1. DeepAR

DeepAR is a sophisticated probabilistic forecasting model developed by Amazon, leveraging the power of recurrent neural networks (RNNs) to handle complex time series data. It is particularly advantageous for forecasting scenarios involving multiple related time series, where learning from shared patterns across these series enhances prediction accuracy. DeepAR effectively captures temporal dependencies by training on multiple time series concurrently, which allows the model to learn generalizable patterns across different series. This characteristic is precious for demand forecasting, inventory management, and other applications requiring probabilistic estimates of future values, offering not only point forecasts but also credible intervals that quantify uncertainty (Salinas et al., 2020; Flunkert et al., 2019; Gasthaus et al., 2019). Compared to traditional statistical methods, DeepAR performs better in modeling time series with intricate dependencies, especially when large volumes of data are available.

From a mathematical standpoint, DeepAR employs an RNN to model the conditional distribution of future observations given past data and covariates. Specifically, given a time series y_t and associated covariates x_t, the model aims to predict the conditional probability $p(y_{t+1} \mid y_{1:t}, x_{1:t})$ by minimizing the negative log-likelihood of the observed sequence during training.

$$\mathcal{L} = -\sum_{t=1}^{T} \log p\left(y_t \middle| y_{1;t}, x_{1;t}\right)$$

The RNN's hidden states are updated at each time step to capture temporal dependencies, thus allowing the model to learn dynamic patterns from complex time series (Salinas et al., 2020).

2. GP Forecaster

The Gaussian Process (GP) Forecaster, part of the GluonTS library, utilizes the power of Gaussian Processes for time series forecasting, offering a non-parametric approach that is particularly adept at handling uncertainty and non-linear relationships in the data. Gaussian Processes provide a flexible framework that allows for a probabilistic interpretation of predictions, which is beneficial in fields where uncertainty quantification is crucial (Rasmussen & Williams, 2006; Wilson & Adams, 2013; Snelson & Ghahramani, 2006). However, a notable limitation of GP models is their computational complexity, which scales poorly with the size of the dataset, thus restricting their scalability for large-scale forecasting tasks.

Mathematically, the GP Forecaster represents the time series as a distribution over functions, characterized by a mean function $m(t)$ and a covariance function $k(t, t')$. The forecasting process involves calculating the posterior distribution of the function values given the observed data points. Specifically, the GP prior is expressed as:

$$f(t) \sim \mathcal{GP}\left(m(t), k(t, t')\right)$$

where $m(t)$ is the mean function and $k(t,t')$ is the covariance function, which defines the correlation between any two points in time. The posterior distribution is then calculated based on the observed data, allowing the model to generate both point forecasts and credible intervals, thereby incorporating uncertainty inherent in the prediction (Rasmussen & Williams, 2006).

3. Deep State

   Deep State is a hybrid model from the GluonTS library that integrates the strengths of state-space models with recurrent neural networks (RNNs). By combining the ability of state-space models to explicitly account for trend and seasonal components with the flexibility of RNNs in modeling complex, non-linear relationships, Deep State is highly effective for a variety of forecasting problems (Rangapuram et al., 2018; Hyndman et al., 2019; Gasthaus et al., 2019). This approach is particularly useful for time series with intricate seasonal structures or those that require the modeling of latent states, which capture underlying temporal dependencies.

   The mathematical formulation of the Deep State model involves using an RNN to encode observed time series into a latent representation, which is then combined with a state-space model. The latent state x_t is updated according to the transition equation:

$$x_t = f(x_{t-1}) + v_t$$

   and observations are generated via the equation:

$$y_t = g(x_t) + w_t$$

   Where x_t represents the latent state and v_t and w_t are noise components representing process and observation noise, respectively. This formulation allows the model to leverage both temporal patterns and complex dependencies in the data (Rangapuram et al., 2018).

4. N-BEATS (Neural Basis Expansion Analysis for Time Series)

   N-BEATS is an advanced neural network-based model designed for time series forecasting without relying on domain-specific feature engineering. N-BEATS adopts a deep feed-forward architecture composed of stacks of fully connected layers, each of which contributes to generating the final forecast through backward and forward residual connections (Oreshkin et al., 2020; Smyl, 2020; Hyndman et al., 2019). This architecture allows the model to effectively capture temporal dependencies and leverage both short-term and long-term patterns in the data, outperforming many traditional methods and deep learning-based alternatives in benchmark comparisons.

The core of N-BEATS is its block structure, where each block is responsible for a different part of the forecast. Mathematically, the model aims to minimize the squared error loss function:

$$Loss = \sum_{t=1}^{T} (y_t - \hat{y}_t)^2$$

where $\hat{y}_t$ represents the model's prediction for time t , and y_t is the actual observed value. The use of residual connections between blocks enhances the ability of the model to propagate information through the network, making it particularly effective for time series with diverse dynamics (Oreshkin et al., 2020).

