SMU Data Science Review

Volume 7 | Number 2 Article 2

2023

Traditional vs Machine Learning Approaches: A Comparison of Time Series Modeling Methods

Miguel E. Bonilla Jr.
Southern Methodist University, mbonilla@smu.edu

Jason McDonald
Southern Methodist University, mcdonaldj@smu.edu

Tamas Toth
Southern Methodist University, ttoth@smu.edu

Bivin Sadler
Southern Methodist University, bsadler@smu.edu

Traditional vs Machine Learning Approaches: A Comparison of Time Series Modeling Methods

Tamas Toth, Jason McDonald, Miguel Bonilla, Bivin Sadler Ph.D.
Master of Science in Data Science
Southern Methodist University
Dallas, Texas USA
{ttoth, mcdonaldj, mbonilla, bsadler}@smu.edu

Abstract. In recent years, various new Machine Learning and Deep Learning algorithms have been introduced, claiming to offer better performance than traditional statistical approaches when forecasting time series. Studies seeking evidence to support the usage of ML/DL over statistical approaches have been limited to comparing the forecasting performance of univariate, linear time series data. This research compares the performance of traditional statistical-based and ML/DL methods for forecasting multivariate and nonlinear time series.

1 Introduction

Time series modeling is important in many applications in science, including medicine, economics, and engineering. In the real world, most time series data are nonlinear, and their accurate forecasting is highly valuable for a wide range of studies such as GDP, EEG, ECG, epidemiological data, etc.

Over the last twenty years, there has been an explosion in new time series methods that leverage the power of Machine Learning computing for forecasting time series (Crone et al., 2011). Since these novel modeling techniques use more complex algorithms with higher computational requirements, they are regularly assumed to be superior to traditional methods. Frequently, these novel techniques tend to be introduced as having better forecasting performance than traditional approaches. However, in many cases, this conclusion is based on results produced with a small number of samples and datasets, focusing predominantly on short-term forecasting horizons and without a benchmark comparison against statistical methods (Makridakis et al., 2018). In the past, several studies have found that these newer Machine Learning models have worse forecasting performance than traditional statistical approaches (Makridakis et al., 2018), casting doubt on the assumption that these newer methods are better than established, traditional techniques.

Additionally, previous open forecasting competitions, such as the M3 and NN3 competitions, found no conclusive evidence that Machine Learning models outperformed statistical approaches in time series forecasting (Crone et al., 2011). Moreover, these events were often dominated by statistical approaches, with Machine Learning methods showing comparatively poor performance.

In recent years, advances in Machine Learning (ML) have significantly improved the performance of Deep Learning (DL) techniques compared to traditional methods.

These newer and more sophisticated ML models have been found to produce more accurate predictions than traditional statistical models (Makridakis et al., 2022). However, these studies have focused on comparing traditional and Machine Learning methods using univariate, linear data; more extensive analysis is needed to compare forecasting performance for multivariate, nonlinear datasets (Hewamalage et al., 2021).

The lack of multivariate time series forecasting in past research studies and forecasting competitions limits the generalization of their results. In some industry cases, the additional information that is contained in variables other than the main time series can increase the accuracy of a model's forecasts. These additional variables, which can include hierarchical relationships, categorical features, and even additional time series, can sometimes add more value to the model's forecast than the time series data (Fry & Brundage, 2020). Further studies are needed to properly compare the performance of statistical approaches to ML models when forecasting multivariate time series.

This study focuses on comparing traditional statistical time series methods to Machine Learning techniques using multivariate, linear, and nonlinear data. The study aims to find if there is a statistically significant difference in forecasting performance between these two approaches, which justifies the use of more complex and computationally expensive modeling methods.

As noted by Hewamalage et al. (2021), the fact that these studies have focused on comparing the forecasting performances of different methods using solely univariate time series is problematic. This represents a gap in knowledge since multivariate time series forecasting might be currently being done without empirical evidence that compares the performance of these different methods, which would provide support for using one method over the other depending on the type of time series.

This research aims to provide empirical evidence that supports the use of Machine Learning methods over traditional statistical methods (or vice versa) for forecasting multivariate, linear, and nonlinear time series. The study will focus on comparing statistical methods to ML/DL methods, by creating forecasts over multiple horizons for each model and evaluating their performance based on error metrics associated with the forecasted predictions of each method.

2 Literature Review

The literature review focuses on this study's main areas of concern: Previous research comparing statistical and ML models; multivariate (bivariate) time series; generating nonlinear, bivariate time series; time series models.

2.1 Time Series Models

Although most research and applications focus on implementing linear time series modeling and forecasting, it is often the case that simple linear time series models can leave some of the complexity of the data unexplained (Zivot & Wang, 2006). Handling

higher complexity time series data often requires using methods that can properly and effectively model nonlinear and/or multivariate behaviors.

Some of the most widely implemented nonlinear forecasting models include Threshold AR (TAR), Self-Exciting Threshold AR (SETAR) and Smooth Transition Threshold AR (STAR). SETAR and STAR are variants of the TAR model which was originally introduced by Tong (1978). These are a type of statistical time series models which can be used with nonlinear time series data. TAR and its variants SETAR and STAR are a type of regime-switching model that finds separate AR functions to model nonlinear time series behavior based on different thresholds (Godahewa et al., 2022). For time series modeling, regime-switching is based on the concept that the time series data exhibits different patterns at different time periods, the TAR model thresholds are the values which delimit these separate time series behaviors. Although these models are considered statistical methods, estimating the thresholds is a computationally expensive task, which in the past limited the implementation of these methods in practice.

The TAR model uses the threshold value to divide the time series data into separate piece-wise regimes, each with a different linear structure. Though most conventional TAR models contain a single threshold variable, two or more thresholds can be applied resulting in a larger number of regimes (Chen et al., 2012). The SETAR model is an extension of the TAR model, which uses the lag-dependent variable as the threshold variable (Tong, 1990).

The STAR model implements smooth transition functions allowing regime switching without hard threshold cutoffs, as is the case with TAR models (Zivot & Wang, 2006). Two of the main implementations of STAR are logistic and exponential STAR, which implement logistic and exponential functions respectively as the smooth threshold functions. These functions act as a continuum threshold, where the weight of each regime varies over time.

