M5 product sales forecasting
Team B6

Andrew Pizzuto (14564912), Matei Penca (14187639), Keane Mizzi (14624125) and Mattia Cintoli (14484560)
Applied Forecasting in Complex Systems 2022
University of Amsterdam

ABSTRACT

In our age, supply chain management has become one of the most profitable and vital parts of the retail industry. To be able to satisfy their customers, while also maximizing their profits, companies such as Walmart have a need for optimal forecasting algorithms that can predict the number of goods sold each day. Our study makes use of the M5 dataset to compare traditional forecast methods and popular tree-based machine-learning models. We have used average RMSE as our validation metric while also considering the training time needed for each algorithm. Our study showed that with little fine-tuning applied, traditional methods achieve an average of 1.8 RMSE, beating machine learning models such as LightGBM in both accuracy and time.

1 INTRODUCTION AND MOTIVATION

In the retail industry, it is important to have the appropriate amount of products available for customers to purchase. A reliable forecast helps ensure that popular products will be in stock, even if a customer visits the store at a busy time.

The overall purpose of this case study is to develop a machine learning model that can accurately forecast the number of units of each type of product that will be sold in a Walmart store located in Texas over the next 28 days. The utilized data comes from Walmart's M5 Forecasting Accuracy competition [1] and includes details about specific items and product categories, as well as variables such as price, promotions, and special events.

Research Question and Outcomes

Our data set includes 823 diverse products with varied sales patterns. Therefore, training a single general model would not be effective in accurately predicting the sales of each individual product. On the other hand, most forecasting models require some level of fine-tuning to work or to produce optimal results. In our case, this approach is unfeasible because of the high number of different models that we would have to manually fine-tune. In recent years, tree-based machine learning models have gained popularity in the forecasting field due to their ability to handle complex data and, in some cases, outperform traditional models [1]. These algorithms work differently than traditional models by constructing a tree-like model to make predictions [2].

As a result, this paper focuses on comparing simple traditional forecasting techniques to more advanced machine learning based techniques which do not require too much manual fine-tuning as we will build them for each product. In conclusion, the research question we aim to answer is:

Without any advanced fine-tuning, are tree-based machine learning techniques better than traditional forecasting techniques?

By answering this question, we plan to learn which of these techniques can serve as a good baseline considering that building a complex model for each product is an intensive, and possibly unneeded task. Throughout the paper, we will be comparing both the accuracy of the models, but also the effort and time needed for the training.

Paper Outline

Section 2 goes over our exploratory data analysis on the possible predictor variables from the input files, while Section 3 does the same for the target variable, the daily sales. Section 4 explains how we transformed our data for the forecasting, while section 6 explains what models we have used in our analysis, and how these were implemented. Section 7 goes over the results of our comparison, while Section 8 contains our conclusions, discussion and possible future work that can be done as a follow-up.

2 PREDICTOR VARIABLES EDA

The data consists of four files. File 1 contains information about the dates the products are sold. File 2 contains information about the price of the products sold for each date. Files 3 and 4 contain the historical daily unit sales data for each product, with file 3 being used for training and file 4 being used for validation. The goal of the EDA is to analyze and gain insights on the predictor variables and information from past observations of the time series. Combining the first three files, three possible predictor variables are considered: Events, SNAP, and Prices.

[1]https://www.kaggle.com/c/m5-forecasting-accuracy

Events

There are in total 30 different events that occur during a year. These 30 events are grouped into four event types: cultural, national, religious, and sporting. The first question that arises is: will the events have an impact on sales? To answer this question, sales were grouped by year and cultural event, and the daily mean of sales was taken for comparison.

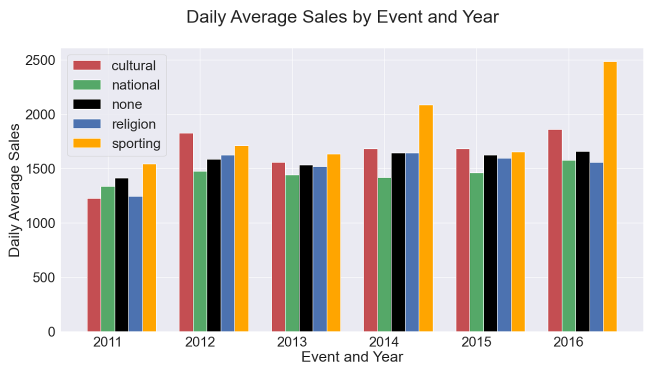

Figure 1: Daily Average Sales by Event and Year

We can observe from Figure 1 that the average sales when national events take place are lower compared to days with no events, while, during sporting events, sales are usually higher. There could also be a lag between the day of the event and the purchase made for that particular event, for example, one might buy food products the day before an event, not necessarily the same day. Indeed, from Figure 2 the average sales are higher for cultural, national, and sporting events.

