

Assignment 2

ML as a Service

Alexander Schou
Student ID: 25636299

04/10/2024

Github Username	hvf24hyv
Github Repos	Experiment Repo: https://github.com/hvf24hyv/AT2_experimentation.git Package Repo: https://github.com/hvf24hyv/AT2_experimentation.git API Repo: https://github.com/hvf24hyv/AT2_api.git Streamlit Repo: https://github.com/hvf24hyv/AT2_api.git
URLs	Backend: https://api-service-gmeq.onrender.com/ Frontend: https://streamlit-app-ba45.onrender.com/

36120 - Advanced Machine Learning Application
Master of Data Science and Innovation
University of Technology of Sydney



Table of Contents

1. Executive Summary	2
2. Business Understanding	3
a. Business Use Cases	3
b. Key Objectives	3
3. Data Understanding	4
4. Data Preparation	9
5. Modeling	10
a. Prophet	10
b. XGBoost	10
a. Approach 1 - Prophet	11
b. Approach 2 - XGBoost	11
6. Evaluation	13
a. Evaluation Metrics	13
b. Results and Analysis	13
c. Business Impact and Benefits	16
d. Data Privacy and Ethical Concerns	16
7. Deployment	18
8. Conclusion	19
9. References	20



1. Executive Summary

This project tries to accurately forecast and predict sales revenue using machine learning models to support inventory management and enhance decision-making for a retailer with multiple stores. By leveraging machine learning algorithms like XGBoost and Prophet, the project seeks to address the challenges of predicting sales patterns for many items across different stores and states and handling seasonal fluctuations and forecasting.

The report centers around the need for reliable forecasts to optimize stock levels, minimize costs, and improve customer satisfaction. The context of the project involves analyzing historical sales data across different states and product categories to identify trends and patterns.

The achieved outcomes include successfully deploying two models, with hyperparameter tuning for XGBoost and the integration of U.S. holidays in the Prophet model significantly improving forecasting and prediction. These results provide actionable insights that enable the retailer to make data-driven decisions, which hopefully can enhance operational efficiency and profitability.





2. Business Understanding

a. Business Use Cases

In the context of an American retailer with stores across California, Texas, and Wisconsin, this project aims to forecast total sales revenue and predict sales for specific items at individual stores on designated dates. By leveraging machine learning algorithms, we can enable the retailer to make informed decisions about inventory management, budgeting, and resource allocation.

Accurate revenue forecasting allows the retailer to understand future cash flow, creating better financial planning and budgeting. It also helps in determining optimal stock levels, as predicting sales for a specific date, item, and store can guide decisions on how much product to order. This reduces the risk of overstocking which ultimately can enhance customer satisfaction..

b. Key Objectives

The project's primary objectives are to develop two machine learning models for the retailer: one for forecasting total sales revenue across all stores and items for the next seven days and another for predicting sales revenue for specific items at designated stores on particular dates. These models aim to improve decision-making, enhance inventory management, and optimize budget allocation.

Key stakeholders include the retailer's management team, store managers, and inventory planners. The management team seeks insights into overall sales trends for strategic planning, while store managers need accurate predictions for specific items to optimize inventory levels and reduce excess stock. Overall, the web application deployed is meant to be used internally in the company.



3. Data Understanding

The dataset for this project provides detailed sales data for a retailer with 10 stores across California, Texas, and Wisconsin. Each store sells items across three major categories: hobbies, foods, and household. The data captures sales for individual items at specific stores, along with additional information like item prices, store locations, and dates.

We were provided with five datasets for this project:

- Training data: Contains historical sales data for training machine learning models where we are given “id”, “item_id”, “dept_id”, “cat_id”, “store_id”, “state_id” and the sales for every single date.
- Evaluation data: Used for assessing the performance of the forecasting model with sales and date.
- Calendar: Provides a mapping of dates to days and weeks where we are given “date”, “wm_yr_wk” and “d”.
- Events: Lists special events (e.g., holidays) that might impact sales trends where we are given “date”, “event_name” and “event_type”.
- Items price per week: Contains weekly pricing information for items, essential for revenue calculation where we are given “store_id”, “item_id”, “wm_yr_wk” and “sell_price”.

On average, items sell 1.09 units per day, but there is high variability (standard deviation of 3.93). The majority of days report 0 sales, as shown by the median and 25th percentile both at 0. However, the maximum number of units sold in a single day is 763, indicating occasional high-volume sales.

The average price per item is \$4.38, with significant price variation (standard deviation of \$3.36). Prices range from \$0.01 to \$107.32, with the majority of items priced between \$2.08 (25th percentile) and \$5.84 (75th percentile). The revenue generated mirrors this variability, with a mean revenue of \$3.03 and a maximum of \$2,164.32, highlighting the potential impact of high-volume sales on overall revenue.

