Point Forecasts of Future Sales

Xinyi Tan
New York University
xt2038@nyu.edu

Shengshi Yuan
New York University
ssy293@nyu.edu

Abstract

In this project, we focus on generating point forecasts for future daily unit sales using the M5 competition dataset provided by Walmart[4]. We built the univariate models (Seasonal Naive, Simple Exponential Smoothing, Seasonal ARIMA, Random Forest) on the product-state level and the multivariate model (LSTM) on product-state level. We demonstrate that although the multivariate LSTM does not significantly outperform the univariate models, it is more efficient in dealing with a large-volume time series with high computational complexity.

1 Introduction

Forecasting sales has been one of the major tasks in retail analysis. Accurate Sales forecasting helps retail service providers to better prepare for peaks in business and ensure that there are enough stock and workforce to support the demand. Also, accurate sales forecasting allows companies to estimate cost and revenue, which help them to understand when they would have the budget for location expansion or marketing activities. What's more, consistent and accurate sales forecasting helps sales managers to identify potential issues such as decreasing trends and investigate the causes ahead.

The M5 competition dataset covers the daily unit sales of over 3,000 products at 10 Walmart stores in California (CA), Texas (TX), and Wisconsin (WI). The dataset also contains explanatory variables such as promotions and special events. 4 univariate models (Seasonal Naive, Seasonal ARIMA, Simple Exponential Smoothing, and Random Forest) and a multivariate model (LSTM) were examined on the given dataset. The classic time series models only makes predictions based on historical data while the two machine learning models can incorporate additional explanatory variables or perform multivariate analysis. We found that it is hard to significantly outperform Seasonal ARIMA on every single time series, but including additional information does improve performance when the data does not have a clear pattern. Also, tuning the parameters of Seasonal ARIMA models separately for each time series is not feasible when dealing with a large number of time series. Therefore the multivariate approaches such as LSTM is a more suitable for large scale time series forecasting.

Code for this project available at:

`https://github.com/MinnieTan/Point-Forecasts-of-Future-Sales`

2 Problem Definition

2.1 Task

The goal of the project is to make 28 days ahead point forecasts for the unit sales of products at 10 Walmart stores using historical daily sales data from 2011 to 2016. For the Random Forest model, the explanatory variables in the *calendar.csv* were also taken as input.

Given the significant amount of time-series available (30490 series in total) in the original datasets, we decided to focus on the unit sales predictions for a randomly selected product at 10 stores and explore different algorithms in the following sections.

From exploratory analysis (Figure 1.), we found that the original datasets contain mainly items with category *FOODS* and among which *FOODS 3* department has the most number of unique items. Therefore, we randomly sampled a product, distinguished with product id *FOODS 3 099*, and built our univariate (product-store level) and multivariate (product-state level) time series models on it.

Figure 1: Unique Item Counts for Categories (Left) for Departments within FOODS Category (Right) in Stores

2.2 Data Description

The dataset is organized in the form of grouped time series. More specifically, the 3,000 products were classified into 3 categories (*HOBBIES*, *FOODS*, and *HOUSEHOLD*), and 7 departments. Besides the daily sales data, the dataset also includes explanatory variables such as promotions, transaction time, and special events.

Each row in *calendar.csv* (Table 1.) contains information about the dates on which products are sold. Each row in the *sales train validation.csv* and *sales train evaluation.csv* (Table 2.) represents the daily unit sales history for a specific item in a certain store. The consective 0s at the beginning of the series indicate that the corresponding product was not on sold at the specific store yet.

Table 1: Example Data Instance for *calendar.csv*

date	wm yr wk	weekday	wday	month	year	d	event name 1
011-01-29	11101	Saturday	1	1	2011	d 1	NaN

event type 1	event name 2	event type 2	snap CA	snap TX	snap WI
NaN	NaN	NaN	0	0	0

Table 2: Example Data Instance for *sales train validation.csv* and *sales train evaluation.csv*

id	item id	dept id	cat id	store id	state id	d 1	...	d 1941
HOBBIES 1 001 CA 1 evaluation	HOBBIES 1 001	HOBBIES 1	HOBBIES	CA 1	CA	0	...	1

