Beating the beast: Kaggle our own way Team35

Adam Mehdi Arafan (11595019), Riccardo Fiorista (14012987), Emiel Steegh (14002558) and Nina Spreitzer (13725378)
Applied Forecasting in Complex Systems 2021
University of Amsterdam

ABSTRACT

This paper describes the forecasting approaches implemented by Team 35 (Appendix A) for participating in a Kaggle forecasting challenge using a subset of the M5 Accuracy data. The objective is to forecast the daily demand for food items 28 days in advance. Two of the biggest challenges are handling time series with intermittent demand and including additional explanatory variables. Various forecasting methods are applied to test whether more complex models outperform simple ones. The forecasting methods range from simple benchmark methods over more complex ones, such as ARIMA, to a machine learning approach. The best result on the test set is achieved by the machine learning method XGBoost.

1 INTRODUCTION

The M competitions are one of the most cited and significant challenges in the field of forecasting [23]. In general, the objective of forecasting challenges is the evaluation of known forecasting models, and elaboration of new methods [24]. The M5 Forecasting - Accuracy contains data presenting unit sales of the world's largest retailer, Walmart [9]. The data includes hierarchical item levels, detailed item information, and individual daily demand. Supplementary, descriptive data such as sell price and calendar-related information containing weekdays and events is provided. According to Bandara et al. [3], real-world forecasting patterns align closely with the structure of this competition. The three main innovations introduced by this competition are the focus on intermittent time series, incorporation of hierarchical elements, and consideration of exogenous variables [12].

This paper refers to a course-specific challenge called *Sales Time Series Forecasting* hosted on Kaggle[1]. The challenge uses a subset of M5 Forecasting - Accuracy sales data focusing on one store *TX3* in the State of Texas, taking only item category *FOODS* into account. This competition also aims at forecasting daily sales for 28 days in the future. The data consists of around five and a half years in the past. The set-up of the challenge removes the barrier of dealing with hierarchical structure within this work. Nevertheless, the intermittency mentioned above still occurs in the time series, which means sporadic demands with many zeros are included. According to Ghobbar et al. [10] it is highly beneficial to identify an accurate method to deal with such series, as they are typical for retail sales at a store level and difficult to predict with conventional forecasting methods. Furthermore, it is challenging to include external data such as calendar details and sell price as they need to be merged with the demands appropriately. Our approach aims to test if more complex models result in more accurate forecasts than simple methods. We want to efficiently use additional information by focusing on appropriate feature engineering, as researchers [13] [18][19] claim that adding exogenous variables to a time series can be used to improve performance. The code for the forecasting methods explained in this paper can be found on a GitHub repository [2].

The remainder of this paper consists of five main parts. The following Section highlights related work, giving a background of M competitions. Subsequently, we describe the utilized dataset and preparation steps, followed by the methodology, including a description of the applied forecasting models. Subsequently, we discuss the results of the generated forecasts. The last part discusses encountered limitations, challenges, and future steps. The paper concludes with a summary of our findings.

2 RELATED WORK

A non-systematic review of forecasting methods based on intermittent demand, including parametric and non-parametric approaches, is provided by Petropoulos et al. [29]. Fildes et al. [7] identify key problems that retailers face when forecasting retail demand and conclude that there is evidence that ML methods could outperform the benchmark and causal models. The previous competition, namely M4 [21], was conducted in 2018 and reported a dramatic increase in participation compared to former ones. Various discussions and comments were made about this competition [6][8][11][14][20][27]. Following these comments the M5 challenge was designed aiming to address the concerns

[1] https://www.kaggle.com/c/sales-time-series-forecasting-tx-afcs2021

[2] https://github.com/ninaspreitzer/Forecasting_Project

raised. Makridakis et al. [20] describe the background, implementation, and organization of the competition, as well as a recommendation on how the data should be used appropriately. The results and best-performing methods are presented in [23]. The paper outlines the major findings of the competition and their implications. All top performers used Machine Learning (ML) methods, as they performed significantly better than those using statistical benchmark methods and their combinations. The method that achieved the highest accuracy was an ML algorithm performing non-linear regression by using gradient-boosted trees [16].

3 DATA AND PRE-PROCESSING

It is necessary to prepare the underlying data so that models can retrieve complete information. By doing that, forecasting models can achieve good results. The available data includes four independent datasets. The training dataset consists of 823 rows, representative for each food item, and 1913 columns indicating the demand of each item per day. The data used for validation shows the same structure as the training set, except holding the upcoming 28 days to predict. The actual test dataset is stored in Kaggle is not available to us. The other two datasets include additional information. The calendar data includes 11 columns providing information such as weekday, week, month, year, event name, and types (e.g., *Superbowl* with type *Sport*), as well as dates on which eligible families receive monetary benefits from the Supplement Nutrition Assistance Program (SNAP) citefalk2014supplemental. Additionally, the price dataset shows weekly sell prices for each item.

Pre-processing aims to merge our training data with the exogenous datasets and perform feature engineering to have one dataset holding interpretable values. As training data is only captured each day by *d1...d1913*, we merged it with the calendar data as the dates are in the same order. With a dataset holding demand and calendar information for each day, the next step is to include weekly sell prices. We do so by using the unique item IDs and week IDs. Finally, this results in one dataset holding all available information; the next step is appropriately dealing with categorical values.

