VN1 Forecasting Competition: Exploration of LGBM

1 Introduction

This document details my exploration of the VN1 Forecasting Competition, as well as popular Statistical/ML models in demand forecasting. The VN1 Competition covered 11.1k products, 46 clients, and 328 warehouses. Contestants were provided about three years of data at a weekly grain. Given a dataset of ~3.5 years (July 2020 - December 2023), the objective was to create a 13-week forecast starting January 2024. Submissions were ranked by a competition score of Mean Absolute Error % + |Bias|%. Prize money totaled $20k.

2 Motivation

While I was familiar with classical Statistical models like Exponential Triple Smoothing (ETS) and Seasonal AutoRegressive Integrated Moving Average (SARIMA), I had not yet used tree-based, gradient boosting forecasting models like XGBOOST or LGBM. In addition, most of my work had been in R and I wanted to refamiliarize myself with Python.

3 Statistical Learning vs Machine Learning

The main difference between these paradigms is that with Statistical Learning, one is choosing a model with a pre-defined mathematical shape, while in Machine Learning, the model learns potential complex, non-linear patterns. Machine Learning models offer two advantages to Statistical models for this competition; speed and the global-training structure. While SARIMA individually forecasts each “unique id” (product-client-warehouse), for LGBM, information is shared between unique ids. LGBM has had a lot of success in forecasting competitions (e.g., M5), hence my choosing the model for this project.

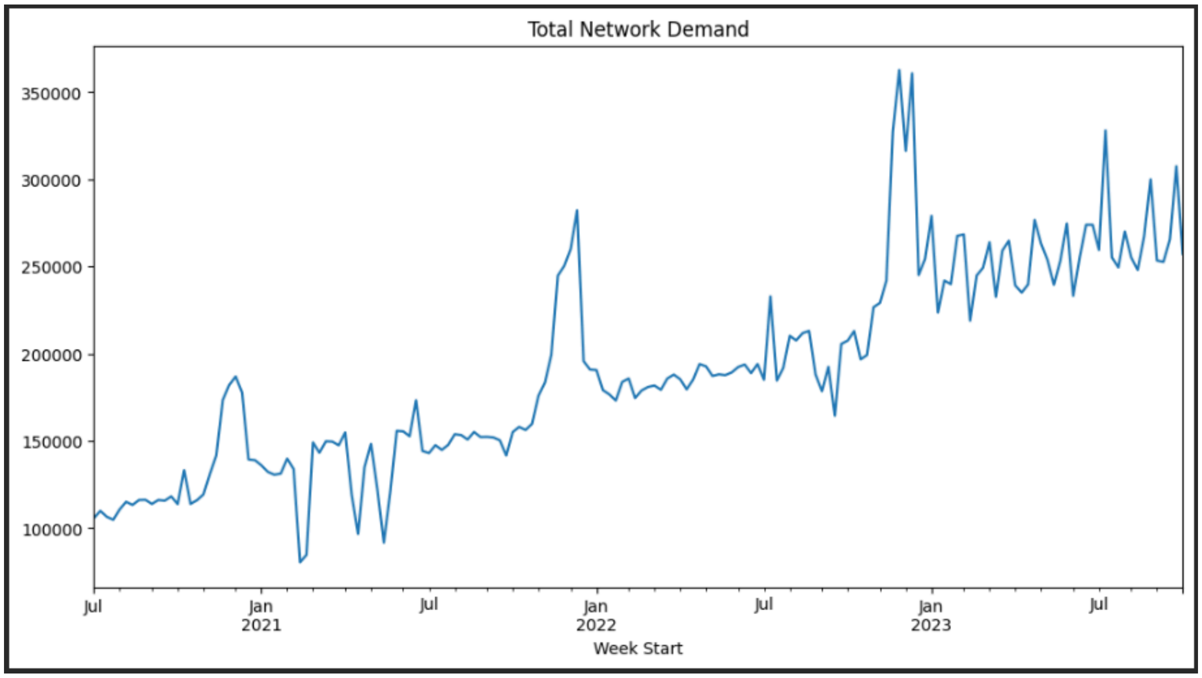
4 What is LGBM doing?