Table 3: Example Instance for Random Forest Modeling Input

sales	roll avg 42	roll avg 28	roll avg 14	roll std 42	roll std 28	roll std 14	wday	month	snap CA	event
0.342	-0.9206	-1.045	-0.8727	-0.8675	-0.4581	-0.1585	-1.0003	-0.7148	-0.6991	-0.2948

To incorporate external information for more thorough and accurate forecasting, we combined the date information and daily unit sales data together and performed column-wise standardization for Random Forest modeling inputs. The event feature indicates whether that date has an event happening (via *event type 1* and *event type 2* features in the original *calendar.csv*). Rolling averages and standard deviations at varying lengths for daily unit sales were also adopted for Random Forest modeling (Table 3.). The exact lengths to keep were determined via the feature importance vector available in *sklearn* RandomForestRegressor class.

3 Models

3.1 Baseline Modes

We adopted the seasonal naive model and the simple exponential smoothing as baseline models for this project.

In the seasonal Naive method, the forecast at time T is equal to the last known observation of the same period ($\hat{Y}_T = Y_{T-7}$).

In the Simple Exponential Smoothing model, forecasts are the weighted averages of past observations, where the weights decrease exponentially as observations trace further back to the past. For each time series, the parameter α is chosen by the python module $Statsmodels$ using maximum likelihood estimation.

$$\hat{Y}_{T+1|T} = \alpha Y_T + \alpha(1-\alpha)Y_{T-1} + \alpha(1-\alpha)^2 Y_{T-2} + ...$$

3.2 Seasonal ARIMA

A seasonal $ARIMA(p,d,q)(P,D,Q)_7$ was formed by including additional seasonal terms in the non-seasonal ARIMA model. The order of differencing, d, was determined using the ADF test and KPSS test. Seasonal differencing offer was determined using the Canova-Hansen test. To determine the optimal values of p, d, P, and Q, we used the step-wise algorithm outlined in Hyndman and Khandakar(2008) and chose the model with the lowest AIC value. After a model is chosen, we manually adjusted the parameters to check if the residuals still had remaining autocorrelations.

3.3 Random Forest

Random Forest is the ensemble extension of Decision Tree that fixes Decision Tree's tendency to overfit. The classification result of an input instance is a combination of classification results of each tree in the forest, as described in the algorithm (Hastie, Tibshirani and Friedman, 2008) in Figure 2.

In this project, time series forecast is converted to a regression problem by matching the current-day vector (as shown in Table 3.) with next-day unit sales as the input-output pairs. We generated rolling average and rolling standard deviations of unit sales in the input vectors to capture the underlying pattern. The evaluation and test metric RMSSE (refer to Section 4.2 for detail description) was plotted against maximum depth of trees, minimal number of samples for leaf nodes, minimal number of samples to split on, and number of trees in the forest to tune the hyper parameters for each product-store model. Figure 3. is an example hyper parameter tuning process for our Random Forest model built for $TX3$ store.

To take a concrete example, our final Random Forest model for $CA1$ has 60 trees in its forest. The final prediction for the input instance shown in Table 4 is approximately 17, which is a combination of results from those 60 trees, which can be 10, 15, etc.

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Figure 2: Random Forest Algorithm

Figure 3: Random Forest Hyper Parameter Tuning for the TX3 Store

3.4 Long Short-Term Memory

Long Short-Term Memory is a popular choice of machine learning algorithm for prediction tasks that preserves long-term information and skips short-term input by utilizing latent variables (Zhang, Lipton, Li and Smola, 2020).

In this project, we built LSTM models at product-state levels where each model took parallel daily unit sales series (column-wise standardized to give equal consideration across stores) for $FOODS$ 3 099

