A Flexible Forecasting Stack

Tim Januschowski*
Databricks
tim.januschowski@databricks.com

Jan Gasthaus*
jgasthaus@gmail.com

Yuyang Wang
Amazon
yuyawang@amazon.com

Syama Rangapuram
Amazon
rangapur@amazon.com

Caner Türkmen
Amazon
atturkm@amazon.com

Jasper Zschiegner
Amazon
jzieg@amazon.com

Lorenzo Stella
Amazon
stellalo@amazon.com

Michael Bohlke-Schneider
Amazon
bohlkem@amazon.com

Danielle Maddix
Amazon
dmmaddix@amazon.com

Konstantinos Benidis
Amazon
kbenidis@amazon.com

Alexander Alexandrov*
Materialize
alexandrov@materialize.com

Christos Faloutsos*
Amazon & CMU
christos@cs.cmu.edu

Sebastian Schelter*
BIFOLD & TU Berlin
schelter@tu-berlin.de

ABSTRACT

Forecasting extrapolates the values of a time series into the future, and is crucial to optimize core operations for many businesses and organizations. Building machine learning (ML)-based forecasting applications presents a challenge though, due to non-stationary data and large numbers of time series. As there is no single dominating approach to forecasting, forecasting systems have to support a wide variety of approaches, ranging from deep learning-based methods to classical methods built on probabilistic modelling.

We revisit our earlier work on a monolithic platform for forecasting from VLDB 2017, and describe how we evolved it into a modern forecasting stack consisting of several layers that support a wide range of forecasting needs and automate common tasks like model selection. This stack leverages our open source forecasting libraries *GluonTS* and *AutoGluon-TimeSeries*, the scalable ML platform *SageMaker*, and forms the basis of the no-code forecasting solutions (*SageMaker Canvas* and *Amazon Forecast*), available in the Amazon Web Services cloud. We give insights into the predictive performance of our stack and discuss learnings from using it to provision resources for the cloud database services DynamoDB, Redshift and Athena.

PVLDB Reference Format:
Tim Januschowski, Yuyang Wang, Jan Gasthaus, Syama Rangapuram, Caner Türkmen, Jasper Zschiegner, et al. A Flexible Forecasting Stack. PVLDB, 17(12): 3883 - 3892, 2024.
doi:10.14778/3685800.3685813

*Work done while at Amazon.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.

1 INTRODUCTION

Forecasting predicts time series into the future and has many critical applications, ranging from inventory management in supply chains to provisioning of compute resources in the cloud [35, 54].

Challenges in forecasting. The recent growth of time series corpora has led to methodological challenges and puts existing forecasting systems under stress. In particular, the diversity and scale of modern forecasting problems poses the following challenges [8, 63]:

- *Model selection* – Empirical evidence suggests that there is *no single dominant forecasting algorithm* and instead, a wide range of methods and paradigms are applied in practice. For example, most classical forecasting methods [31, 71] employ one *local model* per time series in a large corpus of time series. The canonical implementations of these methods typically do not explicitly distinguish between training and inference, and fit parameters at inference time. At the other methodological extreme are high-capacity deep learning methods [6, 34, 40, 49, 57, 59, 60, 70] where a single *global model* is learnt over a large number of time series. Since so far no generally dominant approach is available, the best approach for a given scenario depends on the data [14, 39, 47, 48] and a modern forecasting stack needs to support both local and global models, each with their own computational patterns.
- *Non-stationary data* – Auxiliary machine learning (ML) tasks such as hyperparameter tuning or backtesting have their own pecularities in the forecasting task given the non-stationarity of the data and the importance of time in forecasting, hence standard supervised learning approaches need (often non-trivial) adaptations.
- *Variety of user expertise* – Real world forecasting systems need to support users with highly varying needs and levels of expertise, ranging from scientists developing new methods to non-technical business users that need friction-free delivery of forecasts.

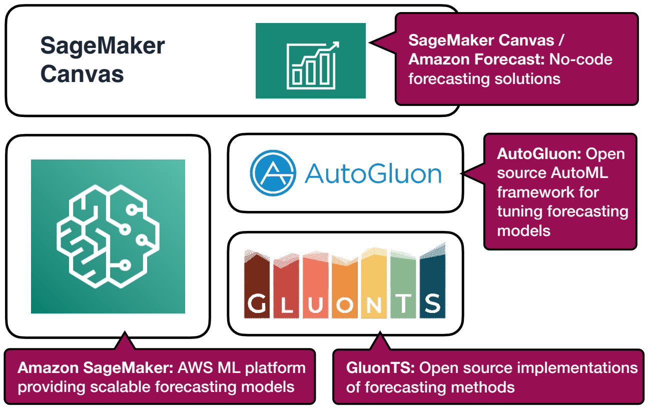


Figure 1: High-level overview of the AWS forecasting stack.

History of infrastructure for forecasting at AWS. Our approach to the aforementioned challenges has evolved over time. In 2016/2017, we designed a monolithic end-to-end system for probabilistic forecasting [8], centered on Apache Spark [76]. This system improved upon existing ad-hoc solutions for which end-to-end experimentation was not possible. We chose a monolithic architecture with a single system for both experimentation and production to avoid the common pitfall of diverging experimentation and production codebases [63]. We had to implement many custom forecasting algorithms and experimentation techniques into this monolith, due to a lack of publicly available viable alternatives at this time.

With the advent of deep learning and its associated ecosystem, our next step was to focus on methodological improvements to forecasting algorithms [19, 55, 61, 62, 73]. We provide these models as part of the Apache-licensed open source library *GluonTS* [1]. Concurrently, we helped conceive the SageMaker [41] platform for scalable ML, which offers infrastructure for productization and scalability that does not restrict experimental capabilities as much as our original monolith.

The AWS forecasting stack. We combined the advantages of the previous monolith with the composability of the current infrastructure in the AWS forecasting stack. It provides end-to-end experimentation and production-grade quality with high composability, to ensure that a a seamless offering is available for different life cycle stages and data scales of forecasting scenarios.

As illustrated in Figure 1, our stack consists of four major layers (Section 2). Its foundation is the open-source Python library *GluonTS* [1] for probabilistic forecasting, and we leverage *SageMaker* [41] for scalable and elastic model training and inference. *AutoGluon-TimeSeries* [65] builds on GluonTS, SageMaker, and the popular AutoML framework AutoGluon [15]. AutoGluon-TimeSeries provides an easy-to-use interface for automatically building, tuning and combining accurate forecasting models. Furthermore, we provide no-code forecasting services (Time Series Forecasts in SageMaker Canvas and Amazon Forecast), built on top of GluonTS and AutoGluon-TimeSeries, to automate tedious issues such as hardware provisioning and feature preprocessing.