We used multiple models for the forecasting task, so we decided to continue with two different datasets. One dataset transforms categorical values with one-hot encoding, and the second one keeps most of the categorical values as they are. This is because tree-based models do not perform better on encoded categorical values. However, they greatly improve the regression models. When pre-processing the dataset for regression models, one-hot encoding is used to present all categorical values, except the column *year*. This is because applying one-hot encoding means that the order of categories does not matter. However, forecasting recent years are more valuable. Hence the order does matter, and we keep those values as factors. One-hot encoded weekdays result in six columns, as Sunday is implicitly presented. Similarly, for months with eleven dummy variables. Weekdays and months stay as one categorical column for the second dataset. However, we transform all event columns to a dummy variable indicating if an event exists for this date for both datasets. Our assumption here is that it does not matter significantly to what type an event belongs (e.g. *Cultural* or *Religious*. Additionally, we add columns capturing dates one to five and five to ten before an event to specify each event date. This is because we assume increasing food purchases days before an event. All columns that hold redundant information are then dropped. Rows with neither demand nor sell price are also dropped, as we assume that the products did not exist then. The final datasets, presented in table 1 and 2, are then stored as *tsibble* objects to allow for a simplified model fitting, validating, and forecasting pipeline. To successfully forecast demand for the next 28 days within the Kaggle competition, we had to prepare a dataset that contains the required predictors for the corresponding dates.

Table 1: Dataset for Regression Models

Column	Type	Columns cont.	Type
item_id	Integer	snap_TX	Binary
demand	Integer	Monday	Binary
date	Date	...	...
year	Factor	Saturday	Binary
sell_price	Numeric	Month1	Binary
isEvent	Binary	...	...
FiveDays	Binary	Month11	Binary
TenDays	Binary	isWeekend	Binary

Table 2: Dataset for Tree-based Models

Column	Type	Columns cont.	Type
item_id	Integer	FiveDays	Binary
demand	Integer	TenDays	Binary
date	Date	snap_TX	Binary
year	Factor	weekday	Binary
sell_price	Numeric	month	Binary
isEvent	Binary	isWeekend	Binary

4 METHODOLOGY

Exploratory Data Analysis

To give an impression of how the demand and price of all products evolved over the years, we plot the average of demand for all products in Figure 1. It can be seen that a striking seasonal element with dips in demand before new years and

peaks during the summer occurs. Also, there was an outlying peak in the summer of 2016, which could be the result of inconsistent underlying data.

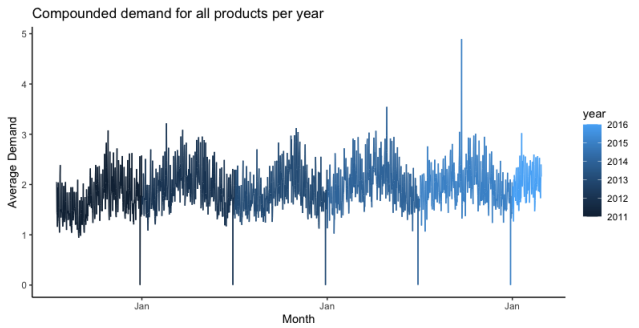

Figure 1: Compounded demand for all products per year

Figure 2 illustrates the average sell price over the years. The time series shows an increasing trend with some cyclic behavior. It needs to be mentioned that the sell price data is far from complete to draw a sufficient picture.

Figure 2: Compounded demand for all products per year

One of the key-finding and, at the same time also, one of the greatest challenges with this dataset is the extremely unbalanced target value. This results from dealing with a time series of counts with intermittent values and high inflation of zeros. In fact, more than 60% of demands are 0.

Models

To forecast the demand for each item 28 days in the future, we fit the training data by applying various forecasting methods. First, we will start with very simple benchmark methods, shifting to more advanced methods like ARIMA and exponential smoothing (ETS). These models are based on the description in *Forecasting: Principles and Practice* [15]. We also use methods beyond this book's scope by exploring General Linear Models, Zero-inflated Poisson Regression, and XGBoost.

By making forecasts, we make the assumption that the residuals have a mean of zero and are normally distributed

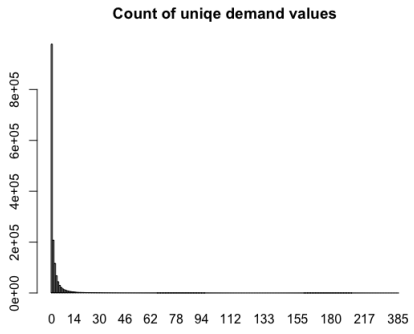

Figure 3: Count of unique demand values

with $N(0, \sigma^2)$. Furthermore, we assume that the errors don't show auto-correlation and are unrelated to the predictor variable.

Baselines. As extremely simple forecasting methods are, in some cases, surprisingly effective, we will first apply the following four simple forecasting methods as benchmark models are used to fit the training set: Mean, Naïve, Seasonal Naïve, and Drift.

- **Mean**: In the first attempt, the forecasts of all feature demands are equal to the mean of the historical values. This is done by averaging the corresponding daily demands for each product.
- **Naïve**: As the name suggests, the Naïve method sets all forecasts to the value of the last observation. This method is primarily used for data following a random walk, which typically shows long periods of apparent trends and unpredictable changes in direction. As the underlying data does not appear to follow a random walk, we know a priori that this method will not be the most appropriate for this dataset.
- **Seasonal Naïve**: This method is similar to Naïve, with the difference that it takes seasonal behavior into account. The forecast is equal to the last observed demand from the same season. Since the average of all demands seems to behave as seasonal data, this approach performs most certainly better than the Naïve method.
- **Drift**: Another variation of the Naïve method takes trends into account. The drift is equal to the average change over a time period. Therefore the Drift method allows an increasing or decreasing forecast over time. Considering that the demand does not seem to have an obvious trend over the observed years, we expect this method not to perform better than Naïve.