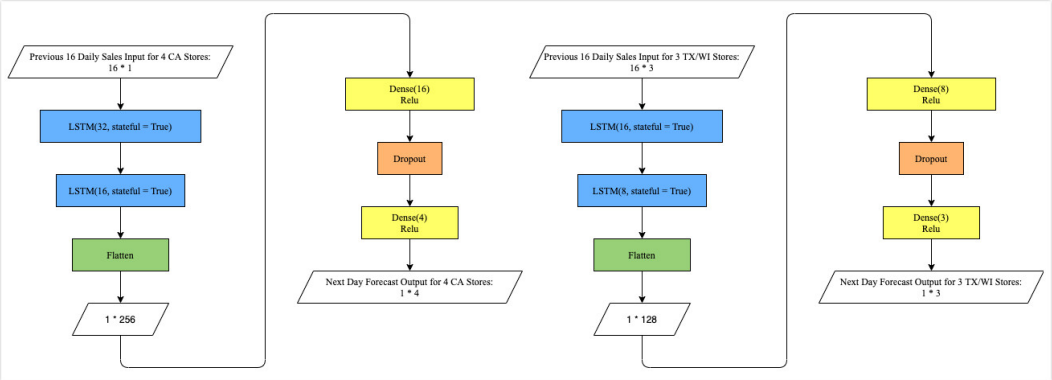

Figure 4: LSTM Architecture for CA Stores (Left) TX/WI Stores (Right)

in states (4 series for CA, 3 for TX and WI) and predicted future sales for those stores in parallel. Average RMSSE was plotted against window length, batch size, L2 weights regularization, and maximum number of epochs to fine tune the hyper parameters (refer to Figure 5. for example learning curves).

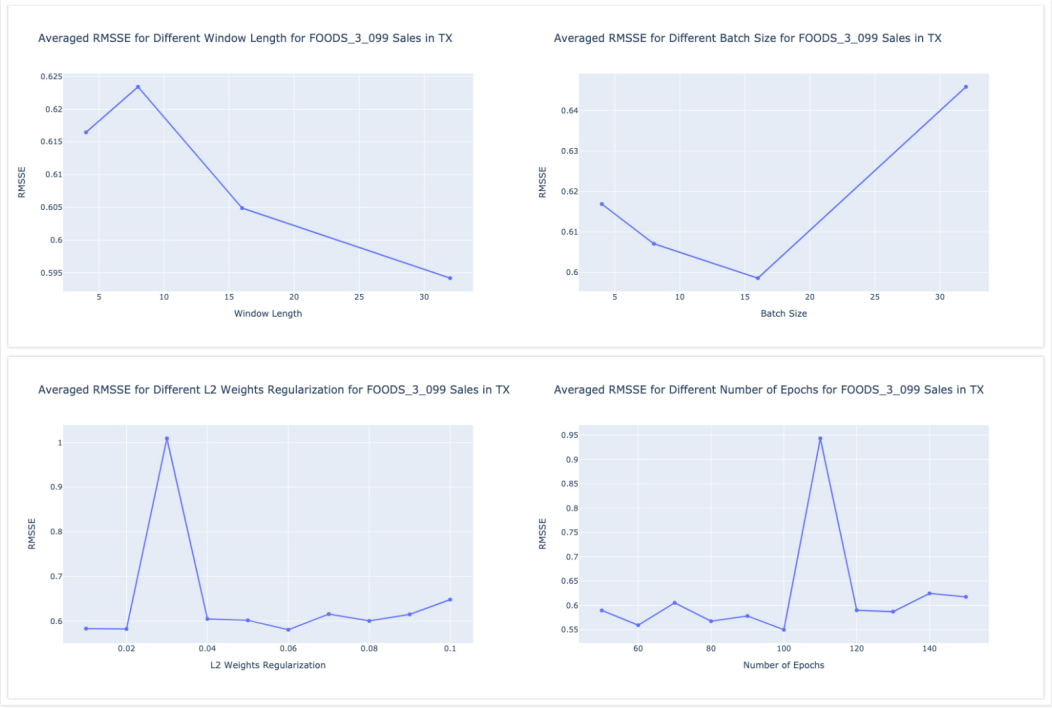

Figure 5: LSTM Hyper Parameter Tuning for WI Stores

Take our model for CA stores as an instance, after passing previous 16 days standardized daily sales series of 4 stores through our stacked LSTM model (indicated in Figure 4.), we generated prediction <16, 7, 12, 8> (<11, 0, 12, 8> for ground truth) for 4 stores specifically according to trained weight and bias parameters.

4 Experimental Evaluation

4.1 Methodology

Since the goal of the project is to give 28 days ahead predictions of daily unit sales for products, the last 28 days of each time series in the $salestrainevaluation.csv$ were adopted as the test data, 28 days before that as the validation data, and all previous remaining days as training data.