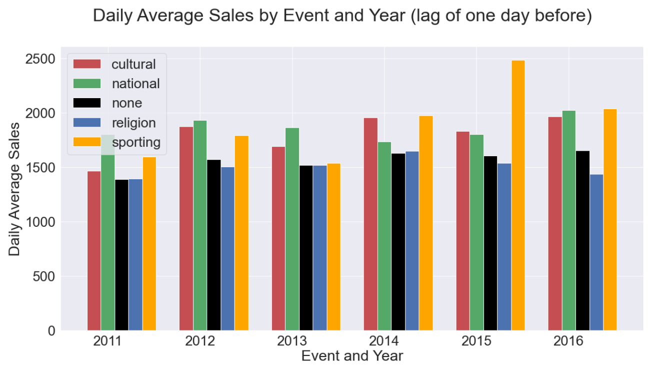

Figure 2: Daily Average Sales by Event and Year (lag of one day before)

Based on the data, there are only three sporting events each year, and two of them (the NBA finals start and end) have yet to occur. Of the 28 days that need to be forecasted, four days have events, and two of those events are the NBA finals start and end. Therefore, including these event variables in our modeling process may improve the accuracy of our predictions.

The third event of the 28 forecasting days is Ramadan. Since we are forecasting food products, it is worth investigating whether Ramadan has a negative impact on food sales. The intuition behind this hypothesis is simple: Muslims or people who practice Islam will eat less during the religious event and therefore be more likely to purchase less food. By looking at sales, the day of Ramadan (Figure 3), three days before, and three days after, there is no clear and significant difference between before and after Ramadan.

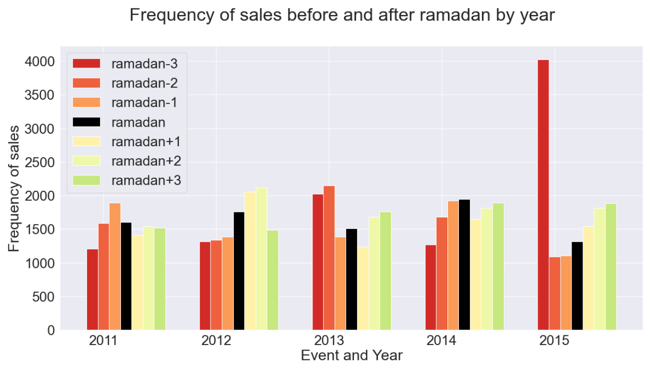

Figure 3: Frequency of sales before and after Ramadan by year

SNAP

SNAP, also known as the Supplemental Nutrition Assistance Program, is a government program that provides financial assistance to low-income individuals and families in the form of benefits that can be used to purchase food[2].
Walmart allows SNAP purchases only on certain days. From the plots (Figure 4), snap frequency is constant throughout the years (note that 2011 starts from the 29th of January, that's why in 2011 there are 110 days instead of 120, and there is only partial data for 2016). Moreover, days with snap don't happen randomly, there is a trend, however, it will not be relevant in our analysis since we already have the snap data for the days we need to forecast.
Finally, sales are higher with snap (1699 with snap, 1502 without), therefore, including a snap dummy variable in our modeling process may improve the accuracy of our predictions.

[2]https://www.fns.usda.gov/snap/supplemental-nutrition-assistance-program

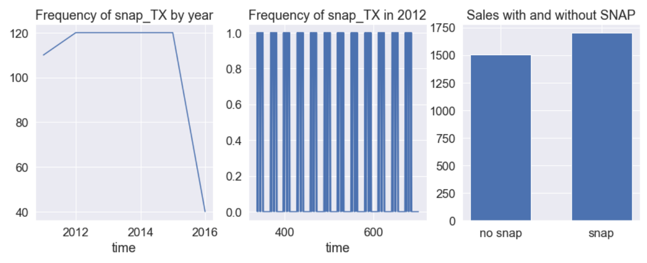

Figure 4: SNAP

Prices

Analysing how price changes throughout the whole dataset would be counter productive and plotting figures for so many products would not give us any opportunity of understanding anything relevant from these kinds of plots. Moreover, aggregating prices together would be hard since many products substantially differ in price (prices range from 20 cents to 19 euros). Therefore, to better understand the structure of our data and gain insight into the overall dataset, we first divided the data by year. This allowed us to investigate the data on a yearly basis and obtain figures that could help us better understand the data. For example, Figure 5 displays the total average selling price of items per year.

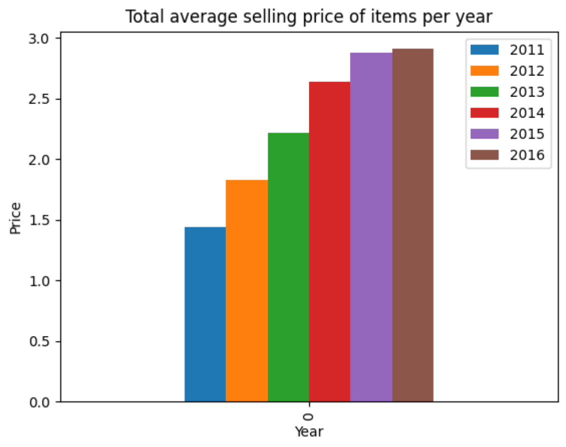

Figure 5: Total average selling price of items per year

Two groups of specific products were analyzed to see how the selling price differed throughout the years. These groups were 1) the eight most sold products and 2) most sold product, the one at the 75th percentile, the median, the one at the 25th percentile, and the one with the minimum number of sales. Specifically, looking at group number 1 a line chart was plotted to analyse the eight most sold products (Figure 6). One reason for this was to answer questions about the sales history of these products, including when they first went on sale and whether they have regularly lower prices, in order to understand why these products have been the most popular over the years. Moreover, we check for any important characteristics such as trend and seasonality.

