Measuring Time Series Forecast Stability for Demand Planning

Steven Klee*
sklee@amazon.com
Amazon Web Services
Seattle, WA, USA

Yuntian Xia*
alicixia@amazon.com
Amazon Web Services
Atlanta, GA, USA

Abstract

Time series forecasting is a critical first step in generating demand plans for supply chains. Experiments on time series models typically focus on demonstrating improvements in forecast accuracy over existing/baseline solutions, quantified according to some accuracy metric. There is no doubt that forecast accuracy is important; however in production systems, demand planners often value consistency and stability over incremental accuracy improvements. Assuming that the inputs have not changed significantly, forecasts that vary drastically from one planning cycle to the next require high amounts of human intervention, which frustrates demand planners and can even cause them to lose trust in ML forecasting models. We study *model-induced stochasticity*, which quantifies the variance of a set of forecasts produced by a single model when the set of inputs is fixed. Models with lower variance are more stable.

Recently the forecasting community has seen significant advances in forecast accuracy through the development of deep machine learning models for time series forecasting. We perform a case study measuring the stability and accuracy of state-of-the-art forecasting models (Chronos, DeepAR, PatchTST, Temporal Fusion Transformer, TiDE, and the AutoGluon best quality ensemble) on public data sets from the M5 competition and Favorita grocery sales. We show that ensemble models improve stability without significantly deteriorating (or even improving) forecast accuracy. While these results may not be surprising, the main point of this paper is to propose the need for further study of forecast stability for models that are being deployed in production systems.

ACM Reference Format:
Steven Klee and Yuntian Xia. 2025. Measuring Time Series Forecast Stability for Demand Planning. In *Proceedings of the 1st Workshop on "AI for Supply Chain: Today and Future" @ 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 Introduction

Forecasting future product demand is the first step in a supply chain system. Planners require accurate forecasts so that they can meet customer demand by delivering the right quantity of products to the right locations at the right time. When designing a forecasting system, it is common to perform backtesting experiments, measuring the performance of different models across test forecast windows. For a chosen forecast start date, training and cross validation are done over dates before the start date, and testing is done over the window after the start date. One or more metrics may then be reported on the test window(s) so that the performance of different models can be compared.

The simplicity of reporting a single global accuracy metric is convenient for comparing models, whether that experimentation is done in the context of developing a new model or deciding on which model to deploy in a production system, however it does not tell the whole story. One issue comes from simple mathematics: if multiple time series are forecast over a multi-week horizon, then the same or similar values of a global metric (for example root mean squared error) can be achieved by infinitely many different combinations of individual forecasts.

Many local statistical models (e.g., ARIMA) are deterministic, meaning the same set of inputs will yield the same set of forecast outputs as long as the hyperparameters (number of lags, number of differences, order of moving average) are kept constant. However this is not always the case, especially for many of the new state-of-the-art deep learning models. Potential sources of what we will call *model-induced stochasticity* include randomized train/test split methods or stochastic gradient descent methods in the optimization engine. Coupled with non-convex loss functions, this means the same set of forecast inputs can yield different outputs.

In practice, variance in demand forecasts presents a significant challenge to demand planners. A signal whose global accuracy is acceptable but highly variable frustrates planners who ultimately make purchasing decisions across different planning horizons based on different vendor lead times. We propose therefore that it is important to consider model stability when deploying forecasting models in production systems.

We quantify forecast stability by using the same model with the same hyperparameter settings to perform training and inference on a fixed data set. By repeating this process several times, changing only the random seed, we obtain multiple forecasts drawn from some unknown sample space, and we compute the variance of the outputs over all items to be forecasted and across the forecast time horizon. Precise details are outlined in Section 2.

We note also that there is a second dimension to this problem, namely cycle-to-cycle change, which measures the extent to which model outputs change from one forecast snapshot date to the next. For the purposes of this paper we fix the set of inputs so that the output variance can be attributed to the model alone. Cycle-to-cycle change is undoubtedly important and worthy of additional study.

