Rags to Riches: A Journey from Traditional Forecasting to Deep NeuralNets

Nolan Cardozo s1034065
Ajinkya Indulkar s1034517
Rodrigo Perez Victoria s1034694

June 2020

1 Introduction

This report discusses our work in the M5 Forecasting challenge. The fifth iteration of the Makridakis competitions, the aim of this challenge is to find forecasting solutions using the Walmart hierarchical sales data. The overall goal includes the use of traditional, machine learning or deep learning methods which can predict the accuracy as well as uncertainty of future sales. Hence, the competition is presented in 2 parts and our work focuses on the accuracy of sales forecasting.

2 Approach

We explored several methods which include traditional and automated forecasting methods, gradient boosting techniques and deep neural networks. We discuss each of these methods in this section.

2.1 Traditional Statistical Forecasting

In this competition, the hosts have provided code and baselines for 24 different point forecasting methods ranging from simple moving average (MA) to exponential time smoothing (ETS) and Auto-Regressive Integrated Moving Average (ARIMA) [1] and its variants. We tried to explore more by decomposing the time series at an overall level as done in Figure 1.

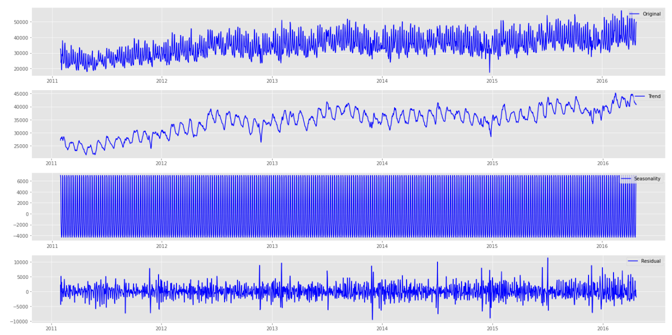

Figure 1: Decomposition of the time series

We decompose the time series into its trend, seasonal and residual components. As can be seen in Figure 1, it is clear that the time series has a strong positive trend and the sales increase over time. Also, we can see a strong seasonality pattern in the series. The ARIMA method which is the best of the baseline methods works well when there is a trend in the series but does not work well when the series contains a strong seasonal pattern. To get around this, we decided to use Seasonal Auto-Regressive Integrated Moving Average with external variables (SARIMAX) [1].

To perform SARIMAX, we need the trend as well as the seasonality to be stationary. We examine this by visualising the mean and standard deviation of the series as seen in Figure 2.

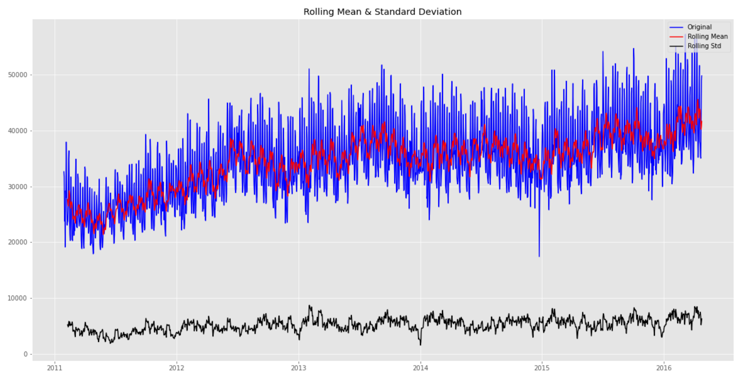

Figure 2: Stationarity of the time series

As seen in Figure 2, it is clear that the time series is not stationary as the mean of the series changes over time. We also confirm this by conducting the Augmented Dickey–Fuller test which gives us a p value of 0.51. It accepts the null hypothesis that a unit root is present, thus the series is non-stationary.

The SARIMAX method makes the time series stationary by the means of differencing. To find the optimal degree of differencing, we ran a grid search for the model parameters. Finally, we used the parameters that gave us the best Akaike information criterion (AIC) score for the final model.

2.2 Automated Forecasting

As seen in section 2.1, traditional time series algorithms require a lot of statistical understanding and fine tuning the parameters for the model to work well. To get around this, Facebook created its own forecasting framework called `Prophet` [2].

`Prophet` is an automated forecasting method that uses an additive model and takes into account complex trend, seasonality patterns and also holiday events. It is also said to be robust to missing values and outliers. This makes it a perfect fit for our analysis since our data also contains all these components as explained in the previous section.

2.3 Gradient Boosting

Among various ML algorithms, we explore Ensemble Learning as an approach due to its high prediction power which is achieved via combination of weak learners. Among the various ensemble methods, we focus on gradient boosting algorithms: LightGBM [3] and XGBoost [4]. Gradient boosting, in general, provides a flexibility in training and works well with categorical and numerical data, along with handling of missing data [5]. Despite the flexibility offered by gradient boosting methods, we perform a number of feature engineering steps to ensure a high model performance. First, a simple feature extraction is implemented to capture important aspects such as store-based product release date, product prices and calendar dates. For the product prices, normalization along with computation of basic aggregations, rolling aggregations and momentum is performed. Next, we extract lag features which is implemented on the newly extracted features using a range of multiple shifting and rolling periods. Finally, we extract mean encoding of the categorical data. The resulting extracted feature set is used to train both gradient boosting methods.

