Time Series Forecasting with Memorized Seasonal Structures

Abstract

Companies are dedicating more and more resources to the centralization and automation of forecasting capabilities. As organizations move towards scalable, generalized algorithms, standardized logic puts item-level forecasts at risk of being produced with limited causal information, or on antiquated history which no longer represents the current structure of the time series.

The objective of DE4S (Double Exponential Smoothing with Switching Seasonal Structures) is to decrease forecast sensitivity to mis-specified training data through a combination of decomposition forecasting and a state-space approach to seasonal forecasting. The full code for DE4S can be found at https://github.com/kmana1995/DE4S_Model.

1 - Introduction

Forecasts motivate business decisions including but not limited to inventory planning, facility selection, network optimization, financing, sourcing, and labor planning. As such, even incremental improvements in forecast accuracy can have considerable impact on an organization.

Although automated time series forecasting models can produce low-touch forecasts to great levels of accuracy, forecasts still exhibit some of their greatest improvement via constant adjustment and feature engineering/incorporation from a dedicated analyst. Unfortunately, due to the expected prioritization of a few, high visibility SKU’s, many items do not receive the level of attention they need to be optimally forecasted.

Similar to the motivation of the Facebook Prophet algorithm, the objective of DE4S is to enable more accurate and robust forecasts at scale, across an entire product portfolio (*see Taylor and Letham, 2017 [1])*. Specifically, DE4S is focused on accurately forecasting the effects of varying seasonal structures within the same product, store, or other hierarchical level desired with minimal to no intervention.

While some forecast models assume the seasonal structure may change over time, very few assume anything can be learned from past seasonal structures committed to memory. DE4S not only assumes seasonal structures can change over time, but commits all identified structures to memory with the assumption that they provide useful information for future predictions.

2 - Common Techniques

To evaluate the accuracy of DE4S, we benchmarked its performance against several other commonly used seasonal and non-seasonal time-series forecasting methods. We will use the widely accepted forecast metric MAPE (mean absolute percent error) to evaluate our results.

$$MAPE = \left(\sum_{t}^{n}((f_t - a_t) \div a_t)\right)/n$$

Where:

f_t = forecast of step t
a_t = actual value at time step t
n = number of observations

A brief summary of the models being benchmarked and their strengths/weaknesses:

1. *Simple Moving Average* - a simple average of the last n observations is propagated forward as a prediction. The forecast can be good for short time horizons, and if n is large the forecast can be very stable to structural shocks. However, a lack of trend or seasonal component can lead to sub-optimal forecasts, particularly over extended horizons.
2. *Holt-Winters/Triple Exponential Smoothing* - a decomposition forecasting method comprised of trend, level, and seasonal components. Seasonality can be modeled as either an additive or multiplicative component, and the flexibility of all components can be controlled by the analyst with smoothing parameters alpha, beta, and gamma *(see Alfares and Nazeeruddin, 2002 [2]).*
3. *ARIMA* – it is useful to think of ARIMA in three parts: AR, I, and MA each with corresponding parameters p, d, and q. “AR” stands for autoregressive, in which coefficients are applied to p lagged observations of a time-series. “I” represents an order of integration, managed by the d parameter to control for stationarity. And “MA” signifies a moving average component, in which coefficients are applied to the autoregressive error term *[2]*.
4. *Facebook Prophet* - an additive decomposition model designed to model trend, seasonality, and holidays. Trend is modeled using either piecewise linear regression or logistic saturating growth, seasonality is modeled using a Fourier series, and the impact of holidays are incorporated as exogenous regressors *[1]*.

Simple average and ARIMA do not model seasonal structures at all, holt-winters immediately forgets prior seasonal structures in lieu of new ones, and Facebook prophet’s Fourier series limits the model to only one seasonal structure per level (hour, week, year, etc.). DE4S does not aim to outperform these models universally, but rather equip an analyst or ensemble with a forecast model designed to handle multiple seasonal effects.

3 - DE4S (Double Exponential Smoothing with Switching Seasonal Structures)

DE4S is a decomposition model with multiplicative seasonality. The general formulation is:

Eq 3.1 Generalized Forecast Formula

$$f_t = (L_{t-1} + G_{t-1})S_t$$

Where:

f_t = forecast of step t

L_{t-1} = level at time step t-1

G_{t-1} = growth factor at time step t-1

S_t = multiplicative seasonal effects at step t

3.1 - Trend and Level - Double Exponential Smoothing

Trend and level are estimated using double exponential smoothing. Exponential smoothing is applied on a rolling aggregate of the forecasted variable, with the number of observations being controlled by the seasonal level defined. To illustrate, if we are forecasting weekly seasonal effects, double exponential smoothing will derive the level and trend components from rolling seven-day windows to avoid variance which may be attributable to seasonal effects.

