THE INVENTORY PERFORMANCE OF DIFFERENT FORECASTING METHODS - BASED ON M5 RETAIL DATA

Hari Prasath John Kennedy, Li Liu
Software & Societal Systems
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{hjohnken,ll3}@andrew.cmu.edu

ABSTRACT

Effective inventory management is crucial for the profitability of retailers in the highly competitive retail industry. Retailers face the challenge of finding the right balance between holding sufficient stock to meet customer demand and avoiding excess inventory. In this context, forecasting is an essential tool that enables retailers to gain valuable insights into the demand for their products, which in turn helps them make informed decisions about inventory levels. We have observed rise of many deep-learning forecasting methods rise to the state of the art in recent years, and we wish to develop a inventory simulation algorithm that can be used to evaluate the inventory performance when operating with different forecasting models. In this project, we have built such system, and used it to compare the inventory performance for N-BEATS and Seq2Seq models.

1 MOTIVATION

The M5 Kaggle competition offers a unique opportunity for us to delve into the fascinating domain of retail sales and inventory management. This contest emphasizes forecasting sales across a diverse array of products in various retail scenarios, making it an ideal dataset for analyzing and developing deep learning models capable of accurately predicting future sales patterns.

In order to accomplish this, we intend to investigate multiple deep learning forecasting architectures, using Naive Seq2Seq and LSTM as our baselines and plan to expand our implementation by incorporating N-BEATS and other state-of-the-art techniques, subject to the constraints of time and computational resources. By comparing the performance of these models, our goal is to pinpoint the most effective forecasting architectures for retail sales data and inventory management.

Our project's focus extends beyond merely attaining high levels of accuracy in our forecasting models. Although the competition highlights the significance of precise sales predictions, we contend that the true value of these forecasts lies in their ability to shape and enhance inventory management practices. By utilizing the forecasting data to inform inventory decisions, retailers can optimize their inventory levels, minimize waste and boost overall efficiency. Drawing upon the inventory simulation and performance calculation mechanism proposed by Petropoulos et al. (2019) we aim to assess the efficacy of different deep learning models in guiding inventory management decisions.

2 LITERATURE SURVEY

2.1 M5 FORECASTING COMPETITION

To gain an overall context of the M5 competition, we started by exploring the Kaggle information page for the M5 competition Howard & Makridakis (2020). The website contains many good resources such as the overview of the competition, competition handout, and sample dataset. The competition handout is especially helpful because it explains some basic benchmark methods such as Simple Exponential Smoothing and Moving Averages. We could easily gain access to the dataset with Kaggle's API, and perform preliminary data analysis. Since the competition finished three

years ago, we were able to draw some insights from reading the competition analysis article by Makridakis et al. (2022). Two things we have found particularly interesting from the paper are: (1) Many top winning solutions utilize a non deep learning LightGBM algorithm, and the algorithm has proven superior in analyzing hierarchical retail sales data. (2) State-of-the-art deep learning methods such as DeepAR and N-BEATS have also shown potential in accurately predicting sparse time series (e.g. retail sales data).

2.2 Deep Learning Forecasting Techniques

To further study the feasibility and best practices of deep learning in time series analysis and forecasting, we examined Amazon research team's literature survey paper by Benidis et al. (2020). What we particularly liked about this paper is that it provides a holistic view of the neural forecasting problem landscape rather than diving deep into specific architectures. The paper eloquently formalizes the forecasting problem solutions into local univariate model, global univariate model and global multivariate model. We were genuinely surprised by the amount of machine learning techniques can be utilized to solve the forecasting problem. The techniques include MLP, CNN, RNN, Transformer, GANs, State space, etc. The paper enumerated many architectures, and we have found following interesting and promising in the point forecast area:

1. LSTM RNN model Sherstinsky (2018)
2. Seq2seq RNN model
3. Transformer based model
4. N-BEATS MLP model Oreshkin et al. (2019)

2.3 Inventory Performance Definition

A shared concern of many companies to incorporate machine learning components to operations is the uncertain return from almost guaranteed high research cost. Thus, our motivation behind this project is to evaluate the business impact of different forecasting models. Specifically, we want to model the inventory operation at retail industry and quantify the impact of accurate forecasting on inventory performance. Petropoulos et al. (2019) utilized the data from M3 competition and proposed a relatively simple framework to model inventory performance under several assumptions. We intend to follow the paper to build out our inventory model, and evaluate them with Walmart retail sales data in M5 competition.

2.4 M5 solutions and tutorials

Kaggle platform has provided us ample amount of exploratory data analysis and solution tutorials code to browse.
For exploratory data analysis and statistical benchmarks, we have found tutorial by Paparaju (2020) helpful. The tutorial provided intuitive visualizations and explanations to the data characteristics. Furthermore, the tutorial has implemented a toy version of the statistical benchmark methods, which we intend to expand to the entire dataset.
For forecasting modeling, we have found following three tutorials helpful:

1. LSTM: Shechter (2021a) provided a great tutorial notebook on Kaggle. It implements a PyTorch version of LSTM model for M5 forecasting. The notebook includes crucial steps of data cleaning, transformation, feature engineering, model training, etc.
2. Seq2seq: Shechter (2021b) also wrote a Seq2seq version tutorial with M5 data. The notebook structure resembles the LSTM notebook.
3. Encoder-decoder: Durgaprasad (2020) implemented his own version of forecasting model based on winning solution of a web traffic prediction challenge posted by Google. We have chosen this notebook to test out our model development environment, and we were able to reproduce similar RMSE result on our version of notebook. We shall discuss the result more in the next section.

3 Exploratory Data Analysis

This section we present some key exploratory data analysis and simple statistical model benchmark.