5. Prophet

Prophet is an open-source forecasting tool developed by Facebook, tailored for time series data characterized by daily observations, strong seasonal effects, and missing values. Prophet utilizes an additive model to decompose time series into components, including trend, seasonality, and holidays, which makes it particularly interpretable and accessible for practitioners with limited statistical expertise (Taylor & Letham, 2018; Hyndman & Athanasopoulos, 2018; Seabold & Perktold, 2010). Its primary advantage lies in its flexibility to model diverse seasonal patterns and its robustness to outliers, making it well-suited for business and social science applications. Mathematically, the model is defined as :

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

where $g(t)$ is the trend component, $s(t)$ is the seasonal component, $h(t)$ represents the holiday effects, and ϵ_t is the error term. This additive structure allows Prophet to effectively capture the different factors influencing time series dynamics, making it versatile across various domains (Taylor & Letham, 2018).

6. ARIMA

The AutoRegressive Integrated Moving Average (ARIMA) model is a classical approach for time series forecasting, widely recognized for its effectiveness in modeling univariate time series with linear structures. ARIMA integrates three main components: autoregression (AR), differencing (I) to achieve stationarity, and moving average (MA) (Box et al., 2015; Hyndman & Athanasopoulos, 2018; Brockwell & Davis, 2016). Despite its popularity, ARIMA faces challenges when dealing with non-linearities or complex seasonal patterns, which often necessitate careful pre-processing and parameter tuning. The ARIMA model can be mathematically expressed as.

$$\phi(B)(1-B)^d y_t = \theta(B)\epsilon_t$$

where $\phi(B)$ represents the autoregressive (AR) part, d is the degree of differencing, $\theta(B)$ denotes the moving average (MA) part, and ϵ_t is the error term. This formulation allows ARIMA to capture both past values and past forecast errors in predicting future values (Box et al., 2015).

7. Prophet Boost

   Prophet Boost is an advanced extension of the standard Prophet model, designed to enhance forecast accuracy by incorporating gradient boosting to model residual errors. This hybrid approach leverages Prophet's capacity to model seasonality and irregular data patterns, while gradient boosting effectively captures non-linearities that Prophet might miss (Taylor & Letham, 2018; Friedman, 2001; Hyndman & Athanasopoulos, 2018). This combination yields improved performance, particularly for datasets exhibiting complex interactions between seasonal and trend components. Mathematically, Prophet Boost extends the standard Prophet model by adding a residual correction via gradient boosting. The Prophet model itself is expressed as.

$$\hat{y}_{prophet}(t) = g(t) + s(t) + h(t) + \epsilon_t$$

   where $g(t)$ represents the trend component, $s(t)$ represents seasonality, $h(t)$ models holiday effects, and ϵ_t is the error term. In Prophet Boost, the residuals ϵt\epsilon_tϵt are further modeled using a Gradient Boosting Machine (GBM), where the boosted model adjusts for the non-linear dependencies left unexplained by the initial Prophet model:

$$\hat{y}(t) = \hat{y}_{prophet}(t) + \mathrm{GBM}(\epsilon_\mathrm{t})$$

   Here, $\mathrm{GBM}(\epsilon_\mathrm{t})$ captures the complex, non-linear residual errors from the base Prophet model, refining the final prediction. This formulation allows Prophet Boost to better capture non-linear dependencies and interactions between seasonal and trend components that might be missed by the standard Prophet model (Taylor & Letham, 2018; Friedman, 2001).

8. ARIMA Boost

   ARIMA Boost is a hybrid model that combines the strengths of ARIMA with gradient boosting to improve predictive accuracy. In this approach, an ARIMA model is first fitted to capture linear temporal dependencies, and the residuals are subsequently modeled using a gradient boosting machine to account for non-linear patterns that ARIMA may not capture (Box et al., 2015; Friedman, 2001; Hyndman & Athanasopoulos, 2018). This sequential modeling of residuals enhances the model's capacity to forecast complex time series with both linear and non-linear dynamics. Mathematically, ARIMA Boost can be represented as.

$$\hat{y}(t) = \hat{y}_{ARIMA}(t) + \mathrm{GBM}(\epsilon_\mathrm{t})$$

   where $\hat{y}_{ARIMA}(t)$ represents the linear forecast provided by the ARIMA model, and $\mathrm{GBM}(\epsilon_\mathrm{t})$ denotes the gradient boosting model applied to the residuals ϵ_t from the ARIMA model. The residuals $\epsilon_t = y_t - \hat{y}_{ARIMA}(t)$ are the errors from the ARIMA model, and the GBM models the non-linear dependencies in these residuals.

9. NNETAR

   NNETAR (Neural Network Time Series Autoregression) is an advanced forecasting technique that extends the traditional ARIMA framework by incorporating feed-forward neural networks to capture non-linear relationships in the data (Hyndman &

Athanasopoulos, 2018; Zhang, 2003; Crone et al., 2011). Unlike conventional ARIMA, which is limited to linear dependencies, NNETAR uses a neural network to model complex temporal dependencies, making it particularly effective for datasets with intricate non-linear patterns. The NNETAR model is represented by the following form.

