SMU Data Science Review

Volume 7 | Number 2 Article 1

2023

A Hybrid Ensemble of Learning Models

Bivin Sadler
Southern Methodist University, bsadler@mail.smu.edu

Dhruba Dey
Southern Methodist University, ddey@mail.smu.edu

Duy Nguyen
Southern Methodist University, dnguyen760397@gmail.com

Tavin Weeda
Southern Methodist University, tweeda@mail.smu.edu

A Hybrid Ensemble of Learning Models

Dhruba Dey[1], Tavin Weeda[2], Duy Nguyen[3], Bivin Sadler[4]
Master of Science in Data Science, Southern Methodist University,
Dallas, TX 75275 USA
[1]ddey@mail.smu.edu
[2]tweeda@mail.smu.edu
[3]duynq@mail.smu.edu
[3]dnguyen760397@gmail.com
[4]bsadler@mail.smu.edu

Abstract. Statistical models in time series forecasting have long been challenged to be superseded by the advent of deep learning models. This research proposes a new hybrid ensemble of forecasting models that combines the strengths of several strong candidates from these two model types. The proposed ensemble aims to improve the accuracy of forecasts and reduce computational complexity by leveraging the strengths of each candidate model.

1 Introduction

The task of forecasting research is a crucial part of academia and forecasting practitioners. Over the years, various forecasting models have been developed, ranging from traditional statistical models to the more advanced deep learning models that have broadened the landscape in forecasting research.

An in-depth empirical evaluation was conducted recently on the top performers from the M3 competition organized by Spyros Makridakis (Makridakis et al., 2022a). The M3 competition was a large-scale forecasting competition conducted in 2000 that involved 3,003 time series realizations from different domains including industry, finance, and demography. The focus of the competition was to identify the most accurate forecasting models across the different time series and their domains rather than on the technicalities of the models used. One of its key findings was that no single model outperformed other models cnsistently among the time series realizations. This finding confirms the need for developing a combination of models that can target both the strengths and weaknesses of multiple models in the attempt of pushing the forecasting accuracy even further.

To address this, the best performing forecasting models from each type of statistical and deep learning models are considered in this research. Statistical models can be generalized to easily capture linear relationships in time series data, require different distribution assumptions, and be computationally fast and easily interpreted. Deep learning models, however, do not require specific distribution assumptions, but can be generalized to capture both linear and non-linear relationships. Additionally, they are more computationally complex, require big data to perform efficiently, and are harder to interpret.

This research uses the same dataset from the M3 competition, specifically the 1,428 monthly realizations from the total of 3,003. Each of the realizations have lengths that varies from 66 to 144 observations and comes from six of the following domains:

Micro, *Macro*, *Finance*, *Industry*, *Demographic*, and *Other*. The forecasts are generated with statistical and deep learning models on horizons of 2 through 18, or rather 2-step-ahead through 18-step-ahead, which are then evaluated using the symmetric mean absolute percentage error, or sMAPE, a comparison metric used extensively by Makridakis et al. (2022a). See Appendix Section 4 for sMAPE definition.

Components of both trend and seasonality are exhibited throughout the 1,428 monthly realizations. Satisfactory performance may not be guaranteed when implementing some models on realizations with these components. There are many ways to determine the exhibiting realizations and transform them appropriately. Most of the models in this research may not require prior transformations, since those models have internal tests and calculations to handle these exhibiting components. Conversely, the model Autoregressive Integrated Moving Averages, or ARIMA, requires the evaluation of these components before its parameters are properly estimated. Woodward et al. (2022) recommended the Cochrane-Orcutt estimation as an efficient way to determine trends. And once the trend is determined, a simple differencing of the data is performed followed by the parameter estimation of ARIMA. However, this method has inflated type I error rates, which tend to increase as the ARIMA model stumbles across realizations with lesser observations.

Statistical models are among the most widely used applications for time series forecasting. This is often due to their use of mathematical formulas that can easily be simplified and interpreted. Exponential smoothing, another simple and effective statistical model, is also under this umbrella together with ARIMA. Exponential smoothing can be used to effectively estimate the level, trend and seasonality of a time series realization, as well as effectively produce short-term forecasts. On the other hand, ARIMA is especially useful for non-stationary data, or rather data that have means and variances that change over time; such data is unpredictable and can produce incorrect forecasts when used for modeling. To reiterate, statistical models can be useful for capturing linear relationships between variables; however, they often rely on certain distribution assumptions and struggle to capture more complex relationships in the data. Additionally, statistical models are known to be sensitive to outliers which produce forecasts that may misrepresent the data.

Deep learning models are designed to work for data with higher complexity such as images, speech, and video processing applications. Examples include recurrent neural networks, or rather RNN, which can be quite effective when used for time series data. RNNs are designed to handle sequences of inputs with various lengths where the order of the inputs is most crucial. DeepAR is a sophisticated deep learning model that is designed specifically for time series forecasting and using RNNs. It can handle difficult pre-modeling problems such as multiple seasonal components, missing data, and non-linear trends, making it well-suited for the M3 data. According to Makridakis et al. (2022a), only in certain cases DeepAR is shown to outperform the other models in their study. The focus of this research is to utilize DeepAR and attempt to reproduce the how and why the realizations are affected by it.

As mentioned earlier, by using ensembles that combine the satisfactory forecasts of multiple models, the forecasting accuracy can be improved even further. This research aims to ensemble various high-performing statistical and deep learning models to potentially achieve greater forecasting ability.

2 Literature Review

The literature review focuses on four principal areas: Statistical models, DeepAR, the performance capacities of statistical and deep learning models, and the ensemble schemes and their dynamic performances.

2.1 Statistical Models

2.2.1 Simple Exponential Smoothing

Exponential smoothing was first pioneered by Robert G. Brown in 1944 (Gardner, 2006). Brown's model produces the forecast for the next period as a weighted average of the previous forecast and observation. It uses only one parameter alpha (α) to smooth the data; for any α between 0 and 1, the weights attached to the more distant (previous) observations decrease exponentially, hence the name "exponential smoothing." if α is closer to 1, more weight is given to more recent observations, and conversely if α is closer to 0, more weight is given to the more distant past observations.

However, this model does not correctly handle trends and seasonality that might be present in some time series. For example, the annual passenger numbers for Australian airlines from 1990 to 2009, namely the *ausair* dataset from the R package *fpp*, plotted in Fig. 1 shows clear evidence of a trend. The application of simple exponential smoothing (SES) shows a constant range of forecasted values over a five-year period, plotted in blue, with prediction intervals that are increasing exponentially.

The constant range of forecasts are the results of SES only using information from the past to make forecasts. Specifically, the forecast for the next period is based on the last observed and forecasted values. With the 5-step-ahead forecast shown in Fig. 1, the subsequent forecasts after the first one can only be based on the first forecast, namely the 2010 forecast, hence the constant range of forecasts.

The prediction intervals in pink here show that, given the constant forecasts not providing much information, there is still significant uncertainty in the future values of Australian airline passengers over the five-year forecast period.

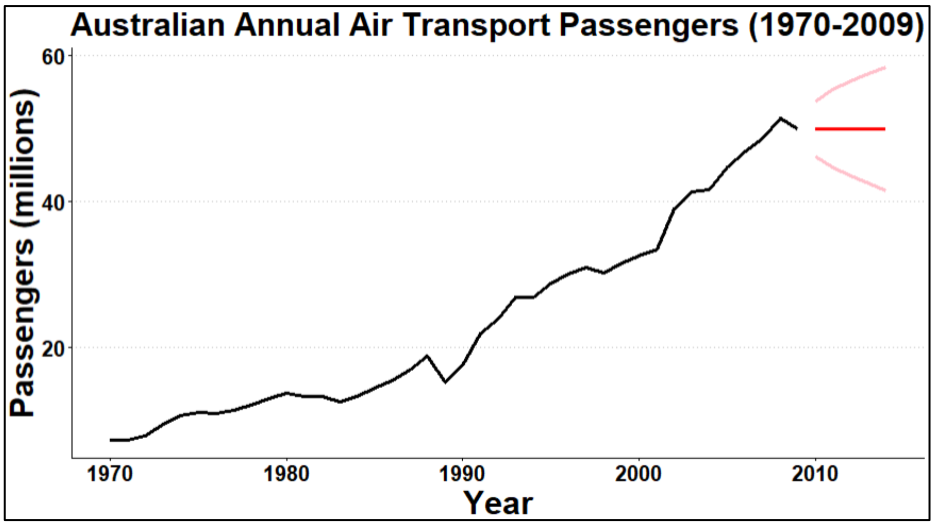


Figure 1. *ausair* Dataset. Simple exponential smoothing (red) implemented for 5-step-ahead, where the prediction intervals (pink) show an exponentially increasing uncertainty.

2.2.2 Holt-Winters' Method

Charles Holt and his student Peter Winters are well known for their contributions to exponential smoothing. In 1957, Charles Holt created Holt's linear method as an extension to simple exponential smoothing, which allows forecasting time series data with trends. This method involves using another parameter beta (β) that is used as the smoothing constant for the trend. Simply put, with this new parameter the forecasts are no longer flat but trending, essentially making the forecasts a linear function of the forecast period. With that said, the trend displayed in Holt's linear method is indefinite and can potentially overfit. The damped trend method introduced by Gardner & McKenzie (1985) solves this problem by involving another parameter phi (ϕ) that dampens the trend to a flat line. Both applications of Holt's linear and damped trend on the *ausair* dataset can be seen in Fig. 2, plotted in green and red, respectively. While the 15-step-ahead forecast can be seen here instead of a 5-step-ahead forecast, it is not practical to forecast so far in the future. The 15-step-ahead forecast was done only to showcase the curvature behavior over time of the damped trend method.