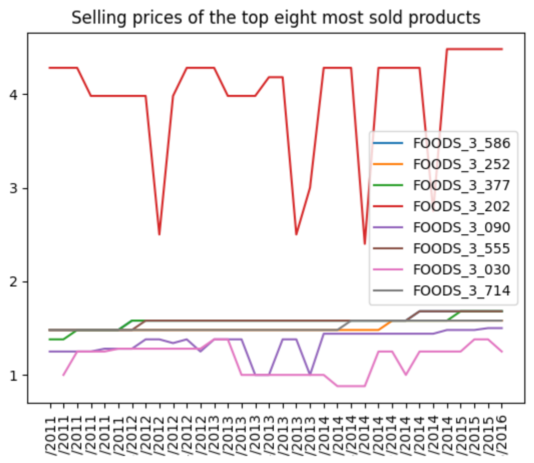

Figure 6: Selling prices of the top eight most sold products

On the other hand, for group number 2, once again a line plot is displayed to show how the different products from each end of the spectrum are different or have similar features to each other. Once more, we try to take note of any seasonality or trend noticed in these plots. Figure 7 shows how the different products started to sell at different time periods which is the main factor as to why some of them sold more than others. Even though this is the case we still see price changes in three of the five products even though they started being sold at a later stage.

Interpreting both of these graphs reveals that the prices of some products dropped slightly or significantly towards the end of 2013 and at the beginning of 2014, but there is not enough evidence to conclude that the selling price of these particular products follows any seasonal or trend-based pattern when examined year by year.

3 TIME SERIES EDA

In this section, we will conduct an exploratory analysis of our target variable, daily sales, using time series techniques. By examining the characteristics and patterns of the sales data over time, we can gain insights into the underlying trends and dependencies that may be present. This will provide

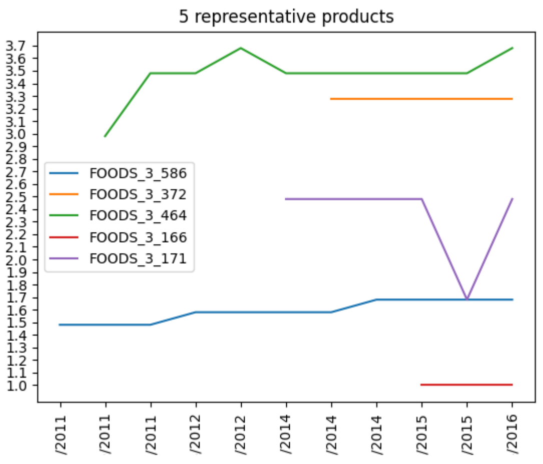

Figure 7: 5 representative products

a foundation for more advanced modeling and forecasting approaches in the future. Since there are 823 products and analyzing every single one of them would be unnecessarily long and non-informative, the analysis was conducted on the sum of the products. Plotting the time series, we can already observe a trend and strong seasonality ("Figure 8"). This is further validated by looking at the STL decomposition (Figure 9): the trend is increasing and there is both yearly and weekly seasonality.

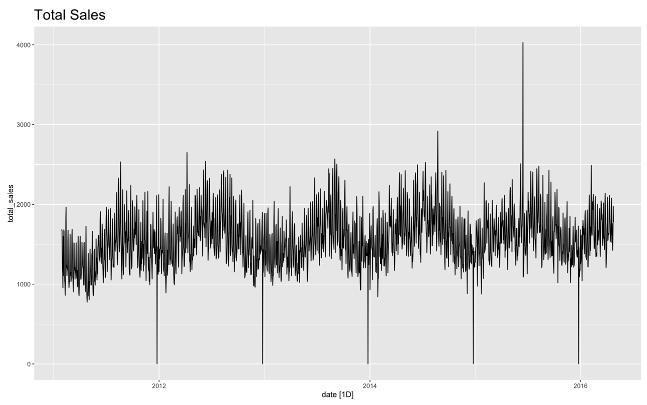

Figure 8: Total Sales

An STL of the box-cox transformed series was also performed but the results were almost identical to the non-transformed series. From the ACF (Figure 10), we once again corroborate our previous findings. First of all, the decreasing lags mean that indeed there is a trend, moreover, the highest spikes are at lags that are multiples of 7, clearly showing daily seasonality. Further, it is interesting to note that high