In Section 5, we experimentally evaluate the predictive performance, the distribution of algorithm choices, as well the distribution of runtimes of our service. Our results indicate that the main challenge in forecasting is not a computational one, but rather identifying the appropriate model from the wide range of families of well-working forecasting algorithms (e.g., classic time series smoothing, deep-learning or boosted decision trees [34]).

Resource provisioning for cloud database services. Finally, we discuss how we used our forecasting stack to optimise the resource provisioning (in terms of the hardware demand of internal customer teams, or in terms of deciding on the size of a warm pool of pre-provisioned machines) for Amazon's cloud database services DynamoDB [66], Redshift [50] and Athena [9] in Section 5.2. For each use case, we describe which layer of our stack was chosen, and how this choice contributed to the developed forecasting solution. As a result of improved forecasting methods, Athena was able to achieve an enormous reduction in warm pool sizes between 30-45%, and Redshift saw utilization gains between 35% and 91% relative to an existing baseline solution.

Contributions. The contributions of this paper are as follows:

- We detail the design of the AWS forecasting stack, which is part of the machine learning offering of the Amazon Web Services cloud (Section 2).
- We give examples of the prediction quality, runtime and model selection capabilities of our stack (Section 5.1).
- We discuss our learnings from using the AWS forecasting stack to provision resources for the cloud database services DynamoDB, Redshift and Athena (Section 5.2).

2 THE AWS FORECASTING STACK

We describe the AWS forecasting stack, which is available as part of the AWS machine learning offering at https://aws.amazon.com/machine-learning/.

2.1 Overview

As mentioned in the introduction, our stack consists of four major layers: The open source forecasting library *GluonTS* [1] (Section 2.2), the intregration with *Amazon SageMaker* for scalable and elastic training and inference (Section 2.3), hyperparameter optimization and ensembling via the *AutoGluon-TimeSeries* library (Section 2.4), and no-code functionality provided by SageMaker Canvas and *Amazon Forecast* (Section 2.5).

Forecasting algorithms. The available algorithms cover the main families of forecasting, see Table 1 for an overview. A major distinction are local methods [31, 71], which train a model per time series, and global models, which train a model for a large set of time series [6, 34, 40, 49, 57, 59, 60, 70]. We choose methods based on their popularity and maturity of implementation. Global models, which employ deep learning or gradient-boosted trees [27, 69] are in the majority, as they are particularly amenable for larger-scale forecasting problems.

Input data. All layers of our stack operate on a simple, JSON-based input format for multiple time series, as illustrated in the snippet below. Users have to specify the `start` time of a time series, as well as the time series itself (`target`), and can specify additional categorical covariates (e.g., via the `cat` attribute). Note that the granularity of the time series (e.g., minutes, seconds, hours, ...) is globally configured.


```
{"start": "2016-01-16", "cat": 1, "target": [4.96, 5.15, 5.12, ...]}
{"start": "2016-01-01", "cat": 1, "target": [3.04, 3.19, 3.65, ...]}
{"start": "2016-02-08", "cat": 2, "target": [2.51, 2.81, 3.14, ...]}
```


2.2 GluonTS

The foundation of the forecasting stack is the *GluonTS* [1] library for probabilistic forecasting with backends for popular deep learning engines such as PyTorch [51] and MXNet [11], available at https://ts.gluon.ai. Apart from forecasting, it contains elementary support for time series analysis tasks such as classification and anomaly detection. GluonTS allows its users to quickly build both forecasting prototypes and develop new models, via support for automated backtesting, rich featurization, an exhaustive list of forecast accuracy metrics and a plethora of state-of-the-art forecasting models (as detailed in Table 1). GluonTS is in active use in research [56, 59, 60, 62, 73], forecasting competitions [37] and industrial applications (see Section 5.2 for details).

Scientists working on model improvements require a high degree of standardization and reproducibility for comparison with the state-of-the-art. Unlike in production use cases, the data for most scientific experiments is assumed to be static. GluonTS is integrated with many common openly available forecasting datasets, including those collected in [23], and makes it easy for users to run standardised experiments on these. Furthermore, our library supports the declarative generation of synthetic datasets to assist data scientists with model design, debugging and testing. Additional components in GluonTS are feature transformations for generating common date features, whose implementation follows the estimator/transformer pipeline approach that scikit-learn [52] popularised. Note that time series have to be carefully featurised: it is important to scale features without leaking data from the future, so if for example a mean is calculated as part of the normalization, it must only be calculated with respect to observed values from the relative past of the time series. Moreover, GluonTS supports comparisons with other well-known forecasting packages such as R Forecast [31] or Prophet [71] via Docker containers with custom data conversion code.

2.3 Forecasting with SageMaker

While GluonTS focuses on experimentation and algorithm development, tasks such as distributed training, extensive hyperparameter optimization and scalable deployment are covered by SageMaker [41]. SageMaker already provides the prebuilt algorithm DeepAR [62] for forecasting out-of-the box and enables the usage of GluonTS containers with low integration costs. GluonTS makes no general assumptions on the execution environment and can therefore be run on a variety of cloud platforms. However, it provides a custom shell module for a direct integration with SageMaker. Users can dynamically select a specific model by setting a hyper-parameter. Unlike SageMaker's DeepAR, users of GluonTS need to build and upload their own algorithm container. To ease this process, GluonTS provides prebuilt Docker files, which, in our experience, often facilitate a working setup within minutes.

Table 1: A selected subset of the forecasting methods available in the AWS forecasting stack. Readiness denotes whether a given implementation is production ready (prod) or only available in an experimental state (exp). The level details whether the model has either robust default parameters or an auto-tuning component (auto), requires basic forecasting knowledge (basic) or deep expertise (expert). The availability column indicates whether algorithms are available in GluonTS (GTS) or Forecast (AF).

Algorithm(s)	**Type**	**Readiness**	**Level**	**Availability** GTS	AF
ETS [30], ARIMA [30]	local	prod	auto	✓	✓
Prophet [71], NPTS	local	prod	auto	✓	✓
DeepAR [62], CNN-QR [2]	global	prod	auto	✓	✓
MQ*NN [74]	global	prod	basic	✓	✓
DeepNPTS	global	exp	expert	✓	✓
Rotbaum [27]	global	exp	expert	✓	
TFT [42], NBEATS [49], FeedForward	global	exp	basic	✓	
DeepVAR [61], DeepState [56], DeepRenewal [72], DeepFactors [73]	global	exp	expert	✓	

Development and deployment. SageMaker uses algorithm containers to run training and inference workflows. This requires that either the algorithm code is part of the container image or that the code is loaded dynamically before execution. For production use-cases, a static image which contains the forecasting algorithm is desirable. During development however, flexibility is key. To reconcile these two requirements, a custom shell module can recursively search for Python modules and packages, which are then installed within the container, before the process gets restarted. In subsequent runs, the newly installed packages have priority over possible existing versions, which means that existing packages can be patched to updated versions.