Another widely used statistical time series forecasting method is the Integrated Nested Laplace Approximation. This is a statistical method for Bayesian inference, which was originally proposed by Rue et al. (2009). The INLA method utilizes a hierarchical structure approach, making it suitable for modeling high-complexity time series such as multivariate and nonlinear data (Ravishanker et al., 2022). This model calculates the Laplace approximations for the posterior distributions with highly efficient computational time as compared to the fully Bayesian inference implementations.

In recent years, Artificial Neural Networks (ANNs) have become a popular tool for forecasting time series, including nonlinear and multivariate data. A main advantage of ANNs for time series forecasting is that they can model any type of data, meaning no prior assumptions of linearity and underlying data relationships are required (Zhang & Qi, 2005).

Additionally, Recurrent Neural Networks have become a particularly popular ANN for competitive time series forecasting based on the results of the M4 competition. The advantage of these networks lies in the feedback loop of the recurrent cells that can identify the time lag and the dependency in a time series realization (Hewamalage et al., 2021). The neural networks contain a recurring feedback, which accounts for the previous step and input data before generating the new output (Makridakis et al., 2018).

The two main types of RNN units for time series modeling are Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

The DeepAR, or Deep Autoregressive model, is an autoregressive, recurrent neural network for forecasting time series, which is trained on all data from a time series by fitting a global model (Salinas et al., 2019). Since this forecasting method learns a global model from all the time series data, it is able to generate forecasts for series with little to no available data based on similarity to other available data. The model is both autoregressive and recurrent, since the observation of the last time step is taken as an input, and each output produced becomes an input for the subsequent step.

Transformers are another type of Machine Learning model which has gained significant attention over the past few years. These models were first implemented in the fields of Natural Language Processing and computer vision, on which they achieved superior performance over established methods (Devlin et al., 2019). The interest generated based on the success of Transformers in NLP and other fields and their suitability for capturing long-range dependencies has led to the proposals of variations of Transformer for a variety of time series applications (Wen et al., 2023). Many variations of Transformers have been applied to time series data with various measures of success, these address challenges like forecasting, anomaly detection, and classification. However, using transformers to model short and long-range time dependencies with seasonality remains a challenge.

Two of the main Transformers variations categories implemented for time series forecasting are module-level and architecture-level variants (Wen et al., 2023). Module-level variants are slight variations of the basic Transformer model with time series learning assumptions. Architecture-level variants implement a different architecture from the basic Transformer model which utilizes different frameworks.

Wen et al. (2023) proposed a basic taxonomy of Transformer based models for time series applications. This taxonomy has two major components that group the time series modeling from the Network Modification and Application Domains perspective. The network modification entity describes the modifications made to the Transformer architecture to address special time series characteristics. The application entity creates distinct application groups to which time series Transformers can be applied to.

2.2 Generating Nonlinear, Bivariate Time Series

Time series data differs from most of the data sets used for traditional Machine or Statistical Learning methods. Time series data is not randomly sampled since the observations are not independent and time series data exhibits a variety of behaviors, such as stationarity, or the lack thereof, periodicity and seasonality. The simplest form of time series data is univariate which only consists of a single dimension that has been measured repeatedly over time. In practice, the interest is to forecast quantities like cost, margin, or volume, etc. that are dependent on multiple other dimensions and the past behaviors of these variables. A linear function of input variables plus a noise component can model linear data. In other words, the problem can be approximated by a straight line or in a multidimensional space with a hyperplane.

Well known parametric statistical methods used to model univariate or multivariate linear time series models are Autoregressive (AR), Moving Average (MA), or the

combination of the two (ARMA), as well as Vector Autoregressive models (VAR) for multivariate models. A slight nonlinearity in the data can invalidate these linear methods and produce a less accurate forecast.

Although there are variety of real-life nonlinear time series datasets, artificially generated (synthetic) time series data offers advantages for research (Nielsen, 2019, chapter 4). The reason synthetic data is attractive is that there is no limitation on volume, given that as much data can be generated as needed; the research team controls the generation of the data, therefore, it can be adjusted to the scope of the study, and it supports hypothesis testing and model performance comparisons. Furthermore, synthetic data allows researchers to eliminate data contamination, which is usually present in real-life data, where an early-stage error can propagate through subsequent stages and hinder objective measures. Synthetic data should be generated so that it is as free of bias as possible.

There are multiple methods available for generating linear univariate time series data, e.g., gen.arma.wge function in the TSWGE R package. To the contrary, there is little research available on generating multivariate nonlinear time series datasets (Kang et al., 2019). A major challenge with generating nonlinear time series data is to develop mathematical or statistical models that can approximate the dynamics that are observed in real-life data, such as economic time series.

Two techniques are prevalent in generating nonlinear multivariate time series data. One approach uses more traditional statistical methods described in NTS: An R Package for Nonlinear Time Series Analysis (X. Liu et al., 2020). The second approach uses Time-series Generative Adversarial Networks introduced by several recent papers (Li et al., 2022; Seyfi et al., 2022; Yoon & Jarrett, n.d.).

The NTS package has been developed to address a gap present for nonlinear time series modeling (X. Liu et al., 2020). Besides modeling and forecasting, the R package implements methods to generate univariate and multivariate nonlinear time series data. The mTAR.sim function is used to generate multivariate SETAR process with two-regimes. The function has several input parameters, enabling the generation of multivariate nonlinear datasets using Vector Autoregressive (VAR) coefficients.

The function outputs a multivariate nonlinear dataset with n number of observations, with the threshold value for the TAR process, and with the difference between an observed value at time 't' and the optimal forecast value based on a prior datapoint at time 't-1' (Innovation) and with the delay for threshold variable.

The NTS package provides a comprehensive set of tools for non-linear time series modeling. The threshold estimation algorithms implemented in NTS utilize recursive least squares and are less resource intensive than the least squared algorithms implemented in the tsDyn package (Narzo et al., 2023, Stigler, 2020).