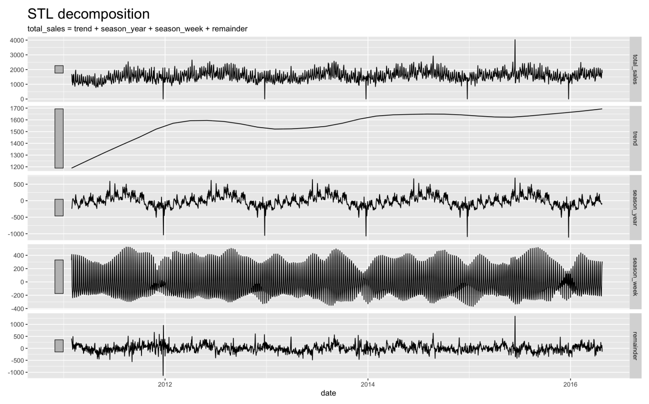

Figure 9: STL decomposition

spikes are also at lags 6 and 8. This means, for example, that sales on a Friday are not only correlated to sales of previous Fridays but also to sales one day before (Thursdays) and one day after (Saturdays). Figure 11 shows the mean sales for each day of the week and how the sales change for each day of the week in time. Sundays and Saturdays have the highest average sales, followed by Mondays and Fridays. Tuesdays, Wednesdays, and Thursdays have the lowest number of sales.

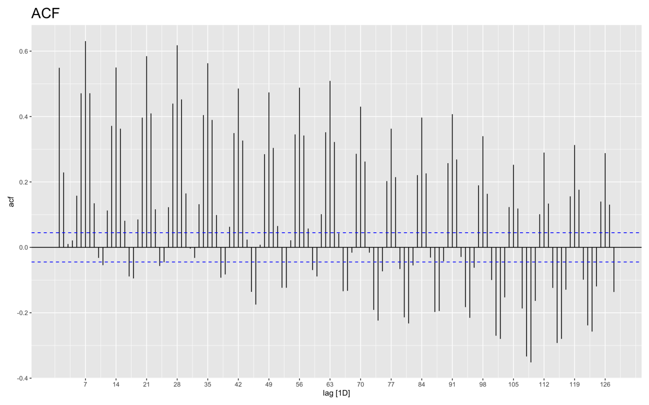

Figure 10: ACF

To provide a more comprehensive view, we displayed the time series and weekly plots (Figure 12, Figure 13) of selected individual products based on their number of sales. These products include the one with the maximum number of sales, the one at the 75th percentile, the median, the one at the 25th percentile, and the one with the minimum number of sales. The plots exhibit a range of trends, seasonality, and average sales. Some of the plots start in 2011 while others start at a later date. Additionally, some of the plots have numerous zero values while others do not.

In conclusion, on average the sales of all products actually increase over time, however, there are some products that do

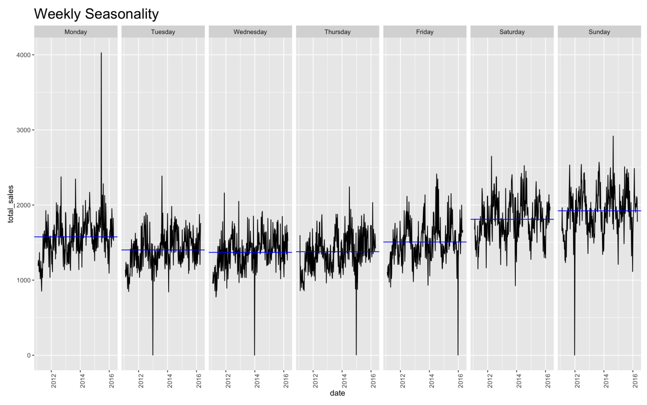

Figure 11: Weekly Seasonality

not show a significant trend. In addition, many products do have both weekly and monthly seasonality, however, each has its own one. Due to these differences, each product will be modeled individually to ensure an accurate representation of their sales trends and patterns.

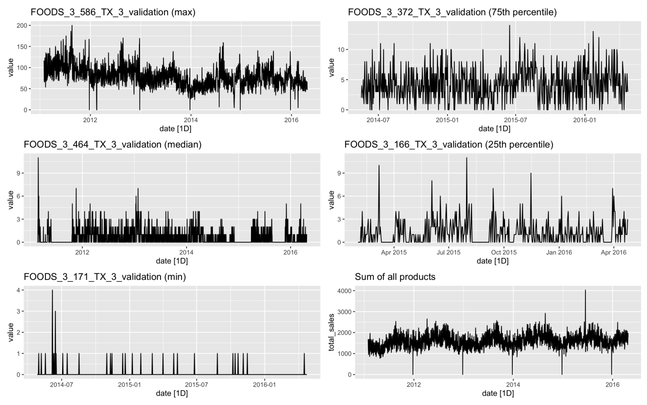

Figure 12: Total Sales 5 products

4 DATA PRE-PROCESSING

This Section goes in-depth with all the pre-processing techniques applied throughout the exploration phase of the project. These were needed to be able to work smoothly with our data and extract as much information as possible. All the modifications that will be explained below have been implemented in Python, where we took advantage of the powerful library called Pandas [3] which can handle tabular data well. It should be noted that these transformations do not include the data pipelines specific to each forecasting method, as those will be explained in the forecasting sections.