2.4 Forecasting with DNN

We tried several approaches to apply neural networks like LSTM, CNN, and regular fully connected layers. We obtained the best result (0.64) by using an initial categorical embedding to characterize time variables followed by four dense layers ending with a single linear unit to make the prediction. Changing the architecture of this model into an LSTM was not beneficial, given that all temporal information was thrown into categories that were already considered in the embedding layer.

Hybrid methods such as mixing exponential smoothing and LSTM show good results [6]. For this reason, we transformed the time series and visually searched which transformation extracted the most significant variations. The Discrete Wavelet Transformation (DWT) [7] seemed to capture the most relevant features of the time series.

We use multiple techniques in parallel: five rolling averages (with parameters 7, 30, 60, 90, and 180), their standard deviation, and DWT, each of which was fed into the neural network. After seeing no improvement from including DWT, it was removed to save memory. The neural network exhibited signs of bias and issues with generalization, to deal with this, the original kernel's network was improved by adding batch normalization layers and dropout.

2.5 Optimum Weighted Average

With the validation labels of the public leader-board published, we used them to create a function that calculated the submission score WRMSSE. With this as an error function, we create a meta learner; a function (with weights per model as input) to perform a weighted average of our model's forecasts. The weights associated with each model were optimized by variating each by a $\delta = 0.005$ and iterating, updating on each improvement.

To find out how much this approach could help, we tried to combine eleven model's CSV public predictions, including ours. None of the predictions had a score lower than 0.474, but this method got a score of 0.468. Training time for this model does not consider individual training times per model, but rather the final optimization from a set of fixed forecast. By using only our five models the optimization of the coefficients in the weighted average was an asymptotic convergence to the best score in our collection, but always combining DNN and LightGBM (See Figure 3).

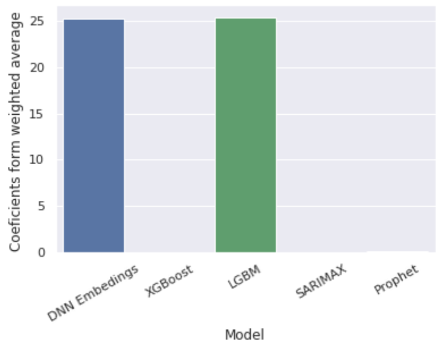

Figure 3: Optimized coefficients of weighted average

3 Results and Discussions

Table 1 presents the performance of all models discussed in Section 2.

Table 1: Comparison of Model Performance

Model	Test WRMSSE	Training Time
SARIMAX	1.06706	864 mins
Prophet	0.80324	258 mins
LightGBM	0.47506	101.4 mins
XGBoost	0.80436	596.6 mins
DNN	0.64157	71.09 mins
Weighted Average (Our Models)	**0.47813**	**104.5 mins**
Weighted Average (Multiple Models)	**0.46833**	**87 mins**

As observed in Table 1, the SARIMAX method performed the worst compared to all the other methods with an WRMSSE of 1.06. This is likely because, though this method works well on time series containing trend and seasonality, it fails to work well with outliers and does not take events like holidays into account. Additionally, traditional statistical methods are extremely difficult to model and fine tune due to their complexity. Also,this method can only handle uni-variate time series and the final evaluation was based on forecasting unit sales of the products aggregated at a store level. This meant that we had to forecast and grid search for optimal parameters at this granularity of the dataset which was very time consuming as can be seen in the table.

To get around the above drawbacks, we also decided to model the time series using `Prophet` as mentioned in section 2.2. It did not come to us as a surprise that `Prophet` performed much better than SARIMAX as it is known to take holiday effects and also extreme outliers in consideration. Additionally, `Prophet` also was able to forecast in one third of the time taken by SARIMAX as it fine tunes the parameters by itself.

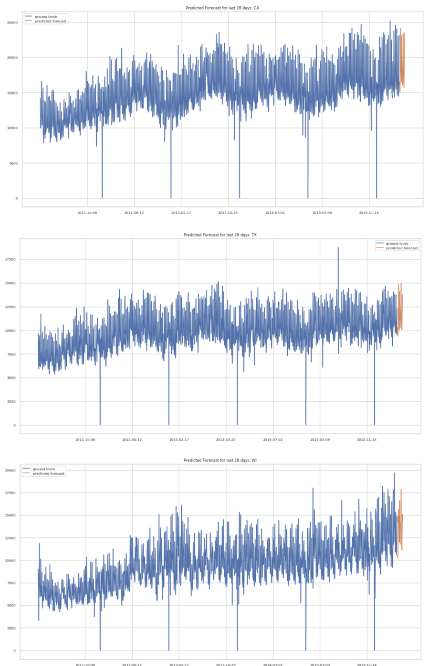

Figure 4: Predicted Forecasts per `STATE_ID` by LightGBM

As mentioned in Section 2.3, both the gradient boosting models (LightGBM and XGBoost) were trained on the same feature set. It is observed that LightGBM provides nearly two times better WRMSSE score than XGBoost which is achieved by far less training time than XGBoost. Similar model parameters were used for both models with a single exception: LightGBM was trained using Tweedie Regression as an optimizer whereas XGBoost was trained using Squared Error Loss (Linear) Regression as an optimizer. Both methods involved training individual models per `STORE ID` and the model predictions were performed accordingly, i.e. per `STORE ID`. As LightGBM performed the best among all our models, we observed LightGBM model predictions for `d_1914 - d_1941` (Figure 4).