Integration benefits. The close integration with SageMaker comes with several benefits: New GluonTS models have a direct path to large-scale experimentation and productization. Users can benefit from existing components in SageMaker, such as state-of-the-art hyper-parameter optimization [53], a feature store, data processing in pipelines, model debugging [67] and profiling, and support for batch and online inference modes. When developing new forecasting models, scientists can initially prototype their ideas independently of a framework. However, once a prototype is integrated with GluonTS, it can directly be compared against established approaches via SageMaker on existing datasets.

2.4 AutoGluon-TimeSeries

Our next component focuses on the model selection challenge in forecasting (Section 1). AutoGluon-TimeSeries (AG-TS) [65], is a specialised module within the broader AutoGluon framework, is designed for time series forecasting. AutoGluon [15] is a popular state-of-the-art AutoML framework for tabular data tasks [21], developed by an active community, led mainly by the AI research division at AWS. It is worth noting that time series pose unique challenges for the standard AutoML framework. Cross-validation, a primitive that both multi-layer stacking and ensembling hinge on, has to adhere to the order in the data imposed by the presence of time. This is taken into account during backtesting with AG-TS, which ensures that the temporal order of the splitting is preserved at all times, and information leakage is avoided.

Our previous monolithic forecasting system from 2017 [8] already had rudimentary support for model selection and ensemble creation. However, this functionality was difficult to use and required a lot of trial-and-error by the end users, since they had to manually define complex combinations of predicates to select subsets of the training data as inputs for different algorithm variants. Moreover, they also had to manually specify the weights for combining the predictions of the resulting model variants into the final ensemble. As a consequence, we put a significant emphasis on the automation of hyperparameter tuning and ensembling in our reworked stack.

AutoML for forecasting. AG-TS adopts the core principles of AutoGluon, particularly the reliance on ensembling techniques, diverging from a sole focus on hyperparameter optimization or neural architecture search. AG-TS builds ensembles from a broad selection of models (most of which depend on GluonTS implementations) [10]. AG-TS also aims to serve domain experts with limited experience in forecasting, by embracing the convention-over-configuration principle, as it provides default hyperparameter settings tuned to deliver strong performance across various scenarios. In particular, the API in AG-TS shields the user from complex considerations regarding model selection, specification, tuning, and training. AG-TS also provides a range of high-level configuration options to choose from a portfolio of models and hyperparameters, in order to balance between rapid training times and enhanced accuracy.

Integration. Like GluonTS, AG-TS integrates seamlessly with SageMaker for training and production-ready inference deployment. AutoGluon is part of SageMaker's deep learning containers, available at https://github.com/aws/deep-learning-containers, which provide a well-maintained, robust, portable and secure environment for a wide array of machine learning applications within Amazon's machine learning ecosystem. Furthermore, AG-TS can be utilised through AutoGluon Cloud on SageMaker, offering an even more streamlined and efficient deployment pathway.

2.5 SageMaker Canvas and Amazon Forecast

The forecasting capabilities in SageMaker Canvas and *Amazon Forecast* are managed forecasting services, built around GluonTS and SageMaker. As such, they aim to free the user from having to worry about hardware provisioning and operations. As both Canvas and Amazon Forecast are based on the same technical underpinnings, we refer them collectively as Forecast here. Forecast uses SageMaker for training and inference, and executes its feature processing via extract-transform-load (ETL) pipelines in Apache Spark [76]. Forecast limits customization opportunities, as its primary goal is the abstraction and automation to lower the technical bar for building ML-based forecasting applications.

Model tuning, selection and combination. Forecast inherits large parts of the model tuning functionality from SageMaker or contains specializations that are similar to what we discussed before in Section 2.4. For example, its hyperparameter optimization combines the backtesting capabilities of our forecasting stack here with SageMaker's generic hyperparameter tuning [53].

Backtesting. Setting up a backtesting scenario properly in forecasting is challenging for ML practitioners without prior exposure to forecasting. This is due to the fact that time series data is not assumed to be identically and independently distributed (in contrast to data for supervised learning scenarios such as classification and regression). Therefore, partitioning the data into train, test and validation sets always needs to take the time dimension into account to avoid data leakage, preferably in a rolling forecast horizon fashion [30]. Furthermore, forecasting comes with its own set of evaluation metrics [22, 32].

Data augmentation. Additionally, Amazon Forecast provides a prepopulated feature store that allows data scientists to augment existing data with available covariates. Examples include calendar events such as holidays, weekends, or workdays, as well as weather index features. The latter often help to "explain away" anomalies (e.g., a drop in attendance due to extreme weather) and to predict future values (e.g., hot temperatures are indicative for sales of groceries in retail). However, incorporating weather data correctly in forecasting during model training and backtesting is challenging. We need to take into account that the actual observed weather is only available for the past and not in the future, where it is only available in the form of forecasts. Training such models requires incorporating the correct past weather forecast to avoid data leakage. Forecast aims to abstract away these subtleties, and joins the time series from customers with our internal weather data based on the user-provided timestamps during the ETL process. Other features, like calendar data, can be produced on the fly via standard date libraries and training data (over)sampling techniques are inherited from GluonTS.

2.6 Retrospective

We would finally like to mention that the components of our stack, especially GluonTS, were designed to allow for addressing Amazon internal use cases in general. One particular such use case was the conception of Forecast, where the necessity arose to quickly build, evaluate and deliver state-of-the-art forecasting algorithms. Forecast in turn relies internally on SageMaker. Hence, the stack presented here externalises many of the learnings of Amazon's forecasting teams.

3 BACKGROUND

Forecasting. Given a regularly sampled time series $z_{i,t}$ representing the measurements of an item i at time t, the goal is to estimate $P(z_{i,T+1}, \ldots, z_{i,T+h} | z_{i,1}, \ldots, z_{i,T})$, the predictive distribution over the a forecast horizon $T+1, \ldots, T+h$ of length h conditioned on historical values. Forecasts, especially probabilistic ones, are instrumental for decision making problems that require optimising expected (future) costs.

Note that we often have additional co-variate information available and we may be interested in a multi-variate forecast as well. Forecasting typically requires one to train a model parameterised by θ either over a single time series i (*local* models) or over groups of time series (leading to *global* models). Examples for forecasting models range from classical state space models [29], dynamical factor models [20], linear regression [30] and combinations thereof [64] to modern methods including gradient boosted trees [36, 69] and deep learning [7, 16, 70].

Amazon SageMaker. SageMaker, available at https://aws.amazon.com/sagemaker/, is a scalable machine learning platform for training models on large, continuously evolving datasets, which supports incremental training, resumable and elastic learning as well as automatic hyperparameter optimization [41].

