**Abstract**

Accurate sales forecasting is crucial for inventory management, minimizing losses, and optimizing business strategies. In this study, we explore machine learning techniques to predict monthly sales of camping gear in a retail store. The dataset, sourced from the Makridakis Open Forecasting Competition (MOFC), includes hierarchical sales data from Walmart across three U.S. states—California, Texas, and Wisconsin. This dataset features product categories, store details, price fluctuations, promotions, and external factors such as special events.

To improve forecasting accuracy, we implement and compare two models: **LightGBM**, a gradient boosting method known for its efficiency in handling structured data, and **LSTM**, a deep learning approach designed to capture long-term dependencies in sequential data. The study involves preprocessing steps such as feature engineering, scaling, and time-series transformation. The models are trained and evaluated using the Weighted Root Mean Squared Scaled Error (RMSSE), the competition’s primary metric.

This comparative analysis highlights the strengths and limitations of each model in the context of retail sales forecasting. The findings aim to contribute to the ongoing development of forecasting methodologies, helping businesses improve decision-making, reduce uncertainty, and optimize inventory management.

**Introduction**

This document outlines the implementation and comparison of two machine learning techniques: **LightGBM (Gradient Boosting)** and **LSTM (Long Short-Term Memory)**. The goal is to analyze their effectiveness in forecasting monthly sales of camping gear in a retail store and justify the selection of the most suitable model. Sales forecasting at this level requires an understanding of historical sales trends, seasonal influences, promotions, and external factors affecting consumer demand.

**EDA (Exploratory Data Analysis)**

To gain insights into sales trends and potential influencing factors across different stores, I conducted an **Exploratory Data Analysis (EDA)** using a **sample of 10% of the dataset** for efficiency. The analysis began with a **missing values assessment**, where a heatmap revealed gaps in certain features, particularly event-related columns. These missing values may require **imputation or exclusion**, depending on their impact on model performance. Next, I examined the **distribution of key numerical variables** such as sales, sell prices, and time-related features. The results showed that **sales were highly skewed**, with most transactions involving small quantities, while a few large sales acted as outliers. This suggests that **some products or stores experience bulk purchases**, which could impact forecasting accuracy.

Categorical variables such as **weekdays, events, and SNAP program participation** were analyzed, revealing that **certain events occur infrequently but can significantly boost sales**, suggesting that event-driven promotions could be leveraged for sales optimization. A **correlation analysis** indicated a weak relationship between **sell price and sales**, implying that **pricing alone may not be the primary driver of demand**. Instead, external factors such as **seasonality, promotions, and local economic conditions** could play a more significant role.

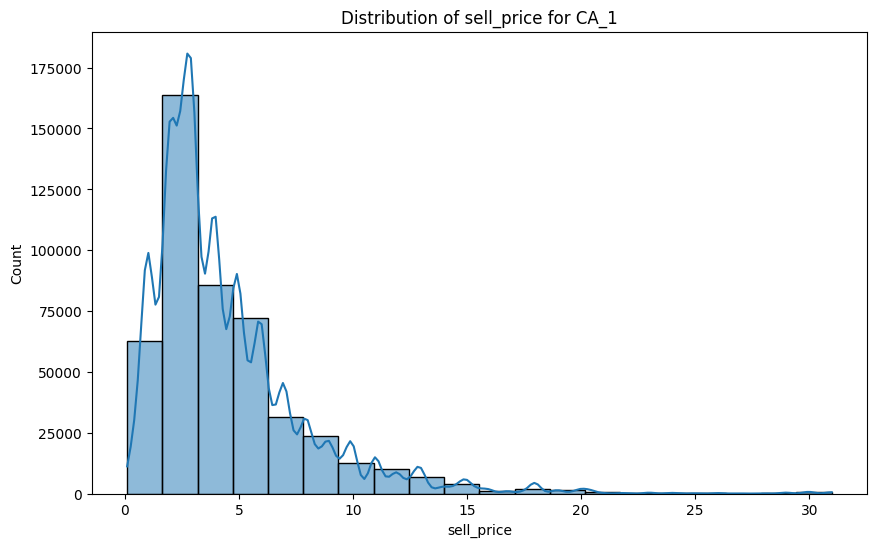
A **time series analysis** across stores demonstrated that sales exhibited **seasonal fluctuations**, with peaks around holidays and promotional events. This suggests that **seasonal forecasting methods should be considered when predicting future sales**. Additionally, a **boxplot analysis of sales** revealed the presence of **outliers**, likely due to bulk purchases or anomalies. These outliers may need to be **handled appropriately during model training** to avoid skewing predictions.

