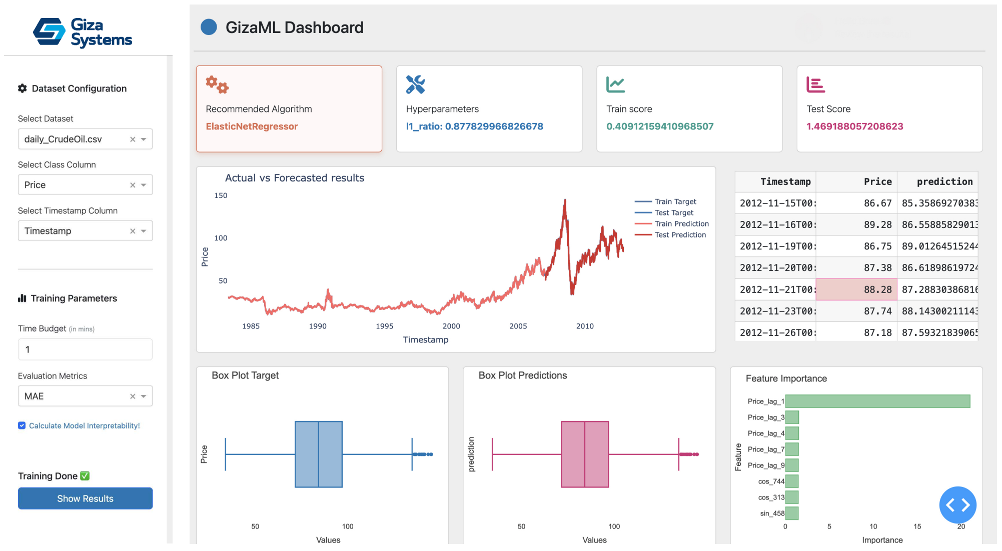


Figure 2: The search space recommendation for Time-Series AutoML task using a prompt tuned LLM to generate the dataset embedding vector and a vector database for querying similar datasets.

phase, and b) the recommendation and optimization phase. The typical user flow of GizaML starts with the data source layer when a time-series dataset in CSV format is uploaded as input, with the user specifying the timestamp and target feature columns, the sampling frequency for data source generation, the time budget constraint (**T**) for optimization, and the metric to be used in evaluation. In the data preprocessing and feature engineering phase, GizaML begins by resampling the dataset to ensure that consecutive instances are equally spaced in the time domain. [10] statistical time-series model, trained on the input training split. Subsequently, a fully automated process computes various time series-based features, starting from fundamental time-based features such as Hour-Of-Day, Day-Of-Week, and Month-Of-Year. The trend component is derived from the decomposed time-series target feature using the FBprophet [10] statistical time-series model. Fourier features serve as indicators for the seasonality components in the target feature, determined through observations in a periodogram [3], providing insights into the strength of different frequencies in the time-series data. Additionally, lagged features of the target column are extracted by computing the partial auto-correlation of a lag, accounting for all previous lags up to a maximum of 30 lags, to mitigate the curse of dimensionality in the dataset. Finally, a set of statistical [12] and time-based meta features described in Table 1 are computed and used to construct a description paragraph to characterize a dataset. In the recommendation and optimization phase, we employ meta-learning to recommend instantiations of models to warmstart the optimization procedure. More specifically, across a substantial number of datasets, we collect both performance data and the embedding representation vector of the dataset-specific description paragraph obtained from an instruct Large Language Model (LLM), aiding in the determination of the optimal algorithm for application to a new dataset. Given a new dataset, we compute its meta-features, as detailed in Table 1, and generate a descriptive paragraph. Subsequently, leveraging the instruct LLM to obtain the embedding representation vector for the dataset-specific description. Using this embedding vector, we query a



Dataset Configuration

Select Dataset: daily_CrudeOil.csv

Select Class Column: Price

Select Timestamp Column: Timestamp

Training Parameters

Time Budget (in mins): 1

Evaluation Metrics: MAE

[✓] Calculate Model Interpretability!

Training Done ✅

Recommended Algorithm: ElasticNetRegressor

Hyperparameters: l1_ratio: 0.877829966826678

Train score: 0.40912159410968507

Test Score: 1.469188057208623

Timestamp	Price	prediction
2012-11-15T00:	86.67	85.3586927038
2012-11-16T00:	89.28	86.5588582901
2012-11-19T00:	86.75	89.0126451524
2012-11-20T00:	87.38	86.6189861972
2012-11-21T00:	88.28	87.2883038681
2012-11-23T00:	87.74	88.1430021114
2012-11-26T00:	87.18	87.5932183906

Figure 3: The interface of GizaML. Forecasting job configurations are on the Left. The model performance, forecasted predictions, and model explanation providing the importance scores of the most important input feature are on the Right.