Time-series Generative Adversarial Network is proposed as a method for "generating realistic time-series data that combines the flexibility of the unsupervised paradigm with the control afforded by supervised training" (Yoon & Jarrett, n.d., p. 1). This TimeGAN model builds on the GAN concept for sequence generation but combines it with the advantages of sequence predictions from the AR models. The main idea for data generation is to use four network components which are responsible for learning to encode the features, generating new time series representations, and iterating across time (Yoon & Jarrett, n.d.). The generator part of the model uses two types of training input, one is a synthetic embedding from the model previous output to its

embedding space. The second input is a time series realization from real-life data. These inputs enable the model to generate realistic time series data with the characteristics of the training dataset which may originate from different domains.

Generating nonlinear, multivariate time series data requires the training dataset to bear the same characteristics. An assessment of training data linearity is recommended before generating synthetic time series data. One of the most popular tests for nonlinearity is the BDS test which "can be used to detect remaining dependence and the presence of an omitted nonlinear structure" (Zivot & Wang, 2006, p. 654). This hypothesis test helps determine if the dataset is linear (Null hypothesis) or nonlinear (Alternative hypothesis).

Nonlinear, multivariate time series data generation is currently in a research state without established industry standards today. However, simulating time series data is an important discipline as it yields high value when it comes to modeling datasets which are difficult or even dangerous to collect (Nielsen, 2019, Chapter 4).

2.3 Multivariate (Bivariate) Time Series

Though much research has been focused on univariate time series, forecasting of multivariate time series is often what is seen in practice with time series such as sales or costs which are influenced by other variables in addition to the past behavior of the dependent variable (Woodward et al., 2022). Using only the time-delay terms, as in univariate forecasting, without incorporating the other explanatory variables can cause difficulties in revealing details such as the nonlinearity in financial time series (Niu et al., 2020).

Even though multivariate time series are commonly found in real world applications, their forecasting is still a challenge, given their nonlinear and non-stationary properties, as well as their spatial-temporal characteristics. The forecasting methods used for multivariate time series can generally be grouped into classical and Machine Learning approaches (Long et al., 2022).

More traditional statistical models for multivariate time series typically rely on a small number of samples (Niu et al., 2020). Deep Learning models require large sample sizes to train successfully. Without a high number of trainable parameters or with small sample sizes, Deep Learning models can be susceptible to overfitting and low generalization to the data (Li et al., 2022).

Hewamalage et al. (2021), posit that many users of the traditional univariate techniques lack the expertise needed to make use of complex models such as recurrent neural networks. This is likely to cause these users to apply easy-to-use-and-deploy models like the ones they use in production on univariate time series.

Y. Liu et al. (2020) concluded that three main challenges need to be addressed to improve the long-term prediction of multivariate time series, “representing and learning (1) the spatial correlations between different attributes at the same time, (2) the spatio-temporal relationships between different attributes at different times, and (3) the temporal relationships between different series” (p. 11). Their paper proposed two dual-stage two-phase attention based recurrent neural network (DSTP-RNN) models which achieved better performance in both short-term and long-term predictions but failed to

statistically improve the short-term predictions for small datasets over current state of the art methods.

Niu et al. (2020) proposed a two-stage feature selection model integrated with a Deep Learning model (consisting of a multivariate LSTM-LSTM-GRU) with error correction that showed advantages over 16 competing models, including traditional and Machine Learning methods. Their research found the performance to be sensitive to the choice of the optimizer and its learning rate.

A common theme found across research of multivariate time series is the lack of real-world data sets for use by researchers. In contrast to other multiple feature domains, such as computer vision and natural language processing, where numerous data sets are found publicly across the internet; multiple time series often involves physical and biological processes, such as those involving human subjects (Li et al., 2022). Regulations and security concerns can prevent data sets such as those involving power systems from being made public. These concerns have led to the creation of large-scale synthetic simulation models which are made available for analysis. Nonetheless, gaps still exist between the simulation models and real-world systems, limiting real-world time series data exploitation for research purposes (Zheng et al., 2022).

2.4 Comparing Statistical and ML Models

With the advancements in Machine Learning over the last several years, and the recent increase in the number of models available for forecasting time series, several studies have aimed to compare the performance of these new methods to those of traditional statistical approaches. However, despite the fact that most real-life time series data is multivariate, these studies have largely focused on comparing forecasting performance for univariate time series datasets.

The M-competitions, and their derivatives, provide researchers an opportunity to assess forecasting performance of different methods available at the time of the event, which serves as empirical evidence of how Machine Learning methods compare to statistical approaches (Crone et al., 2011). The results from competitions before 2020 highly favored statistical approaches over Machine Learning techniques for forecasting the univariate time series datasets that were a part of the competition (Crone et al., 2011; Makridakis et al., 2018).

The NN3 competition from 2006-2007, which had its origins on the M3 competition of the past, was aimed at expanding on the M3 findings with a focus on ML methods such as Neural Networks. The NN3 datasets were a subset of the M3 time series datasets, including series with short and long, seasonal and non-seasonal, and short/medium/long horizons. A study conducted on the results of the NN3 competition was unable to find conclusive evidence that ML approaches could outperform established statistical methods, however, they did not find statistically significant differences between the two approaches, and as such, provided empirical evidence that ML approaches were no longer completely outclassed by statistical methods (Crone et al., 2011).

A study conducted in 2018, which utilized a large subset of the M3 competition time series data, aimed to identify and address concerns with respect to statistical and Machine Learning methods for time series forecasting (Makridakis et al., 2018). A main

concern identified by the study is the lack of empirical evidence which supports the claims that Machine Learning methods outperform traditional statistical methods in forecasting time series. The study compared several of the most popular Machine Learning methods at the time with eight statistical methods and found that Machine Learning methods were largely outperformed by statistical methods based on both MASE and sMAPE metrics, moreover, the six most accurate methods (ETS, ARIMA, Damped, Comb, Theta, and SES) were all statistical in nature (Makridakis et al., 2018).