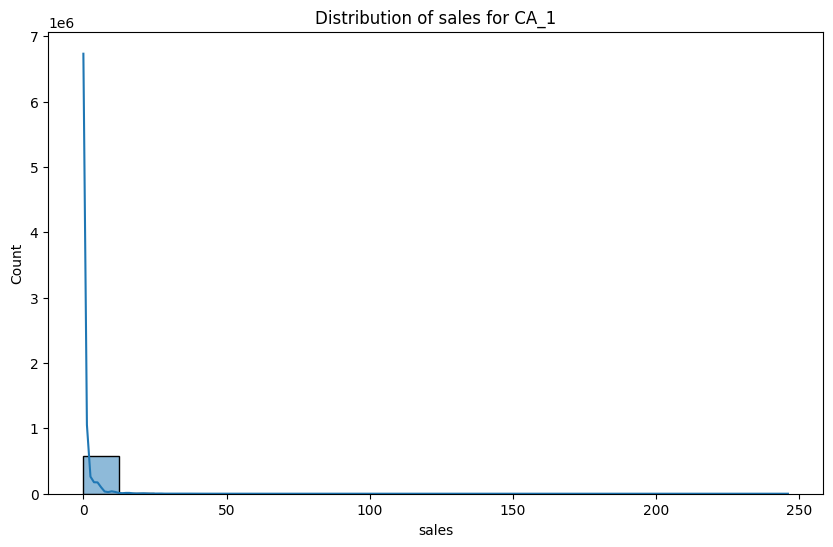
### **Recommendations:**

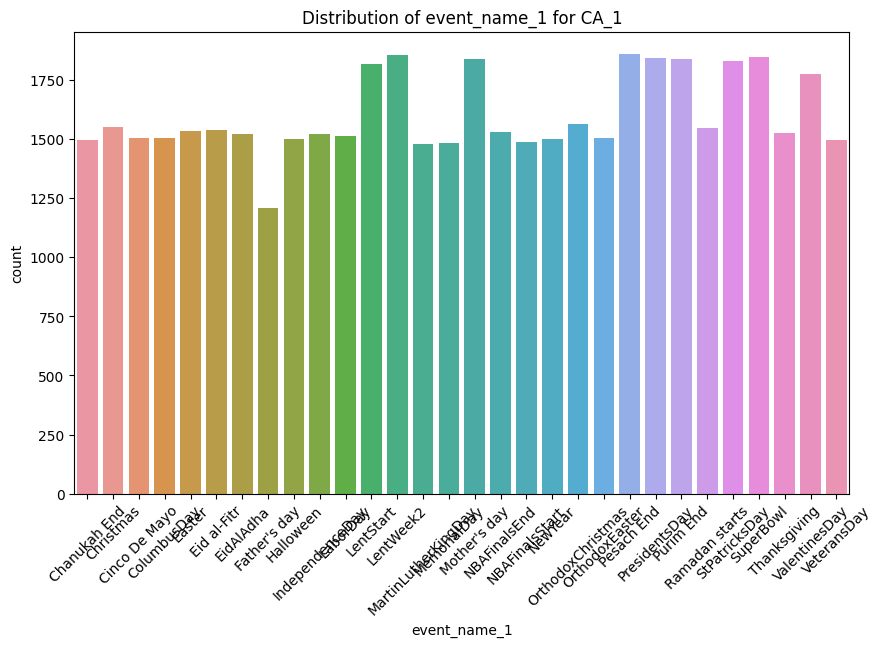
1. **Handle Missing Values:** Investigate the cause of missing values, particularly in event-related features, and determine whether imputation (e.g., filling with "No Event") or exclusion is appropriate.
2. **Outlier Treatment:** Consider using techniques such as **capping extreme values** or **log transformations** to prevent large purchases from distorting the model.
3. **Feature Engineering:** Since pricing alone is not a strong predictor, incorporate **event-based features, seasonal trends, and external economic indicators** into the model to improve accuracy.
4. **Time Series Modeling:** Utilize **seasonal decomposition and moving averages** to identify underlying patterns in sales data for better forecasting.
5. **Targeted Promotions:** Since certain events significantly impact sales, **strategic pricing adjustments or promotions** during high-sales periods could maximize revenue.
6. **Store-Specific Modeling:** Given variations in sales trends across stores, consider developing **store-specific models or clustering stores based on similar sales patterns** for better prediction accuracy.