The fundamental question we wish to address in this paper is: how much inherent stochasticity do different models introduce; or

*Both authors contributed equally to this research.

conversely, how stable are different forecasting models? We use the AutoGluon-TimeSeries library [1, 12] to train multiple models on publicly-available data sets that represent different forms of consumer demand: the M5 data set [6, 8], which contains Walmart product demand in North America; and the Favorita data set [4], which contains grocery demand for a South American retail chain.

For each data set and each forecasting model, we measure forecast stability through the coefficient of variation (CV) across ten forecast runs with different random seeds but otherwise the same sets of inputs and model hyperparameters. We then demonstrate trade-off between overall model accuracy and forecast stability. Our main results demonstrate that, perhaps unsurprisingly, ensembled models are more stable than individual deep learning models while also achieving comparable or better accuracy.

The rest of the paper is structured as follows. In Section 2, we provide an overview of our methodology. In Section 3 we summarize our results on forecast stability and forecast accuracy. Details on hyperparameters and the AutoGluon ensemble model are included in the Appendices.

2 Methodology

2.1 Measuring model stability

Given N samples drawn from a distribution, the coefficient of variation is defined as $CV = \frac{\sigma}{\mu}$, where σ and μ are, respectively, the sample standard deviation and mean. The CV is a normalized measure of dispersion of the sample about the mean. When the sample is a set of model outputs, lower values of CV correspond to higher model stability.

In the domain of time series forecasting, suppose we have M time series with histories of length T and values $\{x_{i,t}\}_{i=1}^{M}{}_{t=1}^{T}$. A forecasting model predicts future values $\{\hat{y}_{i,t}\}_{i=1}^{M}{}_{t=T+1}^{T+H}$, where H is the length of the forecast horizon. If a model outputs a distributional forecast, we take $\hat{y}_{i,t}$ to be the center (mean) of the forecast distribution. Re-training and performing inference R times yields multiple forecasts $\{\hat{y}_{i,t,r}\}_{i=1}^{M}{}_{t=T+1}^{T+H}{}_{r=1}^{R}$. Treating the set of forecasted values for each time series and timestamp as a sample drawn from an unknown distribution gives $M \cdot H$ different coefficients of variation $CV(i,t)$, computed over the sample of R different forecasts $\{\hat{y}_{i,t,r}\}_{r=1}^{R}$.

The CV metric has a few drawbacks. After stating the drawbacks we will describe how they can be mitigated in the context of demand forecasting.

(1) If μ is negative, then so is CV, which does not make sense as a measure of dispersion.
(2) If μ is zero, CV is undefined.
(3) The CV value can be skewed if μ and σ are small. For example, if $\sigma = 10^{-3}$ and $\mu = 10^{-8}$, then $CV = 10^5$.

Most time series forecasting libraries do not safeguard against negative forecasts because there may be use cases where negative forecasts are valid (e.g., weather forecasts). Even in the context of demand forecasting, small or zero forecast values may be expected, especially when working with intermittent/sparse demand. To overcome these issues, we apply the following forecast post-processing logic before computing $CV(i,t)$:

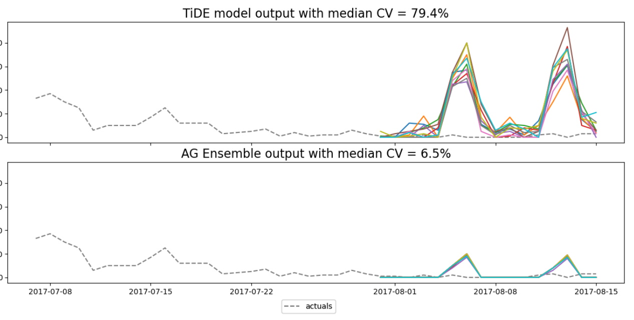

Figure 1: Model outputs across multiple runs for an item from the Favorita catalog. Higher CV values (top) indicate a less stable forecast with a wider range of predicted outputs. Lower CV values (bottom) indicate a more stable forecast. Dashed lines are actual values; solid lines are the outputs of 10 forecast runs.