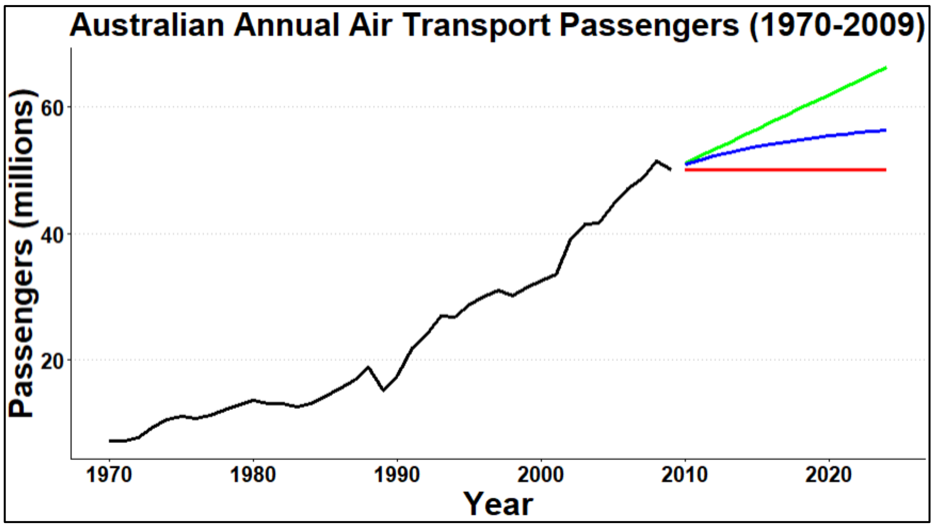


Figure 2. *ausair* Dataset. Simple exponential smoothing (red), Holt's method (green), and damped Holt's method (blue) applied for 15-step-ahead. This longer forecast is done instead of 5-step-ahead to clarify the difference between Holt's method (green) and damped Holt's method (blue).

In 1960, Charles Holt and his student Peter Winters created the Holt-Winters method (HW) to extend Holt's linear method which allows forecasting time series data with seasonality. The user can choose two variants of HW to handle seasonality within a time series. If the seasonal effects are constant over time (imagine a waves plot with the same amplitude throughout the *x*-axis, and the average level is either constant or changing), the additive variant of Holt-Winters is most appropriate. If the seasonal effects are proportional in size to the average level or change at the same rate as the trend (imagine a waves plot that increase in amplitude as the average level increases), then the multiplicative variant is most appropriate. "The user must decide whether to use the additive or multiplicative seasonal model or a non-seasonal model. … As each new observation becomes available, the local mean, the trend and the seasonal factors are all updated by exponential smoothing using three smoothing constants which we will denote by α, β, γ respectively" (Chatfield, 1978).

An example of a time series data that is most appropriate for additive HW is the quarterly *austourists* dataset from the R package *fpp*. This data can be seen in Fig. 3, along with an additive HW model for 8-step-ahead, or 8 quarters (2 years), applied to it. On the other hand, the monthly *AirPassengers* dataset from the R package *datasets* is a good example of a time series most appropriate for multiplicative HW. This dataset, which displays a growing and multiplicative seasonality, can be seen in Fig. 4, along with a multiplicative HW model for 24-step-ahead, or 24 months (2 years) applied to it.

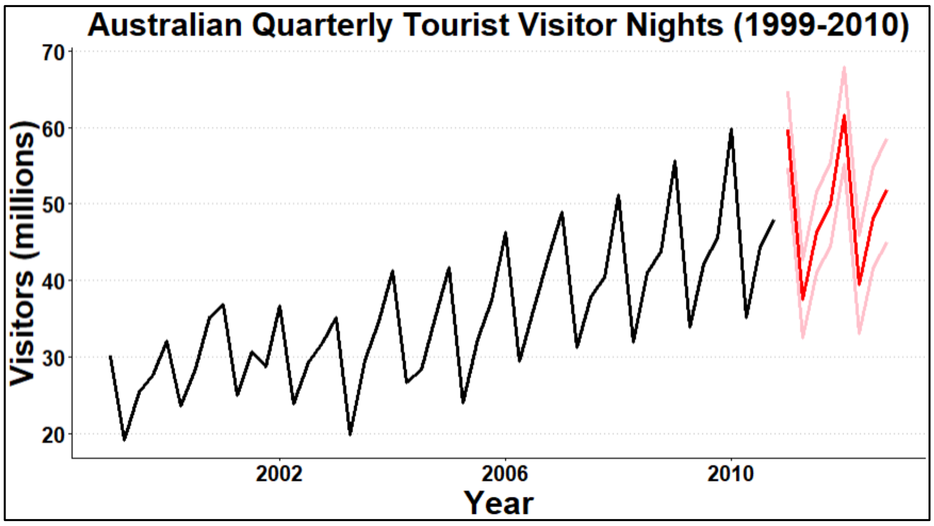


Figure 3. ***austourists*** **Dataset (1999-2010).** Holt-Winters' method with additive seasonality (red) applied for 8-step-ahead, or rather 8 quarters (2 years) into the future. The prediction intervals (pink) show a constant amount of uncertainty throughout the horizons for both upper and lower intervals.

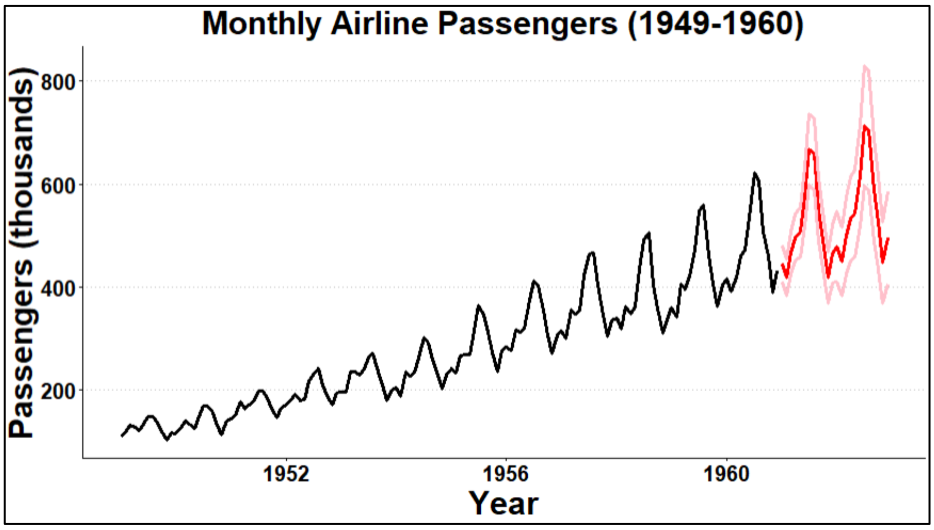


Figure 4. ***AirPassengers*** **Dataset.** Holt-Winters' method with multiplicative seasonality (red) is applied for 24-step-ahead, or rather 24 months (2 years) into the future. The prediction intervals (pink) show a constant amount of uncertainty throughout the horizons for both upper and lower intervals.

2.2.3 State Space Exponential Smoothing and the Theta method

"A state space is the set of all configurations that a given problem and its environment could achieve" (Sutcliffe, n.d., para. 1). In 2002, a new approach to exponential smoothing was proposed that allows the automation of forecasting using state space models and exponential smoothing methods (Hyndman et al., 2002). The authors explore different exponential smoothing models and argue that this framework provides a more flexible and robust way to generate forecasts that can adapt to the changing patterns in the data. The authors used the M3 data, a standard benchmark dataset widely used in the forecasting community (Makridakis & Hibon, 2000). Their goal is to investigate the advantages of different smoothing models and their effects on different horizons, time series types (monthly, yearly, etc.), and domains that are apparent in the M3 dataset.

Simply put, they developed 12 different models by combining different types of exponential smoothing methods with various specifications of the state space framework. Additionally, they have assumed that a time series can possess both the additive and the multiplicative error assumptions. These two model assumptions give equivalent forecasts despite different prediction intervals and likelihoods. Ultimately, this results in 24 total models considered and evaluated before being chosen as the best exponential smoothing model for a particular time series.

Empirical evaluations in the research suggest that this state space framework surpasses the Theta method (Assimakopoulos & Nikolopoulos, 2000), the top performer in the M3 competition in 2000, especially for the seasonal series in the shorter forecast horizons, namely horizons 1 to 4, and the *Micro* domain (Makridakis & Hibon, 2000). However, the same cannot be said for the other time series frequencies and other domains in the M3 dataset. Conversely, the Theta method produces forecasts using a weighted average of the two components resulting from the classical multiplicative decomposition method. It requires the estimation of a single parameter θ based on the trend and seasonality present in the data, hence the two components, as opposed to the multiple parameters required in the state space exponential smoothing framework. Despite its lack of complexity, the Theta method offers a wide range of practical applications as opposed to the state space exponential smoothing framework, however, the latter would still be useful in some cases with seasonal data and domain-specific cases like the *Micro* domain.

2.2.4 Complex Exponential Smoothing

Complex Exponential Smoothing (CES) was proposed by Svetunkov et al. (2022) to address the challenging task of identifying the underlying trend of a given time series data that significantly impacts forecasting accuracy. It does not exhibit model selection and uses only two parameters that derive from complex numbers. The given historical data is modeled using complex numbers, which helps capture both the trend and seasonal patterns of the time series data, to which exponential smoothing is used to make forecasts with the complex numbers. The reason for processing complex numbers in this way is solely to link the actual value to the forecast error. This gives the model extra insights into possible errors, rather than only modeling the observed value of the time series data and breaking it down into different components.

The authors also used the M3 data as a benchmark dataset. They were aware of the three popular statistical models: State space exponential smoothing, ARIMA and the Theta method. They had shown that CES can outperform those three models separately in terms of median values of root mean squared scaled error (RMSSE) used in the M5-Competition (Makridakis et al., 2022b), mean absolute scaled error (MASE;

Hyndman & Koehler, 2006), and mean scaled interval score (MSIS), but not the mean values of MASE and MSIS. Additionally, Svetunkov included his own work as another scope of benchmark, namely his contributions to SCUM (Petropoulos & Svetunkov, 2020), and found that while comparing between SCUM without CES and SCUM with CES that it originally did, the latter performed better in terms of median values of the same 3 metrics but bested by the prior only in average MSIS.

In general, including CES in ensembles can lead to improvements in accuracy. Compared to other exponential smoothing methods, CES can better capture long-term relationships as argued by the authors, as well as capture non-linear trends without the need to creating separate components to capture level and trend.

2.2 DeepAR

In "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks," Salinas et al. (2020) explained the mathematical foundation behind the DeepAR method. It is a supervised learning algorithm that leverages recurring neural networks (RNN) and the concept of autoregression. In this model, the researchers included two sets of data: first a sequential time series and second a set of covariates that should be constant for the given time series. Covariates can be time dependent and item dependent. Item dependent covariates bring in the related data concept in the model. For example, a shoe is a part of the shoe category and should show similar time series characteristics. So, shoe category could be related data for shoe time series forecasting. However, the data used in the mentioned DeepAR study is univariate, which aligns with this paper's objective.