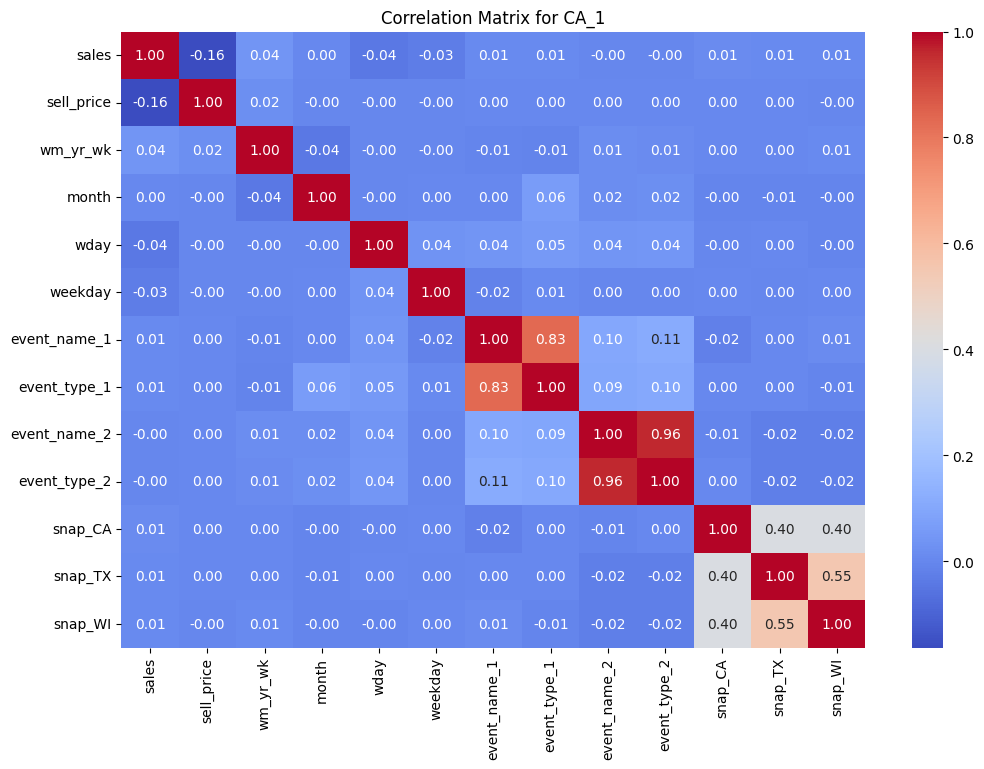
These insights provide a strong foundation for further **predictive modeling** and strategic decision-making, ensuring more accurate demand forecasting and optimized sales strategies.