Double exponential smoothing accepts two smoothing parameters, alpha and beta for level and trend respectfully. Low level or trend smoothing parameters will equate to a relatively stable forecast, robust against structural shocks albeit potentially slow to adapt. High smoothing parameters will do the opposite, and provide a highly adaptive model that may be prone to over-reaction. The level and trend are updated via:

Eq. 3.2 Double Exponential Smoothing

$$L_t = \alpha Y_t + (1 - \alpha)(L_{t-1} + G_{t-1})$$
$$G_t = (1 - \beta)G_{t-1} + \beta(L_t - L_{t-1})$$

Where:

L_t = level at time step t

Y_t = observation at time step t

G_t = growth factor at time step t

α = level smoothing parameter

β = growth smoothing parameter

3.2 - Initial Seasonal Structure Estimation

We estimate the initial seasonal structures using KMeans clustering, considering discrete points in our now partially decomposed time-series as a feature. If we were evaluating weekly seasonality, each day of the week would be considered a feature for a given week (cycle).

We iteratively fit KMeans clustering to the training set up to a user-defined parameter for max profiles. With each iterative clustering, we discard any members which are assigned to their own individual cluster. This reduces our training sample for the proceeding clustering; however, it is by design. Discarding a solely-assigned member serves as a form of outlier detection. Given the intent of this step is to identify generalized seasonal structures, structures assigned to their own cluster provide no value.

Once we have iteratively fit clusters up to the max profiles, we select the optimal cluster number using second-level differencing of distortion to identify where additional clusters return marginal benefits, this serves as an analytical proxy to the “elbow point” of distortion (*see Bholowalia and Kumar, 2014 [3])*. This optimal cluster number effectively becomes the accepted number of seasonal structures. We also initialize the structure parameters by estimating the mu and sigma parameter sets of each seasonal structure. These parameter sets represent the estimated average seasonal effect of a given state, and the uncertainty of that estimate.

The mu and sigma parameter sets are differentiated across each “point” of a cycle. A state modeling weekly seasonality would have seven sets of (μ, σ) parameters per seasonal state, one for each day of the seasonal cycle.

3.3 - Seasonal Switching Model

After the parameter sets of initial seasonal structures are evaluated, we consider each structure to be emblematic of a “hidden state” to be modeled using a hidden Markov model (HMM). HMM’s operate under the assumption that observations are not necessarily random, but omitted under the influence of an unobservable but influential state. These unobservable states are furthermore perceived to follow a Markovian process, in which the likelihood of a future state is solely dependent on the current state *(see Jurafsky and Martin, 2019 [4])*. HMM’s are composed of two major parameter sets:

- the prior information of hidden states, which is used to evaluate the probability of an observation being omitted from a given state
- A transition matrix, which considers the likelihood of “switching” from state to state

Once the parameters of an HMM are fairly estimated, the model offers state and observation probabilities at a given time step (predictions).

HMM’s require a unique two-step fitting procedure; one to tune the parameter set, and another to fit the most likely “forward pass” of a time-series (a necessary process which blinds the HMM to future observations). Fitting the parameter set is achieved using the Baum-Welch algorithm.

3.3.1 - Baum-Welch Algorithm

The Baum-Welch algorithm is a form of expectation-maximization algorithm which iteratively estimates the parameter set of an HMM until convergence. The Baum-Welch algorithm is often referred to as the forward-backward algorithm, as it leverages forward and backward passes through the observation set. Consider a time series of *n* observations:

- The forward pass evaluates the likelihood of observing series t_1:t_i given placement in state *j* at step *i*

- The backward pass evaluates the likelihood of observing series t_i:t_n given placement in state *j* at step *i*

By including both the forward and backward pass, the Baum-Welch algorithm avoids inputting any bias by nature of the direction being considered [4].