(1) Replace negative forecasted values with zeros. This is a common approach in demand forecasting to overcome the issue of negative forecasted values.
(2) Round each forecasted value to the nearest integer. This overcomes the issue of small values of μ which can arise when the values of $\hat{y}_{i,t,r}$ are near zero; and once again we argue it is sensible in the context of forecasting physical units of demand, which must be integers.
(3) If $\mu = 0$ and $\sigma = 0$, declare $CV = 0$. This happens only in the case of constant zero forecasts. If $\sigma = 0$ it makes sense to also say $CV = 0$ independent of the mean because the data are unchanging.

As an example, Figure 1 shows two distribution outputs from an item in the Favorita data set. The TiDE model output has a high CV, with the median value of $CV(i,t)$ near 80% as the time variable ranges over the forecast horizon. In comparison, the AutoGluon ensemble produces a more stable forecast, with a median CV value of 6.5%. We see this reflected in the plot as the range of values output by the TiDE model have more uncertainty over the 10 model runs. Again, to reiterate the methodology, each visualization of a run output shown in this plot is the mean forecast value produced by retraining at TiDE/AutoGluon model with the same set of inputs and only changing the random seed.

2.2 Measuring forecast accuracy

In addition to measuring forecast stability we must also demonstrate a trade-off with forecast accuracy. For each data set we measure forecast accuracy using root mean squared error (RMSE) over the test window. For each forecast run, RMSE is computed as

$$RMSE(r) = \sqrt{\frac{1}{M \cdot H} \sum_{i=1}^{M} \sum_{t=T+1}^{T+H} \left(\hat{y}_{i,t,r} - x_{i,t}\right)^2}$$

It is important to note that RMSE is measured in the same units of demand as x and $\hat{y}$. Because it is not normalized it cannot be used to make comparisons across data sets with different demand volumes. This could be overcome with proportional metrics such as MAPE, but that leads to a multitude of other issues, especially for intermittent/sparse demand patterns with many zeros.

2.3 Summary of data sets

We use two public data sets of retail demand data in our analysis.

- The M5 data set contains sales data for retail products sold by Walmart in North America [6, 8]. It was used in the 2020 M5 accuracy competition hosted through Kaggle. The M5 data are given at a very low level of granularity (product + geographical region). To mitigate some of the effects of data sparsity, we aggregate to the product level. This was a common approach in the M5 contest as well, where aggregation-disaggregation methods were used to forecast lower levels of the hierarchy.
- The Favorita data set contains sales data for Corporación Favorita, a large Ecuadorian-based grocery retailer. The data were used in a 2017 forecast accuracy competition, also hosted through Kaggle [4]. Once again we aggregate to product-level granularity.

The properties of these data are summarized in Table 1.

2.4 Summary of models used

Table 2 summarizes the time series forecasting models that will be used for measuring stability and accuracy. This includes some models that do not introduce stochasticity but are nonetheless important for accuracy comparisons. For deep learning models which are used in a standalone fashion, we performed hyperparameter tuning using grid search. Those models are DeepAR [11], Temporal Fusion Transformer (TFT) [7], Patch TST [9], and TiDE [3].

In addition, we include the Chronos model [2] as zero shot and with fine tuning, along with the AutoGluon time series ensemble (AG ensemble) [1, 12] with "best quality" presets. The AutoGluon ensemble is a stacked ensemble comprised of 12 individual models. It includes some of the above deep learning models; however we do not perform hyperparameter tuning for those models, instead relying on the default settings determined by AutoGluon as an AutoML solution. Further details on the ensemble are outlined in Table 4 of Appendix A.

All model training and execution is done through the AutoGluon TimeSeries Library (AutoGluon release 1.2.0). In Table 2 below we indicate whether hyperparameter tuning was performed in the "HPO" column. The "is stochastic" column indicates whether the model outputs change based on setting a random seed. The chosen hyperparameter configurations, which were found by grid search, are listed in Table 5 of Appendix B.

2.5 Related work

Problems around forecast variance have been studied in the work of Godahewa et al. [5] and Pritularga and Kourentzes [10]. Godahewa et al. [5] consider consider multiple perspectives on this. The first, which they call *horizontal* stability, is the variance of a forecast made on a single snapshot date over the time horizon. The second, which they call *vertical* stability, is the variance across a set of forecasts made on different snapshot dates, but for a fixed future timestamp. Pritularga and Kourentzes [10] take a similar perspective, measuring variance in the time domain across multiple forecast snapshot dates.