A 2021 study aimed at assessing the current state of Recurrent Neural Networks (RNN) for time series forecasts found that RNN models are not inherently superior to statistical based approaches, but have clear benefits over statistical methods in forecasting specific types of time series (Hewamalage et al., 2021). Researchers compared the two competing approaches using publicly available datasets from competitions, including the M3, M4, and NN5 competitions. The authors of the study acknowledged that while statistical methos are easier to implement, neural networks are more sophisticated and require additional steps to fit a model, they proposed guidelines and a best practices framework for leveraging these tools efficiently. A key concern raised by the authors, was that the study was focused entirely on univariate time series data, the authors suggested that further studies should be performed aimed at comparing statistical methods with RNNs using multivariate time series data.

The results of the M4 competition in 2020 provided empirical evidence that Deep Learning methods were able to outperform traditional statistical methods in many cases. A study conducted in 2020, following the results of the M4 competition, was aimed at giving an update on the 2018 paper by Makridakis et al. (2018) by comparing new Deep Learning ML models developed using the Gluon Time Series toolkit to the statistical and ML models compared in the 2018 study using all the time series from the M3 competition. The study compared four DL models (DeepAR, Feed-Forward, Transformer, and WaveNet), two statistical models (ARIMA and ETS), as well as the ML models from Makridakis et al. (2018), over short, medium, and long forecasting horizons. The methods used included training 50 different DL models of each DL model type, creating an ensemble for each group, and an overall Ensemble-DL of all 200 models; an Ensemble-S (for ARIMA and ETS) was also formed. The study found Ensemble-DL performed generally better than ARIMA, ETS, and Ensemble-S, even though these statistical approaches usually outperformed individual DL methods; except for short horizon forecasts, for which ARIMA, ETS, and Ensemble-S tended to perform better (Makridakis et al., 2022). The results showed DeepAR to have the best performance of all DL methods, but the Ensemble-DL model tended to perform better overall. Researchers proposed that the better performance of statistical approaches over short horizon forecasts is due to the fact that the errors calculated for fitting these are based on minimizing the one-step forecasts, whereas the DL models look at all the data simultaneously, therefore being better at modeling longer step forecast than statistical models.

The comparative studies and the time series forecasting competitions have limited their scope to univariate linear time series data. This current research aims to find a statistically significant difference in forecasting performance between the traditional statistical-based and ML/DL methods for modeling multivariate, linear and nonlinear time series data.

3 Methods

3.1 Data

Synthetic, bivariate, linear, and nonlinear data of different sizes was generated to compare the different modeling methods.

Synthetic Bivariate Nonlinear Time Series. A custom function was developed to generate synthetic bivariate data based on a nonlinear autoregressive model. This data was inspired by the Threshold AR approach, limited to two regimes, using a mathematical equation.

The function generated random noise, ε, and iteratively generated time series for x and y using a custom set of equations (Eq. 1, Eq. 2). The equations generate bivariate time series data that incorporates nonlinear lags for each variable, nonlinear lag dependencies between variables (Figure 1), and includes an Autoregressive (AR) order of 1. The functions consist of the following parameters: n (length of the time series), α (AR coefficient for x), β (AR coefficient for y), r (threshold value for switching between the two regimes), ε (random noise), σ (standard deviation of the random noise term), γ (scaling factor for both regimes), and f (frequency of the time series).

$$f(y) = \begin{cases} \beta * y_{i-1} - \alpha * x_i + \sum_1^i (1 - \varepsilon_i), & y_{i-1} \le r \\ -\beta * y_{i-1} + \beta * x_{i-1} + \gamma * (1 - \varepsilon_i), & y_{i-1} > r \end{cases} \quad (1)$$

$$f(x) = \begin{cases} 10 + \alpha * x_{i-1} + \left(4 * \varepsilon_i * (1 - \varepsilon_i)\right), & y_{i-1} \le r \\ \alpha * x_{i-1} + \gamma * 4 * \varepsilon_i * (1 - \varepsilon_i), & y_{i-1} > r \end{cases} \quad (2)$$

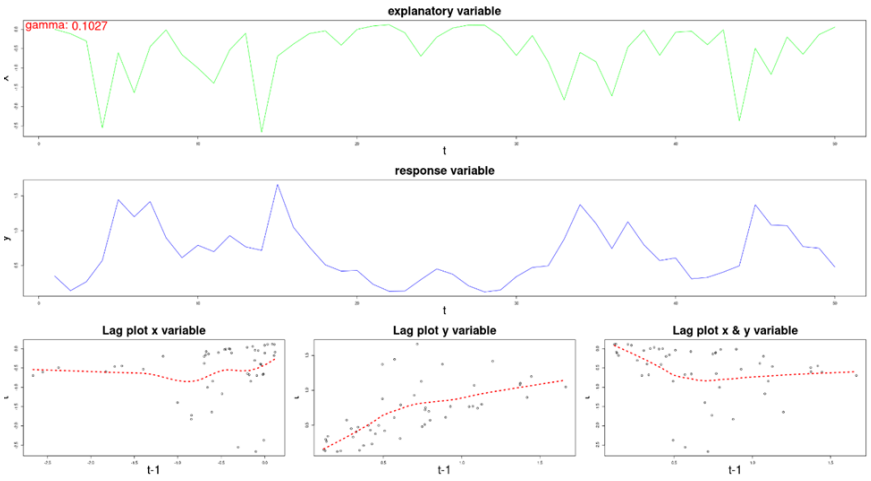

Fig 1. Synthetic bivariate nonlinear time series sample

Time series of different lengths were generated, with a set of ten short-term time series (n=50), ten mid-term time series (n=500), and ten long-term time series (n=2000). For each iteration, a random value of γ was generated within the range of -0.5 to 2. The use of a random scaling factor, γ, introduced variation which resulted in different time series for each iteration.

Synthetic Bivariate Linear Time Series. A custom function was built to generate short-term (n=50), mid-term (n=500) and long-term (n=2000) time series using mathematical equations that incorporate an AR of order 2 lag for both variables (Eq. 3, Eq. 4).

$$x_t = 3 + 0.8 * x_{t-1} - 0.5 * y_{t-1} - 0.2 * x_{t-2} + 0.3 * y_{t-2} \quad (3)$$

$$y_t = 4.5 + 0.4 * x_{t-1} + 0.3 * y_{t-1} - 0.4 * x_{t-2} + 0.1 * y_{t-2} \quad (4)$$

The time series realizations were generated by combining the linear model with scaled normally distributed noise using the simulate function. For each observation length, ten time series were generated using a randomly generated scaling factor within the range of 0.5 to 2, matching the range of the nonlinear time series data.