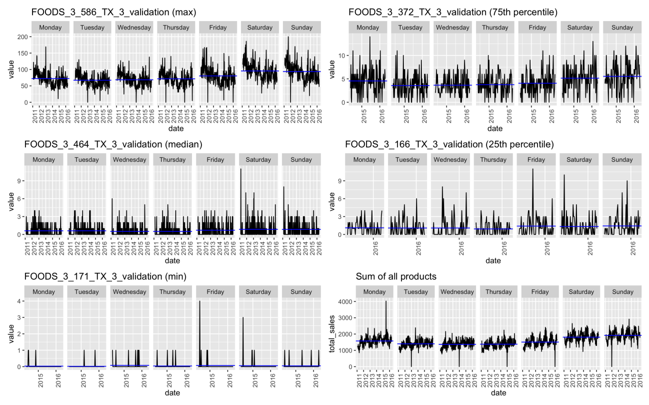

Figure 13: Weekly Seasonality 5 products

Changing the structure of the input

The data was initially in a wide format, meaning that each date was represented as a column rather than a row entry. Because most forecasting packages and models in R expect the date component of our data as a row, we have decided to transform our data into a long format as seen in Figure 14. This operation was achieved by using the melt function in pandas which transformed the columns into rows based on the product identification number.

Moreover, we decided it would be better to have the date

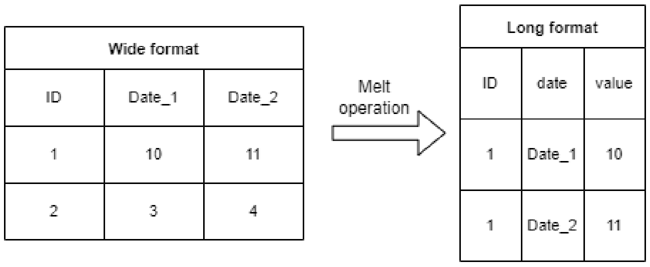

Figure 14: Illustration of melting from wide to long dataframe

column contain an actual date rather than the day number. We knew that the first day in our training set is January 28th 2011, so using Pandas and the internal Python libraries we transformed all integers in the date column into actual date objects. Furthermore, we also changed the date format under the calendar data to match the one in the sales one.

Combining the data sets

As mentioned in Section 2, the data was split into multiple files that each contain possibly important information that could help the forecasting models. We decided to concatenate all of this information into 1 data set that could be more easily imported between environments. To achieve this we have first merged the calendar data with the price data as both of them shared a column on which we joined them. After that, merging with the sales training data was easy as our previous transformation of the date meant we now had the same format in both datasets.

It is important to note that by merging all of the files, we have indirectly removed the entries for the products that had no price on that day. This was a beneficial outcome as having entire periods for a product with 0 sales would not help the model in any way. Moreover, this reduced the final size of our data set to around 1.2 million rows composed of 823 products. The new data set has been saved as a comma-separated value (CSV) file which can be easily imported into R for the forecasting models.

5 BACKGROUND LITERATURE AND DESIGN CHOICES

This chapter provides background information about the core forecasting methods used and the reasoning behind our choices. As mentioned in Sections 1 and 4 our data contains 823 different products and our final goal is to predict for each one of them an interval of 28 days. Because it is not feasible to manually tweak a model for each product, we have decided to use models which can work relatively well without fine-tuning.

Traditional forecasting methods

One of the simplest forecasting methods that can be implemented as a baseline is the Naive method. This method assumes that the predicted values should be equal to the most recent ones. As a result, this method is easy to implement, but in general, will not generate good results. Because in our exploratory data analysis we have seen that most product seems to have some seasonality, we decided to use the Seasonal Naive method, which can account for it.

Next, we used a random walk method with drift. In this type of method, the predicted value is influenced by both a random error, but also by a constant drift term. The reason we tested this model was to see if it would do better as a baseline compared to the Naive method [4].

For our final traditional statistic method, we chose Exponential Smoothing model [5]. This model uses weighted averages of past observations to create future predictions. We utilized the Holt-Winters seasonal method which allows us to also account for possible trends or seasonality in the data. We also tried to apply the ETS model to the seasonality-adjusted data to see how that influences the final results.

While all the methods above use past observations to make new predictions, our dataset contained extra predictors that, as shown in Section 2, might hold useful information. In terms of the traditional methods, dynamic regression ARIMA can handle these types of data. On the other hand, ARIMA is known not to be able to handle daily seasonality well [6]. Moreover, for this type of model, we would have to check if we need to differentiate the data (to make it stationary) and possibly tweak the parameters. Because we decided to build one model per product, we have decided to skip ARIMA in our case study.

We also attempted to use Harmonic Regression for some of our products [7]. While we obtained good results for some of the products, we encountered errors for others. We suspect that these errors may have been due to insufficient data, so we decided to explore other models as an alternative.

Tree-based Machine Learning models

As mentioned in Section 1, tree-based models have become very popular in the forecasting field, as they have scored high in many competitions over the last few years, beating many well-known forecasting techniques.