Algorithm Name	Numerical Parameters	Categorical Parameters	Package
Adaboost	2	1	scikit-learn
SVR	2	1	scikit-learn
RandomForest	2	0	scikit-learn
Lasso	1	0	scikit-learn
GaussianProcess	1	0	scikit-learn
XGBoost	7	0	xgboost
Lightgbm	9	1	lightgbm
ElasticNet	1	0	scikit-learn
ExtraTrees	4	1	scikit-learn

Table 2: Search Space of Regression Algorithms in GizaML.

knowledge base employing cosine similarity to identify the most similar time-series datasets. The top-performing configurations (machine learning algorithms and hyper-parameters) from the most similar N datasets are recommended to warm start the optimization process, with N defaulting to 5. For further optimization, we employ the SMAC algorithm. The input dataset is split into training and validation time-series splits. A time budget T is then allocated to exploit the hyper-parameter search space of the recommended models, maximizing the performance on the validation split. The resulting optimized pipeline, along with the embedding vector of the input dataset, is stored in the knowledge base for subsequent use in fine-tuning the LLM during an offline phase. Finally, the optimized machine learning algorithm with hyper-parameter configurations is saved along with the feature extraction stages to be used later in online forecasting scenarios. Additionally, an interpretability machine learning package (`Lime`)[3] is integrated to explain the most important features impacting the model forecasted predictions to the non-technical user. The predictions and evaluation scores of the model are demonstrated in the presentation layer dashboards.

[3]https://github.com/marcotcr/lime

3 META-MODEL FINE TUNING

Our meta-learning approach works as follows. In an offline phase, for each dataset within a knowledge base, consisting of 512 synthetic datasets and 30 real datasets from open data platforms including Kaggle [4] and Nasdaq stock market [5]), we apply grid search over the search space described in Table 2 to store an instantiation of the given machine learning framework with the strong empirical performance for that dataset. The synthetic time-series datasets are generated by varying the seasonality components, sampling frequencies, signal-to-noise ratios, percentages of missing values, and the nature of signal components (additive or multiplicative), aiming to encompass a broad spectrum of meta-features describing the datasets. Next, for each dataset in the knowledge base, we assess a set of meta-features described in Table 1, and construct a characterizing paragraph for the description of the dataset. Then, a set of instructions is constructed to fine-tune the LLM in the prompt tuning paradigm [13] with additive parameter-efficient fine-tuning mode [4]. Llama2 (7B) model [11] is utilized as the base LLM for our meta-model training. The description paragraphs of all the datasets in the knowledge base are used as prompts and the best-performing machine learning algorithms with the set of default hyper-parameters are considered as the completions. The final layers of the LLM network are removed to be used as an encoder generating embedding representation vectors for the datasets. These vectors of the knowledge base datasets are further stored in a vector database to be queried later in the online stage. We used the open source vector similarity search feature for `PostgreSQL` databases (pgvector) [6]. In an online phase, for a new dataset, we calculate its meta-features and create the characterizing paragraph. Using the LLM, we then derive the embedding representation vector for the dataset-specific description paragraph. A similarity-based algorithm utilizing the cosine distance metric is applied to query the closest datasets in the knowledge base. The LLM is fine-tuned regularly in an offline

[4]https://kaggle.com/
[5]https://www.nasdaq.com
[6]https://github.com/pgvector/pgvector

				5 min (MSE)			10 min (MSE)		
Dataset	Len.	Miss%	Rate	GizaML	AutoSk	TPOT	GizaML	AutoSk	TPOT
room_temperature	7056	0%	1 Hour	**3.0164**	4.0518	3.4936	**2.9645**	3.7679	3.4643
daily_CrudeOil	10831	31%	1 Day	101	2337	**48**	101	1714	**48**
daily-min-temperatures	3652	0%	1 Day	**4.7084**	6.0245	4.7310	**4.7086**	4.8848	4.6301
nasdaq_Brazil_Financial	10211	1%	1 Day	**0.0068**	0.0299	0.0070	**0.0068**	0.0299	0.0072
nasdaq_Brazil_deposits1	4059	8%	1 Day	**0.0299**	0.0360	0.0338	**0.0299**	0.0360	0.0340
nasdaq_Brazil_deposits2	11825	8%	1 Day	**0.0022**	0.0194	0.0031	**0.0022**	0.0192	0.0036
weekly_Gasoline_price	652	0%	7 Days	**0.0011**	0.0012	0.0014	**0.0011**	0.0012	0.0012
monthly_CO2_PPM	192	0%	28 Days	**0.5147**	1.8433	1.2688	**0.5148**	1.0197	1.2501
monthly_water_price	423	0%	28 Days	42	65	**36**	42	51	**34**
nasdaq_Brazil_Referential	185	0%	31 Days	**0.0026**	0.0030	0.0036	0.0026	**0.0025**	0.0031
nasdaq_Brazil_deposits3	133	0%	31 Days	0.0081	**0.0047**	0.0131	0.0081	**0.0058**	0.0126
nasdaq_EIA_PET_RWTC	434	0%	28 Days	**62**	129	68	**62**	129	65
Average Rank				**1.25**	2.75	2.08	**1.42**	2.58	2.00

Table 3: Performance Comparison: GizaML VS Auto-Sklearn and TPOT in terms of MSE. The average ranking represents that The lower the ranking, the better performance achieved by the framework compared to the other ones.

phase over the collaborative knowledge base incorporating the newly added data from the users' trials.