Croston's method. This is the most known and used method to predict intermittent demand or, generally speaking, count data and was first proposed in [5]. The method involves separating our original time series into two separate ones. One time series is denoted by a, represents the periods containing zero values, and is coined *inter-arrival time*. The other, referred to as *demand*, represents the periods containing non-zero values. We then forecast both time series using Simple Exponential Smoothing (SES) and estimate from the data the smoothing parameters $\alpha_q, \alpha_a \in [0,1]^{IR}$. Based on a sum, weighted by the estimated smoothing parameters α_q, α_a, between the last observed values for q, a and the forecasted ones $\hat{q}, \hat{a}$, we obtain the one-step forecasts $\hat{q}_{i+1|i}, \hat{a}_{i+1|i}$ (as in Equation 1). Finally, we use them to create the forecast $\hat{y}_{T+h|T}$ in Equation 2 [15].

$$\begin{aligned} \hat{q}_{i+1|i} &= \left(1-\alpha_q\right)\hat{q}_{i|i-1} + \alpha_q q_i \\ \hat{a}_{i+1|i} &= \left(1-\alpha_a\right)\hat{a}_{i|i-1} + \alpha_a a_i \end{aligned} \tag{1}$$

$$\hat{y}_{T+h|T} = \hat{q}_{j+1|j} / \hat{a}_{j+1|j} \tag{2}$$

This method does not have an underlying statistical model; hence we cannot retrieve any confidence intervals on our forecasts. Furthermore, this method only allows us to create a next-step prediction, meaning that the computed value will remain unchanged over our 28-day forecasting horizon.

ARIMA. An Auto-Regressive Integrated Moving Average (ARIMA) model explains the time series based on its own lags and the lagged forecast errors. An integrated series refers to a time series that needs to be differenced to behave stationary. Autoregressive terms of the model hold lags of the stationarized time series, and lags of the time series errors are called moving average terms. For a stationary series, the model is written as follows:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{3}$$

The right-hand side of the equation holds the predictors, including lagged values and lagged errors. This refers to an $ARIMA(p,d,q)$ model, where p equals to order of the autoregressive part, q to the order of the moving average part, and d to the minimum degree of differencing to obtain stationarity. The described model does not take seasonality into account. A seasonal ARIMA model most likely results in more accurate forecasts as we observe seasonal patterns when looking at the time series of averaged demands. Therefore we will add an additional seasonal part to the model, resulting in an $ARIMA(p,d,q)(P,D,Q)_m$, where m equals seasonal periods and (P,D,Q) holds the seasonal part to the model, consisting of similar terms than non-seasonal components, but involve backshifts of seasonal periods.

ETS. As the aggregated and average product demand shows clear seasonal and trend behavior, we also assess an exponential smoothing approach. We use the generalized ETS model, which accounts for our time series' error, trend, and seasonal components. The possible combinations of the terms, depending on the temporal behavior of our data, are Error $= A, M$, Trend $= N, A, A_d$ and Seasonal $= N, A, M$. We choose N to disregard the trend or seasonal component, A to describe an additive, A_d a dampened additive, and M a multiplicative behavior.

While our general model assumptions above hold, we have an additional requirement for the multiplicative error, namely that the data has to be strictly positive and high level. For illustration, we take the ETS(M,N,N) model, which underlies the SES method with multiplicative error and includes no trend nor seasonal component. The next step prediction y_t as in Equation **??** is composed of $\hat{l}_{t-1}$, the level of the data at $t-1$, and the multiplication of $(1+e_t)$, the estimation of the true, white-noise distributed error ϵ_t. At the same time, the error is defined for both positive and negative errors, if the error covers values below -1, the ETS model breaks. This is due to the multiplicative error ETS model, similarly as it is for its parent model SES, is only defined for positive values. As discussed in [32], this issue becomes apparent with low-level data where the variance can quickly escalate to big negative terms.

$$\begin{aligned} y_t &= \hat{l}_{t-1}(1+e_t) \\ \hat{l}_t &= \hat{l}_{t-1}(1+\hat{\alpha}e_t) \end{aligned} \tag{4}$$

Dynamic Harmonic Regression. As mentioned in Section 3, our day-to-day average demands show clear, fixed seasonality over a 365-day period. We, therefore, the reason that the individual products have to reflect this behavior individually, which is a suitable property of the time series for the dynamic regression with Fourier terms or dynamic harmonic regression. Here, the model is built on the principle that any periodic function can be approximated by a combination of sine and cosine [15]. Young et al. presented in [36] that the observed time series y_t can be expressed in terms of a low-frequency component (trend) T_t, a cyclical component C_t and a seasonal component S_t. Furthermore, the $f(\mathbf{u}_t)$ component captures the influence of exogenous variables (e.g., predictors influencing our target value). At the same time, the N_t is a noise component modeled as an *ARMA* process. Here we can also use the trend AR and MA components, p, q, and the seasonality-specific AR and MA components, P, Q. Finally, e_t models the discrete-time white noise we assume to be present in our recorded data. All the above is summarized in Equation 5

$$y_t = T_t + C_t + S_t + f(\mathbf{u}_t) + N_t + e_t \quad e_t \sim N\{0, \sigma^2\} \tag{5}$$

For the purposes of this study, we consider a model consisting of T_t, S_t, N_t, and the error e_t. Below, in Equation 6, we show how the K Fourier components allow us to add cos and sin pairs, which are respectively weighted by the stochastic, time variable parameters $a_{i,t}, b_{i,t}$. These stochasticity and time-dependency allow us to ignore the possible non-stationarity of the time series. Then, ω_i, t are the fundamental and harmonic frequencies associated with the seasonality. The number of K Fourier terms is limited to $m-1$ where m is the number of lags of the seasonality in the data.

$$S_t = \sum_{i=1}^{K} \{a_{i,t} \cos(\omega_i t) + b_{i,t} \sin(\omega_i t)\} \tag{6}$$

However, as we model our errors as an ARMA process, we note that all of the variables in the model are required to be stationary. Otherwise, the estimates of the coefficients will be inconsistent. This is because we cannot rely on differences to bring stationarity to our variables, as there is no notion of "removing" a variable to remove auto-correlation from the others. Furthermore, note that here, unlike in the direct ARMA or ARIMA modeling approach, we require stationarity in the variables rather than in the time series itself.