Eq 3.3 Baum Welch Algorithm

Step 1 - Forward Pass

1a. Initialize $\alpha(1) = \pi_j b_j(y_1)$ for observation 1

1b. For all i∈(2:N):

$$\alpha_j(i+1) = b_j(y_{i+1}) \sum_{l}^{N} \alpha_l(i) t_{lj}$$

the probability of i+1 coming from state j, multiplied by an aggregate transition probability from all states l to j

Step 2 - Backward Pass

2a. Initialize $\beta = 1$

2b. For all i∈(N-1:0):

$$\beta_j(i) = \sum_{l}^{N} \beta_l(i+1) t_{j_l} b_l(y_{i+1})$$

the probability of i+1 coming from state l, multiplied by an aggregate transition probability from all states j to l

Step 3 - Evaluate the probabilities of observation i emitting from state j for all i∈N, leveraging forward and backward probabilities computed in steps 1 and 2:

$$P_j(i) = \frac{\alpha_j(i)\beta_j(i)}{\sum_l^N \alpha_l i \beta_l(i)}$$

Step 4 – Evaluate the temporary transition probabilities $u_{lj}(i)$ from state l to state j, for all observations i∈N-1:

$$u_{lj}(i) = \frac{\alpha_l(i)\beta_j(i+1)b_j(y_{i+1})}{\sum_k^N \sum_w^N \alpha_K(i) a_{kw} \beta_w(i+1) b_w(y_{i+1})}$$

Step 5 – Re-asses the HMM parameter set:

State distribution π = P

Transition matrix value $t_{lj} = \frac{\sum_i^{N-1} U_{lj}(i)}{\sum_i^N P_l(i)}$

Step 6 – Iterate steps 1-5 until convergence. For our purposes, convergence is estimated when the aggregate value of $P_j(i)$ for all i∈N changes by no more than one percent of its previous value.

Where:

S = number of states
N = number of observations
α = forward probability
β = backward probability
X_t = state of time step t
T = transition matrix of SxS dimensions comprising of t_{lj}, which is probability of transition from state l → j
B = an array of SxN dimensions, containing the likelihoods ($b_j(y_i)$) of observing observation i from state j for i∈N and j∈S
π = initial state distribution
θ = the parameter set (A, B, π)

At termination of the Baum-Welch algorithm, we have produced our final estimates of the HMM parameter sets, which can be summarized as an estimate of the underlying seasonal structures, as well as the probability of switching between seasonal structures. These parameter sets are visualized in figure 1.

