Fine Grained Demand Forecasting & Predictive Analytics Using Prophet & XGBoost ML Models

Minh Pham

KSU MS Data Analytics Student

Abstract

This study aims to utilize machine learning models to predict fine grained demand forecasts on a historical sales data set. The ML models that will be compared for forecast accuracy are Prophet and XGBoost.

1. Introduction

Demand forecasting plays a critical role in supply chain management by enabling businesses to anticipate future demand, optimize inventory levels, and streamline production processes. According to McKinsey and Company, improving forecasting accuracy by 10-20% can potentially reduce inventory costs by 5%, raising overall revenue by 2-3% (McKinsey Report, 2018). As supply chains become increasingly complex, the need for scalable and accurate demand forecasting solutions is more critical than ever.

The goal of this project is to evaluate and compare two machine learning models—Facebook's Prophet and XGBoost—for fine-grained demand forecasting at the store-item level. Accurate forecasts at this granular level allow businesses to manage inventory more efficiently, reduce operational costs, and respond quickly to market changes.

To conduct this analysis, a historical sales dataset from Kaggle's *Store Item Demand Forecasting Challenge* was used, featuring five years of daily sales data across 10 stores and 50 items. Prophet was chosen as the baseline model due to its ability to automatically detect seasonality and trends with minimal manual tuning, supported by foundational code from Databricks' *Fine-Grained Demand Forecasting Accelerator* (Databricks Solution Accelerator). XGBoost, a powerful gradient boosting algorithm, was selected as the second model for its proven accuracy in regression tasks and ability to incorporate custom features (Brownlee, 2021). By comparing these models, this study assesses how advanced machine learning techniques can enhance demand forecasting accuracy and improve overall supply chain efficiency.

2. Background and Related Work

My project falls under the prediction/classification category, specifically within the time-series forecasting topic. The task involves sequence prediction focused on overall accuracy and trend analysis.

My interest in this project stems from my background in supply chain management. Having earned an undergraduate degree in this field and completed several internships, I have seen how accurate forecasting can significantly impact business operations. My knowledge from previous experiences in analyzing forecasts was used in designing this experiment.

The dataset includes five years of daily sales data from 2013 to 2017, totaling 913,000 rows (Kaggle, 2018). It contains key variables such as date, store, item, and units purchased. Sales are broken down by the 10 different store locations and 50 different items, enabling a detailed analysis of demand trends over time. This structured dataset was chosen due to its comprehensiveness and potential for generating fine-grained demand forecasts at the store-item level.

Prophet serves as the baseline model due to its ease of use and built-in support for seasonality detection. It handles uneven sales patterns by modeling continuous time-series data without requiring manual feature engineering. Its ability to automatically detect weekly and yearly trends makes it easily implementable (Databricks Solution Accelerator).

I consulted ChatGPT to find a starting point in researching potential time series prediction models and came across the XGBoost model (see Appendix 1). Other models considered were LSTM, which was dismissed due to high computational requirements and complexity for large-scale forecasting tasks. Ultimately, XGBoost was chosen as the second model since it is highly customizable and able to efficiently handle data at granular levels. According to an article by Jason Brownlee, XGBoost consistently demonstrated strong performance in time-series forecasting challenges, as it is both accurate and fast (Brownlee, 2021).

A Kaggle tutorial submitted to the Kaggle Store Item Demand Forecasting Challenge by ENO5 was used as a reference for building the XGBoost model (Kaggle, 2018).

3. Methodology

Data Importation:

Since my dataset contained 913,000 rows, I leveraged Apache Spark for efficient data processing.

I first configured a Spark session with custom memory settings to optimize resource usage and ensure smooth data processing. Below are the configurations included:

```
spark = SparkSession.builder \
    .appName("Forecasting") \

.config("spark.executor.memory",
"8g") \  # Memory allocation for
parallel tasks

.config("spark.driver.memory",
"8g") \   # Memory for the
driver node

.config("spark.sql.shuffle.parti
tions", "100") \  # Optimize
shuffle operations
    .getOrCreate()
sc = spark.sparkContext
```