3.1 Stores' sales distribution

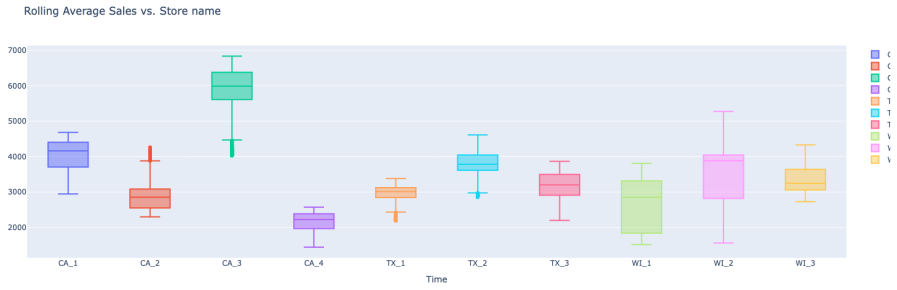


Figure 1: Rolling average store sales vs. store name

We can observe that all though all stores have different average sales, the amount of variation in sales are relatively consistent across all stores.

3.2 Stores' sales trend across states

Figure 2. shows the average stores' sales trend across different states. We can observe nearly all stores exhibit the seasonal behavior, similar to the ones in macroeconomics. Another interesting fact is that different stores may experience different growth factor based on its locations. This effect is prevalent in Wisconsin state data as the lines of average sales cross regularly.

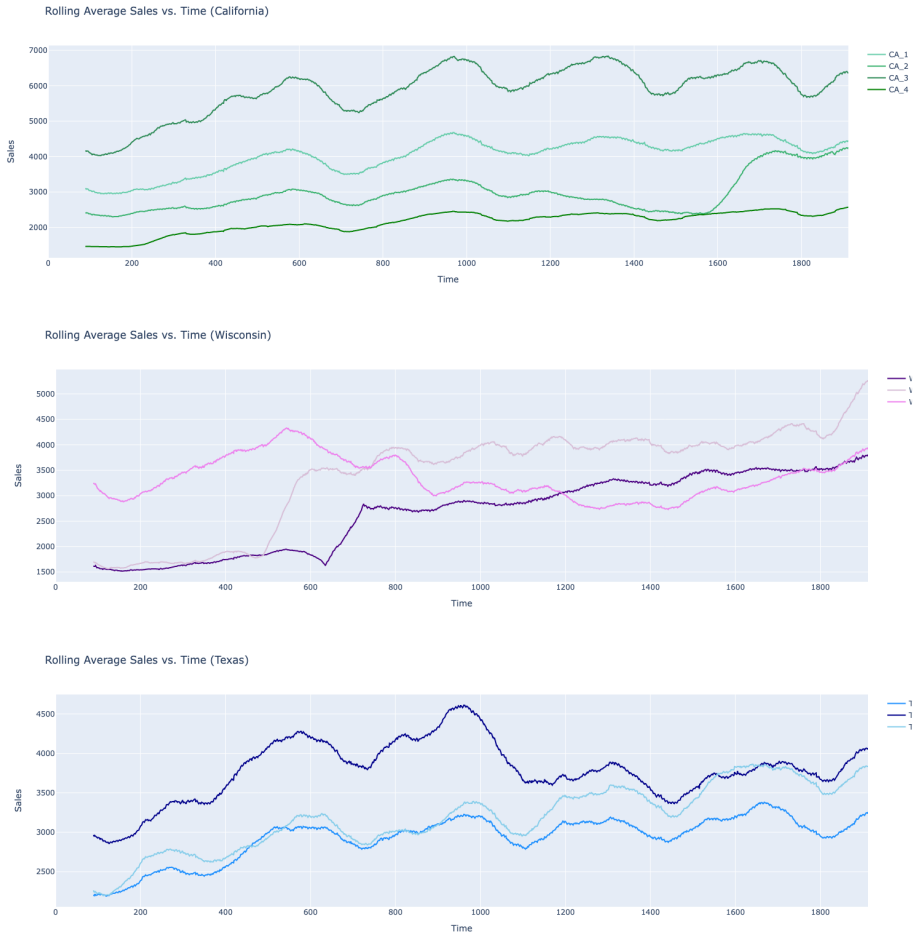


Figure 2: Stores' average sales across different states

3.3 DENOISE SALES DATA

Unlike some generic time series forecasting problems, the retail sales data can be erratic and intermittent. As demonstrated in the Figure 3, the original sales data often have huge spikes and valleys that are difficult for the algorithm to identify the underlying patterns. In this case, denoising the original data to a smooth curve is desirable. We find the wavelet denoising algorithm proposed by Paparaju (2020) is appealing, and intend to implement in our future work.

```
def maddest(d, axis=None):
    return np.mean(np.absolute(d - np.mean(d, axis)), axis)

def denoise_signal(x, wavelet='db4', level=1):
    coeff = pywt.wavedec(x, wavelet, mode="per")
    sigma = (1/0.6745) * maddest(coeff[-level])

    uthresh = sigma * np.sqrt(2*np.log(len(x)))
    coeff[1:] = (pywt.threshold(i, value=uthresh, mode='hard')
                 for i in coeff[1:])
    return pywt.waverec(coeff, wavelet, mode='per')
```