Most algorithms are designed based on a computational model for incremental updates, which resembles the way in which distributed aggregation functions are executed in relational database management systems. It can leverage both CPUs and GPUs seamlessly, as many built-in algorithms in SageMaker use common deep learning engines as an interface to the underlying hardware. Furthermore, SageMaker also supports the scalable deployment of blackbox algorithms provided in the form of a Docker image.

SageMaker Canvas is a visual, no-code tool designed to make it easier for business analysts and other non-developers to build ML models without requiring any machine learning expertise. Canvas provides a user-friendly interface that allows users to generate predictions by creating ML models based on their data. Users can simply upload their data, use the graphical interface to prepare and clean it if necessary, and then let Canvas automatically build a model that can make predictions.

4 RELATED WORK

Contemporary forecasting solutions are either available under open-source or proprietary licenses. For the open-source side [33], popular examples include R Forecast [31], StatsForecast [18] and Prophet [71] for classical methods, and PyTorch-forecasting [5] or Darts [28] for deep learning based methods [6], which have seen an explosive growth over the last two years [40, 49, 57, 59, 60, 70]. Open source solutions enable control and customisability, but typically require significant effort for productionization. For example, scaling classical methods to large datasets is challenging [69] and may often require systems like SageMaker.

Recently, the emergent capabilities of large language models have sparked strong interest in developing so-called foundation models for time series forecasting [3, 12, 13, 17, 24, 25, 38, 58, 75]. Despite the promising zero-shot performance of such pre-trained forecasting models on benchmark datasets, it remains to be seen how well they perform in real-world use cases. That said, as a framework, AG-TS in the proposed stack is well poised to embrace the latest developments in foundation time series models, with Chronos [3] being a part of the ensemble.

On the other hand, closed-source commercial solutions from enterprise software vendors and start-ups like BlueYonder often lack customization opportunities but are production ready. We note that forecasting solutions using SAS or Matlab sit in between these extremes, but their closed-source nature does not offer the same sort of transparency that in particular scientists have come to value. Our stack incoporates both open source and proprietary methods, and attempts to take the best from both worlds by allowing different entry points to reduce the engineering efforts for productionising customised, open-source based solutions with SageMaker. To the best of our knowledge, our solution is the first comprehensive stack for forecasting-related problems (in contrast to stacks for general ML tasks such as Google TFX [4]).

5 EVALUATION & LEARNINGS

We give examples of individual model performance, overall algorithm performance for our methods and the time required to find a well-working method in Section 5.1. In Section 5.2, we discuss our learnings from three internal use cases of applying our stack in resource provisioning for the database services DynamoDB [66], Athena [9] and Redshift [50] in the Amazon Web Services cloud.

5.1 Predictive Performance & Runtime

We discuss model performance and runtime in the following.

Individual model performance. Often, our individual models are already highly competitive. As a showcase, we compare DeepAR (with default settings) to the winning solution [68] of the M4 forecasting competition [46]. We find that DeepAR achieves state-of-the-art performance, with a mean absolute scaled error (MASE) of 1.50 compared to 1.54, a symmetric mean absolute percentage error (sMAPE) of 0.12 compared to 0.114, and a mean scaled interval score (MSIS) of 12 compared to 12.2.

No one-size-fits-all solution. Even though deep learning-based global models often perform extremely well, the field of forecasting is not dominated by these approaches in a similar fashion such as natural language processing or computer vision [45, 46]. As discussed in Section 2, we explicitly designed our stack to incorporate methods from many different forecasting approaches and provide AutoML components that learn ensembles or select the best algorithm for a given dataset and forecasting scenario.

We validate this claim by discussing the results from a recent study of ours [65], where we evaluate five individual forecasting algorithms from Gluon-TS (Section 2.2) as well as ensembles learned by AG-TS on a collection of twenty-weight public forecasting datasets. The study for example includes data from the Monash Forecasting Repository [23], such as the M1, M3 and M4 competition data [44, 47], and the range of datasets covers various scenarios that can be encountered in practice – from small datasets (M1 and M3) to datasets with a few long time series (Electricity, Pedestrian Counts) and to large collections of medium-sized time series (M4). The neural methods evaluated in this study are DeepAR [62] and TFT [42] (which applies transformers) and the classical methods included were ARIMA and ETS as well as an ensemble of them (ARIMA+ETS). We generated probabilistic (quantile) forecasts and compute their mean weighted quantile loss (wQL) averaged over nine quantile levels $q \in \{0.1, 0.2, ..., 0.9\}$.

We detail the resulting scores per dataset and forecasting algorithm in Table 2, where we indicate the best individual algorithm performance in bold. The columns on the right side of this table additionally indicates the performance of a forecasting ensemble learned by AutoGluon-TS and whether this ensemble wins or loses

Table 2: Probabilistic forecasting performance (measured by weighted quantile loss (wQL), lower is better) of five forecasting algorithms from Gluon-TS on twenty eight public datasets. The best individual algorithm performance is indicated in bold. The columns on the right indicate the performance of a forecasting ensemble learned by AutoGluon-TS and whether this ensemble wins or loses (or has a tie) against the best performing individual forecasting algorithm. The results show that no individual family of methods (classical or neural forecasting) dominates performance. Furthermore, the ensemble learned by AutoGluonTS achieves on-par or better performance than the best single model in 19 out of 28 cases.