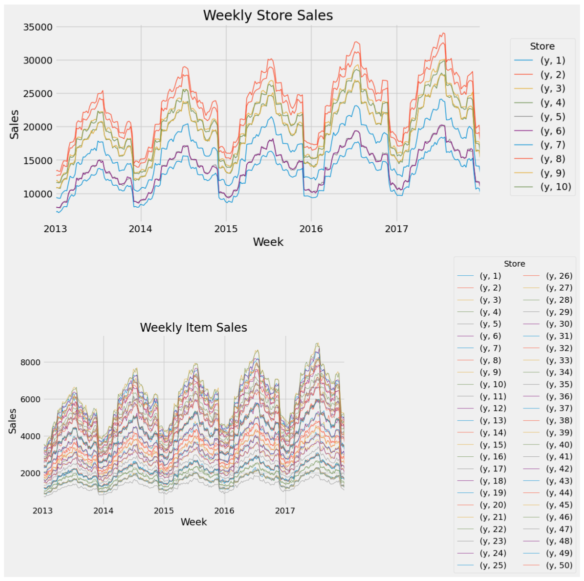
The sales data was imported using Spark's SQL interface, allowing for efficient query-based filtering and aggregation. I executed a SQL query to group sales records by store, item, and date, creating a structured Spark DataFrame for future analysis.

```
sql_statement = '''
  SELECT
    store,
    item,
    CAST(date as date) as ds,
    SUM(sales) as y
  FROM train
  GROUP BY store, item, ds
  ORDER BY store, item, ds
'''

store_item_history = (
  spark
    .sql(sql_statement)

.repartition(sc.defaultParalleli
sm, ['store', 'item'])  # Data
partitioning for parallel
processing
).cache()
```

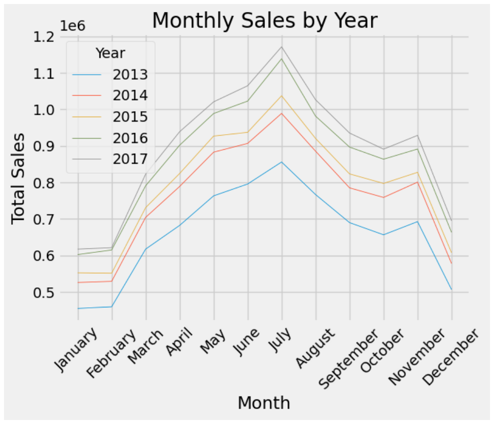
The data was partitioned by store and item to parallelize the data. Partitioning the data allowed Spark to distribute tasks efficiently, improving query execution times and reducing memory bottlenecks.

Data Exploration After importing the data, I converted the PySpark Data Frame to a Pandas Data Frame for visual analysis. Date columns were converted to datetime format, and additional columns such as year, month, and day were created for more granular time-based analysis.

I explored the dataset by generating line graphs to compare store sales and items over the years. This visual analysis revealed key sales patterns, including weekly, monthly, and yearly trends. Sales consistently increased year over year, with peak sales observed in July and November, and January with the lowest recorded sales.



These insights informed feature engineering decisions and model training configurations.



Model Building

Prophet Model

The Databricks solution accelerator's prebuilt code was followed to build the Prophet model. To install Prophet into my Google Collab notebook, the following command was used:

```
pip install prophet
```

Manual feature engineering is not necessary, making Prophet easily implementable. As mentioned earlier, trends are automatically fitted to the dataset. Below are the parameters set up for the model:

```
model = Prophet(
    interval_width=0.95,
    growth='linear',
    daily_seasonality=False,
    weekly_seasonality=True,
    yearly_seasonality=True,

seasonality_mode='multiplicative
'
    )
```

Prophet utilizes the whole dataset to train, allowing it to provide "predictions" for historical sales data, along with future forecasts. Model accuracy can be tested through back-testing with the new predictions and actual values. To ensure a consistent comparison with the XGBoost model, I filtered Prophet's forecast evaluation to include only 2017 sales data. The 2017 predictions and future 90-day forecast were saved to separate parquet files for future analysis (see Apendix 3 for ChatGPT prompt).

XGBoost Model

Before training the XGBoost model, features need to be created, as XGBoost is not inherently a time series model. I created features based on insights gained from initial data exploration to create relevant features that could capture temporal and historical sales patterns.

The features created include:

- **Lag Features:** Previous three days' sales (`lag_1`, `lag_2`, `lag_3`) to capture short-term sales patterns and temporal dependencies.
- **Rolling Averages:** A 3-day rolling average of past sales (`rolling_mean_3`) to smooth daily sales volatility and highlight longer-term trends.
- **Date-Based Features:**
    - **Year, Month, Day:** To capture broad seasonal patterns.
    - **Day of the Week:** To account for sales fluctuations driven by

customer behavior on different weekdays.
    - **Weekend:** weekends tend to have the highest sales

These features helped the XGBoost model incorporate both historical and time-based sales trends, supporting a more robust and accurate forecasting process.