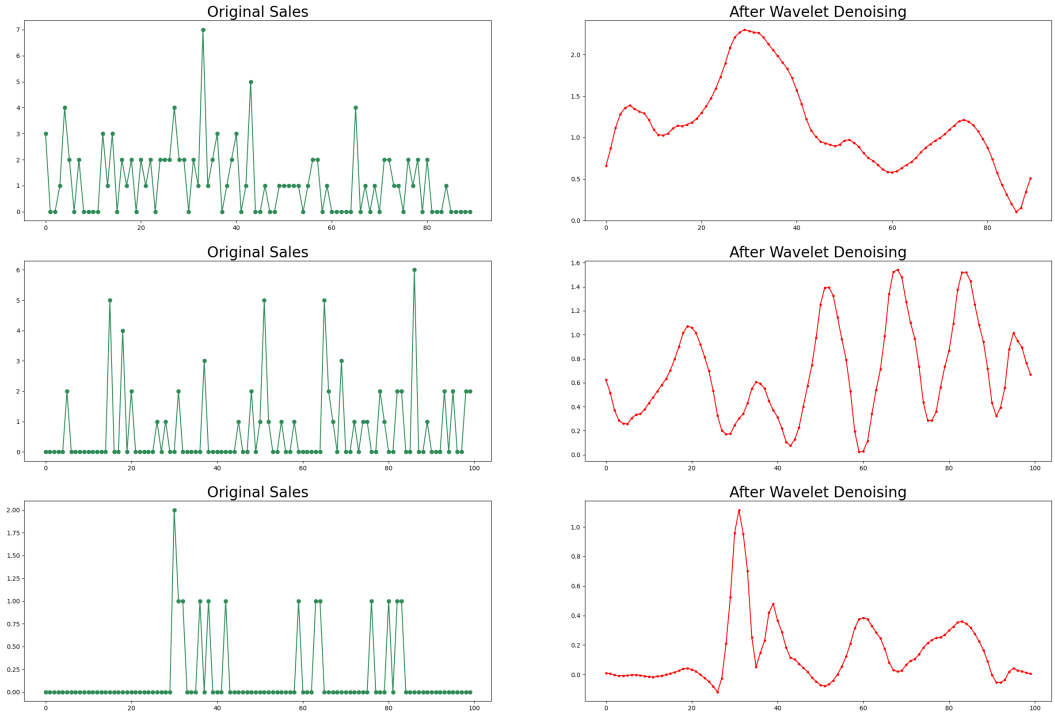


Figure 3: Denoise sales data

3.4 Statistical benchmark

Finally in the analysis, we look at some naive statistical model for benchmark. The statistical models we look at are Moving average, Exponential smoothing, and Autoregressive integrated moving average. The result we got here is from a sample 130-day sales data from a store. We plan to expand the benchmark to predict the sales data for all stores and record the result in the next phase.

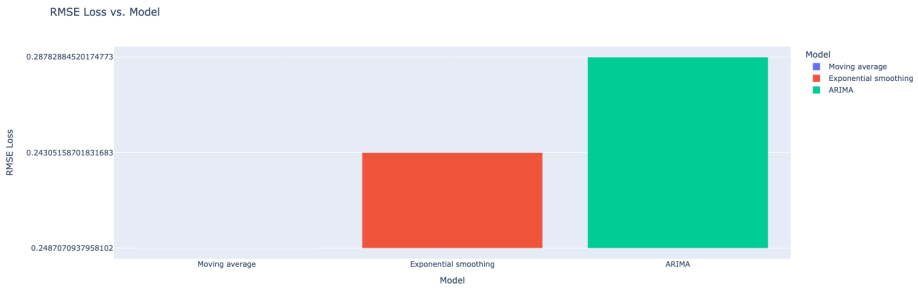


Figure 4: Statistical benchmark

4 Methodology

4.1 Data Source and Equipment

The data source is the Walmart sales data from 2011 to 2016, provided by the M5 competition Howard & Makridakis (2020). For model training and inventory simulation development, we used

Kaggle Notebook platform and Google Colab. Both platforms provide excellent tools for experimenting with data, training baselines, and developing simulation system.

4.2 LSTM Forecasting Model

Our LSTM forecasting model is inspired by the work of Shechter (2021a) and is further customized by incorporating a 1D convolutional layer. The motivation behind adding the 1D convolution is based on the survey by Benidis et al. (2020). The time-series data in our problem exhibits cyclical and seasonal patterns, which can be effectively captured by convolutional layers. By incorporating a 1D convolutional layer, we aim to enhance the model's ability to recognize and learn from the underlying temporal patterns in the dataset.

4.2.1 Feature Engineering

We employ lag features to capture temporal dependencies in the data. The sales data is shifted by 1, 7, 14, 28, and 365 days to create additional input features. By incorporating this information into the model, we aim to improve its ability to learn the underlying patterns and dependencies in the time series data.

4.2.2 Training Constraints

The training process is subject to certain constraints to ensure computational efficiency and prevent overfitting. The LSTM model is trained for 700 epochs with a batch size of 1. To mitigate the exploding gradient problem, gradient clipping with a maximum norm of 1 is employed.

4.2.3 Model Architecture and Hyperparameters

The model architecture consists of the following layers:

- A 1D convolution layer with 32 output channels, a kernel size of 3, and padding of 1. This layer is used to extract local features from the input data.
- An LSTM layer with two layers and 512 hidden units. The LSTM layer is designed to capture long-range dependencies in the data.
- Three fully connected layers with 256, 128, and 1 neurons, respectively. These layers are used to map the LSTM output to the final prediction.
- ReLU activation functions and dropout layers with rates of 0.25 and 0.2 to prevent overfitting.

The model is initialized with uniform weights in the range of -0.08 to 0.08. The loss function used is mean squared error (MSE). The Adam optimizer is employed with a learning rate of 1e-3 and weight decay of 1e-5. The learning rate is adjusted using a scheduler with a patience of 100 and a factor of 0.5.

4.2.4 Evaluation Metrics

We assess the customized LSTM model's ability to predict time series data accurately. We employ the RMSE as the evaluation metric which measures the average difference between the predicted values and the actual values. The model's performance is visualized by plotting the predicted values against the actual time series data for both the entire dataset and the test set.

4.3 Seq2Seq Forecasting Model

Our Seq2Seq model's implementation is heavily influenced by Kaggle tutorial by Shect (2021). We did not perform much customization on the model because it is used as a baseline model for the inventory evaluation system.

4.3.1 FEATURE ENGINEERING

Before developing the actual Seq2Seq forecasting model, we have to transform the raw data through a series of feature engineering. First, we transformed the data frame to be items sales per day with store id attached to it. Then we picked a single time series to analyze, with the shape of (1913, 1). Then we did a simple difference transform to remove the weekly seasonality by following the formula $X(t)' = X(t) - X(t-7)$. After that, we normalized the difference distribution to the range -1 to 1.

4.3.2 SPLIT DATA

We reserved the last 365 days as the test data for the test data. Then the 67% of the data will be used for training, and 33% will be used for validation.

4.3.3 ARCHITECTURE

Our Seq2Seq baseline model follows a relative simple architecture.

- Encoder: The encoder is a recurrent neural network (RNN) that uses Long Short-Term Memory (LSTM) cells. The LSTM takes input sequences of length 1, applies three layers of LSTM with a hidden size of 512, and outputs a single vector for each input sequence.
- Attention: The attention consists of a linear layer that maps the concatenated hidden states of the encoder and decoder to a vector of size 512, and another linear layer that maps this vector to a scalar. The resulting scalar is used to weight the hidden states of the encoder, and the weighted sum of these hidden states is used to compute a context vector that is passed to the decoder.
- Decoder: The decoder is an LSTM RNN that takes as input the concatenation of the previous output and the context vector, and generates the next output in the sequence. The decoder also uses the attention mechanism to compute the context vector. The output of the decoder is passed through a linear layer to produce the final output sequence.