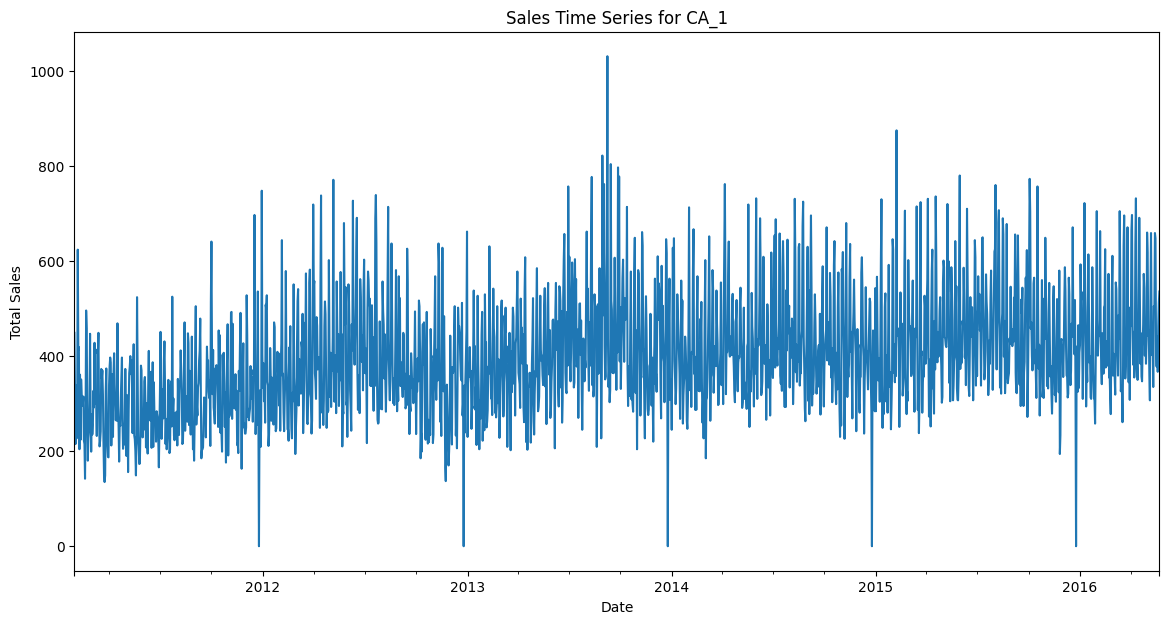
Since most of the results were similar, only the data visualizations for store\_id CA\_1 will be shown.











Another note to add is that because the sale prices are heavily skewed left, meaning a lot of the purchases from that store were cheap, it’s possible both the LightGBM and LSTM models would be biased towards those kinds of purchases. Meaning, it might not be able to capture relationships between very cheap products and very expensive products. Further hyperparameter tuning and/or data cleaning might be needed to make sure the models can handle larger price ranges.

**Data Preprocessing**

Before applying the models, the dataset undergoes the following preprocessing steps:

* **Data Cleaning**: Handling missing values and removing outliers. I used primarily forward fill with the mean of that specific parameter (column).
  + Another thing to note is that I aggregated the 4 datasets that came with the competition based on the store\_id, meaning which store that value represented.
* **Feature Engineering**: Creating meaningful features, including time-based variables, event indicators, and lag-based sales features, to improve model performance.
* **Encoding Categorical Variables**: Using label encoding and one-hot encoding where necessary for product categories and store details.
* **Scaling and Normalization**: Preparing the data for deep learning models (LSTM) by ensuring consistent feature distributions. Scaling is necessary when working with deep learning techniques instead of traditional machine learning ones such as LightGBM.
* **Train-Test Splitting**: Dividing the data into training and testing sets while maintaining the time-series structure.
* **Time-Series Preparation (for LSTM)**: Transforming data into sequences with appropriate windowing techniques to capture long-term dependencies. I made sequences of 28 to represent 28 days of sales within a month.