In contrast, the focus of this paper is neither horizontal nor vertical stability, but rather model-induced stability/instability. We measure the extent to which a forecasting model can be viewed as a random process where the variance comes only from changing a random seed, when the snapshot date, input data, and forecast horizon are fixed. Godahewa et al. [5] call this *forecast replicability* but do not study this aspect because the focus of their work is mitigation strategies against such forecast variance, which they note can be mitigated through ensembling in this case. This is consistent with our observations, however they do not attempt to quantify forecast replicability, or what we call model-induced stochasticity.

3 Results

Figure 2 shows histograms of CV scores over each data set for each stochastic model. The tails of the distributions are clipped for visualization purposes, but as we see in the summary statistics of Table 3 the histograms contain over 90% of the data. Note the vertical axes are on a logarithmic scale. In each plot, the dashed red line shows the median of the distribution. As noted in Section 2, for each data set we compute CV for $M \cdot H$ different combinations of item and forecast horizon. To simplify the visualizations, we show only the global distributions of all $M \cdot H$ values of CV without separating the time component from the individual items.

From this we see that the AG ensemble consistently minimizes CV, meaning its outputs are more stable than individual deep learning models. This is not surprising given that several of the constituent models in the ensemble (especially Chronos) do not introduce stochasticity. From a business perspective, the following interpretation of the results in Table 3 can be made: at least 10% of the time series forecast by deep learning models can see at least 10%, and in some cases nearly 20%, normalized variance, *even when trained on the same set of inputs*. In contrast, the AutoGluon ensemble mitigates that variance to less than 5%. From a planning perspective, this is a stark difference; 20% changes in output which can be attributed only to stochasticity are likely to be unacceptable for demand planners.

Figure 3 shows the distribution of forecast errors as measured by RMSE for each data set and model. For the M5 data we see that the AG ensemble consistently produces more accurate forecasts with tighter distribution of forecast error. The results on Favorita are different. In that case, the AG ensemble is out-performed by both Chronos and the deep learning forecast models. This result is somewhat surprising. On the one hand, the hyperparameter tuning for the deep learning models was not passed into the AG ensemble. Therefore it is conceivable that the tuned models could outperform their counterparts in the AG ensemble. On the other hand, the Chronos models were selected as part of the AG ensemble, and so it is still unexpected that the ensemble would be outperformed by the Chronos models by such a wide margin. At this point, business planners would need to understand their tolerance for risk and variance in forecast outputs as a trade-off against accuracy improvements.

Table 1: Summary of data sets

Data set	Granularity	# time series (M)	History length (T)	Forecast horizon (H)
M5	daily	3049	1913	28
Favorita	daily	4036	1672	16

Table 2: Models used in experiments

Model	is stochastic	HPO
Chronos (zero shot)	no	no
Chronos (fine tuned)	no	no
DeepAR	yes	yes
TFT	yes	yes
Patch TST	yes	yes
TiDE	yes	yes
AG ensemble	yes	no

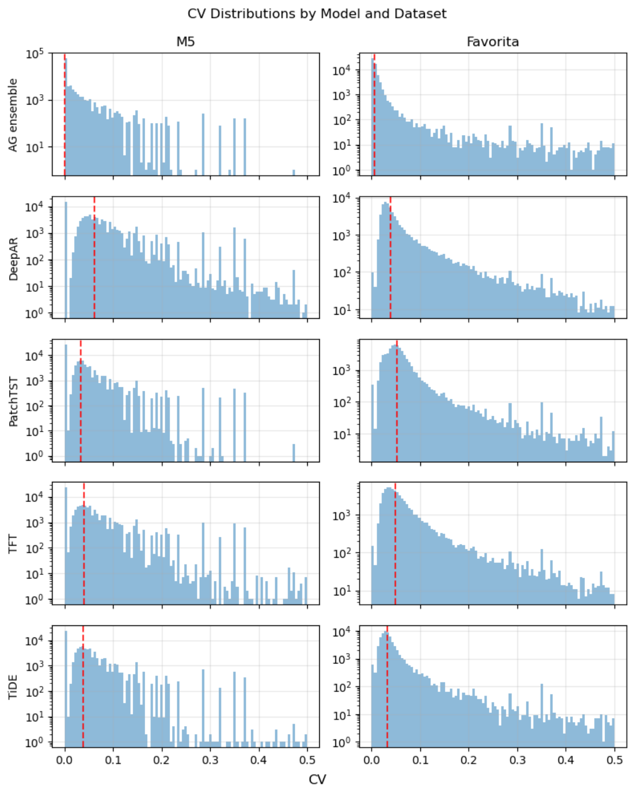

Figure 2: Distribution of CV scores across data sets and stochastic models.