The architecture of the network can be summarized with following output.

```
Seq2Seq(
  (encoder): Encoder(
    (rnn1): LSTM(1, 512, num_layers=3, batch_first=True, dropout=0.35)
  )
  (attention): Attention(
    (attn): Linear(in_features=1024, out_features=512, bias=True)
    (v): Linear(in_features=512, out_features=1, bias=False)
  )
  (decoder): AttentionDecoder(
    (attention): Attention(
      (attn): Linear(in_features=1024, out_features=512, bias=True)
      (v): Linear(in_features=512, out_features=1, bias=False)
    )
    (rnn1): LSTM(513, 512, num_layers=3, batch_first=True, dropout=0.35)
    (output_layer): Linear(in_features=1024, out_features=1, bias=True)
  )
)
```

4.3.4 TRAINING

The loss criterion used for training is RMSE. In addition, we use ADAM for optimizer and learning scheduler is CosineAnnealingLR. We trained the network for 30 epochs.

4.4 N-BEATS Forecasting Model

During the time of the project, we were having difficulties to augment the LSTM forecasting model to predict the sales for the future 28-day window. Therefore, we explored an alternative N-BEATS model architecture, and the model work is inspired by author mpware (2021) on Kaggle.

On the high level, N-BEATS (Neural basis expansion analysis for interpretable time series forecasting) is a deep learning architecture for time series forecasting. It was introduced in a paper by Oreshkin et al. in 2019.

The N-BEATS model decomposes a time series into a sequence of basis functions that are learned using fully connected neural networks. These basis functions are then combined to produce a forecast for future time steps.

The architecture of the N-BEATS model consists of several stacks, each of which contains a number of fully connected neural networks called "blocks". Each block produces an output that is a weighted sum of the outputs of its fully connected layers, and the weights are learned through backpropagation.

Our version of N-BEATS model consists two generic stacks, and it is a Global Multivariate forecasting model, which means the the model takes items data from all past items, and predicts all retail items future sales at the same time. In our model, the backcast time window (84 days) is three times the forecast window (28 days)

4.5 Inventory Simulation and Performance Calculation

The inventory simulation is broken down into three major pieces.

- Generate all 28-day forecasts for all simulation time period t.
- Iterate through all t and make relevant inventory decision based on forecast, inventory points and forecasts standard error. Record the decisions.
- After iterating through all time periods, use records to calculate inventory performance metrics

To simplify the simulation and focus more on the forecasting models, in the simulations we will be assuming there is no supply chain issue, and the stock will arrive with a constant lead time of 5 days. In addition, there is infinite amount of storage, but positive net inventory will incur holding costs. The unsatisfied demands from customers will be backlogged as a negative inventory.

4.5.1 Generate forecasts

Since the model utilizes the past 90 day sales data to predict the next 28 days sales. We need to first repetitively generate 28-day forecast for each day of the simulation period. As the result, the generated forecasts will be a matrix with shape (simulation_period, 28).

4.5.2 Iterate through time

During each time t, the inventory decision will be made with order-up-to policy. The total inventory replenishment should be calculated as:

$$o_t = F_t^{t+L} + ss_t - ip_t$$

where o_t is the amount of stock to order at time t; F_t^{t+L} is the total forecast demand from time t to $t + L$, L is the lead time (5 days); ss_t is the safety stock to order, and it can be calculated as $ss_t = \sigma_e * invsnorm(\alpha)$. σ_e is the accumulative lead time demand forecast error, and α is the desired service level (percent of time when the item is in stock).; ip_t is the inventory point at time t, and inventory point can be interpreted as the sum of net inventory i_t and work-in-progress inventory w_t (ordered but on the way) $ip_t = i_t + w_t$.

4.5.3 Compute Inventory Performance

The inventory-associated costs can be roughly broken down into three parts: ordering costs, holding costs and under stock costs. Without actual retail financial information, it is hard to compute these costs in actual numbers. However, we can proxy these costs with other metrics.

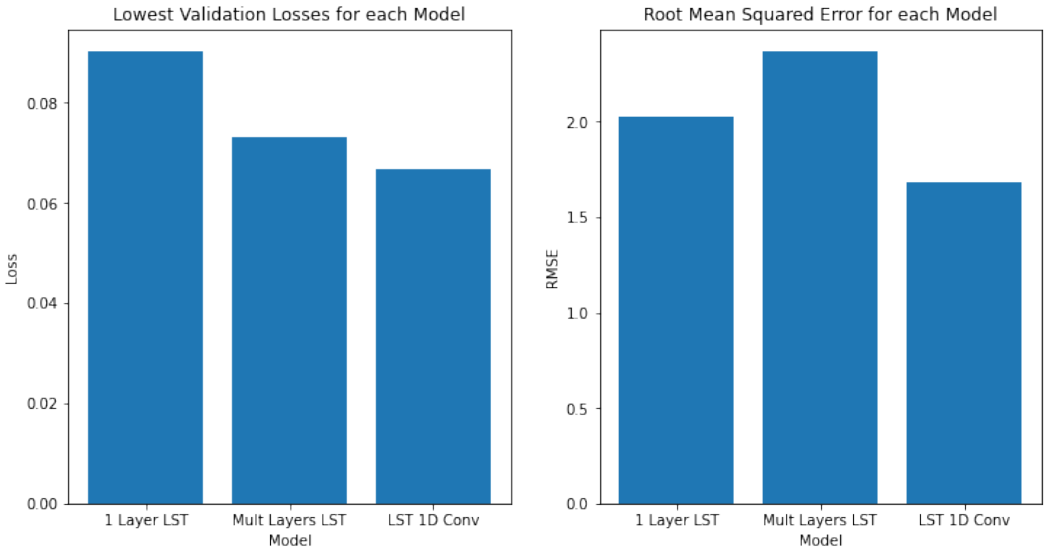
- Ordering costs: usually ordering costs are fixed or a step-wise function. It also has a high positive correlation to the variance of the orders. The higher the orders variance, the higher the ordering costs.
- Holding costs: Holding costs can be estimated with the total net inventory amount over the entire simulation period.
- Under-stock costs: When the item is out of stock, the store will suffer from loss of potential sales. In this case, under-stock costs can be estimated with the service level. The total days when net inventory less than 0 divide by entire simulation period.