2.3 Performance Capacities of Statistical and Deep Learning Models

Statistical techniques, such as auto-regressive (AR), auto-regressive moving average (ARMA), auto-regressive integrated moving average (ARIMA), and vector auto-regressive (VAR) have been dominating the landscape of forecasting. However, these statistical models assume stationarity and linearity in forecasting, simplifying the understanding but overlooking the complexity built into the information. Machine learning brings solutions such as support vector machine (SVM), random forest (RF), and gradient boosting to deal with non-linearity in the data which can be used as additional tools to time series forecasting. Recently, with larger amounts of data and better computing power availability, deep learning models are producing higher forecasting performance using their complex techniques.

Khalil et al. (2022) compared statistical, machine learning, and deep learning forecasting results on energy consumption based on multiple attributes such as modeling functionality, characteristics, location, occupancy, system, and control operational data. The study noted that the mean absolute percentage error (MAPE) is a widely used comparison metric for performance, to which Makridakis et al. Had also used in their study. Khalil et al. emphasized the use of temporal granularity, encompassing horizon, which consists of short, medium, and long-term forecasting, and interval, which consists of daily, monthly, and weekly. This research team will treat intervals as time series frequencies.

Suradhaniwar et al. (2021) investigated the performance between a one-step-ahead forecast and a multi-step-ahead forecast across statistical, machine learning, and deep learning models. They discovered that the performance of a one-step-ahead forecast is superior to a multi-step ahead forecast, given that the one-step delivers the next observation before making another prediction. Their research achieved to present which family of models generates better forecasts for one-step-ahead, particularly in multi-step-ahead scenarios. Their research concludes that ARIMA and support vector regression (SVR) perform well in both scenarios, but MLP and RNN falter badly in multi-step scenarios. Additionally, the paper illustrates that root mean square error (RMSE) and sMAPE produce similar results for both one-step-ahead and multi-step-ahead forecasting horizons.

Makridakis et al. (2018) conducted research using the M3-Competition monthly data to understand the efficacy of recent machine learning and deep learning models vis-à-vis traditional statistical models. In this ongoing research, the researchers expanded their scope by adding two additional deep learning models, namely the Multi-layer Perceptron (MLP) and Bayesian Neural Network (BNN), as well as five machine learning models. Two accuracy measures, sMAPE and MASE, were introduced to compare performances among all three of the model types mentioned. Research concluded that simple statistical models outperformed machine learning and deep learning models. Interestingly, their conclusion held steady even when the computation complexity and the good of fit were considered.

Two years later, the deep learning approach by Makridakis et al. (2022) was quite monumental. Aside from deep learning models, they also created an ensemble consisting of the best performing statistical models via the median, called Ensemble-S, as one of the competing methods in their study. Forecasts from multiple deep learning models, precisely 50 models, for each of the four deep learning model types DeepAR, Feed-Forward, Transformer and WaveNet were ensembled via the median to enhance the forecasting accuracy. In other words, there is one final forecasted single value (or set of values), depending on the forecasted horizon of either 1, or 2 and greater, for each of the 3,003 time series that represents the calculated median forecasted single value (or set of values) of the 50 different deep learning models, identified by their own initial conditions, hyperparameters or random seeds that are trained and validated on the same time series. Additionally, they considered an even larger ensemble of 200 deep learning models, called Ensemble-DL, done via the median the same way that the 50 deep learning models mentioned earlier had contributed from each of the four deep learning model types. This ensemble ended up majorly outperforming all the models from all three method categories, only losing to Ensemble-S in the shorter forecasted horizons by around 2% for horizons lower than 5 and 8.1% for horizon 1, which consists of the combination via the median forecasts produced by the statistical models ARIMA and State Space Exponential Smoothing (ETS). This approach by Makridakis et al. (2022) is intricate and well thought out but ultimately carries the burden of being too computationally intensive. Ensemble-DL was reported to take 345 hours (about 2 weeks) to conduct, not to mention the time it took to optimize the numerous amounts of hyperparameters required, as well as the time required to conduct each of the four deep learning model types. Despite being groundbreaking at the time, an approach like this is impractical and unwieldy to the average forecasting practitioner. In contrast, Ensemble-S was mentioned to resemble the ensemble that won 6th place out of 61 competing methods in the M4-Competition in 2018. Compared to its top five competitors, this Simple Combination of Univariate Models (SCUM; Petropoulos & Svetunkov, 2020), which is a combination via the median of the forecasts of four models, namely ARIMA, ETS, Complex Exponential Smoothing (CES) and Theta. Petropoulos & Svetunkov claims that this ensemble would be the simplest forecasting approach in the M4-Competition to tackle its 100,000 time series realizations, supplying a balance between algorithm runtime and forecasting accuracy.

Therefore, this research aims to build and analyze the performance of a combination of ARIMA, ETS, CES, Theta, Multi-layer Perceptron (MLP), LSTM and DeepAR, via an appropriate ensemble scheme, on the 1,428 monthly time series from the M3-Competition monthly data, where it will be evaluated against the comparison metric sMAPE.

2.4 Ensemble Schemes

In 2019, hourly air quality data from an Italian city had been analyzed using ARIMA, SVM and ANN models (Li & Ngan, 2019). The time series contained 93,578 observations that represent the hourly average carbon monoxide levels from multiple sensors in the vicinity. Using a rolling-window on a horizon of length 1, forecasts were generated for each window. Taking various combinations of the three proposed models, a weight was applied to each model based on the proportion of absolute distance of forecasts from the actual of a model over the sum of absolute distance from all models for each ensemble. In general, the results across ensembles show an increase in mean absolute error (MAE) and MAPE metrics when SVM and ANN weights are heavily favored and ARIMA is penalized.

Another weighted-ensembling approach between statistical, machine learning and deep learning models involved 34,616 observations of 15-minute power consumption frequencies in the ICC building in Hong Kong (Fan et al., 2014). Out of the eight different models used in the ensembling process, there are three of interest: ARIMA, multiple linear regression (MLR), and MLP. Using a combination of forecasts generated from multiple univariate and multivariate models, ensembles were produced using the genetic algorithm as a weighting scheme. The efficacy of the forecasts was evaluated using RMSE, MAE and MAPE. These results showed an increase in performance amongst the ensembled models that penalized ARIMA, MLR and MLP.

There are common themes among the research conducted by both Li et al. and Fan et al. Both studies utilize a single realization over 30,000 instances, perform the one-step ahead forecasting and use a weighting scheme to construct ensembles that penalized statistical models. While these papers provide some evidence against the use of statistical models in the ensembling process, they do not provide a baseline of univariate ensembles based on the mean, median and mode operators.

These measures of central tendency are widely used in various ensembling methods, yet there is scant literature on their effectiveness in ensembling statistical and deep learning models. Kourentzes et al. partially addresses the issue by comparing the mean, median and mode on an ensemble of neural networks (Kourentzes et al., 2014). The research examined 3000 realizations from the Federal Reserve and applied the mean, median and mode operators to the forecasts. The modes of the forecasts are calculated through kernel density estimation, while the means and medians of the forecasts are calculated in the standard fashion. The means and medians were affected by outliers, albeit the median less so and the mode showed the most resistance. In other words, the use of both the median and mode operators for ensembles is suggested.

These simple ensemble operators will act as a base before exploring other weightings schemes. An ensemble of nine different statistical and deep learning models was proposed on the NN5 Competition dataset (Andrawis et al., 2011). There were 115 realizations of daily ATM withdrawals in the UK over a period of 2 years that were forecasted for 56 days (about 2 months) into the future. The researchers produced 140 models from nine different types such as ARMA, MLP, HW and simple moving average. To select nine models from those that are used in the study, the researchers simply took an average of the models that individually produced the lowest sMAPE. The researchers concluded that the ensembled model is the best performing model in terms of sMAPE.

This research will extend statistical and deep learning ensembles from single realizations to 1,428 monthly realizations in the M3-Competition data as well as explore and compare combinations via the mean and median.

3 Methods

The M3-Competition (Makridakis & Hibon, 2000) was the most eventful "M" competition organized by Spyros Makridakis, since the data from this competition is still being used to test the performance of new forecasting models. The data source for this research is a subset of the 3,003 time series data from this competition, particularly the 1428 monthly time series. The subset contains six domains, namely the *Micro*, *Macro*, *Finance*, *Industry*, *Demography*, and *Other* domains.

Table 1. General Statistics of M3 Monthly Data.

	Micro	Macro	Finance	Industry	Demography	Other
Number of Series	474	312	145	334	111	52
Max Length	126	144	144	144	138	120
Min Length	68	66	68	96	71	71

To prepare the data for modeling, the general statistics of the M3-Competition monthly data are calculated and recorded in Table 1. Observations were drawn from market driven indicators, though it is not clear which indicators were used for each domain to construct the data. Indices of these domains do not move in tandem, even though these data are from related fields. This research team aims to pay close attention to the behavior of the individual domains, and in that pursuit, the mean and standard deviation of the *Micro* and *Macro* domains are calculated and plotted in Fig. 5 and Fig. 6. Since the lengths of series are asymmetric, the last 60 months (about 5 years) observations for each realization are considered for these plots. To declutter the plots, only 10 random samples are culled for presentation along with the realization mean and the standard deviation, though the realization means, and the standard deviation are computed from all realizations from respective domains. Fig. 5 and Fig. 6 represent *Micro* and *Macro* domains, respectively.

Realization is a particular instance of time series and in this case, multiple realizations are provided, which form an ensemble of realizations. Mean is the expected value of all possible realizations in each time, and similarly, standard deviation tracks the variation of all possible realizations in each time. For time series to be stationary, both the realization means, and the standard deviation must be constant for each unit in time. In Fig. 5, the realization means, and the standard deviation are observed to change with time but oscillate around the same levels without displaying any upward or downward trend. On the other hand, in Fig 6, most series are exhibiting an upward trend, indicating that many in the *Macro* Domain are non-stationarity, though the realization standard deviation tends to display the stationary behavior given that it moves around the same level.

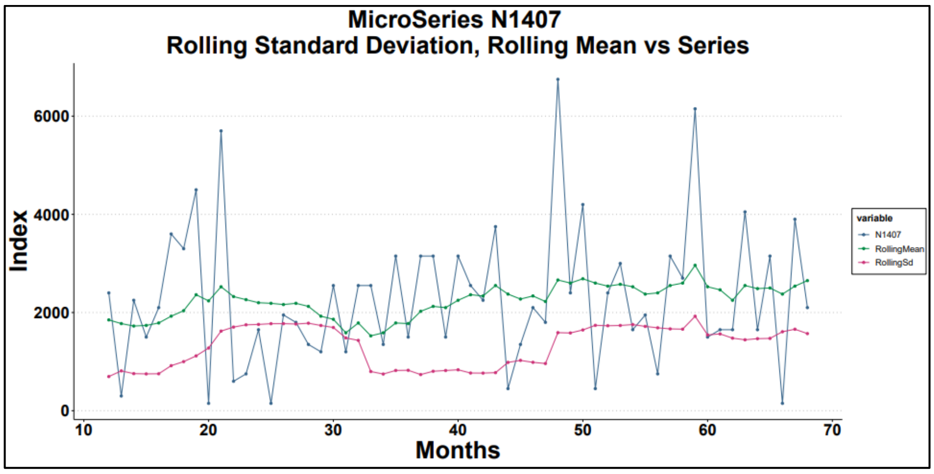


Figure 5. ***Micro* Domain.** Realization means and standard deviation with 10 sample realizations.