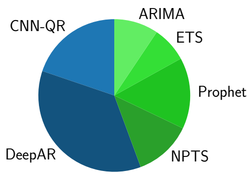
Dataset				Individual forecasting algorithms					Ensemble learned by AG-TS			
Name	#Series	Frequency	Seasonality	ARIMA	ETS	ARIMA+ETS	DeepAR	TFT	wQL score	Win?	Tie?	Loss?
Car Parts	2,674	M	12	1.589	1.338	1.324	0.963 (0.009)	**0.878** (0.004)	0.923 (0.0)			✓
CIF 2016	72	M	12	0.017	0.039	0.028	0.114 (0.024)	**0.010** (0.002)	0.019 (0.0)			✓
COVID	266	D	7	**0.030**	0.046	0.046	0.072 (0.02)	0.031 (0.003)	0.030 (0.0)	✓		
Electricity Hourly	321	H	24	-	0.100	-	**0.081** (0.002)	0.097 (0.001)	0.076 (0.0)	✓		
Electricity Weekly	321	W	1	0.138	0.144	0.141	0.123 (0.041)	**0.118** (0.011)	0.088 (0.0)	✓		
FRED-MD	107	M	12	0.056	**0.050**	0.054	0.054 (0.021)	0.114 (0.011)	0.056 (0.0)			✓
Hospital	767	M	12	0.058	**0.053**	**0.053**	**0.053** (0.001)	0.054 (0.001)	0.051 (0.0)	✓		
KDD Cup 2018	270	H	24	-	0.550	-	**0.363** (0.014)	0.488 (0.054)	0.323 (0.014)	✓		
M1 Monthly	617	M	12	0.146	0.163	0.152	**0.136** (0.008)	0.224 (0.016)	0.135 (0.0)	✓		
M1 Quarterly	203	Q	4	0.088	**0.081**	0.083	0.084 (0.003)	0.093 (0.006)	0.090 (0.0)			✓
M1 Yearly	181	Y	1	0.160	0.139	0.142	0.142 (0.029)	**0.127** (0.004)	0.134 (0.001)			✓
M3 Monthly	1,428	M	12	0.102	0.093	**0.092**	0.098 (0.001)	0.109 (0.003)	0.089 (0.0)	✓		
M3 Other	174	Q	1	0.035	0.032	**0.031**	0.036 (0.002)	0.033 (0.001)	0.031 (0.0)		✓	
M3 Quarterly	756	Q	4	0.079	0.069	**0.068**	0.073 (0.001)	0.071 (0.001)	0.065 (0.0)	✓		
M3 Yearly	645	Y	1	0.162	0.129	0.128	**0.117** (0.002)	0.133 (0.001)	0.114 (0.0)	✓		
M4 Daily	4,227	D	7	**0.023**	0.025	**0.023**	**0.023** (0.0)	**0.023** (0.0)	0.022 (0.0)	✓		
M4 Hourly	414	H	24	**0.036**	0.070	0.037	0.065 (0.03)	0.038 (0.002)	0.030 (0.001)	✓		
M4 Monthly	48,000	M	12	0.085	0.085	**0.082**	0.092 (0.003)	0.089 (0.001)	0.081 (0.0)	✓		
M4 Quarterly	24,000	Q	4	0.082	0.079	**0.076**	0.084 (0.005)	0.083 (0.001)	0.075 (0.0)	✓		
M4 Weekly	359	W	1	0.050	0.052	0.050	**0.046** (0.001)	0.049 (0.001)	0.041 (0.0)	✓		
M4 Yearly	22,974	Y	1	0.130	0.111	**0.109**	0.124 (0.006)	0.116 (0.004)	0.104 (0.0)	✓		
NN5 Daily	111	D	7	0.169	0.162	0.164	0.148 (0.002)	**0.145** (0.001)	0.140 (0.0)	✓		
NN5 Weekly	111	W	1	0.090	0.088	0.089	**0.084** (0.007)	0.085 (0.001)	0.078 (0.0)	✓		
Pedestrian Counts	66	H	24	-	0.764	-	**0.230** (0.006)	0.261 (0.008)	0.238 (0.013)			✓
Tourism Monthly	366	M	12	0.095	0.101	**0.085**	0.086 (0.005)	0.103 (0.01)	0.083 (0.0)	✓		
Tourism Quarterly	427	Q	4	0.098	0.070	0.070	**0.068** (0.002)	0.083 (0.005)	0.072 (0.0)			✓
Tourism Yearly	518	Y	1	0.156	0.157	0.155	0.141 (0.016)	**0.102** (0.006)	0.152 (0.0)			✓
Vehicle Trips	262	D	7	0.100	0.115	0.103	**0.090** (0.002)	0.099 (0.005)	0.087 (0.0)	✓		
Web Traffic Weekly	145,063	W	1	0.475	$8 \cdot 10^{13}$	0.474	-	**0.223** (0.011)	0.225 (0.0)			✓

(or has a tie) against the best performing individual forecasting algorithms. The results confirm our claim that no individual family of methods (classical or neural forecasting) dominates performance, as can be seen by the distribution of the best performance between classical and neural methods.

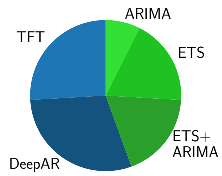
We additionally report dataset characteristics (such as the number of time series, the frequency and the seasonality of the series) to showcase that there are no simple patterns to identify the best model class. If we investigate the performance on datasets with more than 1,000 time series as an example, we see five cases where classical methods perform best and three cases where neural models perform best. Furthermore, the results show that the ensemble learned by AutoGluonTS achieves on-par or better performance than the best single model in 19 out of 28 cases, which confirms our design choice of including automated ensembling and model selection functionality in our stack.

Automated model selection. Even though ensembles perform well in general, users are often interested in running a single well-performing model only, where the forecast is easier to interpret and debug. Our AutoML component based on AG-TS supports this use case as well, and we plot its distribution of algorithm choices in Figure 2a to validate that model selection and a wide variety of approaches is indeed necessary for real-world forecasting. This data is compiled from a sample of more than ten thousand invocations of our service with highly varying datasets. Note that we sampled from a uniform distribution from all service calls to obtain a representative distribution of the "winning" model in the AutoML component. We use the selection of the winning algorithm in the AutoML component as a proxy for which algorithm is "best", as a means to arrive at a distribution of best algorithms for practical problems on proprietary datasets. We observe that deep learning-based methods (DeepAR, CNN-QR) perform best in about 60% of cases. However in the remaining 40% of scenarios, a simpler, non-neural approach (NPTS, Prophet, ETS or ARIMA) is chosen due to superior accuracy. This confirms our decision to make non-neural methods a first-class citizen in our stack.

We validate the discussed choices of our AutoML component with a comparison to the results in Table 2 on the twenty-eight public datasets from the previously discussed study of ours [65]. We plot the resulting distribution of best-performing algorithms on the public datasets in Figure 2. We observe a similar distribution of algorithm choices between neural and non-neural methods (55%



(a) Distribution of forecasting algorithm choices by our AutoML component on a large number of proprietary datasets.



(b) Distribution of the best forecasting algorithm results from our recent study [65] on 29 publicly available datasets.

Figure 2: Distribution of forecasting algorithm choices for deep learning-based methods (blue) and classical methods (green). The distribution of algorithm choices by our AutoML component (60% neural methods versus 40% classical methods, shown on the left side) reflects the distribution of the best algorithm results on publicly available datasets (55% neural methods versus 45% classical methods, as shown on the right side).

neural methods versus 45% classical methods), which confirms the representativity of our results on prorietary datasets.

Runtime. Next, we focus on the training time of our most used deep learning approaches DeepAR+ and CNN-QR during the hyperparameter search in our AutoML component. We measure the time that it takes to find a well-working configuration for runs on a large sample of several thousand complex, real-world time series datasets. The corresponding SageMaker instance is adaptively chosen based on availability and efficiency for training. For both approaches, a well-working model can be found within several minutes in the average (median) case, and this time is only increased by a factor of roughly four when we investigate the 90th percentile of the runtimes. These results indicate that the main challenge in forecasting is not computational cost, but rather how to algorithmically select an appropriate model.