LGBM is a gradient boosting framework. It trains one model to start, gets the residuals (errors) from that model, and then creates a new model to predict those errors. This process can repeat many times, sequentially. The essence is that many “weak” models make a strong model through ensembling. I choose a recursive implementation, where one predicts the next week, and then feeds that observation (and its errors) back into the model to generate the second week.

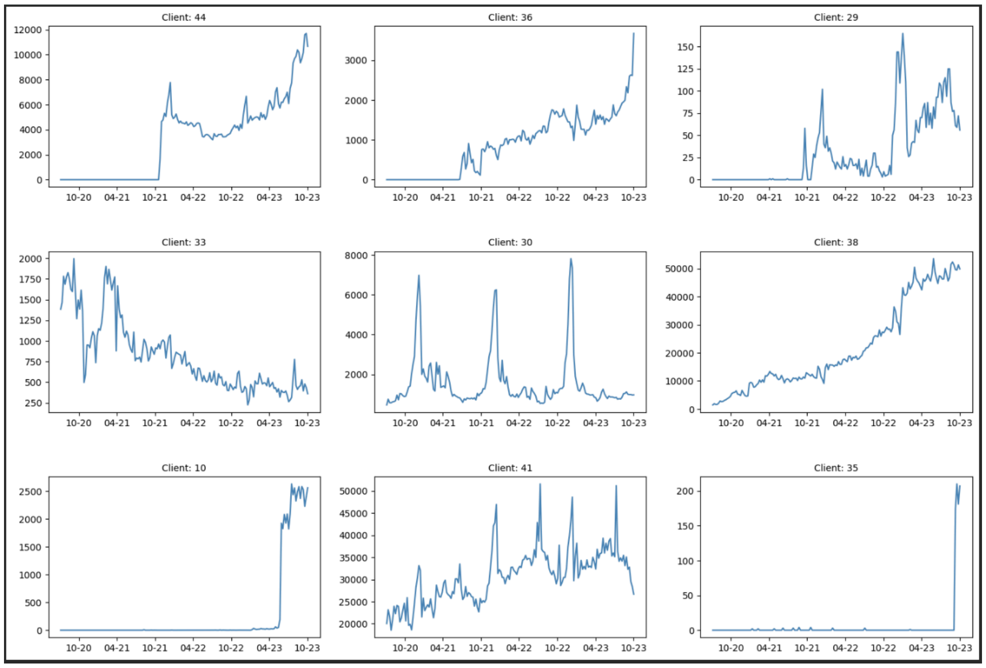
email: dan.morrissey1@gmail.com

5 Exploratory Data Analysis

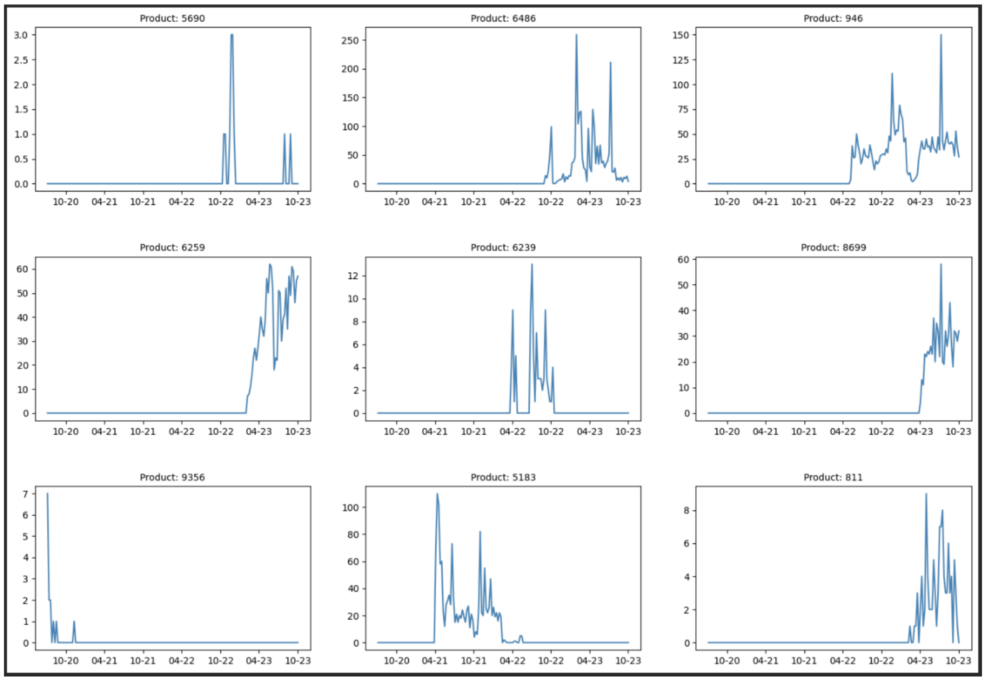
There are 168 million unique ids (11.1k products x 46 clients x 328 warehouses). Based on this, I wanted to first plot demand at different hierarchies (overall-level, product-level, warehouse-level, etc). To do this, I wrote a function which outputs 9 random time series at a pre-selected hierarchy.



This plot shows the total demand for all 168m unique ids, where one can observe an increasing trend over three years and strong seasonality in Q4. I needed to make sure to include features in the model for seasonality (e.g., lag 52, etc).



The grid to the left shows 9 random client time plots. You can observe the seasonal spikes and the general increasing trends.



My takeaway at the product level is that most of the time series are zero-inflated, and I would like to include this in feature engineering.

6 Establishing Benchmarks / Theta

It's common practice to fit basic models to start to get a sense of the variability and what "good" might look like. Over a 13 week backtest, Theta performed the best out of a suite of benchmark models implemented using the StatsForecast package from Nixtla. Its competition score was 56%, while a 4 week moving average was 64%. As Theta is a good model for decomposing a time series and picking up on seasonality, my EDA observations lead me to incorporate it in ensembling.