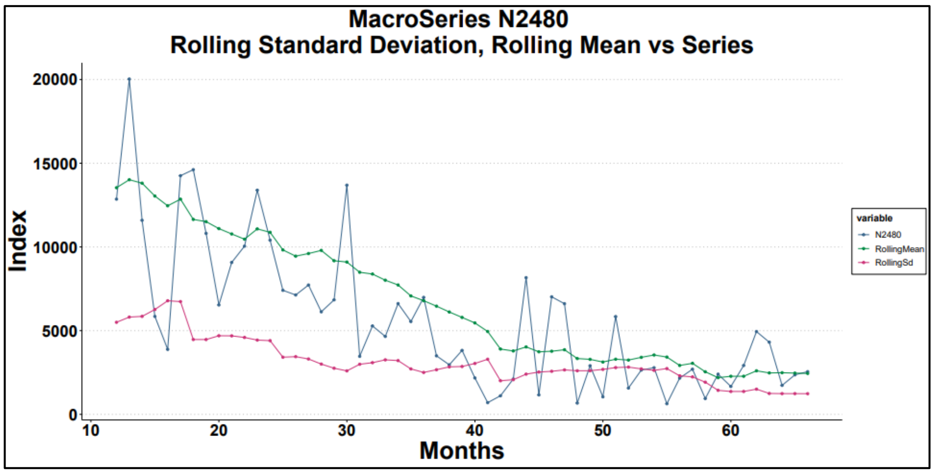


Figure 6. ***Macro* Domain.** Realization means and standard deviation with 10 sample realizations.

At individual time series, only one realization exists, and that realization constitutes the entire time series. However, the base assumptions of constant mean and standard deviation still hold. Mahdy et al. (2020) leveraged the rolling mean and the rolling standard deviation to calculate and test the stationarity of individual series. On a similar note, one sample realization each from *Micro* (N1407) and *Macro* (N2480) is picked up, and the rolling mean and the standard deviation for each series are calculated using a rolling window of 12, given the interval of the dataset is monthly and the financial data such as *Micro* and *Macro* is assumed to have a yearly cycle. Fig 7 and Fig 8 illustrate the rolling mean, the rolling standard deviation, and the actual series for *Micro* (N1407) and *Macro* (N2480) respectively. *Macro* series (N2480) seems to be non-stationary as the rolling mean and the rolling standard deviation display downward trend. From an overall and an individual sample time series perspective, the *Micro* domain is more stable as both tests illustrate stationarity. On the other hand, the *Macro* domain hints at non-stationarity as both tests point to non-stationarity. Both displays suggest that the mean and the standard change with time.

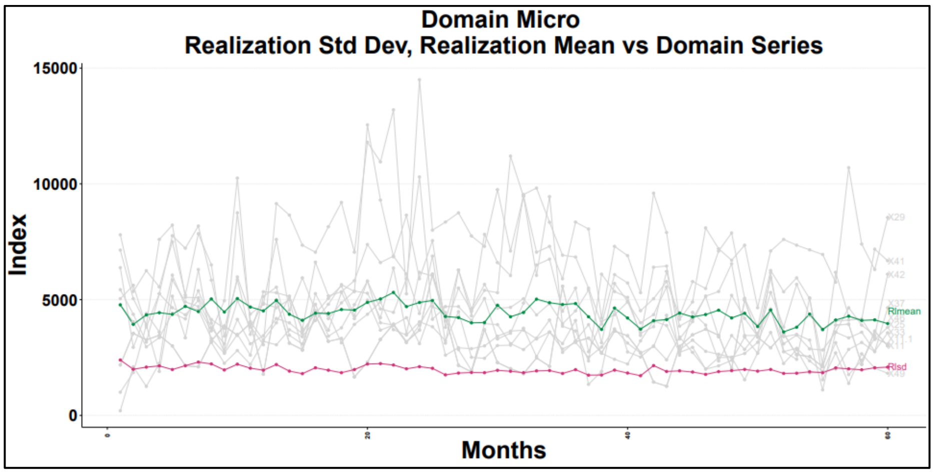


Figure 7. ***Micro*** **Time Series N1407.** Rolling mean, Rolling standard deviation and actual *Micro* series (N1407).

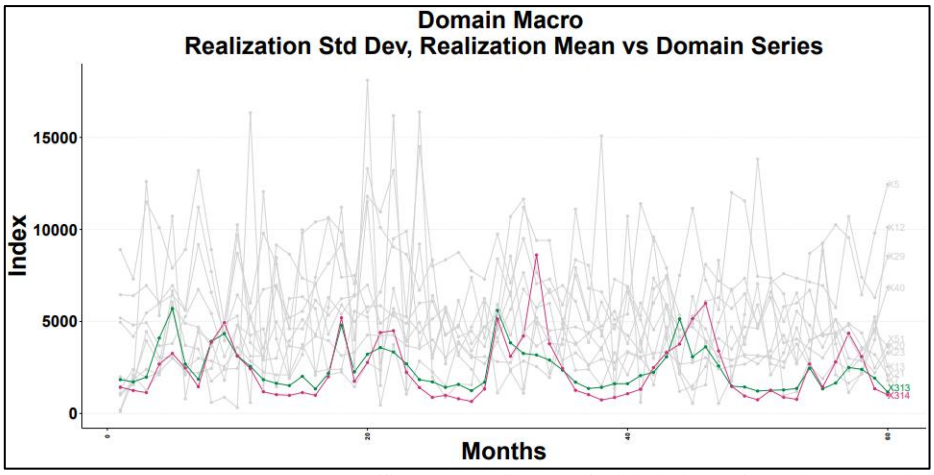


Figure 8. ***Macro*** **Time Series N2480.** Rolling mean, rolling standard deviation and actual *Macro* series (N2480).

To build contrast, the yearly dataset can be used to compare to the monthly dataset. The yearly dataset has 645 series across six domains. Table 2 provides a general statistic of the yearly dataset. The *Demographic* domain has a higher proportion of series than that of monthly data. The series maximum length and minimum length of *Micro* and *Macro* data are almost equal, highlighting the quality of data. The maximum length and minimum length are identical across *Finance*, *Industry*, *Demography*, and *Other* domains, meaning data collection happened simultaneously for these domains.

Table 2. General Statistics of M3 Yearly Data.

	Micro	Macro	Finance	Industry	Demography	Other
Number of Series	146	83	58	102	245	11
Max Length	20	22	47	47	47	47
Min Length	20	23	20	21	20	20

For modeling, multiple forecasting models will be trained and tested separately, before combining them together via an appropriate ensemble scheme like taking the median of the forecasts.

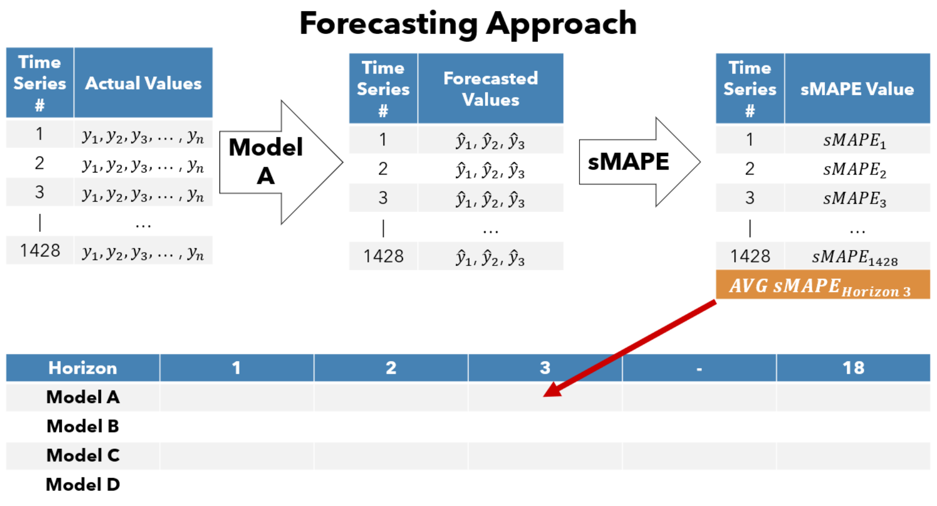


Figure 9. The Forecasting Approach Workflow.

According to the forecasting approach depicted in Fig. 9, actual values of 1428 time series are represented in y_1, y_2, y_3, to y_n. In this sample workflow, Model A, which could be any method such as ARIMA, MLP etc., generates forecast of horizon 3, which 3 units of time into the future using multi-step forecasting. The forecasted values for horizon 3 are represented by $\hat{y}_1$, $\hat{y}_2$, $\hat{y}_3$. The sMAPE for all 1428 series are calculated using these forecasted values. The average of these sMAPE values of the 1428 series is reported in the cell for Horizon 3 of Model A. The same process is repeated for Model A for each horizon, starting from 2 through 18.

ARIMA is a classical linear time series method developed by George Box and Gwilym Jenkins in the 1970's. These models are used to model out data that exhibits serial correlation among its points (Woodward et al., 2022). The AR or Autoregressive is comparable to linear regression except for the explanatory variables are the response variables lagged in time. The MA or moving average is representative of modelling a moment of time as a combination of error terms. Any moving average part of a model can be written in terms of strictly an AR portion, however adding an MA component simplifies the model. The final portion is the integrated or I term. This in essence represents the number of times a realization has been differenced with itself. In addition, these models can handle seasonality; This can be performed by differencing the data at the frequency it exists. There are 3 assumptions that must be met for an ARIMA model to be valid: all windows of the realization have a constant mean, all windows of the realization have a constant variance, and the correlation between a point and its lagged variables is constant at all points in the realization.

A preprocessing step of removing trend was performed before evaluating phis (ϕ) and thetas (θ) for the ARIMA models. Two separate ARIMA models were built using two different methodologies for trend identification. The first trend method was born by Nelson et al. (1999) who suggested that deseasonalization on non-seasonal data bore little consequence while non deseasonalization on seasonal data can be detrimental to performance. This line of reasoning was extended to trend removal by simply differencing all the 1,428 series. Suggested by Woodward et al. (2022), the second trend identification, Cochrane-Orcutt method is statistical in nature. If identified, trend was removed with a simple differencing of the data. In the tables below, 'ARIMA Differenced' and 'ARIMA Cochrane' will represent these methodologies respectively.