Figure 1 shows sales across different states. In general, most days exhibit similar sales figures across California, Texas, and Wisconsin, indicating a baseline level of demand for products. However, California stands out with certain days showcasing significantly higher sales volumes than the other states. In contrast, Wisconsin has the lowest maximum sales figures for any product on a given day, suggesting a different market landscape.

The observed differences in sales patterns across states highlight the importance of incorporating the state feature into a machine learning model. Given that California shows higher sales on certain days compared to Texas and Wisconsin, and that Wisconsin consistently records the lowest maximum sales, this feature can provide valuable context for predicting sales.

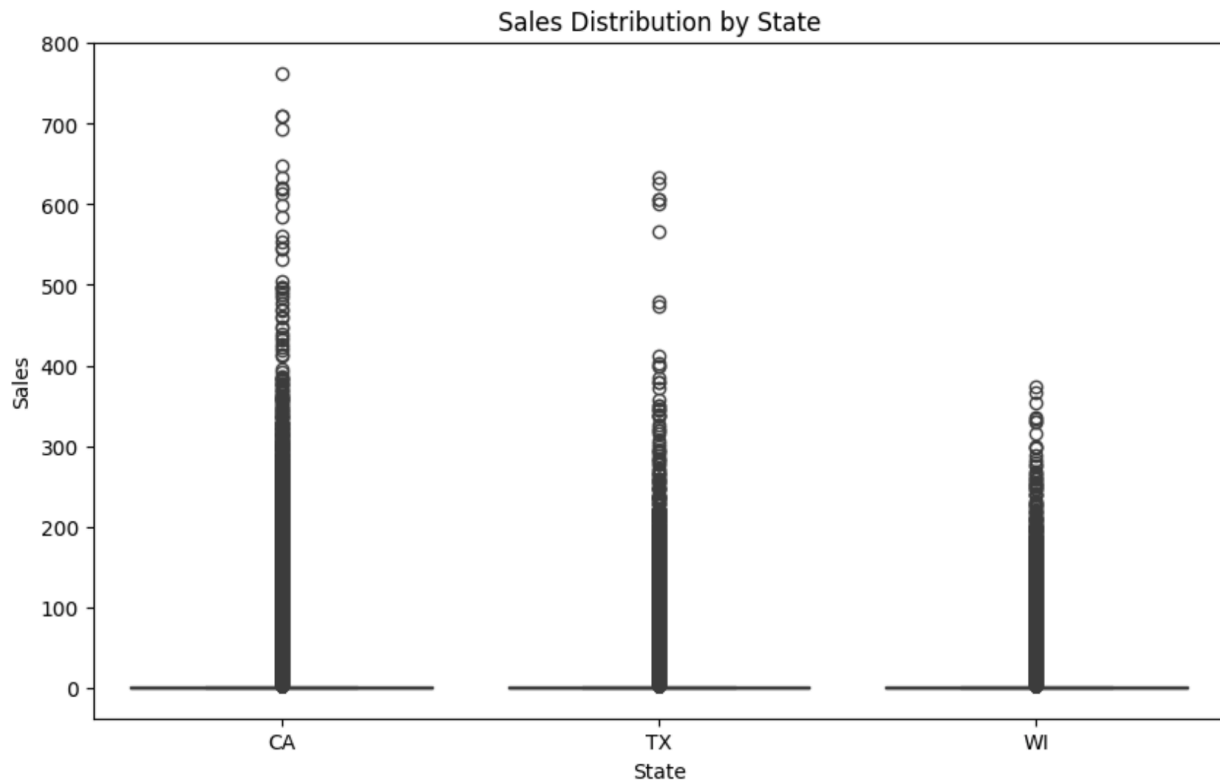


Figure 1: Sales Distribution across States

Figure 2 shows the total sales over time. The plotted data reveals several days with exceptionally high or low sales figures, particularly on December 25th, which exhibits notably low sales. This decline on Christmas Day is probably caused by the fact that people are at home celebrating and socializing.

Additionally, the data shows clear seasonality, with sales peaking in July and reaching a low at the end of the year. This seasonal trend indicates that summer months drive higher consumer engagement, while sales tend to decline as the year draws to a close.