The generated time series data show a linear relationship between observations for the current time point (t) and observations for time lagging two steps behind (t-2) (Figure 2).

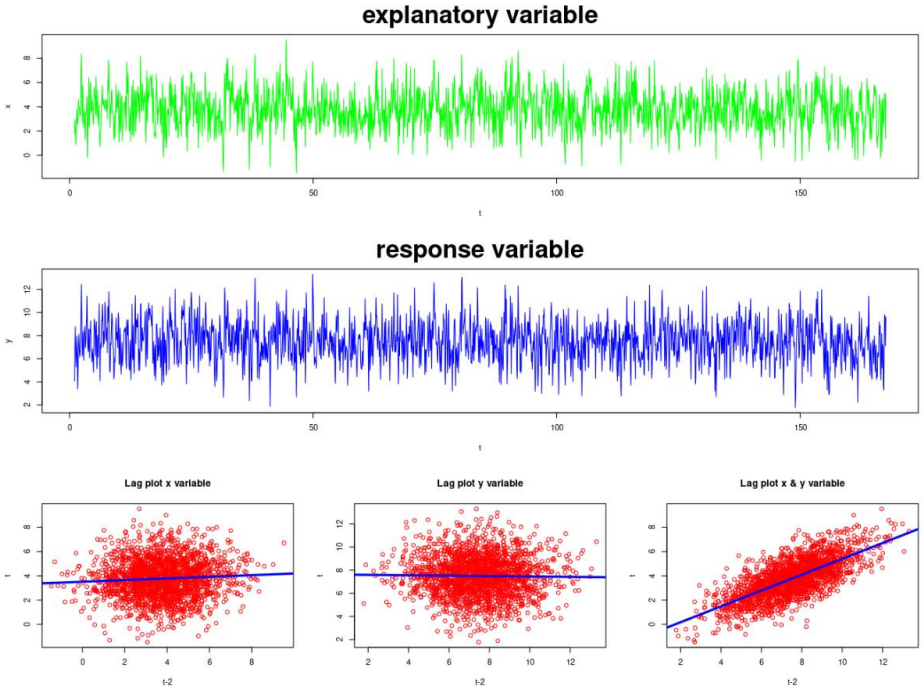

Fig 2. Synthetic bivariate linear data sample

3.2 Methods

To compare the performance of statistical and ML methods, a total of six different models were used. These models included 2 statistical models, Vector Autoregressive (VAR) and Integrated Nested Laplace Approximation (INLA); and 4 ML models, Long Short-Term Memory (LSTM), Deep Vector Autoregressive (DeepVAR), Neural Basis Expansion Analysis Time Series (NBEATS), and Transformer.

The models were trained on a training set consisting of all but the last h observations, where h corresponds to the forecasting horizon for each time series length. The forecasting horizons used correspond to h values of six, eight, and eighteen for the short, medium, and long-term time series lengths, respectively.

VAR. The VAR model is primarily designed for modeling linear time series. However, for this study, VAR was also utilized on the nonlinear time series data to compare its performance against models explicitly designed for nonlinear cases.

The dataset was divided into training and testing set for each time series. With the training set consisting of 80% of the data, a holdout set equal to the forecasting horizon, reserved for final predictions, and a test set for validation during the training.

To optimize the performance of these models, a sliding window method was implemented, tailored to the size of the dataset. For the dataset with 2000 observations, a window size of 100 was chosen. Similarly, a window size of 50 was implemented for the dataset with 500 observations, and a window size of 2 for the smallest dataset with 50 observations.

For each series, the VAR model was fitted using the VARselect function to determine the optimal lag order (p) based on the Akaike Information Criterion (AIC). The VAR model was then trained with the optimal lag order, to predict the forecasting horizon for each time series length.

INLA. The INLA model was trained by constructing a data frame of length equal to the time series length, consisting of a time variable (row index) and variables x and y. To create the training set, the last values of x and y with length equal to the forecasting horizon were removed from the data frame and replaced with NAs.

The missing values of x were then forecasted using a function that calculated x based on time implementing a random walk model 'rw1' with gaussian likelihood. The predictions were then obtained from the summary function and were used to replace the missing values for x on the training data frame. The process was then repeated for y, using time and x as predictor variables to forecast the response variable.

LSTM. To train the LSTM model the input data was scaled within the range of 0 to 1, based on the minimum and maximum values of each variable. To optimize the training, the preprocessed data was subsequently divided into training and test sets, with 80% of the data allocated for training. A custom function was developed to create training and test data using a sliding window approach. This function transformed the time series data into a supervised learning dataset by generating lagged input-output pairs for the LSTM model, considering a specific number of time steps. The sliding window approach involved using a subset of previous time steps as input to predict the subsequent time step.

For each time series realization, the number of time steps and the size of the sliding window were set to predetermined values, determining the historical information incorporated into each prediction.

The LSTM model architecture was constructed using the Keras library, comprising a sequential model with an LSTM layer followed by a dense layer. The LSTM layer was configured with 50 units, and the input shape was defined based on the dimensions of the training data. The model was then trained using the sliding window approach. The loop iterates over the training data, selecting each window of data, and fitting the model with one epoch at a time. The model was compiled using the mean squared error (MSE) as the loss function and the Adam optimizer, which aims to optimize the model's performance during training.

To facilitate comparison between predictions and actual values, a denormalization function was implemented. This function restored the normalized values back to their original range, utilizing the minimum and maximum values from the original dataset.

To evaluate the model's performance, the root mean squared error (RMSE) and the SMAPE (Symmetric Mean Absolute Percentage Error) were computed.