A schema was then defined for our output results, including date, store, item, actual values, predicted values, and our training date.

To apply the XGBoost model at scale, I defined a Pandas UDF, called forecast_last_year, using PySpark's `GROUPED_MAP` functionality. This UDF allows each store-item combination to be processed independently (see Apendix 2 for ChatGPT prompt).

Within the function, a cutoff date is defined to split and train the dataset. Since we will be using an 80/20 split, it was determined that January 1, 2017 would be set as the cut off date. Once the data was split, X_train, y_train, and X_test data sets were defined with the features engineered earlier for training the XGBoost model. After our model was trained, it was then applied to our test set (X_test). This marks the end of our function.

Our function is applied to the original spark data frame store_item_history and saved to a Parquet file for future analysis.

4. Evalutation

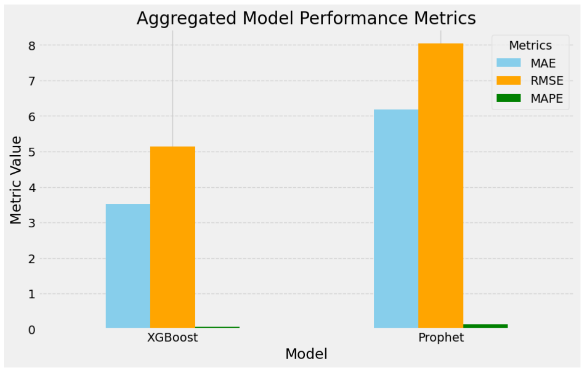
To ensure fairness in evaluating both models, data was limited to using 2017 predictions for analysis. The following metrics were used to measure forecast accuracy in both models (see Apendix 4 for ChatGPT prompt):

- **Mean Absolute Error (MAE):** measures the average absolute difference between predicted and actual values, equal weight given to all errors.
- **Root Mean Squared Error (RMSE):** RMSE measures variation in forecast errors and emphasizes larger variation from actual values by squaring the MSE
- **Mean Absolute Percentage Error (MAPE):** measures average percentage difference between predicted and actual sales.

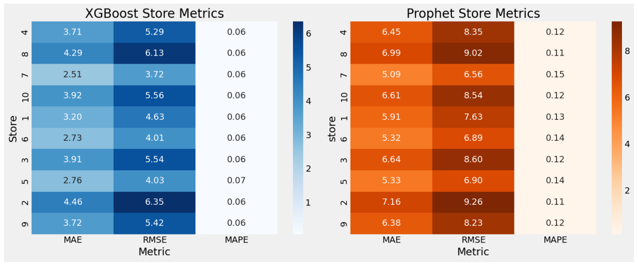
  Low scores for all three metrics indicate better accuracy. The overall MAE, RMSE, and MAPE were calculated for each model. The results are shown in the below table:

Model	MAE	RMSE	MAPE
Prophet	6.187	8.047	0.127
XGBoost	3.521	5.142	0.063

XGBoost had lower scores across the board, meaning its forecast accuracy is better.



MAE, RMSE, and MAPE metrics were also calculated for individual stores. Again, it is shown that XGBoost consistently has more accurate forecasts. A heat map of individual store metrics from both models is below.



To further test the accuracy of both models, a one sided t-test was performed using the individual store metrics of MAE, RMSE, and MAPE.

- null hypothesis: Prophet's performance is equal to or better than XGBoost's performance in terms of the selected evaluation metrics (MAE, RMSE, MAPE)
- alternative hypothesis: XGBoost performs significantly better than Prophet

I defined a function to run through each metric from both models using the spicy stats package (see Apendix 5 for ChatGPT prompt). The T-test calculates a t-statistic, which corresponds with a pvalue. If a p-value is less than 0.05, the difference in models is statistically significant. The results are listed below:

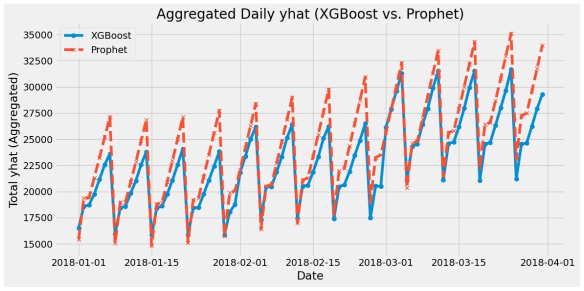
```
MAE T-Statistic: -134.7186
MAE P-Value: 1.7382e-16
Reject the null hypothesis for
MAE: XGBoost performs
significantly better.

RMSE T-Statistic: -104.8531
RMSE P-Value: 1.6564e-15
Reject the null hypothesis for
RMSE: XGBoost performs
significantly better.

MAPE T-Statistic: -15.4742
MAPE P-Value: 4.3032e-08
Reject the null hypothesis for
MAPE: XGBoost performs
significantly better.
```

Based on the t-test, all of the corresponding p-values are less than 0.05. We can conclude that our null hypothesis is not true and XGBoost significantly outperforms Prophet based on the MAE, RMSE, and MAPE values.