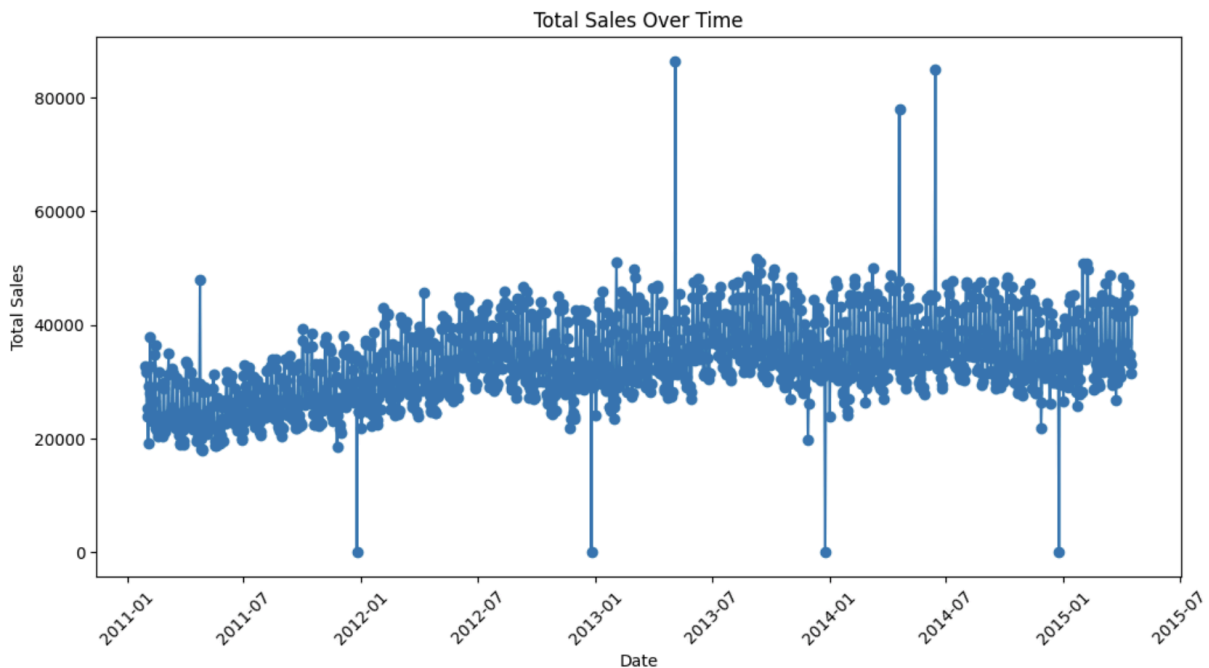


Figure 2: Total Sales Over Time

Moreover, a subtle upward trend in sales can be observed over the entire period, indicating an overall increase in consumer spending. This suggests that while seasonality influences sales fluctuations, the market is gradually expanding, reflecting positive growth. The upward trend can also be seen when we look at monthly revenue in figure 3.

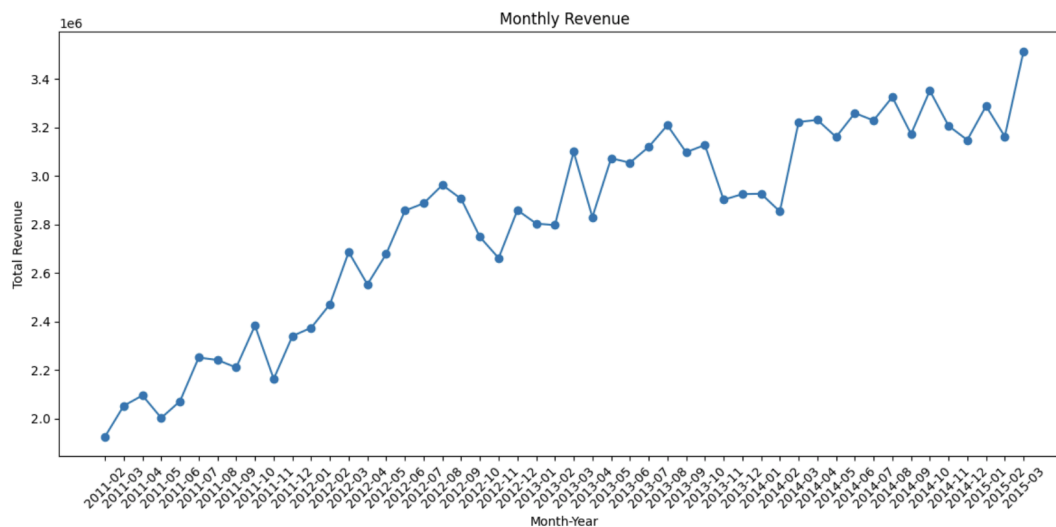


Figure 3: Monthly Revenue

Figure 4 reveals that food items consistently achieve higher maximum sales compared to household and hobby products, indicating stronger consumer demand for food-related purchases. Among the three categories, hobbies exhibit the lowest maximum sales figures, suggesting that consumers are less likely to engage in hobby-related spending compared to food and household items. This trend underscores the importance of adding the food category to our machine learning model. We also see differences in sales when looking at the different stores. A difference in sales across stores can also be seen in figure 5.