**Techniques Used**

The implementation of both models involves the following key techniques:

### **LightGBM Techniques:**

1. **Gradient Boosting Decision Trees (GBDT)**: LightGBM is based on GBDT, which iteratively improves weak learners.
2. **Hyperparameter Optimization**: Tuning parameters like learning rate, number of leaves, and boosting rounds to optimize performance.
3. **Early Stopping**: Preventing overfitting by stopping training when validation loss stops improving.
4. **Train-Validation Split**: Dividing data into training and validation sets instead of K-fold cross-validation to maintain time-series integrity.

### **LSTM Techniques:**

1. **Custom RMSSE metric calculation:** LSTM(s) do not have a built-in metric to calculate this type of error, so a custom function was created to calculate the RMSSE. RMSSE is a way to measure how bad or good a forecast (prediction) is, **while accounting for the scale of the data**. It helps us compare errors fairly, even if the values are big or small. The formula to calculate it is shown below:

In simple terms:

1. **Find how wrong the predictions are**
   1. Take each actual value (​) and predicted value (​).
   2. Find the difference between them.
   3. Square the differences (to remove negative signs and give more importance to bigger errors).
   4. Take the average.  
       → This gives the **numerator (top part)** of the formula.
2. **Find how much the training data naturally changes**
   1. Look at how the real values in the training data change from one point to the next.
   2. Compute the difference between each point and the previous one (​)
   3. Square these differences and take the average.  
       → This gives the **denominator (bottom part)** of the formula.
3. **Take the square root of their ratio**
   1. This adjusts the final result so that it's more in line with real values instead of squared errors.

The reasoning behind why we do this is that if the denominator is large, the data is most likely naturally changing over time, so errors are less serious. If the denominator is **small**, it means the data is usually stable, so even small errors matter more.

1. **Dropout Regularization**: Preventing overfitting by randomly dropping neurons during training.
2. **Optimizer Tuning**: Using Adam optimizer with learning rate adjustments to improve convergence.

## **LightGBM Implementation**

**LightGBM**, a gradient boosting framework, is implemented for structured data analysis. Key steps include:

1. **Feature Selection**: Identifying relevant features, such as past sales, price changes, and promotional periods, to enhance model performance.
2. **Hyperparameter Tuning**: Using techniques like grid search or Bayesian optimization to optimize performance.
3. **Training and Evaluation**: Training the model and evaluating it using Weighted Root Mean Squared Scaled Error (RMSSE), the primary metric for this forecasting problem.
4. **Interpretability**: Utilizing feature importance scores to understand model decisions and identify key sales drivers.

### **Advantages of LightGBM:**

* Efficient for tabular data with categorical and numerical features.
* Faster training time compared to deep learning models.
* Good performance with relatively small to medium-sized datasets.

## **LSTM Implementation**

**LSTM**, a type of recurrent neural network (RNN), is implemented for sequential data prediction. Key steps include:

1. **Data Reshaping**: Transforming data into a format suitable for time series analysis.
2. **Model Architecture Design**: Defining LSTM layers, dropout layers, and dense output layers tailored for sales forecasting.
3. **Training and Optimization**: Using backpropagation and optimizers like Adam to minimize loss.
4. **Evaluation and Predictions**: Assessing model performance using RMSSE and visualizing predicted versus actual sales trends.

### **Advantages of LSTM:**

* Effective for time-series and sequential data prediction.
* Can learn long-term dependencies in sales data, capturing seasonal trends and promotional impacts.
* Handles non-linearity better than traditional models.

## **Comparison and Model Selection**

| **Criteria** | **LightGBM** | **LSTM** |
| --- | --- | --- |
| Training Speed | Faster | Slower |
| Handling of Sequential Data | Limited | Excellent |
| Interpretability | High | Low |
| Accuracy for Structured Data | High | Variable |
| Requirement of Large Data | No | Yes |

Given the hierarchical sales data structure and the requirement for short-term forecasting, **LightGBM** is highly effective in capturing relationships between product sales and influencing factors. However, **LSTM** provides a deeper understanding of temporal dependencies, making it beneficial for long-range forecasting. The final choice depends on the dataset characteristics, computational constraints, and forecast horizon.