5 Results

5.1 LSTM Forecasting Result

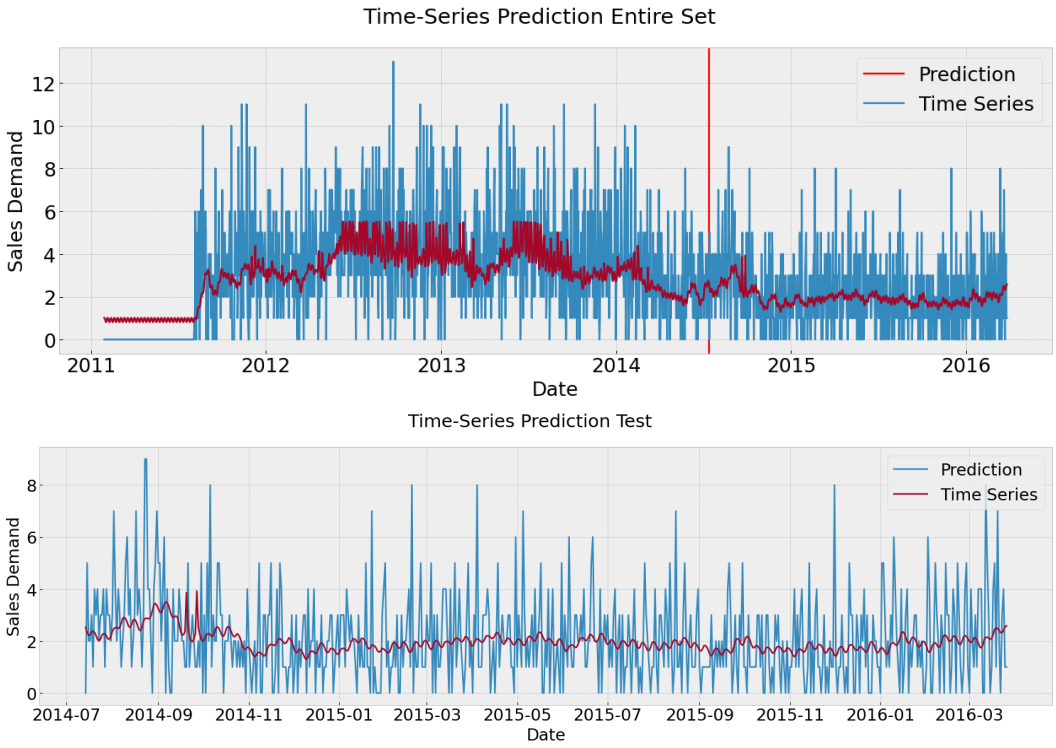
5.1.1 Evaluation

The graph presents a comparison of three different LSTM models for time series forecasting: Single Layer LSTM, Multiple Layers LSTM, and Customized LSTM with 1D Convolution. The comparison is based on two key metrics: the lowest validation loss and the RMSE.



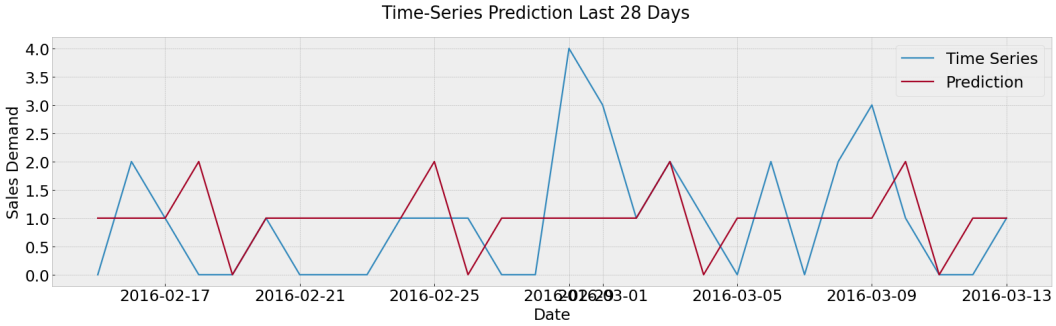
The results indicate that the customized LSTM with 1D Convolution outperformed the Single Layer LSTM and Multiple Layers LSTM models in terms of RMSE. The customized LSTM model achieved the lowest RMSE value of 1.6790483, followed by the Single Layer LSTM with an RMSE of 2.0252094, and finally, the Multiple Layers LSTM with an RMSE of 2.3665297. The customized LSTM with 1D Convolution achieved the lowest validation loss (0.0667267) in fewer epochs (49) compared to the other two models. This suggests that the customized LSTM with 1D Convolution not only performs better but also converges faster than the other models.

5.1.2 Time-series Prediction



5.2 Inventory Simulation Result

To run the inventory simulation, we used the naive Seq2Seq model described in the section 2.3. We used the last 365 days of sales to run the simulation, and all the earlier sales as the training and validation data. Below is a sample 28-day prediction sales by Seq2Seq



Following result is obtained by running simulation with lead time 5 days, and desired service level 99%.