ETS is a state space model that selects from 24 exponential smoothing models that exhibit two components: a trend component that divides into 4 categories no trend, additive trend, multiplicative trend and damped trend, as well as a seasonal component that divides into 3 categories no seasonality, additive seasonality and multiplicative seasonality. The first set of three models with no trend component consist of SES (denoted as NN), exponential smoothing with additive seasonality (NA) and exponential smoothing with multiplicative seasonality (NM). The second set of three models with an additive trend component consists of Holt's linear method (AN), additive Holt-Winters' method (AA) and multiplicative Holt-Winters' method (AM). The third set of three models with a multiplicative trend component consist of multiplicative trend no seasonality (MN), multiplicative trend additive seasonality (MA), multiplicative trend multiplicative seasonality (MM). And finally, the fourth set of models with damped trend, created to tackle the fact that the trend estimated from historical data may not continue in the same way in the future and continuing to use the final estimate for the growth rate at the end of the historical data would lead to unrealistic forecasts, the trend is gradually dampened as the length of the forecast horizon increases, consist of damped trend no seasonality (DN), damped trend additive seasonality (DA) and damped trend multiplicative seasonality (DM). According to Hyndman et al. (2002), a time series can possess both the additive error assumption and the multiplicative error assumption. Therefore, a total of 24 models are considered and evaluated via AIC before being chosen as the best exponential smoothing model.

CES is a 2022 extension to exponential smoothing that also uses both trend and seasonality components to make forecasts; however, it uses a complex number to represent the smoothed value of the time series data. The real part of the complex number represents the smoothed value of the time series, while the imaginary part represents the rate of change of the time series. The smoothing factor is applied to both the real and imaginary parts, and the resulting complex number is used to forecast future values. This would translate to two smoothing factors that CES uses. The first smoothing factor is alpha, which determines the weight given to the most recent observation in the time series data. The second smoothing factor in CES is beta, which determines the weight given to the trend component of the time series data. CES first separates the input time series into two components: the level component which represents the average value of the series, and the trend component which represents the direction and speed at which the series is moving over time. Next, CES applies exponential smoothing to each component separately using the estimated alpha and beta to calculate the weight given to each observation; the level component is updated using alpha, and the trend component is updated using both the alpha and beta. Finally, CES combines the resulting level and trend components to generate the forecasts, which is done by adding the level component to the trend component multiplied by the forecast horizon.

The Theta method is a very strong candidate among the statistical models that performed exceptionally well in the M3 competition, particularly for the monthly series and the *Micro* domain (Assimakopoulos & Nikolopoulos, 2000). The authors state that a time series can be decomposed into two Theta lines, a trend component and a seasonal component, which identify as $\theta = 0$ and $\theta = 2$ respectively. With that said, five stages are created to work with these two components, consisting of validating the need for and performing the classical multiplicative decomposition method (via a comparison between the *t*-statistic value, or standard normal distribution *z*-score, for a 90% confidence interval and the lag one year autocorrelation function value, that is 12 observations for monthly timeseries and 4 observations for quarterly timeseries) into two components: a trend component, which is extrapolated as a normal linear regression line, and a seasonal component, which is extrapolated using SES. The next step is to combine the two components using equal weights, in other words, multiply both the first and second component's forecasts by .5 then summing them up to get the overall forecasts for the time series. Then the final step is to multiply the overall forecasts by the respective seasonal indices found in the validation stage, only if decomposition was necessary earlier.

Multi-layered Perceptron (MLP) is a general-purpose deep learning model that introduces non-linearity to the data relationships and can model complex trend and seasonality within a realization (Andrawis et al., 2011). The general structure of an MLP consists of k number of layers with n number of hidden nodes. Each layer consists of a unique activation function, such as ReLu that introduces the nonlinearity to the current weighted sum. The layers may have any number of nodes and consist of many types of activation functions. These layers are called the 'hidden' layers. The final layer, however, consists of a single activation function appropriate for the task at hand. A single pass through all the data is called an epoch, and a model may be trained on as many epochs as desired. After each pass, the results from the final activation layer are evaluated and a correction to the model weights are found via a gradient-descent type method; This process is called backpropagation.

The MLP algorithm from the R package *nnfor* has been used to generate the forecasts. All preprocessing for trend/seasonality and lags has been automatically selected by the MLP function in this package. The model for each series for each horizon is determined by taking the median value of 10 individual models built at said series/horizon. There are cases where the auto selection for trend/seasonality as well as the number of models built had to be adjusted to produce results. These cases have been noted in Appendix 1.

Long short-term memory (LSTM) is a gradient based deep learning method proposed by Hochreiter et al. (1997) that is constructed using Recurrent Neural

Network (RNN) and contains memory units for closer and farther observations from the present time. The LSTM is designed to mitigate 'forgetting' distant observations from the current time point as is typical with a traditional RNN. This is accomplished by having a long-term memory unit that updates its value more slowly when given new points in the sequence. This allows for more influence from points in time from farther lags in the past. LSTM utilizes sigmoid and tanh activation functions to update its long and short-term memory units. More specifically, the tanh function determines the degree to which the current value is retained, while the sigmoid functions determine the proportion of that degree. The result is a model that may be used on sequential data to forecast future values.

The LSTM algorithm from TensorFlow for Python was used to calculate the forecasts on the monthly time series. The full list of hyperparameters may be seen in section 3 of the Appendix. The modelling of LSTM consisted of running 3 repetitions for each series on each horizon and taking the median value of each. This process was performed to offset situations where the initial starting weights caused the model to not update quickly enough and therefore produce poor results.

DeepAR, as explained by Salinas et al. (2020), uses RNN with a component of autoregression, which utilizes covariates and brings in the notion of time series having related covariates will behave similarly. Covariates are computed using appropriate lags so that the correlation between data points becomes time independent. In RNN, the sequential data are fed conditioned to the previous sequence, but in DeepAR, the sequential data are supplied conditioned to the previous sequence and the covariate. Amazon implemented DeepAR in 2017 for commercial use. Other deep learning frameworks, such as MXNET, GluonTS and TensorFlow, offer packages for DeepAR implementation. This study will use GluonTS package (deepar.estimator) built on Apache MXNET framework for DeepAR implementation. The deepar.estimator are trained using multiple hyperparameters (Mahdy et al., 2020) such as num_layers, which represents the number of RNN layers, hidden_size, which denotes the number of RNN cells for each layer, dropout_rate, learning rate, batch_size, epoch and so forth. The loss function used for optimization of the process is negativeloglikelyhood, which converts product of likelihood function to a sum of log functions for mathematical convenience. Thereby, minimizing the negativeloglikelyhood function is equivalent to maximizing loglikelihood function. After training the estimator, the GluonTS leverages a probabilistic approach to generate several prediction paths for forecasting. The forecasting values of multiple predictions are captured in the list and the median point out of forecast sample values is considered for our forecasting. Monthly data are univariate time series and mostly clean, besides 29 rows of 71 observations from the *Other* domain miss the starting year and the starting month values, which are needed for building the time series model in GluonTS. In deepar.estimator, the starting time with month, day and year format are the required values. Similar series are observed, and the corresponding starting month and year are picked to populate these values for 29 rows of the *Other* domain. The High-Performance Computing (HPC) environment (M3) of SMU is used for a high computing intensive workload of the DeepAR processing. With 128 CPUs and 500 GB memory, 128 parallel processing are initiated, and forecasting for all 1428 rows for a single horizon takes 3 hours on average.

4 Results

The research has focused on the statistical models ARIMA, HW Additive and Multiplicative, Theta, CES, and ES and the deep learning models MLP, LSTM and DeepAR. To avoid inconsistencies in forecasting across models, the common forecast approach is adopted and delineated in the method section. A static script is coded in R to calculate the sMAPE values for all models. The generated model forecasts are loaded into respective folders. These forecasts are read and the sMAPE values are computed for each horizon, ranging from 2 to 18, and shown in Table 3 and Fig.10. The lowest sMAPE values for each horizon have been highlighted in gold.

Two different types of ARIMA models are generated; one where all series are differenced to remove trend (All Differenced) and the other for Cochrane-Orcutt which finds evidence of trend before differencing. Both variations produced consistent results with low variability. However, All Differenced performed better than Cochrane-Orcutt because the sMAPE of Cochrane-Orcutt stayed consistently below that of All Differenced, and it is the best performing individual model. On the other hand, the HW multiplicative model fails to generate lower sMAPE throughout prediction horizons and produces an inconsistent prediction with a wide range of sMAPE. The MLP model, like ARIMA, exhibits a narrower range of sMAPE values across different horizons, despite having higher overall sMAPEs. The CES model beats the ARIMA models in the short range, but in the mid and long range, the ARIMA takes over the CES. HW Multiplicative and Additive have demonstrated similar patterns in forecasting, but the additive dominates over the multiplicative in all ranges. The ARIMA models outperformed both the Additive and Multiplicative HW models, except for one case. At horizon 12, the HW models sMAPE values have a noticeable dip. The research team ranks individual models in the following order: ARIMA All Differenced, ARIMA Cochrane-Orcutt, HW Additive and HW Multiplicative, Theta, CES, ES, MLP, DeepAR, and LSTM.

Table 3: sMAPES of Horizons 2 to 18 or Individual Models. The values from this table have been plotted in Fig.10 to better showcase the change in sMAPE values for each model as the horizon increases.