Generalized Linear Models. Presented by Nelder and Wedderburn in [25], Generalized Linear Models (GLMs) provide a more flexible approach to the parameter estimation of the linear coefficients in linear regression. GLMs share the assumption about the independence of y_i and y_j for $i, j \in S, i \neq j$ (S being the set of possible values) with linear models. However, the assumption of the target variable's underlying distribution is extended from a normal distribution to the entire family of exponential distributions. Furthermore, we don't assume that the mean itself is related to the linear predictor variables (i.e., $\mu_i = x^T \times \beta$ where x are our predictors and β our estimated linear coefficients) but any transformation of the mean $g(\mu) = x^T \times \beta$. Finally, we use Maximum Likelihood Estimation (MLE) as a generalized approach to estimate the linear coefficients.

While GLMs are not specifically envisioned for time series forecasting (see assumption 1 on the independence between target variables), they provide a flexible framework adaptable to the data that is being modeled.

Zero-inflated Poisson regression. respectively represent special configurations of GLMs specifically designed for count-based time series. As aforementioned, we can specify our data's underlying distribution when using GLMs. Hence, the discrete distributions such as the Poisson, negative binomial, and geometric distribution are well fitting for our case of count-based data [17]. For this study, we focus on the Poisson regression as it describes most of the product demands in our dataset best, based on the likelihood ratio tests (χ^2) we conducted to compare it with the negative binomial GLM.

Moreover, as we noted in Section 3, the value of demand counts with the highest frequency across all products is **zero**. Thus, the model of our choice needs to appropriately reflect both the intermittent behavior and the generally low level of counts and this high amount of zeros. We, therefore, choose to use the zero-inflated version of the Poisson regression. These two-component mixture models combine a point mass at zero with a count distribution. Hence, there are two sources of zeros. They may come from both the point mass and from the count component [37]. This allows us to give them more significance while having a specific component with a count-centered mass.

Vector Autoregressive Models. The main difference between all the models mentioned above and the autoregressive models is the direction. Where most, if not all, the above models are unidirectional, this vector autoregressions consider two directions. Where the forecast can be influenced by the predictor and vice-versa by allowing bi-directional modeling, dependent features such as price and demand can increase a forecast's accuracy as they affect each other both ways. By allowing for symmetry when modeling, vectorial models generate forecasts using a recursive relationship as summarized in the equation below with two features:

$$y_{1,t} = c_1 + \phi_{11,1} y_{1,t-1} + \phi_{12,1} y_{2,t-1} + \varepsilon_{1,t} \tag{12.1}$$
$$y_{2,t} = c_2 + \phi_{21,1} y_{1,t-1} + \phi_{22,1} y_{2,t-1} + \varepsilon_{2,t}, \tag{12.2}$$

Through this recursive relationship, VAR models can then test whether a variable is useful in forecasting another. Finally, the effect of sudden changes in one variable can be spotted in the auxiliary variable (Impulse response analysis cases). These properties make VAR models suitable candidates for potentially impulsive dependent variables such as price and demand.[2]

Prophet. Created by Facebook's Taylor and Letham [33], the prophet model's automated nature is a useful solution when dealing with a copious amount of time series. The model automatically selects the knots and identifies the seasonal component using Fourier terms, while holiday effects, for instance, are added as simple dummy variables. Therefore, the prophet model's strengths lie in its strong seasonal awareness and automated knotting, which is difficult to achieve, making this model a nonlinear regressive solution:

$$y_t = g(t) + s(t) + h(t) + \varepsilon_t, \tag{7}$$

XGBoost. The final model we assess for the time series forecasting task at hand is a gradient-boosted, tree-based regression referred to as eXtreme Gradient Boosting, or XGBoost presented in [4]. It is a model with high predictive power for both classification and regression tasks and is based on decision trees. While the basic application of decision trees is simple to implement, XGBoost leverages a model ensemble techniques referred to as *boosting*, which allows combining predictors from several trees. The model selection of the ensemble is driven by the gradient descent of a, in our case, squared error loss function $\mathcal{L}$ defined in Equation 8. While this approach has existed before, XGBoost is optimized for efficiency, parallelization, and ease of use on a wide range of different dataset structures.

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i \right)^2 \tag{8}$$

However, following the no-free-lunch (NFL) theorem [35], there is no *one* configuration of an algorithm that performs exceptionally well on all tasks. Hence, the complexity of this method lies in tuning the hyper-parameters such as the choice of the *booster*, or the learning parameter [26], as well as the number and maximum depth of the decision trees, to name a few. We're will be using H2O.ai [1] to implement XGBoost.

Model selection, fitting and forecasting

All methods presented above are either generally accepted baselines, have shown excellent performance on time series forecasting or are specifically designed for intermittent count data. For each method, we first fit a model for every product in our training data, forecast the demand for the next 28 days and then validate them on the available ground truth taken from the validation set. In our Results (Section **??**), we then identify the method which performed best across all product-specific models on performance metrics such as *MAE*, *ME*, and *RMSE*. This best-performing model will then also be used to predict the 28 days forecast horizon past the final date of the validation dataset to be submitted to the Kaggle competition.

As previously mentioned, several analyzed methods, namely ARIMA, ETS, dynamic harmonic regression, as well as GLMs, (zero-inflated) Poisson regression, and XGBoost, require specific model selection procedures. We will discuss the specific choices in the following.

ARIMA. For this method, we use the automatic parameter inference on the trend and seasonal AR, MA, and differencing parameters, as well as the drift.

ETS. Additionally to the automatic parameter inference provided by R implementation, we choose models using additive and additive dampened trends.