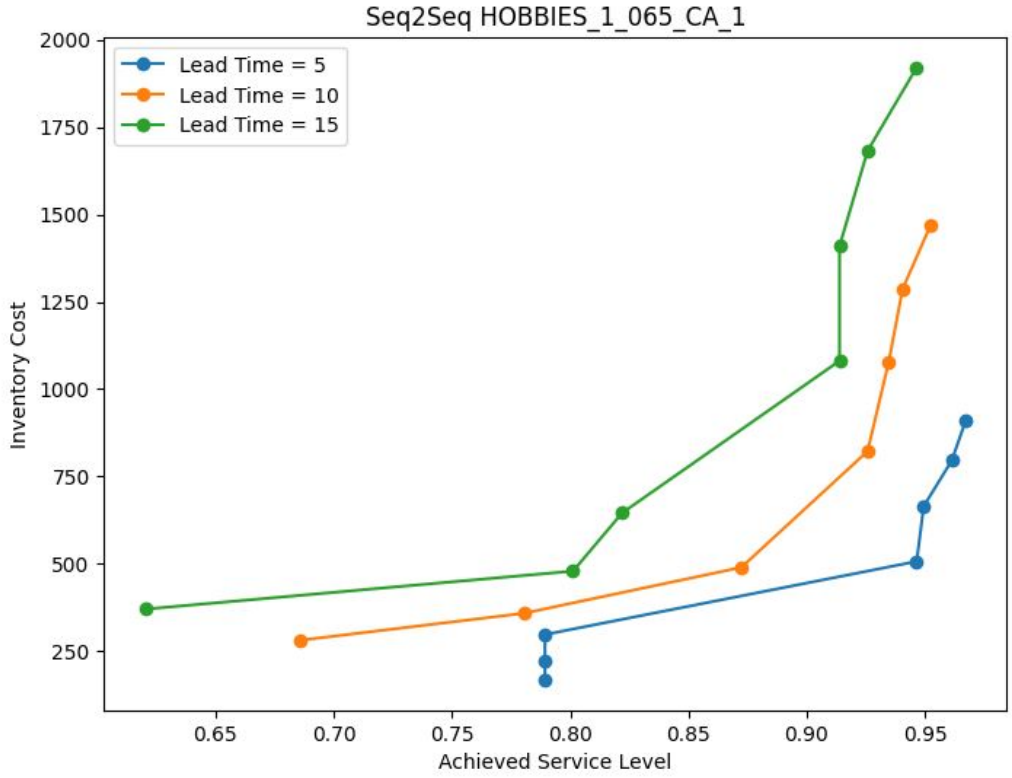
```
[ ]  simulate = SingleItemSimulate(ACTUAL_SALES, forecast, 5, desired_alpha=0.99)
     simulate.run()
     simulate.calculate_performance()

     (1262.0710600451366, 2.750434135255739, 0.8813056379821959)
```

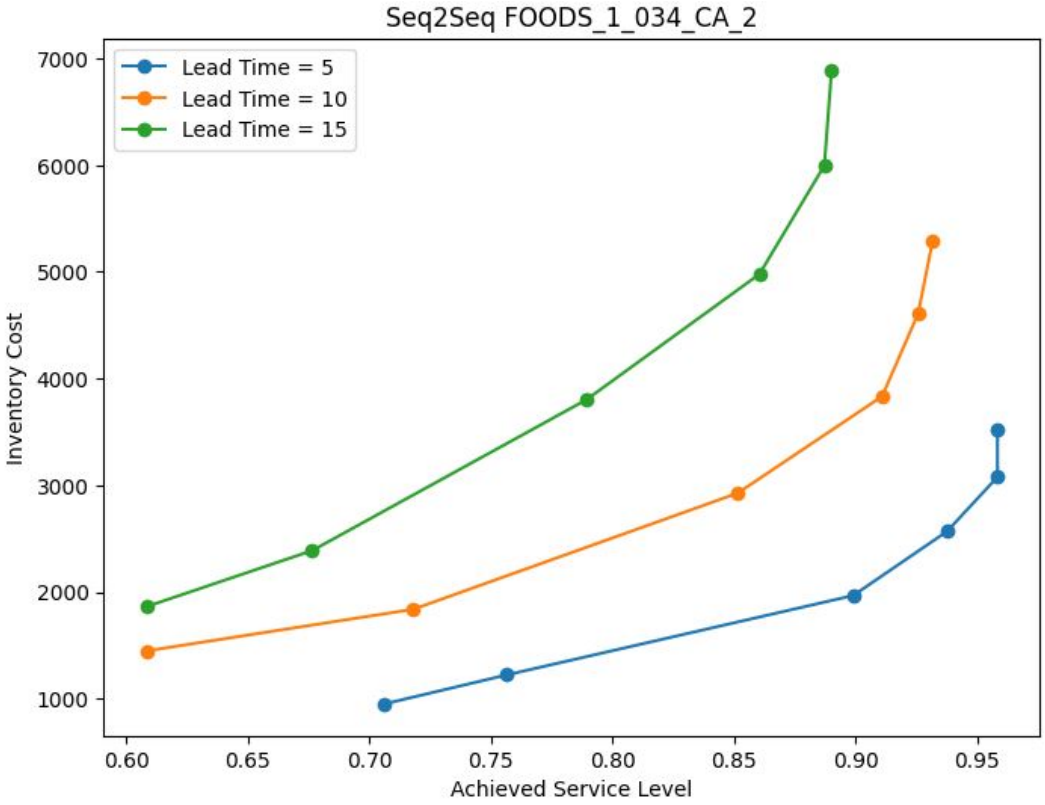
The three outputs correspond to:

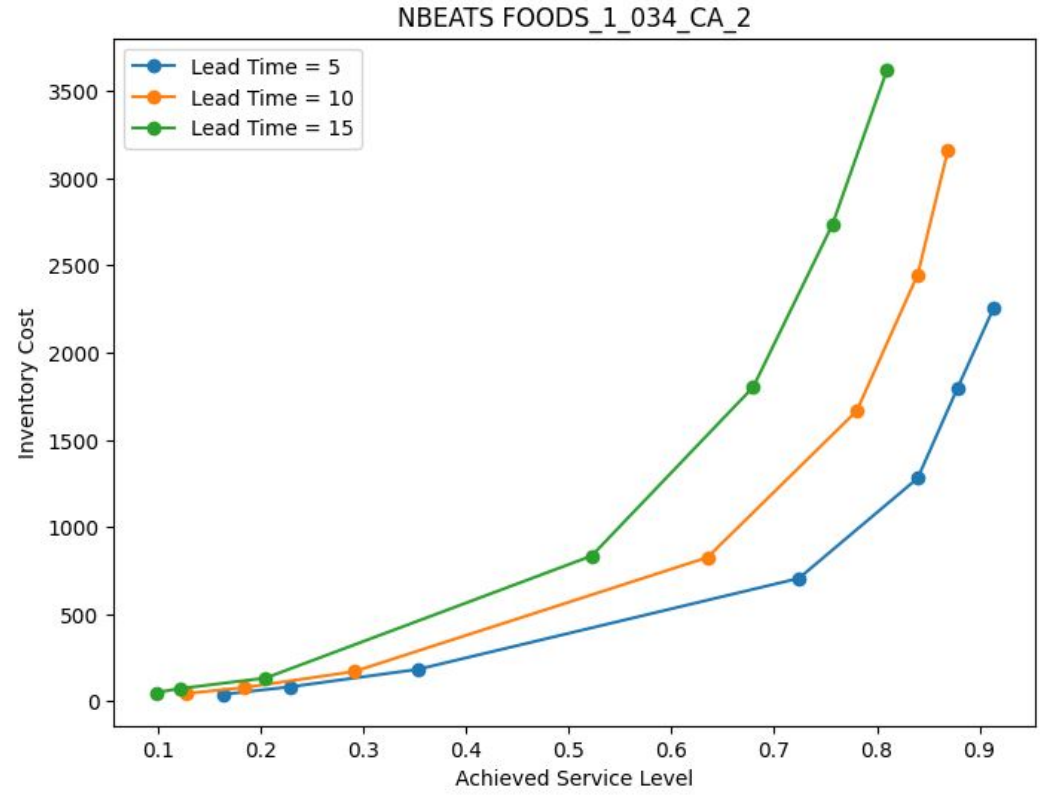
- Holding cost: 1262
- Order variance: 2.75
- Achieved service level: 88.13%