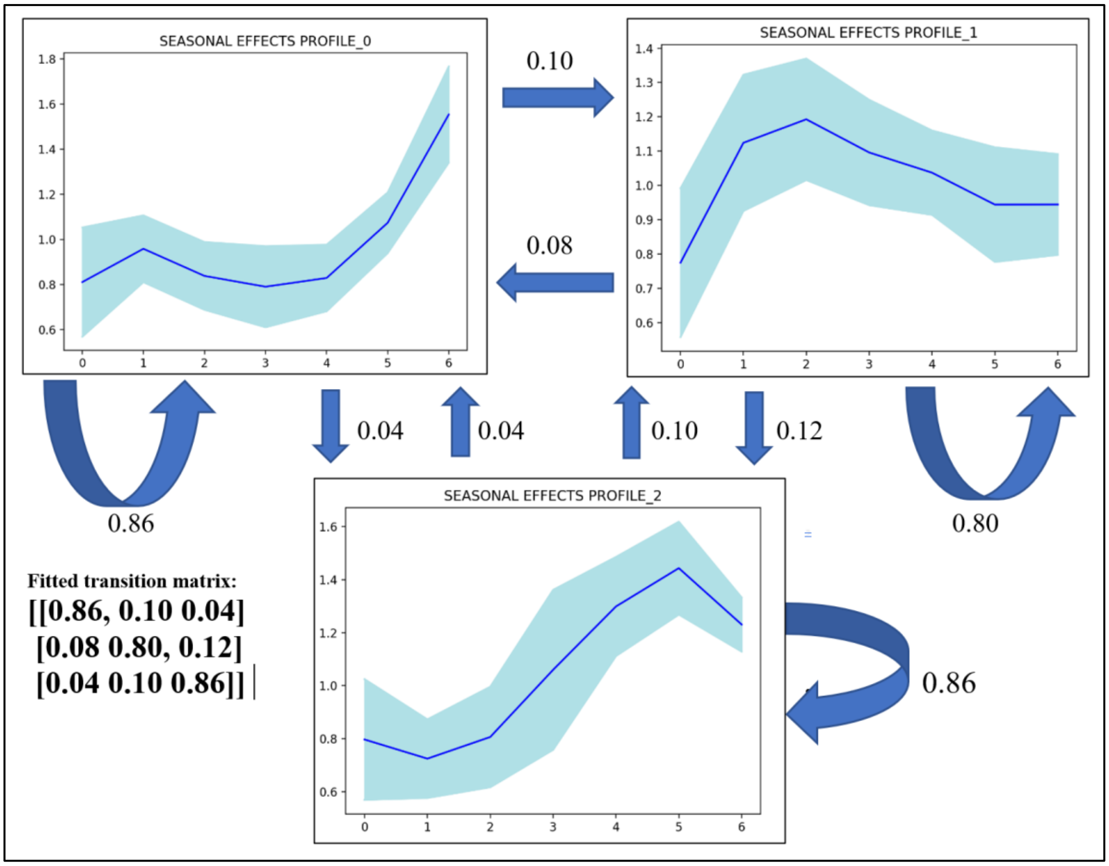


Figure 1 – Visual representation of the parameter set of an HMM. Each state's seasonal structure and uncertainty (80% confidence interval) is displayed. Additionally, the likelihood of transitioning between states is represented.

3.3.2 - Viterbi Algorithm

The objective of the Viterbi algorithm is to assemble the most likely sequence of states which could have produced the observation sequence. Similar to the Baum-Welch algorithm, it depends on a forward pass to evaluate emission probabilities from a given sequence of states. The number of possible scenarios compounds exponentially with the number of observations in a time series due to combinatorial explosion. This mandates the use of dynamic programming storing the optimal path in a forward trellis, containing the MAP of each state at step *t*. At the end of assembly, the forward trellis is back-traced, starting with the most likely state observed at point n and working backward through the trellis to determine the optimal path [4]. This process is used to create fitted values of our seasonal components.

3.3.3 - Generating Seasonal Predictions Using HMM

Once the HMM parameter sets and forward pass are estimated, we can generate predictions of the seasonal effects by using the Maximum a Posteriori probability of the HMM's hidden state and observation. This probability is summarized as:

$$S_t = MAP\ of\ p(s_t|\theta, \alpha_{t-1})$$

Where:

S_t = seasonal impact of time step S
α_{t-1} = forward state probabilities of t-1
θ = parameter set of the fitted HMM (A, B, π)

5 - Data and Testing Method

We use three datasets for the preliminary benchmark of DE4S against the aforementioned time series forecasting methods. The first dataset is 550 days of historical web searches for the term "NFL". This data was pulled from Kaggle at https://www.kaggle.com/c/web-traffic-time-series-forecasting. This data strongly expresses the time series structure that DE4S is designed to accurately forecast; varying seasonal structures which are repeatedly visited by the time series (figure 2). We will perform 180 out-of-bag (OOB) predictions on the NFL data.

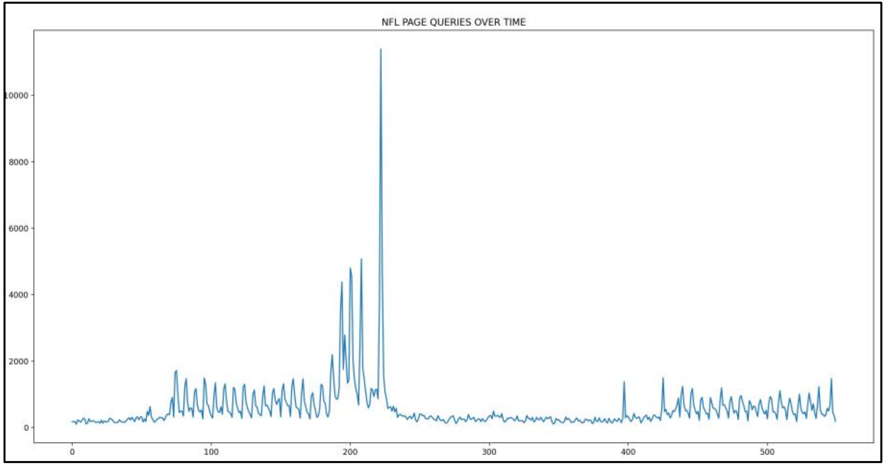


Figure 2 – plotted time series of NFL page queries. Note the varying seasonal structure from observation ~75 to ~190, revisited again at observation ~425 and onward.