5.2 Learnings from Resource Provisioning for Cloud Database Services

Next, we discuss our learnings from Amazon-internal use cases of database services within AWS in order to showcase the flexibility of our forecasting stack. In contrast to this, our previously presented monolithic platform [8] was geared towards retail demand forecasting only. The presented cases leverage different parts of our stack and confirm that its design allows users to choose the appropriate entry point for their scenario, based on their ML expertise and willingness to invest in operations and maintenance.

We choose to present the resource provisioning use cases because we think they are of particular interest to the data management community. However, we would like to note that our stack is in active use for a wide variety of forecasting problems, including retail demand forecasting and traffic forecasting.

DynamoDB and GluonTS. At Amazon, internal teams are typically asked to predict their usage of a service over a longer horizon (e.g., a financial year in advance). This gives the team operating the service leverage to plan compute resources accordingly. The team running the distributed key-value store DynamoDB [66] provides their internal customer teams with guidance for their usage planning. This team has access to data about the demand of several internal customer teams, which is highly seasonal and regular, but consists of complex patterns. We showcase three corresponding example time series in Figure 3, which are quasi-periodic with multiple seasonalities and have spikes with differing degrees of noise levels, i.e., they are heteroscedastic. In light of such data, the team specifically looked for a forecasting method that could handle complex patterns gracefully, without common failure modes such as error accumulation over long periods of time.

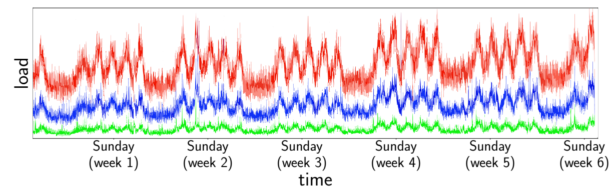


Figure 3: Examples of highly seasonal time series in internal planning scenarios for DynamoDB.

Furthermore, the team has its own data science capability so they can leverage expert-level methods in their algorithmic choice. The team settled on directly working with the deep state space models [55] available in GluonTS, which allows them to model complex seasonalities in a data-driven way, but provides control (via the state space underpinning) so that error accumulation is avoided. In order to make this choice, the data science team mapped their requirements to the model landscape. They understood that they need a data-driven, yet model-based method due to the large amount of data available to them (hence the data-driven requirement), yet they need full control on the repeating patterns (hence the model-based requirements). Such methods are refered to as deep probabilistic models, and there is only a limited number of academic publications on them available. To the best of our knowledge, none of these are available in commercial solutions (where either data-driven or model-driven approaches exist, but no hybrids). As a consequence, the team concluded that no other method fulfills their needs and confirmed this qualitatively.

Due to the open source nature of GluonTS, the team was even able to modify and improve the deep state space models for their particular use case. When examining the the model's capability, they found that solely data-driven approaches either suffered from halluscinations for longer horizons or were not able to capture the complex seasonalities well enough. Solely model driven approaches on the other hand had difficulties with the large number of overlapping seasonalities. The productization of the model via the integration of GluonTS into SageMaker only required minor engineering efforts.

Athena and SageMaker DeepAR. Athena, available at https://aws.amazon.com/athena/, is an AWS offering to analyze data in Amazon S3 using standard SQL. It is built on a serverless design for AWS customers, meaning that resource provisioning is automated. From an operations perspective, this requires a sizable warm-pool

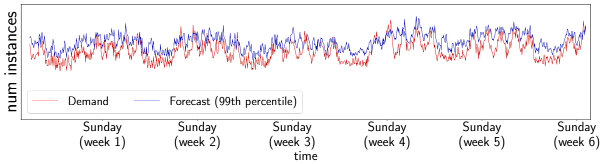


Figure 4: Demand for Athena instances (red) versus the 99th percentile of a forecast (blue) for 12 hours ahead.

of instances to be able to react quickly to customer demand. The Athena team initially employed heuristics for warm-pool sizing, and later on replaced them with DeepAR in SageMaker which led to an enormous reduction in warm pool sizes (between 30%-45% for specific instance types and availability zones) without impacting the customer experience. An interesting insight from this use case is that cross-learning across many time series is beneficial, based on pooling data from different regions for example. Global forecasting models such as DeepAR can take advantage of this, and we saw the expected increase in accuracy.

The Athena team was able to increase its cluster utilization by a factor of four. In order to guarantee a consistent customer experience, a probabilistic forecast is crucial, in order to account for the level of uncertainty when making decisions. This requires a high degree of customization of the settings for the core forecasting algorithm while guaranteeing a relatively low latency at inference time. Therefore, the Athena team chose to build on the SageMaker layer and DeepAR in SageMaker. Figure 4 shows the forecast for Athena instances (blue) versus the actuals (red) for the 99th percentile of the forecast distribution. The forecast is generally above the actuals which is required to ensure that enough instances are available, while drastic overprovisioning is avoided.

Redshift and Forecast. Redshift [50] is a data warehousing service. For the classical version, where customers explicitly choose a Redshift cluster, Redshift maintains a cache pool of EC2 instances which brings down the time to have a cluster fully operational, to improve customer experience. Analogous to the previous use case, a trade-off has to be made between reducing the resource provisioning time and having a high internal cost for reserving instances. These cache pools are specific per configuration, where a typical configuration consists among other things of the EC2 type or the data center region as well as Redshift-specific configuration settings. This means that the actual usage time series can be of an intermittent nature (where many configurations have no requests over a given period of time).

For Redshift, addressing the cost for overstocking inventory was important, but they wanted a maintenance-free solution without having to dive deeply into ML details. Therefore, Amazon Forecast was the logical choice for them: due to its nature as a hosted service, the Redshift team did not have to worry about the maintenance of ML components. Furthermore, Amazon Forecast supports a range of models which the AutoML component picks automatically on behalf of the customer. In this specific case, the AutoML component of Forecast chose a model that is specifically geared towards intermittent or sparse data. The probabilistic nature of the forecast allows the Redshift team to define service levels using the quantiles of the forecast which made the trade-off between provisiong time and cost savings explicit. The remaining work for the Redshift team consisted of making their data amenable for training and inference around Forecast, which they conducted with standard AWS tooling [43]. The utilization gains relative to an existing baseline solution ranged between 35% and 91% for specific instance types and availability zones, resulting in millions of dollars of annual operational cost savings.

6 CONCLUSION

We detailed the Amazon Web Services (AWS) forecasting stack, consisting of several layers that support a wide range of forecasting models and automate common tasks like model selection. This stack is the basis for the Amazon Forecast service and leverages our open source forecasting library GluonTS and the scalable ML platform SageMaker. Furthermore, we evaluated the predictive performance and runtime of the contained forecasting models, and discussed our learnings from using this stack to provision resources for the cloud database services DynamoDB, Redshift and Athena.