Horizon	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ARIMA Differenced	10.53	11.39	11.58	12.08	12.31	12.34	12.51	12.65	12.79	12.86	13.19	13.4	13.42	13.6	13.68	14.15	14.53
ARIMA Cochrane	11.31	12.47	12.74	13.33	13.13	13.12	13.36	13.45	13.64	13.58	13.75	13.83	13.84	14.21	14.24	14.65	15.01
HW Add	12.65	14.28	15.63	16.43	17.24	16.86	17.08	16.77	16.4	16.17	14.5	16.86	18.06	19.28	20.56	21.27	22.02
HW Multi	11.56	13.16	14.42	15.17	16.11	15.84	16.14	16.03	15.65	15.38	13.65	16.16	17.36	18.7	20.04	20.76	21.57
Theta	10.28	11.92	13.15	13.66	14.17	13.94	13.96	13.74	13.32	12.77	11.36	13.05	14.09	15.06	16.14	16.61	17.05
CES	8.92	10.01	10.67	11.3	12.04	12.35	12.78	13.4	13.76	14.06	14.28	14.58	14.84	15.23	15.6	15.97	16.35
ES	10.55	12.27	13.57	14.08	14.62	14.46	14.55	14.32	13.92	13.34	11.83	13.71	14.83	15.92	17.08	17.6	18.04
MLP	12.32	13.38	13.98	14.79	14.59	14.92	15.46	15.15	15.22	15.6	15.64	15.88	15.95	15.93	16.33	17.25	18.12
DeepAR	13.87	14.39	14.26	15.16	14.93	15.45	15.63	15.59	15.67	16.3	16.84	17.48	18.16	18.03	18.57	19.01	19.1
LSTM	26.83	26.15	26.39	25.81	26.35	25.45	27.29	27.49	28.02	26.63	28.17	31.85	27.33	55.97	30.74	38.35	31.31

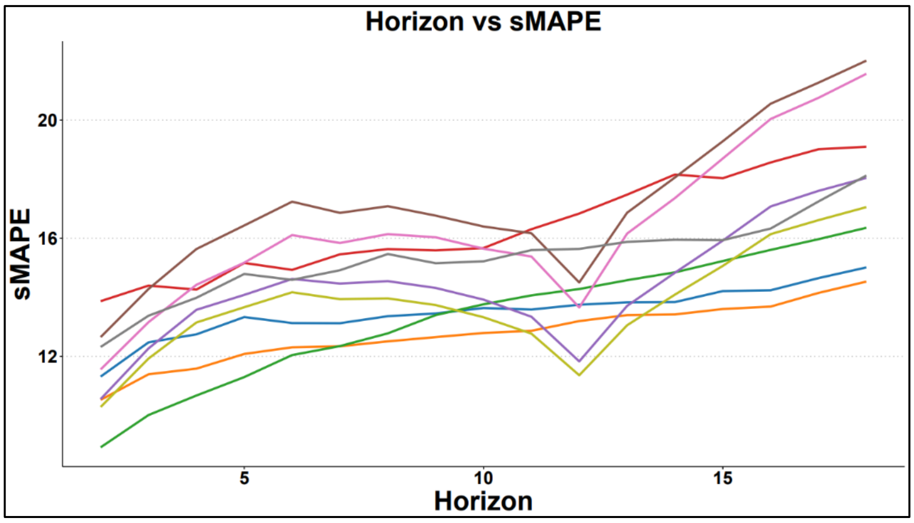


Figure 10. Performance of Individual Models Visualized.

Table 4. Individual Models Key. This table acts as a key to identify the implemented individual models and the corresponding color code.

Key
🟦 **ARIMA (Cochrane-Orcutt)**
🟧 **ARIMA (All Differenced)**
🩷 **Holt-Winters (Multiplicative)**
🟫 **Holt-Winters (Additive)**
🟪 **ES**
🟩 **CES**
🟨 **Theta**
⬜ **MLP**
🟥 **DeepAR**

The requirements for an ensemble to be built were at least one statistical and one deep earning method be used; however, this is only a minimum requirement with many ensembles containing more models. Using a randomized number and selection of models, 200 total ensembles were created using either the mean or median operator. In determining which ensembles performed the best, a simple sum of each ensemble mean sMAPES for horizon 2-18 was calculated. The best ensembles are determined to be the ones with the lowest sum of mean sMAPEs, thereby giving equal weight to all the horizons. The results may be seen in Table 5 and Fig. 11.

Table 5. sMAPES of Horizons 2-18 for Top Ensembles. The values from this table have been plotted in Fig. 11 to better showcase the change in sMAPE values for each model as the horizon increases.

Horizon	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Ensemble 1	8.93	9.78	10.06	10.65	10.99	10.89	11.05	11.17	11.15	11.11	10.86	11.39	11.64	11.95	12.25	12.45	12.83
Ensemble 2	9.41	10.2	10.56	11.23	11.46	11.36	11.52	11.53	11.37	11.46	11.28	11.73	11.98	12.14	12.51	12.75	13.18
Ensemble 3	9.16	10.05	10.35	10.95	11.35	11.17	11.36	11.39	11.25	11.15	10.66	11.33	11.67	12.02	12.36	12.56	13.01
Ensemble 4	9.49	10.32	10.76	11.42	11.67	11.36	11.54	11.45	11.21	11.16	10.64	11.37	11.66	11.99	12.4	12.66	13.16

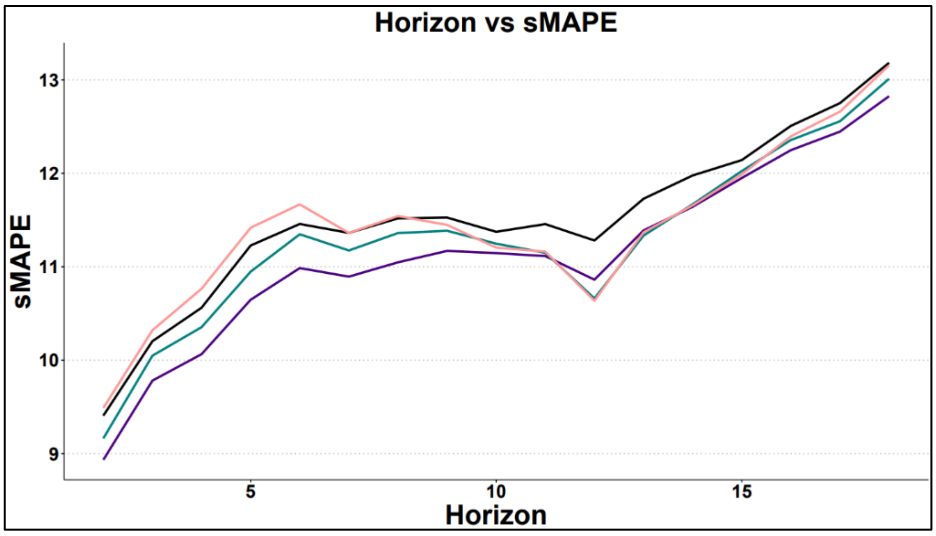


Figure 11. Performance of Top Ensembles Visualized.

Table 6. Ensembles Key. This table acts as a key to identify which ensembling scheme and models are used in the top 4 ensembles.

Ensemble Name	Models Used	Scheme
🟪 **Ensemble 1**	DeepAR + ARIMA (All Differenced) + CES + Theta	Median
⬛ **Ensemble 2**	DeepAR + ARIMA (All Differenced) + CES + Theta	Mean
🟩 **Ensemble 3**	DeepAR + ARIMA (All Differenced) + ARIMA (Cochrane-Orcutt) + ETS + CES	Median
🟥 **Ensemble 4**	DeepAR + ARIMA (All Differenced) + ARIMA (Cochrane-Orcutt) + ETS + CES	Mean

All four top performing ensemble methods have DeepAR, ARIMA (All Differenced) and CES in common. Ensembles 1 and 2 share the same individual models while only differing in ensemble schemes. This is the same pattern for ensembles 3 and 4.

Ensemble 1 exhibited the lowest sMAPES across a majority of the horizons and therefore is deemed the top performing ensemble overall. This ensemble is comprised of 3 individual models (CES, ARIMA All Differenced, and Theta) that produced at least 3 of the lowest sMAPES on any horizon. DeepAR as an individual model did not perform well across all the horizons, however it does appear in Ensemble 1.

Table 7. sMAPES for Horizons 2-18 for Top Individual Models and Ensembles. The values from this table have been plotted in Fig. 12 for purposes of clarity, namely the change in sMAPE values for each model as the horizon increases.

Horizon	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ARIMA Differenced	10.53	11.39	11.58	12.08	12.31	12.34	12.51	12.65	12.79	12.86	13.19	13.4	13.42	13.6	13.68	14.15	14.53
Theta	10.28	11.92	13.15	13.66	14.17	13.94	13.96	13.74	13.32	12.77	11.36	13.05	14.09	15.06	16.14	16.61	17.05
CES	$\boxed{8.92}$	10.01	10.67	11.3	12.04	12.35	12.78	13.4	13.76	14.06	14.28	14.58	14.84	15.23	15.6	15.97	16.35
Ensemble 1	8.93	$\boxed{9.78}$	$\boxed{10.06}$	$\boxed{10.65}$	$\boxed{10.99}$	$\boxed{10.89}$	$\boxed{11.05}$	$\boxed{11.17}$	$\boxed{11.15}$	$\boxed{11.11}$	10.86	11.39	$\boxed{11.64}$	$\boxed{11.95}$	$\boxed{12.25}$	$\boxed{12.45}$	$\boxed{12.83}$
Ensemble 2	9.41	10.2	10.56	11.23	11.46	11.36	11.52	11.53	11.37	11.46	11.28	11.73	11.98	12.14	12.51	12.75	13.18
Ensemble 3	9.16	10.05	10.35	10.95	11.35	11.17	11.36	11.39	11.25	11.15	10.66	$\boxed{11.33}$	11.67	12.02	12.36	12.56	13.01
Ensemble 4	9.49	10.32	10.76	11.42	11.67	11.36	11.54	11.45	11.21	11.16	$\boxed{10.64}$	11.37	11.66	11.99	12.4	12.66	13.16

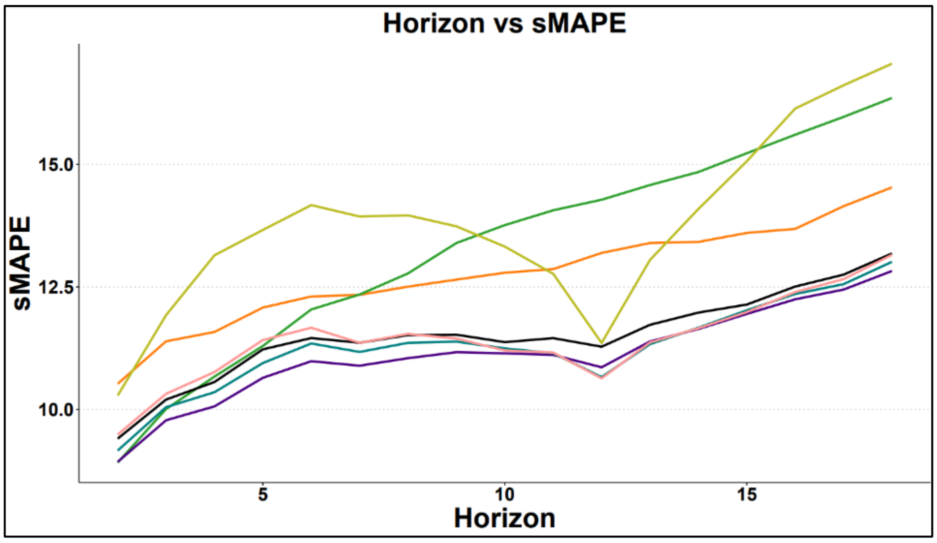


Figure 12. Performance of Top Individual Models and Ensembles Visualized.