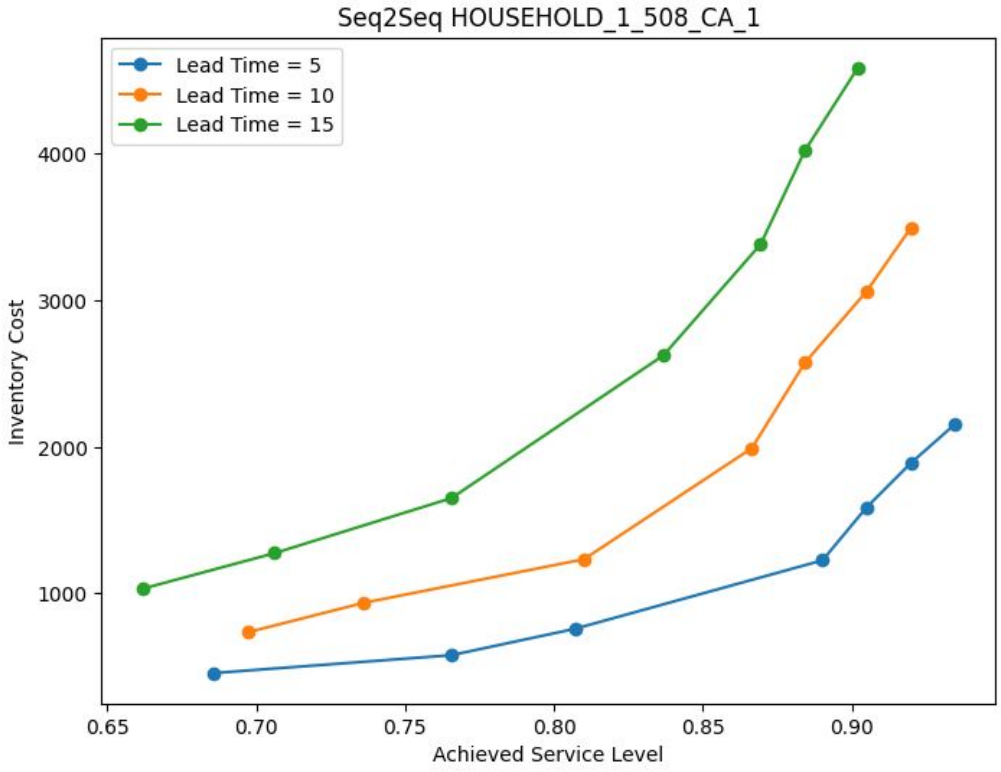
After getting the initial result, we proceed to run many simulations on two different models. In addition, since the forecast and simulation is run on a single item, we have run simulations on multiple items as well. We randomly chose three items from the hobbies, foods, and household categories (namely, HOBBIES_1_065_CA_1, FOODS_1_034_CA_2 and HOUSEHOLD_1_508_CA_1).

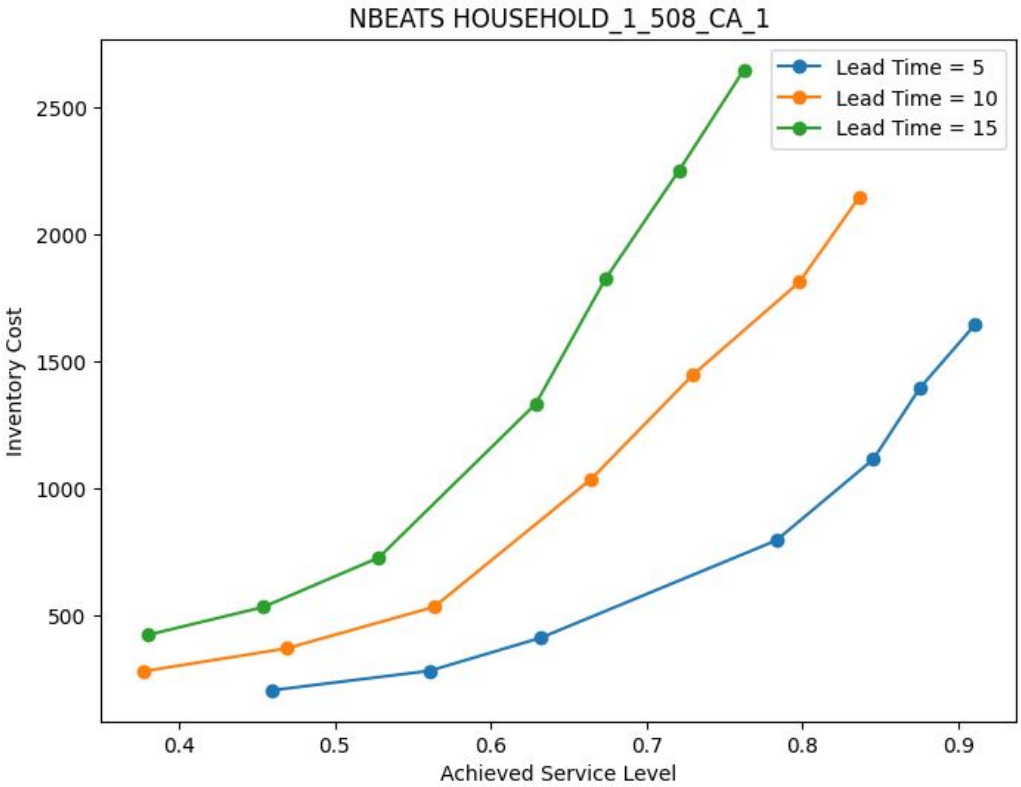
Here are the representations of inventory cost plotted against achieved service level for the items mentioned above, for both NBEATS and Seq2Seq forecasting models.