4 DEMO SCENARIO

`GizaML` is available as a Web application and a python package [7]. In this demonstration [8], we present the workflow of `GizaML` framework (See Figure 3 left). In particular, we show how our approach can help non-expert machine learning users to effectively build machine learning forecasting jobs with little effort. We start by introducing the tackled challenges and clarifying our framework's primary objectives and functionalities. Subsequently, we guide the audience through the automated algorithm selection and hyper-parameter tuning process using sample datasets. We start by showing the different features provided for the end-user. For example, the user can configure different options related to the time budget for the forecasting task, and specify whether model interpretability is needed. We guide the audience through the distinct phases of the framework, culminating in the presentation of the final results (See Figure 3 right).

Table 3 shows the performance comparison between `GizaML`, `Auto-Sklearn` (AutoSk), and `TPOT` using the 12 real datasets that were not used in building our knowledge base from Kaggle and Nasdaq stock market. As shown in Table 3, these datasets have different number of instances (Len.), missing values %, and sampling frequencies (Rate). Small time budgets of 5 and 10 minutes, respectively have been allocated for each dataset in each framework with a target of minimizing the Mean Squared Error (MSE) loss. The results show that `GizaML` outperforms `Auto-Sklearn` and `TPOT` in 9 out of 12 datasets achieving an overall ranking of 1.25 compared to 2.75 and 2.08 for the other frameworks with 5 minutes time budget, respectively. Using a longer time budget (10 min), `GizaML` average ranking declined to 1.42 while keeping the best performance across 8 out of the 12 datasets, which suggests the importance of the proposed meta-learning approach with small time budgets.

As an integral aspect of our demonstration, we offer a real-time opportunity to assess the efficacy of `GizaML` in comparison to other AutoML frameworks across diverse datasets. In future work, we will evaluate our framework on multivariate datasets and support classification tasks with multi-step forecasting.

[7]The source code of the `GizaML` is available on: https://github.com/giza-data-team/GizaAutoML

[8]A demonstration screencast is available on: https://youtu.be/533KoUSJb8M

ACKNOWLEDGMENTS

We appreciate the efforts of Omar Marie, Mahmoud Taha, and Ahmed Wael for their contributions to the demonstration. This work is funded by the innovation hub at Giza Systems [9].

REFERENCES

[1] Ana Almeida, Susana Brás, Susana Sargento, and Filipe Cabral Pinto. 2023. Time series big data: a survey on data stream frameworks, analysis and algorithms. *Journal of Big Data* 10, 1 (2023), 83.

[2] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2020. Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074* 24 (2020).

[3] David F Findley, Demetra P Lytras, and Tucker S McElroy. 2017. Detecting seasonality in seasonally adjusted monthly time series. *Statistics* 3 (2017).

[4] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366* (2021).

[5] Aaron Klein, Louis C Tiao, Thibaut Lienart, Cedric Archambeau, and Matthias Seeger. 2020. Model-based asynchronous hyperparameter and neural architecture search. *arXiv preprint arXiv:2003.10865* (2020).

[6] Trang T Le, Weixuan Fu, and Jason H Moore. 2020. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics* 36, 1 (2020), 250–256.

[7] Mohamed Mohamed Maher Zenhom Abdelrahman Maher and Sherif Sakr. 2019. Smartml: A meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. In *EDBT: 22nd International conference on extending database technology.*

[8] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. 2023. Machine learning advances for time series forecasting. *Journal of economic surveys* 37, 1 (2023), 76–111.

[9] Marc Schmitt. 2022. Automated machine learning: AI-driven decision making in business analytics. *arXiv preprint arXiv:2205.10538* (2022).

[10] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.

[11] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[12] Joaquin Vanschoren. 2019. Meta-learning. *Automated machine learning: methods, systems, challenges* (2019), 35–61.

[13] Chaozheng Wang, Yuanhang Yang, Cuiyun Gao, Yun Peng, Hongyu Zhang, and Michael R Lyu. 2022. No more fine-tuning? an experimental evaluation of prompt tuning in code intelligence. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* 382–394.

[14] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. 2023. Data-centric ai: Perspectives and challenges. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM).* SIAM, 945–948.

[9]https://gizasystems.com