In the future, we expect foundation models to make their way into forecasting (due to promising early results [17, 26, 58]), and become available on services like Hugging Face or Bedrock. Our stack can be used to build and train such foundation models, and may be extended to incorporate them via fine-tuning.

REFERENCES

[1] 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 Türkmen, and Yuyang Wang. 2019. GluonTS: Probabilistic Time Series Models in Python. *arXiv preprint arXiv:1906.05264* (2019).
[2] Amazon Web Services. 2023. CNN-QR Algorithm. Retrieved July 16, 2024 from https://docs.aws.amazon.com/forecast/latest/dg/aws-forecast-algo-cnnqr.html
[3] 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. 2024. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815* (2024).
[4] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. Tfx: A tensorflow-based production-scale machine learning platform. *KDD* (2017), 1387–1395.
[5] Jan Beitner. 2024. PyTorch Forecasting. Retrieved July 16, 2024 from https://pytorch-forecasting.readthedocs.io/en/stable/
[6] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, Laurent Callot, and Tim Januschowski. 2020. Neural forecasting: Introduction and literature overview. arXiv:2004.10240
[7] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. 2022. Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. *ACM Comput. Surv.* 55, 6 (2022), 36. https://doi.org/10.1145/3533382
[8] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang. 2017. Probabilistic demand forecasting at scale. *PVLDB* 10, 12 (2017), 1694–1705.
[9] Nicolas Bruno, Johnny Debrodt, Chujun Song, and Wei Zheng. 2022. Computation reuse via fusion in Amazon Athena. *ICDE* (2022), 1610–1620.
[10] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. *ICML* (2004), 18.
[11] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *NeurIPS Workshop on Machine Learning Systems* (2015).
[12] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. 2023. A decoder-only foundation model for time-series forecasting. *arXiv:2310.10688* (2023).

[13] Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. 2023. ForecastPFN: Synthetically-Trained Zero-Shot Forecasting. *NeurIPS* (2023).
[14] Carson Eisenach, Yagna Patel, and Dhruv Madeka. 2020. MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention. *arXiv preprint arXiv:2009.14799* (2020).
[15] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505* (2020).
[16] Christos Faloutsos, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. 2018. Forecasting Big Time Series: Old and New. *PVLDB* 11, 12 (2018), 2102–2105.
[17] Azul Garza and Max Mergenthaler-Canseco. 2023. TimeGPT-1. arXiv:2310.03589
[18] Federico Garza Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G. Olivares. 2022. StatsForecast: Lightning fast forecasting with statistical and econometric models. Retrieved July 16, 2024 from https://github.com/Nixtla/statsforecast
[19] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. 2019. Probabilistic Forecasting with Spline Quantile Function RNNs. *AISTATS* (2019), 1901–1910.
[20] John Geweke. 1977. The dynamic factor analysis of economic time series. *Latent variables in socio-economic models* (1977).
[21] Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. 2024. Amlb: an automl benchmark. *Journal of Machine Learning Research* 25, 101 (2024), 1–65.
[22] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. 2007. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69, 2 (2007), 243–268.
[23] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. 2021. Monash Time Series Forecasting Archive. arXiv:2105.06643
[24] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. MOMENT: A Family of Open Time-series Foundation Models. *arXiv preprint arXiv:2402.03885* (2024).
[25] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023. Large Language Models Are Zero-Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems*.
[26] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. 2024. Large language models are zero-shot time series forecasters. *NeurIPS* 36 (2024).
[27] Hilaf Hasson, Bernie Wang, Tim Januschowski, and Jan Gasthaus. 2021. Probabilistic forecasting: A level-set approach. *NeurIPS* 34 (2021), 6404–6416.
[28] Julien Herzen, Francesco Lassig, Samuele Giuliano Piazzetta, Thomas Neuer, Lao Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Koacisz, Dennis Bader, Frederick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gael Grosch. 2022. Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research* 23, 124 (2022), 1–6.
[29] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. 2008. *Forecasting with exponential smoothing: the state space approach.* Springer Science & Business Media.
[30] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice.* OTexts.
[31] Rob J Hyndman and Yeasmin Khandakar. 2008. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* (2008).
[32] Rob J Hyndman and Anne B Koehler. 2006. Another look at measures of forecast accuracy. *International Journal of Forecasting* 22, 4 (2006), 679–688.
[33] Tim Januschowski, Jan Gasthaus, and Yuyang Wang. 2019. Open-Source Forecasting Tools in Python. *Foresight: The International Journal of Applied Forecasting* 55 (2019), 20–26.
[34] Tim Januschowski, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot. 2020. Criteria for classifying forecasting methods. *International Journal of Forecasting* 36, 1 (2020), 167–177.
[35] Tim Januschowski and Stephan Kolassa. 2019. A Classification of Business Forecasting Problems. *Foresight: The International Journal of Applied Forecasting* 52 (2019), 36–43.
[36] Tim Januschowski, Yuyang Wang, Kari Torkkola, Timo Erkilla, Hilaf Hasson, and Jan Gasthaus. 2022. Forecasting with Trees. *International Journal of Forecasting* 38, 4 (2022), 1473–1481.
[37] Yunho Jeon and Sihyeon Seong. 2022. Robust recurrent network model for intermittent time-series forecasting. *International Journal of Forecasting* 38, 4 (2022), 1415–1425.
[38] 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. *ICLR* (2024).
[39] Manuel Kunz, Stefan Birr, Mones Raslan, Lei Ma, Zhen Li, Adele Gouttes, Mateusz Koren, Tofigh Naghibi, Johannes Stephan, Mariia Bulycheva, Matthias Grzeschik, Armin Kekić, Michael Narodovitch, Kashif Rasul, Julian Sieber, and Tim Januschowski. 2023. Deep Learning based Forecasting: a case study from the online fashion industry. arXiv:2305.14406
[40] Richard Kurle, Syama Sundar Rangapuram, Emmanuel de Bézenac, Stephan Günnemann, and Jan Gasthaus. 2020. Deep Rao-Blackwellised Particle Filters for Time Series Forecasting. *NeurIPS* 33 (2020), 15371–15382.
[41] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Amir Astashonok, Piali Das, Can Balioglu, Saswata Charkravarty, Madhav Jha, Philip Gaultier, Tim Januschowski, Valentin Flunkert, Bernie Wang, Jan Gasthaus, Syama Rangapuram, David Salinas, Sebastian Schelter, David Arpin, and Alexander Smola. 2020. Elastic Machine Learning Algorithms in Amazon SageMaker. *SIGMOD* (2020), 731–737.
[42] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. 2021. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.
[43] Zhixing Ma, Srinivas Chakravarthi Thandu, Rohit Menon, Vivek Ramamoorthy, and Bernie Wang. 2020. Automating your Amazon Forecast workflow with Lambda, Step Functions, and CloudWatch Events rule. Retrieved July 16, 2024 from https://aws.amazon.com/blogs/machine-learning/automating-your-amazon-forecast-workflow-with-lambda-step-functions-and-cloudwatch-events-rule/
[44] Spyros Makridakis and Michele Hibon. 2000. The M3-Competition: results, conclusions and implications. *International journal of forecasting* 16, 4 (2000), 451–476.
[45] Spyros Makridakis and Evangelos Spiliotis. 2021. The M5 Competition and the Future of Human Expertise in Forecasting. *Foresight: The International Journal of Applied Forecasting* 60 (2021), 33–37.
[46] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2018. The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34, 4 (2018), 802–808.
[47] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting* 36, 1 (2020), 54–74.
[48] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting* 38, 4 (2022), 1346–1364.
[49] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *ICLR* (2020).
[50] Ippokratis Pandis. 2021. The evolution of Amazon redshift. *PVLDB* 14, 12 (2021), 3162–3174.
[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS* (2019), 8024–8035.
[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[53] Valerio Perrone, Huibin Shen, Aida Zolic, Iaroslav Shcherbatyi, Amr Ahmed, Tanya Bansal, Michele Donini, Fela Winkelmolen, Rodolphe Jenatton, Jean Baptiste Faddoul, Barbara Pogorzelska, Miroslav Miladinovic, Krishnaram Kenthapadi, Matthias Seeger, and Cédric Archambeau. 2021. Amazon SageMaker Automatic Model Tuning: Scalable Gradient-Free Optimization. arXiv:2012.08489
[54] Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K. Barrow, Souhaib Ben Taieb, Christoph Bergmeir, Ricardo J. Bessa, Jakub Bijak, John E. Boylan, Jethro Browell, Claudio Carnevale, Jennifer L. Castle, Pasquale Cirillo, Michael P. Clements, Clara Cordeiro, Fernando Luiz Cyrino Oliveira, Shari De Baets, Alexander Dokumentov, Joanne Ellison, Piotr Fiszeder, Philip Hans Franses, David T. Frazier, Michael Gilliland, M. Sinan Gönül, Paul Goodwin, Luigi Grossi, Yael Grushka-Cockayne, Mariangela Guidolin, Massimo Guidolin, Ulrich Gunter, Xiaojia Guo, Renato Guseo, Nigel Harvey, David F. Hendry, Ross Hollyman, Tim Januschowski, Jooyoung Jeon, Victor Richmond R. Jose, Yanfei Kang, Anne B. Koehler, Stephan Kolassa, Nikolaos Kourentzes, Sonia Leva, Feng Li, Konstantia Litsiou, Spyros Makridakis, Gael M. Martin, Andrew B. Martinez, Sheik Meeran, Theodore Modis, Konstantinos Nikolopoulos, Dilek Önkal, Alessia Paccagnini, Anastasios Panagiotelis, Ioannis Panapakidis, Jose M. Pavía, Manuela Pedio, Diego J. Pedregal, Pierre Pinson, Patrícia Ramos, David E. Rapach, J. James Reade, Bahman Rostami-Tabar, Michał Rubaszek, Georgios Sermpinis, Han Lin Shang, Evangelos Spiliotis, Aris A. Syntetos, Priyanga Dilini Talagala, Thiyanga S. Talagala, Len Tashman, Dimitrios Thomakos, Thordis Thorarinsdottir, Ezio Todini, Juan Ramón Trapero Arenas, Xiaoqian Wang, Robert L. Winkler, Alisa Yusupova, and Florian Ziel. 2022. Forecasting: theory and practice. *International Journal of Forecasting* 38, 3 (2022), 705–871.
[55] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep State Space Models for Time Series Forecasting. *NeurIPS* 31 (2018).