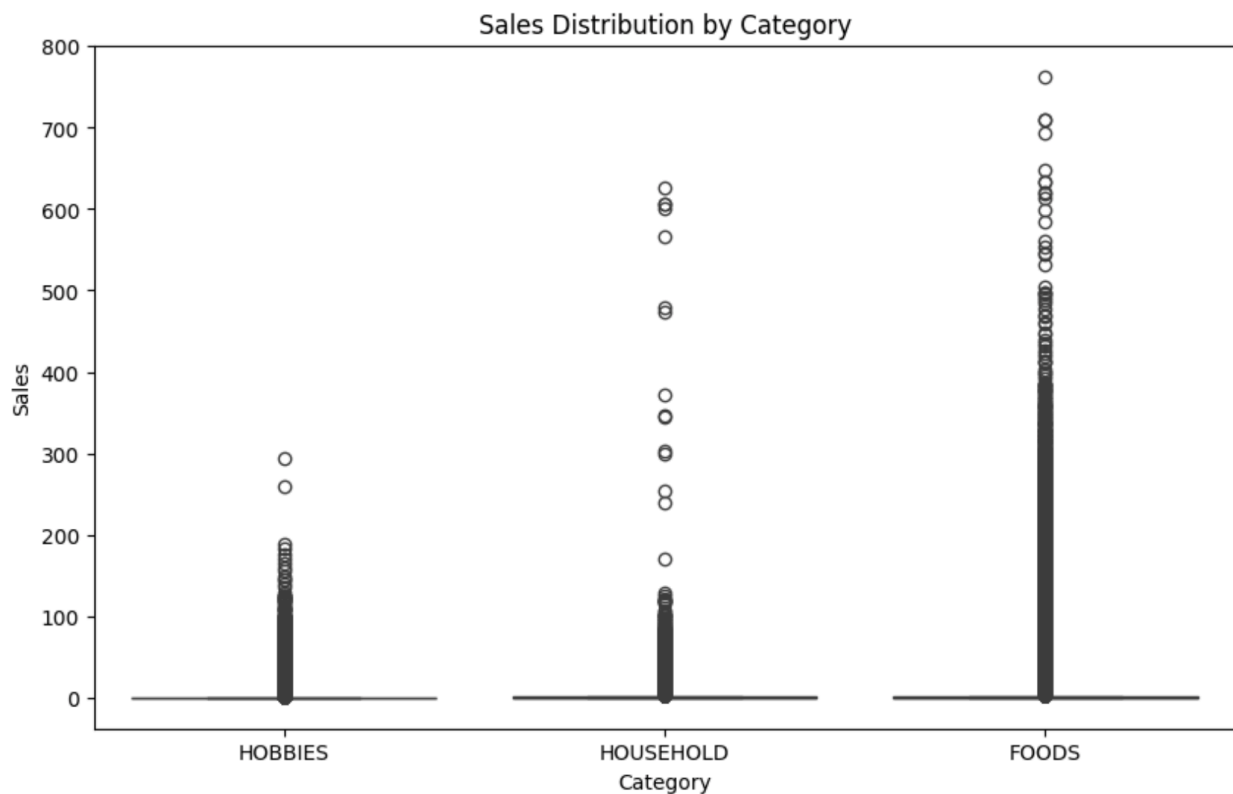


Figure 4: Sales Distribution across Item Category

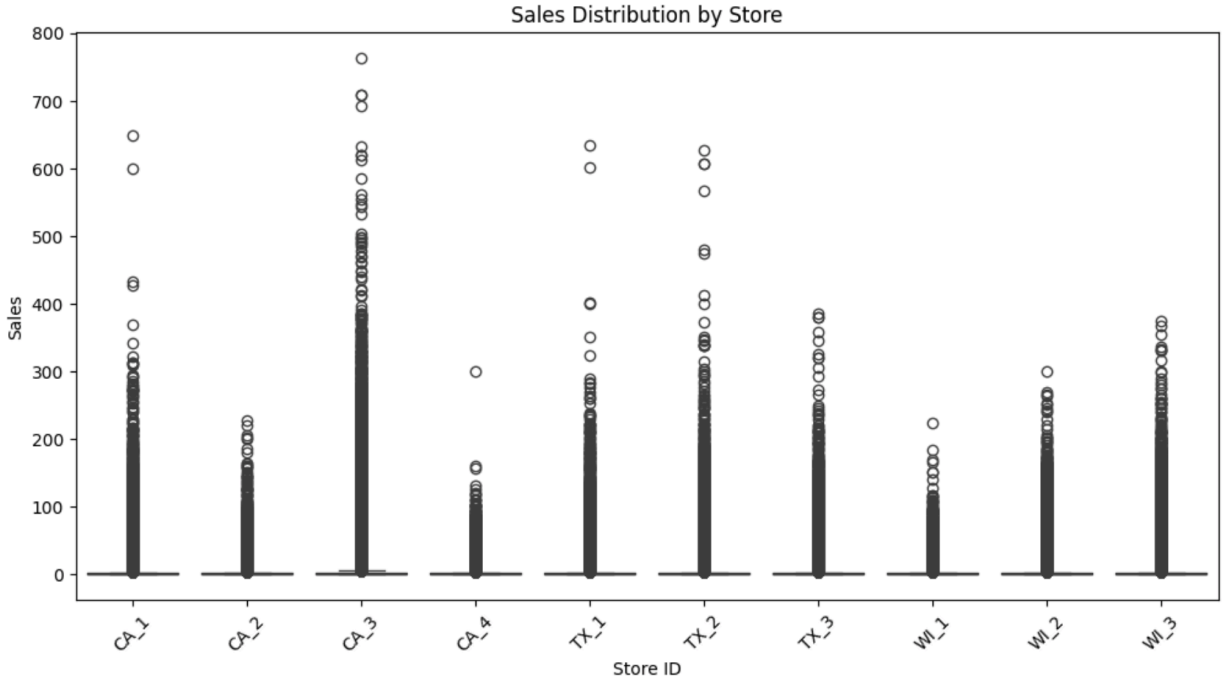


Figure 5: Sales Distribution across Stores



4. Data Preparation

In preparing the data for modeling, several specific steps were taken to ensure its quality and readiness. The dataset was first cleaned by removing duplicate entries and irrelevant columns that were not used in the analysis. We examined the dataset for missing values, particularly in the event-related columns. Since there were no significant events on certain dates, we created a new category called "None" for these columns to maintain consistency.

Feature engineering techniques were applied to enhance the dataset. New features were created, including extracting the month and year from the date column to capture temporal patterns in sales. A new revenue column was also created.





5. Modeling

a. Prophet

In this project, we implemented the Prophet algorithm for time-series forecasting, chosen for its robustness in handling seasonal data and its ability to incorporate holidays, making it suitable for retail sales prediction. The rationale behind selecting Prophet was its user-friendly nature and effectiveness and computational speed (Anthony, 2024, S2).

The modeling process began with a simple implementation of Prophet, where we first focused on the baseline performance. We then enhanced the model by adding U.S. holidays that could influence sales patterns. Following this, we explored hyperparameter tuning to optimize the model's performance further. The tuning process involved adjusting several key hyperparameters, such as `changepoint_prior_scale`, `seasonality_mode`, and `holidays_prior_scale`, to identify the best combination that minimized the Mean Absolute Error (MAE) on the validation set.


Additionally, we experimented with adding custom holiday events based on the event data. However, this did not yield improved performance compared to the standard holiday settings. The final step in the modeling phase involved evaluating each configuration, plotting the forecasted versus actual sales, and selecting the best-performing model based on MAE.

b. XGBoost

The algorithm used in this project for prediction is XGBoost (Extreme Gradient Boosting), a powerful and efficient implementation of gradient-boosted decision trees. The rationale for choosing XGBoost stems from its ability to handle complex datasets while offering excellent predictive accuracy and scalability. XGBoost is particularly well-suited for regression tasks like predicting sales revenue, as it can capture non-linear relationships and interactions between features (Anthony, 2024, S5).