In this paper, we have focused on two popular variants, starting with Extreme Gradient Boosting (XGBoost). This algorithm creates multiple weak tree-based models and combines their predictions in order to create a much more accurate forecast. Different implementations of this algorithm can take advantage of advanced machine learning concepts such as gradient boosting, learning rates, and parallelism, but this also means that a more advanced level of fine-tuning is needed [8]. In our study, we have tried to keep the hyper-parameters tuning to the bare minimum to see how well it does compared to the traditional methods.

Our second option was Light Gradient Boosting Machine (LightGBM), which is fundamentally similar to XGBoost, but it builds the weak trees in a different, more efficient way [9]. This enables the algorithm to generally train faster, and use less memory without sacrificing any accuracy. Both algorithms are powerful and widely used, and determining which one is better depends on the data.

6 STUDY EXECUTION

This chapter will go in-depth with how we have implemented all the algorithms described in Section 5.

General pipeline for forecasting

For the forecasting part of our study, we have decided to use the R programming language as it has a lot of built-in libraries for this task. Because we have already pre-processed our data as described in Section 4, we were able to easily import it into R as one data set. Furthermore, we have decided to transform our data into a `tsibble` time series object. Because we have to build a model for each product, we have created a for-loop that would filter the timeseries entries per product. The next step was fitting a model on the product, generating a forecast of the next 28 days and validate the model on the test data set provided to us.

The metric of choice for this research is the root mean squared error (RMSE) which denotes the average of the squared differences between the predicted values and the actual values. This metric is also a good fit for our research questions as it allows us to compare different models based on its values [10]. The key aspect is that our pipeline would generate a separate RMSE for each model-product combination which would make it hard to compare. As a result, we have decided to take the mean value of all the metrics per model.

A popular technique that can improve forecasting accuracy is to apply different transformations to the initial timeseries. In our case, we have tried different simple log transformations, and even a Box-Cox transformation [4] (as all of our data is positive), but in almost all cases there was no good transformation for the type of data we had. Thus we have decided to not apply any transformations without manually picking the best one for each product.

Traditional forecasting methods pipeline

For the simple traditional forecasting methods, we only used the past values of the stock for each product as input data for future predictions. This meant no extra data pre-processing was needed.

Because our data was represented as a `tsibble` we were able to use the well-known `forecast` package for all the traditional forecasting methods. Fitting the Naive, and Random Walk with Drift has been straightforward as no fine-tuning is needed or possible. The models also ran very fast as no complex operations are needed to generate the forecasts.

For the Exponential Smoothing model, we had to decide whether we should incorporate any trend or seasonality. From our EDA, we discovered that on average products do have a trend (although many single products do not have a trend), and weekly, or sometimes even monthly seasonality existed. Adding seasonality to the model using the `ETS` R function was not hard, but different combinations have been tested using additive and multiplicative seasonalities. Because this model is still light-weight we were able to apply some fine-tuning in order to get the best performance.

Machine learning methods pipeline

The tree-based machine learning models such as XGBoost and LightGBM can use predictor values to improve the accuracy of the predictions. Our intuition was that extra information such as the price of the product, and the eventuality of a sporting event, or a SNAP event on a specific day would tune the predicted values. On the other hand, this meant more pre-processing was needed in contrast with the simpler methods. To simplify the process and automate it, we have used the `recipes` and `modeltime` packages in R which helped us create data and training workflows.

The `recipes` [3] library was used to split the data into the predictors and outcome variables, and create new variables based on the timeseries aspect. This library allows the creation of new variables based on lags, rolling means or various time series aspects such as day, week, and month of the year which help capture seasonality. There are many combinations of new predictors that can be created, but as our time has been limited, and the training of models took a lot of time we decided to only focus on new lagged predictors and derived variables based on the timeseries aspects.

For the new lagged predictors, we have chosen to look at the lagged values for the sold products on 7, 14, and 21 days in the past based on the `ACF` plots. Another combination included also the days before and after as those seemed to be correlated too. The derived variables based on the timeseries aspects were automatically chosen by the `timeseries-signature` R function and reflected changes in sales per year, quarter, month, or week.

Lastly, the fitting of the models was done through `modeltime` library [4] which allowed us to create a general workflow for loading the recipe to be applied on the data and train regardless of the model. To compute the RMSE and average RMSE, we have used the `Metrics` R package on the predicted values and ground truth.

7 RESULTS

This section will go over the results achieved in our experiments on the traditional and tree-based machine learning models. Moreover, we also do some simple, but effective model diagnoses to make sure we can trust our predictions.

Evaluation Metrics

As mentioned in Section 6, for this project we used RMSE to measure the goodness of the fit for the models. Moreover, on top of accuracy, our team decided to also evaluate based

[3]https://recipes.tidymodels.org/
[4]https://business-science.github.io/modeltime/

Model	Accuracy (RMSE)	Running time (s)
ETS(A,N,N)	**1.88**	**448**
ETS(A,N,A)	1.89	441
ETS(A,A,A)	1.90	450
STL-ETS	2.17	498
SNAIVE	2.31	436
Random Walk w/ Drift	2.35	430

Table 1: Results on validation set for traditional forecasting methods.

