Forecasting Retail Demand at Scale

Team: Aashi Patni, Apurva Baru, James Ashwin, Olivia Boe, & Sameeksha Mohan

1. Introduction

This document presents a structured overview of our modeling approach for the M5 Forecasting competition. It begins with a summary of the models explored and their corresponding hyperparameter configurations (Table 1), followed by a detailed rationale behind the selected features and feature engineering techniques used to enhance forecasting performance (Table 2). Lastly, we provide a comparative evaluation of model performance using key metrics to determine the most effective approach (Table 3). Together, these sections capture the key design decisions, iterative experimentation, and insights that guided our solution strategy.

2. Model Architectures and Hyperparameter Configurations

Model	Hyper Parameters
LightGBM (Baseline)	**Objective:** regression **Learning Rate**: 0.05 **Subsample**: 0.7 **Colsample_bytree**: 0.7 **Num Leaves**: 128 **Verbose**: -1
LightGBM (Improved)	**Objective**: poisson **Force Row Wise**: True **Learning Rate**: 0.01 **Num Iterations**: 3000 **Sub Row**: 0.85 **Bagging Freq**: 1 **Lambda L1**: 0.05 **Lambda L2**: 0.2 **Num Leaves**: 128 **Min Data In Leaf**: 100 **Verbosity**: 1
XGBoost	**N_estimators**: 150 **Max Depth**: 4 **Learning Rate**: 0.05 **Subsample**: 0.75 **Colsample_bytree**: 0.75 **Tree_method**: hist **N_jobs**: -1
Keras GRU	**Dropout**: 0.3 **Optimizer**: Adam

	Epochs: 4 **Batch Size**: 64 **Patience**: 5 **Restore Best Weights**: True
Keras LSTM	**Epochs**: 10 **Batch Size**: 64 **Optimizer**: Adam **Patience**: 2 **Min Learning Rate**: 1e-5 **Restore Best Weights**: True **Early Stopping**: True

Table 1. Different models that were explored with their hyperparameters

3. Feature Engineering Overview and Rationale

To effectively model retail demand patterns in the M5 Forecasting task, we engineered a set of temporal, statistical, categorical, and economic features based on domain knowledge and literature-backed practices. Our goal was to capture both short-term variations and long-term seasonality while ensuring the feature space remained scalable for millions of time-series entries. This section outlines the key features generated across both our LightGBM-based and GRU-based models, along with the rationale behind their inclusion.

Feature Category	Feature Names	Description	Rationale
Lag Features	lag_7, lag_28	Sales values from 7 and 28 days ago	Captures **weekly and monthly seasonality**, critical for forecasting retail demand
Rolling Statistics	rmean_7_7, rmean_28_7, rmean_7_28, rmean_28_28	Rolling mean of lag_7 and lag_28 with windows of 7 and 28 days	Smooths lag features, captures recent trends and **momentum effects** in sales
Date Features	wday, week, mday, month, quarter, year	Extracted from the calendar date	Captures **cyclical patterns** like day-of-week, month-end effects, and yearly trends
Categorical Encodings	item_id, dept_id, store_id, cat_id, state_id, event_name_1, event_type_1, snap_CA, etc.	Label-encoded or used as categorical variables	Enables the model to learn **store/item-specific** or **state/event-based** demand behaviors

Price Feature	sell_price	Actual product price	Introduces **price elasticity of demand**, an essential economic signal
Memory Optimization	Downcasting float64 to float32, converting IDs to category	Reduces memory usage significantly	Important when training on **tens of millions** of rows without crashing the kernel

Table 2. Overview of feature engineering and the rationale behind them

4. Performance Evaluation Across Model Variants

The metric for our models is **Weighted Root Mean Squared Scaled Error (WRMSSE)**

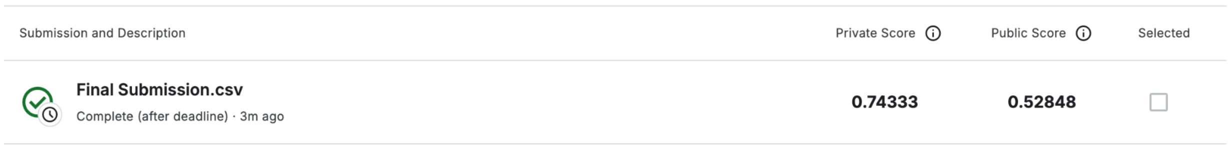
Model	WRMSSE Score
LightGBM (Baseline)	3.91
LightGBM (Improved)	**0.76**
XGBoost	5.39
Keras GRU	1.16
Keras LSTM	5.12

Table 3. Model performance comparison

We stacked **XGBoost** and **LightGBM** due to their shared tree-based structure, which allows for a unified preprocessing pipeline and streamlined tuning. In contrast, GRU requires separate sequence-based preprocessing, adding complexity without proportional gains in efficiency or robustness.

5. Results

The WRMSSE of our stacked model can be found below:



Submission and Description	Private Score	Public Score	Selected
Final Submission.csv Complete (after deadline) · 3m ago	0.74333	0.52848	☐

6. Closing Thoughts

Our final stacked model, combining LightGBM and XGBoost, delivered strong forecasting accuracy while maintaining scalability and ease of deployment. Although GRU performed well individually, its integration added complexity without significant gains in ensemble performance.

This approach highlights the value of combining robust feature engineering with efficient machine learning to support better inventory, pricing, and supply chain decisions.