Seq2Seq FOODS_1_034_CA_2
Lead Time = 5
Lead Time = 10
Lead Time = 15
Inventory Cost
Achieved Service Level
7000
6000
5000
4000
3000
2000
1000
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95


NBEATS FOODS_1_034_CA_2
Lead Time = 5
Lead Time = 10
Lead Time = 15
Inventory Cost
Achieved Service Level
3500
3000
2500
2000
1500
1000
500
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


Seq2Seq HOUSEHOLD_1_508_CA_1
Lead Time = 5
Lead Time = 10
Lead Time = 15
Inventory Cost
4000
3000
2000
1000
0.65
0.70
0.75
0.80
0.85
0.90
Achieved Service Level


NBEATS HOUSEHOLD_1_508_CA_1
Lead Time = 5
Lead Time = 10
Lead Time = 15
Inventory Cost
2500
2000
1500
1000
500
0.4
0.5
0.6
0.7
0.8
0.9
Achieved Service Level

6 CONCLUSION

Inventory management is a critical component of the retail industry as it directly affects the profitability of the business. One of the key challenges faced by retailers is determining the optimal inventory levels to ensure that they have enough stock on hand to meet customer demand while minimizing excess inventory. Forecasting plays a crucial role in inventory management as it provides retailers with insights into the demand for their products, enabling them to make informed decisions about how much inventory to hold. Accurate demand forecasting helps retailers optimize their inventory levels by ensuring that they have the right products in the right quantities at the right time. This helps retailers avoid stockouts, which can lead to lost sales and dissatisfied customers, as well as excess inventory, which can tie up capital and increase storage costs. By forecasting demand, retailers can ensure that they have the right inventory levels to meet customer demand while minimizing the cost of carrying inventory.

Our study wish to investigate how current deep learning forecasting results impact the inventory performance. Through our research, we observed that different forecasting models have different inventory performance implications, and such subtle implications cannot be simply summarized by a RMSE calculation. Overall, the inventory simulation system has worked properly, and the results generated by the simulation system complies with the real-world knowledge. For example, as the lead time in the supply chain increases, both the inventory holding cost and ordering variability go up and the achieved service level go down.

Here are some possible future work extension of this project we believe are valuable.

- **Benchmark against traditional forecasting method**: In this project we examined the inventory performance difference between the N-BEATS and the Seq2Seq model. It could be valuable to extend the benchmark to include some of the traditional forecasting methods such as exponential smoothing. One concern that is holding modern businesses in investing in data science is that the management often does not see how value is created.
- **More robust inventory simulation**: Current version of inventory simulation is performed under many naive assumptions such as infinite inventory space and backlogged demand. One can develop a more robust inventory simulation system to account for the greater variability that is present in the supply chain.
- **Models expansion and optimization**: This project has only examined the model architectures in their most basic forms. Architectural optimizations pertained to specific models as well as increasing the model size can be performed to generalize the findings.
- **Entire store inventory evaluation**: Inventory management is usually performed at the store-level, which contains tens thousands of items. Under the real world scenario, factors such as performance also need to take into considerations.

REFERENCES

Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle C. Maddix, Ali Caner Türkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, Laurent Callot, and Tim Januschowski. Neural forecasting: Introduction and literature overview. *CoRR*, abs/2004.10240, 2020. URL https://arxiv.org/abs/2004.10240.

Gopi Durgaprasad. M5 web traffic implementation: Pytorch. Kaggle notebook, 2020. URL https://www.kaggle.com/code/gopidurgaprasad/m5-web-traffic-implementation-pytorch. Accessed on April 2, 2023.

Addison Howard and Spyros Makridakis. M5 forecasting - accuracy, 2020. URL https://kaggle.com/competitions/m5-forecasting-accuracy.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.11.013. URL https://www.sciencedirect.com/science/article/pii/S0169207021001874. Special Issue: M5 competition.

mpware. N-beats basics. https://www.kaggle.com/code/mpware/n-beats-basics, 2021. Accessed: April 30, 2023.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. *CoRR*, abs/1905.10437, 2019. URL http://arxiv.org/abs/1905.10437.

Tarun Paparaju. M5 competition - eda + models, 2020. URL https://www.kaggle.com/code/tarunpaparaju/m5-competition-eda-models. Accessed on April 2, 2023.

Fotios Petropoulos, Xun Wang, and Stephen M. Disney. The inventory performance of forecasting methods: Evidence from the m3 competition data. *International Journal of Forecasting*, 35(1):251–265, 2019. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2018.01.004. URL https://www.sciencedirect.com/science/article/pii/S0169207018300232. Special Section: Supply Chain Forecasting.

Omer Shechter. Learning pytorch lstm: Deep learning with m5 data. Kaggle notebook, 2021a. URL https://www.kaggle.com/code/omershect/learning-pytorch-lstm-deep-learning-with-m5-data. Accessed on April 2, 2023.

Omer Shechter. Learning pytorch seq2seq: Deep learning with m5 data set. Kaggle notebook, 2021b. URL https://www.kaggle.com/code/omershect/learning-pytorch-seq2seq-with-m5-data-set. Accessed on April 2, 2023.

Omer Shect. Learning pytorch seq2seq with m5 data set. https://www.kaggle.com/code/omershect/learning-pytorch-seq2seq-with-m5-data-set, 2021.

Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018. URL http://arxiv.org/abs/1808.03314.