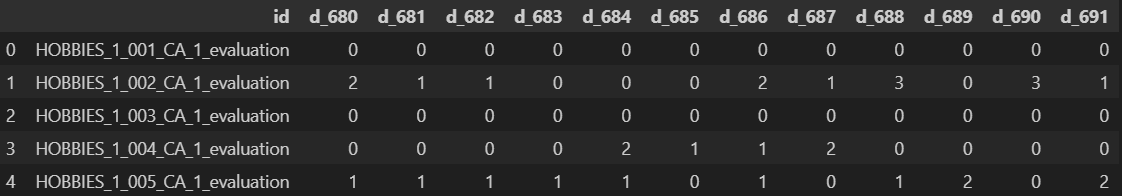
**Results**

**Naive Solution**

I first created some baseline results using a naive method by following the steps below:

1. **Read** the wide sales\_train\_evaluation.csv, which has 30,490 rows (\_evaluation IDs).
2. **Compute** the average sales for the last 28 days (d\_1914 ~ d\_1941) for each row, calling that naive\_mean.
3. **Replicate** those rows as \_validation by replacing \_evaluation with \_validation in the id string.
4. **Fill** F1..F28 for both \_validation and \_evaluation rows with the same naive\_mean.
5. **Combine** them into one final submission.csv containing 60,980 rows (30,490 \_validation + 30,490 \_evaluation).

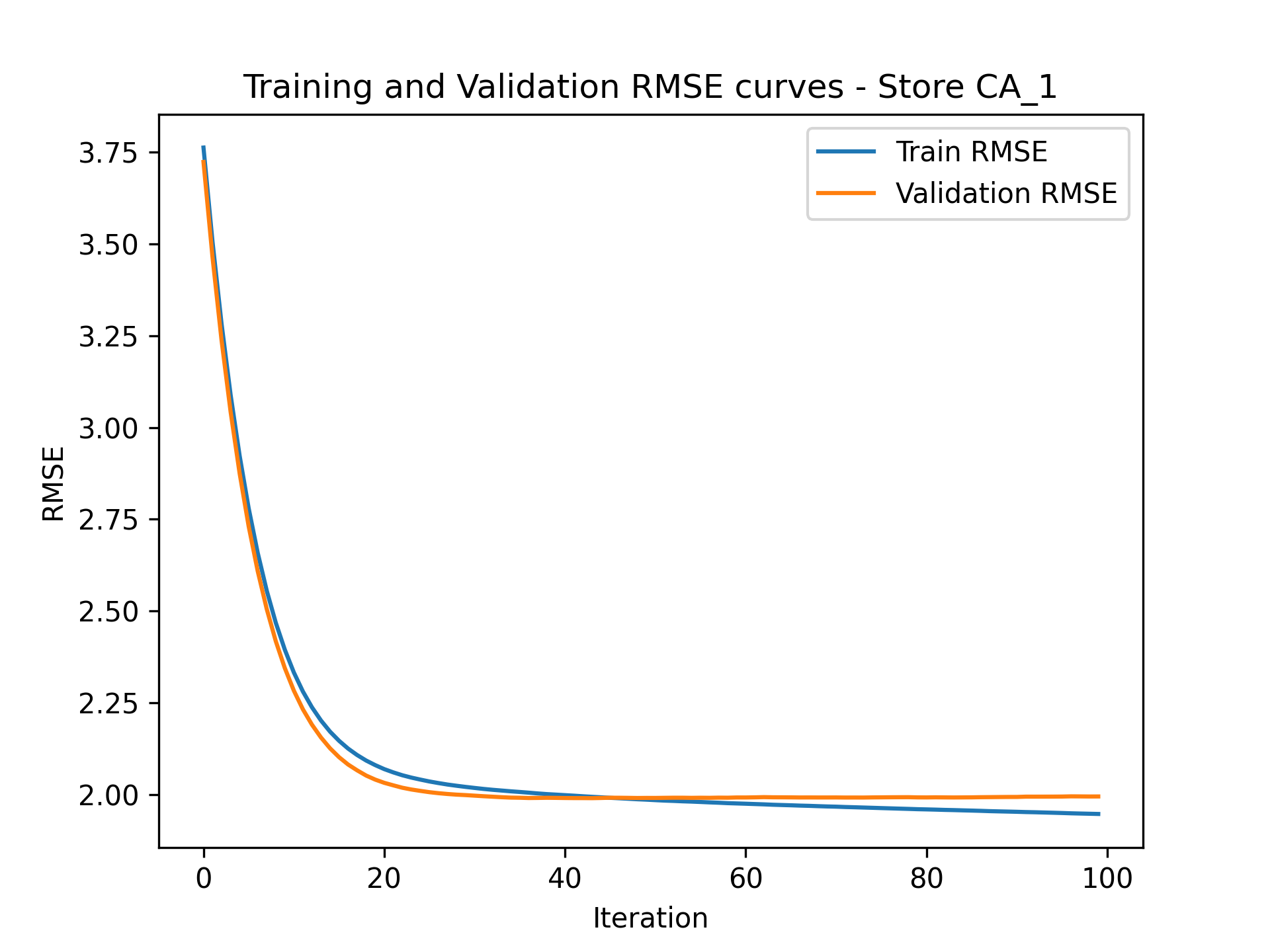
Part of the results are shown below. This was just to get an idea about how the submission file is supposed to be structured and also how to calculate the average sales for the previous 28 days, which is important for the next few experiments.