Table 8. Top Individual Models and Ensemble Key. This table acts as a key to identify which ensembling scheme and models are used in the top 4 ensembles.

Key	Models Used	Scheme
🟧 **ARIMA (All Differenced)**		
🟩 **CES**		
🟨 **Theta**		
🟪 **Ensemble 1**	DeepAR + ARIMA (All Differenced) + CES + Theta	Median
⬛ **Ensemble 2**	DeepAR + ARIMA (All Differenced) + CES + Theta	Mean
🟦 **Ensemble 3**	DeepAR + ARIMA (All Differenced) + ARIMA (Cochrane Orcutt) + ETS + CES	Median
🟥 **Ensemble 4**	DeepAR + ARIMA (All Differenced) + ARIMA (Cochrane Orcutt) + ETS + CES	Mean

Of the top ensembles, the median ensemble scheme generally produces a lower sMAPE than the mean ensemble scheme. This implies that some realizations are producing high sMAPE values making the distribution skewed. However, without comparison to other mean/median ensemble scheme pairs, it is difficult to say whether this pattern extends beyond these examples.

The individual top ensembles have been identified and now may be compared to the top individual models. Table 7 and Fig. 12 below combine the top performers from these categories for comparison.

In Fig. 12, the performance of the top four ensemble methods and top 3 individual models re plotted together to compare performances of ensemble models with those of individual models. It is evident that the ensemble methods are clear winners as ensemble methods dominated across all horizons besides horizon 2 where CES method beats all ensemble methods. The result exhibits that individual models may perform better in some horizons, but ensemble models prove to be superior in the overall performance.

A question remains on the extent to which domain influences each model's performance and how the ensembling may balance out poor individual domain performance. Two models, CES and DeepAR from the top ensemble have had their mean sMAPES across the horizons delineated by domain for comparison below in Fig. 13 and 14. ARIMA All Differenced and Theta delineated by domain may be seen in section 2 of the Appendix.

In Fig. 13, the performance of CES method is broken by domains: *Demography*, *Finance*, *Industry*, *Micro*, *Macro* and *Other*. The worst performer is the *Other* domain and the second to the worst is the *Micro* domain, whereas the top performer is the Macro domain. And in Fig. 14, the performance of DeepAR method is broken by domains. Here, the worst performer is the *Micro* domain and the third to the worst is the *Other* domain; however, the top performer is the *Macro* domain. So, it is evident from this result that the performance of separate domains varies with different models.

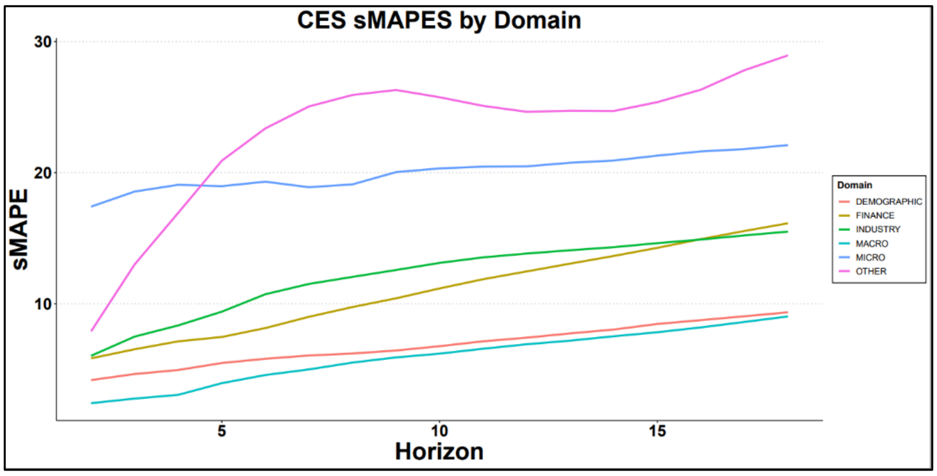


Figure 13. CES sMAPE Performance by Domains.

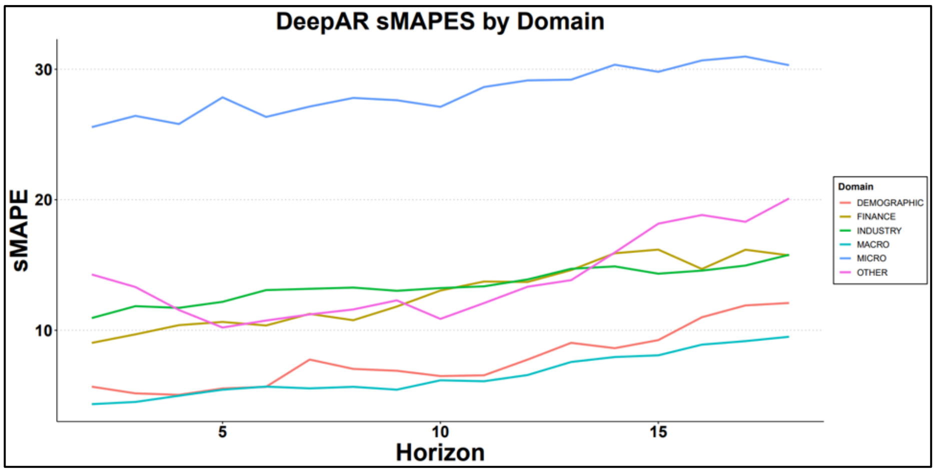


Figure 14. DeepAR sMAPE Performance by Domains.

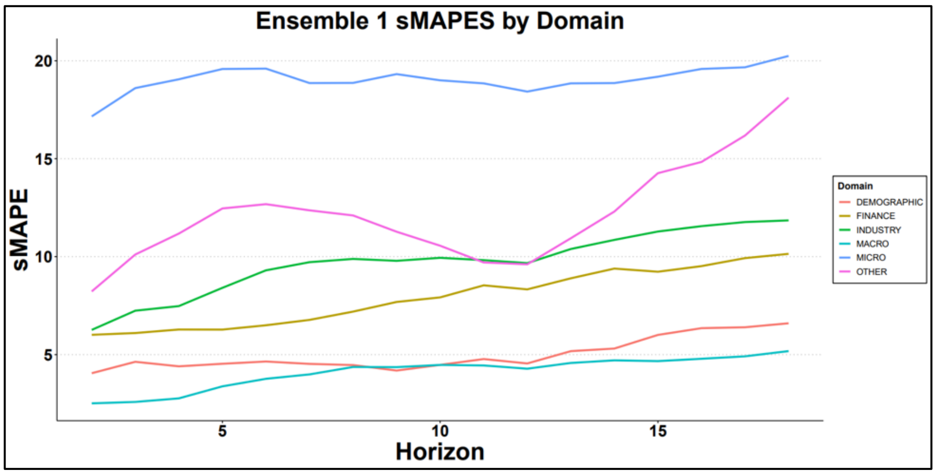


Figure 15. Ensemble 1 sMAPE Performance by Domains

In Fig. 15, the performance of the ensemble 1, the best ensemble model, is broken down by domains. In this case, the worst performer is the *Micro* domain and the second to the worst performer is the *Other* domain. The *Other* domain performed the worst in the CES method and the third to the worst in the DeepAR method. But these models are ensembled, the performance of the *Other* domain has moved to the second to the worst, meaning that the ensemble methods neutralize the bias of the separate models, in this case DeepAR and CES, and produce a balanced result. It also tames down the variability and generates a stable model. In other words, the ensemble generalizes the performance of individual models.

In addition to model performance, the computational runtime is also an important consideration. All models have their own data processing, model training and testing, and prediction. In addition, different models are trained on different computer engines as shown in Table 9.

Table 9: Models, Time Taken, and System Used.

Method	Hours	System Specifications
HW	0.00027	AMD Ryzen™ 7 5800X3D CPU @ 3.4GHz 64GB RAM
ETS	68	AMD Ryzen™ 7 5800X3D CPU @ 3.4GHz 64GB RAM
CES	1	AMD Ryzen™ 7 5800X3D CPU @ 3.4GHz 64GB RAM
Theta	0.04	AMD Ryzen™ 7 5800X3D CPU @ 3.4GHz 64GB RAM
ARIMA	0.33	Intel® Xeon® W-1290P CPU @ 3.7GHz 32GB RAM
MLP	13	Intel® Xeon® W-1290P CPU @ 3.7GHz 32GB RAM
LSTM	25	NVIDIA® Quadro RTX™ 4000 GPU @ 1GHz 8GB GPU Memory
DeepAR	54	AMD EPYC™ 7763 Server @ 2.45GHz 1.5GB Server Memory (Single Node, 128 CPUs, and 500GB Memory)

5 Discussion

The monthly M3 time series demonstrated trends, which statistical models tackled by detrending a particular series as appropriate, and some cases, the entire series while some deep earning models overlook trend and seasonality as these models inherently handle it. However, the ensembling of statistical and deep learning models generates superior results. Multi-step forecasting is used, and the shorter horizons performed better results than the longer ones. From the domain perspective, the *Micro* domain was the worst performer, because many *Micro* indices lumped to form the *Micro* dataset, whereas the *Macro* domain was the best performer because the series follow a predictable pattern.

The research team expected that the ensemble models would perform better than separate models because the ensemble method brings in diversified information and counters the bias of separate models to generate a stable and better performance. Therefore, the variation in ensemble methods is consistently lower than that of any individual model.

This paper furthers the work done by Makridakis and explores other combinations of ensemble to corroborate the notion that in time series the ensemble method produces superior performance. Secondly, the team investigated the domain level performance to see how different models performed in different domains. Interestingly, the simpler model handles irregular time series such as the *Micro* domain better than the sophisticated models such as DeepAR. In addition, the ensemble method counters the bias of individual domain performance.

It should be noted that since the ensembles are forced to have a minimum of one statistical and one deep learning, this research does not provide any ensembles that are only statistical or only deep learning. The top performing ensembles all only include one deep learning method yet are composed of multiple statistical models. The statistical models in the top ensembles are all included in the top performers for individual models. It is therefore difficult to say whether the improvement in ensembling is derived from deep learning models without more comparisons.