DeepVAR. The DeepVAR model is an implementation of DeepAR, which utilizes a multivariate loss function. The time series data were split into a training set and a test set, with the test set consisting of the last observations, with length equal to the forecasting horizon for each series. The model was then trained on each bivariate train set, applying scaling within the DeepVAREstimator, with batch size of 40, 500 epochs, and a learning rate of 0.001.

The forecasts were calculated by implementing the predict function of GluonTS, taking the median prediction for each time point.

N-BEATS. To forecast multivariate time series using N-BEATS, the DARTS library was used to process the data into a flattened set of inputs to fit into a one-dimensional series that is then structured into a tensor of the appropriate dimensions.

To train the model, the data was first split into a testing set and a validation set on a per-component basis using the scaler function within DARTS. Then, an instance of the NBEATSModel was called passing in several hyperparameters. The hyperparameter, ‘generic architecture’, accepts a Boolean value to indicate whether to use a standard generic implementation of the model or an interpretable architecture that creates only two stacks to represent a seasonality component and a trend component.

For the nonlinear time series, the generic architecture was chosen, whereas the interpretable architecture was chosen for the linear time series. Testing showed that these choices allowed the model to best fit the respective time series.

An early stopping callback was implemented to stop the training when the training loss failed to reach a minimum improvement threshold over any 25 consecutive epochs.

After fitting the model with the scaled data, predictions were made by calling the predict functions of the N-BEATS model to forecast the corresponding horizon of each time series length. These scaled predictions were then passed through an inverse transform function, restoring the predictions to the original scale of the time series.

Transformer. The Transformer model was implemented using an encoder-decoder architecture. The time series data was split into training and testing sets, with a separate

hold out set for final forecasts. The split data was then scaled using the scaler function within DARTS.

The model was trained using an instance of TransformerModel. Several hyperparameters were tuned during the training process, including different activation functions, and different epochs. The training process implemented an early stopping callback to stop the training when the loss reduction over 25 consecutive epochs was below a predetermined threshold.

Predictions were then forecasted using Transformer's predict function. These were then passed through an inverse transform, which restored the data to its original scale.

4 Results

4.1 Time Series Length 50

For the linear time series of length 50, VAR achieved the lowest median forecasting errors, with a median RMSE of 0.745, and a median sMAPE of 12.3 over the ten time series. Transformer and DeepVAR achieved similar median results, with DeepVAR having a median RMSE of 0.935 and median sMAPE of 14.6, and Transformer having errors of 0.908 and 14.8, respectively. The boxplots of the errors for the linear forecasts show that all models appear to have achieved similar results in terms of both sMAPE and RMSE, with the exception of INLA which had significantly higher errors for both (Figure 3). The RMSE and sMAPE for the INLA model were consistently higher than other models over the ten time series, with four time series resulting in sMAPE values of 190 or higher (Figure 4).

In terms of the nonlinear time series forecasts, DeepVAR achieved the lowest median RMSE (4.04) and second lowest sMAPE (94.6), with VAR having the lowest median sMAPE (93.2) and third lowest RMSE (4.56). The INLA predictions resulted in the highest median sMAPE (177), with LSTM having the highest median RMSE (8.49).

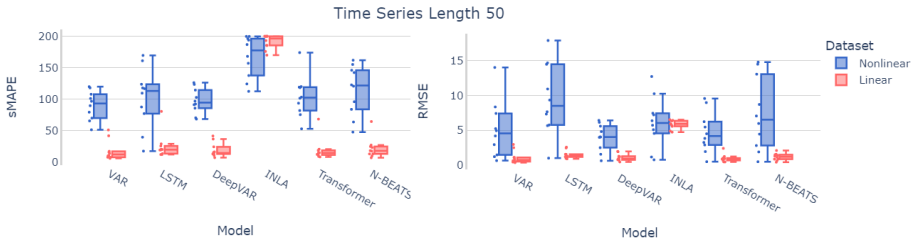

Fig 3. Boxplot distributions for sMAPE (left) and RMSE (right) for time series of length 50

The forecasting errors produced by the best performing statistical model, VAR, showed similar performance to the ML models for both the linear and nonlinear time series(Figure 4).

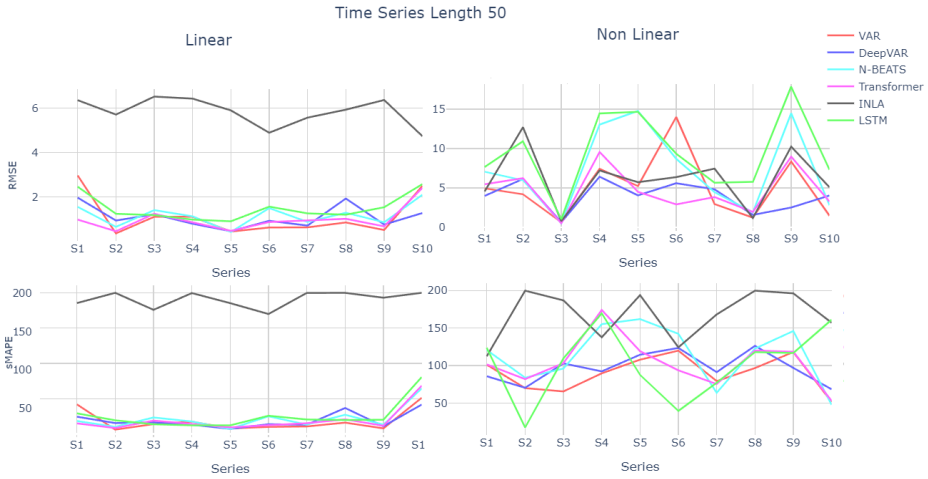

Fig 4. Prediction errors for time series of length 50

4.2 Time Series Length 500

For the ten linear time series of length 500, Transformer achieved the lowest median RMSE (0.988) and sMAPE (18.4) values. The VAR model achieved the second lowest RMSE (1.12), and DeepVAR the second lowest sMAPE (18.6). The errors produced by the INLA predictions had the highest median RMSE (5.90) and sMAPE (194.0).

The nonlinear forecasting errors showed the lowest median RMSE for Transformer (3.97) and lowest median sMAPE for LSTM (42.1), however LSTM showed the largest variation in terms of sMAPE (Figure 5). INLA produced the highest median RMSE and sMAPE with values of 6.50 and 170, respectively.