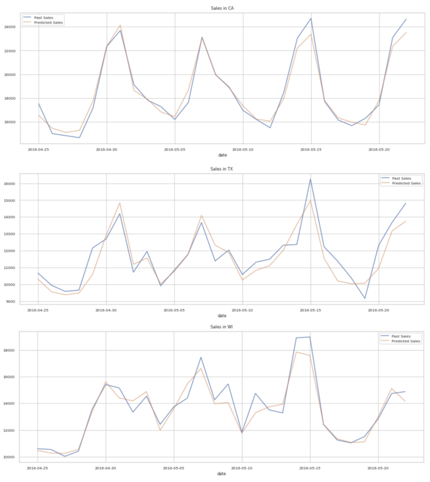

Figure 5: Prediction comparison with Evaluation dataset per `STATE_ID` by LightGBM

With the validation labels of the public leader-board published, figure 5 shows a comparison between ground truth and LightGBM Predictions for `d_1914 - d_1941`.

Concerning DNN, one unexpected result was that we obtained better results from shallow networks. This could be due to insufficient features for the model, or a small validation set that showed contradictory indications of bias and overfit.

4 Conclusion and Discussion

In this report, we explore four primary methods to forecast point sales. We believe that although the traditional methods like SARIMAX did not perform on par with the LightGBM or DNNs, it still gives us a good insight into the time series and its trend and seasonality components. Additionally, automated forecasting can be used when you want to quickly experiment, get a baseline and forecast without being too worried about the structure of the underlying data or its components.

Recently, deep neural networks have been used to get state of the art results on different problems and in various domains. It is the go to approach for any problem at hand. However, in the current case we see that a simple tree based ensemble method like the LightGBM can perform way better than DNN's with not too much difference in training times. This might be because DNN's tend to overfit the problem easily and are not generic enough for future forecasts. On the other hand, a lighter model like the LightGBM surprisingly performed much better than its counter part XGBoost.

In line with [6], Figure 3 provides evidence that hybrid methods that combine DNN's and other classic techniques yield improved results. This result could mean that these two approaches interact positively, canceling each other's bias, whereas the rest behave independently. However, it is possible that combining DNN's and other models besides LightGBM would yield good results, but given that LightGBM gets the best score, it is expected that the optimizer would stick to it.

Averaging all predictions was poor (score of 0.81), but using a weighted average guided by a subset yielded promising results. Future work could measure the expected gain in performance as a function of the number of predictions and the type of forecast.

5 Author Contributions

The report acknowledges the following contributions:

Name	Contribution
Nolan	Traditional and Automated Forecasting Method, Flash Talk Preparation, Documentation
Ajinkya	Gradient Boosting Techniques, Flash Talk Presentation, Documentation
Rodrigo	Forecasting with DNN, Weighted Averaging, Flash Talk Preparation, Documentation

Our work is spread across multiple Kaggle kernels which is listed as follows:

- **SARIMAX**: Kaggle Kernel Link
- **Prophet**: Kaggle Kernel Link
- **LightGBM**: Kaggle Kernel Link
- **XGBoost**: Kaggle Kernel Link
- **DNNs with Categorical Embeddings**: Kaggle Kernel Link
- **Optimum Weighted Average**: Kaggle Kernel Link
- **Multiple Models**: Kaggle Dataset Link

6 Process and Supervision Evaluation

As the M5 competition is presented in two parts: Accuracy and Uncertainty, we decided to focus on the "Accuracy" part of the competition. The public Kaggle kernels were quite beneficial as a lot of methods had already been explored before we started and kernels which focused on Exploratory Data Analysis (EDA) helped us to better understand the data.

We discussed the possibility of exploring multiple methods simultaneously with Twan during our first teacher-team meeting. This gave us an insight into how we should manage the work load and work efficiently as a team. We primarily divided the tasks as mentioned in section 5 but that did not constraint us from assisting each other with the other tasks. We had regular team meetings and used discord to constantly stay in touch.

In our second teacher-team meeting with Elena, we were provided with a number of literature for reference along with `Facebook Prophet` framework which we employed for automatic forecasting. This was quite useful in experimenting with traditional forecasting methods. In our final team-teacher meeting with David, we discussed about the final tasks of our project which involved combining our best models which we successfully did as described in section 2.5.

References

[1] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

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

[3] 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 *Advances in neural information processing systems*, pages 3146–3154, 2017.

[4] 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*, pages 785–794, 2016.

[5] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[6] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

[7] Mark J Shensa et al. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on signal processing*, 40(10):2464–2482, 1992.