XGBoost was selected due to its ability to handle large-scale datasets, its built-in regularization to avoid overfitting, and its flexibility in hyperparameter tuning. Additionally, it supports parallel processing, which makes it faster to train compared to some other models (Anthony, 2024, S5).

Hyperparameter tuning was performed using Hyperopt, a library designed for optimizing hyperparameters efficiently. Hyperopt applies the Tree-structured Parzen Estimator (TPE) approach, which allows for a more efficient search of the hyperparameter space compared to traditional grid search methods (Anthony, 2024, S5).



In this case, a subset of the data (10% of the full dataset) was used for tuning to reduce computation time.

After 20 evaluations, the best combination of hyperparameters was found and used to train the final model on the entire dataset. The evaluation metrics used to select the best model were Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2).

a. Approach 1 - Prophet

During the preprocessing stage, we prepared the training and test datasets by converting the date columns to datetime format and aggregating daily sales data to create a consistent structure for the Prophet model, renaming the columns to the expected format (ds for dates and y for values). The feature engineering process included adding U.S. holidays to the model to capture their impact on sales trends.

The training process involved fitting the model to the daily sales data, and during evaluation, we closely monitored MAE, RMSE, and MAPE metrics to assess the accuracy of the predictions.

The best parameters identified for the model were:

- changepoint_prior_scale: 0.001
- seasonality_mode: 'multiplicative'
- holidays_prior_scale: 1.0
- fourier_order: 7.


The results from various configurations were plotted for visual comparison, allowing us to select the best-performing model effectively.

b. Approach 2 - XGBoost

The first model implemented was an XGBoost Regressor, integrated into a pipeline that handled preprocessing tasks like imputation and one-hot encoding for categorical features (item_id and store_id). Missing categorical values were filled using a SimpleImputer, and one-hot encoding was applied to transform these features into numerical format. After doing a simple XGBoost the hyperparameters were in the end tuned by using Hyperopt.

The best parameters identified for the model were:

- n_estimators: 100.
- max_depth: 4.
- learning_rate: 0.2491.
- subsample: 0.6634.