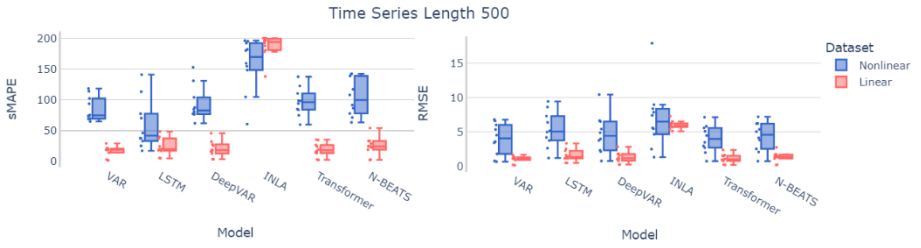

Fig 5. Boxplot distributions for sMAPE (left) and RMSE (right) for time series of length 500

For both linear and nonlinear time series, VAR produced forecasting errors that are comparable to the errors produced by ML forecasts (Figure 6).

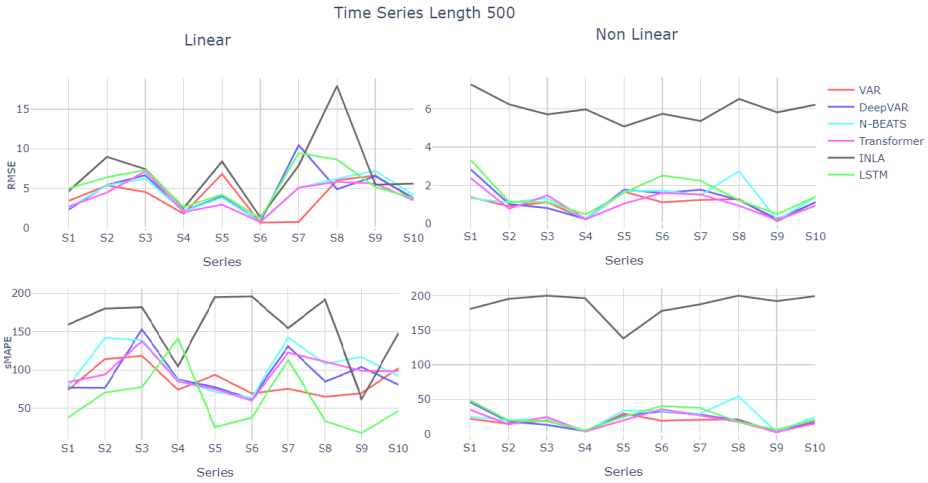

Fig 6. Prediction errors for time series of length 500

4.3 Time Series Length 2000

The forecasts for the linear time series of length 2000, showed similar performance for VAR, DeepVAR and Transformer in terms of sMAPE and RMSE. The VAR model had the lowest median sMAPE at 13.8 and lowest RMSE at 0.817, followed by DeepVAR, with sMAPE of 14.3 and RMSE of 0.881. Apart from INLA, all models showed similar distributions for both error metrics (Figure 7).

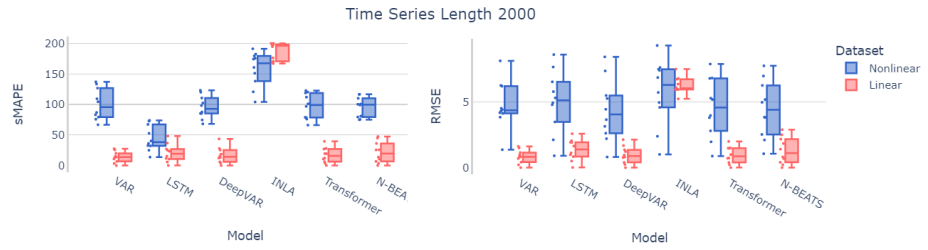

Fig 7. Boxplot distributions for sMAPE (left) and RMSE (right) for time series of length 2000

For the nonlinear time series, LSTM produced the lowest median sMAPE (38.2), and DeepVAR produced the lowest median RMSE (4.04). The model with the worst performance, INLA, resulted in a median RMSE of 6.27 and sMAPE of 167.0.

The VAR model showed similar forecasting errors to the ML models across the 10 linear and nonlinear time series (Figure 8).

Fig 8. Prediction errors for time series of length 2000

5 Discussion

5.1 Implications

The study found no significant difference in performance between the best performing statistical approach, VAR, and the ML methods. The results of the study have significant implications for the usage of advanced ML methods when forecasting time series data and suggest that no assumption of better performance by ML methods should be made when forecasting time series.

Based on the results, and given the lower computational requirements of VAR, a complete approach to modeling data should always include fitting a VAR model for multivariate linear, or nonlinear data.

5.2 Future Work

The data used in the comparison of the models consisted exclusively of bivariate time series data. While the results are intriguing, care should be taken not to draw conclusions that extend to time series with three or more variables. The modeling of the more complex variable interactions of these processes may result in significantly different results, and as such, future research is needed to compare statistical and ML methods in the forecasting of multivariate time series consisting of more variables.

Further validation of the results could be achieved by comparing these models using real bivariate time series data, rather than generated data. Differences in performance might be observed in different fields depending on the types of interactions between the different variables that may be prominent for specific domains. Field specific studies,

comparing statistical and ML approaches using real data for specialized domains might be necessary.

5.3 Ethical Considerations

In this case study, the modeling process did not incorporate personally identifiable data as the dataset utilized was synthetically generated. It is crucial to emphasize that the models examined in this study are intended for application on real-world time series data, where ethics and privacy considerations should be given consideration. Examples of time series datasets that may contain personally identifiable information encompass financial transaction data, including transaction activity; spatiotemporal data that tracks individuals' locations and movements; and healthcare monitoring data collected from wearable devices or medical sensors. When working with such datasets, it is imperative to ensure the implementation of robust ethical safeguards to protect sensitive information and uphold privacy rights.