Model	Accuracy (RMSE)	Running time (s)
LightGBM (auto)	**1.92**	**896**
LightGBM (with lags)	2.1	876
XGBoost (auto)	2.54	973
XGBoost (with lags)	2.59	987

Table 2: Results on validation set for tree-based machine learning methods.

on the time needed to train each model as we expected the machine learning models to achieve better results, but take longer times.

In Table 1 we present the results achieved by each traditional model. For the exponential smoothing algorithm, we have included the results achieved by each variant. The best-performing model in this category is the exponential smoothing model with no trend and no seasonality. The other variants of `ETS` were not far behind, but no improvement could be seen by adding extra information. Furthermore, the random walk and naive model took the last places, but still achieved a decent result. In terms of time, there is no substantial difference between the models as they are all fairly light computation wise.

Table 2 contains the evaluation metrics for the tree-based machine learning models. The LightGBM model performed the best when paired with the automated timeseries predictor extraction augmented automatically by the `recipes` R package. Adding lagged information about the sales made the algorithm run for longer and have less accurate results. This could be mainly because the data has complex seasonality both weekly and monthly which is hard to be captured by the limited amount of lags we have tried. Better combinations of lagged predictors should in theory improve the results. Moreover, XGBoost had worse results for both variants of predictors. The increased time and worse accuracy is probably an outcome of how the model works when it comes to training, specifically the way the weak trees are built to generate the forest.

Model diagnosis

Diagnosing the models to see if we can trust their predictions was harder when having so many different models for each product. As a result, we have decided to pick randomly some plots and check to see if their residuals are white noise or not. To achieve this, we have used both ACF plots and also Portmanteau tests. For the traditional models, using the Box Pierce implementation in R we validated that most products have models for which the residuals are not white noise. Figure 15 and 16 show the `ACF` plots for 2 models for which the Box Pierce p-value was large enough to conclude against white noise.

On the other hand, we also have models for which the white noise test failed with a p-value smaller than 0.05, and inspecting the ACF we can see many spikes that signal correlation in the residuals. This shows that not all products have models that are perfect, and improvements could still be made to make these models better. One such example can be seen in Figure 17.

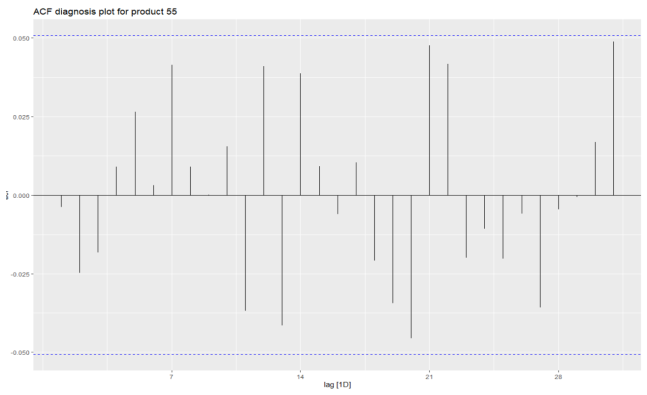

Figure 15: ACF plot for the ETS model of product 55

8 CONCLUSION

Throughout our exploratory study, we compared the accuracy of traditional forecasting methods with tree-based machine learning algorithms. This section summarizes our work while also presenting our thoughts on the main findings, limitations, and possible follow-up research.

Summary of Main Contributions

Our project task was to compute forecasts for a period of 28 days, starting with the 25th of April 2016. Our group decided to generate an individual model for each product