- 
- colsample_bytree: 0.7304.



6. Evaluation

a. Evaluation Metrics

To evaluate the Prophet forecasting model, we used three metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). These were chosen for their ability to highlight different aspects of model performance. MAE offers a straightforward average of errors, making it easy to interpret and serving as the primary metric for selecting the final model. RMSE emphasizes larger errors, which is crucial for identifying significant forecasting mistakes on high-sales days. MAPE presents errors as a percentage.

For the XGBoost model, we also employed MAE, RMSE, and R-squared (R2). MAE was again the primary metric for finalizing the model, reflecting the project's goal of minimizing prediction error. RMSE penalizes larger errors, highlighting critical mistakes in sales predictions, while R2 indicates how well the model explains data variance, providing insights into overall fit.

b. Results and Analysis

For the Prophet forecasting model we started with a simple Prophet model, then enhanced it by adding U.S. holidays and finally tuned the model. The model evaluation can be seen in table 1.

Model	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)	Mean Absolute Percentage Error (MAPE)
Simple Prophet	10201.26	14051.97	908.06%
Prophet with U.S. Holidays	9527.64	12524.60	204.86%
Prophet with U.S. Holidays & Tuning	7280.60	11100.12	836.41%

Table 1: Model Performance

The introduction of U.S. holidays improved the model's performance significantly. The MAE dropped from 10201.26 to 9527.64 with U.S. holidays, and further to 7280.60 after tuning. This decrease in MAE reflects improved accuracy, while the RMSE shows that the model reduced its larger errors. Notably, the MAPE decreased substantially after adding holidays but increased after tuning.

Figure 6 and 7 reveal insights regarding performance with respect to holidays. The model using only U.S. holidays captures the sales dip on December 25th effectively, demonstrating its strength in accounting for holiday effects. However, it struggles with the accuracy of the remaining data points, leading to poorer overall performance. In contrast, the tuned model that incorporates both U.S. holidays and hyperparameter adjustments provides a more consistent fit across the dataset. While it slightly underestimates the December 25th dip, it excels in predicting sales trends during non-holiday periods. This highlights the trade-off between holiday-specific accuracy and overall model reliability. Hence, the evaluation metric used for choosing a final model has a huge impact on which model to choose. Future investigation could analyze an ensemble model.

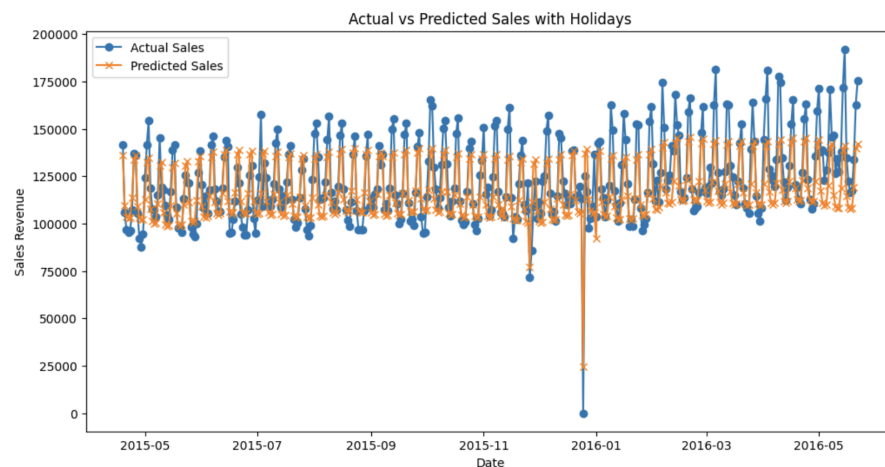


Figure 6: Prediction for Prophet model with US holidays but no tuning

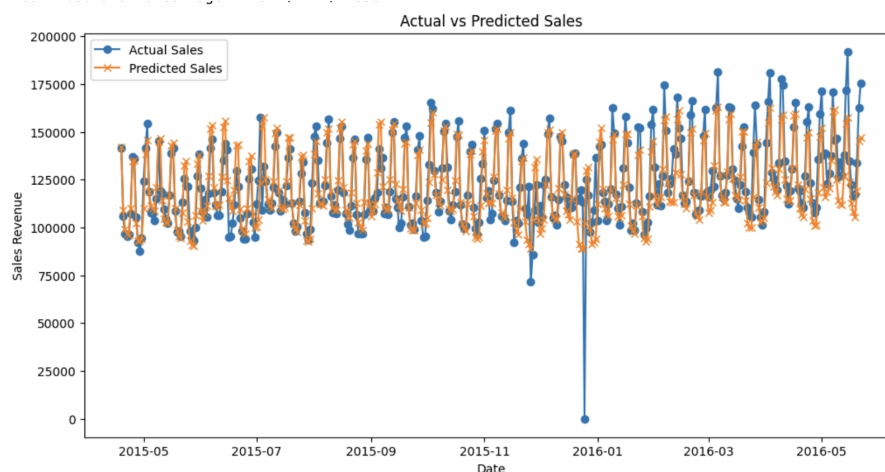


Figure 7: Prediction for Prophet model with US holidays and tuning

For the XGBoost model, we evaluated its performance using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2). Initially, a simple XGBoost model without hyperparameter tuning was trained, yielding the following results:

- MAE: 3.84
- RMSE: 7.66
- R-squared: 0.30

After applying hyperparameter tuning with Hyperopt, the model's performance improved:

- MAE: 3.71
- RMSE: 7.50
- R-squared: 0.33

The tuned XGBoost model showed improvements across all metrics, particularly in MAE, which was reduced from 3.84 to 3.71, indicating better overall prediction accuracy. The RMSE also decreased slightly, suggesting that the model better handled larger prediction errors. The R-squared value, while still modest, increased from 0.30 to 0.33, showing a slight improvement in the model's ability to explain the variance in the data.

Figure 8 shows how the model's loss (MAE) changes over iterations, highlighting the effectiveness of tuning in reducing errors. Figure 9 allows us to see how different hyperparameter combinations were explored giving a clearer understanding of which parameter values contributed to performance improvements.

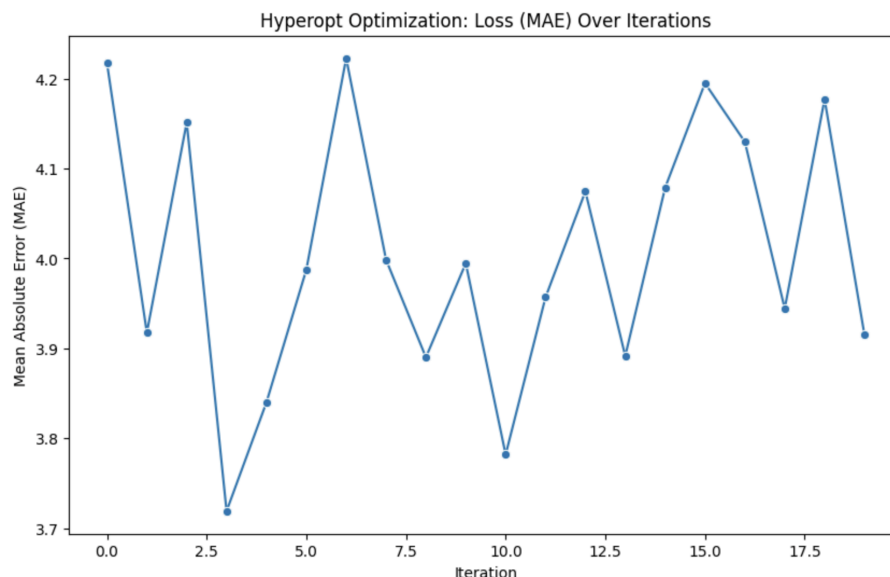


Figure 8: Model Loss versus Iterations

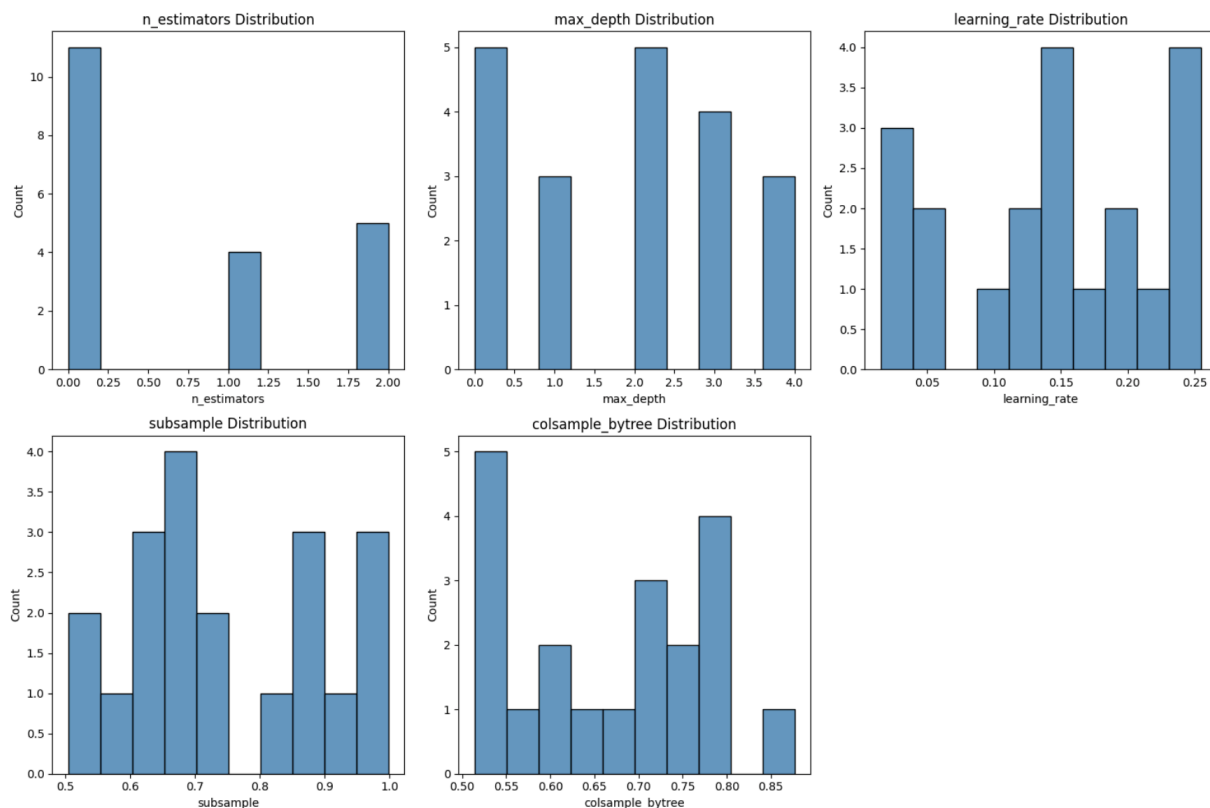



Figure 9: Hyperparameter Distributions from Optimization Trials

c. Business Impact and Benefits

The final models could help the retailer by providing sales forecasts, which are crucial for effective inventory management and demand planning. By addressing the challenge of unpredictable sales patterns, the model enables the company to optimize stock levels, reducing both excess inventory and stockouts. The improvements achieved include a reduction in Mean Absolute Error from previous models, leading to more reliable predictions that help align supply with customer demand. This capability not only minimizes operational costs but also maximizes sales opportunities during peak periods. However, further improvements should be explored to get even better predictions.

d. Data Privacy and Ethical Concerns

The project primarily involves sales data that does not contain personally identifiable information, which mitigates many data privacy concerns. However, ethical considerations still arise regarding the collection and usage of business-sensitive data, such as sales patterns and inventory



information. It is essential to ensure that the data is collected transparently and used solely for the intended analytical purposes, without any risk of exploitation.





7. Deployment

The deployment of the trained model involved setting up both an API and a Streamlit application using Render's free tier. The API, built with FastAPI, enables users to make sales predictions via two main endpoints. The `/sales/national/` endpoint provides a total sales forecast for the next 7 days across all stores, requiring a start date (in YYYY-MM-DD format) as input, and returning forecasted sales in JSON format. The `/sales/stores/items/` endpoint predicts sales for a specific item at a particular store on a given date, requiring the date, store ID, and item ID as input, and returning the predicted sales.