We firstly trained the models on the train data, tuning the hyper-parameters on the validation set according to model performance with respect to RMSSE results. After the optimal hyper-parameters were chosen, we trained the models on train and validation data together and compared their performances in test data via RMSSE values and visualizations.

We chose the Root Mean Squared Scaled Error (RMSSE) as our evaluation metric because it's scaled independent, meaning that it can be effectively used to compare forecasts across series with different sales scales (e.g. napkins and televisions probably have quite different sales scales).

$$RMSSE = \sqrt{\frac{1}{h}\frac{\sum_{t=n+1}^{n+h}(Y_t - \hat{Y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(Y_t - Y_{t-1})^2}}$$

where Y_t is the actual future value of the examined time series at point t, $\hat{Y}_t$ the generated forecast, n the length of the training sample, and h the forecasting horizon.

4.2 Results

Table 4: State-Wise Average RMSSE Results (in bold the best score, the lower the better)

Model	CA	TX	WI
Seasonal Naive	1.0128	0.6743	1.0679
Simple Exponential Smoothing	0.7399	0.5871	0.8145
Seasonal ARIMA	0.7465	0.5212	0.762
Random Forest	**0.713**	0.5774	**0.7398**
LSTM	0.7488	**0.5199**	0.8415

Table 5: Store-Wise RMSSE Results (in bold the best score, the lower the better)

Model	CA 1	CA 2	CA 3	CA 4	TX 1	TX 2	TX 3	WI 1	WI 2	WI 3
Seasonal Naive	0.6901	1.0161	0.692	1.1653	0.8512	0.5747	0.5971	0.9155	1.4564	0.8318
Simple Exponential Smoothing	0.6698	0.9582	0.6601	0.6715	0.7427	0.5236	0.495	0.7351	1.061	0.6474
Seasonal ARIMA	0.6702	1.0523	0.6639	**0.5996**	0.6072	**0.4888**	0.4675	0.6742	**1.0332**	**0.5785**
Random Forest	**0.6152**	**0.8948**	0.6678	0.6742	0.6511	0.5419	0.5393	**0.5711**	1.0696	0.5788
LSTM	0.6816	0.9663	**0.6334**	0.7138	**0.5986**	0.5172	**0.444**	0.7285	1.1137	0.6822

4.3 Model Comparison

As shown in Table 4, Random Forest (adopted external information) and LSTM (multivariate analysis among stores in the same state) have better performances at the product-state level. The decent performances of LSTM product-state models suggest that there should be some correlations and interactions among product sales in the same state.

As shown in Table 5, Seasonal ARIMA and Random Forest gave the best performance on 4 and 3 out of 10 product-store series respectively, suggesting that they are competitive univariate models. However, building and tuning Seasonal ARIMA models and Random Forests for each product-store level time series is time consuming and inefficient given the large scale of the whole dataset.