Table 3: Quantiles of the distribution of CV values over data sets and stochastic models

Data set	Model	25%	50%	75%	90%
M5	AG ensemble	0.000	0.000	0.012	0.048
	DeepAR	0.036	0.061	0.102	0.192
	PatchTST	0.000	0.033	0.057	0.111
	TFT	0.000	0.041	0.077	0.152
	TiDE	0.000	0.039	0.065	0.112
Favorita	AG ensemble	0.007	0.011	0.022	0.043
	DeepAR	0.028	0.039	0.069	0.150
	PatchTST	0.039	0.052	0.069	0.106
	TFT	0.034	0.050	0.077	0.141
	TiDE	0.024	0.032	0.044	0.073

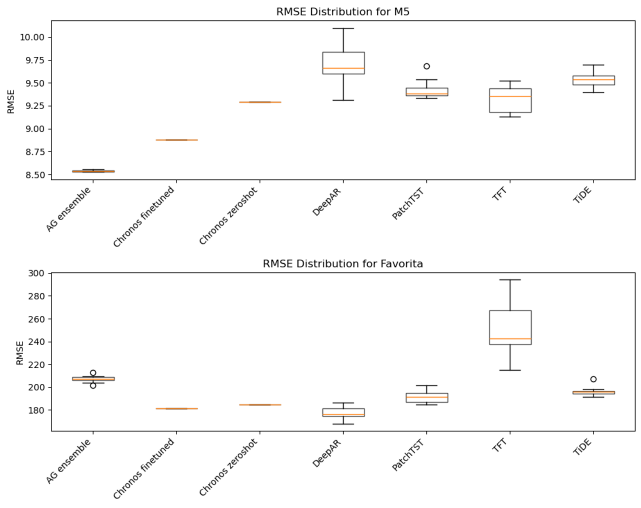

Figure 3: Distribution of forecast error across data sets and models.

4 Conclusion

This study investigated the stability of various time series forecasting models through multiple runs, specifically examining AutoGluon Ensemble, Chronos, TFT, DeepAR, TiDE, and PatchTST. Stability was assessed using both coefficient of variation (CV) of raw outputs and the distribution of Root Mean Square Errors (RMSE). The results demonstrated that AutoGluon Ensemble exhibited the highest stability among the tested models, showing the lowest variance in both metrics, except the RMSE for the Favorita data set. This superior stability can be attributed to its ensemble approach, as well as the fact that some statistical or otherwise deterministic models were selected in the ensemble.

The findings suggest that when considering both predictive performance and reliability of results, AutoGluon Ensemble may be particularly suitable for applications where consistent outputs across multiple runs are crucial. Future research could explore how model stability can be impacted by other factors such as model complexity, ensemble configurations, or small perturbations to the inputs. Additionally, as noted in the introduction, a deeper understanding of cycle-to-cycle stability across models is an important practical consideration for forecasting models that are deployed in production systems. It would be interesting to study cycle-to-cycle change in a way that separates the contribution of model-induced stochasticity from changes that can be attributed to the addition of new points in the training data. These insights contribute to our understanding of model reliability in practical applications and can guide practitioners in selecting appropriate forecasting models based on their stability requirements.

Acknowledgments

Thanks to an anonymous referee for pointing out the related works of Godahewa et al. and Pritularga and Kourentzes.

References

[1] [n. d.]. AutoGluon TimeSeries Forecasting Documentation. https://auto.gluon.ai/stable/tutorials/timeseries/index.html.

[2] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Syndar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. 2024. Chronos: Learning the Language of Time Series. *arXiv preprint arXiv:2403.07815* (2024).