The second dataset is a time-series provided through Kaggle by the Russian Software firm 1C Company, found at https://www.kaggle.com/c/competitive-data-science-predict-future-sales/overview. It contains sales data split by product. For this exercise, we've aggregated sales to the day grain to test the performance of our relative models. This leaves us with 1,045 observations. It is worth noting that this time-series does not hold the seasonal structure seen in the NFL search dataset, and any seasonal structures are much more difficult to discern with visible inspection (figure 3). We will perform 350 OOB predictions on the 1C Company data.

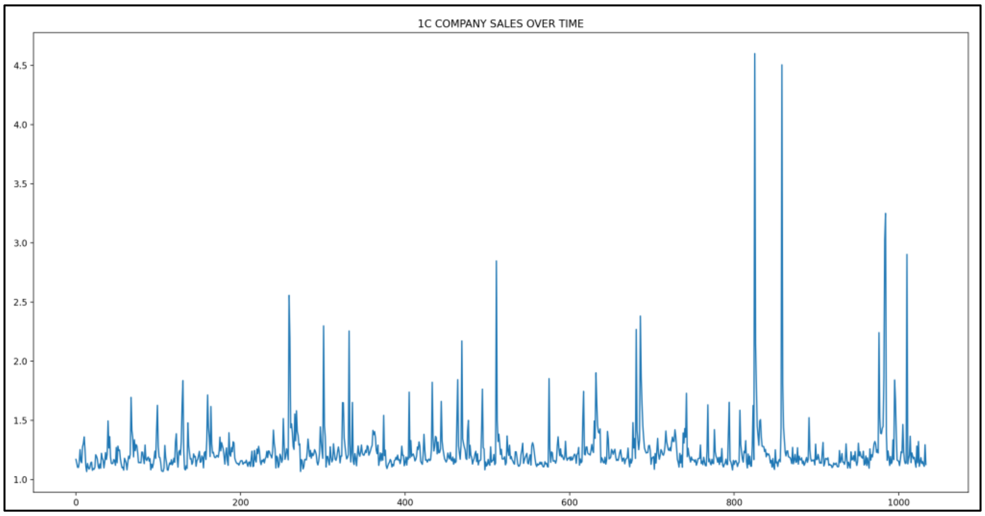


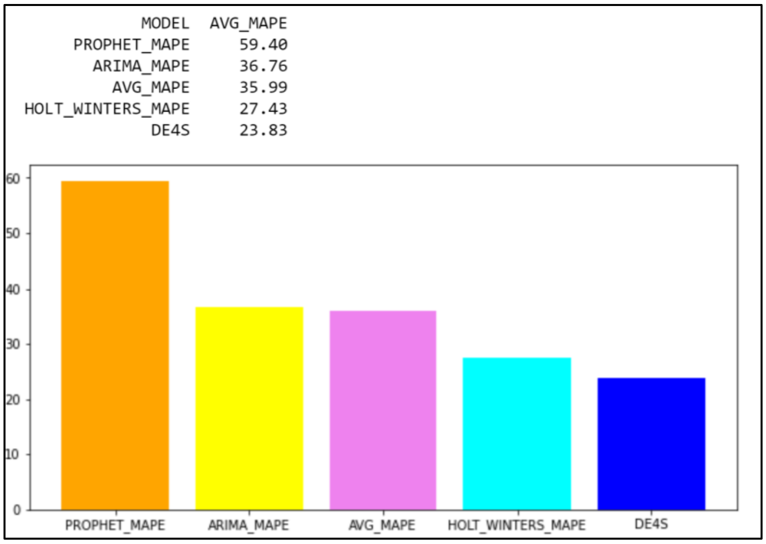
Figure 3 – plotted time series of 1C Company sales data.

The third and final dataset is the time-series data found at https://www.kaggle.com/c/demand-forecasting-kernels-only. This is our most robust dataset and can be considered our "full benchmark" due to the large evaluation size. This data is part of a competition evaluating forecast performance, and contains five years of data on fifty items across ten stores. To perform our benchmark, we randomly select five stores, twenty items, and thirty dates from the last half of the date range. We then subset the data by store, item, and truncate it to only fit the model on data less than the prediction date. This creates a sample of 3,000 predictions to evaluate the models against.