This research invokes the ethical dilemma that is the effectiveness of training complex, computationally heavy deep learning models because it consumes so much power without yielding a meaningful better result and outweighs the benefits of undertaking such time and energy consuming endeavor. Therefore, data scientists should ponder to some extent before training a complex model.

The research realized that the performance of different varies based on models used. Therefore, a good opportunity to improve the overall results by training different domains such as *Micro*, *Macro*, etc... separately and finally combining them.

Future research should also include other deep learning, statistical and machine learning models for ensembling. Recognizing that there is no one size fits all solutions, the exploration of ensemble combinations can offer insights into improving the generalizability of forecasting accuracy.

The models selected play a prominent role in the results, yet different ensembling schemes that involve weighting methods based on past performance is another area that warrants attention. There is potential to optimize the ensemble's predictive capabilities if able to determine prior performance through methods such as a Rolling Window to adapt the weighting scheme.

Furthermore, examining the impact of different series lengths on the ensembles sMAPE values across the horizons is another area of future research. This would shed light on how the length of the time series influences the forecasting performance and effectiveness of the ensemble approaches to specific domains.

6 Conclusion

The research findings indicate that ensembling forecasts with multiple statistical and deep learning models using median or mean schemes leads to better performance in variability and sMAPE values across all horizons. Although individual models may occasionally produce lower sMAPE values at some horizons, there is greater confidence in the consistency of the results when models are ensembled. When it comes to the individual models, statistical models demonstrated superiority, but the model performance is derivative of the data. It is observed that the performance of separate domains varies considerably by model. So, instead of encompassing the entire dataset in a single frame, the dataset should be broken down to domain levels where deep learning models may bring forth a better result in some domains.

Acknowledgments.

The work of this research would not have been possible without the continuing guidance of Dr. Bivin Sadler and Dr. Jacquie Cheun-Jenson, as well as the entire faculty of the Data Science Graduate Program at Southern Methodist University. Gratitude and thanks are extended to these individuals and the program.

References

1. Makridakis, S., Spiliotis, E., Assimakopoulos, V., Semenoglou , A. A., Mulder, G., & Nikolopoulos, K. (2022a). Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward. *Journal of the Operational Research Society*, 1-20. 10.1080/01605682.2022.2118629
2. Woodward, W. A., Sadler, B. P., & Robertson, S. (2022). Time Series for Data Science: Analysis and Forecasting (1st ed.). *Chapman and Hall/CRC.* 10.1201/9781003089070
3. Khalil, M., McGough, A. S., Pourmirza, Z., Pazhoohesh, M., & Walker, S. (2022). Machine Learning, Deep Learning and Statistical Analysis for forecasting building energy consumption — A systematic review. *Engineering Applications of Artificial Intelligence, 115*, 105287. 10.1016/j.engappai.2022.105287
4. Suradhaniwar, S., Kar, S., Durbha, S. S., Jagarlapudi, A. (2021). Time Series Forecasting of Univariate Agrometeorological Data: A Comparative Performance Evaluation via One-Step and Multi-Step Ahead Forecasting Strategies. *Sensors, 21*(7) 2430. 10.3390/s21072430.
5. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS One, 13*(3), e0194889. 10.1371/journal.pone.0194889
6. Petropoulos, F., & Svetunkov, I. (2020). A simple combination of univariate models. *International Journal of Forecasting, 36*(1), 110-115. 10.1016/j.ijforecast.2019.01.006.
7. Gardner, E. S. (2006). Exponential smoothing: The state of the art—Part II. *International Journal of Forecasting*, *22*(4), 637-666. 10.1016/j.ijforecast.2006.03.005
8. Gardner, E. S., & McKenzie, E. (1985). Forecasting trends in time series. *Management Science, 31*(10), 1237-1246. 10.1287/mnsc.31.10.1237
9. Chatfield, C. (1978). The Holt-Winters Forecasting Procedure. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 27*(3), 267-279. 10.2307/2347162
10. Hyndman, R. J., Koehler, A. B., Snyder, R., & Grose, S. D. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting, 18*(3), 439-454. 10.1016/s0169-2070(01)00110-8
11. Sutcliffe, G. (n.d.). *State Spaces.* https://www.cs.miami.edu/home/geoff/Courses/COMP6210-10M/Content/StateSpaceSearch.shtml
12. Svetunkov, I., Kourentzes, N., & Ord, J. K. (2022). Complex exponential smoothing. *Naval Research Logistics (NRL), 69*(8), 1108-1123. 10.1002/nav.22074
13. Assimakopoulos, V., & Nikolopoulos, K. (2000). The theta Model: a decomposition approach to forecasting. *International Journal of Forecasting, 16*(4), 521-530. 10.1016/s0169-2070(00)00066-2
14. Makridakis, S., & Hibon, M. (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting, 16*(4), 451-476. 10.1016/s0169-2070(00)00057-1
15. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022b). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting, 38*(4), 1346-1364. 10.1016/j.ijforecast.2021.11.013

16. Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting, 36*(3), 1181-1191. 10.1016/j.ijforecast.2019.07.001
17. Li, L., & Ngan, C. K. (2019). A Weight-adjusting Approach on an Ensemble of Classifiers for Time Series Forecasting. *ICISDM 2019: Proceedings of the 2019 3rd International Conference on Information System and Data Mining*, 65-69. 10.1145/3325917.3325920
18. Fan, C., Xiao, F., & Wang, S. (2014). Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques. *Applied Energy, 127*, 1-10. 10.1016/j.apenergy.2014.04.016
19. Kourentzes, N., Barrow, D. K., & Crone, S. F. (2014). Neural network ensemble operators for time series forecasting. *Expert Systems with Applications, 41*(9), 4235-4244. 10.1016/j.eswa.2013.12.011
20. Andrawis, R. R., Atiya, A. F., & El-Shishiny, H. (2011). Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition. *International Journal of Forecasting, 27*(3), 672-688. 10.1016/j.ijforecast.2010.09.005
21. Mahdy, B., Abbas, H., Hassanein, H. S., Noureldin, A., & Abou-zeid, H. (2020). A Clustering-Driven Approach to Predict the Traffic Load of Mobile Networks for the Analysis of Base Stations Deployment. *Journal of Sensor and Actuator Networks, 9*(4), 53. 10.3390/jsan9040053
22. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (November 15, 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
23. Nelson, M., Hill, T., Remus, W., & O'Connor, M. (1999). Time series forecasting using neural networks: should the data be deseasonalized first? Journal of Forecasting, 18(5), 359–367

Appendix

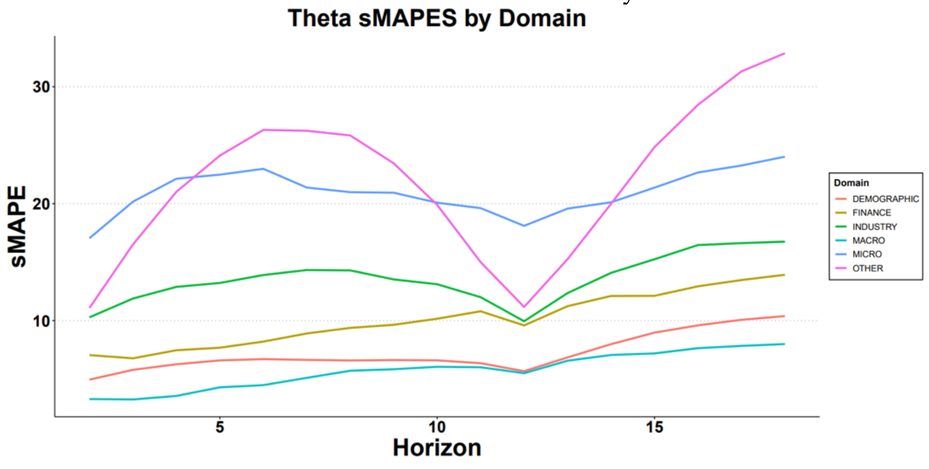
1.

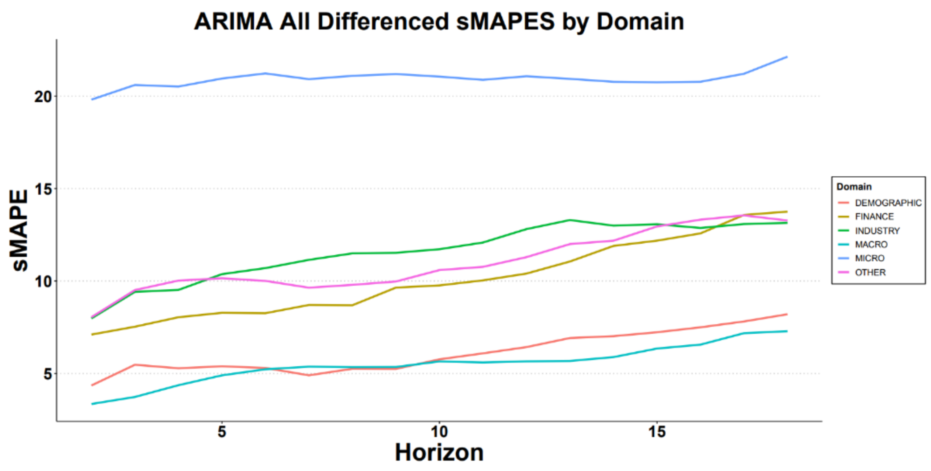
These adjustments were made to fine-tune the MLP models and facilitate the model building process.

- For series 689:
    - For sub-series 4 and 7, the auto seasonality selector was turned off and the number of repetitions was set to 1.
    - For sub-series 8, both the auto seasonality selector and lag selector were turned off, and the reps were set to 1.
    - The remaining sub-series of 689 had the reps set to 1.
- For series 1201:
    - For sub-series 7, 13, 14, and 16, the reps were set to 1, and both the seasonality selector and lag selector were turned off.
- For series 369:
    - For the sub-series with a forecast horizon of 15, the reps were set to 1.

2.

Theta and ARIMA All Differenced sMAPES delineated by domain.





3.

LSTM Hyper Parameters

Ensemble Name	Value
N_steps	25
Cells	512
Learning Rate	0.35
Epochs	1000
Patience	10
Min_Delta	0.01
Dropout Rate	0.071
Batch Size	32

4.

Symmetric Mean Absolute Percentage Error (**SMAPE**)

$$sMAPE = 100 \cdot \frac{1}{N} \cdot \sum_{t=1}^{N} \frac{|F_t - A_t|}{\left(|A_t| + |F_t|\right)/2}$$

$F_t = Forecasts$

$A_t = Actuals$

$t = fitted\ point$

$N = number\ of\ fitted\ points$

Lower is better

5.

Code Base Repository

https://github.com/HybridELM/HELM