6 Conclusion

The statistical model, VAR, produced forecasts with errors that were consistent with those of ML models for both linear and nonlinear bivariate time series. While this model was designed to work with multivariate linear time series, the scrolling window approach with a train-test split that was implemented performed as well as the more sophisticated ML models, such as DeepVAR, at forecasting nonlinear bivariate time series of different lengths.

Given the lower computational costs required to train VAR, and the similar forecasting performance, this method appears to be suitable for forecasting bivariate nonlinear time series when training time or processing power are limited.

Acknowledgements Bivin Sadler, Ph.D. – Capstone Advisor

References

Chen, H., Chong, T. T.-L., & Bai, J. (2012). Theory and Applications of TAR Model with Two Threshold Variables. Econometric Reviews, 31(2), 142–170. https://doi.org/10.1080/07474938.2011.607100

Crone, S. F., Hibon, M., & Nikolopoulos, K. (2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. International Journal of Forecasting, 27(3), 635–660. https://doi.org/10.1016/j.ijforecast.2011.04.001

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Fry, C., & Brundage, M. (2020). The M4 forecasting competition – A practitioner's view. International Journal of Forecasting, 36(1), 156–160. https://doi.org/10.1016/j.ijforecast.2019.02.013

Godahewa, R., Webb, G. I., Schmidt, D., & Bergmeir, C. (2022). SETAR-Tree: A Novel and Accurate Tree Algorithm for Global Time Series Forecasting (arXiv:2211.08661). arXiv. https://doi.org/10.48550/arXiv.2211.08661

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. International Journal of Forecasting, 37(1), 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Kang, Y., Hyndman, R. J., & Li, F. (2019, March 7). GRATIS: Generating Time Series with diverse and controllable characteristics. ArXiv.Org. https://doi.org/10.1002/sam.11461

Li, X., Metsis, V., Wang, H., & Ngu, A. H. H. (2022). TTS-GAN: A Transformer-based Time-Series Generative Adversarial Network (arXiv:2202.02691). arXiv. http://arxiv.org/abs/2202.02691

Liu, X., Chen, R., & Tsay, R. (2020). NTS: An R Package for Nonlinear Time Series Analysis. The R Journal, 12(2), 266. https://doi.org/10.32614/RJ-2021-016

Liu, Y., Gong, C., Yang, L., & Chen, Y. (2020). DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction. Expert Systems with Applications, 143, 113082. https://doi.org/10.1016/j.eswa.2019.113082

Long, L., Liu, Q., Peng, H., Wang, J., & Yang, Q. (2022). Multivariate time series forecasting method based on nonlinear spiking neural P systems and non-subsampled shearlet transform. Neural Networks, 152, 300–310. https://doi.org/10.1016/j.neunet.2022.04.030

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLOS ONE, 13(3), e0194889. https://doi.org/10.1371/journal.pone.0194889

Makridakis, S., Spiliotis, E., Assimakopoulos, V., Semenoglou, A.-A., Mulder, G., & Nikolopoulos, K. (2022). Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward. Journal of the Operational Research Society, 1–20. https://doi.org/10.1080/01605682.2022.2118629

Narzo, A. F. D., Aznarte, J. L., & Stigler, M. (2023). tsDyn: Nonlinear Time Series Models with Regime Switching (11.0.4). https://CRAN.R-project.org/package=tsDyn

Nielsen, A. (2019). Practical time series analysis: Prediction with statistics and machine learning (First edition.). O'Reilly Media, Incorporated.

Niu, T., Wang, J., Lu, H., Yang, W., & Du, P. (2020). Developing a deep learning framework with two-stage feature selection for multivariate financial time series forecasting. Expert Systems with Applications, 148, 113237. https://doi.org/10.1016/j.eswa.2020.113237

Ravishanker, N., Raman, B., & Soyer, R. (2022). Dynamic Time Series Models Using R-INLA: An Applied Perspective. CRC Press LLC. http://ebookcentral.proquest.com/lib/southernmethodist/detail.action?docID=7042339

Rue, H., Martino, S., & Lindgren, F. (2009). INLA: Functions which allow to perform a full Bayesian analysis of structured additive models using Integrated Nested Laplace Approximation.

Salinas, D., Flunkert, V., & Gasthaus, J. (2019). DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks (arXiv:1704.04110). arXiv. http://arxiv.org/abs/1704.04110

Seyfi, A., Rajotte, J.-F., & Ng, R. T. (2022). Generating multivariate time series with Common Source Coordinated GAN (COSCI-GAN) (arXiv:2205.13741). arXiv. http://arxiv.org/abs/2205.13741

Stigler, M. (2020). Nonlinear time series in R: Threshold cointegration with tsDyn. In H. D. Vinod & C. R. Rao (Eds.), Handbook of Statistics (Vol. 42, pp. 229–264). Elsevier. https://doi.org/10.1016/bs.host.2019.01.008

Tong, H. (1978). On a Threshold Model. Pattern Recognition and Signal Processing, 42. https://doi.org/10.1007/978-94-009-9941-1_24

Tong, H. (1990). Non-linear time series: A dynamical system approach. Oxford University Press.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2023). Transformers in Time Series: A Survey (arXiv:2202.07125). arXiv. http://arxiv.org/abs/2202.07125

Woodward, W. A., Sadler, B. P., & Robertson, S. (2022). Time Series for Data Science: Analysis and Forecasting. Chapman and Hall/CRC. https://doi.org/10.1201/9781003089070

Yoon, J., & Jarrett, D. (n.d.). Time-series Generative Adversarial Networks.

Zhang, G. P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. European Journal of Operational Research, 160(2), 501–514. https://doi.org/10.1016/j.ejor.2003.08.037

Zheng, X., Xu, N., Trinh, L., Wu, D., Huang, T., Sivaranjani, S., Liu, Y., & Xie, L. (2022). A multi-scale time-series dataset with benchmark for machine learning in decarbonized energy grids. Scientific Data, 9(1). https://doi.org/10.1038/s41597-022-01455-7

Zivot, E., & Wang, J. (Eds.). (2006). Nonlinear Time Series Models. In Modeling Financial Time Series with S-PLUS® (pp. 653–712). Springer. https://doi.org/10.1007/978-0-387-32348-0_18