4 - Performance

4.1 - NFL Dataset Performance

We evaluate the performance across 180 samples of the NFL dataset with a forecast horizon of one day. Analyzing the results of DE4S on this highly seasonal time series, we see DE4S outperforms all other evaluated models including Prophet, ARIMA, Simple MA, and Holt-winters (figure 4). Notably, Prophet struggles due to the multi-seasonal structure of the time-series and Prophet's inability to handle such inconsistencies (figure 5). Continuing discussion surrounding a multi-seasonal structure, it is perhaps unsurprising that holt-winters was closest to DE4S in terms of performance, as it is the only other model capable of adjusting its seasonal assumptions.



```
            MODEL  AVG_MAPE
     PROPHET_MAPE     59.40
       ARIMA_MAPE     36.76
         AVG_MAPE     35.99
HOLT_WINTERS_MAPE     27.43
             DE4S     23.83
```

Figure 4 – Average MAPE by forecast type on NFL data.

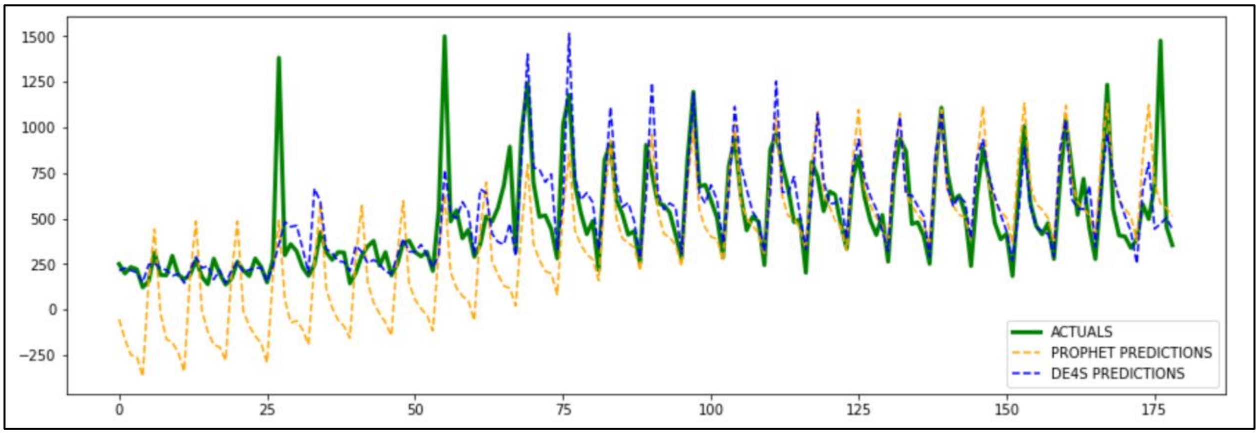


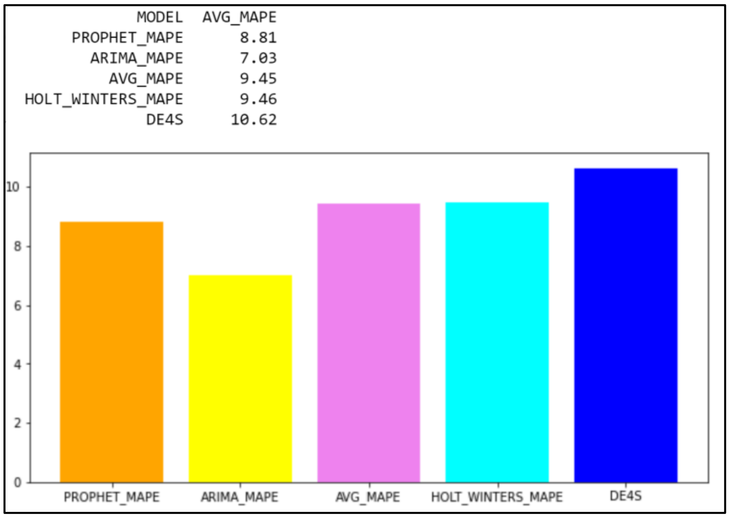
Figure 5 – Prophet Forecast (orange), DE4S Forecast (blue), and actuals (green). Note the extent to which Prophet struggles at the beginning of the period, as it maintains a single seasonal structure throughout. DE4S clearly "switches" seasonal structures around observation 70.

4.2 - 1C Dataset Performance

We also evaluate the performance of each model across 350 samples of the 1C Company sales data. It is important to remember that this sales data expressed little to no weekly seasonal structure. This is an important test for the DE4S model in relation to its peers, as it will evaluate the model's ability to contend when seasonal structures are less obvious or even non-existent.