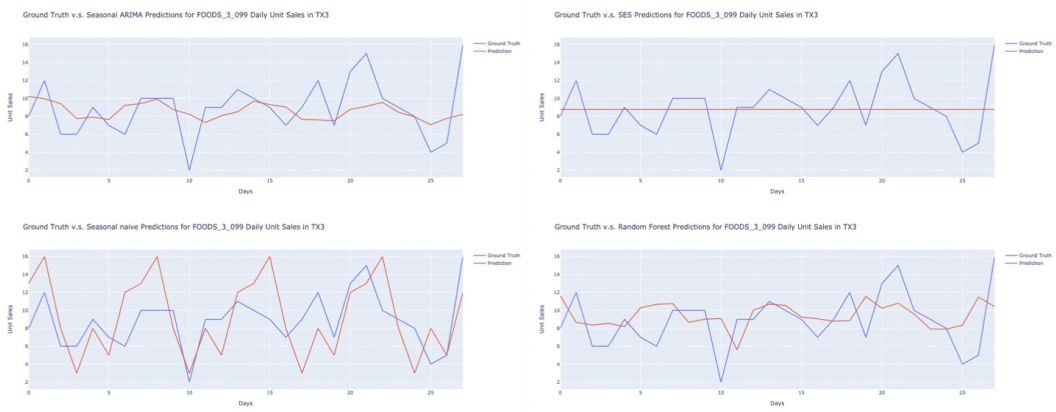

Figure 6: Ground Truth vs. Models Predictions for the TX3 Store

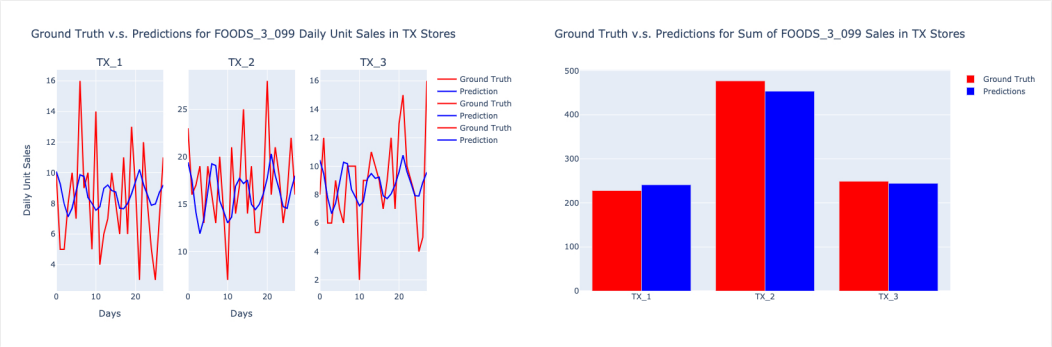

Figure 7: Ground Truth vs. LSTM Predictions for TX Stores

Figure 6 is the example graphical Ground Truth vs. Prediction comparison for our univariate models built for $TX3$ store and Figure 7 is for the LSTM model built for TX stores. Our models may not seem very strong in fitting the exact daily sales units, but they were able to extrapolate the mean tendency of sales. When comparing the sum of ground truth vs. prediction results for the 28 days in the test data, the difference is approximately 4 percentage of the ground truth sales.

5 Conclusions and Future Work

When considering the overall state-wise average model performance, Random Forest and LSTM have better performances. This indicates that external information and multivariate analysis should be considered for such large-scale time series forecasting. Multivariate models such as LSTM, due to computational efficiency, are preferred even though they wouldn't necessarily improve the performance significantly. In our project, LSTM models were built on parallel daily unit sales series at product-state level. Future forecasters should consider adopting external information in multivariate analysis as well to further boost the forecast accuracy.

We noticed that the unit sales of many products are intermittent, containing periods of zeros. However, we are not sure if this was caused by stock-outs or decreased demand and interest from customers. If it was caused by stock-outs, modeling these periods would result in an underestimation of the true demand. Therefore, it would be helpful to have the stock information of each product as an additional explanatory variable. With the stock information, we can better understand the true demand and help the company to manage stock wisely.

A surprising finding is that Seasonal ARIMA gave very similar performance to Simple Exponential Smoothing on $CA1$ and $CA3$ stores. Simple Exponential Smoothing even outperformed Seasonal ARIMA on $CA2$ store prediction. This highlights the importance of including additional information when the time series does not have a clear trend and seasonality pattern.

Student contributions

Both Shengshi Yuan and Xinyi Tan contributed to problem formulation, exploratory data analysis, model evaluation, presentation preparation, and report write-up.

Shengshi Yuan worked on Seasonal Naive, Simple Exponential Smoothing, and Seasonal ARIMA model building.

Xinyi Tan worked on data processing, Random Forest model building, and LSTM model building.

References

[1] Hastie, T., Tibshirani, R. and Friedman, J., 2008.*The Elements Of Statistical Learning.* 2nd ed. Stanford: Springer, p.588.

[2] Hyndman, R. J., and Khandakar, Y., 2008. *Automatic time series forecasting: The forecast package for R.* Journal of Statistical Software,27(3), 1–22.

[3] Zhang, A., Lipton, Z., Li, M. and Smola, A., 2020. 9.2. *Long Short-Term Memory (LSTM) — Dive Into Deep Learning 0.15.1 Documentation.* [online] D2l.ai. Available at: <https://d2l.ai/chapter_recurrent-modern/lstm.html>.

[4] The M5 Competition Dataset(2020). Available at: https://www.kaggle.com/c/m5-forecasting-accuracy