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-p}) + \epsilon_t$$

where f is a non-linear function approximated by a neural network, and $y_{t-1}, y_{t-2}, \dots, y_{t-p}$ represents the p lagged observations (past values) used as inputs. The use of neural networks allows the model to learn from past observations in a flexible manner, capturing dependencies that are often missed by traditional statistical models (Hyndman & Athanasopoulos, 2018).

This formulation allows NNETAR to model both linear and non-linear dependencies in the data, offering a more sophisticated and flexible approach to time series forecasting, especially when non-linearity plays a significant role.

10. Exponential Smoothing (ETS)

Exponential Smoothing (ETS) is a classical time series forecasting method that models components of level, trend, and seasonality to generate accurate forecasts. The approach relies on exponentially weighted averages of past observations, making it particularly effective for time series exhibiting consistent seasonal patterns (Hyndman et al., 2008; Gardner, 1985; Hyndman & Athanasopoulos, 2018). ETS models are widely used in practice due to their simplicity, interpretability, and relatively low computational demands. The ETS model can be formulated as.

$$y_t = l_t + b_t + s_t + \epsilon_t$$

where l_t denotes the level component, b_t represents the trend, s_t is the seasonal component, and ϵ_t is the error term. This structure allows ETS models to effectively capture different aspects of the time series, providing a robust approach to forecasting in both business and industrial contexts (Hyndman et al., 2008).

11. Seasonal Regression / TBATS

TBATS (Trigonometric, Box-Cox Transformation, ARMA Errors, Trend, and Seasonal Components) is a time series model explicitly designed to handle complex seasonal patterns, including multiple seasonalities such as daily, weekly, and yearly cycles (De Livera et al., 2011; Box et al., 2015; Hyndman & Athanasopoulos, 2018). TBATS utilizes a trigonometric representation of seasonal components, coupled with Box-Cox transformations and ARMA error correction, to provide a flexible and powerful modeling framework. The TBATS model is represented mathematically as.

$$y_t = l_t + b_t + \sum_{k=1}^{K} s_{t,k} + \epsilon_t$$

where l_t represents the deterministic trend, b_t represents the trend component, $s_{t,k}$ are the seasonal components for different frequencies, and ϵ_t is the error term. This representation allows TBATS to effectively model time series with multiple overlapping seasonal cycles, making it ideal for real-world scenarios such as energy consumption forecasting (De Livera et al., 2011).

12. Stacked Model

    Stacked models, also known as ensemble methods, are powerful approaches that combine multiple forecasting models to enhance predictive performance. By aggregating the predictions from different models, such as Prophet Boost, ARIMA Boost, ETS, NNETAR, and Seasonal Regression, stacked models can leverage the strengths of each individual model while mitigating their weaknesses (Wolpert, 1992; Breiman, 1996; Hyndman & Athanasopoulos, 2018). This ensemble approach is particularly advantageous for time series with diverse characteristics, where no single model may be sufficiently robust across all aspects of the data. Mathematically, a stacked model can be represented as:

$$\hat{y}(t) = \sum_{i=1}^{N} w_i \, \hat{y}_i(t)$$

    where $\hat{y}_i(t)$ represents the forecasts from individual component models, w_i are the weights assigned to each model, and N is the total number of models in the ensemble. The weights w_i are typically determined through cross-validation to optimize the performance of the ensemble. This weighted combination results in a final forecast that often outperforms the constituent models, particularly in terms of robustness and accuracy (Breiman, 1996).

Bibliography

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

Breiman, L. (1996). Stacked regressions. *Machine Learning*, 24(1), 49-64.

Brockwell, P. J., & Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer.

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.

De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496), 1513-1527.

Flunkert, V., Salinas, D., & Gasthaus, J. (2019). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 35(4), 1370-1381.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 1189-1232.

Gardner, E. S. (1985). Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1), 1-28.

Gasthaus, J., Benidis, K., Wang, Y., Flunkert, V., Bohlke-Schneider, M., Salinas, D., & Januschowski, T. (2019). Probabilistic forecasting with spline quantile function RNNs. *International Conference on Artificial Intelligence and Statistics*.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts.

Hyndman, R. J., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with Exponential Smoothing: The State Space Approach*. Springer.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for time series forecasting. *International Conference on Learning Representations*.

Rangapuram, S. S., Seeger, M., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. *Advances in Neural Information Processing Systems*, 31, 7785-7794.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181-1191.

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. *Proceedings of the 9th Python in Science Conference*.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75-85.

Snelson, E., & Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, 18, 1257-1264.

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37-45.

Wilson, A. G., & Adams, R. P. (2013). Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning*, 1067-1075.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241-259.

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50, 159-175.