Dynamic harmonic regression. For this method, the number of Fourier terms K for a product-specific model needs to be found empirically by comparing the models with $K = 1, \ldots, (m-1)$ (m being the seasonal lag number) respectively. For the performance comparison, we use the corrected Akaike information criterion (AICc) [30], which penalizes the sum of squared Errors (SSE) with the number of parameters that need to be estimated [15]. The model with the lowest AICc value is then chosen as the best-fitting one.

(Zero-inflated) Poisson regression. As GLMs are statistical in nature, we fit for each product-specific time series a zero-inflated Poisson regression, a simple Poisson regression and a GLM predicting the mean. Using a likelihood-ratio test, we then test whether the zero-inflated Poisson regression is statistically significantly better than the other two models. If that is not the case, we then repeat the same comparison between the simple Poisson regression model and the GLM. Finally, we choose the model that results as the statistically best one out of this comparison.

XGBoost. Here, we start with a basic XGBoost setup and keep the default parameters (Table 3), training a model on each product's training set. After validation on the test set, we find an *average RMSE* of ≈ 2.304. Because the obtained scores are decent and we expect XGBoost to perform well on unseen data, we retrain the models on the full data for the products. With these full models, we generate predictions. As the predictions include negative numbers we apply a filter $max(0, i) \ \forall i \ \in F$ where F are our forecasts for a product and i the timesteps. We upload the results to Kaggle for an *RMSE* of ≈ 3.965. Not entirely satisfactory but in line with the results from our peers.

Table 3: XGBoost parameters

Parameter	Default	Grid options
learning rate	0.3	0.1, 0.3, 0.4
max depth	6	3, 5, 9
sample rate	1.0	0.8, 1.0
column sample rate	1.0	0.5, 1.0
number of trees	50	30, 50, 160

Following the NFL theorem as mentioned above, no hyperparameter tuning leads to mediocre results. Hence, we perform a grid search for each product-specific model on the training set and use the test set for model validation. Table 3 shows the hyperparameter grid we set. After every grid search, we define the product's leader model as the one with the lowest *RMSE*. Next, we take a look at the average *RMSE* obtained on the test set, which unsurprisingly went down. The new average *RMSE* is ≈ 1.844, a 20% improvement. We retrain all of the leader models on the full data set (train+test). These retrained models generate predictions that are filtered according to $max(0, i)\ \forall i \in F$ again. On Kaggle, they give us ≈ 3.528 *RMSE*, an improvement of about 11%.

Finally, we note that we limited our submissions in the Kaggle competition to $n \leq 10$ to prevent us and our methods from involuntarily overfitting or inferring the Kaggle test set.

5 RESULTS

Main Models

As discussed in the methodology, our strategy comprised the use of benchmarks before the investigation of more advanced and computationally greedy approaches. As an initial step, baseline models were tested. Namely Mean, Naïve, and seasonal Naïve methods before deploying more expensive models such as ARIMA, Zero-Inflated models, XGBoost, prophet, and autoregressive vectorial models. Hyperparameters used for the Naïve and mean methods were limited to specifying the desired outcome variable (demand), while more elaborate hyperparameters were used for the exponential smoothing methods (simple, seasonal, and damped). In fact, as outlined in the methodology section, ETS trends can either be additive, damped additive, or multiplicative. As the data at hand shows an additive trend, experiments were conducted using additive and damped additive trends, with seasonality and errors set to additive. On the more expensive side of our benchmark run is the use of dynamic harmonic regression, where experiments were run using 3 Fourier terms while setting the differentiating argument to zero. The trend AR and MA components (p, q) were set to test between ranges of 0 and 2. Finally, Facebook's prophet model [34] was tested by setting the periods to the number of events per year and the order to 10, as this was the required number for yearly seasonality, an important choice as each logged event occurs on a reoccurring yearly basis. The results from this benchmark run were later on compared with learning tree-based algorithms such as XGBoost and more generalized regressions such as the zero-inflated Poisson regression. From which the results can be seen in the below table:

Considering the RMSE in the above table, the most accurate models appear to be the ARIMA and Fourier models, however, once plotted, the lower RMSE for ARIMA seems to be a result of the stable trend towards the end of the series, a situation which causes the RMSE to lose balance.

From the above plot, it can be clearly seen that the ARIMA has the poorest fit while the Fourier and Damped forecasts follow the learned trend more closely despite achieving a lower or similar score to the ARIMA, a clear sign of overfitting. The risk of overfitting became clear during the Kaggle submissions as the ARIMA was outperformed by both the XgBoost and 3-term Fourier. The differences in fit are even more visually apparent once we stop using the mean and pick one item randomly and plotted the forecast in the below figure:

Vector Autoregressive models

As outlined in the methodology section, VARs can potentially be useful for demand forecasting which we performed during this competition. Specifically in investigating how a bi-directional relationship between demand and price can best capture the variance in the time series. Therefore, demand and selling price was specified as variables for the VAR model which we investigated using both the Bayesian and Akaike's information criteria.

Vector Autoregressive models

The above results table is also another interesting example of the dangers of overfitting and the use of non-robust accuracy metrics such as the ME which can be easily jeopardized by outliers. As can be seen through all metrics and especially the Mean error, both of these VAR models are a very good fit. However, a glance at the compounded demand8, random item demand forecast9, and Kaggle attempt a resulting RMSE score of 6.

The above two examples show demand was specified as the outcome variable using both Akaike's information criterion and the Bayesian information criterion. The volatility is also an example of how "Care should be taken when using the AICc as it tends to choose large numbers of lags." [2]

6 DISCUSSION

This section presents the limitations we have encountered during this project and also outlines avenues for future work.