[56] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*. 7785–7794.

[57] Syama Sundar Rangapuram, Lucien Werner, Pedro Mercado Lopez, Konstantinos Benidis, Jan Gasthaus, and Tim Januschowski. 2021. End-to-End Learning of Coherent Probabilistic Forecasts for Hierarchical Time Series. *ICML* (2021), 8832–8843.

[58] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Vincent Hassen, Anderson Schneider, Sahil Garg, Alexandre Drouin, Nicolas Chapados, Yuriy Nevmyvaka, and Irina Rish. 2023. Lag-Llama: Towards Foundation Models for Time Series Forecasting. arXiv:2310.08278

[59] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. 2021. Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. *ICML* (2021), 8857–8868.

[60] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. 2021. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. *ICLR* (2021).

[61] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. 2019. High-dimensional multivariate forecasting with low-rank Gaussian Copula Processes. *NeurIPS* 32 (2019).

[62] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[63] Sebastian Schelter, Felix Biessmann, Tim Januschowski, David Salinas, Stephan Seufert, and Gyuri Szarvas. 2018. On challenges in machine learning model management. *IEEE Data Engineering Bulletin* (2018).

[64] Matthias W Seeger, David Salinas, and Valentin Flunkert. 2016. Bayesian intermittent demand forecasting for large inventories. *NeurIPS* (2016), 4646–4654.

[65] Oleksandr Shchur, Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Yuyang Wang. 2023. AutoGluon-TimeSeries: AutoML for Probabilistic Time Series Forecasting. *arXiv preprint arXiv:2308.05566* (2023).

[66] Swaminathan Sivasubramanian. 2012. Amazon DynamoDB: A Seamlessly Scalable Non-Relational Database Service. *SIGMOD* (2012), 729–730.

[67] Dylan Slack, Nathalie Rauschmayr, and Krishnaram Kenthapadi. 2021. Defuse: Harnessing Unrestricted Adversarial Examples for Debugging Models Beyond Test Accuracy. arXiv:2102.06162

[68] S Smyl, J Ranganathan, and A Pasqua. 2018. M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model. Retrieved July 16, 2024 from https://eng.uber.com/m4-forecasting-competition

[69] Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke. 2021. Probabilistic Gradient Boosting Machines for Large-Scale Probabilistic Regression. *KDD* (2021), 1510–1520.

[70] Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke. 2022. Parameter-efficient deep probabilistic forecasting. *International Journal of Forecasting* 39, 1 (2022), 332–345.

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

[72] Ali Caner Turkmen, Yuyang Wang, and Tim Januschowski. 2019. Forecasting intermittent and sparse time series: A unified probabilistic framework via deep renewal processes. *PlosOne* (2019).

[73] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. 2019. Deep factors for forecasting. *ICML* (2019), 6607–6617.

[74] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053* (2017).

[75] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592* (2024).

[76] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* 59, 11 (2016), 56–65.