7 Feature Engineering

One important component to tree-based models is that we have to create features. While I did create features manually for a backtesting loop, I ended up using the MLForecast package from Nixtla to create features and do backtests. As Theta was picking up on seasonality, I only incorporated a few lags and rolling mean features. I also included Product, Warehouse, and Client as features. We were given historical data on pricing, however pricing was null in absence of a sale. To help with this I filled in the last known price for a given unique id. To help with associating shifts in price with shifts in demand, I calculated the change between the lag_1 and lag_2 prices. However, when adding this to LGBM, I did not see a benefit in the backtesting and so I excluded it. I did not spend a lot of time tuning the parameters on LGBM, given more time I would run a random search on a range of parameters.

8 Ensembling / Performance

As different types of forecasts create different types of errors, ensembling forecasts from different models is a common method to improve overall performance. Within the Phase 1 test set, I explored a few different weights manually, and landed on using a 70% weighting for Theta and 30% weighting for LGBM, which provided a MAPE + Bias of 53.5% in Phase 1 (1150 bps better than a moving average, 400 bps worse than the leader). If I had more complicated inputs, I could have run an optimization function to find the optimal weighting of models. When running this combination on Phase 2, which was a 13 week sample and the final test set, I saw a MAPE + Bias of 54.1%.

9 Reflections

I completed this project in a few weeks and given more time I would spend it on more EDA, feature engineering, and exploring different parameter tunings. I would also make forecast adjustments for outliers and treat products with multiple weeks without a sale differently.

email: dan.morrissey1@gmail.com