Limitations

As with many statistical learning approaches, time and available computational resources are the main limitations to producing well-performing/well-tuned models. Given the time constraint for this specific research project, we have not experimented with more advanced approaches, such as deep learning. Furthermore, we have not been able to test how our models behave when earlier years (less relevant to our short forecast horizon) are left out entirely from the

Walmart Item price Forecasting - Forecast Accuracy comparison

metric	arima	ETSdamped	ETSsimple	fourier1	fourier2	fourier3	mean	naive	prophet	seasonal_naive	crosston	Zero_inflation_models	xgb
ME	-0.06962448	-0.1201958	-0.1200739	-0.07731163	-0.06131479	-0.05696459	0.1450737	-0.3505034	0.08283752	-0.1193016	0.221849	NA	NA
RMSE	1.70088579	1.7584314	1.7519480	1.70543255	1.70220942	1.70010472	1.9172519	2.2225259	1.79735719	1.7526235	1.900969	0.815226116896996	1.84407583169836
MAE	1.32259374	1.3833306	1.3780320	1.33293584	1.32738070	1.32376923	1.5226035	1.7573772	1.41378766	1.3783399	1.477119	NA	1.41047576827692

Figure 4: Count of unique demand values

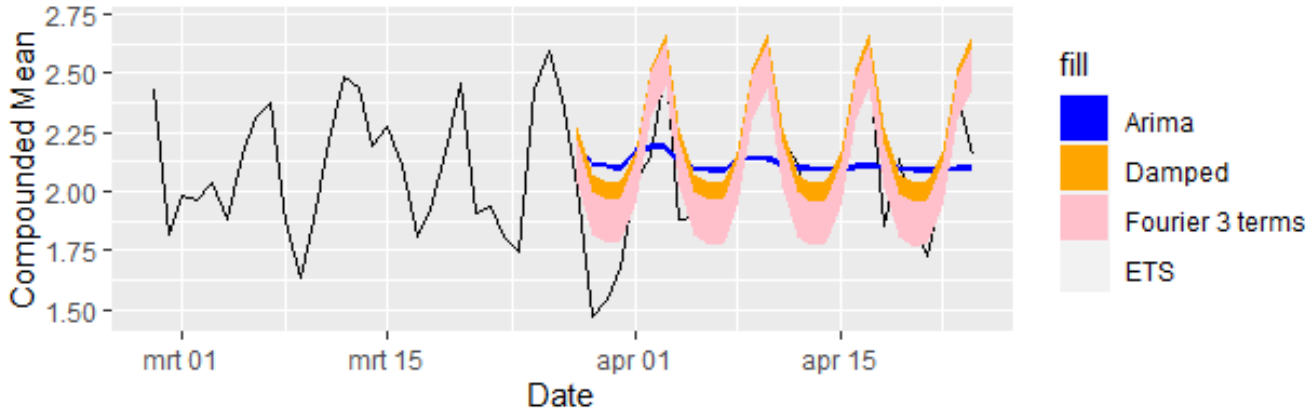

Figure 5: Compounded forecasted demand

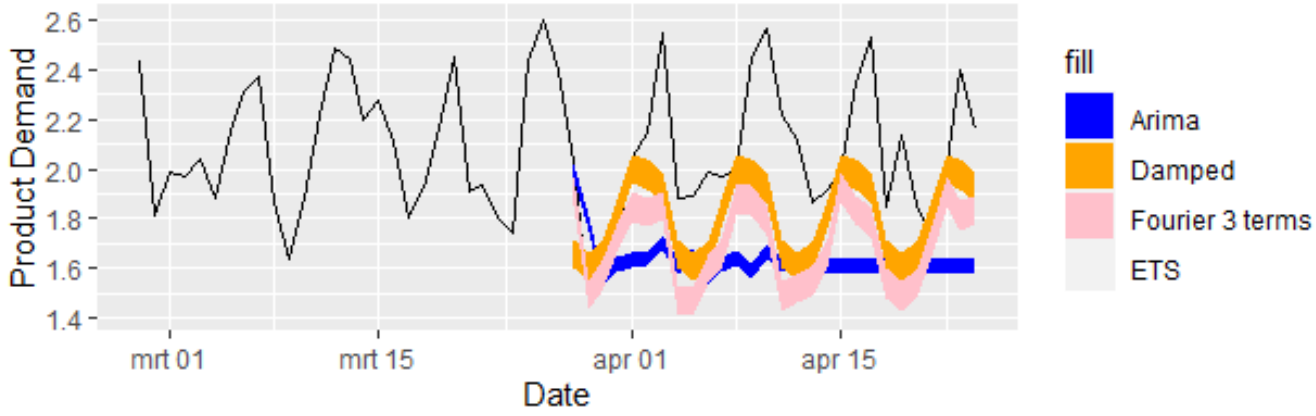

Figure 6: Forecasted demand for a random item

training data. While some models learned to disregard that data by the date, others did not have the power to do so.

Furthermore, we have not investigated the influence of the Kaggle competition leaderboard on our work and methods. We believe that seeking high scores on said leaderboard directly influences the process of model selection and hyperparameter tuning. Hence, we have not focused on improving the generalizability of our models. Connected to computing power, another limitation encountered during this project was the time constraint. After pre-processing steps and model setup, there was not enough time for fine-tuning

Walmart Item price Forecasting - Forecast Accuracy comparison -- VAR models

metric	var_aic	var_bic
ME	0.009381754	0.009381754
RMSE	1.810919653	1.810919653
MAE	1.416126653	1.416126653