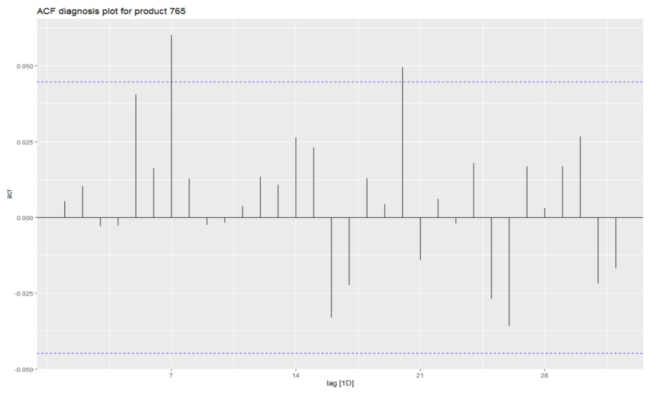

Figure 16: ACF plot for the ETS model of product 765

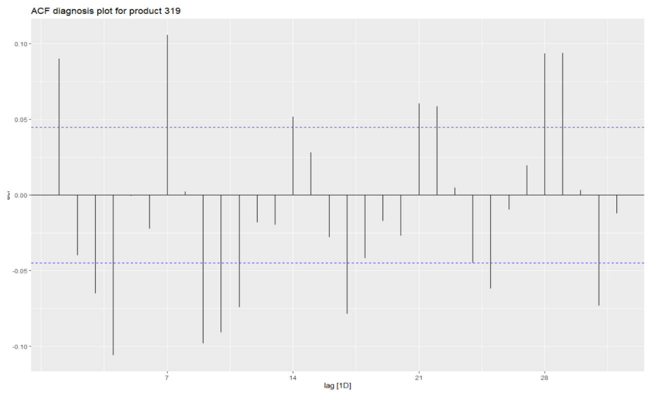

Figure 17: ACF plot for the ETS model of product 319

in the hopes that would allow us to capture the trends per instance better. This decision has been made as a result of our extensive EDA where we saw that each product sold in different quantities with unique patterns. Moreover, because fine-tuning each model was almost impossible to do, we have decided to focus our paper on comparing traditional forecasting methods with machine learning ones with little to no fine-tuning. For the latter models, we have also supplied them with additional predictor values such as calendar, SNAP, and price information. For the evaluation of the models, we have used average RMSE over all models, while also recording the training times needed for each algorithm. To diagnose the models, we have used statistical and visual tests to check if the residuals are correlated or white noise.

Discussion of Main Findings

In Section 7 we presented the accuracy and time scores for all the models we have trained and used for forecasting. The traditional forecasting methods performed better than the more advanced machine learning counterparts. While the difference between the best models of each category is not much, the time it took to train and forecast the models could be another reason why one would prefer the simpler models.

On the other hand, while the traditional models are meant to be run without much fine-tuning, the same cannot be said about the tree-based models. In our case, we programmed these to run with the default setting while we did the minimum amount of predictor search. We believe that with more data pre-processing and with better predictors such as more fine-tuned lagged variables and rolling means, it is possible to achieve a better average RMSE than 1.8 using the tree-based methods.

That said, doing this process for 823 different models is a time-consuming task, which is not always possible. In that case, having a sub-optimal model that can still produce good results in a reasonable time could be more valuable for a company. Another factor regarding our predictions is how relevant or useful they are for a company. Looking at our forecasts, most of them are float variables that are not useful from a business point of view. Generating the forecasts as a confidence interval with minimal and maximal bounds for the sales of a product would be more useful for supply chain management operations.

In conclusion, we were able to showcase that simpler traditional forecasting methods can still achieve decent results with short training times and no extended knowledge needed for fine-tuning. Moreover, we also built a framework that can process the data into a format that is easily accessible to machine learning based algorithms. We have shown that tree-based models can be good for forecasting, but a reasonable amount of fine-tuning is needed in order to get optimal results.

Limitation and Future Work

One of the biggest limitations of our study was not having enough time to train and test the models in such a way that we could achieve the best results possible. As a result, we have had to cut corners by limiting the fine-tuning to basic levels. Machine learning models are known to be very sensible to fine-tuning meaning that they can produce considerably better results with the right parameters in contrast to traditional methods which cannot be improved much more.

Moreover, as mentioned in Section 5, initially we also wanted to try using well-known traditional forecasting methods such as dynamic regression ARIMA or harmonic regression, but because we lacked experience and time, we could not get them to work correctly for all products meaning we had to drop them for the purpose of our research. These methods should, in theory, work well with the extra predictors we have, and also with the nature of our timeseries so it would be interesting as a follow-up research.

Another idea that we had, but was dropped because of limited time was to group the projects based on their trend, similar seasonality or average amount of products sold. With this approach, we could then train a model for each category instead of one model for each product. This would allow us to fine-tune the models while not generalizing too much. We believe this would be a better approach than having separate models as that can become unfeasible if the number of products increases over time. Additionally, we would have liked to also build a single model trained on all products at once that could also predict per class. While models such as LightGBM can do such a task and programming it is not too hard, we did not have enough computational resources for it. Doing such a task means all the training data needs to be loaded into the model which is very expensive.

Finally, we would have liked to be able to spend more time on the model diagnosis and try to understand why some products can be forecasted better than others. This would give us more insights into how we could then fine-tune the models to avoid these issues. This would be a good follow-up research as it would allow more specific work to be done to decrease the RMSE.

REFERENCES

[1] Tim Januschowski, Yuyang Wang, Kari Torkkola, Timo Erkkilä, Hilaf Hasson, and Jan Gasthaus. Forecasting with trees. *International Journal of Forecasting*, 38(4):1473–1481, 2022. Special Issue: M5 competition.

[2] Hongfang Lu and Xin Ma. Hybrid decision tree-based machine learning models for short-term water quality prediction. *Chemosphere*, 249:126169, 2020.

[3] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[4] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[5] Tugba Memisoglu Baykal, H. Ebru Colak, and Cebrail Kılınc. Forecasting future climate boundary maps (2021–2060) using exponential smoothing method and gis. *Science of The Total Environment*, 848:157633, 2022.

[6] Jamal Fattah, Latifa Ezzine, Zineb Aman, Haj Moussami, and Abdeslam Lachhab. Forecasting of demand using arima model. *International Journal of Engineering Business Management*, 10:184797901880867, 10 2018.

[7] Michael Artis, Jose Clavel, Mathias HOFFMANN, and Dilip Nachane. Harmonic regression models: A comparative review with applications. *Institute for Empirical Research in Economics - IEW, IEW - Working Papers*, 09 2007.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[10] Spyros Makridakis, Steven Wheelwright, and Rob J Hyndman. Forecasting: Methods and applications, 3rd ed. 1997.