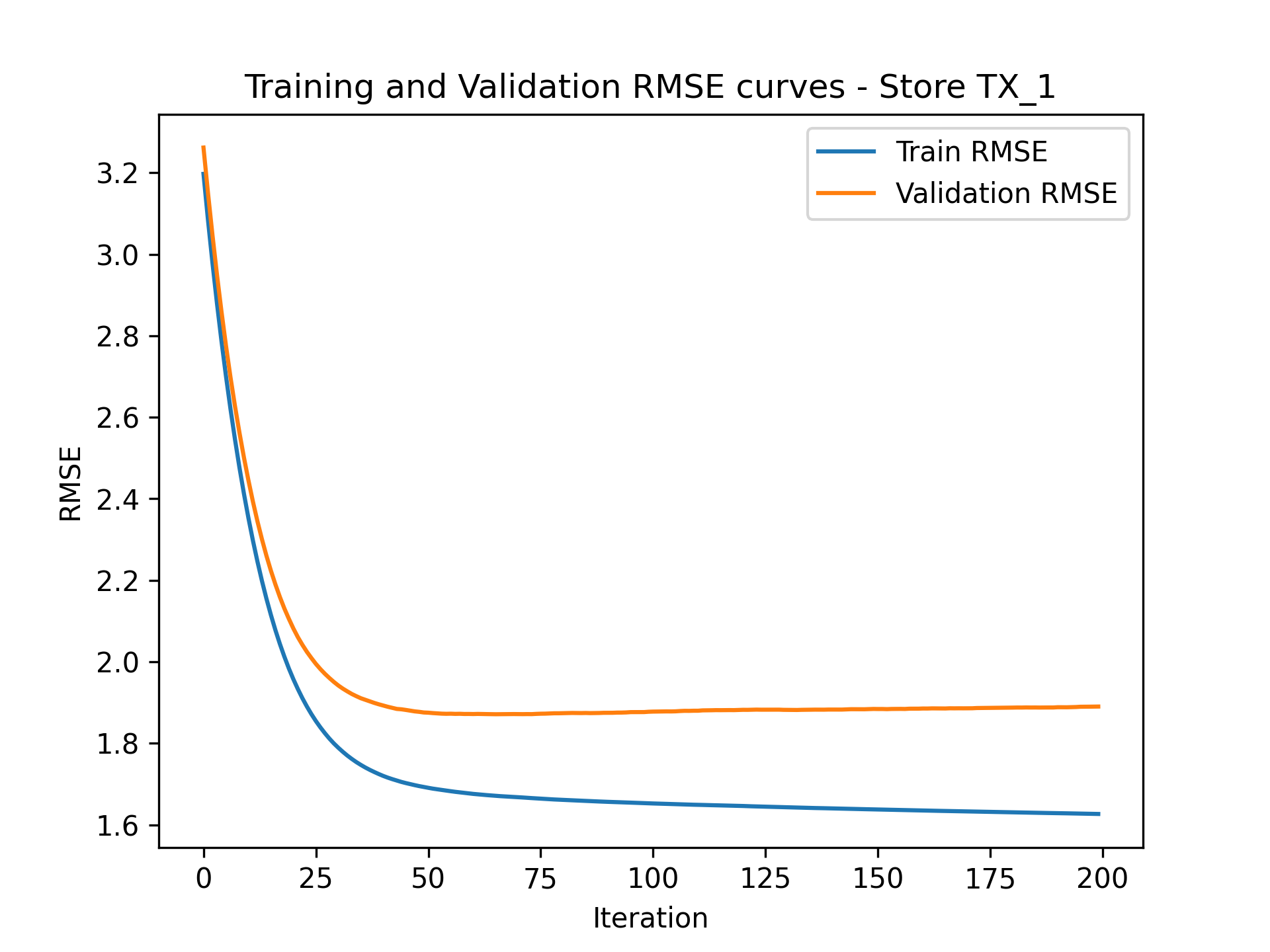


**LGBM Solution**

The results were very promising for the LGBM model. I barely had to do any hyperparameter tuning in order to make the model efficient. During training, I tried my best to clear memory after going through each store’s data, because I was dealing with very large datasets. The validation curve for LGBM indicates minimal overfitting, suggesting that the chosen hyperparameters were well-tuned. The model's ability to predict demand with high accuracy makes it a strong candidate for time series forecasting in retail applications. Further improvements could involve refining hyperparameters further or incorporating additional external features to enhance predictive power.

The validation curves for store CA\_1 and TX\_1 are shown below:





Due to memory limitations, extensive time required for a wide hyperparameter search, and resource constraints, LSTM training could not be completed. As a result, a direct comparison between the LGBM and LSTM models was not possible. However, the LGBM model demonstrated strong performance, closely matching the validation data and generalizing well. The model effectively captured trends and patterns in the dataset, reinforcing its suitability for the M5 Forecasting Accuracy competition.

For the M5 Forecasting Accuracy competition, I implemented an LSTM-based model to predict future sales data. My approach involved preparing time-series data by engineering relevant lag-based and rolling window features before normalizing inputs for efficient training. I designed an LSTM model with hyperparameter tuning using Keras Tuner to optimize key parameters such as the number of LSTM layers, units per layer, dense layer size, and learning rate. To enhance evaluation, I incorporated RMSSE (Root Mean Squared Scaled Error) as a custom metric to better reflect forecasting accuracy compared to traditional RMSE. The model was trained separately on data for different stores, with predictions compiled into a submission file. To assess and compare results, I could have evaluated the LSTM model against a LightGBM model by comparing validation RMSSE scores, forecasting performance across different time horizons, and visualizing actual vs. predicted sales trends. Additionally, I could have performed backtesting by training the model on earlier sales periods and assessing prediction accuracy on a held-out validation set.

**Conclusion**

In this analysis, we explored different modeling approaches for the M5 Forecasting Accuracy competition, comparing LightGBM and LSTM models. Despite the potential of LSTMs for capturing temporal dependencies, practical limitations such as memory constraints and extensive hyperparameter tuning prevented their full implementation. Conversely, LightGBM demonstrated strong performance with efficient training and generalization, closely matching validation data. This study highlights the importance of balancing accuracy with computational feasibility when selecting forecasting models. Future improvements could include optimizing deep learning architectures or leveraging hybrid models to enhance predictive accuracy while maintaining efficiency.