For national sales forecasts, send a GET request to `/sales/national/` with the desired date. For item-specific predictions, send a GET request to `/sales/stores/items/` with the date, store ID, and item ID. For example:

National sales forecast:

```
GET /sales/national/?date=2024-01-01
```

Store item prediction:

```
GET /sales/stores/items/?date=2024-01-01&store_id=CA_1&item_id=HOBBIES_1_001
```

To use the Streamlit app for running sales predictions, start by accessing the app via the provided URL on Render. For the National Sales Forecast, select a start date in the "7-Day Sales Forecast" section and click the "Get Forecast" button. This will display the predicted total sales for the next seven days, shown in both a table and an interactive line chart.

For the Item Sales Prediction, enter the store ID and item ID, then choose the date you want to predict sales for. After entering the required information, click "Get Item Sales Prediction" to see the predicted sales for that specific item at the selected store. To check the status of the API, click the "Check API Health" button, which will return the current status and a welcome message from the API. This allows you to ensure the API is functioning properly before making predictions.





8. Conclusion

The project successfully developed and deployed a forecasting model that predicts sales revenue, providing valuable insights for inventory management and demand planning. Key findings included the effectiveness of the XGBoost and Prophet models, with hyperparameter tuning significantly improving the performance of the XGBoost model and the inclusion of U.S. holidays enhancing the accuracy of the Prophet model.

The project met its primary goals of delivering sales forecasts. Stakeholders can leverage these forecasts to optimize stock levels and enhance decision-making processes.

Looking ahead, future work could involve integrating additional data sources, such as promotional events or competitor pricing, to further enhance prediction accuracy. Recommendations include exploring more advanced modeling techniques and refining the deployment setup to improve performance on the free Render tier. Next steps should focus on monitoring the model's performance in real-world scenarios and iteratively improving it based on feedback and new data.





9. References

Anthony, S. (2024). Session 2 [Subject 36120 Lecture Notes]. UTS Canvas.
<https://canvas.uts.edu.au>

Anthony, S. (2024). Session 5 [Subject 36120 Lecture Notes]. UTS Canvas.
<https://canvas.uts.edu.au>