What we find is that while the marginal performance between models is much smaller, DE4S performs slightly worse than its peers (Figure 6). Note that in contrast with the NFL dataset, in which the MAPE performance ranged by 35.57% from highest to lowest, the MAPE performance on the time series provided by 1C Company ranged by only 3.59%.

A final note worth mentioning is that two non-seasonal models performed in the top three on the 1C Company data (ARIMA 7.03% MAPE and Simple Average 9.45% MAPE), with ARIMA registering the lowest MAPE of all five models. This type of performance is suggestive of weak to no seasonal structure of the data.

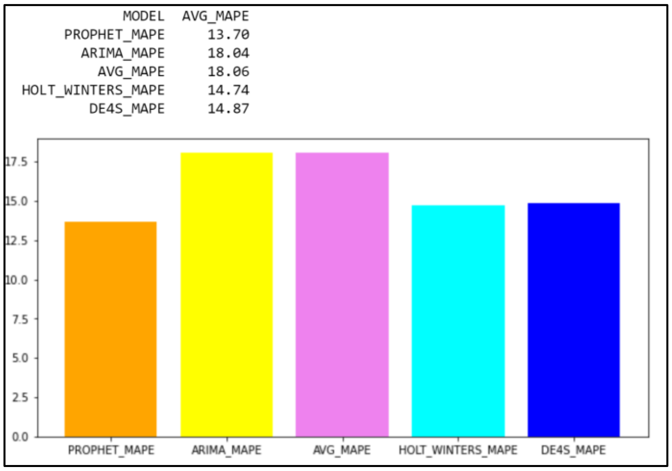


```
            MODEL  AVG_MAPE
     PROPHET_MAPE      8.81
       ARIMA_MAPE      7.03
         AVG_MAPE      9.45
HOLT_WINTERS_MAPE      9.46
             DE4S     10.62
```

Figure 6 – Average MAPE by forecast type on 1C Company Data.

4.3 - Large Dataset Performance

On the third and final benchmark, running 3,000 predictions across five stores, ten items, and thirty randomly selected dates, we again see a much tighter relative performance. Across the 3,000 predictions, the MAPE for DE4S was 14.87%, placing third behind Holt-Winters (14.74%), and Prophet (13.70%) (Figure 7). The performance of seasonal models versus non-seasonal (ARIMA and Simple MA) suggests the time-series contained seasonal structure across a majority of item/store combinations. Furthermore, the marginally poorer performance of DE4S may suggest that the seasonality was rather consistent over time. Across this benchmark, DE4S could be fairly evaluated as a competitive seasonal time-series forecasting method.



```
            MODEL  AVG_MAPE
     PROPHET_MAPE     13.70
       ARIMA_MAPE     18.04
         AVG_MAPE     18.06
HOLT_WINTERS_MAPE     14.74
        DE4S_MAPE     14.87
```

Figure 7 – Average MAPE by forecast type on Miscellaneous Kaggle Data.

Worth considering across this benchmark is not only accuracy as measured by MAPE, but also the precision and bias of each forecasting method (Figure 7). When looking at a histogram of the error for each forecasting model, we observe that DE4S had the second-highest level of precision with regard to forecast accuracy, having a standard deviation of error of only 8.84 (Prophet exhibited the highest precision with 8.07). Additionally, DE4S similarly exhibited the second-lowest level of bias, with an average bias of only -0.24 (Prophet again exhibited the lowest bias of -0.12).