Figure 7: Results Table for VAR

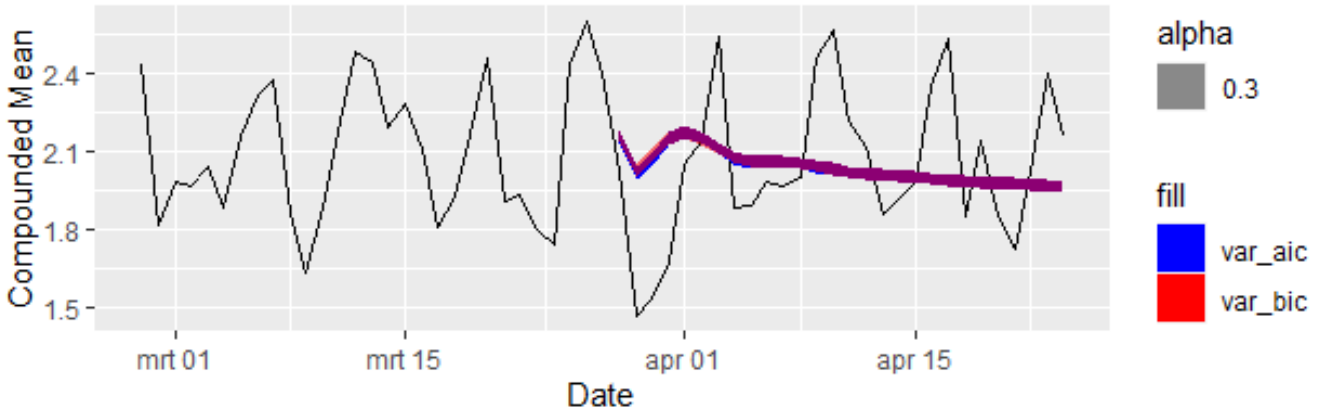

Figure 8: Compounded forecast

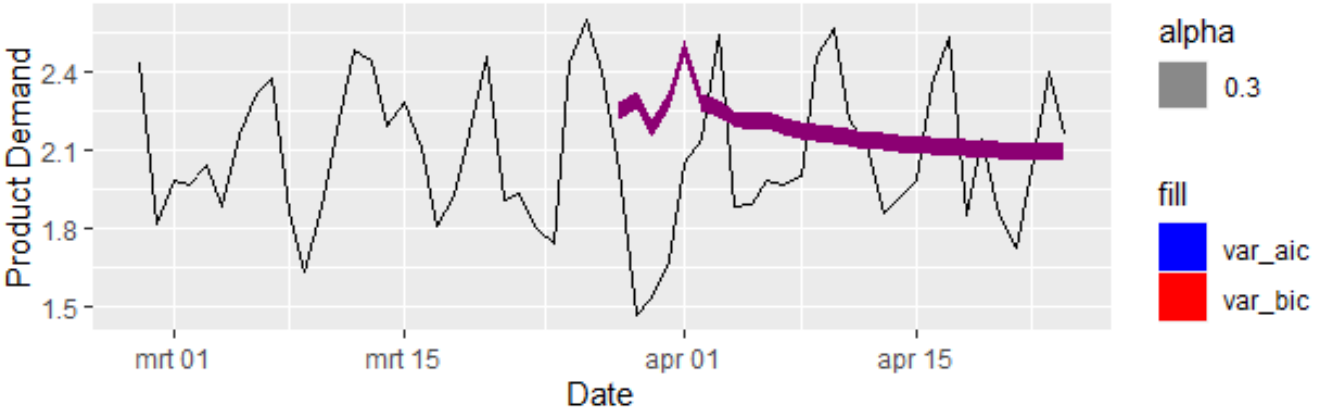

Figure 9: Random Item forecast

and adjustments. Performing accurate forecasts and building ML algorithms takes longer than two weeks.

Future work

The first suggestions for future work that come to mind are deep-learning approaches for count time-series forecasting methods, such as Long-Short Term Memory (LSTM). Most recently, the NeuralProphet method has been presented in [34] which promises explainable state-of-the-art results on time series forecast. Here Facebook's Prophet method has been extended with a dense, fully-connected Neural Network for the Autoregressive component allowing the method to capture the local context of the time series behavior. Furthermore, methods like DeepAR [31] and N-BEATS [28] can provide advanced, state-of-the-art ML implementations and show potential for further improving forecasting accuracy in hierarchical retail sales applications. Finally, to leverage the full capability of the models, we assessed in this report, Bayesian-Optimization-driven hyper-parameter tuning for XGBoost could have allowed for a better allocation of our available computational resources.

Furthermore, referring to the concern of model performance, generalizability, and over-fitting above, we could have deployed a fitting of each model separately for each product and chosen the best-performing one. Doing so, we believe that a powerful model ensemble could be built, further increasing our ranking, i.e., reducing the RMSE score on which the model is graded. In terms of generalizability, a future endeavor could be to train all the methods discussed on all products and use the product id as a predictor. This way, the data variance introduces a regularization effect that could allow the fitted model to generalize well on unseen data.

To challenge our results, it is also reasonable to apply our models to the whole dataset provided for the actual M5 competition, including other states. However, we would need to consider the hierarchical structure of the individual stores and modify the code accordingly. Additionally, forecasting could be further improved by using cross-validation (CV). An effective CV strategy can result in better forecasts by avoiding overfitting and mitigating uncertainty. Another future contribution would be to implement forecasting models to support decision-makers practically. It is crucial to focus on the extra costs of running expensive models rather than simpler ones and determine if their accuracy improvements justify much higher costs.

7 CONCLUSION

This report presents the results of Team 35 in the Kaggle competition *Sales Time Series Forecasting* on a subset of the M5 competition sales data. The objective was to forecast 28 days of demands for individual food items of one Walmart store in Texas. We focused on appropriately adding exogenous variables like calendar information and sell prices. This resulted in using two different datasets to maximize the performance models given the data. We discuss and implement various forecasting methods for time series data and present our model selection process and their performance on our test set and the Kaggle competition. The models applied were simple benchmark methods, more complex models like ARIMA and zero-inflation Poisson generalized linear models, and a gradient- and tree-based method referred to as XGBoost, which performed best in our experiments. Finally, we discuss our findings, their limitations, and possible avenues for future work.