5. Results

After evaluating both models, I went ahead and used the XGBoost model we trained to predict a 90 day forecast. The last available sales date is identified, so a future date range can be created. Our forecast will start on January 1, 2018. The same features and function used to train our test XGBoost model are used again to forecast future predictions.

Once the XGBoost future forecasts are saved to a pandas data frame, the forecast from our Prophet model is converted from a spark data frame to a pandas data frame for comparison. With both datasets saved as pandas data frames, we can plot both forecasts. Below is a 90 day forecast with total sales forecast for all items and stores combined.



As shown above, both models generally follow the historical weekly and monthly sales pattern. Historically, sales spike during the weekend and gradually increase after January. We can see a spike in total sales towards the end of each week and overall, a gradual growth over the 90 days.

When comparing the two model's forecasts, the XGBoost model is smoother with less variability. The Prophet model is able to capture seasonality, but seems to over and under forecast the sales peaks and troughs.

6. Conclusion

Overall, XGBoost's gradient tree boosting capabilities allow users to create scalable and accurate granular level demand forecasting. With feature engineering the model allows precise predictions at the store-item level and is more accurate.

On the other hand, Prophet is a good option for making quick and scalable predictions. Its ability to automatically detect trends and seasonality means it requires less manual effort to create predictions.

In conclusion, when comparing our baseline solution from Databricks, the XGBoost model outperforms the Prophet in terms of forecast accuracy. Future research should be done on evaluating the baseline Prophet model after further improving the model. Other seasonality trends such as monthly or quarterly are some options available. Including holidays and other regressors would also improve accuracy of the baseline Prophet model.

Other models like ARIMA and time series forecasting tools can be explored. As technology advances, businesses will be able to create more accurate forecasts than ever before.

Appendix

1. ChatGPT Prompt: I am doing a project on fine grained demand forecasting. I am following a databricks solution that uses Prophet, what are some other alternatives to this forecasting model that are simple but still accurate?

2. ChatGPT Prompt: How can I ensure my xgboost model is forecasting for every store item combination?

3. ChatGPT Prompt: After I run my function to apply my model, what can I do to optimize processing speed for future analysis. Example code

4. ChatGPT Prompt: How do I calculate MAE, RMSE, and MAPE?

5. ChatGPT Prompt: How do I run a ttest with spicy package?

References

Brownlee, J. (2020, August 4). *How to Use XGBoost for Time Series Forecasting*. Machine Learning Mastery. https://machinelearningmastery.com/xgboost-for-time-series-forecasting/

ChatGPT. (2024). Chatgpt.com. https://chatgpt.com/auth/login

Chui, M., Manyika, J., Miremadi, M., Henke, N., Chung, R., Nel, P., & Malhotra, S. (2018, April 17). *Notes from the AI frontier: Applications and value of deep learning*. McKinsey & Company. https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-applications-and-value-of-deep-learning

Demand Forecasting at Scale. (n.d.). Databricks. https://www.databricks.com/solutions/accelerators/demand-forecasting

Eno5. (2018). *Blocked*. Kaggle.com. https://www.kaggle.com/code/enolac5/time-series-arima-dnn-xgboost-comparison#Model-(2)---Feed-Forward-Neural-Network-with-Daily-Data

Fine_Grained_Demand_Forecasting - Databricks. (2021). Databricks.com. https://notebooks.databricks.com/notebooks/RCG/Fine_Grained_Demand_Forecasting/index.html#Fine_Grained_Demand_Forecasting_1.html

Multiplicative Seasonality. (2023, October 18). Prophet. https://facebook.github.io/prophet/docs/multiplicative_seasonality.html

robikscube. (2018, November 9). *[Tutorial] Time Series forecasting with XGBoost*. Kaggle.com; Kaggle. https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost/notebook#Train/Test-Split

Store Item Demand Forecasting Challenge. (n.d.). Kaggle.com. https://www.kaggle.com/competitions/demand-forecasting-kernels-only/data