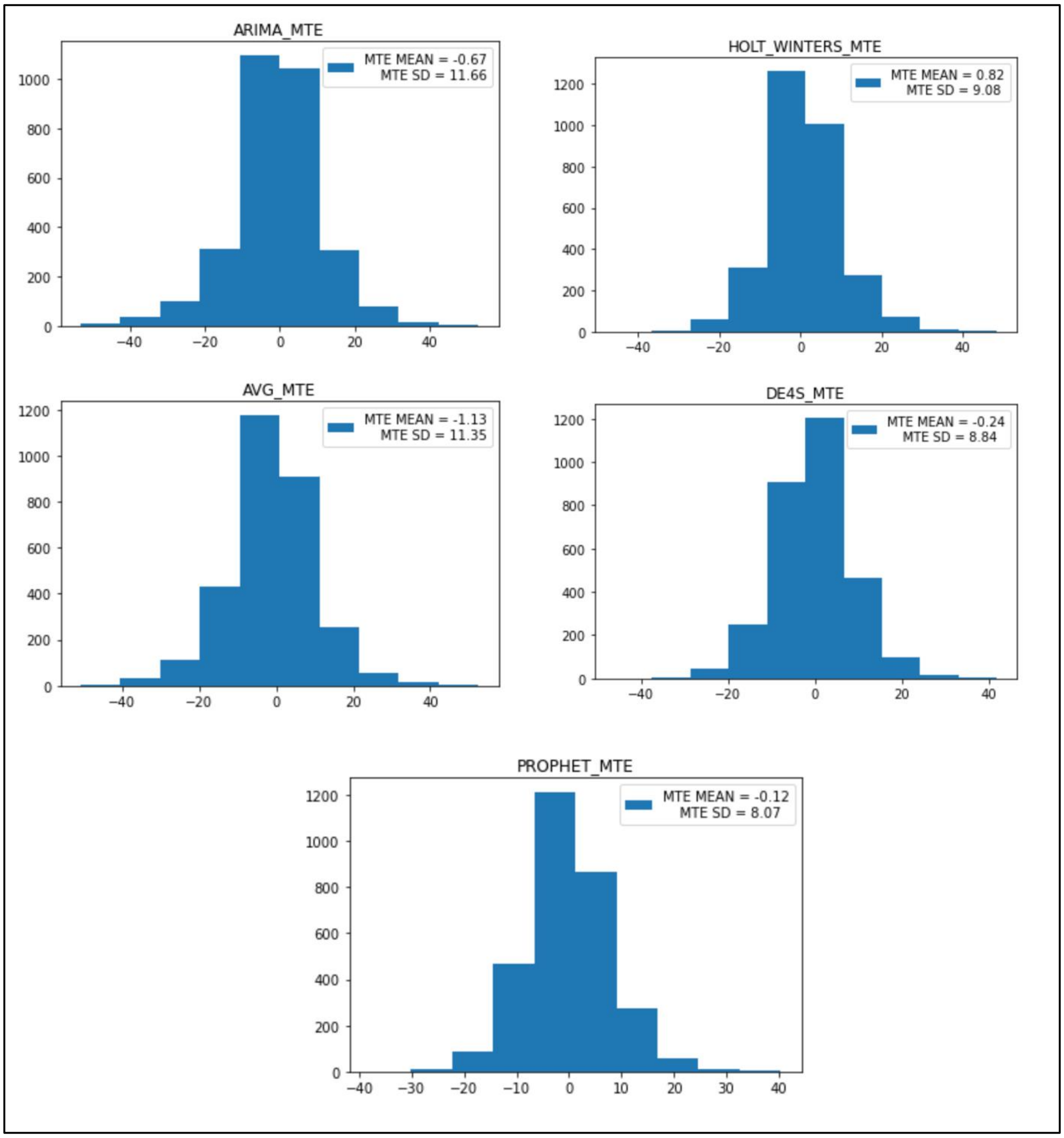


Figure 7 – Distribution of error showing precision and bias of each model.

5 - Conclusion

Key takeaways from the benchmark results are:

1. In the presence of multiple seasonal structures, DE4S is extremely competitive in relation to traditional time-series methods, and worth consideration as either a sole method, or as a member within a broader ensemble.
2. In the presence of low to no seasonal structures, DE4S performs worse than non-seasonal methods such as ARIMA and Simple MA, and exhibited a minor deficiency in relation to other seasonal methods. As such, it may be worth considering the seasonal structure of a time-series before applying DE4S.
3. Across a large sample of items which may or may not exhibit seasonal structures, DE4S exhibits forecasting accuracy in line with other seasonal methods. When considering MAPE, bias, and precision, DE4S proves a capable forecasting method.

Given these results, DE4S is worth consideration particularly in forecasting applications which are expected to scale across a large number of items. DE4S exhibits competitive accuracy measurements when run on non-seasonal structures, and when run at scale. More importantly, the performance of DE4S when forecasting on the NFL dataset clearly exhibits DE4S's ability to accurately forecast items with seasonal elements that may be impacted by underlying forces which are difficult to associate at scale such as weather, macro-economic health, nearby competition, universities, local events, local employment, etc.

WORKS CITED

[1] Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2

[2] Alfares, H. K., & Nazeeruddin, M. (2002). Electric load forecasting: literature survey and classification of methods. *International journal of systems science*, *33*(1), 23-34.

[3] Bholowalia, P., & Kumar, A. (2014). EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications*, *105*(9).

[4] Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing. *Upper Saddle River, NJ: Prentice Hall*.