[3] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K Mathur, Rajat Sen, and Rose Yu. 2023. Long-term Forecasting with TiDE: Time-series Dense Encoder. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=pCbC3aQB5W

[4] Corporación Favorita, inversion, Julia Elliott, and Mark McDonald. 2017. Corporación Favorita Grocery Sales Forecasting. https://kaggle.com/competitions/favorita-grocery-sales-forecasting. Kaggle.

[5] Rakshitha Godahewa, Christoph Bergmeir, Zeynep Erkin Baz, Chengjun Zhu, Zhangdi Song, Salvador García, and Dario Benavides. 2025. On forecast stability. *International Journal of Forecasting* (2025). https://doi.org/10.1016/j.ijforecast.2025.01.006

[6] Addison Howard, inversion, Spyros Makridakis, and vangelis. 2020. M5 Forecasting - Accuracy. https://kaggle.com/competitions/m5-forecasting-accuracy. Kaggle.

[7] Bryan Lim, Sercan Ö. Arık, 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. https://doi.org/10.1016/j.ijforecast.2021.03.012

[8] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting* 38, 4 (2022), 1346–1364. https://doi.org/10.1016/j.ijforecast.2021.11.013 Special Issue: M5 competition.

[9] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

[10] Kandrika Pritularga and Nikolaos Kourentzes. 2024. Forecast Congruence: A Quantity to Align Forecasts and Inventory Decisions. (2024). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4711817

[11] 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. https://doi.org/10.1016/j.ijforecast.2019.07.001

[12] Oleksandr Shchur, Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Yuyang Wang. 2023. AutoGluon-TimeSeries: AutoML for Probabilistic Time Series Forecasting. In *International Conference on Automated Machine Learning*.

A AutoGluon Ensemble

Table 4 shows the component models trained in the AutoGluon "best quality" ensemble model. The ensemble uses two cross validation windows, and fits a global ensemble by selecting the optimal convex combination of the component model outputs over the validation windows. For more information on individual model configurations we refer to the AutoGluon Model Zoo [1].

Table 4: AutoGluon Best Quality Model Ensemble

Model Name
Seasonal Naive
AutoETS
Nonparametric Time Series (NPTS)
Dynamic Optimized Theta
Recursive Tabular
Direct Tabular
TFT
Patch TST
DeepAR
Chronos bolt zero shot
Chronos bolt fine tuned
TiDE

B Hyperparameter Tuning

Table 5 shows the hyperparameters learned by grid search for each data set and each of the deep learning forecasting models.

Received 23 May 2025; revised 11 June 2025; accepted 11 June 2025

Table 5: Hyperparameter settings learned by grid search for each data set and deep learning model.

Favorita			M5		
Model	Hyperparameter	Value	Model	Hyperparameter	Value
TiDE	dropout_rate	0.01	TiDE	dropout_rate	0.4
	num_head	1		num_head	4
	batch_size	64		batch_size	256
	lr	1e-4		lr	1e-3
	distr_hidden_dim	32		distr_hidden_dim	256
	feat_proj_hidden_dim	2		feat_proj_hidden_dim	8
	encoder_hidden_dim	32		encoder_hidden_dim	256
	decoder_hidden_dim	32		decoder_hidden_dim	256
	temporal_hidden_dim	32		temporal_hidden_dim	64
	num_layers_encoder	1		num_layers_encoder	1
	num_layers_decoder	1		num_layers_decoder	2
TFT	hidden_size	76	TFT	hidden_size	92
	dropout_rate	0.7		dropout_rate	0.2
	batch_size	128		batch_size	256
	num_head	1		num_head	4
	lr	1e-3		lr	1e-3
	hidden_dim	240		hidden_dim	40
PatchTST	num_encoder_layers	4	PatchTST	num_encoder_layers	4
	nhead	4		nhead	1
	batch_size	256		batch_size	128
	d_model	320		d_model	320
	lr	1e-4		lr	1e-4
DeepAR	hidden_size	62	DeepAR	hidden_size	46
	dropout_rate	0.1		dropout_rate	0.1
	batch_size	128		batch_size	64
	num_layer	3		num_layer	4
	lr	1e-2		lr	1e-2