REFERENCES

[1] 2020. H2O.ai | R Interface for H2O. R package version 3.30.0.6. https://github.com/h2oai/h2o-3

[2] George Athanasopoulos, D. S. Poskitt, and Farshid Vahid. 2012. Two Canonical VARMA Forms: Scalar Component Models Vis-à-Vis the Echelon Form. *Econometric Reviews* 31, 1 (2012), 60–83. https://doi.org/10.1080/07474938.2011.607088

[3] Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. 2019. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *International conference on neural information processing*. Springer, 462–474.

[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[5] John D Croston. 1972. Forecasting and stock control for intermittent demands. *Journal of the Operational Research Society* 23, 3 (1972), 289–303.

[6] Robert Fildes. 2020. Learning from forecasting competitions. *International Journal of Forecasting* 36, 1 (2020), 186–188.

[7] Robert Fildes, Shaohui Ma, and Stephan Kolassa. 2019. Retail forecasting: Research and practice. *International Journal of Forecasting* (2019).

[8] Chris Fry and Michael Brundage. 2019. The M4 forecasting competition-A practitioner's view. (2019).

[9] Inge Geyskens. 2018. Retailer power in the grocery industry. In *Handbook of research on retailing*. Edward Elgar Publishing.

[10] Adel A Ghobbar and Chris H Friend. 2003. Evaluation of forecasting methods for intermittent parts demand in the field of aviation: a predictive model. *Computers & Operations Research* 30, 14 (2003), 2097–2114.

[11] Michael Gilliland. 2020. The value added by machine learning approaches in forecasting. *International Journal of Forecasting* 36, 1 (2020), 161–166.

[12] Hansika Hewamalage, Pablo Montero-Manso, Christoph Bergmeir, and Rob J Hyndman. 2021. A Look at the Evaluation Setup of the M5 Forecasting Competition. *arXiv preprint arXiv:2108.03588* (2021).

[13] Tao Huang, Robert Fildes, and Didier Soopramanien. 2014. The value of competitive information in forecasting FMCG retail product sales and the variable selection problem. *European Journal of Operational Research* 237, 2 (2014), 738–748.

[14] Rob J Hyndman. 2020. A brief history of forecasting competitions. *International Journal of Forecasting* 36, 1 (2020), 7–14.

[15] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice*. OTexts.

[16] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.

[17] Tobias Liboschik, Konstantinos Fokianos, and Roland Fried. 2017. tscount: An R package for analysis of count time series following generalized linear models. *Journal of Statistical Software* 82, 1 (2017), 1–51.

[18] Shaohui Ma and Robert Fildes. 2017. A retail store SKU promotions optimization model for category multi-period profit maximization. *European Journal of Operational Research* 260, 2 (2017), 680–692.

[19] Shaohui Ma, Robert Fildes, and Tao Huang. 2016. Demand forecasting with high dimensional data: The case of SKU retail sales forecasting with intra-and inter-category promotional information. *European Journal of Operational Research* 249, 1 (2016), 245–257.

[20] Spyros Makridakis, Chris Fry, Fotios Petropoulos, and Evangelos Spiliotis. 2021. The future of forecasting competitions: Design attributes and principles. *arXiv preprint arXiv:2102.04879* (2021).

[21] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting* 36, 1 (2020), 54–74.

[22] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2021. The M5 competition: Background, organization, and implementation. *International Journal of Forecasting* (2021).

[23] S Makridakis, E Spiliotis, V Assimakopoulos, Z Chen, A Gaba, I Tsetlin, and RL Winkler. 2020. The M5 Uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting* (2020), 1–24.

[24] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. 2020. FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting* 36, 1 (2020), 86–92.

[25] John Ashworth Nelder and Robert WM Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)* 135, 3 (1972), 370–384.

[26] Andrew Y Ng. 2004. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*. 78.

[27] Dilek Önkal. 2020. M4 competition: What's next. *International Journal of Forecasting* 36, 1 (2020), 206–207.

[28] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437* (2019).

[29] Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K Barrow, Souhaib Ben Taieb, Christoph Bergmeir, Ricardo J Bessa, Jakub Bijak, John E Boylan, et al. 2020. Forecasting: theory and practice. *arXiv preprint arXiv:2012.03854* (2020).

[30] Yosiyuki Sakamoto, Makio Ishiguro, and Genshiro Kitagawa. 1986. Akaike information criterion statistics. *Dordrecht, The Netherlands: D. Reidel* 81, 10.5555 (1986), 26853.

[31] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[32] Ivan Svetunkov. 2021. Forecasting and Analytics with ADAM. OpenForecast. https://openforecast.org/adam/ (version: [current date]).

[33] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.

[34] Oskar Triebe, Hansika Hewamalage, Polina Pilyugina, Nikolay Laptev, Christoph Bergmeir, and Ram Rajagopal. 2021. NeuralProphet: Explainable Forecasting at Scale. arXiv:cs.LG/2111.15397

[35] David H Wolpert and William G Macready. 1997. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* 1, 1 (1997), 67–82.

[36] Peter C. Young, Diego J. Pedregal, and Wlodek Tych. 1999. Dynamic harmonic regression. *Journal of Forecasting* 18, 6 (1999), 369–394. https://doi.org/10.1002/(SICI)1099-131X(199911)18:6<369::AID-FOR748>3.0.CO;2-K arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291099-131X%28199911%2918%3A6%3C369%3A%3AAID-FOR748%3E3.0.CO%3B2-K

[37] Achim Zeileis, Christian Kleiber, and Simon Jackman. 2008. Regression models for count data in R. *Journal of statistical software* 27, 8 (2008), 1–25.

A APPENDIX A - FROM TEAM35 WITH LOVE

Figure 10: The members of Team 35 in their usual online call